UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

Rosalia Edith Caya Carhuanina

Adaptatividade - Visão holística e ferramenta de projeto

São Paulo

2023

Rosalia Edith Caya Carhuanina

Adaptatividade - Visão holística e ferramenta de projeto

Tese apresentada à Escola Politécnica da Universidade de São Paulo, para a obtenção do título de Doutora em Ciências.

São Paulo

2023

Rosalia Edith Caya Carhuanina

Adaptatividade - Visão holística e ferramenta de projeto

Versão corrigida
(Versão original disponível na Biblioteca da EPUSP)

Tese apresentada à Escola Politécnica da Universidade de São Paulo, para a obtenção do título de Doutora em Ciências.

Área de concentração: Engenharia de Computação

Orientador: Prof. Dr. João José Neto

São Paulo
2023

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Caya Carhuanina, Rosalia Edith
Adaptatividade - Visão holística e ferramenta de projeto / R. E. Caya Carhuanina -- versão corr. -- São Paulo, 2023.
294 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Adaptatividade 2.Meta-modelo 3.Holístico 4.Projeto de software 5.Sistema Legado I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

CAYA CARHUANINA, R. E. **Adaptatividade - Visão holística e ferramenta de projeto**. 2023. Tese (Doutorado em Ciências) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

AGRADECIMENTOS

Obrigada Deus pela oportunidade, pelos seus planos para mim, pela luz ao longo do caminho, pelos ensinamentos, pelo suporte. Obrigada por me amar mais do que eu posso merecer. Esta conquista é sua.

À família que Deus me deu, Héctor e Ysabel, porque com o exemplo me ensinaram a sonhar, me herdaram o seu 20, me deram a segurança do amor incondicional e me deixaram escolher o meu caminho. Sem vocês eu não seria o que sou. Obrigada pela sua força, por me ensinar a acreditar em mim mesma, ser positiva e ter fé. Obrigada pela sua luz, eu sei que o amor é tão grande que nem o céu se opõe e deixa o meu pai ser meu anjo da grada para ficar sempre do meu lado.

À família que a vida me apresentou ao decorrer dos anos. Àqueles professores que se converteram em amigos, pela sua entrega e motivação, exigências e ajuda para descobrir o meu caminho. Obrigada pelo tempo, fez toda a diferença. Aos amigos que se converteram em família. No Peru, às meninas da escola, ao meu Manis, e aos meus amigos entranháveis da PUCP, porque sempre estiveram, pelo seu tempo e companhia. O seu apoio nos momentos de cansaço, fraqueza e perda de foco me ajudou a recuperar o Norte e me sentir sempre suportada por uma torcida incrível. No Brasil, ao pessoal do LTA, por me acolher e me fazer sentir parte desse maravilhoso grupo. Pelo seu tempo, pelas conversas e risadas, pelo consolo e ajuda nos momentos mais terríveis. Que essa família nunca pare de crescer. À Francisca e Dilma, pelo seu carinho, cuidado e amizade. Sem o seu apoio e acolhida o meu caminho teria sido outro, muito mais difícil e muito mais vazio. Obrigada por ter me deixado ser a sua sobrinha adotiva. Ao João, pela sua confiança em mim desde o começo, pelo seu calor humano, pela sua ajuda e conselhos em todos os âmbitos possíveis. Obrigada João pela paciência, pelo português, pela eterna disposição para responder às inumeráveis preguntas, especialmente aquelas previas ao final de semana. Obrigada também por ser não só um excelente orientador acadêmico, mas também um mentor de vida e me adotar como parte de sua família com os braços e o coração abertos.

Finalmente, muito obrigada ao meu amado esposo Adam, por ser o meu melhor amigo, meu boost, meu exemplo de dedicação e esforço. Pelos cappuccinos, chás, cookies e livros que animavam nossos dias de estudo. Obrigada pelos abraços que reiniciam, por compartilhar tanto o peso quanto as alegrias, por aceitar os desafios e comemorar as conquistas. Sem sua companhia, compreensão, força e ânimo incansável, tudo seria diferente. Obrigada Vida! Por ter mais fé em mim da que eu mesma tinha e me lembrar quem eu sou nos dias mais difíceis.

Muito obrigada, família! Sem vocês teria sido impossível, o seu ânimo e impulso recarregou as baterias, preparou o café, acendeu a luz e me ajudou a voar.

Gracias Dios por la oportunidade, por tus planes para mi, por la luz a lo largo del camino, por las enzeñanzas, por el soporte. Gracias por amarme más de lo que puedo merecer. Esta conquista es tuya.

A la familia que Dios me dio, mis papás Hector e Ysabel, porque con el ejemplo me enseñaron a soñar, me heredaron su 20, me dieron la seguridad del amor incondicional y me dejaron escoger mi camino. Sin ustedes no sería lo que soy, gracias por su fuerza, por enseñarme a creer en mí, a ser positiva y a tener fe. Gracias por su luz, sé que el amor es tan grande que ni el mismo cielo se opone y deja que papá sea mi ángel de la guarda para estar siempre conmigo.

A la familia que la vida me presentó a lo largo de los años. A aquellos profesores que se convirtieron en amigos, por su entrega y motivación, exigencias y ayuda para descubrir mi camino. Gracias por el tiempo, hizo toda la diferencia. A los amigos que se convirtieron en familia. En Perú, a las Chicas del Cole, a mi Manis, y a mis amigos entrañables de la PUCP, porque siempre estuvieron, por su tiempo y compañía. Su apoyo en los momentos de cansancio, debilidad y pérdida de foco me ayudó a recuperar el norte y sentirme siempre respaldada por una barra increíble. En Brasil, a los chicos del LTA, por acogerme y hacerme sentir parte de ese maravilloso grupo. Por su tiempo, las conversaciones y risas, el consuelo y la ayuda en los momentos más terribles. Que esa familia nunca pare de crecer. A Francisca y Dilma, por su cariño, cuidado y amistad. Sin su apoyo y acogida, mi camino hubiera sido otro, mucho más difícil, mucho más vacío. Gracias por dejarme ser su sobrina adoptiva. A João, por su confianza en mi desde el inicio, por su calor humano, por su ayuda y consejos en todos los ámbitos posibles. Gracias João por la paciencia, por el portugués, por la eterna disposición para responder las innumerables preguntas, especialmente las previas al final de semana. Gracias también por ser no sólo un excelente asesor académico, sino también un mentor de vida y adoptarme como parte de su familia con los brazos y el corazón abierto.

Finalmente, gracias a mi amado esposo Adam, por ser mi mejor amigo, mi boost, mi ejemplo de dedicación y esfuerzo. Por los cappuccinos, tés, cookies y libros que animaban nuestros días de estudio. Gracias por esos abrazos que reinician, por compartir el peso y las alegrías, aceptar los desafíos y celebrar las conquistas. Sin su compañía, comprensión, fuerza y ánimo incansable, todo sería diferente. Gracias Vida! por tener más fe en mí de la que yo misma tenía y recordarme quien soy los días más difíciles.

Muchas gracias, familia! Sin ustedes habría sido imposible, sua aliento y empuje recargó las baterías, preparó el café, encendió la luz y me enseñó a volar.

Ad Maiora!

For them, for the team, for the tribe, for the magic of life.

Amen.

"Todo está en la mente."
HacyMa

"Study while others are sleeping;
work while others are loafing;
prepare while others are playing;
and dream while others are wishing."
William A. Ward

RESUMO

CAYA, R. **Adaptatividade - Visão holística e ferramenta de projeto**. 2023. 292p. Tese (Doutorado) - Escola Politénica, Universidade de São Paulo, São Paulo, 2023.

Nos últimos 50 anos, a sociedade humana tem experimentado uma integração massiva e acelerada com tecnologias, transformando sua própria dinâmica e formando uma coevolução entre tecnologias, ambientes e usuários: o potencial tecnológico permite a expansão das atividades humanas, que a sua vez geram novos desafios tecnológicos. Nesse cenário, os sistemas computacionais precisam ter um bom desempenho em contextos com complexidade crescente, mudança constante e incerteza inerente para se ajustar de maneira confiável às necessidades dos usuários. Nos últimos anos, a comunidade científica tem identificado a incorporação de adaptatividade nos sistemas computacionais como uma possível resposta para esse desafio. Porém,a falta de uma teoria geral e unificada de adaptatividade tem como consequência o desenvolvimento de soluções segregadas que raramente consideram os sistemas legados estabelecendo uma barreira para seu uso em sistemas convencionais. Este trabalho propõe uma solução em dois níveis: uma proposta de marco conceitual e um framework de suporte para a implementação de sistemas adaptativos sob essa nova visão. No nível conceitual, é elaborado o meta-modelo holístico como uma nova abordagem para o projeto e implementação de sistemas adaptativos sob uma visão agnóstica de abordagem e com uma estratégia integradora. O meta-modelo holístico apresenta uma definição clara, simples e geral da adaptatividade que procura servir como guarda-chuva para outras abordagens em diversas áreas do conhecimento. No nível tecnológico o meta-modelo holístico é complementado com um framework livre que assiste o projeto e implementação de mecanismos adaptativos com base em sistemas adaptáveis, assim, os sistemas legados são integrados de maneira explícita no processo de desenvolvimento de sistemas adaptativos sempre que satisfaçam as condições mínimas para sua transformação. Ambos os níveis foram avaliados com suítes de testes correspondentes para verificar sua flexibilidade. Dessa maneira, mostramos que o meta-modelo holístico permite a integração de diversas abordagens facilitando o intercâmbio de conhecimento entre diferentes áreas, colaborando como um passo mais na direção do desenvolvimento de uma teoria geral de adaptatividade. Adicionalmente, o framework desenvolvido, é uma das poucas ferramentas disponíveis para pesquisadores que permite enriquecer sistemas adaptáveis com mecanismos adaptativos, se afirmando como uma abordagem de desenvolvimento de sistemas computacionais de longa vida. Finalmente, o trabalho expõe as conclusões derivadas do seu desenvolvimento, e o grande potencial de pesquisa tanto na área de tecnologias adaptativas quanto no desenvolvimento de melhoras da proposta aqui apresentada.

Palavras-chave: Adaptatividade. Meta-modelo. Holístico. Projeto de Software. Sistema Legado

ABSTRACT

CAYA, R. **Adaptivity - Holistic vision and design tool**. 2023. 292p. Thesis (Doctor) - Escola Politénica, Universidade de São Paulo, São Paulo, 2023.

The last 50 years, human society has experienced a massive and accelerated integration with technologies, transforming its own dynamics and forming a co-evolution between technologies, environments, and users: the technological potential allows the expansion of human activities, which in turn generates new technological challenges. In this scenario, computer systems need to perform well in contexts with increasing complexity, constant change, and inherent uncertainty to reliably adjust to user needs. In recent years, the scientific community has identified the incorporation of adaptivity in computer systems as a possible answer to this challenge. However, the lack of a general and unified theory of adaptivity results in the development of segregated solutions that seldom consider legacy systems establishing a barrier to their use in conventional systems. This work proposes a two-level solution: a proposal for a conceptual framework and a support framework for the implementation of adaptive systems under this new vision. At the conceptual level, the holistic meta-model is elaborated as a new approach to the design and implementation of systems with adaptive behavior under an agnostic approach and with an integrative strategy. The holistic meta-model presents a clear, simple, and general definition of adaptivity that seeks to serve as an umbrella for other approaches in different areas of knowledge. At the technological level, the holistic meta-model is complemented with a free and adaptable framework that assists the design and implementation of adaptive mechanisms based on adaptable systems. Thus, legacy systems are explicitly incorporated into the process of developing adaptive systems whenever they satisfy the minimum conditions for their transformation. Both levels were evaluated with corresponding test suites to verify their flexibility. This way, we show that the holistic meta-model allows the integration of different approaches, facilitating the exchange of knowledge between different areas, and collaborating as a further step towards the development of a General Theory of Adaptivity. Additionally, the developed framework is one of the few tools available to researchers that allows the enrichment of adaptable systems with adaptive mechanisms, asserting itself as an approach to the development of long-lived computing systems. Finally, the work exposes the conclusions derived from its development, and the great research potential both in the area of Adaptive Technologies and in the development of improvements to the proposal presented here.

Keywords: Adaptivity. Holistic. Meta-model. Software Design. Legacy system.

LISTA DE FIGURAS

Figura 1 – Evolução de trabalhos científicos relacionados com adaptatividade em Web of Science e Scopus. 27
Figura 2 – Publicações com referência direta à adaptatividade por Área de Pesquisa em Web of Science e Scopus. 28
Figura 3 – Contribuições principais da tese. 35
Figura 4 – Mapa de leitura da tese. 37
Figura 5 – Representação dos diferentes estados e terminologias envolvidas no processo adaptativo. 41
Figura 6 – Alguns exemplos de comportamento adaptativo na natureza. 43
Figura 7 – Mapa conceitual dos termos relacionados ao *comportamento adaptativo*. 45
Figura 8 – Espaço de estados do sistema. 48
Figura 9 – Economias relacionadas à industrialização: (a) Economia de escala, (b) Economia de escopo. 50
Figura 10 – Visão geral de uma fábrica de software. 52
Figura 11 – Diferentes visões da realidade a modelar. 54
Figura 12 – Relação entre modelo e meta-modelo. 55
Figura 13 – Arquitetura de quatro-camadas da meta-modelagem. 56
Figura 14 – Esquema conceitual da relação entre termos da pesquisa 60
Figura 15 – Linha do tempo dos trabalhos mais importantes em *Design Science* e *Design Science Research* 65
Figura 16 – Esquema conceitual da relação entre termos para este projeto de pesquisa 66
Figura 17 – Metodologia de pesquisa aplicada no projeto 69
Figura 18 – Representação das diversas abordagens que estudam o comportamento adaptativo. 72
Figura 19 – Configurações das buscas realizadas nas bases de pesquisa. 75
Figura 20 – Critérios aplicados aos resultados da busca. 76
Figura 21 – Resultados da aplicação dos critérios nos resultados da busca. 76
Figura 22 – Quadro comparativo dos estudos da revisão conceitual. 81
Figura 23 – Configuração da Busca para a Revisão de Escopo. 82
Figura 24 – Distribuição dos estudos da Revisão de Escopo por base de pesquisa. . 83
Figura 25 – Critérios de inclusão e exclusão da Revisão de Escopo. 83
Figura 26 – Processo de Mapeamento Sistemático. 84
Figura 27 – Lista de estudos selecionados manualmente de base de pesquisa especializada. 85
Figura 28 – Configuração da Busca para o Mapeamento Sistemático. 86
Figura 29 – Distribuição dos resultados da Busca para o Mapeamento Sistemático. 87

Figura 30 – Critérios de inclusão e exclusão do Mapeamento Sistemático. 88
Figura 31 – Lista de estudos selecionados para o Mapeamento Sistemático. 88
Figura 32 – Tipo de resultados dos estudos analisados. 89
Figura 33 – Tipo de suporte dos estudos analisados. 90
Figura 34 – Disponibilidade das ferramentas desenvolvidas pelos estudos analisados. 91
Figura 35 – Linha do tempo das ferramentas potencialmente disponíveis obtidas dos estudos analisados. 92
Figura 36 – Comparação entre ferramentas 99
Figura 37 – Guarda-chuva conceitual de adaptatividade. 104
Figura 38 – Arquitetura da solução 110
Figura 39 – Hierarquia de Propriedades Self-*. 118
Figura 40 – Modelo Conceitual dos SAS. 120
Figura 41 – Processos participantes da adaptatividade em *SaS*. 120
Figura 42 – Comparação entre os modelos de (a) processos no Ciclo Adaptativo em Salehie e Tahvildari (2009) e os (b) Processos no ciclo de controle de Cheng *et al.* (2009b). 121
Figura 43 – Características principais da Computação Autonômica por grupo. . . . 125
Figura 44 – Arquitetura geral dos sistemas de Computação Autonômica. 126
Figura 45 – Arquitetura MAPE-K. 127
Figura 46 – Modelo baseado em agentes para o desenvolvimento de CAS. 132
Figura 47 – Níveis de regras em CAS. 133
Figura 48 – Modelo geral de um Dispositivo Adaptativo. 137
Figura 49 – Representação do Dispositivo Adaptativo. 138
Figura 50 – Representação da função de um Autômato Adaptativo. 140
Figura 51 – Interação entre o sistema adaptativo e o ambiente. 141
Figura 52 – Representação básica do ambiente Env. 141
Figura 53 – Elementos básicos da adaptatividade. 144
Figura 54 – Sensores e Atuadores em Sys_{NA}. 147
Figura 55 – Mecanismo de comunicação entre o sistema e a tomada de decisão. . . 148
Figura 56 – Relação entre Autonomia e Lógica Adaptativa. 155
Figura 57 – Elementos que compõem AL. 157
Figura 58 – Elementos de um Sistema Adaptativo. 157
Figura 59 – Relação entre meta-modelo e modelos. 159
Figura 60 – Funcionamento da Meta-adaptatividade. 165
Figura 61 – Relação entre os componentes da ferramenta e os perfis de usuário. . . 167
Figura 62 – Tecnologias utilizadas na implementação do framework HM^2AT. 168
Figura 63 – Estrutura de projetos em Eclipse IDE de ambos componentes do framework HM^2AT. 169
Figura 64 – Estrutura do repositório comum do framework HM^2AT. 170

Figura 65 – Tela de apresentação do HM^{2}AT-Admin. 171
Figura 66 – Casos de uso relacionadas ao HM^{2}AT-Admin. 172
Figura 67 – Tela de incorporação de modelos no HM^{2}AT-Admin. 173
Figura 68 – Tela de incorporação de implementações no HM^{2}AT-Admin. 173
Figura 69 – Tela de incorporação de dependências no HM^{2}AT-Admin. 174
Figura 70 – Tela de incorporação de exemplos no HM^{2}AT-Admin. 174
Figura 71 – Tela de apresentação do HM^{2}AT. 176
Figura 72 – Casos de uso relacionadas à tela de apresentação do HM^{2}AT. 176
Figura 73 – Seleção de Modalidade do HM^{2}AT. 177
Figura 74 – Casos de uso relacionados à Tela de Modalidade Genérica do HM^{2}AT. . 178
Figura 75 – Tela de Modalidade Genérica do HM^{2}AT. 179
Figura 76 – Descarregar Modelo no HM^{2}AT. 179
Figura 77 – Descarregar Implementação no HM^{2}AT. 180
Figura 78 – Descarregar Dependência no HM^{2}AT. 180
Figura 79 – Descarregar Exemplo de implementação no HM^{2}AT. 181
Figura 80 – Casos de uso relacionados à Tela de Modalidade Específica do HM^{2}AT. 182
Figura 81 – Procurar Dispositivo no HM^{2}AT. 183
Figura 82 – Verificar adaptabilidade do Dispositivo no HM^{2}AT. 184
Figura 83 – Dispositivo Verificado no HM^{2}AT. 185
Figura 84 – Carregar Modelo no HM^{2}AT. 186
Figura 85 – Descarregar Lógica Adaptativa no HM^{2}AT. 187
Figura 86 – Quadro Comparativo entre as ferramentas. 188
Figura 87 – Quadro Comparativo entre as ferramentas. 189
Figura 88 – Configuração Padrão do Autômato ANormal 202
Figura 89 – Mudanças em FSMNormal. 204
Figura 90 – Diagrama da Lógica Adaptativa realizada para a prova de conceito. . . 208
Figura 91 – Configuração Padrão do Autômato FSMNormal. 225
Figura 92 – Configuração inicial do Autômato FSMAdaptativo. 226
Figura 93 – Retono do Autómato FSMAdaptativo. 226
Figura 94 – Configuração Padrão do Autómato FSM_MetaAdaptativo. 227
Figura 95 – Retono do Autómato FSMA_MetaAdaptativo. 228
Figura 96 – Exemplo da estrutura dos casos de teste de unidade nos projetos da ferramenta. 229
Figura 97 – Espaço morfológico do verbo *adapt* e termos relacionados. 269
Figura 98 – Tipos de dispositivos com diferentes capacidades de comportamento adaptativo. 271
Figura 99 – Cynefin Framework para a diferenciação dos diferentes tipos de sistemas. 276
Figura 100 – Categorização de tipos de sistema. 277
Figura 101 – Análise de características dos diferentes tipos de sistema. 278

Figura 102 – Taxonomia de Sistemas Adaptativos no nível geral. 282
Figura 103 – Taxonomia de Sistemas Adaptativos no sub-nível considerações de realização. 283
Figura 104 – Taxonomia de Sistemas Adaptativos nos sub-níveis de objeto a adaptar, características temporárias e considerações da interação. 284
Figura 105 – Taxonomia de Sistemas Adaptativos no sub-nível de considerações computacionais focando em paradigmas de computação. 285
Figura 106 – Taxonomia de Sistemas Adaptativos no sub-nível de considerações computacionais focando em modelos e recursos. 286
Figura 107 – Grupos e Categorias utilizadas para a classificação de trabalhos relacionados com adaptatividade. 290
Figura 108 – Exemplo de uso da categorização bibliográfica. 292

LISTA DE TABELAS

Tabela 1 – Lista de estudos para análise. 76
Tabela 2 – Instrumento para análise comparativa do meta-modelo holístico e outras abordagens. 192
Tabela 3 – Comparação entre o meta-modelo holístico e a abordagem das Tecnologias Adaptativas. 193
Tabela 4 – Comparação entre o meta-modelo holístico e a Computação Autonômica. 195
Tabela 5 – Comparação entre o meta-modelo holístico e MUSIC(SaS). 196
Tabela 6 – Comparação entre o meta-modelo holístico e Rainbow(SaS). 198
Tabela 7 – Comparação entre o meta-modelo holístico e o Modelo baseado em agentes dos CAS. 199
Tabela 8 – Resumo de Testes unitários aplicados em HM^{2}AT-Admin. 231
Tabela 9 – Resumo de Testes unitários aplicados em HM^{2}AT. 231
Tabela 10 – Teste unitários particulares de HM^{2}AT. 232
Tabela 11 – Casos de teste para HM^{2}AT-Admin. 233
Tabela 12 – Casos de teste para HM^{2}AT-Modalidade Genérica. 234
Tabela 13 – Casos de teste para HM^{2}AT-Modalidade Específica. 235

SUMÁRIO

1 INTRODUÇÃO . . . 25
1.1 Definição do problema . . . 29
1.2 Objetivo . . . 31
1.3 Contribuições . . . 34
1.4 Organização do trabalho . . . 36

2 CONCEITOS GERAIS . . . 39
2.1 Adaptatividade . . . 39
2.1.1 Terminologia . . . 39
2.1.2 Definição geral e intuitiva . . . 41
2.2 Sistemas Adaptativos . . . 46
2.2.1 Definição formal . . . 46
2.3 Fábricas e Linhas de produção de software . . . 49
2.4 Modelos e meta-modelos . . . 53

3 METODOLOGIA DE PESQUISA:*DESIGN SCIENCE RESEARCH* . 57
3.1 *Design Science* e *Design Science Research* . . . 61
3.2 *Design Science Research Process Model* . . . 64

4 ESTADO DA ARTE E TRABALHOS RELACIONADOS . . . 71
4.1 Panorama Atual . . . 75
4.1.1 Desenvolvimento de um marco conceitual unificado . . . 75
4.1.2 Desenvolvimento de ferramentas para assistir à construção de sistemas adaptativos . . . 82

5 PROPOSTA . . . 101
5.1 Marco Conceitual . . . 101
5.2 Framework Tecnológico . . . 103
5.2.1 Requisitos Funcionais . . . 103
5.2.2 **Requisitos Não Funcionais (NF) . . . 107**
5.2.3 **Arquitetura . . . 109**

6 META-MODELO HOLÍSTICO DE SISTEMAS ADAPTATIVOS . . 113
6.1 Sistemas adaptativos: abordagens das Ciências Naturais . . . 114
6.2 Sistemas adaptativos: abordagens tecnológicas . . . 117
6.2.1 *Self-adaptive Systems* ou *SaS* . . . 117
6.2.2 Computação Autonômica . . . 123

6.2.3 Sistemas Complexos . . . 129
6.2.4 Tecnologias Adaptativas . . . 135
6.3 Definição do Modelo Holístico . . . 140
6.4 Aplicação do Meta-modelo Holístico . . . 156
6.4.1 Instanciação do meta-modelo . . . 156
6.4.2 Paradigmas de Programação . . . 160
6.4.2.1 Programação orientada a aspectos . . . 160
6.4.2.2 Programação orientada a contexto . . . 162
6.5 Meta-adaptatividade . . . 163

7 DESENVOLVIMENTO . . . 167
7.1 HM^{2}AT-Admin . . . 170
7.2 HM^{2}AT . . . 175
7.2.0.1 Modalidade Genérica . . . 177
7.2.0.2 Modalidade Específica . . . 181
7.3 Comparação com outras ferramentas . . . 187

8 TESTES E RESULTADOS . . . 191
8.1 Provas no Nível Conceitual . . . 191
8.1.1 *Tipo de prova: Cobertura conceitual* . . . 191
8.1.2 *Tipo de prova: Prova de conceito* . . . 201
8.1.3 *Meta-adaptatividade* . . . 218
8.2 Provas no Nível Tecnológico . . . 228
8.2.1 Configuração dos Casos de Testes . . . 229
8.2.1.1 Equipamento utilizado . . . 229
8.2.1.2 Ferramentas utilizadas . . . 230
8.2.1.3 Suíte de testes . . . 230
8.2.1.4 Especificação dos casos (fluxos) . . . 232
8.3 Resultados . . . 232
8.4 Análise dos Resultados . . . 235

9 CONTRIBUIÇÕES E CONCLUSÕES . . . 237
9.1 Contribuições Científicas . . . 237
9.1.1 Suporte à pesquisa . . . 237
9.1.2 Teoria da Adaptatividade . . . 238
9.2 Contribuições Técnicas e Tecnológicas . . . 240
9.2.1 Contribuições Técnicas adicionais . . . 240
9.2.2 Contribuições Tecnológicas . . . 241
9.3 Publicações . . . 241
9.4 Conclusões . . . 243

9.5 Trabalhos Futuros **245**
9.5.1 Trabalho com Revisões Sistemáticas 246
9.5.2 Trabalhos com o meta-modelo 246
9.5.3 Trabalhos com a ferramenta 247

REFERÊNCIAS **249**

APÊNDICES **267**

APÊNDICE A – TERMINOLOGIA RELACIONADA **269**
A.1 Terminologia próxima **269**
A.2 Outros nomes **272**

APÊNDICE B – PENSAMENTO SISTÊMICO: SISTEMAS SIMPLES, COMPLICADOS E COMPLEXOS **275**

APÊNDICE C – PROPOSTA DE TAXONOMIA PARA COMPORTAMENTOS ADAPTATIVOS **281**

APÊNDICE D – PROPOSTA DE BIBLIOGRAFIA CATEGORIZADA **287**
D.1 Base de dados bibliográfica em formato digital **288**

1 INTRODUÇÃO

Na atualidade, a tecnologia possui um rol cada vez mais importante na vida diária dos membros da sociedade. De fato, nos últimos 50 anos, experimentamos uma integração tão profunda e acelerada com diversos sistemas tecnológicos massivos que a própria dinâmica de desenvolvimento humano tem se transformado com ela (ROSOL; NELSON; RENN, 2017). Hoje estamos rodeados de sistemas computacionais e dispositivos que estão contidos em objetos de diferentes formatos, desde a forma canônica de um computador, até objetos do dia-a-dia, tais como: carros, televisões, e geladeiras, passando pela ampla gama de dispositivos vestíveis (*wearable devices*). As interações entre tais dispositivos ou sistemas, seus usuários e o ambiente no qual se desenvolvem têm se tornado uma parte inseparável de muitas esferas da sociedade, a maneira mesmo de co-evoluir com ela (TRITCHKOV; GOETZ, 2016; DONGES *et al.*, 2017; WILLIAMS *et al.*, 2015). Como consequência, vivemos uma tendência de aumentos na escala e complexidade das atividades nas sociedades humanas e nas suas necessidades tecnológicas.

Assim, avanços tecnológicos permitem suportar atividades cada vez mais elaboradas e procuram auxiliar na resolução de problemas que modelam a realidade com maior detalhe. Para lidar com esse mundo de crescente complexidade, mudança constante e incerteza inerente é necessário que os sistemas computacionais possuam capacidades específicas como robustez, flexibilidade, e mecanismos de avaliação e seleção, assim como abranger domínios abertos de execução e alta tolerância a falhas. Em particular, é requerido que esses sistemas se ajustem às necessidades humanas de maneira confiável (TOMFORDE; SICK; MÜLLER-SCHLOER, 2017) conforme os seus diferentes contextos de atuação. Para isso é necessário que os sistemas ofereçam algum tipo de otimização onde as informações sejam aproveitadas assim que adquiridas de maneira que o desempenho das tarefas seja mantido ou melhorado correspondentemente. De fato, muitas das novas aplicações suportadas por tecnologias de ponta, tais como: cidades inteligentes (*Smart Cities*), Internet das coisas (*Internet of Things (IoT)*, Computação Ubíqua, Sistemas Multi-Agentes (MaS), Sistemas Evolutivos, Sistemas Ciber-Físicos (CFS), Computação Autonômica entre outros, requerem especificamente capacidades de percepção e reação em tempo real por parte dos sistemas.

No entanto, atender essas necessidades seguindo as técnicas tradicionais para o desenvolvimento de software, derivadas de um enfoque reducionista, é particularmente desafiador. Geralmente, tais abordagens consideram necessidades de usuário estáticas enquanto propõem a construção de software para condições de operação conhecidas e bem estabelecidas. Assim, os sistemas de software desenvolvidos sob essa abordagem incorporam técnicas de controle estritas, evitando situações de incerteza e tratando os problemas complexos com estratégias do tipo divisão e conquista (NETO, 2017) que

permitem desenvolver sistemas robustos e com um elevado nível de confiabilidade dentro das condições definidas inicialmente. Nesses cenários, a incorporação de flexibilidade no software é realizada a alto custo via técnicas intrincadas, tais como: refatoração, reengenharia, adaptação manual dos sistemas entre outras técnicas (KELL, 2008), que dificilmente acompanham e lógica envolvida com clareza. Essa ininteligibilidade tecnológica dificulta a subsequente retroalimentação, reutilização e manutenção do sistema em questão, assim como incrementa o uso de recursos tecnológicos e humanos.

O comportamento adaptativo, como ferramenta tecnológica, apresenta uma alternativa à abordagem tradicional para responder ao incremento da complexidade nos sistemas de software, e para atingir flexibilidade e robustez em diferentes tipos de sistemas. Simultaneamente, o comportamento adaptativo consegue acompanhar e expor de maneira mais natural tanto as características do problema, quanto sua lógica interna e sua representação. De maneira geral, o comportamento adaptativo se baseia na ideia de uma entidade capaz de modificar o seu próprio comportamento conforme as características do entorno no qual se desenvolve e seus objetivos particulares. Uma entidade com essas características é denominada uma entidade adaptativa. Esta ideia não é uma novidade e, de fato, tem sido sujeito de pesquisa considerável desde meados dos anos 60 (MACíAS-ESCRIVá *et al.*, 2013; CAYA; NETO, 2018), e tanto a comunidade acadêmica quanto a comercial têm canalizado vários esforços para estudar os princípios subjacentes ao comportamento adaptativo, assim como sua implementação e análise desde várias abordagens. Em Caya e Neto (2018) os autores apresentam um estudo bibliométrico em diversa bases científicas sobre a evolução dos trabalhos científicos desenvolvidos em relação ao comportamento adaptativo até 2018. Demos continuidade a nesse trabalho realizando uma nova análise considerando os registros disponíveis nas bases de dados acadêmicas Scopus e Web of Science até janeiro de 2023, tendo como resultado as Figuras 1 e 2.

A Figura 1 apresenta o número de trabalhos por ano que mencionam adaptatividade no título, resumo ou palavras-chave para cada uma das bases de dados, assim ela nos permite observar a evolução de publicações científicas relacionadas direta ou indiretamente às características do comportamento adaptativo. Nessa figura podemos observar o crescimento contínuo das pesquisas relacionadas a adaptatividade nos últimos 50 anos, inclusive no período da pandemia de Covid-19 em 2020-2022. A Figura 2 apresenta as principais áreas do conhecimento associadas aos trabalhos analisados e sua porcentagem de participação. Aqui observamos que no Top 10 das áreas relacionadas com adaptatividade estão representadas as Ciências Exatas e Tecnológicas, Ciências Biológicas, Engenharias, Ciências Humanas e trabalhos multidisciplinares, evidenciando a abrangência do interesse por adaptatividade na comunidade científica.

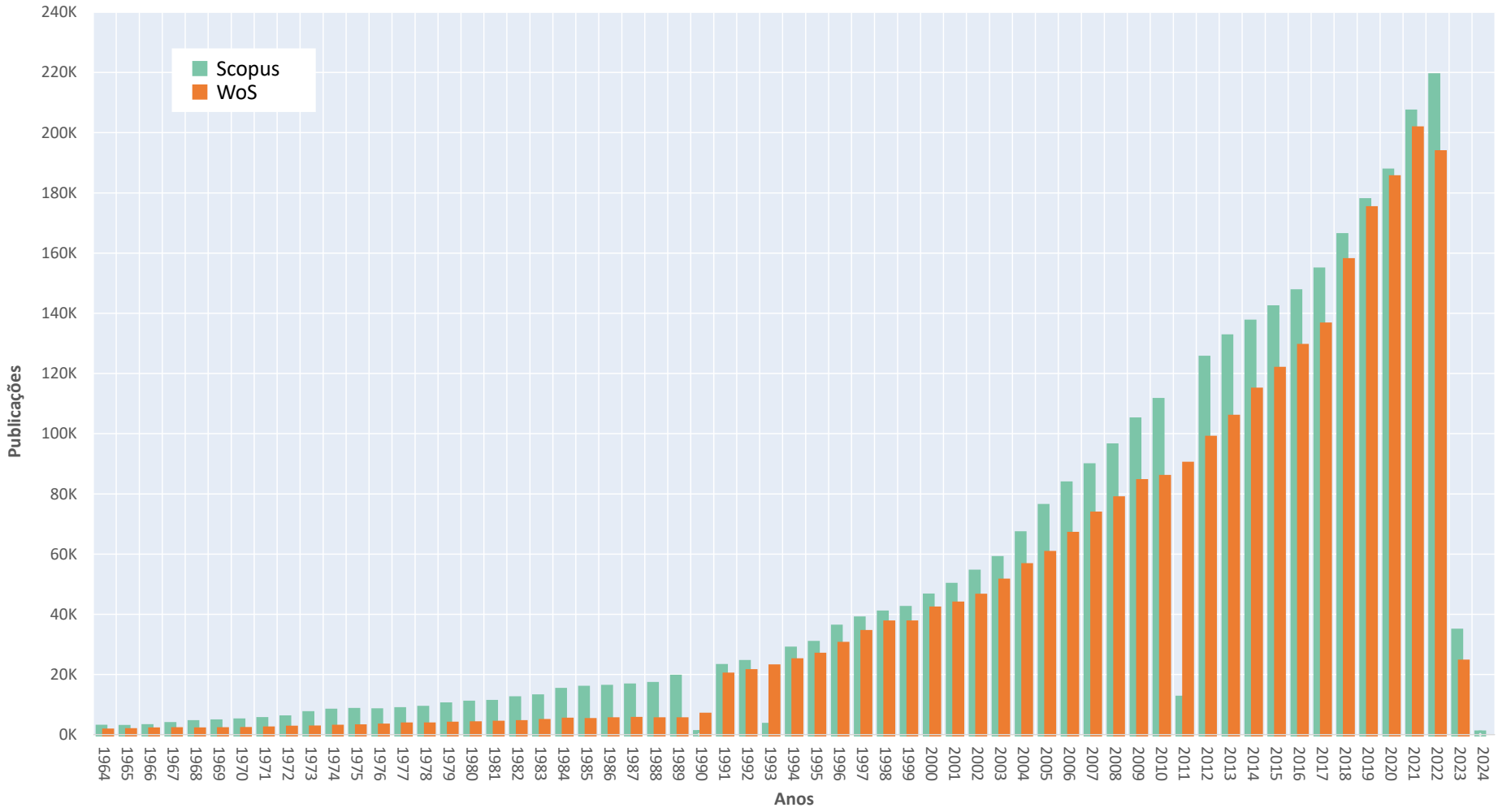


Figura 1 – Evolução de trabalhos científicos relacionados com adaptatividade em Web of Science e Scopus. Adaptada de (CAYA; NETO, 2018).

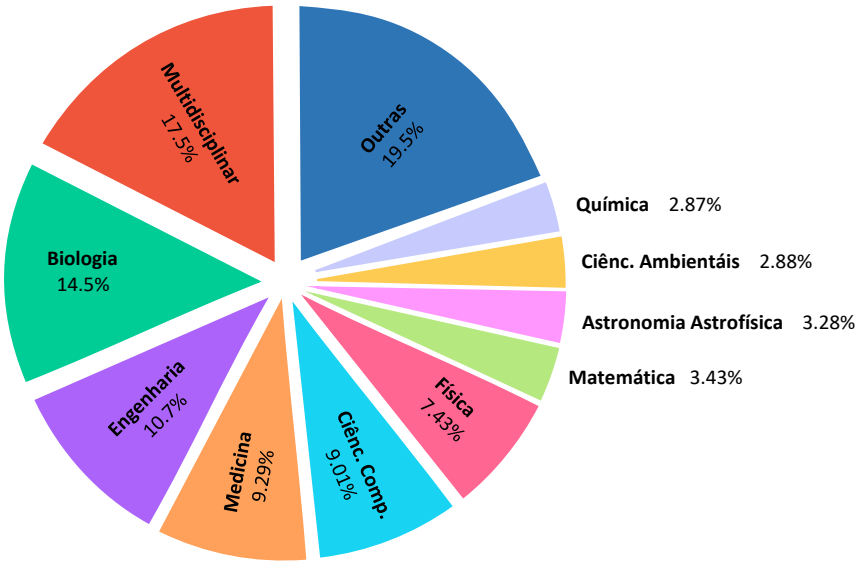


Figura 2 – Publicações com referência direta à adaptatividade por Área de Pesquisa em Web of Science e Scopus.
Adaptada de (CAYA; NETO, 2018)

Esses trabalhos científicos propiciaram o atual posicionamento tecnológico para projetar e implementar abordagens adaptativas que auxiliem na resolução de alguns problemas complexos. Particularmente, uma abordagem que incorpora o comportamento adaptativo oferece duas qualidades interessantes: o seu poder de expressão para situações complexas e sua capacidade de interação com outras técnicas. Em primeiro lugar, sua caracterização incorpora naturalmente os conceitos de complexidade, incerteza e otimização, o que fornece grande poder de expressão e compreensão. Em segundo lugar, é possível trabalhar com o comportamento adaptativo tanto como abordagem principal quanto como ferramenta complementar a outras técnicas, possibilitando a criação de abordagens híbridas que permitem usufruir de vantagens sinérgicas entre diferentes abordagens.

No entanto, mesmo quando o seu potencial é reconhecido pela maioria das instituições de pesquisa, tanto no setor acadêmico quanto em companhias de software, podemos observar que a abordagem adaptativa não tem sido amplamente incorporada no desenvolvimento de sistemas de software convencionais. Essa resistência ao trabalho direto com comportamento adaptativo nos sistemas de software, mesmo reconhecendo suas vantagens, evidencia que ainda existem algumas barreiras por superar. Algumas de essas barreiras são:

- preocupação pelo consumo de recursos no desenvolvimento de sistemas adaptativos

em comparação ao retorno que eles podem gerar;

- dificuldade de compreensão no desenvolvimento de sistemas adaptativos devido ao *lexicon* científico das áreas;
- disponibilidade de recursos para a incorporação das técnicas que suportam o comportamento adaptativo, e
- falta de técnicas de verificação, validação e teste que retornem níveis de confiança do comportamento dos sistemas requeridos em diferentes domínios.

Esta tese propõe responder aos três primeiros desafios através da implementação de modelos e mecanismos que permitam montar uma fábrica de mecanismos adaptativos com o objetivo facilitar o trabalho entre sistemas legados e tecnologias adaptativas via uma abordagem integradora.

1.1 Definição do problema

Nos últimos anos têm sido desenvolvidas diferentes iniciativas que procuram responder ao desafio de auxiliar, facilitar e incentivar a incorporação de comportamentos adaptativos em sistemas de software convencionais para poder explorar o seu potencial. Assim, diferentes áreas do conhecimento, tais como: cibernética, sistemas adaptativos complexos, teoria de controle, computação autonômica, sistemas de software *self-**, entre outras, têm realizado esforços para desenvolver novos métodos, técnicas e modelos que sirvam como diretrizes para trabalhar com adaptatividade. Da mesma maneira, têm sido propostas ferramentas e frameworks que complementam as diversas abordagens e que permitem a incorporação de alguns dos conceitos do comportamento adaptativo. Os trabalhos em cada um desses domínios permitem observar o comportamento adaptativo desde diferentes perspectivas, considerando bases cognitivas, experiências, recursos e técnicas particulares que ajudam na análise dos elementos participantes no comportamento adaptativos e suas interações.

Em termos gerais, a maioria das pesquisas associadas à adaptatividade podem ser categorizadas, segundo a participação interna atribuída ao comportamento adaptativo, em:

a) adaptatividade como ferramenta de aprimoramento: quando as pesquisas pretendem utilizar as vantagens do comportamento adaptativo para incrementar, de alguma maneira, a eficiência de suas soluções ou estender o seu espaço de aplicação.

b) adaptatividade como técnica de solução: quando as pesquisas estudam o comportamento adaptativo visando elaborar soluções para problemas específicos,

cujas características correspondem com algumas das propriedades associadas ao comportamento adaptativo, tais como: complexidade, heterogeneidade, incerteza e dinamicidade.

c) adaptatividade como objeto de estudo: quando as pesquisas estudam as características do próprio comportamento adaptativo, suas interações, vantagens e desafios com a finalidade de compreender o fenômeno.

Nas duas primeiras categorias é claro que o desenvolvimento das pesquisas depende do objeto de estudo (problema) específico, portanto, estão contextualizadas em domínios particulares. A terceira categoria, pela sua parte, corresponde ao tipo de pesquisa comprometida com o desenvolvimento de um marco teórico geral para o comportamento adaptativo que permita sua melhor compreensão. Atualmente, as pesquisas que integram essa última categoria são minoritárias, pois não existe um campo de estudo oficial encarregado de observar e compilar este tipo de pesquisa e consolidar seus avanços em uma teoria geral da adaptatividade. A ausência dessa teoria geral têm como consequência que algumas áreas do conhecimento criem sub-áreas para estudar a adaptatividade visando desenvolver mecanismos e técnicas que atendem as necessidades da área principal, utilizando os conhecimentos e recursos disponíveis dentro do domínio ao qual pertencem.

Ao analisar esse contexto com maior detalhe, é possível identificar duas problemáticas decorrentes:

(i) a maioria das pesquisas desenvolve avanços "circunscritos" a domínios específicos. A falta de um marco teórico unificado para a adaptatividade que permita convergir as informações obtidas desde as perspectivas particulares propicia a proliferação de análises *ad hoc* que, consequentemente, dificultam a tarefa de enriquecer a análise e o conhecimento do próprio fenômeno adaptativo.

(ii) a maioria dos trabalhos propõe abordagens de "**construção integral**", significando que a existência de adaptatividade em um sistema implica que tal sistema deve ter sido planejado e construído necessariamente com uma abordagem adaptativa. Assim, a adaptatividade é considerada uma característica inata do sistema, portanto um sistema adaptável, não é considerado na maioria de propostas de adaptatividade como um potencial ponto de partida para o desenvolvimento de sistemas adaptativos;

Ambas as problemáticas são consideradas de caráter interdisciplinar e têm sido colocadas como desafios e considerações futuras em diversas pesquisas.
A primeira problemática é decorrente da ausência de uma teoria unificada para a adaptatividade, na qual cientistas de diferentes áreas colaborem priorizando uma abordagem geral e transdisciplinar. A crescente demanda por aplicações cada vez mais inteligentes

e realistas, de simulação em alta definição e sistemas críticos, são alguns exemplos que reforçam a necessidade de uma teoria adaptativa unificada. Ao contrário da visão fragmentada fornecida pelas soluções particulares, uma abordagem unificada permitiria construir um modelo mais rico e completo sobre o comportamento adaptativo possibilitando sua melhor compreensão como fenômeno ao integrar as diferentes visões do prisma cientifico. Além disso, uma abordagem centralizada poderia agir como língua franca para auxiliar as pesquisas de áreas particulares ao respeito dos diferentes recursos disponíveis para a construção de sistemas com essa característica.

A segunda problemática reflete tanto a reação à primeira problemática quanto a tendencia nas pesquisas sobre adaptatividade. Em primeiro lugar, a falta de um marco conceitual sobre adaptatividade e as idiossincrasias de cada abordagem conduzem a que diferentes áreas prefiram desenvolver sua própria proposta sobre comportamento adaptativo para satisfazer seus objetivos de pesquisa. A tendência principal tem sido a construção de sistemas adaptativos com recursos próprios (conhecimentos de/para a própria área), para sua posterior aplicação em diferentes objetos de estudo, e análise do seu desempenho. Nesse sentido, a maioria das vezes é proposta a construção de novos sistemas, sistemas planejados para se adaptar e equipados com recursos particulares correspondentes. Frequentemente, o sistema convencional e o sistema adaptativo só diferem na manifestação do comportamento adaptativo pontual e não em alguma característica funcional. Dessa maneira, tem sido pouco explorado o espaço de transição dos sistemas criados sob a abordagem tradicional, que apresentam certas características potenciais para a adaptatividade, para sistemas adaptativos mediante a incorporação de mecanismos complementares adequados. O espaço citado é de grande interesse pois a maioria de sistemas disponíveis na atualidade possuem tais características e a sua transição significaria o acesso às vantagens do comportamento adaptativo, mantendo suas funcionalidades e com um investimento de recursos acessível. Assim, existe a necessidade de uma abordagem de "incorporação" para o desenvolvimento de sistemas adaptativos que permita a reutilização de recursos.

Até o momento de apresentação de este documento, não é do nosso conhecimento que esteja disponível alguma proposta para suportar a evolução de dispositivos adaptáveis, com potencial adaptativo, sobre uma visão holística e flexível para trabalho conjunto com as abordagens, ferramentas, tecnologias, recursos já desenvolvidos. Esta é, portanto a contribuição principal deste trabalho.

1.2 Objetivo

Considerando as problemáticas identificadas anteriormente, o nosso trabalho tem como objetivo principal:

Desenvolver uma proposta ***geral*** *que* ***suporte****, de maneira eficiente, a* ***transformação*** *de sistemas de software convencionais (já existentes) que apresentem as condições adequadas em sistemas adaptativos, através da implementação e incorporação semi-automática dos elementos complementares necessários.*

Em negrito encontram-se destacadas os principais elementos desta tese: (i) desenvolvimento de um suporte conceitual e ferramental que facilite o trabalho com adaptatividade através da (ii) transformação de sistemas de software convencionais com características potenciais e sob uma (ii) abordagem geral que garanta uma interação com diferentes áreas de conhecimento e permita o intercâmbio de conceitos e ferramentas.

Para atingir esse objetivo é necessário responder às diferentes questões de pesquisa nele contidas:

QP1 Geral: Como garantir a generalidade da proposta conceitual?
A proposta deve ser geral num esforço por contribuir com a formação de uma teoria unificada para o estudo da adaptatividade, de maneira que possa ser compreendida, utilizada e complementada por diversas áreas do conhecimento com facilidade. A recomendação de evoluir e hibridizar os conceitos, ferramentas, teorias, métodos e tecnologias de diferentes áreas tem sido um assunto constante entre os cientistas com abordagens multidisciplinares para observar o comportamento adaptativo, como podemos observar em Macías-Escrivá *et al.* (2013).

QP2 Transformação: Como transformar os sistemas convencionais em sistemas adaptativos?
Na atualidade existem sistemas criados sob a abordagem tradicional com diferentes graus de reconfiguração e flexibilidade, alguns só permitem personalizações limitadas enquanto outros aceitam mudanças maiores nas suas configurações. Alguns destes sistemas possuem as condições necessárias para ser transformados em sistemas adaptativos com a assistência de mecanismos especializados. Portanto, é imperativo definir tanto as condições necessárias para a atuação do comportamento adaptativo quanto os requerimentos para a transformação de software. Neste contexto cabe mencionar que é importante conhecer a terminologia do assunto, para distinguir entre sistemas com potencial para exercitar a adaptatividade (adaptáveis) e sistemas que manifestam um comportamento adaptativo (adaptativos). O Apêndice A apresenta uma revisão básica das terminologias do assunto com o intuito de esclarecer algumas ambiguidades.

QP3 Incorporação: Como modelar a incorporação dos elementos necessários?
À diferença da maioria de abordagens, que propõem a (re)construção "integral" de sistemas para atingir o comportamento adaptativo, neste trabalho propomos uma

abordagem de "incorporação" a partir da qual se busca complementar os sistemas convencionais com os elementos adaptativos necessários. A implementação de tais elementos deve ser semi-automática, reutilizável, e manter, na medida do possível, a generalidade conceitual na implementação. Isto é, acessibilidade, desacoplamento, linguagens multi-plataforma, e especificações flexíveis são algumas das características a serem priorizadas.

QP4 Suporte tecnológico: Como implementar os elementos necessários e a sua incorporação com o software?
A solução deve fornecer uma ferramenta que assista o processo de transformação de software com clareza e simplicidade, com um consumo de recursos tecnológicos baixo ou médio, e com uma curva de aprendizado media. Diferentes técnicas de integração de software podem ser utilizadas, dependendo dos recursos disponíveis no sistema, assim como da abordagem tecnológica. De maneira particular, as técnicas relacionadas a fábrica de software são de interesse devido à sua flexibilidade porque visam a produção eficiente de famílias de soluções, porem permitindo atender as idiossincrasias de diferentes contextos.

Assim, o escopo desta tese está limitado em primeiro lugar pelo trabalho com sistemas de software já desenvolvidos, ou legados. Nesse contexto, é importante esclarecer que não é objetivo deste trabalho a manipulação do funcionamento geral do sistema de software original, ou seja, não se procura alterar os processos que satisfazem os requisitos funcionais nos sistemas adaptáveis. Pelo contrário, a proposta procura facilitar a implementação e integração de elementos auxiliares nos sistemas convencionais para suportar comportamentos adaptativos enquanto é preservada, na medida do possível, a funcionalidade base do sistema submetido. Assim, dado um sistema convencional com características adaptáveis, a proposta verifica os requisitos estruturais e funcionais mínimos para sua evolução, cria os elementos correspondentes e monitora a sua integração com o sistema fornecido.

Da mesma maneira, o escopo deste trabalho abrange a construção da estrutura necessária para a evolução de software, e a implementação de algumas instâncias de mecanismos e dispositivos com a finalidade de exercitar o conceito. Não faz parte do escopo o desenvolvimento de uma variedade de mecanismos adaptativos, mas apresentar alguns exemplos de como desenvolvê-los. É esperado que a ferramenta de software seja flexível e possa ir incrementando o volume de recursos viáveis de acordo com sua utilização. Com essa finalidade, tanto a parte teórica de abordagem de incorporação, quanto a parte de modelagem e implementação, usam enfoques orientados a componentes para permitir a maior flexibilidade possível.

Com base no objetivo principal e na resposta às questões de pesquisa são formulados os seguintes objetivos secundários:

- Definição geral do meta-modelo de adaptatividade: um meta-modelo resultado da análise geral do comportamento adaptativo como fenômeno presente em diferentes áreas de conhecimento. O meta-modelo descreve as condições sobre as quais existe o comportamento adaptativo, os elementos participantes, suas interações e algumas das características decorrentes desse comportamento.
- Verificação de requisitos para a evolução de dispositivos: uma estratégia para reconhecer se existem as condições necessárias na tecnologia subjacente para ser submetida ao processo de evolução de software e criação da correspondente camada adaptativa.
- Criação da plataforma de software para a implementação de camadas adaptativas: modelagem e implementação da plataforma de software na qual são realizadas as verificações e configurações necessárias para a criação da camada adaptativa para um sistema em particular. É importante mencionar que se bem o resultado da plataforma é fornecer uma camada adaptativa para um elemento tecnológico em particular, a plataforma mesma é desenvolvida com foco nas metodologias baseadas em componentes. Assim, ela procura reutilizar a maior quantidade de tecnologia disponível para montar uma proposta que à sua vez é geral o suficiente para atender uma diversidade de tecnologias e ser estendida incrementalmente.

1.3 Contribuições

Segundo o seu foco, as contribuições deste trabalho estão divididas em três categorias: Contribuições Científicas, Contribuições Técnicas e Tecnológicas e Publicações. A Figura 3 apresenta as contribuições principais de cada grupo.

As Contribuições Científicas estão subdivididas em:(a) Suporte à pesquisa: reúne os instrumentos criados ao longo do projeto com a finalidade de suportar algumas das tarefas dos pesquisadores, tanto de maneira geral quanto no contexto de pesquisas relacionadas a adaptatividade, e (b) Adaptatividade: contém as contribuições pontuais relacionadas com o estudo do comportamento adaptativo.

As Contribuições Técnicas e Tecnológicas contém as documentações técnicas e ferramentas criadas como parte do desenvolvimento do projeto. Finalmente, a categoria de Publicações apresenta os trabalhos científicos publicados para comunicar os avanços realizados neste trabalho.

Maiores detalhes sobre cada contribuição assim como o acesso aos recursos correspondentes são fornecidos no Capítulo 9 deste documento.

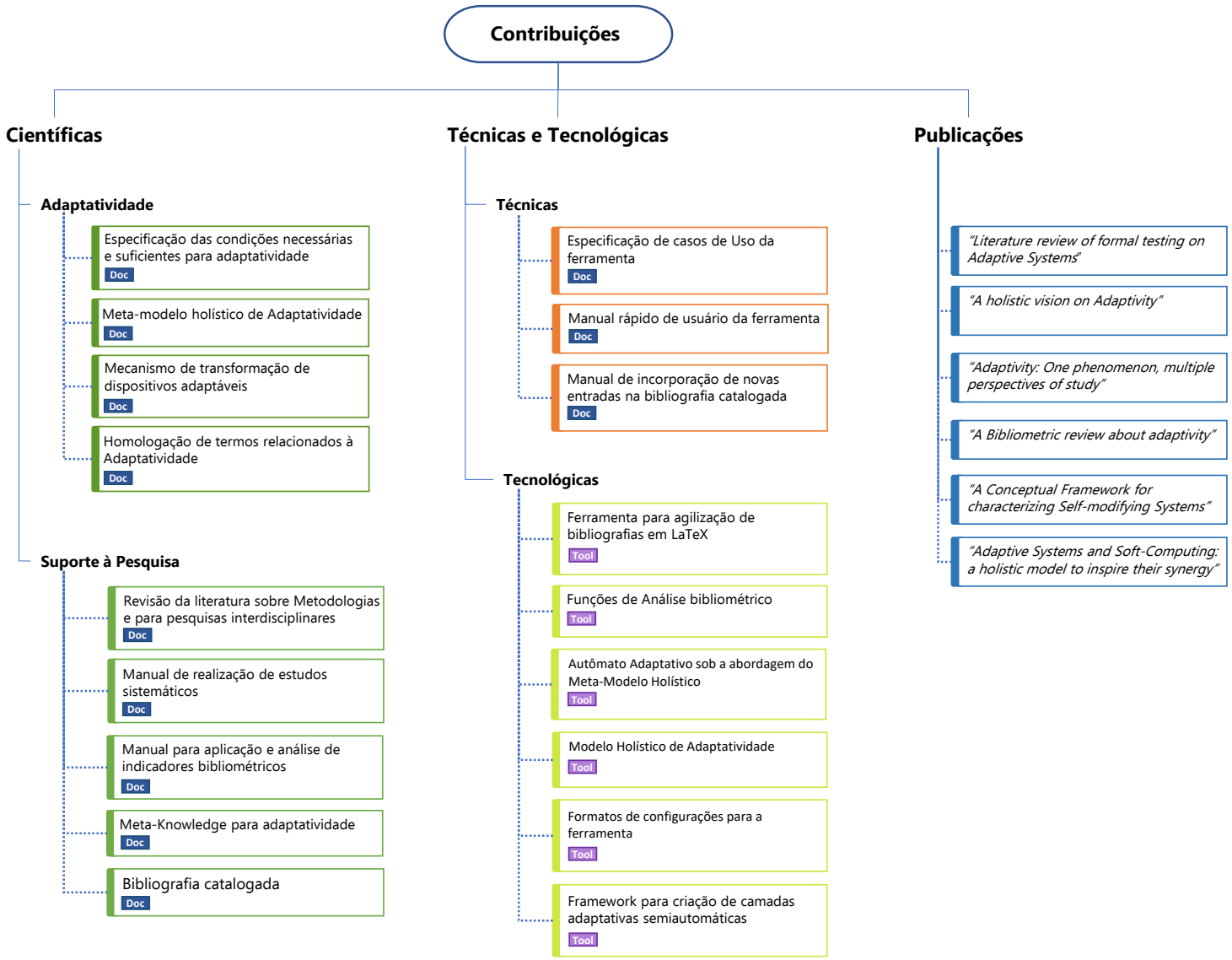


Figura 3 – Contribuições principais da tese.
O Autor(2021)

1.4 Organização do trabalho

Esta tese está organizada em nove capítulos que apresentam primeiro os conceitos básicos necessários para entender o problema e a nossa proposta de solução, e depois se aprofundam em assuntos de implementação dessa proposta. A continuação uma breve descrição do conteúdo de cada um dos capítulos deste documento.

O Capítulo 2 tem como foco a introdução dos conceitos básicos a serem utilizados neste trabalho. Num primeiro momento, é apresentada uma definição intuitiva de adaptatividade com algumas terminologias importantes para diferenciá-la de outros fenômenos relacionados. Num segundo momento, sobre essa base, são definidos formalmente os sistemas adaptativos, sendo fornecida uma representação do comportamento adaptativo nesses sistemas. Finalmente, o capítulo introduz o paradigma de desenvolvimento de software baseado em componentes e descreve a estratégia de fábricas e linhas de produção de software, o seu contexto histórico e motivação, as técnicas e métodos disponíveis para a implementação e algumas das aplicações para adaptatividade com esta estratégia.

O Capítulo 3 está dedicado à metodologia de pesquisa seguida neste trabalho. Nele são descritos os processos e técnicas utilizados para definir o objetivo de pesquisa, formular e responder às perguntas decorrentes dele e elaborar uma estratégia para propor uma solução.

O Capítulo 4 apresenta de maneira mais geral o estado da arte do problema de desenvolvimento de tecnologias adaptativas utilizando dispositivos já existentes. Assim, apresenta alguns dos trabalhos de pesquisa que visam resolver problemas próximos ao problema de estudo neste trabalho, indicando as diferentes abordagens utilizadas para sua solução até o momento de submissão deste documento, e a localização da nossa proposta frente a tais abordagens.

O Capítulo 5 está dedicado à descrição da nossa proposta para o desenvolvimento de tecnologias adaptativas a partir de dispositivos adaptáveis sob uma abordagem integradora. Neste capítulo são detalhadas as características conceituais, técnicas e tecnológicas que viabilizam a construção da nossa proposta.

No Capítulo 6, é apresentada a visão holística da abordagem integradora para o desenvolvimento de tecnologias adaptativas no formato de meta- modelo geral do comportamento adaptativo. Assim mesmo, este capítulo descreve a teoria de evolução de dispositivos com características adaptáveis para dispositivos adaptativos, seus requerimentos, vantagens esperadas e limitações decorrentes.

O Capítulo 7 está dedicado ao desenvolvimento da proposta. Neste Capítulo são especificadas as decisões tomadas para a arquitetura e implementação do suporte tecnológico da abordagem integradora. Assim mesmo, ele descreve as ferramentas utilizadas na construção da ferramenta, esclarecendo quais são partes originais (desenvolvimento

integral) deste trabalho e quais as peças de software incorporadas (desenvolvimento de terceiros) assim como as eventuais modificações (configurações e extensões) realizadas para satisfazer as finalidades deste trabalho.

O Capítulo 8 contém as informações correspondentes às provas a serem aplicadas para observar o desempenho da proposta e os resultados observados. No primeiro lugar, são aplicadas provas relacionadas ao conceito de meta-modelo e sua abrangência. Depois, são descritas as provas tecnológica a serem aplicadas, prova unitárias e provas de sistema e as configurações de cada uma. Finalmente, o capítulo apresenta os resultados recolhidos depois da aplicação das provas e a discussão desses resultados.

O Capítulo 9 apresenta as contribuições técnicas e tecnológicas assim como as conclusões finais do trabalho e algumas sugestões de assuntos a serem tratados em trabalhos futuros. As conclusões incluem aprendizados como parte do projeto de pesquisa científica.

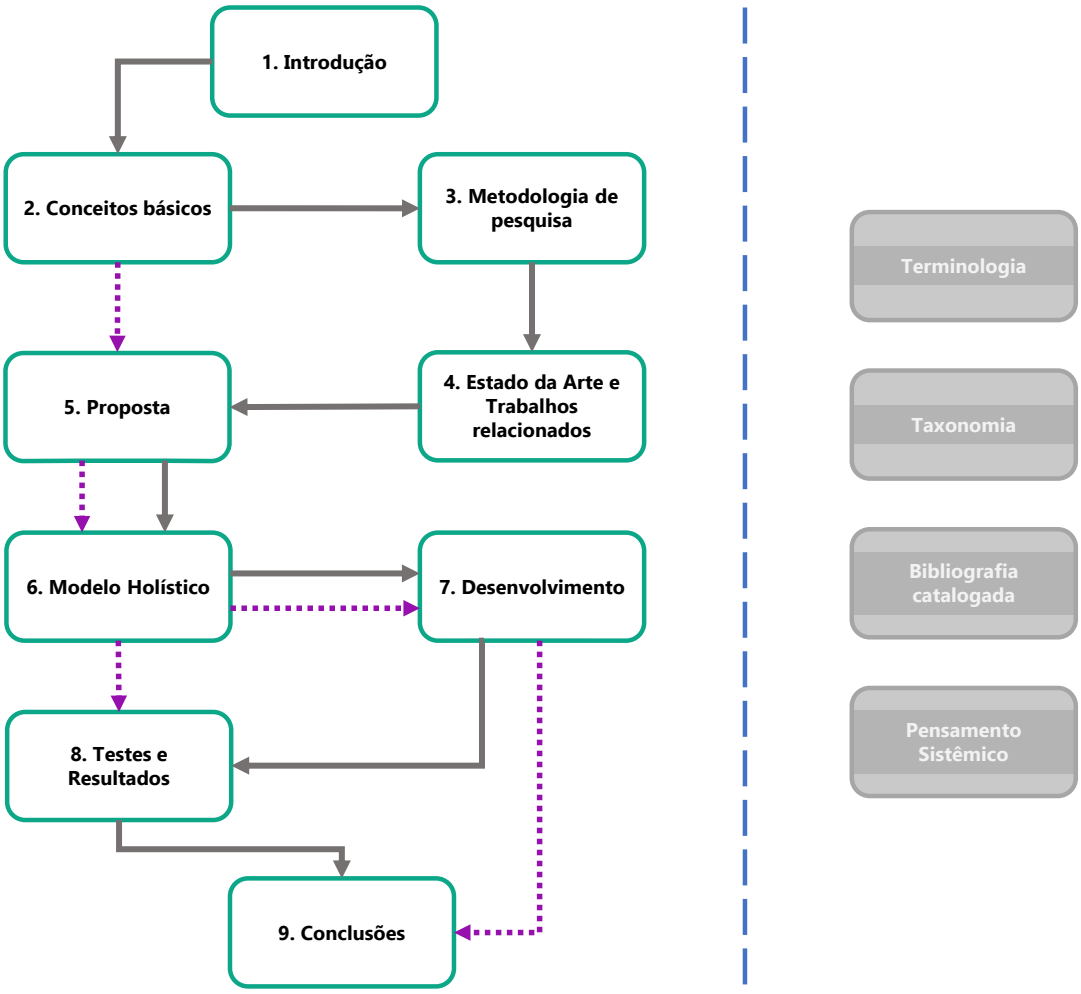


Figura 4 – Mapa de leitura da tese.
O Autor (2020)

A Figura 4 apresenta um mapa com duas possibilidades para a leitura deste documento: um caminho padrão e um caminho alternativo. O caminho padrão, representado por linhas contínuas, segue a sequência dos capítulos no documento. O caminho alternativo, mais breve, e representado pelas linhas pontilhadas fornece uma leitura focada nas contribuições do trabalho.

2 CONCEITOS GERAIS

Neste capítulo são detalhados os conceitos gerais necessários para a compreensão da pesquisa desenvolvida.

Num primeiro momento são apresentados os conceitos relacionados ao comportamento adaptativo como fenômeno de estudo, fornecendo exemplos de comportamentos adaptativos em contextos naturais para elaborar uma definição geral. Depois, são definidos os sistemas adaptativos, tanto de maneira intuitiva como formal, sendo apresentadas algumas das abordagens científicas e tecnológicas mais utilizadas para sua modelagem e a representação do comportamento adaptativo. Em seguida é definido o conceito de meta-modelo, e o seu uso no contexto das Ciências e Engenharias. Adicionalmente são apresentados alguns formatos para a definição e aplicação de meta-modelos e alguns exemplos conhecidos. Finalmente, é introduzido o conceito de fábrica de software, dando ênfase ao foco com o qual o conceito é empregado neste trabalho.

2.1 Adaptatividade

Nesta seção são apresentados os principais conceitos relacionados ao comportamento adaptativo. São introduzidas algumas considerações de terminologia básica para permitir uma referência apropriada ao assunto de estudo. Depois, é elaborada uma definição geral do comportamento adaptativo, identificando suas particularidades, inicialmente sobre a base da intuição, para depois desenvolver uma definição formal, porém flexível.

2.1.1 Terminologia

Quando as palavras da linguagem cotidiana são reutilizadas para servir como termos técnicos, elas são susceptíveis a mal-entendidos, pois às vezes elas não são aplicadas com o rigor necessário. Esse é o caso com a palavra “adaptatividade”, adjetivo que caracteriza o comportamento adaptativo. Esse é um termo utilizado em diferentes áreas do conhecimento com a finalidade de descrever algumas características dos objetos de estudo particulares. Em cada um desses contextos de pesquisa o termo “adaptatividade” pode carregar um valor semântico diferente, por exemplo: sua definição pode empregar terminologia genérica ou terminologia específica da área restringindo a sua compreensão; sua descrição pode ter maior ou menor detalhe; e até pode fornecer mais ou menos informação sobre os processos decorrentes deste comportamento. Em consequência, qualquer tentativa de análise sobre adaptatividade deve começar com sua definição básica. Uma abordagem interessante para

enfrentar este desafio é observar como este termo tem sido caracterizado no passado[1]

A adaptação é um fenômeno observável da vida e uma propriedade essencial de todos os organismos vivos (DOBZHANSKY, 1968; LINTS, 2010), os quais têm a habilidade de crescer e evoluir com o tempo, assim como de se ajustar às mudanças frequentes ao seu redor (BARANDIARAN; MORENO, 2008; DOBZHANSKY, 1968). Como tal, o comportamento adaptativo tem sido estudado por filósofos, biólogos e historiadores naturais desde tempos antigos (REEVE; SHERMAN, 1993). Assim, as primeiras definições e descrições sobre adaptação estão relacionadas à sobrevivência e/ou reprodução de indivíduos, encontrada de maneira natural só em seres vivos. Nas palavras do biólogo evolutivo Theodosius Dobzhansky:

> *Adaptation is the evolutionary process whereby an organism becomes better able to live in its habitat or habitats* (DOBZHANSKY, 1968).

Assim, a adaptação é essencialmente descrita como um processo evolutivo, dinâmico e contínuo pelo qual uma espécie está constantemente se ajustando, de maneira autônoma, ao seu ambiente. No entanto, fora da biologia, o termo "adaptação" é associado de maneira prática com outro significado: ele se refere às características nas espécies resultantes do processo de ajuste contínuo. Neste trabalho usamos o termo "adaptação" (*adaption* em inglês) para nos referir ao processo evolutivo, e "característica adaptativa"(*adaptive trait* em inglês) para falar sobre a resultante de ter passado pelo processo de adaptação, a maneira de distinguir os dos diferentes sentidos com os quais a palavra é utilizada e assim evitar confusões.

O "comportamento adaptativo" (*adaptive behavior* em inglês) é qualquer atividade que permite que um organismo se ajuste a uma situação ou ambiente particular. Assim, enquanto o termo "adaptação" se refere ao processo com uma visão geral, o termo "comportamento adaptativo" observa este mesmo processo destacando as atividades desenvolvidas para viabilizá-lo (foco nas atividades).

Um "organismo adaptativo" é aquele que possui ou exibe a capacidade de se adaptar, de se submeter a mudanças conforme as necessidades impostas no seu ambiente, é dizer, que possui adaptatividade. A "adaptatividade" (*adaptivity* ou *adaptiveness* em inglês) é a capacidade pela qual um organismo exercita a decisão de aplicar mudanças na sua estrutura ou comportamento para se adequar melhor a um ambiente. Assim, a adaptatividade é o acionador do processo de adaptação, que à sua vez se suporta no comportamento adaptativo para efetivar suas considerações.

[1] Os trabalhos de Yates (YATES, 1984) e Reeve (REEVE; SHERMAN, 1993) fazem um excelente trabalho apresentando revisões históricas detalhadas tanto das abordagens quanto das definições mais conhecidas da adaptatividade na Biologia destacando a descoberta de novos elementos e características.

Uma "característica adaptativa" é um aspecto do padrão de desenvolvimento de um organismo que permite ou aumenta a probabilidade desse organismo sobreviver e se reproduzir num contexto determinado desempenhando uma função específica. É uma evidência, uma característica (visível ou não) adicionada ou removida no organismo (que passa a ser um organismo adaptado) como resultado de uma mudança adaptativa em resposta às condições impostas. Uma "mudança adaptativa"(*adaptive change* em inglês) é a decisão de um organismo de realizar um ajuste nas suas características, capacidades e/ou funcionalidades para obter uma melhor adequação ao ambiente em que se desenvolve. Não todas as características adaptativas são mantidas ao longo da vida do organismo, assim algumas delas ficam em desuso e viram vestígios, enquanto outras podem passar a ser utilizadas com propósitos diferentes para o qual foram criadas (exadaptações). É possível, inclusive, que algumas características adaptativas gerem maior dano do que assistência, nesse caso elas são chamadas de má-adaptações.

A "adaptância" (tradução livre do termo *adaptedness* em inglês) ou adequação é a medida que reflete o grau em que um organismo consegue viver e se reproduzir em um dado conjunto de habitats. Uma representação da relação entre esses termos é apresentada na Figura 5.

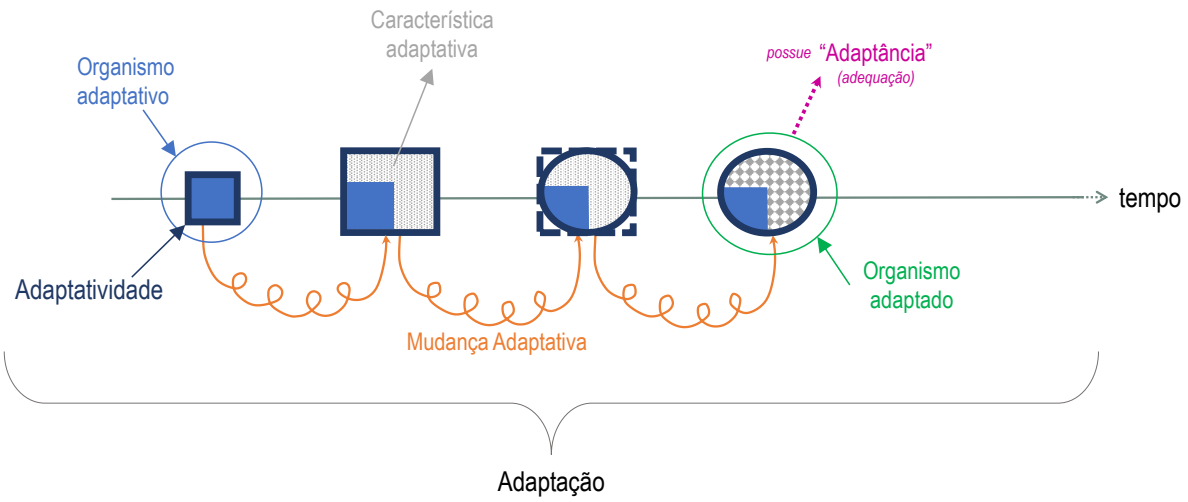


Figura 5 – Representação dos diferentes estados e terminologias envolvidas no processo adaptativo.

O Autor (2017).

O Anexo A analisa com maior detalhe diferentes aspectos de terminologia e alguns dos desafios neste sentido presentes ao pesquisar sobre adaptatividade.

2.1.2 Definição geral e intuitiva

Conforme mencionado anteriormente, a literatura se refere aos estudos nas Ciências Naturais como um dos primeiros campos de exploração do comportamento adaptativo

(HOLLAND, 1992; PRINCIPE; EULIANO; LEFEBVRE, 1999). Os estudos nestas áreas proporcionam diversas informações para a análise dos elementos participantes no comportamento adaptativo e as suas interações.

Existe uma quantidade enorme e diversa de exemplos disponíveis na natureza, mas alguns dos mais facilmente identificados são:

- **Camaleão**: os camaleões possuem uma diversidade de adaptações que os ajudam a sobreviver em diferentes condições desde as florestas tropicais até os desertos. Suas cabeças encapuzadas, olhos giratórios, e pés horizontais são algumas delas, no entanto, a sua adaptação mais reconhecível é a capacidade de mudança de cor da pele. Durante muito tempo pensou-se que a mudança de cor nos camaleões tinha a única função de camuflagem, mas, além disso, ela exercita várias outras finalidades: é usada como sinalização social (comunicação), termorregulação e outras condições [2].

- **Panda Gigante**: para melhorar suas práticas alimentares, os pandas-gigantes desenvolveram algumas adaptações, como, por exemplo, um sexto polegar, e mandíbulas e dentes muito fortes. Ao contrário de outros ursos, os pandas-gigantes não hibernam durante o inverno. Em vez disso, eles realizam migrações para zonas com melhores condições (mais quentes e com maior presença de alimento). Os ursos pandas provavelmente evoluíram para se alimentar de bambu devido à falta de acessibilidade de outro tipo de alimentos. Eles ajustaram sua dieta para consumir diferentes tipos de bambu em diferentes estações baseados nos nutrientes da planta num momento particular do ano (brotos ou maduros) e também o consumo de diferentes partes do bambu dependendo de suas próprias necessidades.

- **Chita**: A chita é o animal terrestre mais veloz, podendo acelerar de 0 a mais de 80 quilômetros por hora depois de apenas 3 passos. Assim, a sua estrutura física dela está preparada para sprints de alta velocidade de até 120 quilômetros por hora. Sua estrutura corporal pequena e aerodinâmica, de esqueleto leve e espinha altamente flexível são algumas das adaptações para ter um desempenho ótimo em altas velocidades. Pernas mais longas e magras que possuem ligamentos especiais para ajudar na resistência, garras semi-retráteis para aumentar a tração, cauda longa para fornecer estabilidade nas curvas em alta velocidade são algumas das suas características.

[2] Numa pesquisa publicada em 2015 (TEYSSIER *et al.*, 2015) os autores descobriram que o mecanismo de mudança de cor nos camaleões está composto por duas camadas sobrepostas dentro de sua pele que controlam sua cor e termorregulação. A camada superficial contém alguns pigmentos amarelos, e a segunda camada está formada por células com cristais de guanina. Estas células podem contrair-se ou dilatar-se, conforme as reações nervosas do animal, e provocar uma repartição desigual dos nanocristais (que altera o comprimento de onda da luz refletida nos cristais), resultando nas modificações de cor.

Figura 6 – Alguns exemplos de comportamento adaptativo na natureza.
O Autor (2017).

Estes três exemplos correspondem com organismos adaptativos individuais, isto é, o organismo é apenas composto por um indivíduo, porém os organismos adaptativos também podem ser coletivos. O exemplo mais facilmente referenciado é o da colônia de formigas. Uma colônia de formigas é a unidade básica na qual as formigas organizam o seu ciclo de vida. Ela é uma estrutura bem organizada e hierárquica, com papéis específicos e comportamento codificado que se baseia no seu sistema de comunicação por feromônios. O mecanismo de comunicação baseado no feromônio é a base da construção de conhecimentos dentro da colônia, tanto no nível individual quanto no coletivo. Assim, no nível individual, uma formiga usa sua experiência recente de interações com outras formigas para decidir o que fazer a seguir. No nível coletivo, o mecanismo de comunicação permite a agregação de informações e a construção de um tipo de memória coletiva que supera a memória individual e até o tempo de vida dos seus membros. É esta memória coletiva a que permite que a colônia altere o seu comportamento como resposta a restrições do ambiente, por exemplo, regulando a sua população segundo a disponibilidade de alimentos (BURNS *et al.*, 2020). Assim, por exemplo, tem sido observado que colônias maiores e mais velhas respondem de maneira mais equilibrada e sabia do que colônias menores e mais novas, mesmo quando as colônias mais velhas não estão compostas por formigas mais velhas e sabias (GORDON, 2018).

Analisando esses e outros numerosos exemplos encontrados na natureza é possível elaborar uma primeira definição intuitiva da adaptatividade. Assim dizemos que:

> Adaptatividade é a capacidade de uma entidade de aplicar, de maneira *autônoma*, continuas *mudanças* nas suas próprias características para se ajustar melhor ao seu *ambiente* para incrementar suas probabilidades de *sobrevivência*. Tais mudanças são o resultado de uma análise tanto das condições percebidas no ambiente circundante quanto das informações prévias que a entidade possui e como resposta a um estímulo.

Em primeiro lugar é necessário esclarecer que o uso do termo “entidade” como alternativa a “organismo” permite incluir nesta definição os objetos de estudo que não possuem vida ou correspondência direta a seres vivos. Assim, esta consideração em terminologia permite fornecer generalidade à definição. Da mesma maneira, uma entidade pode ser tanto individual quanto coletiva.

A “autonomia” significa que a decisão de efetivamente alterar qualquer um dos seus componentes vem unicamente da análise feita pela própria entidade sobre os benefícios que ela mesma pode obter praticando tal alteração.

As “mudanças” são resultados da execução direta do conjunto de ações que mais se adéquam à situação analisada. É importante considerar que elas podem ter repercussões indiretas sobre outras ações disponíveis que definem a estrutura e/ou comportamento da entidade, tais como: incorporação/remoção de recursos para aproveitar oportunidades, evitar ameaças ou se preparar para encarar situações de perigo.

A motivação da realização das adaptações é uma melhor adequação da entidade e do seu objetivo à nova situação. Assim, a entidade tenta responder às alterações ocorridas no seu “ambiente” circundante. Tal ambiente é definido tanto pelas características internas (estrutura e comportamento) da própria entidade, as características externas (recursos, eventos, e os objetos que compõem o cenário) e os canais de interação entre eles.

Neste contexto, o termo “sobrevivência” deve ser entendido como um objetivo de alto nível que a entidade visa alcançar. Geralmente este objetivo é definido por uma autoridade externa de alto nível: um líder, um gerente, uma necessidade ou, dentro do contexto biológico, o próprio processo de evolução e sobrevivência da espécie.

A Figura 7 apresenta de maneira resumida a relação entre os termos, relativos ao comportamento adaptativo. O esquema foi elaborado com base na revisão da literatura realizada para este trabalho.

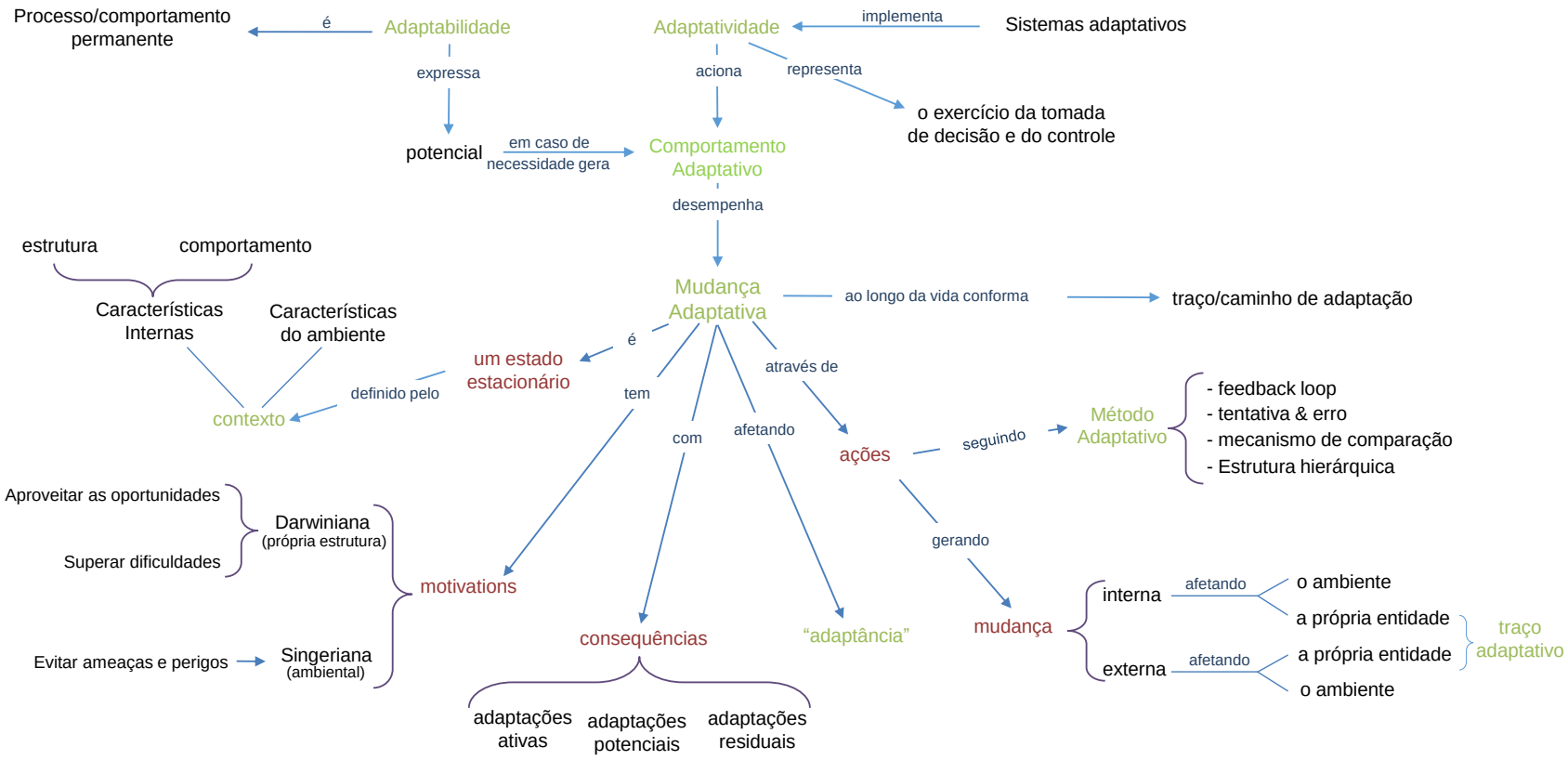


Figura 7 – Mapa conceitual dos termos relacionados ao *comportamento adaptativo*.
O Autor (2017).

2.2 Sistemas Adaptativos

Como observado na seção anterior, as entidades adaptativas exibem uma série de características importantes e, como resultado, fornecem um novo recurso para os pesquisadores. A adaptatividade permite explorar e formular potenciais soluções para problemas que anteriormente pareciam impossíveis de resolver sob abordagens tradicionais (RAIBULET, 2018). De maneira particular, as áreas das Ciências Exatas têm manifestado especial interesse na resolução de problemas empregando elementos que possuem comportamentos adaptativos. Nessas áreas, as técnicas matemáticas e computacionais tradicionais, i.e., hard-computing, se mostraram ineficazes para muitos problemas desafiadores. Assim, nos últimos 30 anos, tanto o avanço das tecnologias quanto o aumento da complexidade dos sistemas fomentaram o interesse das diferentes comunidades científicas na observação, descrição, prescrição do comportamento adaptativo e, em consequência, a análise das suas possíveis contribuições para a resolução de problemas.

Neste contexto, dada a natureza multidisciplinar deste trabalho, é preciso esclarecer a definição de sistema que irá ser considerada. Como apontado por Hitchins (2003), mesmo quando pode parecer um conceito simples, a definição do termo "sistema" apresenta muitos aspectos na literatura e pontos de vista nos quais nem os especialistas concordam totalmente. Assim, a seguinte definição é adotada:

> Um sistema é considerado a combinação ou *assemblage* de elementos complementares, interdependentes, inter-relacionados, interagentes que desempenham um conjunto de funções (ARNOLD; WADE, 2015) com propriedades, capacidades e comportamentos que procedem tanto dos elementos quanto das suas interações (HITCHINS, 2003).

A continuação fornecemos uma definição formal e simples de um sistema adaptativo de maneira geral.

2.2.1 Definição formal

Seguindo a estrutura de descrição formal para um sistemas proposta em Schmeck *et al.* (2010) podemos adaptar as seguintes descrições para a definição de um sistema S geral:

- Seja S o sistema sob observação que transforma uma determinada entrada $\overrightarrow{x}$ em uma saída $\overrightarrow{y}$. O sistema S é assumido como sendo um sistema estruturado, com vários elementos e enlaces que interconectam esses elementos.

- É entendido como estado ou configuração do sistema todas as informações correspondentes ao sistema S, as entradas e saídas dele, e outras informações do meio no

qual se desenvolvem as ações, ao qual denominamos ambiente.

- A qualquer momento t o sistema S se encontra em um estado $\overrightarrow{z}(t)$, onde $\overrightarrow{z}(t)$ é um vetor n-dimensional sendo "n" o número de atributos que descrevem S. O espaço de estados do sistema S é denominado Z.

- O desempenho do sistema é avaliado segundo algum critério de avaliação ou objetivos $\eta_1, .., \eta_k$ mapeando os estados do sistema para valores numéricos que representam a distância entre o estado atual e o objetivo a ser atingido. No caso ideal, a distância do estado do sistema ao objetivo é zero (o objetivo é atingido). Denominamos o conjunto de estados ideais do sistema (espaço alvo) como Z_Ω. O critério de avaliação pode envolver tanto operações simples quanto operações bastante complexas.

- Existem mecanismos de controle CM, tanto externos quanto internos ao sistema, que permitem controlar o seu comportamento através do monitoramento dos valores de alguns dos atributos do sistema ou do ambiente. O mecanismo de controle avalia a situação do sistema no momento corrente e, se necessário e disponível, inicia as ações estabelecidas anteriormente pelos desenvolvedores para gerenciar o evento correspondente.

- O estado do sistema $\overrightarrow{z}(t)$ é dito aceitável quando ele satisfaz algum critério de aceitação ou está dentro dos limites aceitáveis de desempenho. O conjunto de estados que atendem a tais caraterísticas é denominado espaço de aceitação.

- O sistema S pode ser objeto de perturbações ou estímulos provocados por influências, internas ou externas. Uma perturbação δ muda o estado do sistema de $\overrightarrow{z}(t)$ para algum estado $\delta(\overrightarrow{z}(t))$ e consequentemente podem mudar também os parâmetros de desempenho do sistema.

- O espaço de sobrevivência é o conjunto máximo de estados que o sistema S pode assumir temporariamente de forma tal que exista uma sequência de ações de controle que direcionem o sistema de volta para o seu espaço de aceitação. Um sistema que sai do espaço de aceitação pode por isso ser permanentemente danificado.

- O conjunto de estados do sistema, não incluídos no espaço de sobrevivência, é denominado espaço morto, enfatizando que o mecanismo de controle não consegue garantir a restauração de um comportamento aceitável para o sistema S.

A Figura 8 mostra uma representação simplificada do funcionamento de um sistema dentro do espaço de estados. O sistema S sempre tentará alcançar o espaço-alvo, ou pelo menos um estado pertencente ao espaço de aceitação, de maneira que nenhuma ação de controle externa explícita seja necessária. Se alguma perturbação δ muda o estado do sistema $\overrightarrow{z}(t)$ para algum $\delta(\overrightarrow{z}(t))$ fora do espaço de aceitação, o mecanismo de controle

deverá intervir para reconduzir S para algum estado de aceitação. Essa correção, ajuste ou adaptação só é possível se o estado em questão $\delta(\vec{z}(t))$ estiver no espaço de sobrevivência.

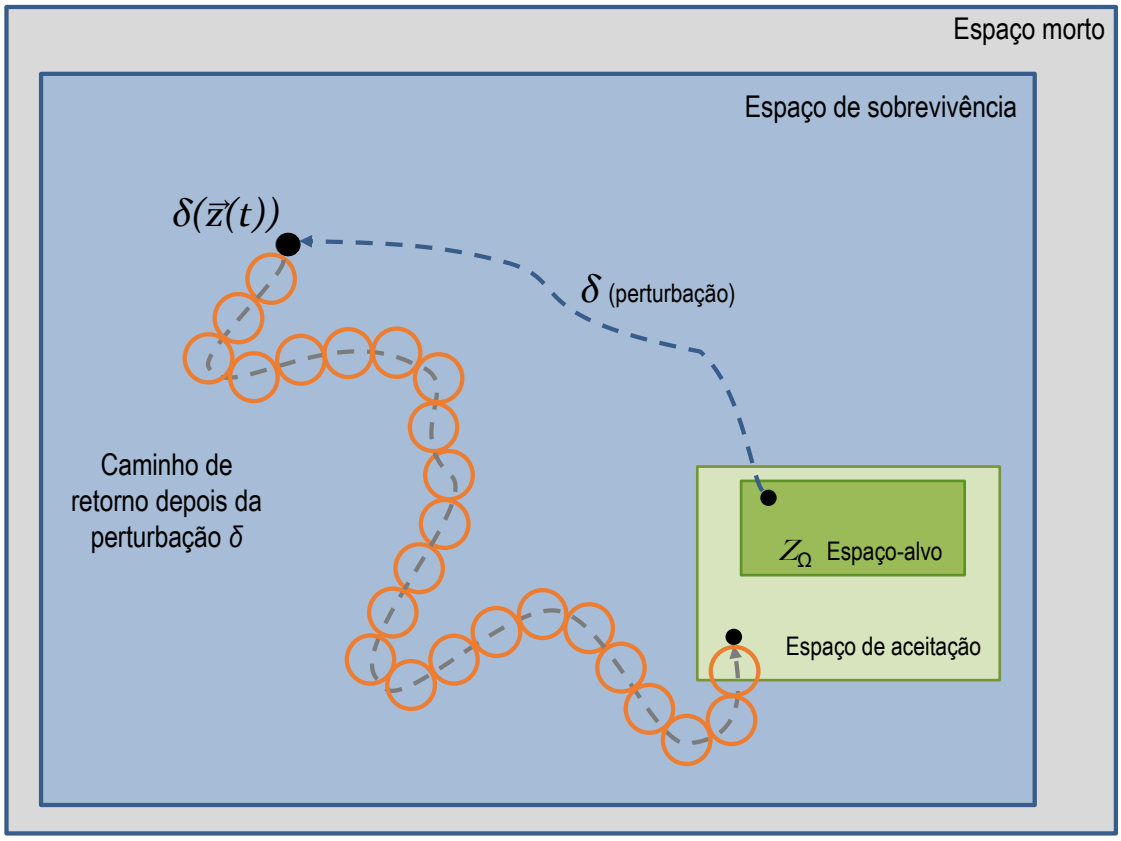


Figura 8 – Espaço de estados do sistema.
Adaptado de Schmeck *et al.* (2010).

Com essas informações prévias podemos definir que, sendo D um conjunto não vazio de perturbações ou estímulos percebidos por S:

- Um sistema S é dito adaptativo em relação a um conjunto de perturbações D, se e só se sem o auxílio de qualquer controle externo, ele for capaz de se movimentar dentro do espaço de aceitação depois de quaisquer das perturbações $\delta \in D$ por meio da execução de uma série de ações de controle interno sempre que S atingir algum estado fora dos estados de aceitação.

Em outras palavras, se δ leva S do estado $\vec{z}(t)$ ao estado $\vec{z'}(t) = \delta(\vec{z}(t))$, então, depois de um intervalo de tempo $\Delta t \geq 0$, no qual são aplicadas de uma série de ações corretivas o estado $\vec{z}(t+\Delta t)$ de S será aceitável.

Estas informações também nos permitem distinguir de maneira específica um sistema adaptativo de um sistema adaptável observando a origem das ações de controle e sua subsequente execução. Assim dizemos que:

- Um sistema é com relação ao conjunto D, se e só se ações de controle explícitos (externos) forem possíveis e necessárias para que S volte a movimentar-se no espaço de aceitação. O comportamento de um sistema adaptável S pode ser externamente modificado com o auxílio de *entradas de controle* $\overrightarrow{c}$ através de: (1) mudança nos valores dos parâmetros de S e/ou, (2) mudança na estrutura (elementos e relações).

Em consequência, a maior diferença entre um sistema adaptável e um sistema adaptativo está na fonte geradora dos estímulos que provocam a execução das ações de controle. No primeiro caso, a fonte é externa ao sistema em observação, i.e., o usuário, ou outro sistema. No segundo caso, a fonte que provoca a execução das ações de controle será o mecanismo de controle interno do próprio sistema em observação.

Uma análise mais detalhada sobre as características que definem os sistemas com comportamentos adaptativos é feita no Capítulo 6.

2.3 Fábricas e Linhas de produção de software

Por quase três décadas, a indústria de software atendeu às demandas de uma sociedade cada vez mais automatizada, aprimorando as habilidades de desenvolvedores individuais. No entanto, este enfoque de produção de software atualmente já encontra dificuldades para atender a demanda da sociedade da informação, avassaladora para esses desenvolvedores (GREENFIELD; SHORT, 2003). Atualmente, a capacidade desta mega indústria se suporta no trabalho manual, limitado por seus métodos e ferramentas e pela mão de obra qualificada disponível. De fato, segundo Greenfield (2004) a tendência nas próximas décadas é que a demanda de software se incremente em uma ordem de magnitude como resultado da influência de novas forças: economia global, crescente função do software na infraestrutura socioeconômica, novos tipos de aplicações e novas plataformas tecnológicas. Para atender essa demanda é necessário que o paradigma para desenvolvimento de software mude assim como seus métodos e práticas.

As fábricas de software são uma estratégia para mitigar o problema propondo o reaproveitamento do conhecimento ganho e dos ativos produzidos durante o desenvolvimento tradicional de software. Esse reaproveitamento permite melhorar a produtividade e eficiência do processo de desenvolvimento ao reutilizar recursos em aplicações semelhantes (ABBAS; ANDERSSON; LÖWE, 2011). Segundo MICROSOFT DOCS (2005) as fábricas de software assistem os arquitetos e desenvolvedores em criar, de maneira eficiente e preditiva, instâncias de alta qualidade de um tipo específico de aplicações que compartilham a mesma arquitetura e o mesmo conjunto de características.

Esta estratégia de reaproveitamento de recursos não é nova, e fora da Engenharia de Software a ideia de criar linhas de produção assistidas e reutilizáveis tem beneficiado o

crescimento das indústrias, sendo a linha de produção automotiva um dos seus exemplos mais conhecidos. Claramente o desenvolvimento de software se distingue da manufatura de produtos, pois é uma disciplina inerentemente orientada a pessoas e não pode ser reduzida a processos puramente mecânicos e determinísticos (GREENFIELD, 2004). No entanto, a maioria dos conceitos das fábricas e linhas de produção têm sido aplicados em projetos bem sucedidos de desenvolvimento de software desde os anos 60s (BRATMAN; COURT, 1975; CUSUMANO, 1989). Isso demonstra que as fábricas de software, se corretamente utilizadas, são um enfoque orientado a pessoas para a industrialização, que preserva as características próprias do domínio do problema e deixa os aspectos mais mecânicos e determinísticos aos cuidados das ferramentas de desenvolvimento.

Quando se mencionam as fábricas de software, existem duas possíveis abordagens analisadas em Greenfield e Short (2003): fábricas dirigidas à economia de escala e fábricas dirigidas à economia de escopo. Por um lado, as fábricas de software dirigidas à economia de escala procuram a produção de várias unidades do mesmo design, assim várias cópias de uma implementação são produzidas mecanicamente a partir de protótipos desenvolvidos pelos engenheiros. Por outro lado, as fábricas de software dirigidas à economia de escopo visam produzir vários designs e suas implementações básicas, isto é, os mesmos estilos, padrões e processos são utilizados para desenvolver vários designs relacionados, os quais são implementados utilizando um mesmo conjunto de ferramentas comuns para obter vários produtos únicos, mas semelhantes. Assim, a primeira abordagem está orientada ao mercado de massa, enquanto a segunda satisfaz um mercado personalizado. A Figura 9 mostra as diferenças entre essas duas abordagens. Neste trabalho adotamos a abordagem de economia de escopo para dirigir o processo de desenvolvimento da nossa proposta.

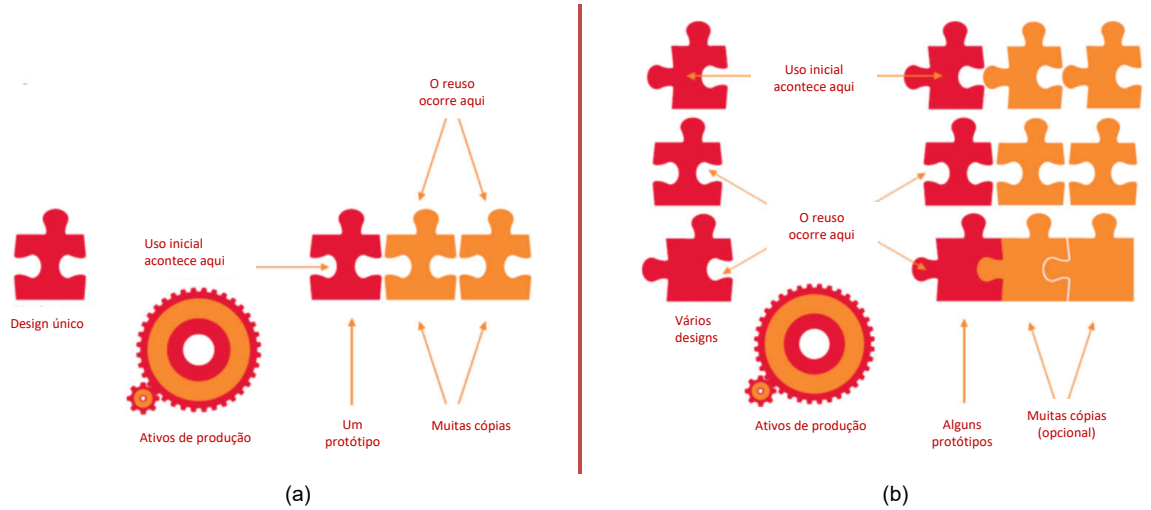


Figura 9 – Economias relacionadas à industrialização: (a) Economia de escala, (b) Economia de escopo.
Adaptado de Greenfield e Short (2003).

Dois conceitos tecnológicos importantes para o desenvolvimento das fábricas de software são: componentes e modelos. No primeiro caso, os componentes são a menor

unidade significativa sobre a qual a fábrica constrói seus produtos. Cada componente encapsula um conjunto de conhecimentos, os quais expõe à fábrica por interfaces padronizadas. Além das funcionalidades, o componente também expõe mecanismos que permitem diferentes níveis de ajustes, tais como personalização, adaptações e extensões. Esses ajustes são suportados por ferramentas próprias de cada cenário, as quais usam diversos recursos como: melhores práticas, padrões e até *frameworks* para domínios específicos. Dessa maneira, o conhecimento (*know-how*) de cada caso pode ser encapsulado e reutilizado para dar suporte ao desenvolvimento de produtos semelhantes. No segundo caso, os modelos definem o processo que a fábrica executa para implementar uma linha de produtos de software. As fábricas de software se apresentam como uma maneira de materializar as propostas do desenvolvimento dirigido por modelos[3], pois permitem processar as informações capturadas pelo modelo para automatizar tarefas de desenvolvimento.

Esta metodologia de desenvolvimento pode ser aplicada tanto no caso das soluções específicas (modelos particulares) quanto no caso da própria fábrica de software como apontado em Greenfield e Short (2003). No entanto, nem todos os padrões podem ser aplicados automaticamente, porque em vários casos, as equivalências entre padrões e modelos não podem ser definidas com antecedência. Em tais casos, os padrões devem ser aplicados manualmente pelos desenvolvedores, com frequência assistidos por ferramentas apropriadas.

Uma linha de desenvolvimento de produtos de software, também conhecida como *linha de produção de software*[4], é uma família de sistemas de software que compartilham um conjunto de características comuns, satisfazendo as necessidades específicas (requisitos) de um segmento de mercado ou missão particular sendo desenvolvidos desde um conjunto comum de ativos (recursos) principais de maneira prescrita (NORTHROP; CLEMENTS, 2012).

Assim, uma linha de produção de software está deliberadamente projetada para tirar proveito das características comuns e das variações mais conhecidas de um modelo geral de desenvolvimento de produto. Esse modelo é chamado *esquema da fábrica* e usa abstrações de alto nível como maneira de especificar as atividades e diretrizes da fábrica, assim como a arquitetura e relações entre os seus componentes. O esquema estabelece quais linguagens de domínio específico (ou DSL [5]) devem ser utilizados e como eles interagem com os modelos.

Os desenvolvedores de linha de produção criam ativos (recursos) de produção usados pelos desenvolvedores de produtos para obter instâncias particulares do modelo

[3] MDD-*Model-driven Development* em inglês
[4] SPL-*Software Product Line* em inglês
[5] DSL-*Domain Specific Languages*

inicial, os chamados produtos finais ou membros da família.
Alguns ativos de produção são:

- ativos de implementação, como arquiteturas e componentes, utilizados para a implementação dos casos específicos, e
- ativos de processos, como os processos que descrevem como utilizar os ativos de implementação, e ferramentas que automatizam partes do processo.

Nas fábricas de software, os ativos de processo são coletados em *templates* de software que fornecem todo o necessário para o funcionamento da fábrica. Esses *templates* devem ser carregados no ambiente de desenvolvimento (IDE) associado a fábrica antes de produzir qualquer produto, enquanto a maioria dos aspectos de implementação são encapsulados na própria fábrica. A Figura 10 apresenta uma visão geral da dinâmica de desenvolvimento de software usando uma linha de produção de software.

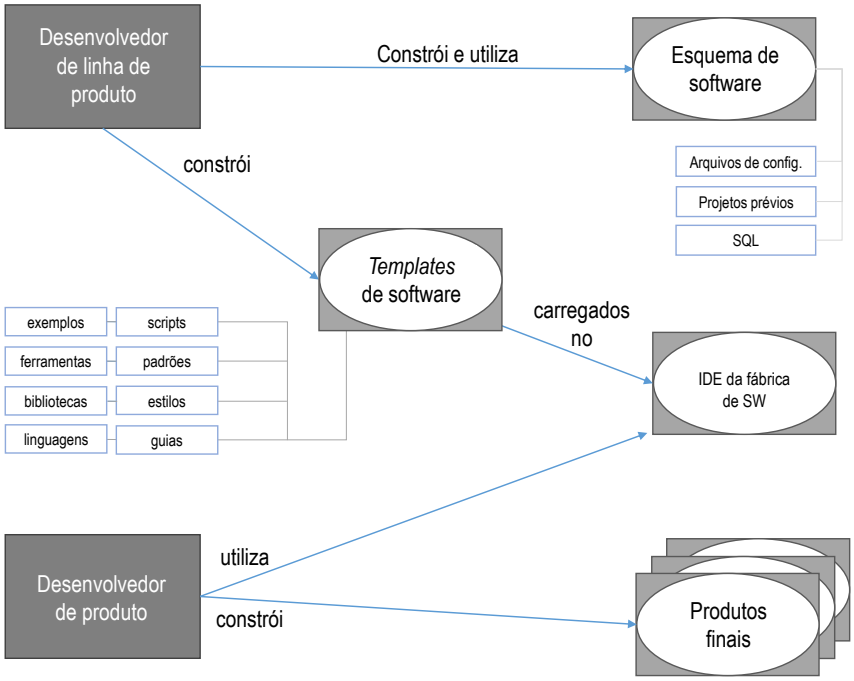


Figura 10 – Visão geral de uma fábrica de software.
Adaptada de Greenfield e Short (2003).

Assim, é possível entender uma fábrica de software como uma linha de produção automatizada por metadados capturados por modelos usando linguagens de modelagem específicos. Em outras palavras, uma fábrica de software constitui um sistema dirigido por modelos que combina o conhecimento do domínio do problema com o conhecimento da plataforma tecnológica, assim como a arquitetura da solução e o processo de desenvolvimento, com suporte fornecido pela documentação da plataforma e pelos desenvolvedores da fábrica.

A aplicação da metodologia de fábricas de software pode fornecer várias vantagens frente à metodologia tradicional, principalmente: consistência, qualidade, produtividade e extensibilidade, e de maneira geral o seu uso simplifica e automatiza as tarefas de rotina, permitindo que os desenvolvedores se concentrem em atividades de alto valor (MICROSOFT DOCS, 2005).

Nesta tese, visamos utilizar a abordagem de fábrica de software para automatizar o processo geral de desenvolvimento de mecanismos adaptativos. Esse processo se baseia nas condições necessárias e suficientes para a existência de comportamento adaptativo, assim como na funcionalidade representada no modelo holístico de adaptatividade. Assim, espera-se que para cada modelo de dispositivo adaptável que satisfaça as condições necessárias seja criado um mecanismo adaptativo específico.

2.4 Modelos e meta-modelos

De acordo com Bézivin (2005), um modelo é uma representação simplificada de uma determinada realidade para um propósito em particular. Assim, os modelos aprofundam nossa compreensão de realidades, também denominadas fenômenos ou sistemas, quer estejamos falando de um mecanismo, um robô, uma fábrica química, uma economia, um vírus, uma ecologia, um câncer ou um cérebro. Neste trabalho utilizaremos o termo sistema para fazer referência à realidade sendo modelada.

Rothenberg, no seu trabalho (ROTHENBERG, 1989) afirma que um modelo é uma abstração do sistema sendo modelado no sentido de que não pretende capturar todos os aspectos dessa realidade, mas principalmente abstrair apenas algumas dessas características, especificamente as características que servem o propósito que o modelo serve. Isso nos permite lidar com o mundo de forma mais simplificada, evitando a complexidade, o perigo e a irreversibilidade da realidade. Da mesma maneira, um modelo captura um certo aspecto ou visão do sistema que observa de maneira que uma mesma realidade pode ser representada por um conjunto de modelos diferentes, cada um capturando alguns aspectos específicos. A Figura 11 ilustra como os diferentes modelos podem representar uma mesma realidade.

Algumas características importantes ao falar sobre modelos são as seguintes:

- Um modelo se refere a um domínio específico do discurso e geralmente adere à suposição de mundo fechado (HENDERSON-SELLERS, 2011). Isso significa que qualquer coisa que não tenha sido especificada ou considerada pelo modelo é proibida e deve ser discriminada.
- Um modelo segue as regras de uma determinada linguagem de modelagem, assim, por exemplo, existem modelos textuais, modelos gráficos, modelos matemáticos, entre outros. Em particular, os modelos matemáticos têm sido de amplo interesse. Um

Figura 11 – Diferentes visões da realidade a modelar.
Adaptada de Bézivin (2004).

modelo matemático é uma descrição de um cenário do mundo real usando conceitos matemáticos e linguagem para ajudar a explicar um sistema e estudar os efeitos de diferentes componentes e fazer previsões sobre o comportamento. Talvez o exemplo mais antigo de um modelo matemático com grandes poderes preditivos tenha sido a lei da gravitação de Newton aplicada ao sistema solar. Em vez de modelar todo o sistema em toda a sua complexidade, Newton tratou o Sol e os planetas como pontos únicos. Isso lhe permitiu escrever as equações básicas do movimento de todo o sistema solar.

- O modelo é caracterizado pela substitutibilidade contextual: um modelo deve ser capaz de responder a um determinado conjunto de questões da mesma forma que o sistema observado responderia ao ser questionado. Assim, um modelo é uma analogia parcial da realidade observada (BÉZIVIN, 2004). Esta característica também é chamada de poder de representação do modelo.

- Um modelo representa um sistema e está conforme um meta-modelo.

Continuando, se um modelo é uma abstração de algum fenômeno real, um meta-modelo é uma abstração que consiste em afirmações e descreve propriedades do próprio modelo (JEUSFELD, 2009). Meta-modelo significa, literalmente, “além do modelo” afirmando que se trata de um conceito relacionado a um nível de abstração que transcende a modelagem. Assim, um meta-modelo também é um modelo, mas o seu domínio é o conjunto

de modelos que interessam ao criador do meta-modelo. Dessa maneira, a meta-modelagem é a análise, projeção, construção e desenvolvimento dos processos, estrutura, semântica, regras e restrições necessários para a criação de uma família de modelos associada a uma classe predefinida de problemas através de uma linguagem particular.

A Figura 12 apresenta a relação entre meta-modelo, modelo e sistema.

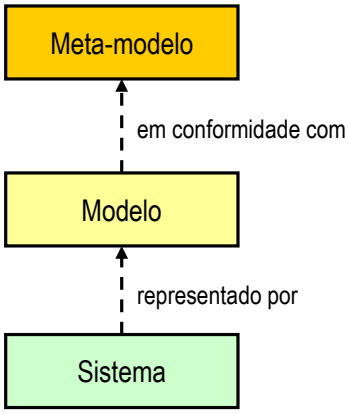


Figura 12 – Relação entre modelo e meta-modelo.
Adaptada de Bézivin (2004).

Assim como os modelos são abstrações de alguma realidade, e os meta-modelos são abstrações de modelos, é possível continuar aplicando a abstração, na teoria, e chegar nos meta-meta-modelos, que são modelos de meta-modelos contendo declarações sobre meta-modelos. A Figura 13 apresenta as arquitetura de meta-modelagem de quatro camadas da OMG (ATKINSON; KUHNE, 2003) no qual é possível observar que os meta-meta-modelos se descrevem a se mesmos, evitando levar a abstração *ad infinitum* com fins práticos.

Os meta-modelos são de vários tipos e têm diversas aplicações. Da mesma forma que os modelos, os meta-modelos podem utilizar várias linguagens, podendo ser textuais, gráficos/visuais, matemáticos, se apresentar como algoritmos que representam as relações de entrada e saída num sistema, entre outros. Alguns exemplos de tipos de meta-modelos incluem equações polinomiais, rede neural, Krigagem, entre outros.

O conceito pode ser útil em matemática com aplicações práticas em Ciências da Computação e Engenharia da Computação/Engenharia de Software. O trabalho em (FATEHAH; MEZHUYEV; AL-EMRAN, 2021) apresenta uma revisão sistemática que analisa o uso de meta-modelos na Engenharia de Software.

Adicionalmente, os meta-modelos permitem o desenvolvimento de diversas pesquisas e ferramentas que exploram a transformação entre modelos para fornecer flexibilidade e explorar novos conceitos (STAHL; VOELTER; CZARNECKI, 2006). A transformação de modelos permite, através da aplicação de uma série de passos, mapear os elementos

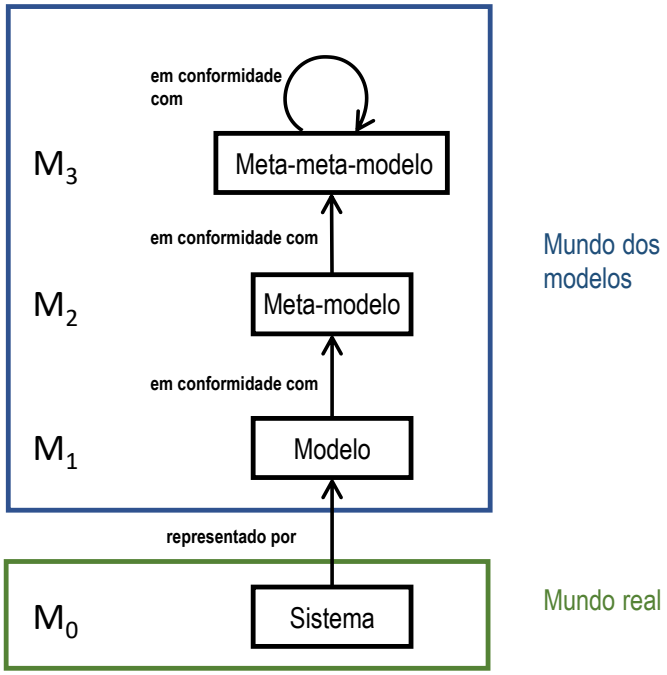


Figura 13 – Arquitetura de quatro-camadas da meta-modelagem.
Adaptada de Bézivin (2004) e Atkinson e Kuhne (2003).

de um modelo-origem em elementos de um modelo-alvo, respeitando as condições e relações correspondentes. O trabalho em (HENDERSON-SELLERS, 2011) apresenta uma abordagem interessante da relação entre meta-modelos e o campo das Ontologias.

3 METODOLOGIA DE PESQUISA:*DESIGN SCIENCE RESEARCH*

Como em todo trabalho de pesquisa acadêmica, o desenvolvimento desta tese precisou do suporte e rigor fornecidos por uma metodologia adequada para os fins do projeto, que dirigisse e acompanhasse o processo de pesquisa.

Ao consultar a literatura sobre as metodologias mais utilizadas na Engenharia da Computação e Ciências da Computação é possível observar que não existe um consenso sobre um grupo de metodologias próprias da área. Dado que a computação possui suas raízes nas ciências puras e na engenharia, é esperado que exista uma herança no uso e aplicação do conhecimento já estabelecido nessas áreas para o desenvolvimento de pesquisas (DEMEYER, 2011).

Em particular, o método científico, que apresenta uma abordagem post-positivista (BORREGO; DOUGLAS; AMELINK, 2009; HASSANI, 2017), é um legado frequentemente referenciado como a diretriz no desenvolvimento dos trabalhos de pesquisa. O método científico, como processo lógico, suporta perfeitamente o objetivo principal da pesquisa: generalização dos resultados obtidos de experimentos em uma determinada área de maneira que possa ser utilizado ou aplicado além da área específica de pesquisa (HASSANI, 2017). No entanto, o problema é que ao ser definido de maneira tão geral, nem sempre corresponde com a natureza da pesquisa, estabelecendo certas limitações e/ou desconsiderando algumas características do assunto e área de pesquisa (GAMUKAMA; POPOV, 2008). Nesse sentido, como coloca Demeyer (2011), vários autores reconhecem uma lacuna no estudo e desenvolvimento de metodologias das áreas da computação que forneçam uma base comum para a construção de uma identidade científica (HASSANI, 2017; DODIG-CRNKOVIC, 2002; DEMEYER, 2011). Essa lacuna é cada vez mais evidente devido às mudanças que o próprio avanço tecnológico trouxe para a área: estudo de problemas mais complexos, novas interfaces, mudanças de paradigma e até domínios de atuação novos ou mais abrangentes, produto das interações com outras áreas (DEMEYER, 2011; HASSANI, 2017).

Por um lado, essa referência categórica ao método científico como diretriz da pesquisa é a razão pela qual a maioria da literatura na computação está concentrada no estudo de métodos e não metodologias: é assumido que a metodologia subjacente corresponde ao método científico. Contudo, existem alguns trabalhos, como revisões da literatura, surveys e artigos, que fornecem maior informação sobre assuntos relacionados:

a) Os métodos mais utilizados nas áreas de computação: por exemplo, Glass, Vessey e Ramesh (2002) em Engenharia de Software; Ramesh, Glass e Ves-

sey (2004),Hassani (2017) e Amaral (2006) em Ciências da Computação; e Gamukama e Popov (2008). [1]

b) As abordagens, ou marco teórico, mais frequentes sobre os quais são desenvolvidas as pesquisas, por exemplo, Borrego, Douglas e Amelink (2009) e Gamukama e Popov (2008).

c) A análise de estilos de pesquisa para realizar uma categorização dos trabalhos na área, por exemplo, Wazlawick (2009).

Por outro lado, o reconhecimento dessa lacuna tem motivado esforços dentro da área para desenvolver propostas de metodologias próprias que consideram as diferentes condições nas quais as pesquisas em computação são realizadas, assim como os seus objetivos e interações. Alguns exemplos desses esforços são encontrados em Holz *et al.* (2006), e Jr, Chen e Purdin (1991).

Nesse contexto, a recomendação principal dos autores citados é estabelecer primeiro as características da pesquisa, o assunto de estudo, os objetivos do pesquisador, o contexto de desenvolvimento e as condições e recursos disponíveis para identificar tanto a abordagem quanto a metodologia, métodos, processos e técnicas que podem ser utilizados para atingir os objetivos.

Neste ponto, é necessário lembrar a terminologia básica participante na organização dos processos de pesquisa: abordagem, metodologia, método, processo e técnica. Uma **abordagem** de pesquisa (ou *research approach* em inglês) é a filosofia básica (conjunto de premissas lógicas, conceitos, valores ou proposições) ou crenças concernentes ao objeto de estudo (HOFLER, 1983; ANDIAPPAN; WAN, 2020). A abordagem determina como o conhecimento será estudado e interpretado, guiando o pensamento e a pesquisa no campo de estudo (MACKENZIE; KNIPE, 2006; ANDIAPPAN; WAN, 2020). Outros termos utilizados na literatura para se referir à abordagem são: paradigma de pesquisa (*research paradigm*) (MACKENZIE; KNIPE, 2006), perspectiva da pesquisa (*research perspective*) (VAISHNAVI; KUECHLER, 2015), marco teórico (*theoretical famework*) (MACKENZIE; KNIPE, 2006) e marco conceitual (*conceptual framework*) (WEBER, 2010). Algumas das abordagens mais conhecidas são apresentadas, de maneira resumida, em Mackenzie e Knipe (2006) e Weber (2010).

A **metodologia** pode ser entendida como um sistema de métodos, ou regras, utilizados em conjunto com una série de critérios para empreender uma pesquisa particular

[1] Em particular os trabalhos de Hassani (2017), Gamukama e Popov (2008) e Dodig-Crnkovic (2002) são interessantes pontos de referência, pois apresentam num primeiro momento uma análise do processo de pesquisa de maneira geral para depois aprofundar na pesquisa nas áreas de computação.

(MACKENZIE; KNIPE, 2006; ANDIAPPAN; WAN, 2020). Uma metodologia descreve a estratégia geral para a resolução do problema em questão, sendo assim a realização da abordagem definida previamente. Uma metodologia funciona como uma diretriz, permitindo ao pesquisador fazer suas escolhas num determinado conjunto de regras ou limites, e assim, seguindo os seus critérios, o método apropriado pode ser escolhido (ANDIAPPAN; WAN, 2020). Outras nomenclaturas para metodologia são: método de pesquisa (*research method*) (DRESCH; LACERDA; JÚNIOR, 2015), estrutura (*framework*) (ANDIAPPAN; WAN, 2020) e inclusive abordagem de pesquisa(*research approach*).

Um **método** é uma diretriz apresentada como um conjunto de passos predeterminados para resolver um problema definido (WEBER, 2010). Mais precisamente, um método é uma implementação formal de construções, por exemplo: modelos, e, simultaneamente, uma representação do espaço de solução da pesquisa (ANDIAPPAN; WAN, 2020). Assim, o método refere-se a modos sistemáticos, procedimentos ou ferramentas usadas na prática para coleta e análise de dados (MACKENZIE; KNIPE, 2006). Como apontado por Andiappan e Wan (2020) não é incomum encontrar confusões entre metodologias e métodos, porém a maior regra para distinguir um do outro é o rigor com o qual é aplicado: um método é aplicado sempre da mesma forma sem importar a situação, uma metodologia é uma diretriz que permite flexibilidade. Um método também pode ser chamado de modelo de processo (*process model*) (WEBER, 2010).

Um **procedimento** é a ordem estabelecida na qual uma sequência de técnicas ou ações são conduzidas (ANDIAPPAN; WAN, 2020). Cada método possui um procedimento que rege sua atividade. Finalmente, uma **técnica** é o passo dado que produz um resultado imediato, ela se refere à atividade específica praticada pelo pesquisador que pode se observada e medida. Uma técnica geralmente faz parte de um procedimento (ANDIAPPAN; WAN, 2020).

A Figura 14 apresenta graficamente a relação entre os termos usados no desenvolvimento da pesquisa.

Considerando as terminologias correspondentes são especificadas as particularidades da pesquisa:

a) A natureza da pesquisa é **interdisciplinar**: abordar o estudo do comportamento adaptativo exige uma abordagem de pesquisa que considere e incorpore, na medida do possível, o conhecimento em ouras áreas, pois o assunto permeia diferentes áreas do conhecimento, campos de atuação e inclusive abordagens.

b) Mistura de **contribuições conceituais e de desenvolvimento**: por causa dessa multidisciplinaridade é preciso revisar, analisar e até formular respostas no nível conceitual, mesmo pretendendo a realização de uma contribuição aplicada

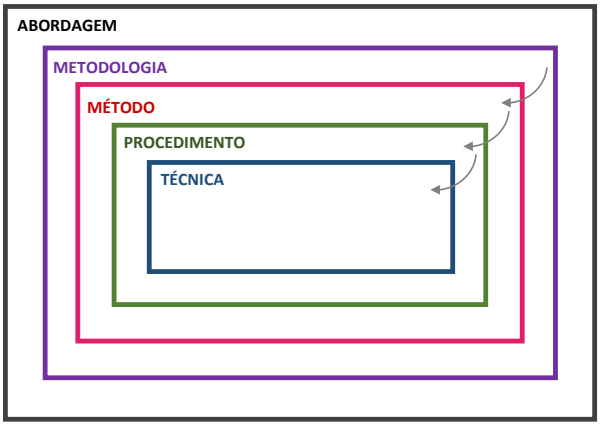


Figura 14 – Esquema conceitual da relação entre termos da pesquisa
Adaptada de (ANDIAPPAN; WAN, 2020)(2021).

na área de desenvolvimento de software.

c) **Flexibilidade**: a pesquisa é um processo de descobertas em diferentes aspectos, portanto é importante que a metodologia considere essa flexibilidade e novidade como parte explícita do projeto, sem perder rigor ou formalidade. Em vários pontos do projeto, os recursos adequados para conduzir as atividades de pesquisa não estão especificados, formalizados ou padronizados para o contexto de desenvolvimento de software, pelo que será necessário misturar e ajustar recursos e métodos de outras áreas. A metodologia deve permitir esta composição de recursos.

d) A contribuição deve ter **persistência no tempo**: os produtos da pesquisa, tanto a solução desenvolvida quanto o conhecimento ganho no processo, devem ser apresentados num formato que permita o seu uso, evolução, enriquecimento e complemento, por outros projetos.

e) O projeto é **individual**: em termos gerais, pode se descrever o equipo de pesquisa como composto de: um indivíduo, o pesquisador, com dedicação a tempo completo, com a assessoria de um consultor especializado, o orientador. Assim, a metodologia deve ser simples, os seus processos e derivados devem ser precisos e eficientes. Documentações ou sub-processos que não são aplicáveis para projetos individuais devem ser evitados, pois consomem recursos escassos nesses contextos, embora eles sejam muito úteis e apropriados para casos de projetos com equipes maiores.

Essas características, mesmo embebidas num contexto diferente, estão alinhadas com os princípios do *Design Thinking*. De fato, *Design Thinking* (DT) é considerado um framework para a resolução de problemas complexos e mal definidos (também conhecidos como *"wicked-problems"* ou *"ill-defined-problems"*) de uma maneira inovadora e criativa, com um processo progressivo que, de maneira co-evolutiva, trabalha tanto definição

do problema quanto em uma solução imerso num ambiente dinâmico. Assim, no caso específico deste trabalho identificamos que era necessário procurar uma abordagem e uma metodologia que abordassem a pesquisa desde essa perspectiva: a *Design Science* e a *Design Science Reseach*. Na seguinte seção apresentamos ambas iniciativas de maneira mais detalhada.

3.1 *Design Science* e *Design Science Research*

Design Science(DS) é um paradigma de resolução de problemas com suas origens no trabalho de Herbert Simon *"The sciences of artificial"* (SIMON, 1996) publicado originalmente em 1969 (HEVNER *et al.*, 2004). No livro, Simon apresenta a distinção entre os ambientes natural e artificial e introduz *"Science of Design"* que depois seria renomeada como *Design Science* (DRESCH; LACERDA; JÚNIOR, 2015).

Segundo Simon, as ciências tradicionais, ou ciências do "universo natural", se ocupam de desenvolver conhecimento sobre os fenômenos do mundo (suas características, como se comportam e como interagem) seja por meio de descobertas e/ou de análises de objetos existentes. Esse raciocínio pode ser aplicado tanto para fenômenos naturais (Biologia, Química, Física) quanto sociais (Economia, Sociologia). Nesse sentido, as pesquisas segundo o paradigma das ciências tradicionais pretendem explorar, descrever, explicar e, se possível, predizer acerca de fenômenos ou sistemas (ROMME, 2003; AKEN, 2004), em outras palavras, elas estão focadas em compreender porque as coisas funcionam da maneira em que funcionam (GEERTS, 2011). Diferente das ciências naturais, Simon argumenta que os esforços científicos no "universo artificial", ou nas "ciências do artificial", não se ocupam de estudar os fenômenos como eles são (realidade), mas como eles poderiam ser (design) (SIMON, 1996, p. xii). Em outras palavras, as ciências do artificial se preocupam com mudar as situações existentes para aquelas preferidas (SIMON, 1996, p. 130) através da concepção de *artifacts*[2] que alcancem objetivos definidos (SIMON, 1996, p. 198).

Esta diferenciação em focos de estudo, permite observar que embora o paradigma tradicional tenha sido amplamente adotado e aplicado, ele pode não ser um método adequado de pesquisa quando se trata de estudar o design, construção ou criação de novos *artifacts*, ou de conduzir pesquisas baseadas na resolução de problemas (DRESCH; LACERDA; JÚNIOR, 2015), pois os objetos em ambos os casos são não existentes.

A DS permite preencher esta lacuna e apresenta uma alternativa para reduzir o distanciamento entre a teoria e a prática (WIERINGA, 2009), entre a criatividade e o rigor científico. Assim, DS é um paradigma pragmático (BORREGO; DOUGLAS; AMELINK, 2009) e prescritivo (DRESCH; LACERDA; MIGUEL, 2015) que se ocupa do *design* e

[2] Simon descreve os *artifacts* como objetos artificiais, construídos pelo homem, que podem ser caracterizados em termos de objetivos, funções e adaptações (SIMON, 1996, p. 28).

desenvolvimento de *artifacts* inovadores para resolver problemas reais e relevantes (SIMON, 1996; ROMME, 2003; AKEN, 2004). Tais *artifacts* definem e descrevem as ideias, práticas, capacidades técnicas e produtos por meio dos quais a análise, design, implementação, gestão e uso de sistemas de informação podem ser realizados de forma eficaz e eficiente (HEVNER *et al.*, 2004). Outra característica importante da DS é que suas contribuições são medidas tanto na utilidade dos *artifacts* para a resolução de problemas, quanto na sua participação para agregar valor ao conhecimento teórico existente para pesquisas e práticas futuras (HEVNER *et al.*, 2004). Assim, a DS é uma ciência de método, rigor, complexidade e criatividade que permite considerar um sistema como sendo mais do que a suma das suas partes, superando o reducionismo e permitindo estudar problemas complicados e sistemas complexos, integrando diversas abordagens. De fato, o DS possui uma latente vocação interdisciplinar que permite a convergência de diferentes disciplinas.

Para mais informações sobre o contexto histórico, evolução e diferenciação do Design Science e o paradigma tradicional de pesquisa, o leitor pode se referir a Lacerda *et al.* (2013), Dresch, Lacerda e Júnior (2015), Rodrigues (2018), Hevner e Chatterjee (2010).

Enquanto a DS é a base epistemológica, o *Design Science Research* (DSR) é uma metodologia de DS para pesquisa (DRESCH; LACERDA; JÚNIOR, 2015; ALTURKI; BANDARA; GABLE, 2012; MACKENZIE; KNIPE, 2006; WEBER, 2010; CARNEIRO; ALMEIDA, 2019). A DSR estabelece e operacionaliza a construção do conhecimento quando o objetivo da pesquisa é o desenvolvimento de um *artifact* ou uma recomendação (DRESCH; LACERDA; JÚNIOR, 2015). Ela pode unir a capacidade da ciência de entender "o que é" e a do design de entender "o que pode ser" (HEVNER; CHATTERJEE, 2010), auxiliando o pesquisador na geração de conhecimento teórico durante processos de concepção e avaliação de *artifacts* para a resolução de problemas relevantes, aplicando o rigor necessário para a constituição de pesquisas de caráter científico.

No contexto geral da DSR, ao falar em desenvolvimento de *artifacts* não se faz referência única e exclusivamente ao desenvolvimento de produtos. Formalmente os *artifacts* desenvolvidos e avaliados pela DSR podem ser de diferentes tipos: Construtos, Modelos, Métodos e Instanciações (MARCH; SMITH, 1995) os quais podem implicar contribuições para o avanço teórico das áreas participantes (HEVNER; CHATTERJEE, 2010).Assim, como indicado por (PEFFERS *et al.*, 2007) a DSR se torna uma metodologia importante para a condução de pesquisas aplicadas por um processo rigoroso e adequado, pois permite a formalização do conhecimento gerado, a justificação das suas decisões e a generalização das suas soluções.

Duas das características importantes das pesquisas que seguem a DSR são mencionadas na definição anterior: relevância e novidade (GEERTS, 2011). Por um lado, a relevância se refere à escolha do problema a ser resolvido, não deve ser de interesse exclusivo da área acadêmica, mas ter um impacto significativo na prática. A DSR aborda com frequência os chamados *wicked-problems* caracterizados por apresentar: complexidade, mudança dinâmica, flexibilidade inerente (HEVNER *et al.*, 2004), co-evolução entre espaço de problema e espaço de solução(feedback), e perspectiva multidisciplinar. Por outro lado, a novidade se refere a que a DSR deve se preocupar em dar soluções a problemas não resolvidos anteriormente de maneiras única e inovadoras, ou fornecer soluções a problemas já resolvidos que sejam mais efetivas ou eficientes (HEVNER *et al.*, 2004). Assim, tanto a relevância do problema quanto o rigor ajudam a diferenciar entre pesquisas que seguem a DSR e práticas puramente de *design* e, ao mesmo tempo fortalezem os fundamentos da DSR.

Adicionalmente, a DSR se identifica como uma metodologia de integração (BORREGO; DOUGLAS; AMELINK, 2009; MACKENZIE; KNIPE, 2006) caracterizada pelo seu foco integrador, pois permite e até incentiva a complementação da metodologia com elementos de outras abordagens metodológicas, especialmente para assistir com o gerenciamento de atividades tanto de natureza teórica quanto prática. Alguns exemplos desse tipo de integração são os trabalhos com métodos ágeis (CONBOY; GLEASURE; CULLINA, 2015), estudo de caso, pesquisa-ação (DRESCH; LACERDA; MIGUEL, 2015), e modelagem de negócios (modelo canvas). Assim, a DSR consegue agir como guarda-chuva metodológico para unir os campos do conhecimento em prol de objetivos comuns, utilizando o melhor que cada um deles pode oferecer para o alcance de tais objetivos. Diferentes autores abordam de maneira mais detalhada as características da DRS, suas contribuições ao conhecimento e como as pesquisas sob esta metodologia devem ser conduzidas e avaliadas. Alguns desses trabalhos são: Hevner *et al.* (2004), Hevner e Chatterjee (2010), March e Smith (1995), Peffers *et al.* (2007), Vaishnavi e Kuechler (2015), Jain, Sinha e Vitharana (2011).

Mesmo quando as atividades de *design* são próprias de muitas profissões, desde o seu surgimento a DSR manifesta uma proximidade particular com as engenharias e as ciências da informação. Tal relação é observável através da considerável literatura no assunto (HEVNER *et al.*, 2004) nessas áreas. Segundo Hevner *et al.* (2004) tal proximidade pode ser atribuída a que os avanços tecnológicos são o resultado do processo criativo e inovador do DS, enquanto (MARCH; SMITH, 1995) afirma que a própria natureza da DS é centrada na tecnologia, formando assim um ciclo de coevolução. Assim, a DSR passou a ser amplamente adotada em projetos da Engenharia e das Ciências da Computação no início dos anos 90s e desde então diferentes autores tem desenvolvido propostas para a condução de pesquisas segundo essa metodologia. Em Dresch, Lacerda e Júnior (2015)

é apresentada uma revisão histórica detalhada dos autores mais importantes em *Design Science* e dos autores que propuseram formalizações para a condução de pesquisas sob DS. A Figura 15 apresenta uma linha do tempo com as informações indicadas em Dresch, Lacerda e Júnior (2015) e atualizadas com os trabalhos de Weber (2010), Alturki, Bandara e Gable (2012), Carneiro e Almeida (2019).

Apesar desses avanços, a falta de um padrão geral para a condução da DSR na área de Sistemas de Informação era uma preocupação na comunidade, como apontado por Geerts (2011). O trabalho de (PEFFERS *et al.*, 2007) propõe uma solução ao problema com o desenvolvimento do *Design Science Research Process Model* (DSRPM também chamado de DSRM na literatura) como um método (WEBER, 2010) que considera as contribuições dos trabalhos anteriores e é consistente com a filosofia de DS nas diferentes áreas nas quais é aplicada. O DSRM contribui para a pesquisa em DSR ao fornecer uma estrutura comumente aceita para realizar pesquisas com DS e um modelo mental para sua apresentação, ajudando no seu reconhecimento e legitimação das ideias através da preservação de conceitos comuns com outras áreas.

3.2 *Design Science Research Process Model*

Este trabalho segue a metodologia DSR aplicando o método *Design Science Research Process Model* (DSRPM) proposto por Peffers *et al.* (2007), com algumas adaptações para as características da nossa pesquisa. O DSRPM foi selecionado porque suporta as características da pesquisa da seguinte maneira:

a) viabiliza a composição com outros métodos para o gerenciamento das suas atividades, facilitando a transmissão de conhecimento entre diferentes disciplinas para obter um objetivo comum;

b) considera contribuições acadêmicas e aplicadas;

c) permite diferentes pontos de começo e aplicação de diferentes métodos e técnicas fornecendo flexibilidade;

d) as atividades podem ser detalhadas em diferentes níveis, permitindo que ela seja adequada tanto para pesquisa individual quanto coletiva; e

e) está bem estabelecido na comunidade de engenharia de software (CONBOY; GLEASURE; CULLINA, 2015).

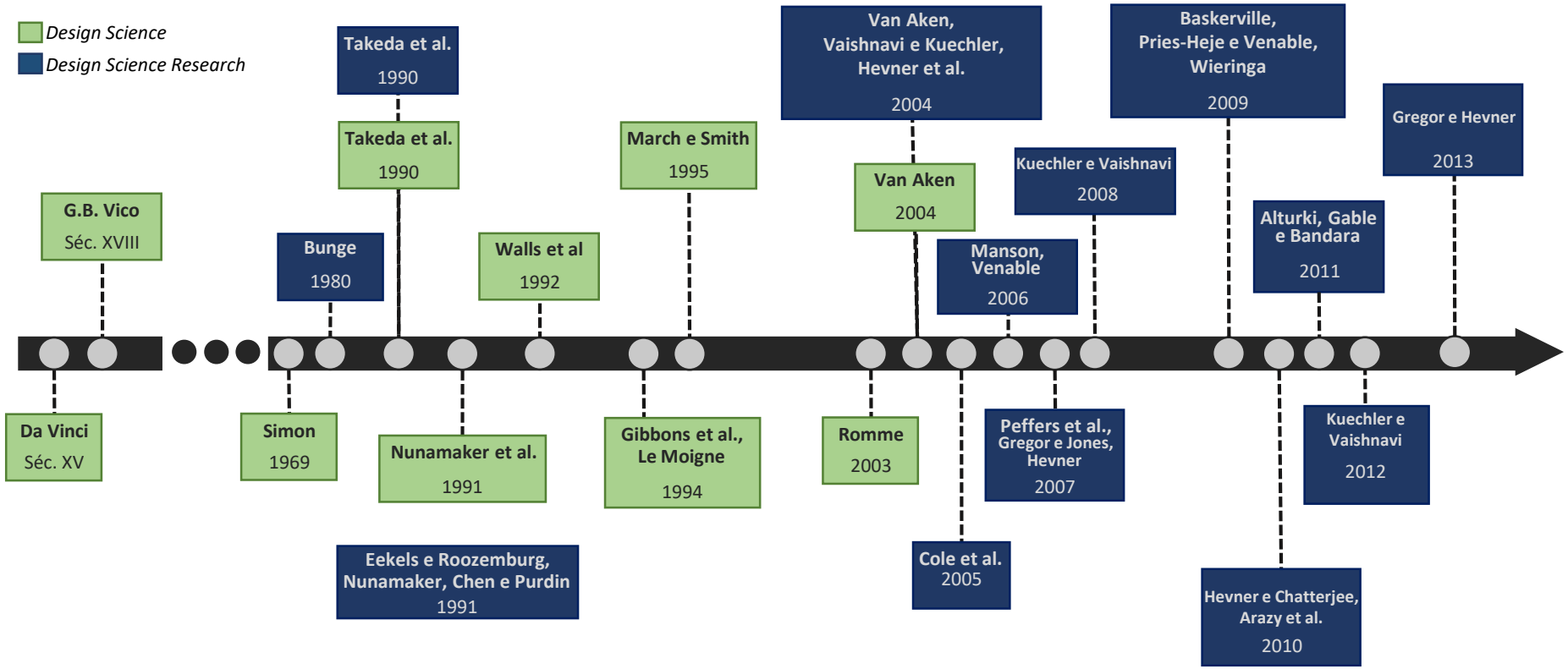


Figura 15 – Linha do tempo dos trabalhos mais importantes em *Design Science* e *Design Science Research*
Adaptada de (DRESCH; LACERDA; JÚNIOR, 2015)(2021)

Assim, o esquema apresentado na Figura 14 é particularizado para o caso deste projeto de pesquisa como indica a Figura 16.

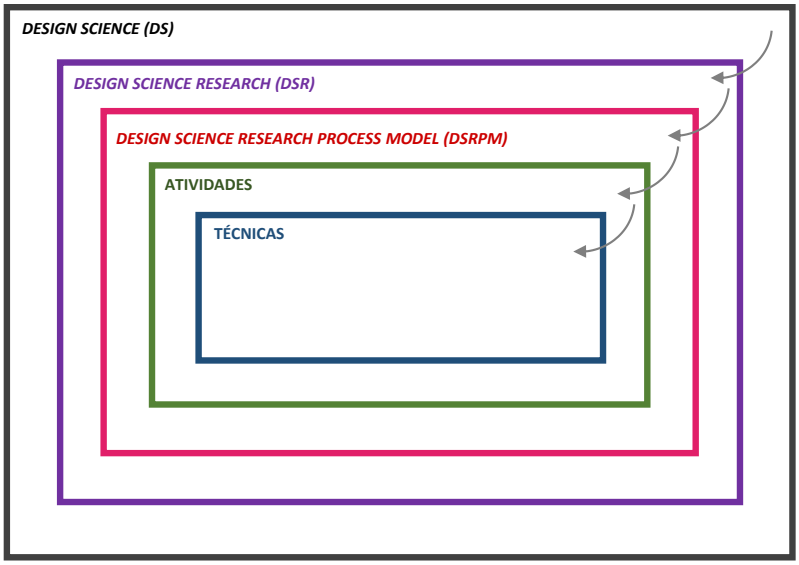


Figura 16 – Esquema conceitual da relação entre termos para este projeto de pesquisa
O Autor (2020)

O DSRM é descrito em Peffers *et al.* (2007) como uma sequência nominal de seis atividades: (i) identificação do problema e motivação, (ii) definição de objetivos de uma solução, (iii) design e desenvolvimento, (iv) demonstração, (v) avaliação, e (vi) comunicação.

A seguir descrevemos cada uma dessas atividades com detalhe, assim como a sua aplicação para o caso deste projeto.

(i) **Identificação do problema e motivação**: esta atividade está relacionada à compreensão da problemática envolvida, ela define o problema de pesquisa específico e justifica a sua relevância (motivação). Para a identificação do problema de pesquisa é indispensável um amplo conhecimento sobre o assunto geral no qual se enquadra a pesquisa. O principal resultado desta atividade é a definição e a formalização do problema a ser solucionado, e a delimitação das suas fronteiras.

Neste trabalho, este conhecimento está representado tanto na revisão da literatura quanto na análise de trabalhos relacionados. Assim, o Capítulo 4 (Estado da Arte e Trabalhos Relacionados) identifica e define o problema analisando o estado da arte no desenvolvimento de ferramentas que auxiliem na transição de sistemas adaptáveis para sistemas adaptativos, colocando especial atenção à identificação de lacunas e desafios nas abordagens presentes. Para a realização dos objetivos desta atividade foram aplicados estudos sistemáticos: revisões da literatura e mapas, e indicadores

bibliométricos, a maneira de garantir a validade interna da pesquisa e evidenciar sua rastreabilidade através de protocolos de condução. Por um lado, os estudos permitem o registro e controle do processo de revisão enquanto permitem obter uma visão geral do campo de estudo e identificar literatura relevante. Por outro lado, os indicadores bibliométricos permitem analisar informações da produção de conhecimento da área e realizar observações dos meta-dados das pesquisas. Em ambos os casos esses recursos permitem definir diferentes níveis de granularidade e abrangência.

Como resultado desta atividade foi identificado um novo problema de natureza teórica: a conceituação geral do comportamento adaptativo que inicia um ciclo paralelo para sua resolução.

(ii) **Definição dos objetivos de uma solução**: são definidos os critérios de aceitação (ou resultados satisfatórios) que deve atingir uma solução proposta para o problema. Os objetivos podem ser tanto quantitativos quanto qualitativos (PEFFERS *et al.*, 2007), em ambos os casos a identificação do problema atua como fonte de conhecimento para o estabelecimento dos objetivos. Para o caso da nossa pesquisa, os objetivos são definidos segundo a natureza do problema no Capítulo 5. O Capítulo 5, na Seção 5.1, identifica as necessidades de uma nova abordagem para a conceituação do comportamento adaptativo, assim como especifica as características de dita abordagem. Na Seção 5.2 são apresentados os objetivos qualitativos que a proposta deve cobrir para abordar o problema identificado no contexto tecnológico: suporte na transição de sistemas adaptáveis para sistemas adaptativos.

(iii) **Design e desenvolvimento**: construção do *artifact* em si i.e. a solução proposta ao problema. É necessário esclarecer que ao mencionar o termo desenvolvimento não se está referindo única e exclusivamente ao desenvolvimento de produtos ou sistemas de software. Tais *artifacts* podem ter diferentes formas, por exemplo: "construções"(*constructions* em inglês), modelos, métodos, "instanciações"(*instantiations* em inglês) (HEVNER *et al.*, 2004), regras tecnológicas, e princípios de implementação (CONBOY; GLEASURE; CULLINA, 2015). Segundo Peffers *et al.* (2007) um *artifact* para o DSRM é "quaisquer objeto projetado no qual foi encapsulada uma contribuição da pesquisa". Assim, o principal resultado do Desenvolvimento é o *artifact* em estado funcional. Este trabalho de pesquisa apresenta dois *artifacts* independentes, porém trabalhando em conjunto na procura de uma sinergia: o meta-modelo holístico da adaptatividade, descrito na Seção 6.3, e a ferramenta tecnológica para o suporte da transformação de sistemas adaptáveis, descrita no Capítulo 7.

(iv) **Demonstração**: aplicabilidade do *artifact* para a resolução de uma instância adequada do problema definido previamente, seja via experimentação, simulação, estudo de caso, prova, ou algum outro meio (PEFFERS *et al.*, 2007). Esta etapa traduz os elementos reutilizáveis e abstratos do *artifact* para contextos operacionais específicos,

expondo não só que a solução pode ser aplicada, mas também como ela pode ser aplicada (CONBOY; GLEASURE; CULLINA, 2015). Para a resolução do problema prático este trabalho segue a abordagem de prova por prototipagem, descrevendo no Capítulo 7 todos os detalhes associados com a configuração e implementação da proposta realizada no Capítulo 5. No caso do problema teórico, a Seção 6.4 apresenta a aplicação do modelo e sua representatividade em diferentes áreas.

(v) **Avaliação**: é processo rigoroso de análise de desempenho do *artifact* para a verificação do seu comportamento, isto é o cumprimento dos objetivos estabelecidos na atividade (ii), para o qual foi projetado. Portanto, é necessário: a) explicitar clara e precisamente a configuração do *artifact* e o ambiente em que irá ser avaliado, b) explicitar como o artefato pode ser testado; e c) descrever os mecanismos e métricas que medem os resultados. Em Hevner *et al.* (2004) os autores propõem alguns métodos que podem ser utilizados para a avaliação dos artefatos gerados na *Design Science Research*. Esta tese aplica diversos métodos de avaliação: testes funcionais, análise estática e análise da arquitetura, experimento controlado, e argumento informado. O Capítulo 8 apresenta os detalhes dos testes aplicados e discute os resultados obtidos através dos testes.

(vi) **Comunicação**: se refere à disponibilização do conhecimento gerado e dos resultados obtidos para as audiências relevantes através de publicações científicas e/ou profissionais. Este documento é produto desta atividade, assim como as publicações decorrentes do trabalho de pesquisa apresentadas para diversos veículos.

A Figura 17 apresenta a aplicação da DSRPM para este trabalho de pesquisa, mapeando tanto as atividades como os capítulos correspondentes neste documento e algumas das suas saídas.

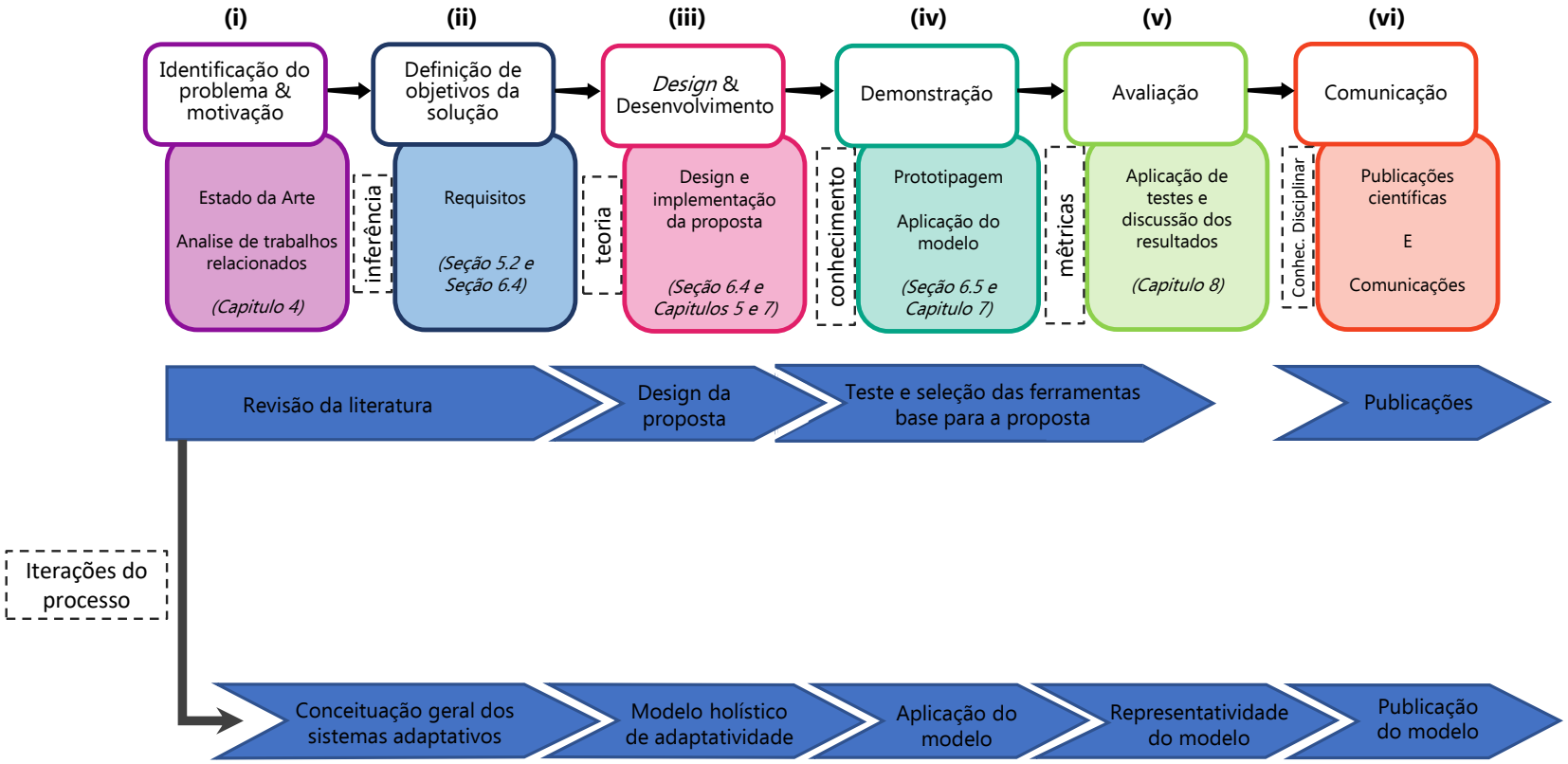


Figura 17 – Metodologia de pesquisa aplicada no projeto
O Autor (2020)

4 ESTADO DA ARTE E TRABALHOS RELACIONADOS

Como mencionado anteriormente, o objetivo deste trabalho é estudar os sistemas com comportamento adaptativo tanto no nível conceitual do fenômeno quanto no nível tecnológico através do desenvolvimento de sistemas computacionais que refletem suas propriedades.

Particularmente, no contexto do desenvolvimento tecnológico, os sistemas adaptativos acompanham as mudanças e necessidades que surgem na sociedade da informação. A medida que os avanços tecnológicos possibilitam o tratamento de problemas cada vez mais complexos, a capacidade de autogestão de sistemas tecnológicos não só é uma vantagem, mas também torna-se uma característica necessária. Assim, a implementação de sistemas com comportamento adaptativo parece uma linha de pesquisa natural e bastante atrativa para responder os desafios atuais. A revisão da literatura, apresentada na Figura 1 do Capítulo 1, mostra que o interesse de pesquisa em tópicos relacionados aos sistemas adaptativos tem longa data, com uma tendência crescente, desde 1991. Um olhar mais detalhado nos permite perceber que devido à sua natureza abrangente, praticamente todas as áreas de conhecimento podem estudar e se beneficiar das propriedades que a adaptatividade tem para oferecer.

A Figura 18 apresenta algumas das abordagens relacionadas ao estudo do comportamento adaptativo que encontramos ao longo do desenvolvimento deste trabalho. É importante observar que a figura não é uma representação fechada nem completa, pelo contrário ela tenta representar explicitamente a natureza aberta da pesquisa em adaptatividade, destacando uma zona cinza que reúne tanto outras abordagens já desenvolvidas que não foram diretamente identificadas neste trabalho como aquelas ainda por desenvolver. Da mesma maneira, em várias dessas abordagens o comportamento adaptativo é referenciado em diferentes graus ou indiretamente, sendo utilizados termos que descrevem suas propriedades como reconfiguração, mudança dinâmica, evolução, autogerenciamento, entre outros. Para representar o grau de proximidade entre o assunto de pesquisa e a adaptatividade, a Figura 18 utiliza elipses concêntricas com centro na adaptatividade, assim quanto mais perto do centro maior o grau de proximidade.

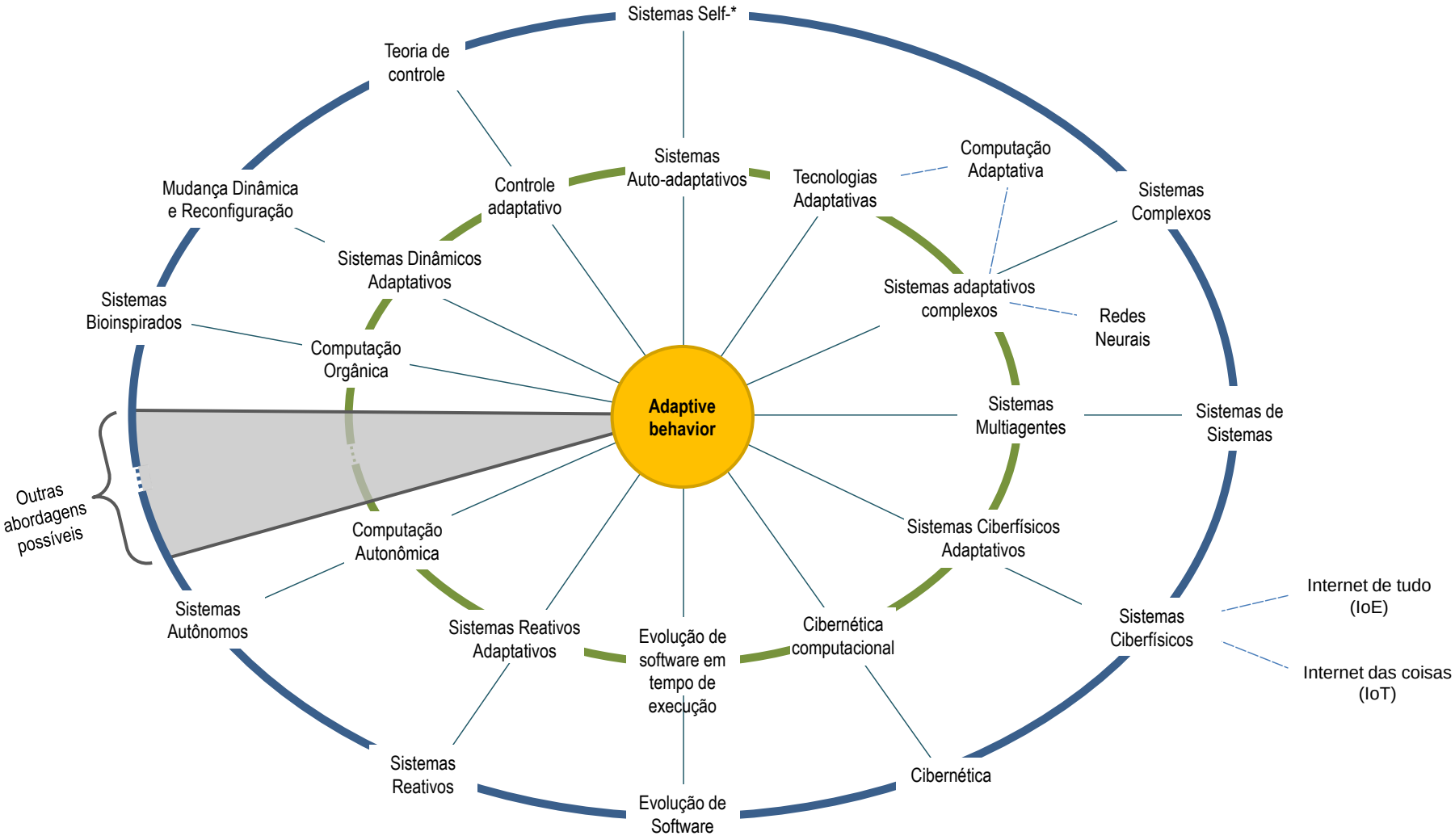


Figura 18 – Representação das diversas abordagens que estudam o comportamento adaptativo.
O Autor (2016).

As características relacionadas com adaptatividade permeiam diferentes assuntos de pesquisa em várias áreas do conhecimento, dentro das quais se geram escolas e abordagens para o estudo deste tipo de comportamento. Essa pluralidade de pesquisa tem duas consequências importantes que, basicamente, são dois lados de uma mesma moeda:

- A autonomia da pesquisa em cada campo, e
- A fragmentação do conhecimento gerado.

A seguir analisaremos ambas consequências com maior detalhe. Por um lado, a autonomia de pesquisa constitui uma vantagem importante para o desenvolvimento de novas abordagens, concretizando novas perspectivas que podem complementar o prisma científico do comportamento adaptativo. Através dessa autonomia, cada campo da ciência pode analisar a adaptatividade com base no seu próprio corpo de conhecimento, facilitando, assim, o desenvolvimento de recursos que permitem aproveitar da melhor maneira as propriedades que a adaptatividade oferece sem ter que sair da sua zona de conforto. Isso permite que os pesquisadores possam aprofundar seus estudos dentro das abordagens particulares com certa rapidez, e sem ter que recorrer a recursos externos ou comprometer suas ideias com outras abordagens. A Computação Autonômica, Computação Orgânica, Algoritmos Genéticos, Sistemas Multi-Agentes, Sistemas Complexos Adaptativos, Sistemas Dinâmicos e Modelos@RunTime são só algumas das abordagens que procuram analisar, modelar e aproveitar as propriedades do comportamento adaptativo nas áreas tecnológicas. As áreas humanas e biológicas apresentam a mesma diversidade de abordagens, por exemplo, a Cibernética e Autopoiese na filosofia, a Teoria de Adaptações na Psicologia, a Biologia Evolutiva, Mercados Adaptativos, Aprendizagem adaptativa, entre outras.

Por outro lado, essa mesma autonomia também gera a fragmentação do conhecimento produzido. Os avanços atingidos através dos estudos realizados numa abordagem particular dificilmente podem ser transmitidos para outras áreas, mesmo quando elas estudam o mesmo fenômeno: a adaptatividade. Isto revela que as abordagens se desenvolvem isoladamente, chegando a perder a relação conceitual entre as diferentes abordagens. Assim, por exemplo, os pesquisadores da área de Computação Evolutiva dificilmente imaginam uma relação com a Teoria de Autopoiese, de fato uma busca exploratória em Web of Science retorna 0 resultados. No entanto, ambas abordagens bio-inspiradas observam com interesse as propriedades do comportamento adaptativo. Essa situação é agravada pela profusão de terminologia em cada domínio que ofusca os conceitos que compõem os estudos.

Neste trabalho, focamos nossa análise no caso das pesquisas desenvolvidas nas áreas tecnológicas. Assim, é de especial interesse a criação e desenvolvimento de tecnologias como ferramentas para a resolução de problemas da vida real. Como mencionamos anteriormente, nessas áreas o interesse por incorporar características adaptativas nos sistemas

computacionais tem marcado tendência, com crescimento anual contínuo por mais de 50 anos (CAYA; NETO, 2018). No entanto, os pesquisadores e desenvolvedores de outras linhas de pesquisa encontram algumas dificuldades em adotar as abordagens desenvolvidas pelas linhas de pesquisa especializadas em adaptatividade, pois a maioria das vezes essas abordagens impõem algumas condições para sua aplicação:

(a) Compromisso com sua própria visão de adaptatividade, e

(b) Construção integral de entidades adaptativas.

Ambos pontos impõem um preço alto para os pesquisadores e desenvolvedores. Por um lado, o compromisso com uma visão particular de adaptatividade significa adotar sua base conceitual, de maneira que seus modelos, decisões de projeto, arquitetura e até paradigmas de programação podem exercer uma influência externa sobre o foco inicial da pesquisa que procurava aproveitar as vantagens da adaptatividade. Por outro lado, a construção integral de entidades adaptativas é uma tendência que identificamos ao analisar as diferentes abordagens que desenvolvem técnicas e recursos para o trabalho com adaptatividade. Segundo essa tendência, os sistemas só podem manifestar comportamentos adaptativos se eles foram projetados e criados para isso desde o começo. Se um pesquisador estivesse interessado em tornar um sistema X adaptativo, ele precisaria construir novamente o sistema X adotando alguma abordagem que permita a manifestação de adaptatividade. É claro que existem casos nos quais os sistemas legados foram desenvolvidos com uma abordagem fechada e rígida que não permite nenhum tipo de modificações ou reconfigurações. Nesses casos, de fato, a única opção para incorporar adaptatividade é reconstruir o sistema em questão. Porém, hoje em dia, a vasta maioria de sistemas em funcionamento permitem certas reconfigurações e admitem mudanças de certo nível, oferecendo um potencial que até hoje tem sido pouco considerado.

Nesse contexto, para tentar dar resposta à necessidade de desenvolvimento de sistemas adaptativos de maneira prática e inclusiva, primeiro é preciso resolver duas problemáticas:

Problema 1: A construção de um marco conceitual unificado para o estudo de adaptatividade que permita reunir as abordagens, técnicas e ferramentas disponíveis sobre adaptatividade; e

Problema 2: Facilitar a engenharia de sistemas adaptativos com base em sistemas legados.

Nas próximas seções exploramos o estado da arte para ambos os problemas e descrevemos as propostas de solução realizadas até o momento, identificando suas vantagens e desafios.

4.1 Panorama Atual

Para conhecer o estado atual dos estudos em relação com ambos os problemas identificados realizamos uma análise da literatura assistidos por dois recursos: (1) buscas em bases de dados científicas reconhecidas, como Web of Science e Scopus, e (2) consulta de revisões sistemáticas e surveys publicados. Em ambos os casos utilizamos tanto os recursos bibliométricos disponíveis de maneira livre como os recursos de pesquisa fornecidos através da filiação institucional com a Universidade de São Paulo, para processar as informações obtidas. Os detalhes dos dados obtidos em cada passo dessas revisões estão disponíveis no Repositório de Estudos Bibliográficos que acompanha este trabalho [1].

4.1.1 Desenvolvimento de um marco conceitual unificado

Realizamos uma busca geral utilizando a Base de pesquisa de *Web of Science* (WoS)[2] com duas cadeias de busca que procuram recuperar os trabalhos que desenvolveram um marco conceitual geral para a adaptatividade. Um resumo dos resultados obtidos por cada configuração se encontra na Figura 19. Visando realizar um levantamento mais abrangente, selecionamos os campos de busca como sendo "Tópico" no qual WoS inspeciona os campos de título, resumo e palavras-chave dos estudos para verificar as coincidências com os termos da cadeia de busca submetida. Com o mesmo propósito formulamos as buscas em inglês, pois é a maioria de publicações são realizadas nessa língua. Esse primeiro levantamento recuperou 166 estudos que cumprem com os termos da busca.

	Cadeia de busca	Base	Registros
#1	**(framework) and (general) and (adaptivity)** (Tópico)	WoS	166
#2	**(framework) and (general) and (adaptivity) and (model)** (Tópico)	WoS	90
#3	**#1 xor #2**	WoS	166

Figura 19 – Configurações das buscas realizadas nas bases de pesquisa.
O Autor (2022).

Depois, foram aplicados dois critérios de aceitação, detalhados na Figura 20, para filtrar os registros recuperados da busca no passo anterior. Assim, encontramos que dos 166 registros analisados só 5 pesquisas cumprem com os critérios estabelecidos, 7 estudam a adaptatividade de maneira geral, e 58 apresentam uma abordagem geral para o objeto de estudo, porém esse foco não é a adaptatividade. A Figura 21 apresenta esses resultados.

Como terceiro passo definimos os estudos que servirão como base da nossa revisão, os quais são tanto estudos obtidos do levantamento prévio quanto estudos selecionados

[1] https://sites.google.com/view/bibliographic-studies/home
[2] https://www.webofscience.com/

	Critérios Aplicados
C1	O assunto de estudo do trabalho é adaptatividade como fenômeno geral e não dentro de uma área ou campo específico.
C2	O estudo desenvolve e apresenta um framework conceitual geral para o assunto de estudo.

Figura 20 – Critérios aplicados aos resultados da busca.
O Autor (2022).

Registros analisados	Cumprem com C1	Cumprem com C2	Cumprem com C1 e C2
166	7	58	5

Figura 21 – Resultados da aplicação dos critérios nos resultados da busca.
O Autor (2022)

manualmente de uma base de pesquisa especializada. Os trabalhos com os quais continuaremos a revisão são apresentados e identificados na Tabela 1. Nela, os itens S1, S2, S3, S4 e S5 correspondem aos registros recuperados na busca na WoS e os registros identificados como ES1, ES2, ES3, ES4, ES5 e ES6 são os estudos extraídos da biblioteca especializada desenvolvida com este trabalho.

ID	Título	Ano
S1	AWaRE2-MM: A Meta-Model for Goal-Driven, Contract-Mediated, Team-Centric Autonomous Middleware Frameworks for Antifragility	2021
S2	The scenario coevolution paradigm: adaptive quality assurance for adaptive systems	2020
S3	Auxo: an architecture-centric framework supporting the online tuning of software adaptivity	2015
S4	What is adaptive about adaptive decision making? A parallel constraint satisfaction account	2014
S5	Harnessing the Power of Semantics-Based, Aspect-Oriented Adaptation for AMACONT	2009
ES1	Autopoiesis: The organization of living systems, its characterization and a model	1974
ES2	On machines, living and otherwise	1980
ES3	A framework for rule-based dynamic adaptation	2010
ES4	A conceptual framework for adaptation	2012
ES5	A white box perspective on behavioural adaptation	2015
ES6	The four types of self-adaptive systems: A metamodel	2018

Tabela 1 – Lista de estudos para análise.
O Autor (2022).

O trabalho em S1 propõe um meta-modelo para o desenvolvimento de sistemas antifrágeis através re recursos de autogerenciamento e autoaperfeiçoamento. O meta-modelo proposto está baseado em recursos dinâmicos fornecidos por middlewares e modelos em tempo de execução, e uma arquitetura orientada a agentes (MAS) e serviços (SOA). Os conceitos relacionados com o meta-modelo são apresentados em 4 categorias: estrutural (S), comportamental (B), intencional (I), ambiental (E) e transitória (T). Os autores também apresentam um exemplo da aplicação do meta-modelo para o caso de um middleware

autônomo anti-frágil baseado em agentes e serviços. A proposta em S1 cobre um domínio bastante específico, o qual é o de design de arquiteturas para middlewares autônomos para desenvolvimento de sistemas anti-frágeis usando modelos arquitetônicos, portanto, não apresenta uma resposta à busca de um marco conceitual genérico para o estudo de adaptatividade. Adicionalmente, as dependências de outros recursos tanto conceituais quanto de aplicação impõem restrições importantes para o desenvolvimento de sistemas, inclusive no próprio domínio escolhido.

Em S2 os autores apresentam um padrão de design para o desenvolvimento automático e simultâneo de software adaptativo e o sua suíte de teste correspondente usando técnicas de aprendizagem de máquina e um cenário antagônico que propicia a coevolução de ambos artefatos. A proposta de S2 é um modelo formal de adaptatividade para a construção de software adaptativo com base nas técnicas de aprendizagem de máquina e a construção de cenários para o aprimoramento gradual do sistema. Da mesma maneira, os autores elaboram a construção de uma suíte de software adaptativa que acompanhe as evoluções do sistema. O modelo é enunciado de maneira híbrida, são inicialmente definidos os conceitos básicos de adaptatividade de maneira geral e abrangente (o que é um sistema adaptativo?) para logo ser complementados com conceitos tecnológicos sobre o comportamento característico (como funciona um sistema adaptativo?). Assim, de maneira integral, o modelo fica definido para o contexto dos sistemas de software onde a adaptatividade será garantida por uma abordagem de aprendizagem de máquina aplicada num organização de agentes. É importante ressaltar que a noção de adaptatividade no caso de S2 implica unicamente incorporação de novas características e nunca remoção.

O trabalho em S3 apresenta Auxo, um framework centrado na arquitetura para o desenvolvimento e aprimoramento de software adaptativo em tempo de execução. A proposta está acompanhada de uma linguagem de descrição da arquitetura (AuxoADL), o processo de desenvolvimento de aplicações baseadas em Auxo e a infraestrutura em tempo de execução que suporta o framework. Assim, a proposta é integral, significando que o modelo, a arquitetura, e os recursos para desenvolvimento e melhora de sistemas adaptativos deve ser realizado dentro da abordagem descrita para garantir sua funcionalidade, limitando, portanto, sua reutilização em outros contextos.

Em S4 os autores apresentam uma comparação entre dois frameworks para tomada de decisão: os modelos de mecanismo único de tomada de decisão e os modelos multiestratégia. O trabalho apresenta 4 experimentos para a comparação de: escolhas, tempos de reação, e confiança de cada um dos frameworks. A análise realizada em cada experimento é profunda e detalhada, com suporte estatístico e critérios de avaliação interessantes. No entanto, o trabalho não realiza uma proposta de framework particular ou complementa alguma já desenvolvida para o estudo de comportamentos adaptativos.

Em S5 é apresentado o framework AMACONT que se baseia em formatos de

documentos baseados em componentes para a criação de aplicativos web adaptativos. O framework também apresenta características para separação de interesses ao desenvolver os aplicativos usando os conceitos de programação orientada a aspectos, e suporte para o uso de web semântica para flexibilizar o raciocínio sobre a base de adaptatividade. O framework está orientado ao desenvolvimento de tecnologias web, dependendo da arquitetura e padrões adotados nesse nicho.

Os estudos ES1 e ES2 correspondem aos trabalhos de Varela, Maturana e Uribe (1974) e Maturana e Varela (1980) nos quais os autores apresentam detalhadamente a teoria de autopoiese. A teoria autopoiética tem como ideia básica um sistema organizado autossuficiente. Este sistema produz e recicla seus próprios componentes diferenciando-se do meio exterior e sua autonomia é resultado de sua organização como sistemas em contínua autoprodução.

O ES3 é um trabalho desenvolvido por Lanese, Bucchiarone e Montesi em Lanese, Bucchiarone e Montesi (2010) que está focado no conceito de adaptação dinâmica. Nele, para a construção de um aplicativo adaptável é necessário que ele exponha atividades de domínio adaptáveis que receberão as mudanças. O aplicativo interage com um gerenciador de adaptatividade que contém as regras de adaptação para o aplicativo. Em particular, o ambiente tem controle total sobre as regras de adaptabilidade e pode alterá-las a qualquer momento, independentemente da configuração do aplicativo. As regras de adaptatividade contém a descrição da atividade a ser adaptada, a condição que deve ser cumprida para sua aplicação, o novo código-fonte para a atividade, as variáveis requeridas para a atividade e algumas informações não funcionais. Em tempo de execução, o gerenciador de adaptatividade verifica que a descrição da atividade e as condições sejam cumpridas, e utiliza um conjunto de políticas para ativar as mudanças se as propriedades não funcionais apresentam uma otimização. Um ponto interessante dessa proposta são algumas decisões de implementação que os autores descrevem, por exemplo, o tempo e o gatilho para as adaptações, a ordem de aplicação das regras e a necessidade de adaptatividade. Além do caso geral, em Lanese, Bucchiarone e Montesi (2010) os autores apresentam o framework JoRBA como prova de conceito da aplicação do modelo proposto em ambientes de Aplicações orientadas a serviços. De maneira particular, os autores enfatizam que a diferença principal do modelo proposto é que ele permite a especificação e atualização das regras de adaptatividade em tempo de execução e não espera que o sistema seja totalmente definido na fase de design. A proposta de Lanese, Bucchiarone e Montesi é formulada para sistemas computacionais e, preferentemente, aplicações e sistemas baseados em serviços, enquanto o marco conceitual de adaptatividade procura uma cobertura maior dos sistemas adaptativos em outras áreas do conhecimento, representando uma opção agnóstica de abordagem.

Os trabalhos em Bruni *et al.* (2012) e Bruni *et al.* (2015), correspondentes a ES4 e ES5 na tabela, apresentam uma abordagem de caixa branca para a análise do comportamento adaptativo nomeado CoDA que se baseia no conceito de Adaptação de Dados de Controle. CoDA define que dado um componente com uma coleção distinta de dados de controle, a adaptação é a modificação em tempo de execução de tais dados de controle (BRUNI *et al.*, 2015), e passa a responsabilidade de especificar os elementos que formam parte do comportamento adaptativo aos designers do sistema. Os autores enfatizam que CoDA é uma abordagem geral o suficiente para integrar várias das outras abordagens utilizadas no desenvolvimento de sistemas com comportamento adaptativo, e de fato esse é o caso para os sistemas computacionais.

Em Bruni *et al.* (2012) é apresentado um modelo formal para a abordagem sob a teoria de sistemas de transições rotulados onde são definidos os conceitos de adaptatividade, adaptabilidade, e algumas outras propriedades dos componentes de software que integram um sistema. Em Bruni *et al.* (2015) os autores apresentam várias comparações e observações importantes para os casos de sistemas adaptativos emergentes e coletivos, assim como a possibilidade de realizar uma torre de adaptatividade para maior hierarquia. Contudo, CoDA apresenta dois desafios: a aplicabilidade do modelo para descrever sistemas adaptativos fora das áreas tecnológicas, e a dificuldade para integrar sistemas adaptáveis no desenvolvimento de sistemas adaptativos. Em CoDA o comportamento adaptativo se suporta nos conceitos de teoria de controle, os quais são gerais e abrangentes para as áreas tecnológicas, porém apresentam um grau de dificuldade maior para sua compreensão em áreas fora desse contexto. Além disso, CoDA apresenta uma abordagem minimalista, sendo que quaisquer outra estrutura necessária para guiar o design de sistemas adaptativos deve ser elaborada e construída para cada sistema. Se bem essa estratégia abre inúmeras possibilidades, também dificulta uma aplicação rápida. Essa estrutura mais elaborada é necessária para facilitar a participação dos sistemas legados, que cumprem com as condições necessárias para se tornar adaptativos, no processo de desenvolvimento de sistemas adaptativos através da incorporação dos elementos faltantes para ganhar adaptatividade.

O estudo em ES6 corresponde ao um trabalho desenvolvido por Sabatucci, Seidita e Cossentino (2018) onde os autores propõem um meta-modelo que permita classificar os diferentes tipos de sistemas adaptativos baseados no nível de autonomia e de grau de adaptatividade que o sistema pode desenvolver. Com cada categoria os autores apresentam uma descrição dos elementos do sistema, o tipo de problema que ele poderia resolver e algumas aplicações conhecidas. No entanto, o trabalho está focado na tarefa de classificação dos sistemas e não no seu projeto ou implementação, por essa razão apresenta poucas informações sobre as características do comportamento adaptativo e a interação dos seus processos internos e externos.

Ao examinar os trabalhos selecionados na Figura 22 podemos observar que as tentativas realizadas até agora para a criação de um marco conceitual unificado para o estudo do comportamento adaptativo não fornecem uma proposta que seja, simultaneamente, clara e detalhada, agnóstico a abordagens e de fácil aplicação e compreensão, tanto no nível geral quanto na sua aplicação em contextos específicos. Portanto, é necessária uma abordagem que permita descrever de maneira completa tanto os elementos participantes quanto os processos que integram o comportamento adaptativo. Tal abordagem deve empregar uma linguagem geral e uma granularidade média das informações oferecidas que permita identificar facilmente o potencial de realização da adaptatividade presente em alguns sistemas. Assim, um modelo sob tal abordagem irá agir como uma ponte que interconecta diferentes áreas do conhecimento e permite a sinergia de estudos no que concerne o fenômeno do comportamento adaptativo.

ID	Abordagem geral?	Apresenta um framework?	Fornece suficiente detalhe?	Fácil de compreender?	Aplicações
S1	Sistemas de software antifrágeis	✓	✓	✗	Depende de conceitos de MAS, middleware e SOA.
S2	✓	✓	✓	✗	Depende de técnicas de aprendizagem de máquina.
S3	Sistemas de software	✓	✓	✗	Depende dos recursos criados para o framework.
S4	Sistemas de software	✗	✓	✗	Comparação entre modelos para tomada de decisão adaptativa.
S5	Sistemas de software	✓	✓	✓	Depende dos recursos do domínio de aplicação: Aplicativos web.
ES1	✓	✓	Muito	✗	Precisa ser especificado para o domínio do problema.
ES2					
ES3	Sistemas de software	✓	✓	✓	Domínio de sistemas em tempo de execução.
ES4	✓	✓	✗	✓	O framework precisa da criação de estruturas para ser aplicado.
ES5					
ES6	✓	✗	✗	✓	Classificação de sistemas adaptativos.

Figura 22 – Quadro comparativo dos estudos da revisão conceitual.
O Autor (2021).

ID	Tipo	Termo principal	Termo alternativo	Cadeia
Set1	População (adaptive)	adaptive	adaption	(adaptive OR adaptation)
		self-adaptive	self-managed	("self-adaptive" OR "self-managed" OR "self-managing" OR "self-configuring" OR dynamic OR autonomic)
			self-managing	
			self-configuring	
			dynamic	
			autonomic	
Set2	População (legacy)	legacy system	existing system	("legacy system" OR "existing systems" OR"pre-existing")
			pre-existing	
Set3	Intervenção	systematic mappings	systematic mapping	("systematic mappings" OR "systematic mapping" OR "mapping study" OR "mapping review" OR "systematic map" OR "slr review" OR "state of the art" OR survey)
			mapping study	
			mapping review	
			systematic map	
			SLR review	
			state of the art	
			survey	

Figura 23 – Configuração da Busca para a Revisão de Escopo.
O Autor (2021).

4.1.2 Desenvolvimento de ferramentas para assistir à construção de sistemas adaptativos

Como segundo ponto precisamos conhecer o estado da arte ao respeito de desenvolvimento de ferramentas que suportem a implementação de sistemas adaptativos. Em particular, nos interessa identificar as abordagens que explicitamente enriquecem sistemas legados, com potencial, e os transformam em sistemas adaptativos. Para atingir esse objetivo aplicamos a técnica de Mapeamento Sistemático da Literatura que trabalha analisando estudos secundários sobre o assunto, tais como surveys e revisões sistemáticas. Novamente, todas as informações e dados obtidos e recursos utilizados para conduzir este mapeamento sistemático se encontram disponíveis no Repositório de Estudos Bibliográficos que acompanha o este trabalho[3]. A continuação apresentamos alguns dos resultados obtidos.

Primeiro procuramos informações sobre estudos sistemáticos existentes que cobrem esse tópico, esta técnica é chamada de Revisão de Escopo ou *Scoping Review* e serve para verificar a necessidade de realizar um mapeamento sistemático no assunto. Realizamos uma busca em 4 bases de pesquisa: IEEE Xplorer, Scopus, WoS e ACM com as configurações indicadas na Figura 23 e obtivemos 295 estudos, dos quais 76 eram duplicados em diferentes bases, restando 219 estudos diferentes.

A distribuição dos resultados dessa primeira busca é apresentada na Figura 24.

Depois, processamos os estudos aplicando os critérios de inclusão e exclusão detalhados no Protocolo da Revisão de Escopo, disponível no repositório citado anteriormente,

[3] https://sites.google.com/view/bibliographic-studies/home

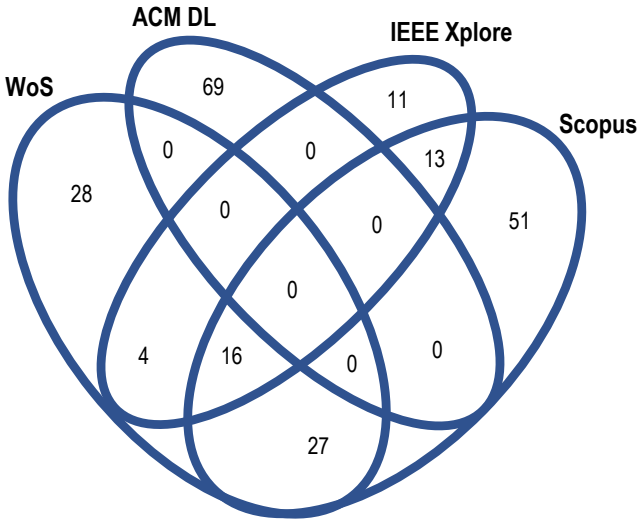


Figura 24 – Distribuição dos estudos da Revisão de Escopo por base de pesquisa.
O Autor (2021).

e resumidos na Figura 25.

ID	Descrição
IC1	O estudo apresenta resumos de abordagens de sistemas adaptativos, ferramentas ou frameworks
IC2	O estudo se desenvolve nas áreas de Ciência da Computação ou Engenharia.
IC3	O estudo é um estudo secundário.
EC1	O estudo apresentou material não revisado por pares.
EC2	A versão em texto completo do estudo não é acessível.
EC3	O veículo de publicação do estudo não é uma conferência, workshop, periódico ou revista.
EC4	O estudo foi publicado de forma incompleta, como artigos curtos, introduções a edições especiais, tutoriais e mini-trilha (menos de 4 páginas).
EC5	O estudo está relacionado a outro estudo do mesmo autor ou já foi selecionado por outra fonte.
EC6	O foco da pesquisa não são os Sistemas Adaptativos, a referência à adaptatividade é vaga.
EC7	O estudo não faz referência a Sistemas Legados.
EC8	O estudo está duplicado.
EC9	O estudo não é uma fonte secundária.
EC10	O estudo não pertence às áreas de Ciências da Computação ou Engenharia.

Critério de inclusão Critério de exclusão

Figura 25 – Critérios de inclusão e exclusão da Revisão de Escopo.
O Autor (2021).

Como resultado, encontramos que não existe nenhum estudo do tipo mapeamento sistemático que agregue e analise as informações na literatura sobre a participação de sistemas legados na construção de sistemas adaptativos. Assim, temos a base bibliográfica que justifica a elaboração de um mapeamento sistemático para esse assunto de pesquisa.

Com base nesses resultados desenvolvemos o Protocolo do Mapeamento Sistemático que irá guiar o processo, também disponível no repositório. Como indicado anteriormente, o mapeamento sistemático consiste em vários passos aplicados de maneira ordenada com a finalidade de reportar e analisar objetivamente os estudos relacionados ao assunto estabelecido. A Figura 26 apresenta, de maneira resumida, os passos seguidos para realizar o mapeamento. O processo começa com uma seleção manual da literatura base a partir de uma base de estudos especializada. Depois, segue a configuração e aplicação de buscas automatizadas em diversas bases de pesquisa. Os resultados dessas buscas passam por uma categorização com base na eliminação de duplicados e aplicando os critérios de inclusão e exclusão. Finalmente, é realizada a leitura completa e extração de informações dos estudos selecionados no processo.

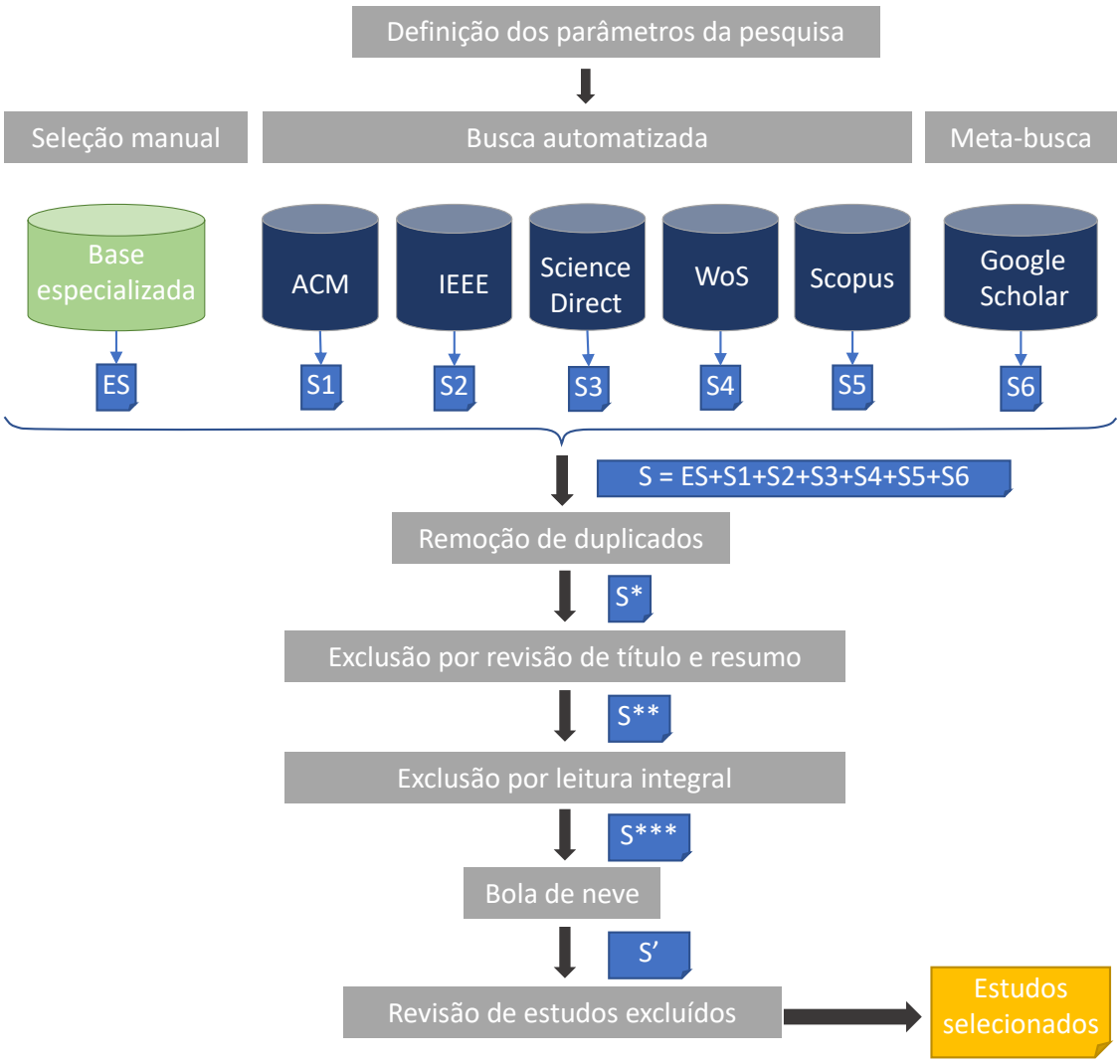


Figura 26 – Processo de Mapeamento Sistemático.
O Autor (2021).

O primeiro passo foi selecionar estudos secundários da nossa base especializada que reúnem e analisam informação sobre ferramentas para a construção de sistemas adaptativos. A lista na Figura 27 apresenta os 9 estudos selecionados.

ID	Título de estudo selecionado de base especializada	Ano
ES1	Comparison of Approaches for developing Self-adaptive Systems (Extended Version)	2017
ES2	Self-Adaptive Software: Landscape & Research Challenges	2009
ES3	Self-Adaptive Systems: A survey of current approaches , research challenges and applications	2013
ES4	A Survey on engineering approaches for self-adaptive systems (Extended version)	2018
ES5	Software Engineering for Self-Adaptive Systems: A Research Roadmap	2018
ES6	Software Engineering for Self-Adaptive Systems: A Second Research Roadmap	2013
ES7	Self-Adaptation for Cyber-Physical Systems: A Systematic Literature Review	2016
ES8	Engineering context-aware systems and applications: A survey	2016
ES9	A Survey of Autonomic Computing Methods in Digital Service Ecosystems	2017

Figura 27 – Lista de estudos selecionados manualmente de base de pesquisa especializada. O Autor (2021).

Como segundo passo elaboramos a configuração das buscas automatizadas, a qual é apresentada na Figura 28, composta de quatro conjuntos de análise. As cadeias de busca concretas dependem dos parâmetros de cada base em particular, porém todas seguem os critérios da configuração geral. As buscas foram aplicadas em 6 bases de pesquisa:

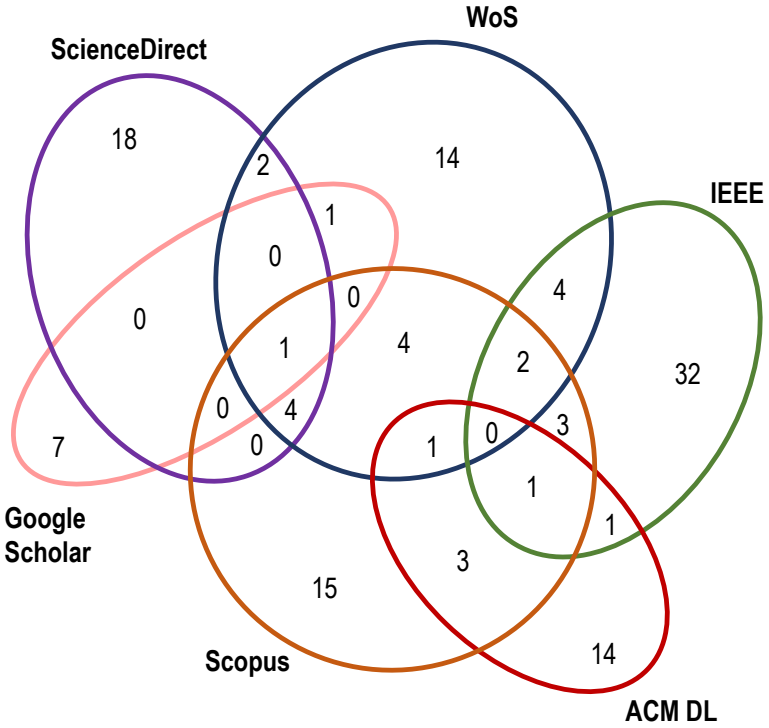
- IEEE Xplorer [4]
- Scopus[5]
- WoS[6]
- ACM[7]
- Google Scholar[8]
- Science Direct[9]

[4] http://ieeexplore.ieee.org/
[5] https://www.scopus.com/
[6] https://www.webofknowledge.com
[7] http://dl.acm.org/
[8] https://scholar.google.com/
[9] https://www.sciencedirect.com/

ID/	Tipo	Termo principal	Termo alternativo	Cadeia
Set1	População (adaptive)	adaptive	self-managing	(adaptive OR self-managing OR self-adaptive OR dynamic OR autonomic OR self-configuring)
		self-adaptive	dynamic	
			autonomic	
			self-configuring	
Set2	Intervenção	review	research roadmap	(review OR "research roadmap" OR "slr review" OR survey OR "state of the art")
			SLR review	
			survey	
			state of the art	
Set3	Comparação	---	---	---
Set4	Resultados	approaches	---	(approaches OR frameworks OR tools)
		frameworks	---	
		tools	---	
Set4	Contexto	development	---	(development OR engineering)
		engineering	---	

Figura 28 – Configuração da Busca para o Mapeamento Sistemático.
O Autor (2021).

As buscas retornaram 165 estudos, representando o conjunto S no processo de mapeamento, distribuídos como mostra a Figura 29. Desses resultados, 11 eram estudos duplicados entre as bases de dados e eram estudos duplicados da seleção manual realizada como primeiro passo. Assim, após sua remoção o conjunto S* ficou com 150 estudos para a próxima fase do mapeamento.



Base	Registros
WoS	34
Scopus	34
ACM Digital Library	20
IEEE Xplorer	43
Science DIrect	25
Google Scholar	9
Total	**165**

Figura 29 – Distribuição dos resultados da Busca para o Mapeamento Sistemático. O Autor (2021).

Em seguida, aplicamos os critérios de exclusão e inclusão, apresentados na Figura 30 e detalhados no Protocolo do Mapeamento Sistemático, sobre esse conjunto de estudos. Dos 150 estudos do conjunto S* só 6 foram selecionados para fazer parte do conjunto S**.

Prosseguimos com a leitura total dos 6 estudos selecionados e todos foram confirmados como selecionados. Assim, no final o conjunto S*** que representa os estudos que são a base do mapeamento teve os 15 estudos apresentados na Figura 31.

Por serem trabalhos de pesquisa secundários, cada um desses estudos contém referências a diversas ferramentas desenvolvidas para suportar o desenvolvimento de sistemas adaptativos. Cada um deles foi analisado e as ferramentas filtradas em base a alguns critérios:

C01: Foco do estudo: identificando o foco de cada estudo. A categoria será construída com base nos próprios dados extraídos dos estudos. Esta categoria serve para discriminar os estudos que desenvolvem ferramentas de software para a construção de Sistemas Adaptativos.

ID	Descrição
IC1	O estudo apresenta resumos de abordagens de sistemas adaptativos, ferramentas ou frameworks
IC2	O estudo se desenvolve nas áreas de Ciência da Computação ou Engenharia.
IC3	O estudo é um estudo secundário.
EC1	O estudo apresentou material não revisado por pares.
EC2	A versão em texto completo do estudo não é acessível.
EC3	O veículo de publicação do estudo não é uma conferência, workshop, periódico ou revista.
EC4	O estudo foi publicado de forma incompleta, como artigos curtos, introduções a edições especiais, tutoriais e mini-trilha (menos de 4 páginas).
EC5	O estudo está relacionado a outro estudo do mesmo autor ou já foi selecionado por outra fonte.
EC6	O foco não está no desenvolvimento ou engenharia de Sistemas Adaptativos, mas na sua aplicação a um problema particular.
EC7	O estudo não apresenta diferentes suportes de software para construção de Sistemas Adaptativos (ferramentas, frameworks, middlewares, etc).
EC8	O estudo está duplicado.
EC9	O estudo não é uma fonte secundária.
EC10	O estudo não pertence às áreas de Ciências da Computação ou Engenharia.

Critério de inclusão Critério de exclusão

Figura 30 – Critérios de inclusão e exclusão do Mapeamento Sistemático.
O Autor (2021).

ID	Título de estudos selecionados para o Mapeamento Sistemático	Ano
ES1	Comparison of Approaches for developing Self-adaptive Systems (Extended Version)	2017
ES2	Self-Adaptive Software: Landscape & Research Challenges	2009
ES3	Self-Adaptive Systems: A survey of current approaches , research challenges and applications	2013
ES4	A Survey on engineering approaches for self-adaptive systems (Extended version)	2018
ES5	Software Engineering for Self-Adaptive Systems: A Research Roadmap	2018
ES6	Software Engineering for Self-Adaptive Systems: A Second Research Roadmap	2013
ES7	Self-Adaptation for Cyber-Physical Systems: A Systematic Literature Review	2016
ES8	Engineering context-aware systems and applications: A survey	2016
ES9	A Survey of Autonomic Computing Methods in Digital Service Ecosystems	2017
S1	A Survey of Self-Management in Dynamic Software Architecture Specifications	2004
S2	Self-Awareness in Software Engineering: A Systematic Literature Review	2019
S3	A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems	2017
S4	A survey of formal methods in self-adaptive systems	2012
S5	A systematic survey on the design of self-adaptive software systems using control engineering approaches	2012
S6	Model-driven performance engineering of self-adaptive systems: A survey	2012

Figura 31 – Lista de estudos selecionados para o Mapeamento Sistemático.
O Autor (2021).

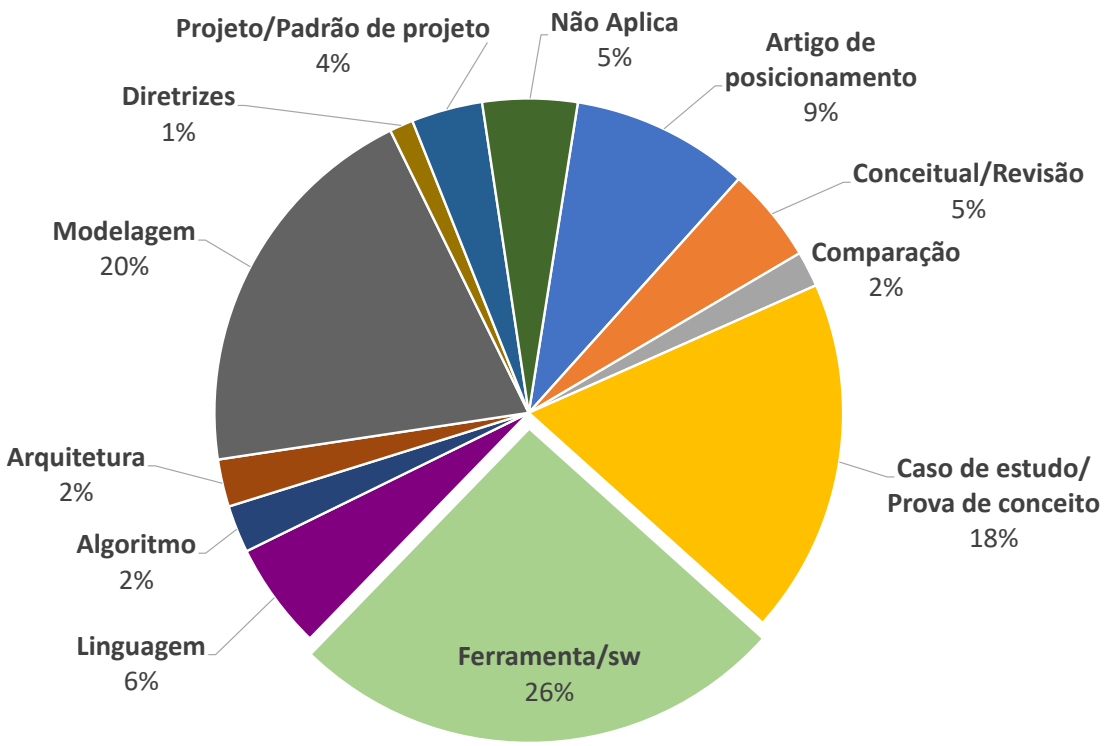


Figura 32 – Tipo de resultados dos estudos analisados.
O Autor (2021).

C02: Disponibilidade da ferramenta: Podendo ser que (1) a ferramenta está disponível, (2) a ferramenta não está disponível, ou (3) a ferramenta está indisponível (significando que deveria estar disponível, porém não é possível ter acesso);

C03: Tipo de abordagem: podendo ser (1) uma abordagem individual do pesquisador, ou (1) uma abordagem do tipo projeto de pesquisa de laboratório. Quando possível é registrado o último ano de trabalho com a proposta;

C04: Suporte para trabalho com sistemas legados: podendo ser que (1) a proposta não fornece suporte para sistemas legados, (2) explicitamente fornece suporte para trabalhar com sistemas legados, ou (3) não menciona o tipo de suporte.

Os estudos selecionados foram revisados aplicando a técnica de revisão sistemática Bola de Neve (*Snowball*) para recuperar os trabalhos relacionados. Assim, foram obtidos e analisados 164 estudos primários, entre os quais identificamos 42 estudos que desenvolveram uma ferramenta ou software de suporte para o desenvolvimento de sistemas adaptativos. Dos estudos analisados no mapeamento encontramos que 26% estão focados no desenvolvimento de ferramentas, 20% no desenvolvimento de recursos de modelagem e 18% desenvolvem software como provas de conceito e estudos de caso, enquanto a minoria, 31% dos estudos, realizam trabalhos focados em comparações, revisões, elaborar diretrizes, entre outros. A distribuição do foco dos trabalhos analisados é apresentada na Figura 32.

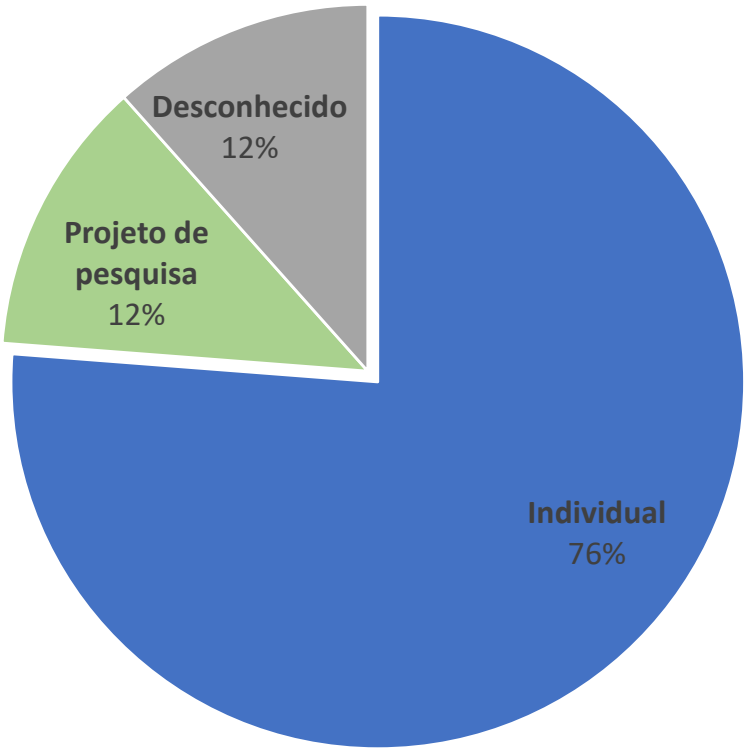


Figura 33 – Tipo de suporte dos estudos analisados.
O Autor (2021).

O mapeamento também revela que a maioria dos estudos analisados correspondem com trabalhos desenvolvidos de maneira individual pelos pesquisadores, representando o 76%. Tanto os trabalhos conduzidos no contexto de um projeto de laboratório quanto aqueles nos que não é possível identificar a abordagem representam o 12% analisado. A Figura 33 apresenta esta informação. Este critério é importante porque permite ter uma noção da continuidade da abordagem, é esperado que recursos desenvolvidos no contexto de projetos de laboratório sejam mantidos mais ativamente, fornecendo maior suporte e também prolongando a vigência da abordagem, devido a que os laboratórios de pesquisa normalmente possuem maiores recursos que um pesquisador individual.

Finalmente, a análise nos permite observar que das ferramentas desenvolvidas como parte dos estudos, as quais como mencionamos anteriormente correspondem com o 26% do total analisado, só 24% estão disponíveis para uso. Na Figura 34 também é possível observar que dentre essas ferramentas só o 10% considera ou menciona algum suporte para o trabalho com sistemas legados.

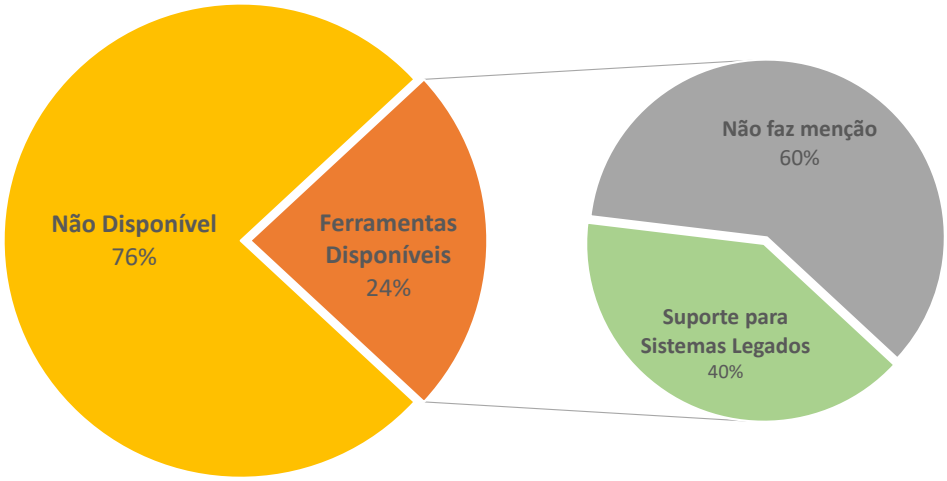


Figura 34 – Disponibilidade das ferramentas desenvolvidas pelos estudos analisados. O Autor (2021).

O Mapeamento identificou que na atualidade as ferramentas disponíveis na atualidade para o desenvolvimento de Sistemas Adaptativos que consideram, em alguma medida, o trabalho com sistemas legados são:(a) Rainbow, (b) Fractal, (c) Mocas, e (d) REACT. A Figura 35 apresenta uma linha de tempo que acompanha o desenvolvimento e publicação das ferramentas selecionadas. A seguir apresentaremos as principais características de cada ferramenta.

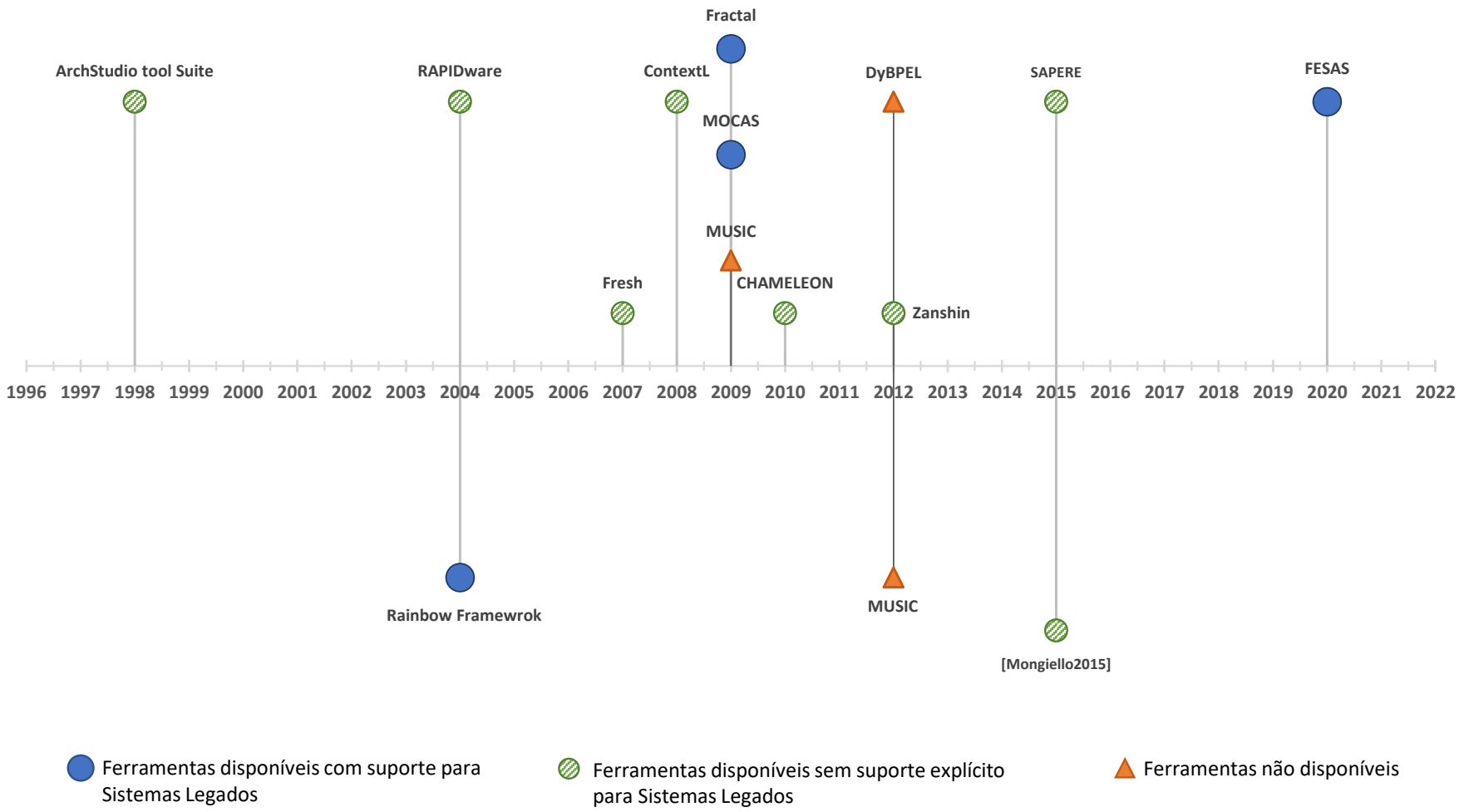


Figura 35 – Linha do tempo das ferramentas potencialmente disponíveis obtidas dos estudos analisados.
O Autor (2021).

Rainbow Framework

O Framework Rainbow é uma proposta desenvolvida pelo Instituto de Pesquisa em Software da Universidade Carnegie Mellon apresentada em Garlan *et al.* (2004) e disponível no repositório oficial do projeto[10] e no repositório alternativo de github[11].

O framework segue uma abordagem baseada em arquitetura e usa modelos de arquiteturas de software e uma infraestrutura reutilizável para monitorar e adaptar sistemas em execução, onde cada aspecto do ciclo MAPE-K é implementado por um componente. De maneira geral, a estrutura de Rainbow pode ser dividida em duas peças: uma infraestrutura de adaptação, e o conhecimento de adaptação específico do sistema alvo. A infraestrutura de adaptação, dividida em: camada de sistema, camada de arquitetura e camada de tradução, fornece mecanismos gerais de suporte para o comportamento adaptativo e é reutilizável em uma ampla gama de sistemas, enquanto o próprio conhecimento de adaptação é tipicamente específico do sistema.

Rainbow monitora as propriedades de um sistema em execução na camada de sistema por meio de *gauges* e *probes* que fazem parte de um modelo abstrato mantido pelo gerenciador de modelos na camada de arquitetura. Os avaliadores verificam a satisfação das restrições e propriedades no modelo e acionam o mecanismo de adaptação se algum problema for encontrado. O gerente de adaptação, ao receber o gatilho de adaptação, escolhe o "melhor" plano de adaptação a ser aplicado e o repassa ao executor, que executa a estratégia no sistema alvo por meio de efetores. A arquitetura e as camadas do sistema interagem por meio da camada de tradução. A melhor estratégia é escolhida com base nas preferências de utilidade das partes interessadas e no estado atual do sistema, conforme refletido nos modelos.

Uma das promessas do Rainbow é fornecer uma abordagem de baixo custo para adicionar recursos de autoadaptação em diversos tipos de sistemas. Para atingir esse objetivo, tanto a infraestrutura de adaptação quanto a linguagem de descrição arquitetônica de suporte e os conjuntos de ferramentas de desenvolvimento também precisam ser reutilizáveis. Em particular, foi desenvolvida uma linguagem específica, chamada "Stitch", para permitir a especificação e fundamentação das estratégias de adaptação em Rainbow, e pode ser usada para automatizar e coordenar adaptações para satisfazer múltiplos objetivos.

Algumas das dificuldades ao trabalhar com o Framework Rainbow que incrementam a curva de aprendizagem necessária para sua utilização são as seguintes:

- Dependência de interfaces predefinidas que permitem conectar o sistema alvo facilmente à lógica de adaptação, precisa de um número significativo de plugins para a

[10] http://www.cs.cmu.edu/~able/self-adaptation.html
[11] https://github.com/cmu-able/rainbow

instalação;

- Não é especificado um processo de desenvolvimento fácil de seguir, e não oferecem muito suporte para os usuários;
- Dependência de uma linguagem específica, Stich, para a especificação de estratégias adaptativas;
- As estratégias têm que ser projetadas através de funções de utilidade, limitando o tipo de raciocínio que pode ser feito;
- A adaptatividade é disparada pela violação das restrições, o que não permite adaptatividade como ação proativa nem preventiva, só corretiva

Fractal

Fractal é um modelo de componente modular, extensível e agnóstico de linguagem de programação que pode ser usado para projetar, implementar, implantar e reconfigurar sistemas e aplicativos, de sistemas operacionais a plataformas de middleware e interfaces gráficas de usuário. O objetivo do Fractal é reduzir os custos de desenvolvimento, implantação e manutenção de sistemas de software em geral.

O modelo de componente Fractal tem os seguintes recursos importantes:

- recursividade: os componentes podem ser aninhados em componentes compostos;
- refletividade: os componentes têm capacidade total de introspecção e intercessão;
- compartilhamento de componentes: uma determinada instância de um componente pode ser incluída (ou compartilhada) por mais de um componente;
- componentes de ligação: uma única abstração para conexões de componentes chamada de ligações. As associações podem incorporar qualquer semântica de comunicação de chamadas de métodos síncronos a chamadas de procedimentos remotos;
- independência do modelo de execução: nenhum modelo de execução é imposto. Nesse sentido, os componentes podem ser executados em outros modelos de execução que não o modelo clássico baseado em ameaças;
- aberto: serviços extra funcionais associados a um componente podem ser customizados através da noção de membrana de controle.

O modelo de componente Fractal usa fortemente o princípio de separação de interesses. A ideia deste princípio é separar em diferentes entidades executáveis os vários interesses ou preocupações de uma aplicação: implementar o serviço prestado pela aplicação,

mas também, por exemplo, tornar a aplicação configurável, segura, disponível, e eficiente. Em particular, o modelo de componentes Fractal usa três casos específicos do princípio da separação de interesses: separação de interface e implementação, programação orientada a componentes e inversão de controle. O primeiro padrão, também chamado de padrão de ponte, corresponde à separação das preocupações de projeto e implementação. O segundo padrão corresponde à separação do interesse de implementação em vários interesses menores combináveis, implementados em entidades bem separadas chamadas de componentes. Por último, Fractal fornece um padrão corresponde à separação das preocupações funcionais e de configuração: em vez de encontrar e configurar por conta própria os componentes e recursos de que precisam, os componentes Fractal são configurados e implantados por uma entidade externa separada.

O princípio da separação de interesses também é aplicado à estrutura dos componentes Fractal. Um componente Fractal é de fato composto de duas partes: um conteúdo que gerencia as preocupações funcionais e uma membrana de controladores, que gerencia zero ou mais preocupações não funcionais (introspecção, configuração, segurança, transações, entre outros). O conteúdo é composto por outros componentes Fractal, ou seja, componentes Fractal podem ser aninhados em níveis arbitrários (componentes Fractal também podem ser compartilhados, ou seja, ser aninhados em vários componentes simultaneamente). As interfaces de introspecção e configuração que podem ser fornecidas pelos controladores permitem que os componentes sejam implantados e reconfigurados dinamicamente. Essas interfaces de controle podem ser usadas programaticamente ou por meio de ferramentas baseadas nelas, como ferramentas de implantação ou supervisão.

Fractal é suportado por uma série abrangente de ferramentas de programação de código aberto: (a)Implementações leves para programação de sistema baseada em componentes; (b)API de tempo de execução para reconfiguração online; (c) Combinação de programação orientada a componentes e aspectos; e Suporte para programação e IDE.

O Projeto Fractal, baseado no modelo do mesmo nome, é um projeto aberto, hospedado pelo consórcio OW2 [12], com vários subprojetos relacionados à definição do modelo, suas implementações e a implementação de componentes e ferramentas reutilizáveis em cima dele. Um dos objetivos do projeto Fractal é o desenvolvimento de uma tecnologia de componentes de software reflexiva para a construção de sistemas distribuídos altamente adaptáveis e reconfiguráveis. O projeto também inclui o Fractal ADL, que é uma linguagem de descrição de arquitetura modular, aberta e extensível que permite a definição de arquiteturas Fractal e pode suportar a criação de diversos módulos para cobrir diferentes necessidades.

Como observamos, Fractal é equivalente a um meta-modelo que deve ser instanciado para aproveitar as vantagens que oferece, tanto no nível de abrangência de sistemas quanto

[12] https://fractal.ow2.io/

na flexibilidade que oferece para reconfigurações em tempo de execução. Em particular, as abordagens DiVa, CAPucine e FraSCAti são modelos para sistemas adaptativos que seguem princípios de arquitetura diferentes e que podem ser implementadas com Fractal. Assim, o problema com Fractal é que antes de implementar um sistema com este recurso o pesquisador interessado deve instanciar Fractal para um modelo em particular. Existem alguns modelos já implementados, para Java e C++, fornecendo certa praticidade para projetos nessas linguagens. Porém, para outros casos o pesquisador interessado deve implementar o modelo e criar toda a estrutura necessária, conforme os requerimentos de Fractal, que são bastante específicos. Esta liberdade para criar novas versões de acordo à necessidade do projeto fornece uma grande flexibilidade, mas impõe um custo muito alto para a pesquisa. Em primeiro lugar, as decisões de projeto já foram tomadas e estão especificadas no modelo, a flexibilidade fornecida é ao respeito da linguagem de programação a ser utilizada e não sobre o modelo em si. Em segundo lugar, Fractal é um projeto que data de 2009, e embora esteja disponível não existem novos trabalhos desenvolvidos com a ferramenta, nem possui suporte ativo por parte da equipe de desenvolvimento. Isso gera uma distância tecnológica de 13 anos difícil de superar devido às grandes mudanças entre as tecnologias usadas e as atuais. Por exemplo, mesmo oferecendo compatibilidade com versões anteriores, atualmente Java não garante o funcionamento de software nem fornece suporte, para Java 6 que foi a versão utilizada para a construção de Fractal.

MOCAS

MOCAS, sigla de *Model Of Components for Adaptive Systems*, é uma iniciativa desenvolvida na Universidade de Pau na França e detalhada nos trabalhos de Ballagny, Hameurlain e Barbier (2009) e Ballagny, Hameurlain e Barbier (2008).

Mocas é um modelo de componente genérico que permite construir sistemas SaS baseados em componentes e abertos para realizar propriedades autonômicas. Ele se baseia em conceitos de engenharia orientada a modelo (MDE, do inglês Model-Driven Engineering) e usa modelos de máquina de estado UML para descrever o comportamento dos componentes de software. Um componente MOCAS faz uma separação clara entre seu comportamento, seu contexto funcional e suas propriedades de negócio. O comportamento dos componentes é especificado por máquinas de estado UML, enquanto sua estrutura é restrita por um perfil UML. Um estado é uma situação de interessante do componente, do ponto de vista do projetista do sistema, em relação ao modo de operação do componente. Um componente MOCAS incorpora, em tempo de execução, o modelo de máquina de estado associado para realizar seu comportamento. Os componentes são instalados em um contenedor, segundo o modelo de componentes MOCAS, que age como um wrapper sendo projetado para mediar a interação entre os componentes e seu ambiente. Assim, o contenedor é o gerenciador do processo de adaptação dos componentes, que não conhecem os mecanismos de adaptação, permitindo que a adaptação seja totalmente transparente

para o componente encapsulado. O contenedor permite que cada parte do componente seja adaptada, ou seja, suas propriedades, seu contexto funcional e seu comportamento podem ser modificados independentemente.

Além disso, o contenedor monitora as invariantes do componente e garante adaptações consistentes, verificando as pré e pós-condições durante o processo de adaptação. Cada componente autônomo de MOCAS (AMC) é dotado de seu próprio um ciclo de controle, permitindo sua adaptação de forma descentralizada. As políticas de adaptatividade são especificadas graças aos modelos de máquina de estado, localmente para cada componente autônomo, e podem ser alteradas em tempo de execução. Este processo mantém os estados ativos para que nenhuma ação anexada a um estado (de entrada, de saída ou de ações) seja executada devido à adaptação, é dizer, elas têm caráter atômico. Isso permite a total transparência do processo para o projetista do sistema e evita interrupções no serviço. Adicionalmente, um componente autônomo está ciente da adaptação de outros componentes, inscrevendo-se neles. Dessa forma, ele pode se adaptar como consequência da adaptação de outros componentes e impulsionar sua adaptação.

Ao observar a proposta de MOCAS podemos identificar que fornece várias vantagens, porém também traz algumas limitações. Por um lado, por ser um modelo, todos os componentes de um SaS desenvolvido com MOCAS devem ser componentes que seguem esse modelo. Assim, um sistema legado, precisa de um wrapper para se tornar um componente MOCAS e logo poder participar do desenvolvimento de um SaS. Se bem MOCAS apresenta um plugin em Eclipse para assistir no desenvolvimento do sistema e verificar a conformidade com o modelo, isso cria duas dificuldades: a primeira tem a ver com realizar uma atividade adicional no processo de desenvolvimento do sistema, e a segunda com a limitação do suporte para a linguagem de programação Java.

Adicionalmente, a participação do contenedor no processo de adaptação pode gerar um peso adicional no desempenho do sistema: qualquer sinal é primeiro interceptado pelo contenedor antes de ser processado pelo componente e depois é replicado para qualquer componente do sistema. Assim, todo o sistema deve ser considerado no processo de adaptação para evitar instabilidades ao se adaptar localmente. Em particular, os componentes de um sistema precisam estar atentos às adaptações que ocorrem nesse sistema para manter sua coerência. MOCAS ainda não fornece suporte para comunicação por meio de protocolos de interação entre componentes, o que significa que essas coordenações devem ser feitas pelo projetista ao criar os componentes.

Por outro lado, MOCAS realiza exclusivamente adaptações comportamentais e não adaptações estruturais, assim a arquitetura do sistema é mantida constante.

REACT

REACT é um ambiente de execução baseado em modelos desenvolvido na Universidade de Mannheim e detalhado nos trabalhos Pfannemüller *et al.* (2020b), Pfannemüller *et al.* (2020a), Pfannemüller *et al.* (2021) e disponível sob a modalidade de software aberto, no repositório de github do projeto[13] . REACT foi projetado pensando especificamente em Sistemas de Comunicação, de fato REACT é uma sigla de *Runtime Environment for Adapting Communication SysTems* e se identifica como um SaS seguindo uma proposta baseada em modelos e componentes. A ferramenta implementa um ciclo de controle e interfaces independentes de linguagem para estabelecer a conexão com o sistema alvo. O ciclo de controle segue o padrão MAPE-K fornecendo algumas funcionalidades prontas para uso. REACT utiliza uma diversidade de tecnologias para fornecer flexibilidade e gerenciar processos internos, tais como Clafer, iPOJO e OSGi para a configuração do sistema e Docker para sua distribuição. A plataforma apresenta uma proposta independente de linguagem de programação recorrendo à ferramenta ZeroC Ice. O diferencial de REACT é que os seus desenvolvedores incluíram a facilidade de uso como uma das suas decisões de projeto, colocando em evidência que um dos seus objetivos é incentivar o uso da ferramenta fora da área de SaS.

O usuário de REACT precisa fornecer dois modelos em tempo de projeto: a especificação das opções de adaptação e a especificação do sistema alvo. O primeiro modelo é uma representação explícita de opções de reconfiguração válidas na linguagem de modelagem estrutural Clafer que descreve o espaço do problema. A especificação do sistema alvo representa a arquitetura do desse sistema em UML e, portanto, descreve o espaço da solução. Esses modelos são usados e atualizados pelo ciclo de controle para realizar adaptações.

Algumas das dificuldades com REACT estão relacionadas à sua flexibilidade e seu foco. Mesmo quando a plataforma foi desenvolvida pensando na facilidade de uso, as tecnologias que utiliza como suporte supõem uma familiaridade do usuário que não necessariamente existe. Além disso, por estar focado em dar suporte para a implementação de adaptatividade em sistemas de comunicação, as demonstrações e exemplos apresentados são próprios dessa área, e se mostram com um grau de complexidade desnecessário, especialmente se a necessidade do usuário é de implementar a adaptatividade ao nível básico.

Podemos concluir com base nos resultados do mapeamento e resumidos na Figura 36 que existem poucas ferramentas disponíveis para o desenvolvimento de Sistemas Adaptativos, e menos ainda são aquelas que fornecem algum suporte para trabalhar com sistemas legados. Além disso, as ferramentas detalhadas anteriormente ainda apresentam

[13] https://github.com/martinpfannemueller/REACT

Ferramenta	Última atualização	Fácil de compreender e usar?	Suporta Sistemas Legados como parte dos sistemas adaptativos?	Proposta de aplicação genérica de adaptatividade em software?	Disponível?
Rainbow	2019	Não	Aqueles desenvolvidos com os componentes associados à ferramenta.	Sim	Sim: github
Fractal	2009	Não	Aqueles desenvolvidos com os componentes associados à ferramenta.	Sim	Sim: site oficial do projeto
MOCAS	2008	Sim	É possível através de desenvolvimento de wraps para integrar os sistemas.	Sim	Parcialmente: SourceForge
FESAS	2020	Sim	Em teoria poderia ser realizada a integração com sistemas legados.	Sim	Parcialmente: 1 de 2 componentes está disponível no github
REACT	2022	Não muito	Em teoria poderia ser realizada a integração com sistemas legados.	Possivelmente, porém ele foi planejado especificamente para sistemas de comunicação.	Sim : github

Figura 36 – Comparação entre ferramentas
O Autor(2021)

dificuldades tanto na simplicidade da implementação da proposta, quanto na liberdade de projeção dos sistemas adaptativos. Rainbow e REACT são as que oferecem maior documentação e têm tido maior continuidade de trabalho, porém ambas fixam as decisões de projeto dos sistemas adaptativos na ferramenta, não são agnósticas de abordagem, e, em consequência, dependem de uma adequada manutenção.

Assim, ainda existe a necessidade de uma proposta para desenvolvimento de Sistemas Adaptativos que ofereça simplicidade, flexibilidade, implementação, seja agnóstica de abordagem, e permita a reutilização de recursos de software como parte do seu ciclo.

No seguinte capítulo detalharemos uma proposta que pretender responder a essas necessidades.

5 PROPOSTA

Com base nas informações obtidas no mapeamento sistemático do capítulo anterior, neste capítulo detalhamos a nossa proposta para colaborar com a resolução dos desafios identificados, tanto no nível conceitual quanto no nível tecnológico. A seguir apresentamos quais são as diretrizes que a proposta deve cumprir para cada nível e descrevemos a ideia básica por trás das escolhas realizadas.

5.1 Marco Conceitual

No nível teórico, observamos a necessidade de um marco conceitual que possa agir como um hiperônimo, ou termo guarda-chuva, e integrar as diferentes abordagens e pontos de vista existentes sobre o comportamento adaptativo nas diversas áreas. Em particular, o marco conceitual deve satisfazer os seguintes requisitos:

- **Ser geral**: não se limitar a resolver apenas um caso particular, mas representar o comportamento adaptativo nas suas diversas naturezas, permitindo concretizar diferentes visões e se especializar quanto necessário para satisfazer as demandas do problema em questão.
- **Ser adaptável**: admitir mudanças, os conceitos apresentados podem ser abstraídos ou especializados para se encaixar de maneira mais adequada no contexto particular em análise. Porém, simultaneamente, eles podem ser utilizados diretamente sem perda de vantagens ou benefícios.
- **Ser aberto**: a proposta não apresenta um marco conceitual único, completo ou rígido, pelo contrário ele sempre estará em desenvolvimento, com a capacidade de integrar novas descobertas e enriquecer o corpo de conhecimento que guarda. Assim, a proposta serve ao propósito de integração entre abordagens.
- **Ser agnóstica de abordagem**: dado que o comportamento adaptativo é um fenômeno observável desde diferentes perspectivas e as suas propriedade são requeridas e desejadas em diversas áreas, os conceitos que definem e descrevem o comportamento adaptativo, de maneira geral, não devem estar restritos a nenhuma área em particular. O marco conceitual não deve exigir que o leitor (usuário) tenha uma bagagem teórica específica para acompanhar e aproveitar os conceitos recolhidos sobre o comportamento adaptativo. O marco conceitual não pretende entrar em conflito com as abordagens particulares de adaptatividade nem negar seus conhecimentos, pelo contrário, o objetivo deste recurso é servir como elemento integrador para a

construção de uma teoria de adaptatividade unificada e facilitar o compartilhamento de conhecimentos sobre a adaptatividade. A proposta deve ser realizada a maneira de motivar e facilitar comparações e paralelos entre as diferentes abordagens que abriga. Através dela deve ser possível estabelecer e recuperar conexões que de outra maneira não seriam facilmente identificados. É importante lembrar que um dos objetivos da proposta é servir de ponte para o compartilhamento de conhecimentos sobre adaptatividade e colaboração entre áreas do conhecimento.

- **Utilizar uma linguagem simples**: a linguagem utilizada para construir o marco conceitual deve ser específica para evitar ambiguidades ou dúvidas e questionamentos desnecessários. Evitar o uso de termos técnicos avançados ou de domínios particulares, dando preferência a termos gerais e formais.

- **Utilizar conceitos básicos**: os conceitos utilizados na proposta devem ser gerais e básicos, significando que são simples de entender, caracterizam de maneira adequada o assunto em questão e permitem realizar diversificações, encontrar exemplos e relacionar com conceitos mais especializados com facilidade.

- **Ser construtiva**: significando que no começo o marco conceitual se baseia em conceitos elementares e amplamente conhecidos (conhecimentos básicos), e, aos poucos, vai construindo novos conceitos de maneira clara e precisa com base no conhecimento prévio (conhecimentos compostos). Assim, o leitor participa da própria construção dos conceitos em cada nível, favorecendo o aprendizado e identificação com o meta-modelo, e, portanto, sua aplicação para o caso específico requerido.

- **Empregar um nível de detalhe adequado**: a granularidade do comportamento adaptativo é ampla, assim, é possível descrevê-lo em um nível abstrato, simples e breve, porém sem se referir a conceitos importantes que suportam os seus processos. Por outro lado, também é possível descrever a adaptatividade no nível anatômico (no sentido abstrato) fornecendo detalhes de cada ação, participação, reação, e possibilidade que seja de interesse. Assim, consideramos que o marco conceitual deve fornecer um grau de detalhe médio, que permita uma descrição clara dos componentes e principais processos que fazem parte do comportamento adaptativo, sem detalhar como esses elementos podem ser realizados.

- **Focar no suporte à participação (integração) de elementos pre-existentes na conformação de sistemas adaptativos**: a revisão realizada no capítulo anterior colocou em evidência que embora a grande maioria de abordagens gerais sobre adaptatividade reconhecem a separação de interesses e a hierarquia como características dos sistemas adaptativos elas perdem esse foco ao descrever os seus processos. O marco conceitual proposto deve evidenciar essas separações ao longo do seu desenvolvimento, deixando claro quais são as atividades e participações de

cada componente nos níveis correspondentes. Isto é importante, pois a maioria de abordagens implica a realização de sistemas adaptativos como a criação de uma entidade única, ao invés da evolução de uma entidade através do trabalho coordenado com novos recursos para formar um novo sistema.

Para cobrir esses requisitos escolhemos desenvolver um meta-modelo para adaptatividade. O meta-modelo deve colocar especial cuidado na generalidade da linguagem utilizada e de seus conceitos básicos para fornecer um marco conceitual autônomo (auto-contido) na medida do possível. Além disso, é necessário que ao construir o meta-modelo a adaptatividade seja analisada como um fenômeno único, que pode tomar diferentes formas e com partes que interagem para manifestar diversas propriedades. Nesse sentido, escolhemos aplicar uma abordagem holística na qual reunimos visões de adaptatividade tanto das Ciências Naturais e Filosofia quanto das áreas tecnológicas.

Dessa maneira, o meta-modelo holístico de adaptatividade age como um guarda-chuva conceitual ao ser um recurso com um nível de abstração superior ao das abordagens particulares, as quais podem ser compreendidas como instâncias particulares do meta-modelo, e, portanto, modelos de adaptatividade. A Figura 37 é uma representação desta relação entre meta-modelo e modelos, a figura não inclui todas as abordagens, só algumas, a maneira de exemplo. Para permitir visualizar esta relação, o meta-modelo está acompanhado de alguns recursos que facilitam a identificação de correspondência com outras abordagens.

5.2 Framework Tecnológico

No aspecto tecnológico, a proposta formulada é a implementação de uma ferramenta que suporte o desenvolvimento de sistemas adaptativos com base no meta-modelo holístico de adaptatividade através da criação de mecanismos adaptativos. A ferramenta deve permitir ao usuário trabalhar com modelos de adaptatividade genéricos (abstratos) ou assistir no projeto e implementação de mecanismos adaptativos para dispositivos legados que cumpram as condições necessárias. A seguir seção são detalhadas as principais características que modelam a ferramenta proposta, assim como algumas considerações consideradas para o seu projeto.

5.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem em alto nível a funcionalidade principal da proposta. Para facilitar sua leitura eles estão divididos em: aspectos gerais, modalidade genérica e modalidade específica.

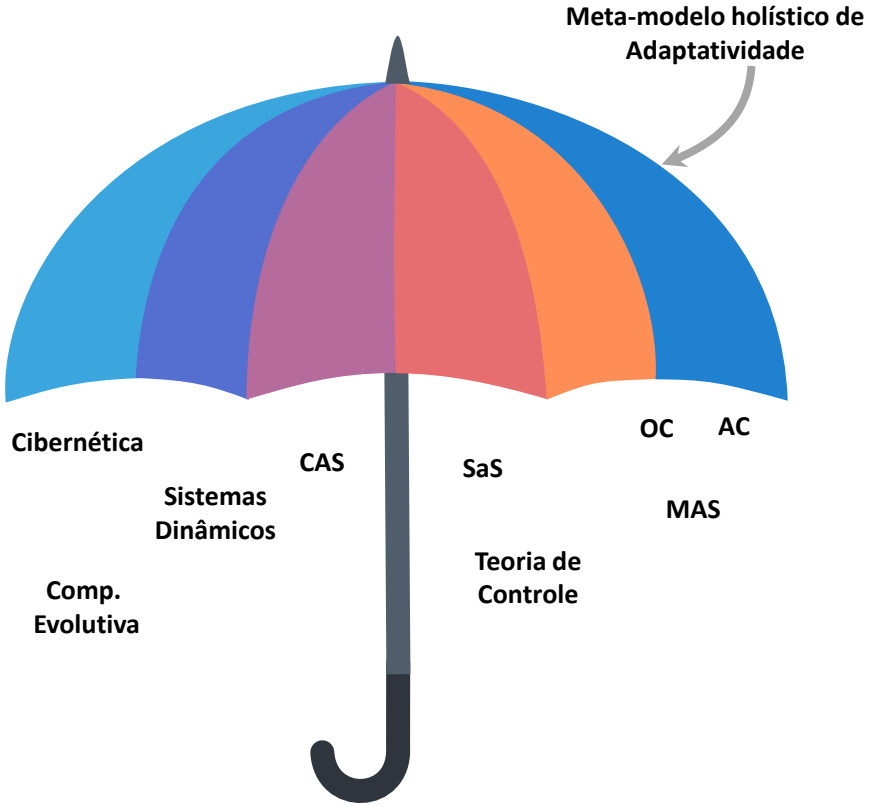


Figura 37 – Guarda-chuva conceitual de adaptatividade.
O Autor (2022).

Aspectos Generais

(*F1*) *A ferramenta apresenta uma modalidade genérica.*
Nessa modalidade o usuário trabalha com recursos associados a modelos desde uma perspectiva genérica (abstrata), sem considerar particularidade de dispositivos concretos.

(*F2*) *A ferramenta apresenta uma modalidade específica.*
Essa modalidade permite que o usuário indique um dispositivo subjacente e um modelo de adaptatividade na ferramenta para guiar o projeto de uma camada adaptativa orientada a atender tais características.

(*F3*) *A ferramenta permite a incorporação de novos modelos de adaptatividade.*
O usuário pode incorporar novos modelos de mecanismos adaptativos, sempre que esses modelos sejam aderentes ao padrão estabelecido para esse tipo de especificação. A ferramenta diferencia entre os modelos pelo nome pelo que recomendamos o uso de uma convenção de nomes do tipo: Nome[Linguagem], por exemplo: “Autonomic Computing [UML]”.

(*F4*) *A ferramenta permite o gerenciamento das implementações dos modelos de adaptatividade por linguagem de programação.*
Dado um modelo de adaptatividade, o usuário pode incorporar novas implementações em linguagens de programação e paradigmas diferentes. Da mesma maneira, o usuário pode atualizar uma implementação existente em uma linguagem particular submetendo os novos arquivos de código-fonte correspondentes.

(*F5*) *A ferramenta permite o gerenciamento de novos exemplos que implementam um modelo em particular para uma linguagem de programação específica.* Dado um modelo de adaptatividade, o usuário pode incorporar novos exemplos em linguagens de programação e paradigmas diferentes que implementam tal modelo. Da mesma maneira, o usuário pode atualizar um exemplo existente em uma linguagem particular submetendo os novos arquivos de código-fonte correspondentes.

Modalidade Genérica

(*F6*) *O usuário pode selecionar um modelo de adaptatividade e exportar sua especificação em uma das linguagens de modelagem disponíveis.*

(*F7*) *O usuário pode selecionar um modelo de adaptatividade e exportar sua implementação, como arquivos de código-fonte, em alguma das linguagens e paradigmas de programação disponíveis.*

(*F8*) *O usuário pode selecionar um modelo de adaptatividade e exportar sua implementação, como biblioteca, em alguma das linguagens e paradigmas de programação disponíveis.*

(*F9*) *Dado um modelo, o usuário pode selecionar um gerenciador de dependências e exportar e/ou copiar a sua dependência de software.*

(*F10*) *Dado um modelo, o usuário pode selecionar um exemplo de uso disponível, assim como uma linguagem e paradigmas de programação e exportar os arquivos de código-fonte que implementam essa configuração exemplo.*

Modalidade Específica

(*F11*) *A ferramenta solicita ao usuário a pasta na estrutura de arquivos que contém o dispositivo subjacente para sua verificação.* A ferramenta confere que o caminho apontado pelo usuário corresponde a um diretório e não a um arquivo. Caso contrário, solicita ao usuário um caminho válido.

(*F12*) *A ferramenta permite o uso de arquivos de configuração para especificar as informações do dispositivo.*
A ferramenta procura um arquivo de configuração (com extensão . json e estrutura

determinada) na raiz do caminho fornecido pelo usuário para carregar as informações correspondentes ao dispositivo. Se tal arquivo for encontrado, a ferramenta apresenta o formulário de verificação do dispositivo preenchido com tais informações.

(*F13*) *A ferramenta permite que o usuário especifique as informações do dispositivo manualmente no formulário correspondente.*
Se não for encontrado um arquivo de configurações, a ferramenta apresenta um formulário vazio para ser preenchido pelo usuário solicitando as informações necessárias sobre o dispositivo.

(*F14*) *A ferramenta verifica que o dispositivo submetido possui as condições necessárias para a adaptabilidade.*
A ferramenta verifica que todas as informações necessárias para descrever o dispositivo foram fornecidas e verifica se as condições necessárias para adaptabilidade foram satisfeitas.

(*F15*) *O usuário pode exportar um arquivo de configurações com as informações especificadas para um dispositivo verificado.*
A ferramenta permite um arquivo de configurações baseado nas informações fornecidas pelo usuário no formulário correspondente para um dispositivo adaptável.

(*F16*) *Caso o dispositivo não for verificado como adaptável, a ferramenta permite que o usuário escolha outro dispositivo ou realize as correções necessárias.*
Se o dispositivo não cumprir com as condições necessárias para a adaptabilidade, o usuário poderá voltar na janela anterior e escolher outro dispositivo ou corrigir os critérios que faltaram satisfazer para verificar a adaptabilidade.

(*F17*) *A ferramenta permite a escolha de um modelo de adaptatividade e uma linguagem de programação como base para a criação do mecanismo adaptativo para o dispositivo adaptável submetido.*
Uma vez verificado que o dispositivo submetido é adaptável, a ferramenta solicita que o usuário escolha o modelo de adaptatividade e a linguagem de programação que irá implementar.

(*F18*) *O usuário preenche os dados necessários para a implementação do processo de Monitoramento.*
A ferramenta solicita ao usuário que indique a conexão concreta ente o Monitor e os métodos que agem como sensores monitorar as mudanças específicas. Da mesma maneira, a ferramenta solicita o caminho dos recursos utilizados como parte do processo de monitoramento e para transferência de informações.

(*F19*) *O usuário preenche os dados necessários para a implementação do processo de Raciocínio.*

A ferramenta solicita ao usuário que indique o caminho para leitura das informações necessárias para realizar o raciocínio, tais informações correspondem tanto as comunicações com o monitoramento quanto a políticas e estratégias de avaliação a serem aplicadas. Da mesma forma, a ferramenta solicita o caminho para transferência das informações.

(*F20*) *O usuário preenche os necessários para a implementação do processo de Seleção.*
A ferramenta solicita ao usuário que indique o caminho para leitura das informações necessárias para realizar a seleção, tais informações correspondem tanto as comunicações com o processo de raciocínio quanto a políticas e estratégias de seleção a serem aplicadas. Da mesma forma, a ferramenta solicita o caminho para transferência das informações.

(*F21*) *O usuário preenche os dados do Executor solicitados.*
A ferramenta solicita ao usuário que indique a conexão concreta ente o Executor e os métodos que agem como atuadores para aplicar as mudanças necessárias.

(*F22*) *A ferramenta verifica que as informações necessárias para a implementação foram fornecidas.*
Se alguma informação não foi preenchida ou suas informações não são consistentes, a ferramenta informa ao usuário a fonte do problema.

(*F23*) *A ferramenta permite exportar o mecanismo adaptativo correspondente com as informações fornecidas.*
O usuário pode exportar o mecanismo adaptativo em formato de código-fonte na linguagem escolhida.

5.2.2 Requisitos Não Funcionais (NF)

(*NF1*) *A ferramenta deve seguir a filosofia de software Livre.*
Para facilitar o seu compartilhamento, distribuição, uso, evolução e melhora. Novamente, um dos objetivos deste trabalho é motivar o uso de comportamento adaptativo por parte dos pesquisadores e usuários em geral.

(*NF2*) *A ferramenta deve ser acessível via conexão à internet.*
Ser disponibilizada em um repositório de código apropriado e livre, que permita o gerenciamento de versões e sua distribuição a terceiros. Assim mesmo, o repositório deve acompanhar os trabalhos e publicações decorrentes para fornecer reprodutibilidade à pesquisa.

(*NF3*) *A ferramenta deve contar com documentação apropriada tanto na própria interface quanto em fontes externas.*
A documentação não precisa ser extensa, porém dever ser o suficientemente clara para

permitir que o usuário realize um teste na ferramenta, acompanhe sua navegação, e esclareça dúvidas eventuais.

(*NF4*) *A ferramente deve apresentar usabilidade segundo os critérios estabelecidos pela Nielsen Norman Group.*
A ferramenta deve adotar e verificar as dez heurísticas para usabilidade para o design da interface do usuário descritas em Nielsen (2020) e Nielsen (2012). A adoção dessas diretrizes permite que a usabilidade da ferramenta possa ser medida segundo essas heurísticas.

(*NF5*) *A ferramenta é adaptável.*
A plataforma segue a mesma filosofia de tecnologia adaptativa e ela mesma pode ser mudada, todavia, os requisitos são diferentes.

(*NF6*) *A ferramenta emprega tecnologias e técnicas que permitam dar longevidade à ferramenta.*
As tecnologias utilizadas no desenvolvimento da ferramenta são as últimas versões estáveis disponíveis, e apresenta funcionalidades para expandir tanto as opções que apresenta para o usuário quanto novos formatos de descrição de camadas adaptativas e protocolos de comunicações.

(*NF7*) *A ferramenta é multiplataforma.*
As tecnologias utilizadas para o desenvolvimento da ferramenta possuem suporte multiplataforma.

(*NF8*) **A ferramenta apresenta uma configuração mínima pronta para uso.**
A ferramenta apresenta uma modalidade que permite a interação rápida com o usuário.

(*NF9*) *A ferramenta adota medidas para fornecer flexibilidade de software.*
Através da implementação de uma forma de fábrica de software com economia de escopo para o desenvolvimento de camadas adaptativas, arquivos de configuração para caracterizar os dispositivos subjacentes e metadados para troca de informações e adoção de políticas. Da mesma forma, os recursos fornecidos pela ferramenta pode ser obtidos em diferentes formatos.

(*NF10*) *A ferramenta contempla a utilização de diversidade de linguagens.*
A ferramente permite a escolha de diferentes linguagens tanto no nível de modelagem quanto no nível de programação.

5.2.3 Arquitetura

Na sequência, com base nos trabalhos desenvolvidos em Serugendo *et al.* (2008), Schmeck *et al.* (2010), Liu e Schmeck (2006), apresentamos as características arquitetônicas de alto nível que servem como diretrizes para o desenvolvimento da ferramenta proposta:

- *DBC como arquitetura-base.*
  O desenvolvimento de sistemas de software baseados em componentes (DBC), conforme foi detalhado em capítulos prévios, oferece mecanismos adequados para gerenciar sistemas formados por componentes autônomos, dinâmicos e heterogêneos de maneira homogênea, modular e baixamente acoplada. Em especial, fornece técnicas e estratégias para assistir à implementação da presente proposta nos seguintes pontos:

  (i) flexibilidade via arquivos de configuração que mantém a separação de interesses claramente identificada.

  (ii) viabiliza o trabalho com peças de software heterogêneas sobre a base de interfaces contratuais.

  (iii) a mudança de peças de software coesas e com baixo acoplamento com menor risco devido às interfaces e padrões que regem as interações.

  (iv) situa a ferramenta em um ponto estável no desenvolvimento de tecnologias, se bem o paradigma não é o mais recente, é um dos mais utilizados e suportados. A transformação de componentes stand-alone para outras abordagens, por exemplo: serviços numa arquitetura SOA, são de custo operacional relativamente baixo, assistida por várias ferramentas e de interesse para a comunidade científica.

- *Arquitetura em camadas.* Nossa proposta estabelece uma separação em três camadas de abstração: a camada básica, a camada intermediária e a camada alta, como mostra a Figura 38. A camada básica gerencia o dispositivo adaptável. A camada intermediária realiza um trabalho de middleware entre o componente adaptável, na camada básica, e os modelos de mecanismos adaptativos, na camada alta. A camada alta gestiona as informações e recursos relacionados com o meta-modelo holístico de adaptatividade e suas instâncias. Dependendo da modalidade executada, a ferramenta pode utilizar só a camada alta para fornecer ao usuário recursos genéricos.

- *Implementação do tipo fábrica de software com economia de escopo.*
  A ferramenta segue a filosofia de economia de escopo das fábricas de software para fornecer implementações de mecanismos adaptativos. Assim, dependendo das condições, a ferramenta fornece soluções adequadas às necessidades específicas. Esta abordagem tem como base o uso de repositórios de modelos de adaptatividade genéricos que descrevem os processos envolvidos no comportamento adaptativo. Para ser utilizados como parte de um sistema, esses modelos devem ser especializados para

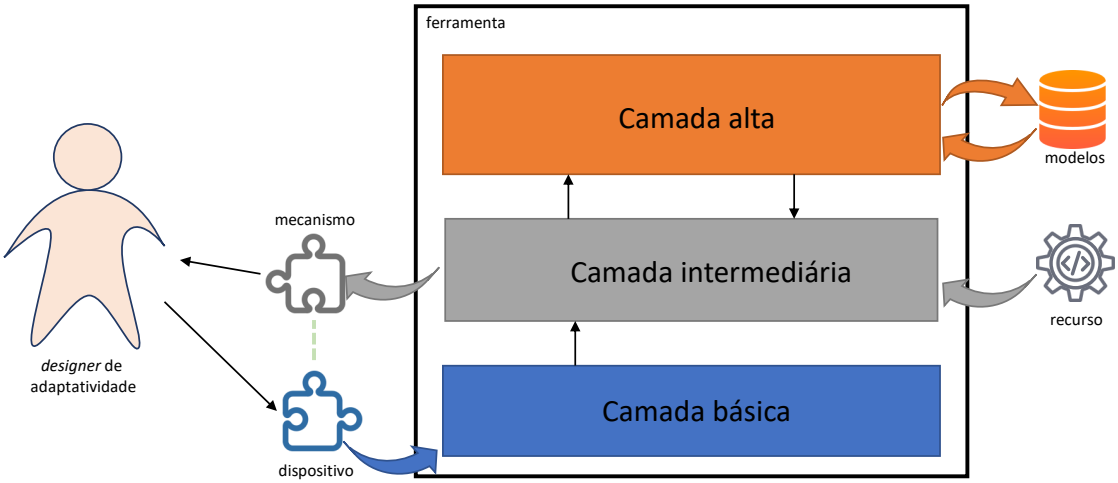


Figura 38 – Arquitetura da solução
O Autor (2021)

considerar as informações de cada dispositivo, metadados, políticas e recursos do caso particular. A implementação dos mecanismos adaptativos para um dispositivo específico é configurada pelo usuário com assistência da "modalidade específica" na ferramenta.

- *Especificação de modelos e padrões.*
  Diferentes pontos da proposta estão suportados por modelos e padrões em vários níveis e domínios de aplicação, desde boas práticas de programação, até o meta-modelo holístico de adaptatividade. Os padrões e modelos, quando utilizados corretamente, procuram oferecer soluções eficientes ou caminhos de solução especializados e confiáveis para conflitos determinados.

- *Uso de arquivos de configuração.*
  A ferramenta permite o uso de arquivos de configuração para o reconhecimento em tempo de execução das características correntes dos componentes. Assim, ela permite tanto a descrição do dispositivo subjacente quanto de políticas e estratégias disponíveis de maneira que suporta o baixo acoplamento dos componentes do sistema e favorece a interoperabilidade. As informações correspondentes são acessíveis por interfaces genéricas, disponíveis em todos os componentes, e especificadas no modelo de componentes usado neste trabalho.

- *Uso de Metadados: aquisição, atualização e monitoramento.*
  Os metadados são as informações declarativas adicionais sobre o sistema em execução (componentes, infraestrutura e ambiente) que podem ser acessadas por outros dispositivos. Eles também podem transmitir informações sobre características funcionais e não-funcionais. Da mesma forma, metadados podem ser utilizados em diferentes níveis e com diferentes focos: para monitorar quão bem uma decisão afeta uma determinada propriedade self-*, como quantificador do grau de adaptatividade do

sistema, como controle do comportamento do sistema ao mudar certos parâmetros e gerar um raciocínio sobre eles. O uso de metadados está particularmente alinhado com o paradigma DBC, pois permite que os componentes sejam autodescritivos, independentes da linguagem e adaptáveis, mantendo a correspondência através da extensão dos metadados (MICROSOFT DOCS, 2017).

Essas características serão concretizadas em dois momentos. Primeiramente, em tempo de projeto serão identificadas tanto as propriedades dos componentes quanto do sistema em geral, bem como os padrões e políticas correspondentes. Depois, em tempo de execução, são explorados os metadados para suportar as decisões e adaptações determinadas pelos mecanismos adaptativos em cada caso.

O desenvolvimento da ferramenta, assim como as considerações técnicas e tecnológicas correspondentes, são detalhadas no Capítulo 7.

6 META-MODELO HOLÍSTICO DE SISTEMAS ADAPTATIVOS

Como mencionado anteriormente, os termos derivados de *adaptatividade* são utilizados amplamente nas diferentes áreas do conhecimento com o intuito de descrever diversas características presentes no objeto de estudo. Para cada um desses contextos os pesquisadores podem ter interesses particulares, de maneira que a definição do termo carrega:

- Mais ou menos quantidade de informações sobre o que ele caracteriza, dependendo de se a adaptatividade é um assunto principal ou complementar na pesquisa;
- Diferentes recursos disponíveis para descrever o processo sob observação, tais como: fontes de informação ou elementos afetados pela adaptatividade;
- Diferente nível de detalhe e rigor na sua definição;
- Terminologia própria da área de pesquisa.

É justamente por causa dessa flexibilidade que não existe uma definição única de adaptatividade. Contudo, isso não significa que o termo seja impreciso ou ambíguo, pelo contrário o comportamento adaptativo possui características que lhe permitem se distinguir de outros fenômenos de maneira clara. Portanto, é importante desenvolver uma familiaridade com o conceito de "adaptatividade" apresentado em capítulos anteriores, para poder observar e reconhecer sistemas que manifestam suas características mesmo quando englobados em diferentes áreas de conhecimento.

Este capítulo centra-se na conceição dos sistemas com comportamento adaptativo. Primeiro é apresentado como esses sistemas são entendidos em diferentes áreas com a finalidade de analisar de maneira abrangente suas características. Posteriormente, é proposto um meta-modelo holístico que procura servir como framework conceitual geral para esse tipo de comportamento. Para facilitar essa análise do meta-modelo é aplicada uma estratégia top-down que começa com as definições mais gerais, e abstratas, para depois seguir com definições e caracterizações mais detalhadas e concretas. Assim, no primeiro lugar, são apresentadas, de maneira sucinta, as observações e caracterizações que algumas das áreas das ciências naturais fazem sobre sistemas adaptativos. Logo, são detalhadas algumas abordagens nas áreas tecnológicas que visam modelar o comportamento adaptativo para sua aplicação em sistemas computacionais. Finalmente, é realizada a proposta de um meta-modelo holístico para descrever sistemas com comportamento adaptativo e o procedimento para sua aplicação em sistemas legados. Adicionalmente, o capítulo demostra

a instanciação do meta-modelo para verificar a cobertura de outras abordagens e, inclusive, adaptatividade de segunda ordem.

6.1 Sistemas adaptativos: abordagens das Ciências Naturais

A habilidade dos seres vivos de ajustar suas capacidades em reação às condições ao seu redor é uma característica que tem sido foco de observação e pesquisa científica nas ciências naturais por séculos. No entanto, foi só depois da aparição dos trabalhos de Charles Darwin e Alfred Russel Wallace com o desenvolvimento da teoria da seleção natural em 1850 que essas pesquisas encontraram uma nova base para suportar o seu progresso. Como apontado em Lewontin (1978): "a teoria da evolução por seleção natural de Darwin pretendia resolver tanto o problema da origem da diversidade quanto o problema da teoria da adaptatividade de uma só vez". Assim, a partir desse momento a adaptatividade passa a ser entendida, de maneira geral no contexto da biologia evolutiva, como o mecanismo biológico pelo qual os organismos se ajustam a novos ambientes ou a mudanças em seu ambiente atual.

Mesmo quando o tema de adaptatividade, no contexto biológico, tem muitas nuances e não existe uma definição única aceita por todos, existe uma série de abordagens disponíveis e consolidadas sobre adaptatividade. A continuação apresentaremos brevemente algumas das abordagens mais importantes.

A adaptatividade de Dobzhansky

O biólogo evolutivo Theodosius Dobzhansky, na sua definição de adaptatividade, ressalta algumas observações adicionais à descrição geral do comportamento adaptativo. Ele afirma que a adaptatividade é um processo teleológico, é dizer que existe com um propósito concreto (DOBZHANSKY; DOBZHANSKY, 1970). Particularmente, em Dobzhansky (1968) o cientista registrou que: "[a adaptação biológica] está preocupada com sobrevivência e/ou reprodução, sendo encontrada [de maneira espontânea] só em corpos vivos" . Da mesma maneira, Dobzhansky também observa que nenhum organismo se adapta no abstrato, ele só pode se adaptar a alguns ambientes (DOBZHANSKY, 1968; DOBZHANSKY; DOBZHANSKY, 1970), deixando em evidência que a adaptatividade só se manifesta situada num contexto concreto e particular. Finalmente, Dobzhansky credita principalmente a seleção natural como o mecanismo interno que rege a adaptatividade biológica, complementada por algumas outras estratégias para equilibrá-la e assim garantir a diversidade genética (DOBZHANSKY, 1968).

A contextualização de Lewontin

Outro biólogo de renome, Richard Lewotin, na sua visão de adaptatividade, coloca em Lewontin (1978) "o mundo externo apresenta certos *problemas* que os organismos precisam *resolver* e a evolução através da seleção natural é o mecanismo para criar essas soluções". Nesse contexto, a adaptatividade é o processo de mudanças evolutivas pelas quais o organismo fornece uma "solução" cada vez melhor para o "problema". Lewotin reforça a importância da contextualização na adaptatividade e afirma que o próprio conceito de adaptação implica um mundo pré-existente, que apresenta alguma situação/problema para o qual a adaptação é uma solução. De fato, em Lewontin (1978), ele se aprofunda mais na natureza da interação entre o organismo e seu entorno afirmando que:

- os organismos não experimentam os ambientes de maneira passiva; eles criam e definem o ambiente no qual vivem, e

- existe uma interação constante entre o organismo e seu ambiente, de modo que, embora a seleção natural possa agir adaptando o organismo a um determinado conjunto de circunstâncias ambientais, a própria evolução do organismo altera as circunstâncias contidas no ambiente.

Esta dinâmica entre organismo e ambiente determina que não existe organismo perfeitamente adaptado, e que o processo de adaptação não tem fim (LEWONTIN, 1978). Outra observação de Lewontin reflete sobre as diferentes formas de adaptação. Ele aponta que os organismos podem se adaptar a um ambiente de diferentes maneiras, em particular ele agrupa os tipos de adaptação em: biológicas e comportamentais. A adaptação biológica significa que os organismos podem alterar as funções do corpo, e eventualmente até sua morfologia. A adaptação comportamental implica que organismos também podem erradicar ou desenvolver certos comportamentos que contribuem à sobrevivência do indivíduo ou da espécie. Com essas observações, Lewontin revela que o comportamento adaptativo pode ser manifesto em diversas esferas da vida dos organismos e não só no nível anatômico.

Finalmente, Lewontin também coloca em evidência que a adaptatividade participa na Teoria da Evolução de Darwin, influenciada pelo chamado princípio de "sobrevivência do mais apto". Segundo esse princípio, as variações que favorecem a sobrevivência de um indivíduo em competição com outros organismos e diante do estresse ambiental tendem a aumentar o sucesso reprodutivo e tendem a ser preservadas (LEWONTIN, 1978), estabelecendo assim o conceito de aptidão biológica, também conhecido como valor adaptativo ou *fitness*.

A aptidão de Reeve e Sherman

Na sua obra Reeve e Sherman (1993) os biólogos Reeve e Sherman exploram as diferentes definições de adaptatividade desde diversas escolas na biologia, alguns dos problemas de definições restritas a determinadas áreas e propõem definir a adaptatividade como a escolha de uma variante fenotípica que fornece o maior aptidão (*fitness*) dentre um conjunto especificado de variantes em um determinado ambiente. Os autores realçam a importância de entender a adaptatividade como um processo de seleção baseado em três componentes: (1) um conjunto de fenótipos, (2) uma medida de avaliação (*fitness*), e (3) um contexto ambiental. Em particular, a definição de Reeve e Sherman colocam como cidadãos de primeira classe tanto o conjunto de alternativas disponíveis (conjunto de fenótipos) quanto o processo seletivo inerente à adaptatividade. Eles alertam sobre os vários problemas decorrentes de não especificar corretamente a medida de avaliação utilizada e das análises como consequência de escolher diferentes conjuntos de fenótipos.

A Teoria de Autopoiese de Valera e Maturana

A autopoiese é uma teoria amplamente conhecida e referenciadas em diversas áreas do conhecimento, ela foi desenvolvida pelos biólogos Francisco Valera e Humberto Maturana na década do 70. O termo vem do grego auto (próprio) e poieses (criação) e significa "criar a si mesmo" ou autoprodução. A autopoiese foi formulada como resposta à pergunta sobre são as condições que constituem um ser vivo, assim ela procura descrever a organização dos sistemas vivos, seus elementos característicos e sua estrutura. Dessa maneira, a autopoiese é a capacidade particular dos seres vivos de produzirem a si próprios. Segundo esta teoria, um ser vivo é um sistema autopoiético, caracterizado como uma rede fechada de processos (produções moleculares) em que os componentes (as moléculas) produzidos geram com suas interações a mesma rede que os produziu. Essa rede de componentes é fechada em termos organizacionais, mas não em termos energéticos ou materiais, pois para realizar as suas transformações de maneira autônoma, o sistema precisa recorrer a recursos do meio ambiente com quem existe uma troca constante de energia e matéria (circularidade produtiva). Em outras palavras, a vida é um sistema autônomo, constantemente se autoproduzindo, autorregulando, e mantendo interações com o meio.

Os desafios que a teoria de autopoiese impõe estão principalmente relacionados com a sua abrangência e foco. Ao se preocupar em descrever os organismos vivos, ela pressupõe a existência de mecanismos, processos e estruturas que muitas vezes não encontram paralelo em sistemas computacionais. Além disso, a autopoiese apresenta um volume grande de informações detalhadas utilizando terminologias da biologia, requerendo um nível de abstração elevado para ser analisadas e encontrar analogias em diversos campos. A autopoiese se dedica à descrição da essência da vida como fenômeno sistêmico e estuda sua

organização, processos e características, assim, suas preocupações são mais abrangentes do que aquelas pretendidas pela adaptatividade, e em consequência acarretam argumentos e condições mais complexas. Porém, no que concerne à adaptatividade, a autopoiese fornece só a informação mínima para compreender que a capacidade de mudança constante dos seres vivos constitui uma característica essencial para o aparecimento de vida. Além disso, os conceitos na teoria de autopoiese precisam de uma transformação significativa (compreensão, abstração e reinterpretação) para serem aplicados em outras áreas, no entanto, a adaptatividade precisa de um marco conceitual geral de fácil aplicação em qualquer área.

O fenômeno do comportamento adaptativo tem sido pesquisado na Biologia por mais de 170 anos, e como resultado desse interesse existem diversas fontes sobre o assunto. Algumas delas expõem várias das abordagens estudadas ao longo da história, por exemplo, Shanahan (2004) e Brock (2000). Outras, como Ruse e Travis (2009), Dobzhansky e Dobzhansky (1970) apresentam uma visão mais geral da adaptatividade. A vasta literatura sobre o assunto também difere no nível de detalhe e o conhecimento prévio esperado do público alvo. Mesmo quando uma revisão mais detalhada sobre o assunto está fora do escopo deste trabalho, a consulta de literatura sobre adaptatividade nas áreas naturais é altamente construtiva e recomendada.

6.2 Sistemas adaptativos: abordagens tecnológicas

A seguir apresentamos algumas das abordagens nas áreas tecnológicas que modelam o comportamento adaptativo em sistemas computacionais desde diferentes perspectivas. Em cada caso descrevemos o modelo geral proposto na literatura e recomendamos algumas referências importantes para o leitor interessado.

6.2.1 *Self-adaptive Systems* ou *SaS*

Um sistema self-* é um sistema que pode lidar com mudanças e incertezas percebidas tanto no seu ambiente quanto no próprio sistema e/ou nos seus objetivos, ajustando seu comportamento de forma autônoma (entendido como sem ou com mínima interferência humana) (BRUN *et al.*, 2009; CHENG *et al.*, 2009b; WEYNS, 2017). Um sistema self-* se caracteriza por manter pelo menos um aspecto da sua operação automaticamente para assim aliviar parcialmente o peso que os sistemas, cada vez de maior tamanho e complexidade, colocam sobre os gestores de TI. O aspecto que é automatizado é denominado propriedade self-*, e faz parte do conjunto de características que definem o comportamento de alguns sistemas complexos (POSLAD, 2009). Em particular, o prefixo *self-* (auto em português) ressalta a natureza autônoma da propriedade, o que significa que o sistema tem o poder de decisão, ação e controle sobre esta propriedade.

As propriedades self-*, mesmo que particularmente aplicadas na descrição de entidades tecnológicas, foram observadas pela primeira vez em organismos naturais (HOLLAND, 1992) que as utilizam eficientemente para atingir objetivos específicos ou superar dificuldades. Assim, esta abordagem também é uma abordagem bio inspirada. No contexto tecnológico, existem diferentes maneiras de organizar o estudo das várias propriedades self-*, dependendo da granularidade com que se deseja observar o fenômeno em estudo. No entanto, uma categorização geral é a hierarquia proposta na Figura 39 reúne às propriedades self-* mais referenciadas na literatura.

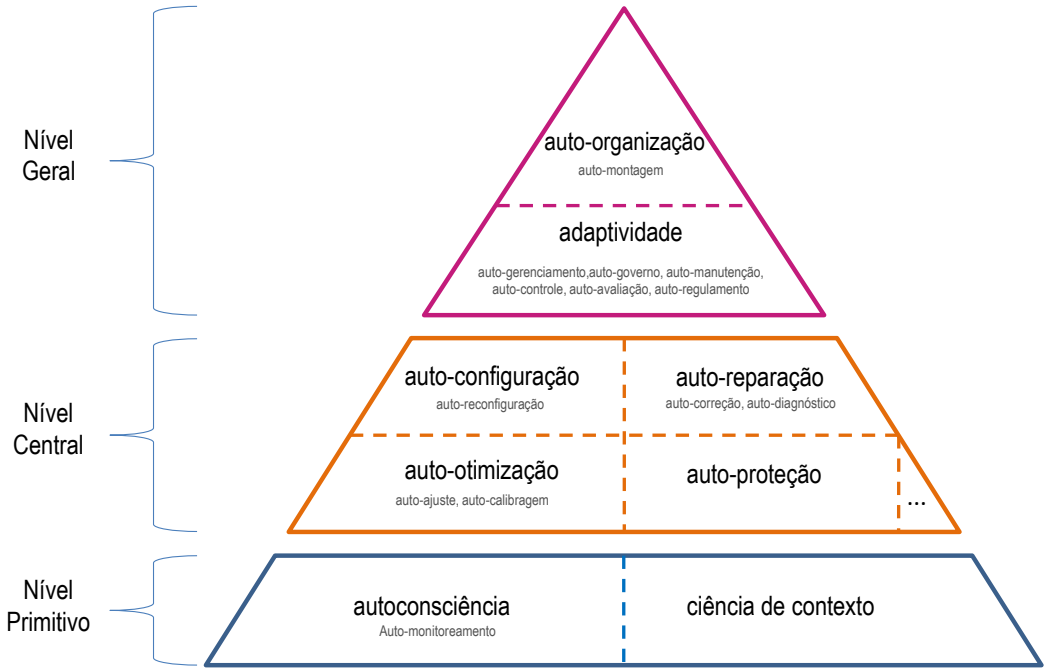


Figura 39 – Hierarquia de Propriedades Self-*.
Adaptada de Salehie e Tahvildari (2009), Bradbury *et al.* (2004), Berns e Ghosh (2009), Rahman *et al.* (2011).

As propriedades no nível primitivo dessa hierarquia são: a autoconsciência (*self-awareness*) e a ciência de contexto (*context-awareness*), ambas condições necessárias para que qualquer sistema seja processar algum tipo de mudança, pois sem elas não é possível sequer identificar a própria existência da mudança.

As propriedades no nível central da hierarquia desempenham funções mais complexas que aquelas no nível primitivo, incorporando mecanismos mais avançados de avaliação e controle. Estas propriedades usualmente são desenvolvidas seguindo um cuidadoso processo que pode implicar o uso de modelos, políticas, regras e métricas. Neste nível as quatro

propriedades mais referenciadas são: autoconfiguração (*self-configuration*), auto-otimização (*self-optimizing*), autorreparo (*self-healing*) e autoproteção (*self-protecting*). Contudo, nas últimas duas décadas o conjunto de propriedades self-* requeridas nos sistemas computacionais têm crescido substancialmente (STERRITT *et al.*, 2005b; STERRITT *et al.*, 2005a; ABEYWICKRAMA; OVASKA, 2017), criando o termo "*selfware*" para caracterizar esse conjunto de propriedades.

As propriedades no nível geral, a auto-organização (*self-organizing*) e a auto-adaptatividade (*self-adaptivity*), são aquelas de maior complexidade nos sistemas que processam mudanças e, geralmente, são desempenhadas e observadas por entidades sofisticadas que exercem controle sobre funcionalidades em diferentes partes do sistema. Por um lado, a auto-organização é a capacidade de um sistema de adquirir uma estrutura (ou organização) espacial, temporal ou funcional sem controle explícito ou restrições externas ao sistema. Isto é, a organização é intrínseca ao sistema e resulta apenas de restrições ou mecanismos internos, e como consequência das interações locais entre seus componentes. Por outro lado, a adaptatividade é entendida como a capacidade de uma entidade de avaliar o próprio comportamento e modificá-lo quando uma avaliação em particular indique que não está sendo atingido o objetivo (LADDAGA, 1998). Isto significa que a entidade consegue processar uma mudança dinâmica de maneira autônoma e inteligente, ajustando algumas características do seu comportamento conforme o ambiente no qual se desenvolve. As mudanças podem ser de naturezas diversas e com diferentes objetivos, portanto, podemos dizer que a adaptatividade está contida nas outras propriedades self-* mais específicas (WARRIACH; ÖZçELEBI; LUKKIEN, 2014).

Assim, é possível observar, que os sistemas *SaS*(*Self-Adaptive Software* ou *Self-Adaptive Systems*) permeiam um conjunto amplo de propriedades e os seus comportamentos particulares. Por essa razão, a modelagem e implementação concreta deles pode variar significativamente de um sistema para o outro. Ainda mais, uma abordagem geral para o desenvolvimento de *SaS* só pode apontar os elementos básicos, e as condições e mecanismos principais que fazem parte da sua estrutura principal. Dentre a literatura que apresenta abordagens gerais se destacam, de maneira particular, os trabalhos em Weyns (2017) e Salehie e Tahvildari (2009). Em Weyns (2017), o autor apresenta os conceitos básicos relacionados com a construção de *SaS* e uma revisão histórica da evolução das abordagens desenvolvidas para a construção deste tipo de sistemas. A revisão é organizada em 6 "ondas" ou gerações que permitem entender as necessidades que deram origem a cada geração, sua principal motivação e os enfoques tecnológicos desenvolvidos para resolver os questionamentos levantados. Além disso, Weyns propõe um modelo conceitual para os *SaS*, representado na Figura 40, onde identifica quatro elementos principais: (1) um sistema gerenciado, (2) um sistema gerenciador, (3) o ambiente de execução, e (4) os objetivos da adaptatividade, que estão conectados via mecanismos de interação.

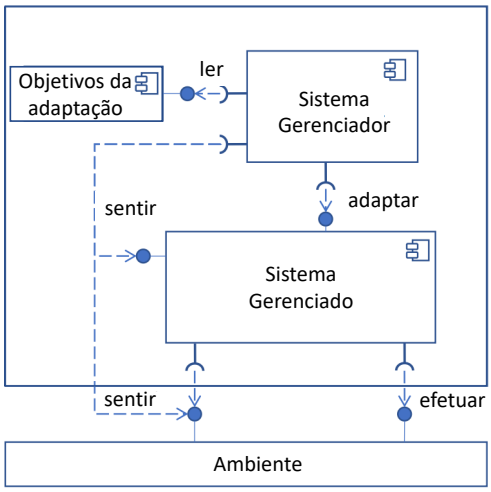


Figura 40 – Modelo Conceitual dos SAS.
Adaptada de Weyns (2017).

O trabalho em Salehie e Tahvildari (2009) apresenta uma visão complementar ao modelo conceitual de Weyns oferecendo um olhar mais detalhado nos processos adaptativos presentes nos SaS, representados na Figura 41. Nesse trabalho os autores identificam quatro processos necessários para o funcionamento dos *SaS*, eles são: monitoramento, detecção, decisão e atuação, esses processos descrevem um ciclo fechado sempre em funcionamento.

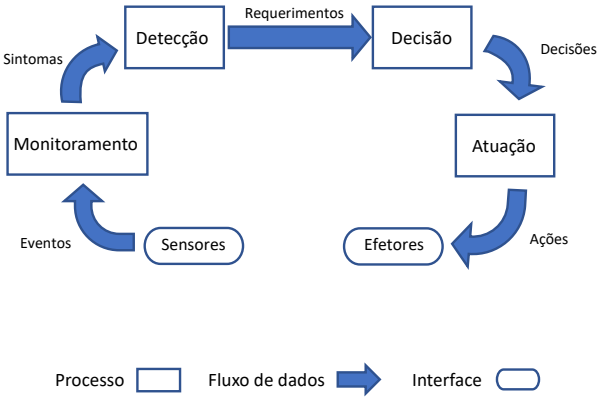


Figura 41 – Processos participantes da adaptatividade em *SaS*.
Adaptada de Salehie e Tahvildari (2009).

Na literatura esses processos têm recebido nomes diferentes, por exemplo: observação, analise, e adaptação em Rohr *et al.* (2006), e coletar, analisar, decidir e agir em Cheng *et al.* (2009b). No entanto, as tarefas que eles desempenham são essencialmente as mesmas, a Figura 42 é um exemplo dessa simetria e correlação. Nessa figura é possível observar que ambas abordagens, a de componentes de Weyns e a de processos de Cheng et al., apresentam um desacoplamento entre as decisões de implementação e implantação, podendo ser considerados meta-modelos para *SaS*.

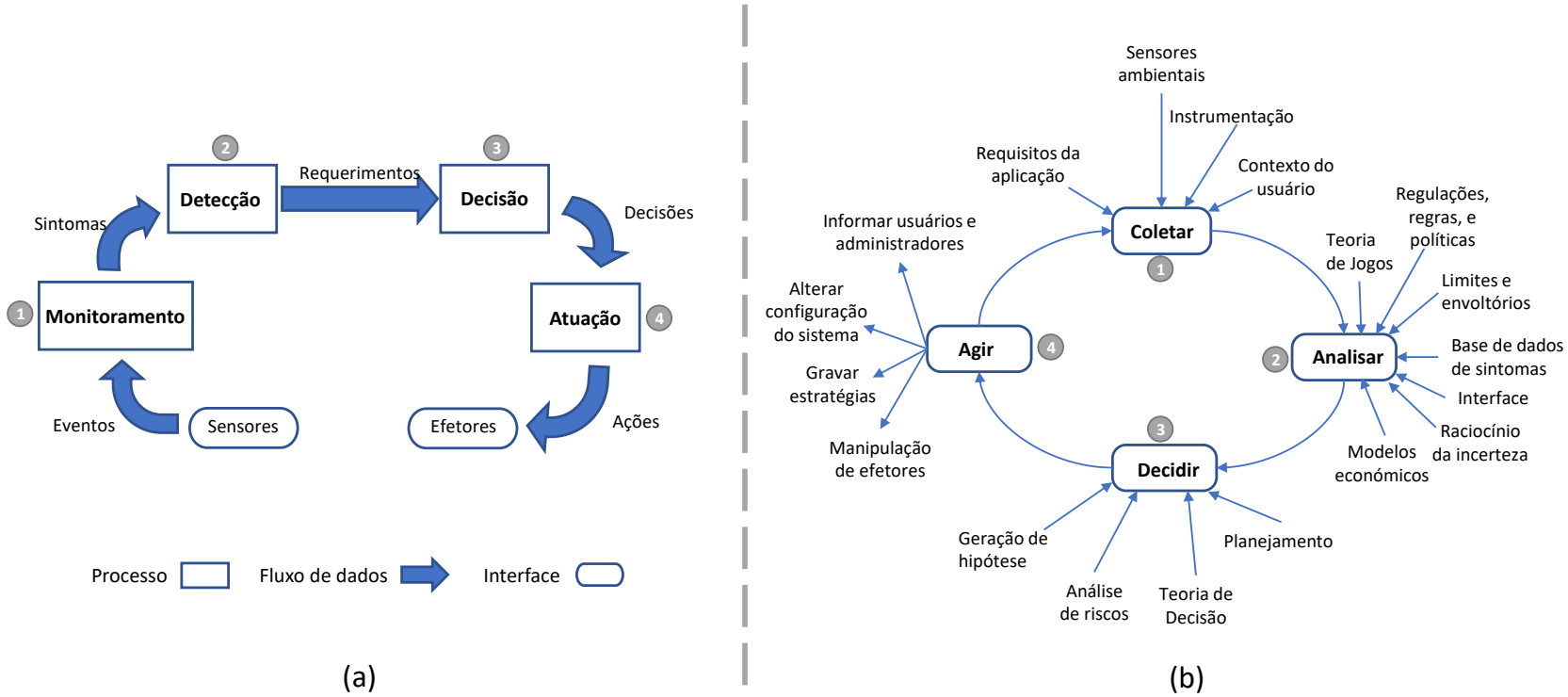


Figura 42 – Comparação entre os modelos de (a) processos no Ciclo Adaptativo em Salehie e Tahvildari (2009) e os (b) Processos no ciclo de controle de Cheng *et al.* (2009b).

Adaptada de Salehie e Tahvildari (2009) e Cheng *et al.* (2009b).

Para o planejamento e implementação dos *SaS* têm sido desenvolvidas e aplicadas diversas técnicas e tecnologias (ROBERTSON; LADDAGA; SHROBE, 2000) com a finalidade de suportar o processo de criação deste tipo de sistemas em diferentes áreas tecnológicas. Em consequência, existe uma diversidade de modelos que têm sido utilizados para o desenvolvimento de *SaS*. No entanto, eles podem ser classificados em três perspectivas diferentes: (a) modelos baseados em especificações, (b) modelos baseados na arquitetura e (c) modelos baseados no design.

Desde a perspectiva de modelos baseados em especificações, descrições formais sobre o comportamento adaptativo têm sido desenvolvidas com base em diversos formalismos e técnicas, por exemplo: teoria de grafos, álgebras regulares e álgebras de processos (como pi-calculus), máquinas de estados (NETO, 2002) (autômatos e statecharts), mecanismos de reflexão (ANDERSSON *et al.*, 2009) e de tratamento de exceções, provadores de teoremas (LADDAGA; ROBERTSON, 2004) e diferentes tipos de lógica e programação declarativa.

Na perspectiva de modelos arquitetônicos, uma série de propostas está sendo desenvolvida. Alguns deles baseiam-se na tradução das especificações para a criação de modelos através de ADL's (*Architecture Design Language*) usando métodos formais para desenvolver padrões funcionais, como Stich (CHENG; GARLAN, 2012) e ACME (GARLAN; SCHMERL; CHENG, 2009). Alguns outros criam arquiteturas para fornecer uma melhor interação entre os elementos do comportamento adaptativo, como na arquitetura MAPE-K da IBM (KEPHART; CHESS, 2003). Outros aproveitam modelos existentes que podem fornecer uma base para a adaptatividade como no caso do modelo Belief-Desires-Intentions (BDI) dos sistemas multiagentes, ou dos modelos de tempo de execução (*models@runtime*) (AMOUI *et al.*, 2012) ou da teoria de controle (WEYNS; MALEK; ANDERSSON, 2012; WHITTLE *et al.*, 2009; KOKAR; BACLAWSKI; ERACAR, 1999). Outros buscam misturar tanto o enfoque formal quanto a arquitetura para explorar a sinergia entre eles, alguns exemplos são: a arquitetura em camadas em Kramer e Magee (2007), FORMS de Weyns, Malek e Andersson (2012), os modelos baseados na Arquitetura baseada em Serviços (SOA), como MUSIC (HALLSTEINSEN *et al.*, 2012) e Rainbow (GARLAN *et al.*, 2004; GARLAN; SCHMERL; CHENG, 2009), e os modelos baseados nas arquiteturas de Sistemas de Sistemas (SoS).

Do ponto de vista do design, como mencionado em Kokar, Baclawski e Eracar (1999), Laddaga (1998), Laddaga e Robertson (2004) e Laddaga (2006), três metáforas têm sido úteis para os pesquisadores em software adaptativo: (1) codificar um *SaS* como um sistema de planejamento dinâmico (BONET; GEFFNER, 2001; GHALLAB; NAU; TRAVERSO, 2004; PANDEY *et al.*, 2017), (2) codificar um *SaS* como um sistema de controle (LADDAGA, 1998; KOKAR; BACLAWSKI; ERACAR, 1999; PATIKIRIKORALA

et al., 2012; VROMANT *et al.*, 2011), ou (3) codificá-lo como um sistema autoconsciente (*Self-aware*) (KOUNEV *et al.*, 2017; ALEGRE; AUGUSTO; CLARK, 2016). Em cada caso, os recursos dos sistemas self- * são mapeados na estrutura do paradigma (sistema de planejamento, teoria de controle ou autoconsciente), visando que alguns discernimentos e técnicas valiosos possam ser emprestados para o desenvolvimento de sistemas adaptativos.

A literatura a respeito de *SaS* é ampla e em vários casos os trabalhos estudam propostas para a implementação de adaptatividade em contextos particulares e com finalidades específicas. No entanto, existem alguns trabalhos que permitem ter uma visão geral sobre o estado da arte dos *SaS* na área de engenharia de software. Assim, o leitor é referido à seguinte bibliografia:

(a) Surveys Classificações e Taxonomias: Salehie e Tahvildari (2009), Macías-Escrivá *et al.* (2013), Krupitzer *et al.* (2015), Rohr *et al.* (2006) Os autores analisam os diferentes aspectos que caracterizam os *SaS* tanto conceitualmente como na sua implementação.

(b) Roteiros de pesquisa:Lemos *et al.* (2013a), Cheng *et al.* (2009b) apresentam o resultado do trabalho colaborativo realizado nos Dagstuhl Seminars em 2008 (CHENG *et al.*, 2009a) e 2010 (LEMOS *et al.*, 2013b) onde a comunidade de Engenharia de Software propõe e analisa modelos e padrões que respondem às necessidades no desenvolvimento de *SaS* e os principais desafios para esses sistemas.

(c) Revisões sistemáticas: Yang *et al.* (2017), Weyns *et al.* (2012), Sucipto e Wahono (2015), Weyns, Malek e Andersson (2012), Whittle *et al.* (2009), Kephart e Chess (2003) apresentam com detalhe as diferentes técnicas que tem sido utilizadas para a modelagem, implementação e avaliação dos SaS.

6.2.2 Computação Autonômica

Computação Autonômica, ou *(Autonomic Computing)* (AC), é um termo adotado pela IBM em 2001 quando o vice-presidente da companhia, Paul Horn, apresentou a ideia em um discurso na Universidade de Harvard (NAMI; BERTELS, 2007) como resposta à dificuldade e complexidade no gerenciamento, configuração e uso de tecnologias modernas. Na proposta da IBM, um sistema de computação autonômico é aquele que pode se autogerenciar atendendo as metas estabelecidas pelos seus administradores humanos (HORN, 2001; KEPHART; CHESS, 2003; IBM Corporation, 2005). Nesse sentido, o termo faz referência à capacidade observada no Sistema Nervoso Autônomo para governar e regular todo um conjunto de funções corporais, tais como o batimento cardíaco, temperatura corporal, fluxo sanguíneo e outras ações de maneira “involuntária”, quer dizer, sem a necessidade de

envolver a consciência humana (HORN, 2001; NAMI; BERTELS, 2007; SINGH *et al.*, 2011).

De acordo com essa linha, os sistemas de computação autonômico devem possuir as seguintes características:

- Compreender o alto nível de intenção e direção atribuído por seus operadores humanos ou por outro sistema, indicado como responsável pela tarefa, com o qual o sistema autônomo se comunica;
- Decidir por curso de ação, entre várias alternativas, sem depender da supervisão e controle humanos;
- Ter uma atitude concordante com os seus próprios objetivos, percepções, estados internos e conhecimento, sem intervenção externa;
- Percepção do seu ambiente de ação;
- Suportar algumas das propriedades *self-**. Em particular, a propriedade de adaptatividade (*self-adaptivity*) é considerada uma característica principal dos sistemas autonômicos (BRUNI *et al.*, 2012) podendo ser especializada em outras propriedades self-*.

Assim, entende-se que um sistema de computação autonômico é criado para funcionar integrando sensoriamento, percepção, análise, comunicação, planejamento, tomada de decisões e ações para atingir seus objetivos. Desta maneira, os sistemas de computação autonômica são capazes de executar a si mesmos, de se ajustar a diferentes circunstâncias e preparar os seus recursos para atenderem devidamente à carga de trabalho que os administradores de Tecnologias de Informação (TI) colocam sobre eles (HORN, 2001). Esta abordagem está, por tanto, especialmente orientada a facilitar a gestão dos especialistas de TI, permitindo que os próprios sistemas realizem a gestão das tarefas de baixo nível, por exemplo: manutenção de software, enquanto os especialistas concentram seus recursos no atendimento e resolução de tarefas de alto nível.

Para atingir este objetivo, a formulação apresentada pela IBM descreve 8 características principais deste tipo de sistemas, as quais podem ser analisadas em três grupos diferentes, como mostra a Figura 43: aquelas que representam a ciência *(awareness)* do sistema, aquelas que se referem às propriedades self-*, e aquelas que gerenciam a interação com outros elementos.

A implementação de sistemas de computação autonômicos em IBM Corporation (2005) IBM propõe uma organização hierárquica de diferentes elementos, chamados *building blocks*, governados pelos administradores de TI. É importante mencionar que são os

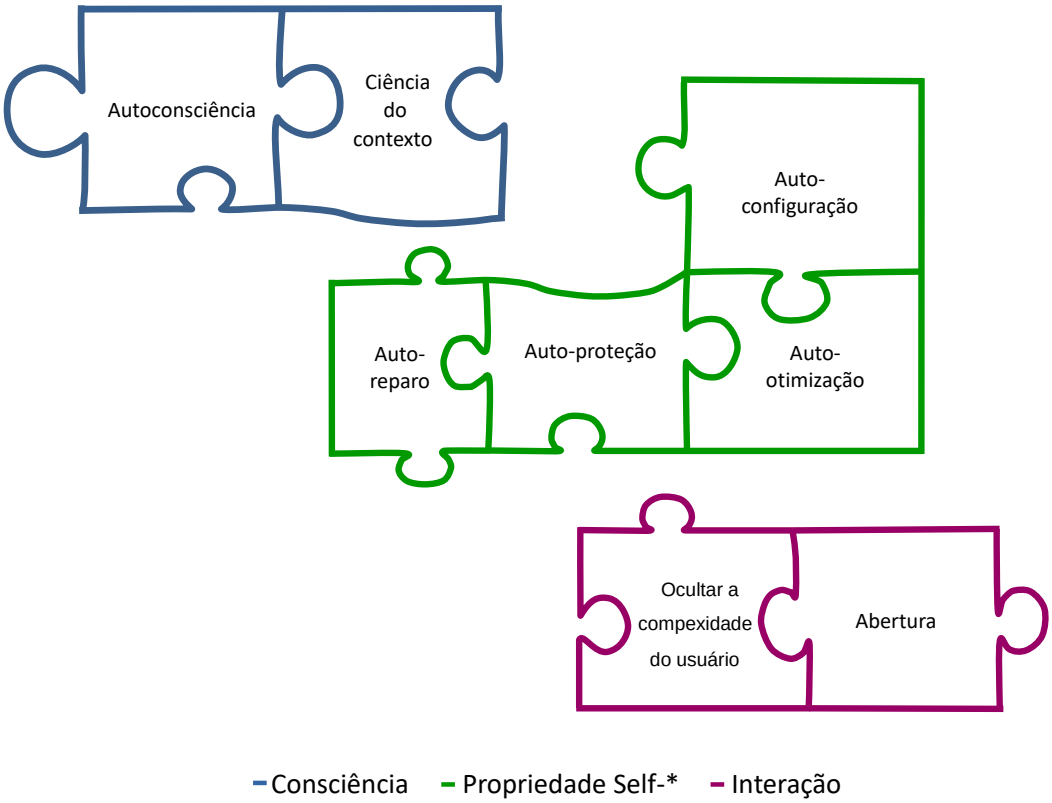


Figura 43 – Características principais da Computação Autonômica por grupo. O Autor (2018).

profissionais de TI que escolhem quais tarefas delegar para os sistemas autonômicos e decidem a arquitetura concreta do sistema. A Figura 44, extraída de (IBM Corporation, 2005), apresenta uma arquitetura hierárquica com a participação dos diferentes tipos de *building blocks*:

- Recurso Gerenciado: é qualquer tipo de componente, de hardware ou software, que pode ser gerenciado e que compõe a infraestrutura de TI do sistema.
- *Touchpoints*: são interfaces de gerenciamento, também chamadas de pontos de controle (LALANDA; MCCANN; DIACONESCU, 2013), para monitoramento e adaptação. Em particular, dois tipos de *touchpoints* devem ser diferenciados: sensores e efetores, os primeiros permitem recolher informações sobre os recursos enquanto os segundos viabilizam ações de reconfiguração. É através da definição explícita de sensores e efetores que o encapsulamento adequado dos recursos gerenciados pode ser garantido.
- Gerente autonômico(*(autonomic manager)*): é um componente do sistema que implementa um ou mais ciclos de controle inteligente, também chamado de ciclos autonômicos (LALANDA; MCCANN; DIACONESCU, 2013). Ele percebe a situação atual do sistema, o seu estado interno e o contexto externo, para logo determinar

as ações de gerenciamento apropriadas conforme as diretrizes fornecidas e acompanhar sua execução. Segundo a proposta da IBM existem dois tipos de gerentes autonômicos:

- Gerentes autonômicos de *touchpoint*: são aqueles que trabalham diretamente com os recursos gerenciados através de seus *touchpoints*. Eles estão em contato direto com os recursos gerenciados.
- Gerentes autônomos de orquestração: são aqueles que trabalham coordenando os gerentes autonômicos de *touchpoint* para fornecer um comportamento de computação autônoma em todo o sistema. Ao contrário dos gerentes autonômicos de *touchpoint* que agem isoladamente nos recursos que ele gerencia, os gerentes autonômicos de orquestração procuram fornecer uma característica global ao sistema.

Adicionalmente, a arquitetura inclui explicitamente a participação dos profissionais de TI via uma interface denominada gerente manual. A interface que permite realizar intervenções pontuais. Assim mesmo, as várias camadas de gerenciadores manuais e autônomos podem obter e compartilhar conhecimento por meio de fontes de conhecimento.

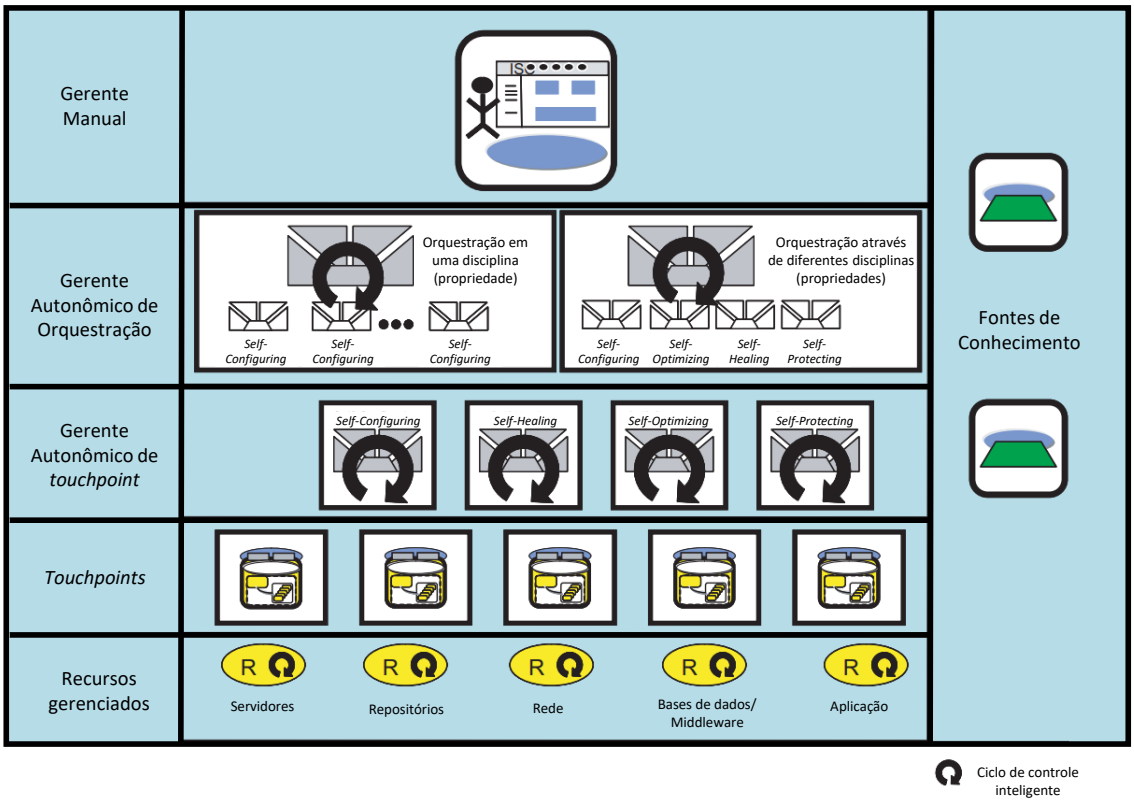


Figura 44 – Arquitetura geral dos sistemas de Computação Autonômica. Adaptada de IBM Corporation (2005).

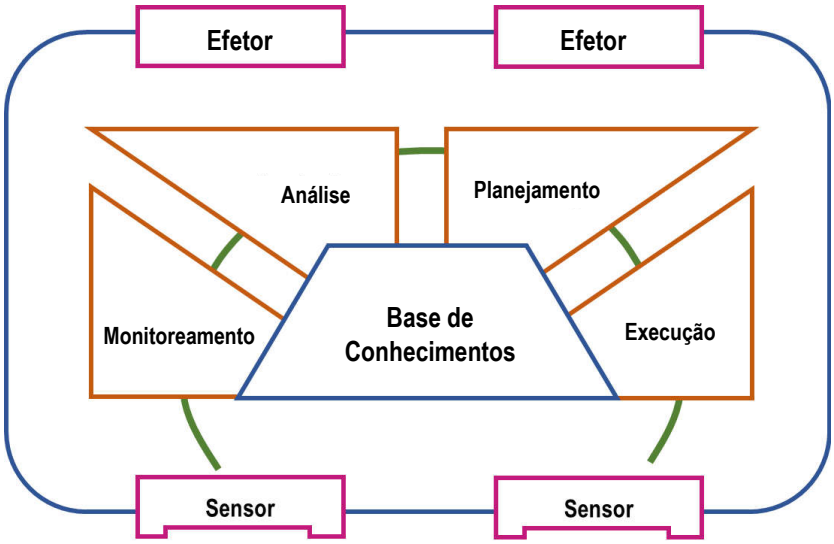


Figura 45 – Arquitetura MAPE-K.
Adaptada de IBM Corporation (2005).

Na Computação Autonômica, o elemento responsável por fornecer adaptatividade ao sistema é o gerente autonômico através da implementação um laço de controle inteligente, conhecido como a arquitetura MAPE-K. A arquitetura MAPE-K como proposta pela IBM, representada na Figura 45, conta com os seguintes componentes:

- Um método automatizado para coletar os detalhes necessários dos recursos gerenciados, através de interfaces com sensores e relacioná-los com sintomas que possam ser analisados. Essa função é chamada **monitoramento**.

- Um mecanismo automatizado para observar e analisar situações para identificar a necessidade de realização de alguma mudança. O processo de tomada de decisão do gestor autonômico tem que raciocinar sobre o presente, talvez sobre o passado e possivelmente até o futuro para compreender corretamente uma situação atual. Assim sendo, em vários casos, esses mecanismos modelam comportamentos complexos e usam técnicas para identificar tendências e padrões recorrentes. Esses mecanismos permitem que o sistema autônomo aprenda sobre o ambiente, preveja comportamentos futuros e até considere as consequências de suas várias ações sobre os recursos gerenciados, por exemplo, via simulação (LALANDA; MCCANN; DIACONESCU, 2013). Essa função é chamada **análise**.

- Um mecanismo automatizado para a construção das ações necessárias para executar uma alteração desejada no recurso gerenciado, a estratégia de mudança, e a trans-

missão lógica desse conjunto de ações para a função de execução. O conjunto de ações e estratégias pode ser desde um simples comando até um fluxo de trabalho complexo. Este mecanismo é conhecido como **planejamento**.

- Um mecanismo automatizado para programar, controlar e executar as ações do plano de mudança em um ou mais recursos gerenciados através das interfaces dos efetores. Esta função é conhecida como **execução**.

Esses mecanismos se comunicam e colaboram um com outro intercambiando dados e conhecimentos apropriados. As informações que conformam a base de conhecimento no centro do gerenciador autônomo podem ser obtidas de várias maneiras, como detalhado em IBM Corporation (2005). A base de conhecimentos consiste em tipos particulares de dados, os quais podem ser acessados e/ou modificados pelos mecanismos do gerenciador. Alguns exemplos desses dados podem ser: políticas, sintomas, métricas e logs.

Como apontado em Abuseta e Swesi (2015) desde sua concepção, a arquitetura MAPE-K (Monitoramento-Análise-Planejamento-Execução-Conhecimento) tem sido aplicada para projetar sistemas além da computação autônoma (KING *et al.*, 2007), funcionando como um modelo arquitetônico para a construção de sistemas que procuram desenvolver comportamentos adaptativos.

Literatura mais detalhada sobre Computação Autonômica que o leitor pode consultar são:

- A respeito da integração de diferentes abordagens com Computação Autonômica: Viroli *et al.* (2016) apresenta uma proposta para o desenvolvimento de Sistemas Adaptativos Coletivos inspirada no ciclo MAPE-K. Em Ramirez *et al.* (2011) é apresentada uma proposta de AC que incorpora algoritmos genéticos para assistir à tomada de decisão. Em Liu e Schmeck (2006) os autores propõem uma arquitetura para o desenvolvimento de sistemas de computação autônomos no contexto de arquitetura orientada a serviços e suas possíveis vantagens.

- Visão geral e detalhada de Computação Autonômica: em Kephart e Chess (2003) os autores apresentam uma visão mais aprofundada sobre o desenvolvimento de sistemas de computação autonômicos e alguns dos desafios relacionados ao desenvolvimento de software com esta abordagem. Em Kephart (2005) o autor apresenta uma visão nos desafios de pesquisa para Computação Autonômica. O trabalho em Lalanda, McCann e Diaconescu (2013) apresenta de maneira muito didática não só as noções básicas por trás da Computação Autonômica, mas também algumas das suas fontes de inspiração e influências, e as diferentes interações que a Computação Autonômica pode ter com outras abordagens para o desenvolvimento de software como: Inteligência Artificial,

Sistemas Complexos, Sistemas Multiagentes. Da mesma forma, os autores oferecem uma análise sobre os métodos de avaliação em Computação Autonômica e outros desafios futuros.

- Surveys e Taxonomias: Surveys de Computação Autonômica, em geral, (HUEBSCHER; MCCANN, 2008; NAMI; BERTELS, 2007), Surveys de Computação Autonômica no contexto de Computação em Grade (RAHMAN *et al.*, 2011), Survey de Computação Autonômica e Ecossistemas de Serviços Digitais (ABEYWICKRAMA; OVASKA, 2017).

6.2.3 Sistemas Complexos

Segundo Holland (2014) o conceito de *complexidade* não tem uma definição exata e aceita de maneira abrangente, e, por tanto, a demarcação nítida e distintiva entre sistemas complicados e sistemas complexos é um desafio, pois existe um meio-termo em que a distinção se torna escura e arbitrária. No Apêndice B apresentamos uma análise resumida das principais diferenças entre as três principais categorias de sistemas: sistemas simples, complicados e complexos. No entanto, essa variedade de definições não impede uma abordagem rigorosa do assunto, pelo contrário, existem enfoques formais que se dedicam ao estudo teórico de sistemas complexos.

Gramaticalmente, o termo complexo deriva do Latim *"plexus"* e significa trançado ou enlaçado junto. Dessa maneira, a complexidade resulta da inter-relação, interação e interconectividade dos elementos num sistema, assim como também, entre um sistema e seu ambiente.

Um sistema é considerado complexo se ele tem muitos componentes, possivelmente heterogêneos ou diversos, sem controle central, que colaboram para criar um todo funcional com um comportamento coletivo complexo, processamento sofisticado de informações e adaptação via aprendizado ou evolução (MITCHELL, 2009). O funcionamento de tal sistema é governado pelas interações dinâmicas interdependentes (PAGE, 2010), tanto entre os componentes internos do sistema quanto entre sistema e ambiente, e não pode ser completamente entendido com a descrição e análise individual dos componentes, como afirma a abordagem reducionista (CILLIERS, 2002; SOMMERVILLE *et al.*, 2012). Esta característica é chamada de interações não lineares e resume a diferença entre os sistemas complicados e os sistemas complexos por dois fatores: (1) a descoberta de propriedades emergentes [1] como consequência da interação entre as partes, enquadrado pela estrutura

[1] É importante notar que em nosso trabalho o termo "emergência", frequentemente mencionado

hierárquica que constrói o sistema (diferentes níveis de organização), e (b) a incerteza na previsão do comportamento do sistema devido às mudanças contínuas em funcionalidade e estrutura (HOLLAND, 2014; CILLIERS, 2002).

Tradicionalmente, a literatura em complexidade tendeu a vir de domínios científicos que observam os sistemas de um ponto de vista sob uma perspectiva sistêmica, tais como: ciências sociais e naturais, biologia, sociologia e filosofia que, frequentemente, têm questionado comportamentos aflorantes, adaptatividade e evolução. No entanto, outras áreas do conhecimento têm recorrido para a complexidade como um recurso para obter respostas a questões que de outra forma permaneceriam inacessíveis, e para oferecer uma chave para novos tipos de entendimento (HOLLAND, 2014). Desde meados dos 80s algumas instituições, como o Instituto de Santa Fé em Novo México, têm realizado trabalhos em que estudam as propriedades gerais dos sistemas complexos, cruzando as fronteiras das disciplinas tradicionais. Na atualidade, os trabalhos sobre complexidade têm se tornado altamente interdisciplinares, suscitando a criação de pontes entre várias áreas.

A Teoria de Sistemas Complexos se dedica ao estudo de sistemas complexos para tentar obter uma compreensão sobre as propriedades, princípios e padrões aflorantes que reaparecem em diferentes sistemas e ambientes complexos tanto naturais como artificiais. O seu objetivo é definir com mais precisão essas propriedades para uma maior compreensão dos sistemas complexos como uma entidade, além da compreensão de um sistema específico ou de um conceito de projeto específico. Formalmente, o estudo de Sistemas Complexos pode ser agrupado em: sistemas físicos complexos, e sistemas adaptativos complexos (HOLLAND, 2014; PAGE, 2010). O primeiro dedica-se a estudar sistemas que seguem regras fixas, como o estudo de Sistemas Físicos Complexos ou *(Complex Physical Systems)*(CPS) nos quais os sistemas se regem às leis da física. O segundo refere-se a sistemas nos quais os elementos que os compõem se adaptam. Tais sistemas são chamados Sistemas Adaptativos Complexos ou *(Complex Adaptive Systems)*(CAS) (HOLLAND, 2014). Os CAS estão compostos por agentes heterogêneos que possuem a capacidade de aprender, se adaptar ou evoluir em resposta às interações com outros agentes. Nos CAS, tanto os agentes quanto o

na literatura, é substituído pelo termo "afloração" com a intenção de preservar a clareza semântica, e evitar conflitos terminológicos entre áreas. O termo emergência está fortemente associado, especialmente fora da área de sistemas complexos, com a aparição imprevisível e sem precedente de entidades ou comportamentos que não existiam previamente. No entanto, no caso dos sistemas complexos a emergência deve ser entendida como a observação inesperada de características, padrões ou comportamentos no sistema estudado como consequência da interação entre os seus elementos. Tais características não eram perceptíveis em estágios anteriores devido, por exemplo, à diferença na granularidade da observação, tecnologia, intervalo de tempo ou viabilidade. Assim, a manifestação destas características é inesperada e não explicitamente introduzida no sistema. Porém, a não é assumida a não existência de tais características nas instâncias prévias à sua percepção por parte do sistema

sistema são adaptativos. Exemplos de CAS incluem ecossistemas, colônias de formigas e outros insetos sociais, o sistema imunológico, sistemas financeiros, mercados de ações e comunidades políticas e culturais. O funcionamento dos agentes participantes nos CAS é descrito como um conjunto de regras, adaptadas segundo a experiência previamente acumulada, de maneira que consigam utilizá-la para melhorar o seu desempenho e sua oportunidade de sobrevivência. A versatilidade do que um agente pode representar é muito grande, e soluções a alguns dos mais importantes problemas do século XXI dependem do entendimento profundo das interações nos CAS. Consequentemente, a técnica mais direta para o planejamento e implementação de CAS é o modelo baseado em agentes. Segundo este modelo, um agente é um sistema individual que age sobre seu ambiente em resposta aos eventos que vivência. Exemplos de agentes são pessoas, empresas, animais, células e moléculas (HEYLIGHEN, 2009). O número de agentes no CAS geralmente não é fixo, pois os agentes podem se multiplicar, "morrer", se associar ou separar do sistema. Normalmente, os agentes são modelados segundo um paradigma orientado a objetivos: suas ações visam maximizar sua aptidão, utilidade ou preferência individual (HEYLIGHEN, 2009). Na falta de um objetivo específico, sua atividade segue uma lógica simples, compilada num conjunto de regras básicas no formato de causa-efeito ou condição-ação que define suas ações e interações com outros agentes e o ambiente (HOLLAND, 1995). Tais regras, embora inicialmente fixadas para um certo tipo de agente, podem mudar em alguns casos por efeito do aprendizado ou variação evolutiva. Este modelo acredita que todas as interações dentro dos sistemas começam a formar padrões que, pela sua vez, retroalimentam o sistema e influenciam ainda mais as interações dos agentes. Assim, as condições ambientais às quais um agente reage são normalmente afetadas pela atividade de outros agentes, quer dizer que a ação de um agente geralmente acionará ações de outros agentes, ativando uma cadeia de atividades que se propaga de agente para agente através do sistema. Tais interações são inicialmente locais: começam afetando apenas os agentes na vizinhança imediata do ator inicial. No entanto, suas consequências são muitas vezes globais, afetando o sistema de agentes como um todo.

O modelo de agentes serve de plataforma para imitar as características não-lineares dos sistemas complexos no mundo real, mantendo a identidade dos elementos e as estruturas envolvidas. A Figura 46 mostra os elementos básicos envolvidos no modelo baseado em agentes para a construção de CAS: uma coleção de agentes heterogêneos e seus estados, as regras que governam suas interações e o ambiente em que eles vivem (SHALIZI, 2006). A representação foi montada com base nas informações de diferentes trabalhos na literatura, principalmente em Holland (1995) e THE HEALTH FOUNDATION (2010).

Como indicado por Kurka e Pitt (2017) o modelo básico de agentes pode ser complementado outros elementos, por exemplo, normas e políticas, que permitem auxiliar os agentes no seu desempenho. A Figura 47 apresenta um esquema de regras em quatro

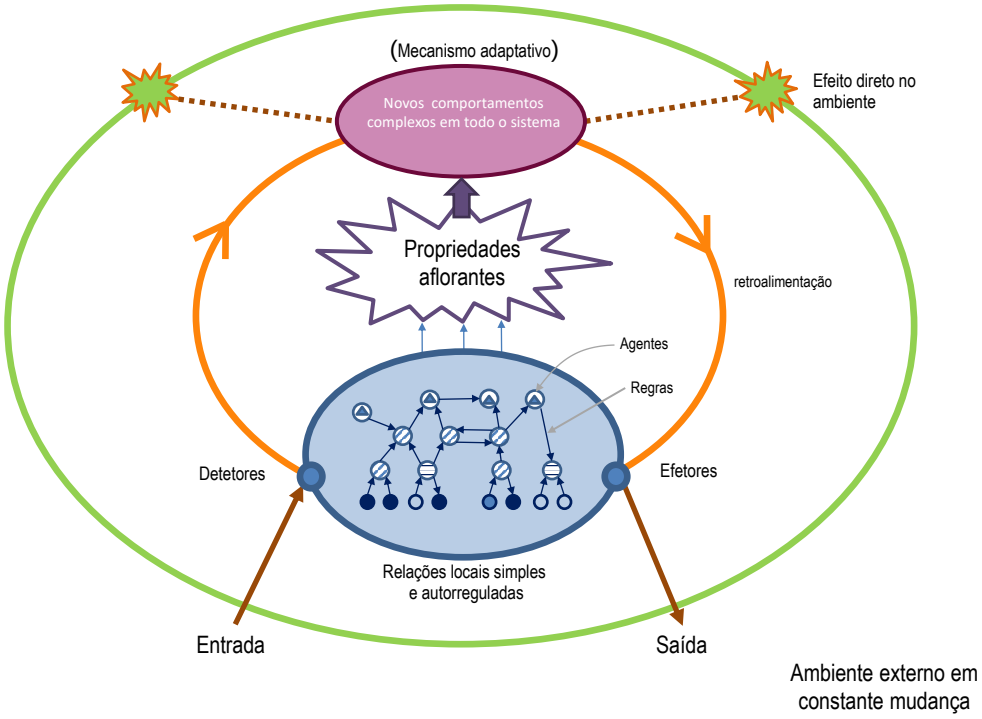


Figura 46 – Modelo baseado em agentes para o desenvolvimento de CAS.
O Autor (2018).

níveis que afetam os agentes em um determinado sistema. Cada nível se expande e orienta a camada além dela e adiciona um novo grau de liberdade à simulação. No nível 1, os agentes interagem e com base no conjunto de regas definido para cada tipo. No nível 2, normas são utilizadas para influenciar os agentes a escolher determinadas ações e orientar seu comportamento, especialmente em situações em que a escolha racional da ação vai contra o benefício coletivo, por exemplo, dilemas e negociações. Como as normas muitas vezes exigem ações adversas aos objetivos individuais, o agente pode enfrentar a questão de decidir seguir ou não uma regra. Se tal liberdade é permitida, então é necessário um terceiro nível, para monitorar se as normas estão sendo cumpridas e determinar sanções em caso de transgressões. No caso da Figura 47 existe ainda um quarto nível, por cima do monitoramento, que orienta como as sanções e o monitoramento devem ser conduzidos e lidando com as exceções. Esse nível permite recriar um certo senso de justiça onde os agentes têm liberdade para determinar quando e como as sanções devem ser emitidas e criar variações e flexibilizações intencionais nas regras de aplicação. Da mesma forma, diferentes tipos de mecanismos e técnicas podem ser incorporados para assistir às ações dos agentes no CAS de maneira que o sistema global exiba um comportamento mais ou menos especializado. Algumas dessas técnicas são: *fitness landscapes*, mecanismos de aprendizagem por reforço, funções de utilidade, entre outras.

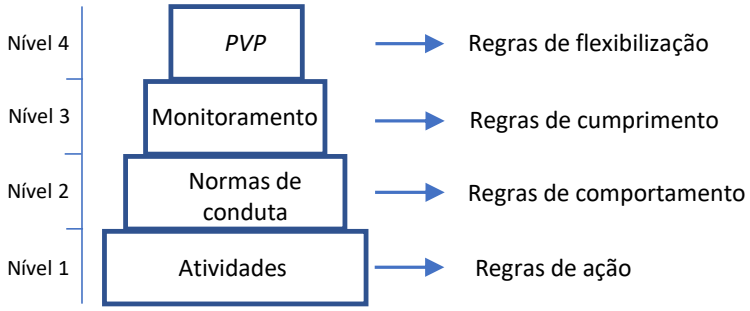


Figura 47 – Níveis de regras em CAS.
Adaptada de Kurka e Pitt (2017).

Outro fator interessante a observar ao trabalhar com os CAS são suas características essenciais. Não obstante, organizar tais características funcionais e estruturais dos CAS é um desafio devido ao seu estudo sob diferentes enfoques. Em Holland (1995) o autor propõe uma maneira geral de caracterizar os Sistemas Adaptativos Complexos por sete princípios básicos: quatro propriedades (Agregação, Não-linearidade, Fluxos e Diversidade) e três mecanismos (Mecanismo de Marcação, Modelos Internos e Padrão de Reorganização). Esses princípios não são os únicos recursos básicos que podem ser selecionados dos sistemas complexos, embora a maioria dos outros possa ser derivada de combinações apropriadas deles. Eles pretendem ser uma estrutura para organizar os diferentes recursos presentes nos CAS de maneira que propicie a liberdade de trabalhar com várias técnicas para implementar os mecanismos e propriedades necessários. De fato, a literatura apresenta diferentes análises de características de CAS de acordo com vários modelos ou teorias básicas.

A lista a seguir, sem pretender ser exaustiva ou completa, recolhe as principais características identificadas em sistemas complexos da literatura reconhecida no campo até o momento (HOLLAND, 2014; FORREST; MITCHELL, 2016; GELL-MANN, 1999; HOLLAND, 1995; MITCHELL; NEWMAN, 2002; CILLIERS, 2002):

- *Auto-Organização*: é a capacidade de estruturar e reestruturar-se, de aprender, se diversificar e aumentar sua complexidade. Os CAS não possuem hierarquia de comando ou controle centralizado, eles se reorganizam constantemente para encontrar o melhor ajuste com o meio ambiente.

- *Não-linearidade* nas relações ou interações entre componentes: uma pequena mudança no sistema CAS pode ativar uma série de reações e levar a efeitos globais desproporcionais (e.g. o efeito borboleta (HOLLAND, 2014)).

- *Comportamento adaptativo*: A capacidade de um agente de mudar o seu comportamento com base nas informações do seu ambiente, o conjunto de regras que possui

e mecanismos que permitam a avaliação, seleção, modificação e criação de regras. A adaptatividade se faz presente em dois níveis: no nível dos agentes através da mudança de regras no funcionamento individual e no nível do sistema como um todo (THE HEALTH FOUNDATION, 2010) para processar as propriedades emergentes.

- *Novidade Perpétua*: não é esperado que os CAS atinjam uma configuração ótima ou em equilíbrio, eles de fato operam em condições longe-do-equilíbrio, porém sempre na procura de atingir a melhor configuração para a situação corrente. Os CAS estão inseridos num ambiente e, por tanto, à medida que o ambiente muda, os sistemas mudam para garantir o melhor ajuste. Isso, influencia o próprio ambiente e cria um ciclo de mudança constante à medida que o sistema se desenvolve para se adaptar ao ambiente e o ambiente muda como resultado das alterações do sistema. Um sistema com esta característica também é conhecido como um sistema aberto.

- *Comportamentos "aflorantes"*: um evento de mudança repentino e imprevisível produzido pelas ações dos mecanismos, uma mudança natural e subsequente estabilização em uma nova ordem que não requer energia externa para acontecer.

- *Descentralização da tomada de decisões*: não existe um mecanismo único de controle que governe o comportamento do sistema. O controle tende a ser disperso e o comportamento geral do CAS é resultado de várias decisões individuais. Cada agente do sistema responde apenas às informações que estão disponíveis para ele localmente e desconhece o comportamento global.

- *Capacidade de aprendizado*: um mecanismo computacional pelo qual um sistema cognitivo poderia construir iterativamente um modelo detalhado e hierárquico de seu ambiente.

- *Mecanismo de retroalimentação*: um mecanismo que busca reforçar as ações de aprendizado, é autoaperfeiçoador e leva a melhores desempenhos ou evita problemas em comportamentos futuros.

- *Estrutura hierárquica padrão*: a maioria dos sistemas complexos possui uma organização hierárquica elaborada para organizar e gerenciar as leis comportamentais em diferentes níveis. Nela os níveis superiores restringem as ações dos níveis inferiores. Componentes de sistema de baixo nível fornecem todos os possíveis comportamentos que um sistema pode ter, e níveis mais altos de sistemas estabelecem os limites para o comportamento. Isso cria limitações de intertravamento entre os níveis que determinam o comportamento geral do sistema. Uma estrutura hierárquica é a consequência natural da propriedade de agregação mencionada nas sete abordagens básicas em Holland (1995).

- *Dependência de trajetória e informações históricas*: o estado de um sistema complexo em qualquer momento depende da sequência de eventos e decisões que precederam esse ponto, em outras palavras, sua história molda seu desenvolvimento futuro. Assim, os mecanismos de aprendizagem e ação permitem que o sistema se adapte sem perder o que já tinha aprendido no passado. Como consequência, não é possível reverter a história do sistema e o futuro é muitas vezes imprevisível (THE HEALTH FOUNDATION, 2010).

- *Dinâmica do estado do paradoxo*: equilíbrio de duas forças ou operações, uma que reforçando o crescimento, aproveitando e mantendo as configurações bem-sucedidas (exploração do já conhecido), e a outra visando descobrir novas combinações de características que podem obter melhores configurações ainda desconhecidas (descoberta).

- *Mindset e modelos*: normalmente, os CAS são regidos, em seu nível mais básico, por princípios simples (THE HEALTH FOUNDATION, 2010), modelos internos criados e usados para aproveitar as oportunidades e evitar ameaças e riscos. Esses modelos podem ser de natureza diferente, mas representam tanto os próprios agentes adaptativos quanto o ambiente em que eles atuam. À medida que os modelos são refinados pela experiência, eles formam homomorfismos (MITCHELL; NEWMAN, 2002) e permitem previsões mais precisas, diminuindo a incerteza sobre o comportamento do sistema em diferentes contextos e melhorando o seu desempenho.

O leitor pode se referir às seguintes fontes bibliográficas para informações mais detalhadas:

- Sobre Complexidade: os trabalhos de Mitchell (2009),Page (2010), Cilliers (2002).

- Sobre Sistemas Complexos em geral: em Newman (2011) o autor apresenta uma organização da literatura em Sistemas Complexos.

- Sobre CAS em geral: os livros de John Holland ((HOLLAND, 1992; HOLLAND, 1995; HOLLAND, 2014)).

- Sobre a aplicação de CAS em diferentes áreas: por exemplo, nas áreas de saúde como em Martínez-García e Hernández-Lemus (2013), e em estrategias de gerenciamento como em Ellis e Herbert (2011).

6.2.4 Tecnologias Adaptativas

A abordagem de Tecnologias Adaptativas (TA) teve sua origem no desenvolvimento dos Autômatos Adaptativos (AA) em Neto (1994), como uma alternativa para lidar com

os problemas de reconhecimento sintático e dependências de contexto nas linguagens de programação. Posteriormente a ideia constituiria a base para vários trabalhos que utilizaram esse formalismo para representar e manipular de forma prática, simples e confortável diferentes dispositivos automodificáveis (NETO, 2011). Em Neto (2007) e Neto (2011) o autor fornece uma revisão das diferentes pesquisas que tem desenvolvido dispositivos adaptativos sob as bases da TA, assim como tendências, áreas de aplicação e desafios futuros.

Para a TA, a adaptatividade é entendida como a capacidade de um sistema de, sem interferências de agentes, responder a estímulos externos, determinando e realizando autorreconfigurações dinâmicas de tipo estrutural e comportamental, conforme a necessidade, considerando para efeituar tais mudanças a informação historicamente acumulada (NETO, 2007). Assim, o termo Tecnologia Adaptativa refere-se à aplicação da adaptatividade para modelagem e criação de dispositivos adaptativos para fins práticos e concretos.

Um dispositivo adaptativo entende-se como sendo aquele cuja operação é guiada por um conjunto finito e explícito de regras que é capaz, por seus próprios meios, de realizar mudanças nesse conjunto regras para se ajustar a condições inesperadas no seu entorno. O dispositivo adaptativo é regido pelo seu funcionamento padrão (ou seja, pelo comportamento não adaptativo) até que a execução de alguma ação adaptativa provoque alterações em sua configuração (NETO, 2002).

A construção formal dos dispositivos adaptativos, descrita em Neto (2002), Neto e Bravo (2003), Neto (2007), representada na Figura 48, expõe, de maneira geral, a participação de duas componentes: um dispositivo subjacente (usualmente não adaptativo), e um mecanismo adaptativo, responsável pela incorporação da adaptatividade (usualmente também denominada camada adaptativa) composta de ações de várias naturezas. Esta arquitetura permite que a gestão da adaptatividade seja exclusivamente realizada pelo mecanismo adaptativo, colocando em evidência uma característica fundamental das Tecnologias Adaptativas: a possibilidade de reaproveitamento integral de formalismos consolidados, na forma de dispositivos subjacentes, oferecendo um aumento no seu poder de representação, ao custo de um pequeno incremento de complexidade na sua especificação formal. Pode-se destacar, entre os vários formalismos adaptativos relatados na literatura, as Gramáticas Adaptativas, Redes de Markov Adaptativas, Statecharts Adaptativos, Árvores de Decisão Adaptativas (PISTORI; NETO; PEREIRA, 2006), Algoritmos Genéticos, Tabelas de Decisão Adaptativas e os Dispositivos guiados por Regras Adaptativos, entre outros (NETO, 2007). Assim mesmo, o trabalho em Inojosa (2012) apresenta a proposta para um formalismo de adaptatividade de segunda ordem, melhor entendida como adaptatividade

propagada no nível hierárquico por meta-modelos.

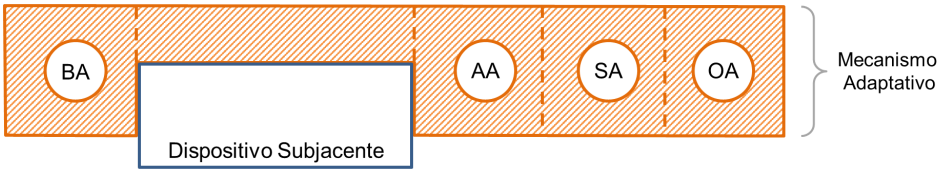


Figura 48 – Modelo geral de um Dispositivo Adaptativo.
O Autor (2019).

Segundo as Tecnologias Adaptativas, o dispositivo subjacente é qualquer forma de dispositivo dirigido por regras cujo comportamento dependa exclusivamente de um conjunto de regras que mapeiem cada possível configuração do dispositivo, começando por uma configuração inicial, para uma configuração seguinte correspondente. Formalmente ele pode ser definido como uma 6-tupla, ND=<C, NR, S, c_0, A, NA> onde:

- C: é o conjunto de todas as configurações possíveis e $c_0 \in$ C é a configuração inicial do dispositivo.
- S é o conjunto finito de todos os possíveis eventos que são considerados estímulos de entrada para o dispositivo. O elemento ϵ que representa o vazio também pertence a S.
- A $\subseteq$ C é o conjunto com todas as configurações de aceitação do dispositivo. Da mesma maneira, F=C-A é o conjunto de todas as configurações de rejeição do dispositivo.
- NA é o conjunto finito com $\epsilon \in$ NA de todos os símbolos de saída possíveis do dispositivo como efeitos colaterais da aplicação das regras de NR.
- NR é o conjunto de regras que definem o funcionamento do dispositivo através da relação NR $\subseteq C \times S \times C \times NA$. Cada regra r$\in$NR tem o padrão $r = (c_i, s, c_j, z)$, o que significa que em resposta de quaisquer estímulo de entrada s$\in$ S, r muda a configuração do dispositivo em questão de c_i para c_j consumindo s e gerando z$\in$NA como efeito colateral.

Entanto, o mecanismo adaptativo transforma um dispositivo tradicional em um dispositivo adaptativo ao permitir que as regras de transição que regem o seu comportamento mudem dinamicamente através da aderência de um conjunto opcional de até duas ações adaptativas correspondentes às regras de transição como representado na Figura 49.

Uma ação adaptativa descreve as modificações a serem aplicadas ao autômato adaptativo no momento em que for executada. Elas também podem ser entendidas como funções adaptativas que funcionam de maneira semelhante às chamadas de função em

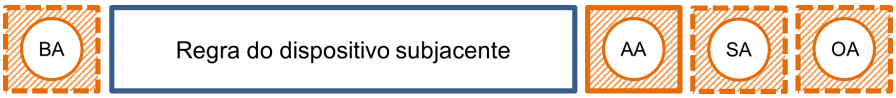


Figura 49 – Representação do Dispositivo Adaptativo.
O Autor (2019).

linguagens de programação. Assim, uma função adaptativa contém uma série de ações adaptativas elementais que dirigem as modificações a serem realizadas. As ações adaptativas elementais podem ser de três tipos, denotadas da seguinte maneira:

- ? : ação de tipo inspeção, a qual procura as regras de transição aderentes a um determinado padrão.
- $-$: ação de tipo remoção, a qual remove uma regra do conjunto de transições do dispositivo se ela adere a um determinado padrão.
- $+$: ação de tipo inserção, a qual incorpora uma nova regra no conjunto de transições, de acordo com determinado padrão.

De maneira geral, o mecanismo adaptativo é definido como AM $\subseteq BA \times NR \times AA$, onde:

- BA ou *Before Action* é o conjunto de ações adaptativas a serem executadas antes da aplicação das regras correspondentes à configuração corrente do dispositivo.
- AA ou *After Action* é o conjunto de ações adaptativas a serem executadas depois da aplicação das regras correspondentes à configuração do dispositivo.
- NR ou *Non-adaptive Rule* é o conjunto de regras particulares definidas para o funcionamento do dispositivo em questão.
- SA ou *Semantic Action* é o conjunto de ações semânticas que podem ser executadas como consequência das ações adaptativas (PISTORI, 2003).
- OA ou *Output Action* é o conjunto de qualquer tipo de ações possíveis a serem executadas como consequência das ações adaptativas.

Sendo que das ações adaptativas BA e AA, só AA é condição suficiente para a existência de comportamento adaptativo como detalhado em Neto (2002).

Esse mesmo trabalho, nos diz que é possível definir o dispositivo adaptativo, representado na Figura 48 como sendo AD=<ND_0, AM>, onde ND_0 designa o dispositivo subjacente inicial, e AM o mecanismo adaptativo correspondente. Para um passo de operação k no funcionamento do dispositivo observamos que:

- ND_k, o dispositivo subjacente no passo de operação k.
- Por definição, qualquer das regras não adaptativas de NR_k possui um equivalente em AR_k.
- C_k é o conjunto de todas as configurações possíveis para ND_k.
- S é o conjunto de estímulos de entrada válidos para AD.
- A natureza de A e F é a mesma que a especificada para ND.
- ϵ representa a cadeia vazia.
- NA é o conjunto finito (com $\epsilon \in$ NA) de todos os símbolos de saída possíveis do dispositivo como efeitos colaterais da aplicação das regras adaptativas em AR.

Assim, em cada estado da execução do autômato são executadas tanto as regras de transição convencionais como as ações adaptativas, até atingir um estado final ou uma condição de parada.

A Figura 50 representa o funcionamento de um autômato adaptativo em três instantes de tempo: em t=0 o autômato está na configuração inicial do dispositivo subjacente e se rege pelas regras do mesmo, em t=k autômato encontra uma ação adaptativa associada para a dupla <entrada, estado> atuais. Nesse instante o autômato executa a regra adaptativa, no instante t=k+1 o autômato possui uma configuração diferente de aquela fornecida inicialmente.

Informações detalhadas do formalismo para autômatos adaptativos pode ser consultada em Neto e Bravo (2003).

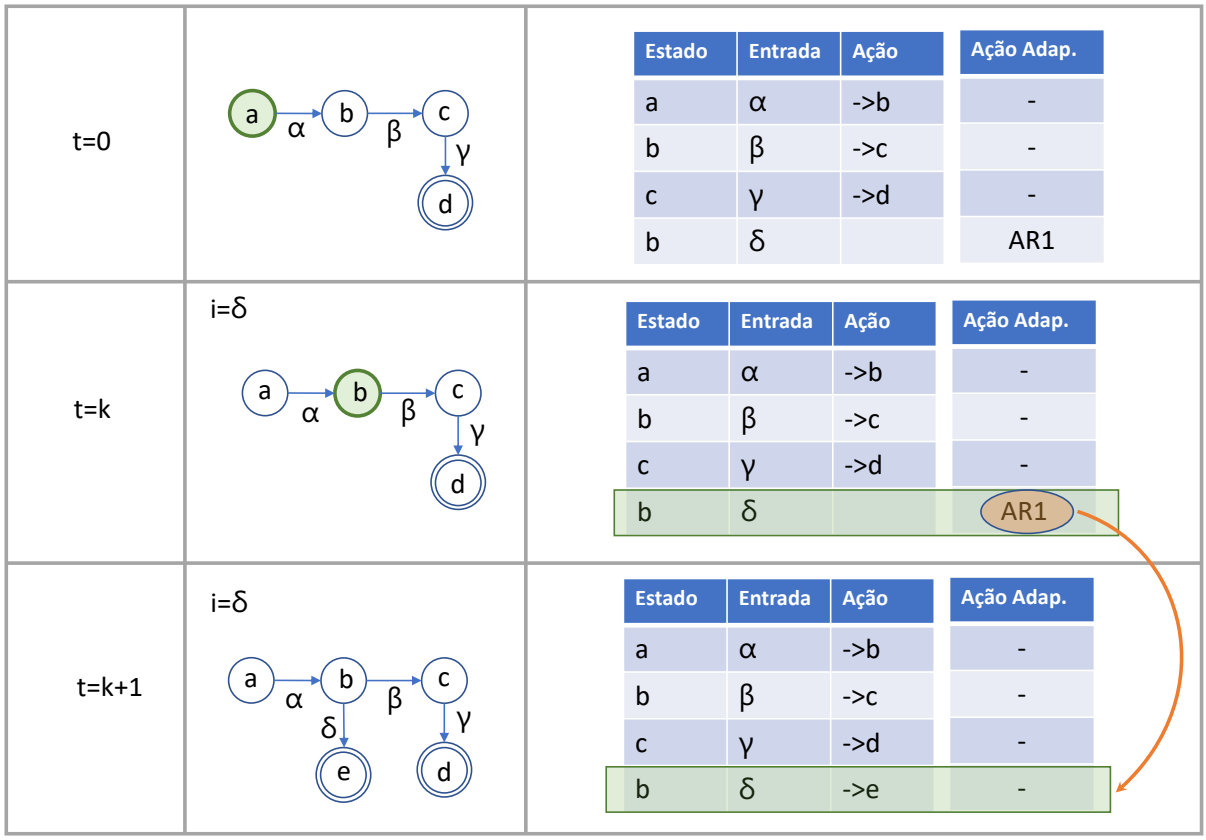


Figura 50 – Representação da função de um Autômato Adaptativo.
O Autor (2021).

6.3 Definição do Modelo Holístico

Embora as definições das diferentes abordagens apresentadas na seção anterior discordam em vários pontos ao descrever o comportamento adaptativo, todas elas compartilham a mesma base conceitual que pode ser abreviada como:

A adaptatividade é uma característica de um sistema que auxilia esse sistema a lidar com as mudanças encontradas no seu ambiente.

A Figura 51 mostra uma representação desta base conceitual que, embora simples, fornece uma base sólida para começar a modelar a adaptatividade. A partir dela inferimos que:

(a) a adaptatividade só pode existir num sistema, e

(b) ela existe com respeito à interação com um ambiente concreto.

Assim, sabemos que a adaptatividade é construída e se relaciona com várias propriedades de uma entidade, enriquecendo o comportamento de tal entidade, mas, por

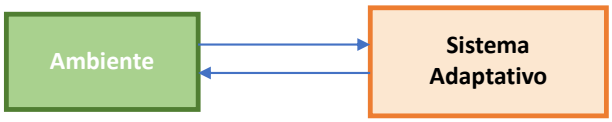


Figura 51 – Interação entre o sistema adaptativo e o ambiente.
O Autor (2021).

si só, não é capaz de se manifestar ou agir. Da mesma forma, o ambiente no qual ela se desenvolver precisa ser concreto e deve existir, pelo menos de alguma forma, antes do comportamento adaptativo (LEWONTIN, 1978). A razão para isso é que a adaptatividade representa a resposta do sistema ao ser exposto à influência de diferentes forças encontradas nesse ambiente específico.

Seguindo este argumento, começamos por definir o ambiente.

Definição 1 (Ambiente) *O ambiente ou contexto de qualquer entidade é qualquer informação que possa ser usada para descrever a situação na qual essa entidade está imersa. Assim, dizemos que o ambiente pode ser entendido como:*

$$\mathbf{Env}_{\text{SYS}} = \{\mathrm{s}_1, \mathrm{s}_2, ..., \mathrm{s}_\mathrm{n}\}$$

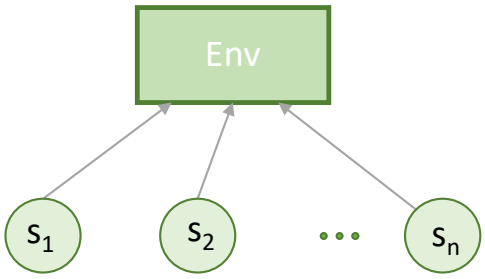


Figura 52 – Representação básica do ambiente Env.
O Autor (2021).

A Figura 52 nos permite visualizar esta definição, onde s_i representa uma variável ou estímulo do ambiente significativo para a interação com o sistema SYS. Isso significa que não é esperado que $\mathbf{Env}_{\text{SYS}}$ seja uma representação completa e exhaustiva do entorno do sistema. Tal representação não existe na prática, não apenas pelas dimensões da tarefa de representar todas as variáveis que existem no entorno, mas também pela natureza dinâmica das interações entre o ambiente e sistema que realiza mudanças constantes em tais variáveis. No entanto, $\mathbf{Env}_{\text{SYS}}$ deve incluir as informações relevantes para caracterizar o ambiente em que ocorrem as interações com o sistema. Isso significa que o $\mathbf{Env}_{\text{SYS}}$ deve listar, no mínimo, os canais de interação com SYS.

Cada estímulo s_i pode ser mapeado para diferentes tipos de fontes disponíveis em um determinado tempo t. Por exemplo, ao analisar o comportamento adaptativo em sistemas computacionais, os estímulos podem ser originados pelo usuário, pelo ambiente físico ou por outros subsistemas. No caso de sistemas biológicos, as fontes dos estímulos podem ser: a geografia, a disponibilidade de recursos, a população de espécies e assim por diante.

A simplicidade dessa definição nos dá espaço para explorar níveis mais profundos de detalhe em relação à representação de s_i. O conceito de modelagem de contexto ou ambientes de funcionamento tem sido um tema de pesquisa na computação desde os anos 90 devido, entre outras coisas, ao interesse em desenvolver sistemas computacionais sensíveis ao contexto. Os autores em Wan (2009) apresentam uma abordagem histórica sobre o uso do contexto na Ciência da Computação, e pesquisas sobre técnicas de modelagem de contexto podem ser encontradas em Bettini *et al.* (2010) e Strang e Popien (2004). Revisões sistemáticas da literatura sobre diferentes abordagens para modelar características ambientais em sistemas autoadaptativos foram realizadas em Shin, Bae e Bae (2021), Hong, Suh e Kim (2009).

Uma vez que estabelecemos o que entendemos como ambiente, podemos prosseguir para a definição do sistema que apresentará o comportamento adaptativo. Como apontado por Bruni *et al.* (2012) a definição geral que apresentamos previamente segue uma abordagem conhecida como "caixa preta" ou definição comportamental, e não é o substancial suficiente para caracterizar propriamente os sistemas adaptativos, permitindo distingui-los de sistemas padrão. Isso acontece porque quase qualquer sistema com tamanho razoável consegue modificar, em alguma medida, o seu comportamento como reação a uma mudança específica no seu contexto de execução. Portanto, é preciso modificar a estratégia de análise e adotar uma abordagem de "caixa branca" para inspecionar a caracterização de sistemas adaptativos com maior detalhe.

Examinando as definições de adaptação que apresentamos anteriormente é possível inferir a seguinte descrição intuitiva sobre o comportamento adaptativo:

O comportamento adaptativo é o acionamento de um mecanismo adaptativo agindo sobre um sistema para ajudá-lo a lidar com as mudanças ambientais enquanto este persegue um objetivo específico.

A partir dessa informação, identificamos que:

(a) Em "condições normais" o sistema de estudo opera sem necessidade de adaptação.

(b) O comportamento adaptativo é direcionado, ele age sobre um objetivo pré-estabelecido do sistema.

(c) A adaptatividade, por si mesmo, não é uma entidade, nem um conceito independente, ela depende tanto do sistema, quanto do contexto e a interação entre ambos.

(d) O mecanismo adaptativo que realiza a adaptação de fato segue um processo lógico particular para elaborar a resposta às mudanças (estímulos do ambiente) com os recursos disponíveis.

A afirmação em (a) indica que a adaptatividade é usada quando necessário. Isso significa que, idealmente, na ausência de necessidade, o sistema deve agir conforme as regras já codificadas em seu comportamento, pois elas encapsulam seu propósito ou objetivo. No contexto da adaptação biológica esta necessidade é expressa como a "luta pela existência" mencionada em Gaines (1972), Lewontin (1978). A afirmação em (b) diz que a adaptatividade ajuda o sistema a atingir um objetivo intrínseco, portanto, é faculdade de tal sistema gerenciar as possíveis variações desse objetivo. Não corresponde ao comportamento adaptativo definir nem alterar o objetivo do sistema, mas ajudar a caminhar em direção a ele. Novamente, na adaptação biológica, é possível identificar esse objetivo em cada indivíduo como a busca pela "sobrevivência e reprodução" da espécie referida por Reeve e Sherman (1993), Dobzhansky (1968), Brock (2000). Em (c) temos que a adaptatividade existe apenas na presença de um sistema interagindo dinamicamente com seu ambiente. Por fim, (d) afirma que a adaptação é o resultado de um habilidoso processo lógico para analisar, avaliar e elaborar as ações necessárias para que o sistema funcione adequadamente nas condições atuais.

A partir dessas quatro afirmações observamos que os sistemas adaptativos manifestam uma separação de interesses, entendido como o desacoplamento entre o mecanismo regulatório e a organização constitutiva regulada, proporcionando independência entre funcionalidade de baixo nível e variação de alto nível (BARANDIARAN; MORENO, 2008). Isso nos permite formular um sistema adaptativo como tendo um sistema subjacente ($\mathbf{Sys}_{\mathrm{U}}$), não adaptativo, trabalhando em coordenação com um mecanismo adaptativo ($\mathbf{AL}$) para perseguir um objetivo por meio da interação com seu ambiente ($\mathbf{Env}_{\mathrm{U}}$) como mostra a Figura 53.

Uma vez que definimos as entidades participantes de um sistema adaptativo, podemos nos concentrar no tipo de propriedades que essas entidades precisam manter para trabalhar em coordenação e conduzir o comportamento adaptativo.

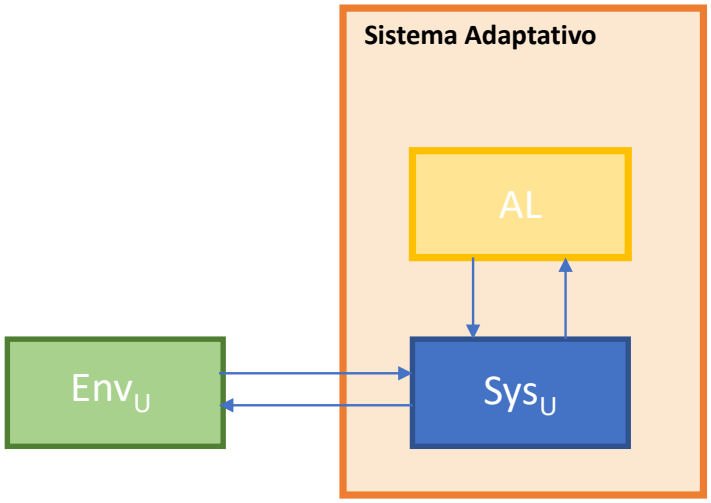


Figura 53 – Elementos básicos da adaptatividade.
O Autor (2021).

O sistema subjacente, denominado **Sys**$_U$, é a entidade cujo comportamento conecta e define totalmente o sistema em estudo. Para suportar um sistema adaptativo, o **Sys**$_U$ precisa satisfazer os requisitos mínimos para suportar a dinâmica do comportamento adaptativo. Em termos gerais, o **Sys**$_U$ deve cumprir o seguinte:

(i) **Ser um sistema**: em termos gerais, um sistema simples não adaptativo, pode ser entendido como tendo os seguintes elementos:

- **modelo interno**: uma representação da própria entidade. Este modelo ajuda a desenvolver a autoconsciência.
- **modelo externo**: uma representação, estrutura conceitual ou esquema, do ambiente onde a interação ocorre. Normalmente, este modelo não é um retrato completo do ambiente real, no entanto, ele precisa retratar todas as informações mínimas relacionadas à natureza da interação no local e tempo específicos. Este modelo é usado pelo sistema para organizar e estruturar informações sobre seu ambiente e ajuda a desenvolver a ciência do entorno ou consciência situacional.
- **resposta característica**: contém o funcionamento da entidade e sua finalidade. Também conhecidos como processos construtivos (BARANDIARAN; MORENO, 2008), metabolismo ou ações funcionais (RUIZ-MIRAZO; MORENO, 2004), eles participam da produção contínua do próprio sistema.
- **comportamento**: descreve o mecanismo/lógica que orienta a entidade a realizar seus processos construtivos e interativos (BARANDIARAN; MORENO, 2008). Descrito por Ziemke (1998) como uma regularidade na dinâmica de interação entre um agente e seu ambiente. Uma das técnicas mais utilizadas para representar o comportamento em sua forma mais básica é por meio de

regras da forma evento-reação. A maioria das técnicas modernas e sofisticadas em estudos comportamentais permitem ser decompostas em regras simples.

Os dois primeiros elementos estabelecem a identidade do sistema determinando os seus limites e os pontos de interação com seu ambiente, também conhecidos como pontos de fluxo de energia, matéria e/ou informação segundo (BARANDIARAN; MORENO, 2008).

Para simplificar, neste trabalho vamos analisar o caso em que um sistema geral, não adaptativo, denominado $\mathbf{Sys}_{\text{NA}}$ tem apenas um elemento, $\mathbf{Sys}_{\text{NA}} = \{na_1\}$. No entanto, $\mathbf{Sys}_{\text{NA}}$ poderia estar composto por um conjunto de elementos, como acontece, por exemplo, no caso de Sistemas Complexos ou Sistemas Multiagentes, e poderia ser descrito como:

$$\mathbf{Sys}_{\text{NA}} = \{na_1, na_2, ..., na_m\}$$

Onde cada elemento na_i está em conformidade com a formulação de $\mathbf{Sys}_{\text{NA}}$ apresentada a seguir e detalhada em Neto (2002).

Definição 2 (Sistema) *Um sistema geral,* $\mathbf{Sys}_{\text{NA}}$ *pode ser definido pela 6-tupla*

$$\mathbf{Sys}_{\text{NA}} = (\text{C}_{\text{NA}}, \text{R}_{\text{NA}}, \mathcal{S}_{\text{NA}}, \text{c}_0^{\text{NA}}, \text{A}_{\text{NA}}, \mathcal{O}_{\text{NA}})$$

Onde:

- C_{NA}*: é o conjunto de todas as configurações possíveis para* $\mathbf{Sys}_{\text{NA}}$ *e* $\text{c}_0^{\text{NA}} \in \text{C}_{\text{NA}}$ *é o seu estado inicial;*
- $\mathcal{S}_{\text{NA}}$ *é o conjunto de todos os estímulos de entrada possíveis para* $\mathbf{Sys}_{\text{NA}}$*,* $\varepsilon \in \mathcal{S}_{\text{NA}}$*;*
- $\mathcal{O}_{\text{NA}}$ *é o conjunto de todos os estímulos de saída possíveis como resultado da aplicação das regras em* $\mathbf{Sys}_{\text{NA}}$*;*
- $\text{A}_{\text{NA}} \subseteq \text{C}_{\text{NA}}$ *é o subconjunto de todas as configurações aceitas para* $\mathbf{Sys}_{\text{NA}}$*. Particularmente,* $\text{F}_{\text{NA}} \subseteq \text{C}_{\text{NA}}$ *é o subconjunto de todas as configurações de rejeição para* $\mathbf{Sys}_{\text{NA}}$ *(*$\text{F}_{\text{NA}} = \text{C}_{\text{NA}} - \text{A}_{\text{NA}}$*);*
- R_{NA} *é o conjunto de regras que descreve o comportamento de* $\mathbf{Sys}_{\text{NA}}$ *como uma relação* $\text{R}_{\text{NA}} \subseteq \mathbf{C}_{\text{NA}} \times \mathcal{S}_{\text{NA}} \times \mathbf{C}_{\text{NA}} \times \text{R}_{\text{NA}}$*.*

Tomando esta descrição como base podemos identificar as condições que apresentamos anteriormente para $\mathbf{Sys}_{\text{NA}}$. Assim, uma configuração $\text{c}_\text{i}^{\text{NA}} \in \text{C}_{\text{NA}}$ neste modelo indica as condições, estado, disponibilidade de recursos e outras informações que

descrevem $\mathbf{Sys}_{\text{NA}}$ no momento i. Isso inclui informações sobre suas características internas, bem como condições situacionais relevantes para seu funcionamento. Então, C_{NA} expressa a consciência do sistema sobre si mesmo e o entorno.

$\mathcal{S}_{\text{NA}}$, pela sua vez, estabelece uma das direções da interação entre o sistema e o ambiente, especificamente do ambiente para o sistema, ao fornecer informações sobre as condições do ambiente de $\mathbf{Sys}_{\text{NA}}$, representando os dados de entrada para o funcionamento do $\mathbf{Sys}_{\text{NA}}$. Geralmente, espera-se que um sistema $\mathbf{Sys}_{\text{NA}}$ eficiente processe todos os estímulos do ambiente no qual está inserido, $\mathbf{Env}_{\text{NA}}$, fazendo $\mathcal{S}_{\text{NA}} = \mathbf{Env}_{\text{NA}}$, pois $\mathbf{Env}_{\text{NA}}$ engloba todas as informações relevantes para a interação entre eles. No entanto, em sistemas de sistemas não é incomum que os sistemas, individualmente, apenas transmitam alguns dos estímulos disponíveis no ambiente, tornando $\mathcal{S}_{\text{NA}} \subseteq \mathbf{Env}_{\text{NA}}$.

A_{NA} expressa os objetivos do $\mathbf{Sys}_{\text{NA}}$ direcionando o comportamento do sistema para o cumprimento de seu propósito (ZIEMKE, 1998). Quando $\mathbf{Sys}_{\text{NA}}$ atinge qualquer $\text{c}_{\text{i}}^{\text{NA}} \in \text{A}_{\text{NA}}$ após processar o estímulo $\text{s}_{\text{k}} \in \mathcal{S}_{\text{NA}}$ dizemos que $\mathbf{Sys}_{\text{NA}}$ manteve uma interação satisfatória com o ambiente (teve um desempenho satisfatório). F_{NA}, por outro lado, é um indicador de distanciamento das metas do $\mathbf{Sys}_{\text{NA}}$ e desempenho insatisfatório.

$\mathcal{O}_{\text{NA}}$ estabelece a outra direção da interação com o ambiente, implementando a influência de $\mathbf{Sys}_{\text{NA}}$, ao causar perturbações dentro do ambiente.

Finalmente, R_{NA} descreve, via um conjunto de regras, a resposta característica do $\mathbf{Sys}_{\text{NA}}$. Uma regra $r \in \text{R}_{\text{NA}}$ é definida como $r = (\text{c}_{\text{i}}^{\text{NA}}, s, \text{c}_{\text{j}}^{\text{NA}}, z)$, onde $\text{c}_{\text{i}}^{\text{NA}}, \text{c}_{\text{j}}^{\text{NA}} \in \text{C}_{\text{NA}}, z \in \mathcal{O}_{\text{NA}}$ e $s \in \mathcal{S}_{\text{NA}}$, indicando que como resposta aos estímulos s o sistema $\mathbf{Sys}_{\text{NA}}$ altera sua configuração atual de $\text{c}_{\text{i}}^{\text{NA}}$ para $\text{c}_{\text{j}}^{\text{NA}}$, processa s e gera z como saída. Diz-se que uma regra r é compatível com a configuração atual c se e somente se $\text{c}_{\text{i}}^{\text{NA}} = c$ e s é ε ou igual ao símbolo de entrada atual.

(ii) **Ser adaptável**: um sistema adaptável é um potencial sujeito de reconfiguração, o que significa que ele pode alterar sua configuração se solicitado. O sistema possui recursos para realizar algumas mudanças, porém, por si só, carece do mecanismo de tomada de decisão para exercer essa capacidade (CAYA; NETO, 2018). Os elementos necessários de um sistema adaptável são:

- **Sensores**: denominados Sen_{ABL}, são um conjunto de elementos que detectam e, possivelmente, codificam alguns estímulos imprevistos, $\mathcal{S}_{\text{H}}$, e os encaminha para processamento posterior. Tais estimulos podem proceder do ambiente do sistema , ou do próprio sistema.

- **Atuadores**: denominados Act_{ABL}, são um conjunto de elementos considerados "blocos de construção" que atuam sobre o próprio sistema para alterar sua configuração. Eles também são conhecidos como efetores.

  Como um sistema adaptável não possui mecanismo de tomada de decisão para esses estímulos inesperados, os sensores e os atuadores precisam estar acessíveis para uma terceira entidade responsável pela tomada de decisão, portanto, eles precisam ser expostos. No entanto, é importante notar que Sen_{ABL} e Act_{ABL} estão disponíveis apenas num escopo a ser especificado pelo domínio do problema e/ou observador, o que significa que não precisam ser de conhecimento público.

  Ambos tipos de elementos estão integrados no que definimos como "**ganchos de adaptatividade**", através dos quais o mecanismo de adaptativo vai estar ligado ao ambiente e ao sistema subjacente. Esses ganchos adaptativos respondem a quais eventos o sistema é adaptável (ou potencialmente adaptativo) e especificam quais recursos o sistema expõe (ou disponibiliza) para construir uma resposta para esses eventos específicos.
  A Figura 54 mostra a representação, inspirada no simbolismo usado em Chen e Crilly (2016), para Sys_{NA} em relação a outras entidades.

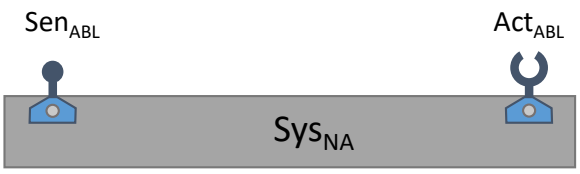


Figura 54 – Sensores e Atuadores em Sys_{NA}.
O Autor (2021).

- **Mecanismo de comunicação**: denominado *Comm*, é o mecanismo pelo qual o sistema gerencia o fluxo de controle e, por meio disso, permite que ele transfira o poder de decisão para outra entidade previamente apontada como responsável. Para a correta transferência de controle é necessário que o sistema pare/pause o seu comportamento atual e transfira o controle de suas

ações para um mecanismo de decisão. Depois que uma decisão foi tomada, o sistema é instruído sobre as ações a serem executadas, ele recupera o controle de suas ações e passa a se comportar conforme solicitado. Este mecanismo é quem vincula a entidade decisória e o sistema sujeito a reconfiguração através dos ganchos de adaptação, ou seja, o mecanismo estabelece uma comunicação bidirecional conforme representado na Figura 55.

Figura 55 – Mecanismo de comunicação entre o sistema e a tomada de decisão.
O Autor (2021).

Em geral, os ganchos de adaptatividade e o mecanismo de comunicação são pré-definidos e não mudam ao longo do tempo de vida do sistema. No entanto, em determinadas circunstâncias, podem ocorrer algumas alterações. Um exemplo destas circunstâncias particulares são a difusão de novos recursos em um Ambiente Inteligente, ou no contexto de Sistema de Sistemas, ou Arquiteturas Baseadas em Serviços (SOA). A continuação propomos uma definição geral para um sistema adaptável.

Definição 3 (Sistema Adaptável) $\mathbf{Sys}_{\text{ABL}}$ *é uma entidade adaptável ou potencialmente adaptativa em relação a algum conjunto de estímulos* $\mathcal{S}_{\text{H}}$*, se:*

(i) inclui um sistema ($\mathbf{Sys}_{\text{NA}}$)*;*

(ii) possui um conjunto de sensores (Sen_{ABL})*, identificando as pertubações que geram possíveis respostas na forma* $\text{Sen}_{\text{ABL}} = \{sen_1, sen_2, sen_3..., sen_k\}$*, com* $k > 0$*, onde cada* sen_k *consegue receber um estímulo de* $\mathcal{S}_{\text{H}}$ *e encaminhar suas informações se solicitado.* Sen_{ABL} *precisa estar disponível para ser usado por outra entidade dentro do mesmo contexto. Em relação com* $\mathbf{Sys}_{\text{NA}}$ *isso significa* $\mathcal{S}_{\text{H}} \subseteq \mathcal{S}_{\text{NA}}$*, pois pode existir uma série de estímulos que não estão sujeitos a reconfigurações;*

(iii) possui um conjunto de atuadores (Act_{ABL})*, contendo tanto as primitivas operacionais que afetam a configuração do sistema quanto seus relacionamentos, os quais constituem o domínio dinâmico capaz de modelar o comportamento adaptativo. Cada elemento no* Act_{ABL} *pode ser representado pela 2-tupla: <estado, transições>. Assim, como com* Sen_{ABL}*,* Act_{ABL} *precisa estar disponível para ser usado por outra entidade dentro do mesmo contexto. É importante notar que as*

respostas potenciais do sistema ao estímulo $\mathcal{S}_{\text{H}}$ *são limitadas pelo conjunto de ações disponíveis através dos atuadores;*

(iv) possui um mecanismo de comunicação (Comm) *que vincula o sistema subjacente,* $\mathbf{Sys}_{\text{NA}}$*, com alguma entidade encarregada da tomada de decisão, estabelecendo uma conexão bidirecional entre eles.*

Assim, podemos formular $\mathbf{Sys}_{\text{ABL}}$ como:

$$\mathbf{Sys}_{\text{ABL}} = (\mathbf{Sys}_{\text{NA}}, \text{Sen}_{\text{ABL}}, \text{Act}_{\text{ABL}}, \text{Comm})$$

Voltando para a representação do sistema adaptativo apresentada na Figura 53, temos que:

$$\mathbf{Sys}_{\text{U}} = \mathbf{Sys}_{\text{ABL}}$$

E, consequentemente, o sistema adaptativo $\mathbf{Sys}_{\text{ADA}}$ é definido como:

$$\mathbf{Sys}_{\text{ADA}} = (\mathbf{Sys}_{\text{ABL}}, \mathbf{AL}, \mathbf{Env}_{\text{NA}})$$

Uma vez que definimos o ambiente de execução, $\mathbf{Env}_{\text{NA}}$, e verificamos que nosso sistema subjacente é um sistema adaptável, $\mathbf{Sys}_{\text{ABL}}$, precisamos definir a lógica adaptativa[2]. Para isso, começamos definindo o conceito de adaptatividade, denominado também comportamento adaptativo na literatura. A definição a seguir é uma proposta própria formulada após leitura dos trabalhos Barandiaran e Moreno (2008), Ashby (1960), Reeve e Sherman (1993), Holland (1995), Caya e Neto (2018), Neto (2002), Gaines (1972), Ruiz-Mirazo e Moreno (2004).

Definição 4 (Adaptatividade/Comportamento Adaptativo) *A adaptatividade é a capacidade de uma entidade, de decidir, a qualquer momento, a modificação de suas próprias características, estrutura e/ou comportamento, através da execução de um conjunto de ações próprias, como resposta a novos eventos percebidos em seu entorno enquanto persegue determinado objetivo. As mudanças efetuadas pela entidade têm como finalidade uma adequação mais eficiente ao novo contexto de funcionamento. Um sistema que mostra esse recurso é chamado de sistema adaptativo.*

Seguindo esta definição identificamos que a lógica adaptativa é a responsável pelo raciocínio, elaboração e comando de uma resposta adequada às mudanças que o sistema percebe. De maneira geral ela pode ser definida como:

[2] Existem vários termos utilizados na literatura para denominar essa entidade, por exemplo, mecanismo adaptativo, camada adaptativa e lógica adaptativa, neste trabalho usamos esse último termo.

Definição 5 (Lógica Adaptativa) *Dizemos que a* **AL** *é uma lógica adaptativa se for um mecanismo regulador que exerce controle ativo sobre alguns dos processos de um sistema adaptável* $\mathbf{Sys}_{\text{ABL}}$ *para responder adequadamente, e de forma autônoma, a algumas mudanças no ambiente com a finalidade de prevenir/evitar/sair de situações que impedem o* $\mathbf{Sys}_{\text{NA}}$ *de cumprir seu objetivo. As respostas elaboradas e ativadas pela* **AL** *consideram experiências anteriores e, às vezes, podem até considerar estratégias de previsão.*

Ao olhar os elementos do AL com mais detalhe, notamos que um elemento chave para a manifestação da adaptatividade é a existência de autonomia, pois ela garante a independência do processo de tomada de decisão e a característica de autogestão necessária para que a adaptatividade exista (BARANDIARAN; MORENO, 2008). Ao se falar em autonomia é importante considerar que, assim como com a adaptabilidade, esse é um termo emprestado das Ciências Naturais e aplicado a diferentes áreas do conhecimento, e, em consequência, existem várias definições e noções para ele. No entanto, no contexto das engenharias, Ziemke (1998) resumiu as principais noções de autonomia em:

- Autonomia Operacional: como a capacidade de operar sem intervenção externa, também conhecida como automaticidade [3].
- Autonomia Comportamental: como a capacidade de formar e adaptar seus princípios de comportamento.

Devemos também observar que a autonomia deve ser percebida como um contínuo e não uma propriedade binária (ou não existe, ou é total) o que significa que existem diferentes graus em que uma entidade está no controle de seu comportamento (ZIEMKE, 1998). O grau mínimo no qual uma entidade consegue suportar interações autônomas (naturais ou plásticas) com seu ambiente é chamado de Autonomia Básica. Baseados nas informações de trabalhos como Barandiaran e Moreno (2008), Ruiz-Mirazo, Peretó e Moreno (2004) e Ruiz-Mirazo e Moreno (2004) no contexto da aplicação da Teoria de Autopoiese nos sistemas computacionais podemos formular a seguinte definição:

Definição 6 (Autonomia Básica) *A autonomia básica é a capacidade de um sistema de gerenciar o fluxo de recursos (matéria, energia ou informação) por si mesmo para poder, simultaneamente, regular, modificar e controlar: (i) processos internos autoconstrutivos e (ii) processos de troca com o meio ambiente.*

Para alegar autonomia básica, um sistema deve apresentar os seguintes elementos:

[3] Em (STEELS, 1995) o autor descreve a diferença entre automaticidade e autonomia, assim como também entre conhecimento e inteligência, representações e emergência.

- **Mecanismos de interação**: são os mecanismos que permitem estabelecer a interação dinâmica com o ambiente para troca de informações e recursos. Esses mecanismos de interação permitem que o sistema perceba seu ambiente de funcionamento, ficando corporizado (ou *embodied*[4]) e situado (FLORIAN, 2003).

- **Medida de desempenho e critério de aceitação**: esses elementos representam o núcleo da autonomia ao implementar a capacidade de decidir se o sistema está ou não interagindo de maneira aceitável com seu ambiente enquanto busca realizar os seus objetivos internos. A medida de desempenho é alguma formulação capaz de decidir quando o objetivo do sistema foi atingido. A medida de desempenho também é referida na literatura como medida de aptidão (*fitness measure*) (REEVE; SHERMAN, 1993), função de aptidão (*fitness function*) (GAINES, 1972) ou função de desempenho (*performance function*)(ZADEH, 1963). O critério de aceitação constitui a norma para determinar se uma tarefa[5] foi satisfatória ou não, e também é conhecida na literatura como procedimento de decisão (ZADEH, 1963; HEYLIGHEN; JOSLYN, 2001).

- **Mecanismos homeostáticos**: são os processos que regulam/mantêm a estabilidade das configurações internas do sistema em resposta a mudanças nas condições ambientais externas. Ele usa o mecanismo de feedback negativo e positivo para responder às mudanças reforçando a tendência ou suprimindo-a.

- **Modificadores ou operadores estruturais**: são os elementos responsáveis por implementar as modificações necessárias para contrabalançar as mudanças no ambiente e sua propagação. São um conjunto de primitivas (variáveis e relacionamentos) que atuam sobre o sistema para modificar as suas configurações.

Uma possível formulação para a autonomia é dada por:

$$Aut = (Inter, \mathcal{p}, \mathcal{W}, Hom, Mod)$$

Onde:

- $Inter$ são os mecanismos de interação do sistema. Em nosso modelo esses mecanismos de interação são fornecidos pelo $\mathbf{Sys}_{\text{ABL}}$ através do $\mathbf{Sen}_{\text{ABL}}$;

[4] Em (FLORIAN, 2003) *Embodiment* é definido como: um sistema X está corporizado em um ambiente E se existirem canais entre os dois através dos quais um provoca perturbações no outro e vice-versa. Isto é, X está corporizado em E se para cada tempo t em que X e existem, algum subconjunto de estados possíveis de E tem a capacidade de perturbar o estado de X, e algum subconjunto de estados possíveis de X consegue perturbar o estado de E.

[5] Segundo Gaines (1972) uma tarefa é um segmento da interação entre o sistema e o ambiente para o qual é possível saber se foi ou não realizado de forma satisfatória.

- $\mathcal{p}$ é a medida de desempenho na forma $\mathcal{p} : \mathcal{S}_{\text{NA}} \times \mathcal{O}_{\text{NA}} \longrightarrow \mathbb{R}$ que dado um estímulo de $\mathcal{S}_{\text{NA}}$ e uma saída de $\mathcal{O}_{\text{NA}}$ retorna um valor real que avalia quão próxima está a saída $\mathcal{O}_{\text{NA}}$ do objetivo (propósito) do sistema;

- $\mathcal{W}$ é o critério de aceitação que verifica se a interação é aceitável ou não. Pode ser definido como um conjunto de preferências e restrições $\mathcal{W} = w_1, w_2, w_3, \ldots w_z$, com $z > 0$, que caracterizam um comportamento aceitável para o sistema;

- Hom é o conjunto de mecanismos que fornece a capacidade de manter uma variável específica ou conjunto de variáveis dentro de certos limites em torno de um ponto de referência, permitindo que o organismo mantenha um ambiente de trabalho ideal. Algumas informações interessantes sobre várias maneiras de alcançar a homeostase artificial são detalhadas em Goldstein e Kopin (2017). Este conjunto de mecanismos também constituem o grupo de operações de controle disponíveis para o sistema, assim, quando alguma ação de afasta do objetivo desejado (determinando uma possível perda de controle) os mecanismos redirecionam o funcionamento do sistema para encaminhá-lo para o objetivo;

- Mod é o conjunto de primitivas operacionais que permite exercer modificações no sistema. Em nosso modelo essas primitivas operacionais são fornecidas pelo $\mathbf{Sys}_{\text{ABL}}$ através do $\mathbf{Act}_{\text{ABL}}$.

Definir autonomia é um passo importante para a definição do comportamento adaptativo. Como mencionado, a autonomia é uma condição necessária para a adaptatividade, porém, ela não é condição suficiente (BARANDIARAN; MORENO, 2008). Um segundo elemento é necessário para complementar a autonomia: a implementação de um mecanismo ou estratégia de aprendizagem. Em seguida definimos o mecanismo de aprendizagem e suas componentes.

Definição 7 (Mecanismo de aprendizagem) *Um mecanismo de aprendizagem (RM) pode ser descrito, de maneira geral, como um mecanismo regulador que trabalha em coordenação com a memória.*

Neste ponto é importante esclarecer que quando dizemos aprendizagem nos estamos referindo ao processo de aprendizagem como o comportamento de adquisição de conhecimentos característico dos seres humanos e alguns animais e plantas. Em particular, não devemos confundir o conceito de aprendizagem mencionado nesta definição com, por exemplo, técnicas de aprendizagem de máquina que já apresentam uma abordagem concreta, sólida e reconhecida sobre o processamento e incorporação de informações para a criação de uma base de conhecimentos.

Definição 8 (Mecanismo regulador) *Um mecanismo regulador realiza um controle ativo de processos interativos e construtivos de um sistema, medindo diferentes condições e monitorando seus próprios processos para evitar ou prevenir situações disfuncionais. O mecanismo regulador é o encarregado de determinar as respostas necessárias ao conjunto específico de condições presentes e colocá-las em prática (BRUNI et al., 2012). A regulação é feita por uma sequência de diferentes processos:*

- ***Monitoramento**: este processo revisa a condição de algumas variáveis no ambiente ou no próprio sistema por um período e relata quaisquer possíveis alterações nelas. Assim, ele é responsável pela interação com o ambiente e a aquisição de dados.*
- ***Raciocínio**: interpreta as condições atuais e descobre quais são as respostas adequadas a elas. Esse raciocínio é realizado com auxílio da memória ou base de conhecimento para identificar semelhanças, diferenças e criar classificações e relações com as experiências anteriores. Este processo é responsável por formular o conjunto de possíveis respostas às informações passadas pelo monitoramento.*
- ***Seleção**: este processo é responsável por escolher, dentre um conjunto de opções, o comportamento específico a ser realizado para lidar com as mudanças percebidas. Ele faz isso avaliando e comparando as respostas potenciais identificadas pelo processo de raciocínio. Primeiro, aplica a medida de desempenho para quantificar a adequação das soluções e, posteriormente, refere-se ao procedimento de decisão e mecanismos homeostáticos para aplicar uma abordagem de diversificação ou especialização no comportamento global do sistema (HEYLIGHEN; JOSLYN, 2001). É equivalente do que Paolo (2005) descreve como mecanismo de criação de sentido.*
- ***Execução**: este processo põe em ação a resposta escolhida para as mudanças percebidas com base nos recursos disponíveis do sistema.*
- ***Realimentação ou Feedback**: em seu processo final, o mecanismo regulatório reúne as informações sobre as consequências, positivas e negativas, das decisões que tomou, tanto sobre sua própria configuração quanto no meio ambiente, e as incorpora ao conhecimento que mantém. Este processo é essencial para aprimorar o raciocínio em novas ocasiões. O processo caracteriza um feedback porque a própria resposta do sistema às mudanças causa, por si só, novas mudanças que podem, por sua vez, iniciar um novo ciclo. Esse ciclo de retroalimentação é uma característica do mecanismo regulatório e não é inerente à autonomia básica (BARANDIARAN; MORENO, 2008).*

Definição 9 (Memória) *: também conhecida como base de conhecimento, é a capacidade cognitiva que contém a lógica que rege todo o sistema e está codificada em algum conjunto*

de instruções. É a capacidade do sistema de codificar, armazenar, reter e depois recuperar informações, como: políticas, regulamentos, e estratégias, assim como experiências passadas. Sua função também compreende a transmissão de informações relevantes para que o sistema possa realizar eficientemente seus processos construtivos no decorrer das gerações subsequentes (RUIZ-MIRAZO; PERETó; MORENO, 2004). A base de conhecimento se desenvolve a cada nova interação, pois recebe novas informações e (possivelmente) as incorpora a sua estrutura.

Neste ponto é interessante notar que o feedback loop e a memória possibilitam a presença de diversas características dos sistemas adaptativos. Uma dessas características é a agência, definida por Barandiaran e Moreno (2008), Millican e Wooldridge (2014) como a capacidade de uma entidade de agir em qualquer ambiente de forma independente, deliberada, proposital e autônoma, fazendo suas próprias escolhas com base em sua vontade. Todos os sistemas adaptativos são agentes (STEELS, 1995) porque todos agem de acordo com um conjunto de regras especificamente formuladas para alcançar um resultado desejado. O contrário, no entanto, não é verdade, porque os agentes podem não ter perspectiva global, apenas interceptando, processando e respondendo a uma quantidade limitada de informações locais, sem estar capacitados para estabelecer seus próprios objetivos. Mais detalhes sobre agência no contexto de inteligência artificial são oferecidos em Millican e Wooldridge (2014). Outro recurso interessante que surge nos sistemas adaptativos é a causalidade circular. O objetivo ou meta de um sistema adaptativo com agência é manter e desenvolver sua ordem interna e capacidade de processar recursos. Eles só podem fazer isso importando recursos do ambiente e exportando o produto da sua função para esse mesmo ambiente, formando um ciclo. Esses sistemas dependem de seu ambiente para garantir seu funcionamento contínuo e se adaptam para manter e maximizar sua situação nesse ambiente. Como observado em Ruiz-Mirazo, Peretó e Moreno (2004), Ruiz-Mirazo e Moreno (2004) observa, o ciclo é aberto, estabelecendo um equilíbrio dinâmico no sistema, no qual a maximização exerce uma força que impulsiona o comportamento em uma direção, e um conjunto de mecanismos(homeostase) ocorre para contrabalançar essa tendência e manter o sistema sob controle em ambientes longe do de equilíbrio. Por outro lado, a representação da memória como participante explícito em nosso modelo de adaptatividade dá lugar para expressar outras características da memória estudadas nas ciências naturais, tais como: diferentes tipos de memórias, o papel do esquecimento no aprendizado (OKANO; HIRANO; BALABAN, 2000; BREM; RAN; PASCUAL-LEONE, 2013; CANTOR, 2021) e alguns problemas relacionados à memória que afetam a sua função, como observado em Forgetfulness... (2021), e assim por diante.

Continuando com a nossa definição da lógica adaptativa, a Figura 56 mostra

explicitamente que os elementos da lógica adaptativa (AL) se baseiam em propriedades da autonomia para exercer a tomada de decisão.

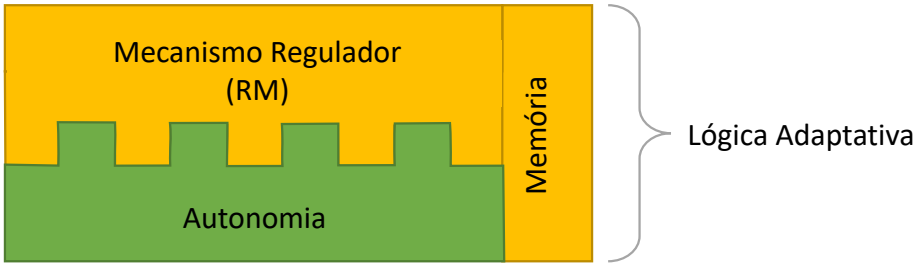


Figura 56 – Relação entre Autonomia e Lógica Adaptativa.
O Autor (2021).

Em nosso modelo, AL pode ser formulada como:

$$\mathbf{AL} = (\mathbf{RM}, \mathbf{Mem})$$
$$\mathbf{AL} = (\mathbf{Mon}, \mathbf{Rea}, \mathbf{Sel}, \mathbf{Ex}, \mathbf{Mem})$$

Onde:

- **Mon** é o processo que monitora um conjunto de variáveis de interesse para a função do $\mathbf{Sys}_{\text{ADA}}$ através dos recursos oferecidos pelos mecanismos de interação contidos em $\mathbf{Sen}_{\text{ABL}}$. Esse processo também pode pre-processar os dados coletados e transformá-los para se adequar melhor ao raciocínio a ser aplicado sobre eles. Também é possível que **Mon** aplique estratégias mais sofisticadas, tais como filtragem e previsão(*forecasting*), para aumentar a eficiência do mecanismo regulatório. **Mon** comunica-se com **Mem** para ter orientação sobre transformações e estratégias nesta fase.

- **Rea** é o processo de raciocínio, também conhecido como análise. Com base nas informações de **Mon** este processo identifica situações disfuncionais e elabora um conjunto de potenciais respostas para elas, sempre considerando o objetivo interno e a configuração atual do $\mathbf{Sys}_{\text{ABL}}$. Este raciocínio está sempre em estreita interação com **Mem**, a partir da qual consulta e recupera o conhecimento histórico sobre experiências semelhantes, estratégias e políticas de interesse e técnicas de negociação para a resolução de conflitos. **Rea** também submete à **Mem** o resultado de seu processo para assim contribuir com a construção de conhecimento para novos eventos.

- **Sel** é o processo que avalia as respostas potenciais retornadas por **Rea** aplicando a medida de desempenho ($\mathcal{p}$) sobre elas, compara-as e seleciona a resposta final com base nos critérios de aceitação ($\mathcal{W}$) e nos mecanismos homeostáticos (Hom) do sistema.

- **Ex** é o processo encarregado a implementar a resposta selecionada por **Sel**. Este processo decompõe a resposta em partes que podem ser executadas pelas primitivas operacionais (*Building blocks*) fornecidas pelo $\mathbf{Sys}_{\text{ABL}}$ através do $\mathbf{Act}_{\text{ABL}}$. Este processo também se encarrega de aplicar as respostas homeostáticas e enviar à **Mem** algumas informações de feedback.

- **Mem** representa a base de conhecimento e alguns operadores básicos para gerenciar sua estrutura e informações. Alguns destes operadores referem-se à codificação do conhecimento e outros à sua manutenção (aquisição, atualização, eliminação/vencimento, etc.). A estrutura de **Mem** pode ser desde um registro relativamente simples a cumulativo até uma rede complexa de elementos interconectados. Assim como especificado, as informações contidas em **Mem** se incrementam com cada nova experiência, por tanto, em alguns casos é possível que estratégias para manter a coerência das informações sejam necessárias, i.e. estratégias de revisão de crenças.

Com os componentes de AL e $\mathbf{Sys}_{\text{ABL}}$ descritos é possível propor uma definição para $\mathbf{Sys}_{\text{ADA}}$.

Definição 10 (Sistema Adaptativo) *Dizemos que uma entidade* $\mathbf{Sys}_{\text{ADA}}$ *é adaptativa se:*

(a) ele compreende um subsistema adaptável $\mathbf{Sys}_{\text{ABL}}$

(b) preenche os requisitos de autonomia básica, e

(c) possui uma lógica adaptativa que, com base nos elementos fornecidos pela sua condição autônoma e em coordenação com uma base de conhecimento (ou memória) implementa um mecanismo para elaborar e conduzir respostas adequadas a determinados estímulos considerando informações procedentes de experiências anteriores. A implementação dessas respostas modifica a própria configuração do sistema utilizando suas funcionalidades disponíveis.

As Figuras 57 e 58 representam os elementos de AL e $\mathbf{Sys}_{\text{ADA}}$ respectivamente.

6.4 Aplicação do Meta-modelo Holístico

6.4.1 Instanciação do meta-modelo

Para acrescentar comportamentos adaptativos em um dispositivo subjacente sob o meta-modelo holístico de adaptatividade é necessário que o responsável por esse procedimento, ao qual chamaremos “designer da adaptatividade”, instancie o meta-modelo aplicando os passos a seguir:

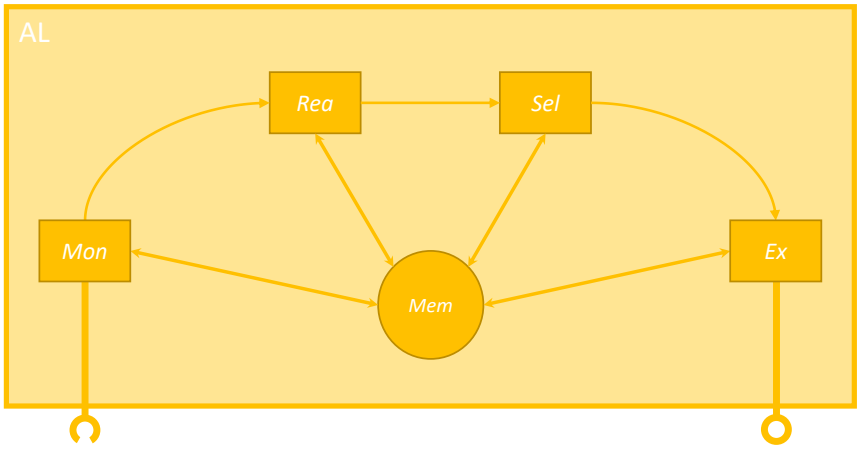


Figura 57 – Elementos que compõem AL.
O Autor (2021).

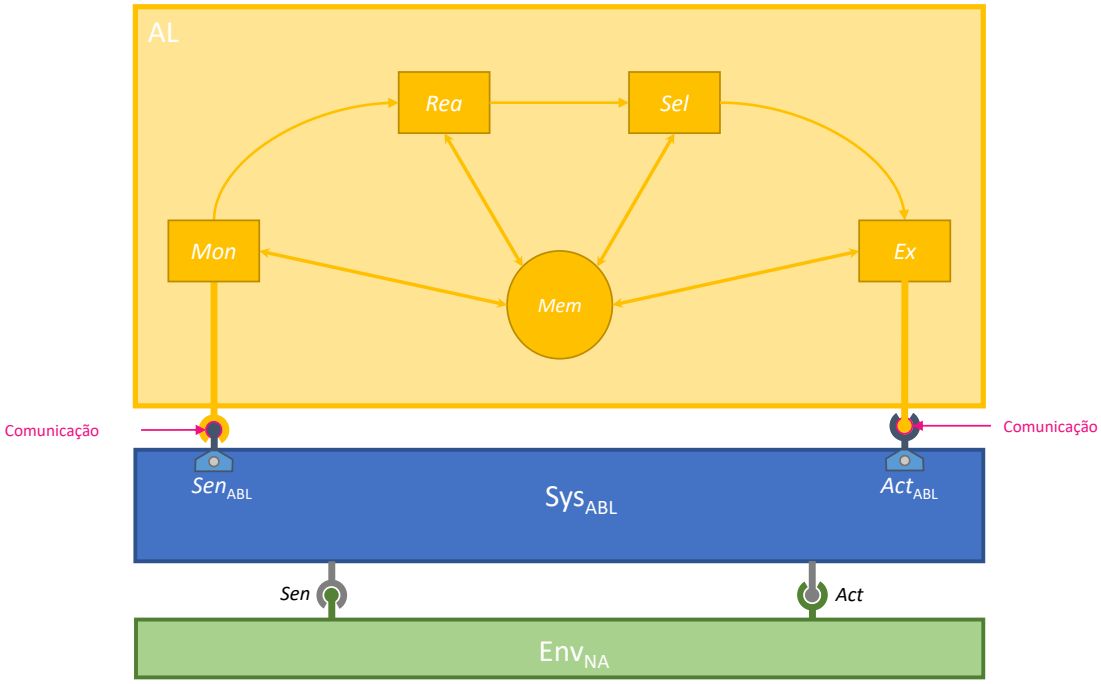


Figura 58 – Elementos de um Sistema Adaptativo.
O Autor (2021).

1. Verificar se o dispositivo subjacente, denominado Sys_U no meta-modelo, descreve um dispositivo adaptável. Para isso são necessárias, como mínimo 3 verificações:

   - Verificar que Sys_U possua mecanismos para modificar as propriedades ou funcionalidades de interesse;
   - Verificar que esses mecanismos sejam acessíveis para outros sistemas/componentes no contexto de execução; e
   - Verificar que Sys_U possua um mecanismo de comunicação através do qual é possível conectar um mecanismo ou lógica adaptativa e garantir o fluxo de controle.

   Se Sys_U não é adaptável, o designer da adaptatividade deve avaliar se é possível e viável utilizar técnicas de encapsulamento (*wrapping*) para fornecer a adaptabilidade requerida. Se o dispositivo Sys_U não é adaptável e não for possível incorporar essa flexibilidade, então, o dispositivo Sys_U descreve um dispositivo fechado e, por tanto, não pode manifestar comportamentos adaptativos finalizando o procedimento. Se, pelo contrário, as verificações forem realizadas com sucesso, o dispositivo Sys_U é considerado adaptável a respeito de um conjunto de propriedades ou características, correspondendo com o Sys_{ABL} da Figura 58, e é possível continuar com a sequência de passos.

2. Testar os mecanismos que permitem as modificações de interesse em Sys_{ABL}. Os testes conferem a funcionalidade fornecida pelos mecanismo, sua acessibilidade e o funcionamento adequado do mecanismo de comunicação.

3. Instanciar o meta-modelo holístico de adaptatividade apropriado para o tipo de dispositivo de Sys_{ABL}. Uma instância do meta-modelo é um modelo concreto que, neste caso, contém a estrutura necessária do mecanismo adaptativo para um tipo de dispositivo em particular. Podem existir diferentes instâncias do meta- modelo, dependendo, por exemplo, de:

   - Critérios de Programação: linguagem, paradigma, entre outros;
   - Versatilidade: abordagem genérica ou especializada; e/ou
   - Recursos.

   Essa relação entre meta-modelo e modelo é representada pela Figura fig:metamodelo-modelo.

4. Verificar a criação dos componentes da lógica adaptativa contidos no modelo e a sua adequada comunicação.

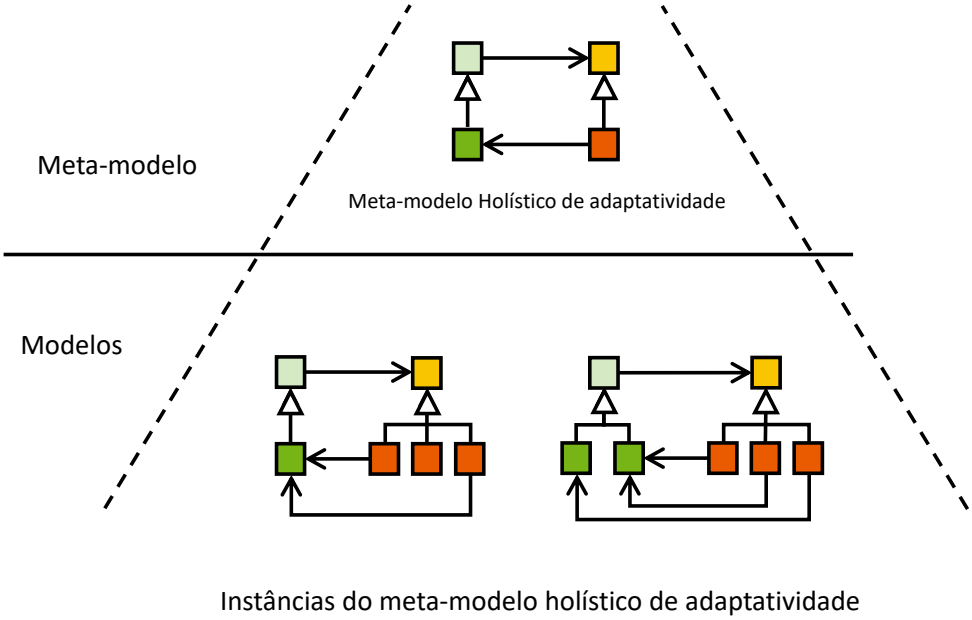


Figura 59 – Relação entre meta-modelo e modelos.
O Autor (2021).

5. Implementar a passagem de informações tanto entre os componentes da lógica adaptativa quanto com a base de conhecimento. É necessário criar as estruturas das informações de interesse para a lógica adaptativa, assim como os mecanismos para salvar e recuperar essas informações da base de conhecimentos. Nesta fase é necessário estabelecer o acesso às diferentes bases de conhecimento disponíveis para a lógica adaptativa, por exemplo: bases de dados, serviços, componentes especializados, políticas, regras de resolução de conflitos e negociações.

6. Implementar as funcionalidades desejadas nos componentes da lógica adaptativa com base nos mecanismos fornecidos pelo Sys_{ABL}. No meta-modelo holístico cada componente da lógica adaptativa é responsável por uma tarefa: monitorar, raciocinar, selecionar e executar, assim, é necessário respeitar essa estrutura em concordância com a arquitetura do modelo instanciado. Por exemplo, se o modelo contempla a lógica adaptativa como um único componente de software com as diferentes tarefas como mecanismos, então o escopo de cada mecanismo deve respeitar a lógica da tarefa. Se, pelo contrário, o modelo contempla uma estrutura de diferentes componentes para cada tarefa, então cada um desses componentes encapsula uma tarefa associada.

7. Conectar o dispositivo com a lógica adaptativa através do mecanismo de comunicação disponível.

8. Inicializar o funcionamento da lógica adaptativa.

9. Inicializar o funcionamento do dispositivo Sys_{ABL}.

Os dois últimos passos do procedimento merecem uma atenção especial. Em primeiro lugar, a sequência supõe que o dispositivo subjacente não está em funcionamento durante o procedimento de incorporação da adaptatividade, a maneira de representar um cenário ideal. Contudo, é possível que o dispositivo subjacente não possa pausar suas atividades e esteja em funcionamento, nesse caso o último passo da sequência é desnecessário. Se o dispositivo subjacente não estiver em funcionamento, é preciso reativar suas funções. Nesse caso é necessário escolher uma ordem de ativação entre o mecanismo adaptativo e o dispositivo. O caso ideal seria ativar primeiro o mecanismo adaptativo para evitar perder potenciais eventos disparados pelo dispositivo subjacente. Se o dispositivo subjacente for ativado em primeiro lugar, existe a possibilidade de perder algum evento a ser monitorado pelo mecanismo adaptativo antes de ele ser ativado completamente. A consequência neste último caso é o ponto no tempo a partir do qual o dispositivo pode apresentar comportamentos adaptativos, e não a lógica com a qual os eventos são processados. Assim, a ordem dos últimos passos no procedimento correspondem tanto com a natureza do dispositivo, quanto com as decisões do designer da adaptatividade.

Da mesma maneira, como mencionado nos objetivos do trabalho, o meta-modelo holístico foca-se principalmente no trabalho com sistemas legados para cobrir a lacuna identificada no Capítulo 4. No entanto, o meta-modelo pode ser utilizado no desenvolvimento de novos sistemas sempre que o desacoplamento entre funcionalidade e lógica adaptativa seja respeitado.

6.4.2 Paradigmas de Programação

Como mencionamos na seção anterior, um dos passos para a aplicação do meta-modelo holístico em sistemas computacionais é a implementação de uma instância do meta-modelo. Essa implementação pode ser realizada seguindo diferentes paradigmas de programação e decisões de arquitetura. O meta-modelo não está sujeito a nenhum paradigma ou linguagem de programação em particular para o desenvolvimento adaptativo. Assim, o designer de adaptatividade possui liberdade para escolher trabalhar com qualquer recurso sempre que garantir o cumprimento das condições do meta-modelo. No entanto, existem alguns paradigmas e linguagens de programação que incorporam recursos que apresentam proximidade com os conceitos de adaptatividade, podendo facilitar alguns processos da implementação. O Apêndice C na Figura 105 e Figura 106 apresenta uma lista com vários paradigmas de programação relacionados. A continuação descrevemos brevemente dois alguns deles com maior detalhe.

6.4.2.1 Programação orientada a aspectos

A Programação orientada a Aspectos (POA) é um paradigma de programação que permite que os programadores organizem o código utilizado num sistema segundo sua utilidade ou importância de uso na aplicação, permitindo que o programador encapsule o

código secundário em módulos separados do restante da aplicação (separação de interesses). A POA surgiu como uma estratégia complementar à Programação Orientada a Objetos (POO) para permitir gerenciar funcionalidades não relacionadas às regras de negócio com alta coesão e baixo acoplamento. Com a POA essas ações podem ser alocadas em partes separadas do sistema e reutilizadas. A POA se baseia nos seguintes conceitos principais:

- Aspecto (*aspect*): unidade não funcional (característica estrutural ou comportamental) que envolve diversos componentes funcionais e deve ser incorporada num sistema via uma linguagem de aspectos.
- Interesses transversais (*cross-cutting concerns*): também chamados de interesses ortogonais, são interesses que se aplicam a diversas partes do sistema de maneira simultânea quebrando o princípio de alta coesão.
- Pontos de junção (*joint points*): partes do sistema que precisam de controle transacional. Quando a execução passa por um joinpoint o aspecto pode agir naquele ponto.
- Conjuntos de junção (*pointcuts*): elementos da semântica da linguagem de componentes com os quais os programas de aspectos coordenam. Eles também representam um conjunto de pontos de junção cujo objetivo é reunir informações a respeito do contexto dos pontos selecionados. Assim, eles agem como gatilhos (*triggers*) sobre os pontos de junção.
- Advices: são os procedimentos realizados nos pontos de junção em um determinado momento a ser escolhido para execução: antes (*before*), depois (*after*) ou após o retorno (*after returning*) , após a ativação (*after throwing*) e ao redor (*around*). Os advices são a implementação dos interesses transversais.
- Regras de junção: código relativo aos interesses ortogonais que deve ser executado nos pontos de junção.
- Processo de combinação (*weaving*): processo de composição, que consiste em aplicar os aspectos sobre os componentes. É importante perceber que ao desenvolver sistemas sob a POA a lógica de controle é invertida, pois são os aspectos que conhecem os componentes do sistema, e os componentes desconhecem os aspectos.
- Combinador de aspectos (*aspect weaver*): processador de linguagem especial que oferece suporte à composição entre os componentes e os aspectos. O weaver pode ser estático, aplicado em tempo de compilação do sistema, ou dinâmico, aplicado durante a execução do código.

A POA pode ser atrativa para a implementação de adaptatividade, pois oferece: (i) suporte para a separação de interesses e (ii) diversas possibilidades temporais para

a execução de aspectos no fluxo de funcionamento do sistema. Assim, o mecanismo de comunicação da adaptabilidade fica garantido pelos recursos da POA e os componentes da lógica adaptativa podem ser implementados como aspectos e serem combinados ou compostos (KRUPITZER *et al.*, 2015) com a lógica do negócio que podem alterar o comportamento do sistema pela aplicação do comportamento adicional contido nos advices.

6.4.2.2 Programação orientada a contexto

A Programação Orientada ao Contexto (COP) é um paradigma de programação que fornece aos programadores abstrações e mecanismos dedicados para representar de maneira direta variações comportamentais dependentes do contexto de execução (COSTANZA; HIRSCHFELD; NIERSTRASZ, 2008). Dessa maneira, a POC oferece suporte direto à composição dinâmica, permitindo que os programadores expressem melhor as entidades de software que adaptam seu comportamento em tempo de execução. Para a POC o contexto é definido de forma bastante ampla e aberta: é qualquer informação acessível computacionalmente (COSTANZA; HIRSCHFELD; NIERSTRASZ, 2008), assim, atores, o próprio sistema e o ambiente fazem parte do contexto do sistema. No entanto, na maioria das vezes, o contexto é considerado como a informação detectada do ambiente por sensores ou mudanças internas do sistema.

Um dos principais conceitos do COP são as variações comportamentais que expressam definições parciais do comportamento básico dos sistemas, que podem substituir ou modificar parcialmente comportamento da lógica da aplicação e que podem ser entendidas como interesses transversais que se espalham pela base de código. É através da ativação dessas variações comportamentais em tempo de execução que a POC pode refletir adaptações de contexto. Quando várias variações estão ativas simultaneamente, elas se combinam dinamicamente para gerar o comportamento emergente do sistema. Outro conceito interessante é a ideia de camadas. Segundo Costanza, Hirschfeld e Nierstrasz (2008) uma camada é uma abstração da linguagem que permite a agrupação das definições parciais do comportamento relacionados a um mesmo aspecto na mesma unidade conceitual. Na maioria das linguagens que implementam COP, as camadas são entidades de primeira classe no sentido de que podem ser atribuídas a variáveis, passadas como parâmetros de função e retornadas como valores. Esse é o mecanismo pelo qual diferentes partes do programa podem comunicar a adaptação a ser realizada (SALVANESCHI; GHEZZI; PRADELLA, 2012). Adicionalmente, as camadas podem ser ativadas ou desativadas em tempo de execução com base no contexto corrente.

A POC é um paradigma de programação que suporta diretamente a adaptação dinâmica de software em relação ao contexto de execução com base na agregação de frações de código organizadas em camadas que formam unidades conceituais. Os elementos da lógica adaptativa definidos no meta-modelo holístico podem ser facilmente traduzidos

em camadas a serem ativadas por eventos detectados. Um conceito interessante deste paradigma é a desativação de camadas dependendo do contexto, esta característica permite pensar na incorporação de técnicas de otimização na própria lógica adaptativa.

Como mencionado anteriormente existem diversos paradigmas que podem ser utilizados para a implementação de sistemas com comportamento adaptativo, os trabalhos de Krupitzer *et al.* (2015), Geihs (2013) e Salvaneschi, Ghezzi e Pradella (2012) são excelentes pontos de partida para revisões mais profundas sobre o assunto.

6.5 Meta-adaptatividade

Sintaticamente, o termo *meta-* vem do prefixo grego $\mu\epsilon\tau\alpha$ que significa *"depois"* ou *"além"* sendo usado para indicar um conceito que é uma abstração por trás de outro conceito com a finalidade de completar ou adicionar ao último. Como apontado por Ferreira (2010) na epistemologia e outros ramos da filosofia, o prefixo *meta-* é frequentemente usado para significar "*sobre sua própria categoria*" revelando duas propriedades subjacentes: reflexão e hierarquia.

Assim, meta-adaptatividade refere-se à aplicação de adaptatividade sobre sistemas adaptativos para efetuar mudanças no próprio mecanismo adaptativo que governa a mudanças de uma entidade (BARANDIARAN; MORENO, 2008). Em outras palavras, é permitir que os próprios mecanismos adaptativos estejam sujeitos a mudanças, é adaptatividade na adaptatividade. Outros termos usados para meta-adaptatividade são: adaptatividade de segunda ordem ou adaptatividade de segundo nível, autoaperfeiçoamento, autoaprimoramento, autorregulação e, em alguns casos, é referido até como evolução.

A ideia de meta-adaptatividade, da mesma forma que acontece com a adaptatividade, encontra sua inspiração nas ciências naturais. Em Biologia, o termo adaptação está vinculado ao comportamento de um organismo durante sua vida. No entanto, as características adaptativas são transmitidas através da genética, superando a limitação temporal estabelecida por um indivíduo. A informação genética dos indivíduos é processada por um mecanismo de nível superior que afetará toda a espécie através do genótipo, mudando o código genético da nova geração da espécie. Este processo é chamado de evolução e pode ser entendido como meta-adaptatividade, pois age sobre os mecanismos adaptativos que procuram garantir a sobrevivência e bem-estar do organismo vivo. Os registros mostram que a meta-adaptação vem sendo estudada na Biologia, na forma de evolução, há mais de um século, e nas áreas tecnológicas assuntos relacionados podem ser observados em campos como a Computação Evolutiva tão cedo quanto 1948 com trabalhos do próprio Alan Turing.

Nas áreas tecnológicas a meta-adaptatividade tem sido foco de diversas pesquisas como uma possível resposta para superar as limitações estabelecidas pelo denominado "envelope adaptativo" (GEROSTATHOPOULOS *et al.*, 2015) do sistema. O termo "envelope adaptativo" tem sido usado para descrever as decisões de design que estabelecem os contextos aos quais o sistema é capaz de se adaptar: gatilhos, casos de sistemas, limites, recursos, etc. Essas decisões em tempo de design ajudam a desenvolver as estratégias de adaptatividade a ser executadas em tempo de execução. No entanto, mesmo com esse interesse pelo estudo da meta-adaptatividade, uma das maiores dificuldades para sua aplicação é que a maioria dos modelos sobre a questão a analisam separadamente da adaptatividade quando é necessário um modelo que contemple ambos os níveis explicitamente.

Na Computação Autonômica ela é minimamente representada sobre o conceito de Gerenciadores Autonômicos de Orquestração, e sua função é pouco desenvolvida na arquitetura proposta pela IBM. No caso dos SaS, a meta-adaptatividade é representada por vários conceitos, sendo os mais comuns: autoaperfeiçoamento, autoaprimoramento, e autorregulação. Segundo a maioria de abordagens de SaS cada um desses conceitos pode ser modelado e implementado de maneira independente, e em muitos casos não existe uma distinção clara entre os mecanismos adaptativos que servem de base para a melhoria. Na abordagem de Sistemas Complexos, a meta-adaptatividade está representada na capacidade dos sistemas de aprender, se auto-organizar, criar padrões emergentes e evoluir com cada mudança. Nesta abordagem é possível modelar o sistema complexo como população de agentes que possuem regras de vida, morte e reprodução, podendo aplicar diferentes mecanismos evolutivos. No entanto, novamente a meta-adaptatividade é uma propriedade emergente do sistema e, por tanto, não é representada em relação direta com a adaptatividade do mesmo. No caso das Tecnologias Adaptativas, o conceito de adaptatividade de segunda ordem foi explorado no trabalho de Inojosa (2012) para o caso particular de autômatos adaptativos e modelado de maneira particular. Algumas propostas, como a apresentada em Gerostathopoulos *et al.* (2015), introduzem a ideia de estratégias no nível de meta-adaptatividade, porém são colocadas de maneira específica para a resolução de alguns problemas e como parte de modelos particulares.

Assim, mesmo quando algumas abordagens sugerem a possibilidade de meta-adaptatividade, não existe um modelo explícito que permita observar este conceito como a adaptatividade aplicada num nível de abstração superior. O meta-modelo holístico proposto permite realizar esta tarefa sem incremento desnecessário de complexidade conceitual. A Figura 60 apresenta a representação de um sistema com meta-adaptatividade sob a abordagem holística.

Neste nível o ambiente $\mathbf{Env}_{\text{ADA}}$ manifesta estímulos com origem tanto no ambiente original $\mathbf{Env}_{\text{NA}}$ quanto na função da lógica adaptativa.

O sistema subjacente é representado pela lógica adaptativa AL, significando que

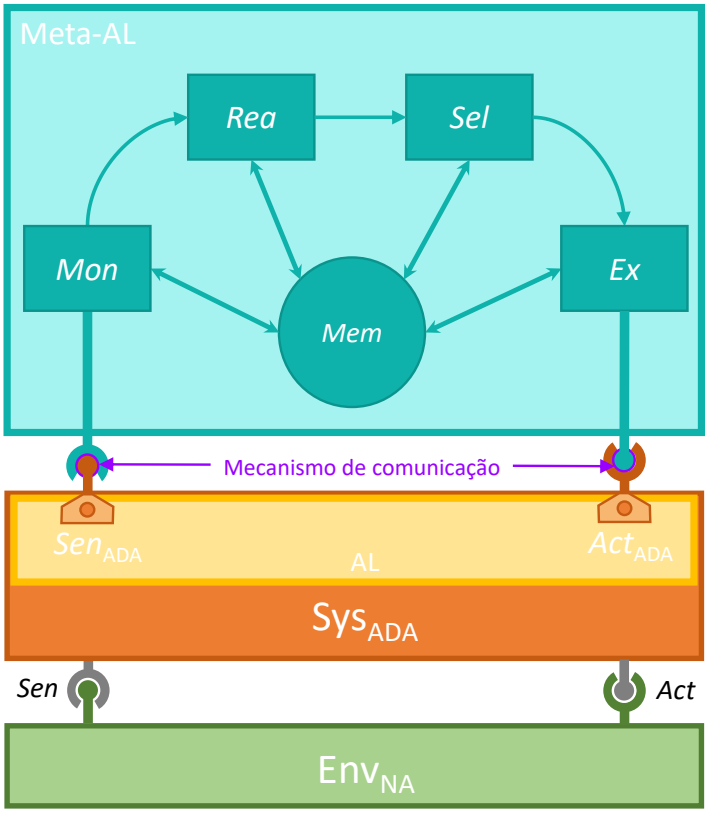


Figura 60 – Funcionamento da Meta-adaptatividade.
O Autor (2021).

são seus componentes os que devem cumprir as condições especificadas no modelo. Deve apresentar um conjunto de sensores Sen_{ADA} e atuadores Act_{ADA}, que permitem realizar mudanças na lógica adaptativa AL. É importante observar que esses elementos são diferentes de Sen_{ABL} e Act_{ABL} apresentados na proposta do modelo, porém representam a mesma ideia de interface de comunicação que no caso original, só que num nível superior. AL também deve apresentar um mecanismo de comunicação para permitir a interação com os mecanismos meta-adaptativos, e assim cumprir com a condição de ser adaptável. Este mecanismo de comunicação não precisa ser o mesmo que no caso original, pode ser diferente.

Em consequência, mesmo quando o modelo de adaptatividade está sendo aplicado duas vezes, cada nível gerencia questões diferentes mantendo uma separação de interesses, refletindo tanto as mudanças no nível individual ou subjacente quanto as mudanças no nível das próprias estratégias de mudança, enriquecendo a flexibilidade do modelo e seu desempenho.

Neste ponto é interessante observar que existe uma diferença importante entre meta-adaptatividade e evolução. Em alguns contextos das áreas tecnológicas o termo evolução é utilizado geralmente em 3 sentidos diferentes:

- para caracterizar o processo de mudança constante e aprendizado pelos quais pode passar um sistema;
- para se referir ao processo de manutenção de software em TI pelo qual as versões de um sistema são atualizadas para fazer frente às mudanças nos seus ambientes;
- para se referir aos processos de otimização global, inspirados na evolução biológica, no contexto de computação evolutiva.

No nosso caso, utilizamos evolução no terceiro sentido pois ele preserva o muito bem firmado significado biológico do termo. Assim, ao comparar o conceito de meta-adaptatividade com evolução percebemos que eles são próximos, porém não equivalentes no contexto de sistemas artificiais. Quer dizer, nos sistemas biológicos meta-adaptatividade e evolução podem ser tratados de maneira quase equivalente pois a evolução é o resultado acumulado dos processos adaptativos realizados no nível dos indivíduos só que com consequências no nível de população das espécies. Além disso, nos sistemas naturais os organismos possuem todas as condições para transmissão de informações necessárias, seja no nível biológico ou cultural (REEVE; SHERMAN, 1993), e é admitida a presença de uma entidade, de maior hierarquia, que realiza o controle centralizado desses organismos. Pelo contrário, ao falar de sistemas artificiais não é garantido que as entidades adaptativas que compõem o sistema estejam equipadas com esses recursos. A meta-adaptatividade fornece estrutura e nível de abstração para a potencial realização de evolução, mas carece de alguns conceitos importantes, entre os quais podemos identificar:

- um controle centralizado que permita registrar as informações de todos os indivíduos de uma mesma espécie;
- um parâmetro que permita distinguir entre gerações diferentes e espécies diferentes;
- representação para o código genético assim como operadores biológicos que permitam superar o limite temporal da vida de um indivíduo.

Consideramos que é possível encontrar respostas sobre como modelar evolução com base no meta-modelo holístico de meta-adaptatividade analisando tanto os estudos nas áreas biológicas quanto os trabalhos nas áreas de tecnologia que procuram preservar o significado de evolução, i.e. computação evolutiva, computação orgânica e adaptação evolutiva.

7 DESENVOLVIMENTO

Para complementar a proposta apresentada no capítulo anterior, desenvolvemos uma ferramenta que auxilia no desenvolvimento de lógicas adaptativas para dispositivos adaptáveis baseada na abordagem do meta-modelo holístico de adaptatividade, a qual denominamos HM^{2}AT (**H**olistic **M**eta-**M**odel for **A**daptivity **T**ool). Este capítulo descreve em detalhe o processo de desenvolvimento dessa ferramenta e suas funcionalidades principais.

A ferramenta procura atender dois perfis de usuários: o Designer de Adaptatividade, e o Desenvolvedor de Sistemas. O Designer de Adaptatividade é um especialista em adaptatividade, ele trabalha com alguma das abordagens de adaptatividade e conhece os conceitos apresentados no meta-modelo holístico. O Designer utiliza a ferramenta para gerenciar instâncias do meta-modelo holístico (modelos de adaptatividade), suas implementações e outros recursos relacionados. Assim, ele define estruturas para o desenvolvimento de comportamento adaptativo e a troca de informações entre seus processos. O Desenvolvedor de Sistemas, por outro lado, está interessado em utilizar (aplicar) o comportamento adaptativo para suportar um sistema particular, assim ele procurar acessar recursos relacionados à implementação dos modelos adaptativos e especializá-los para seu objetivo específico.

Baseados nesta distinção de propósito foram criadas duas componentes: o HM^{2}AT e o HM^{2}AT Admin para atender as necessidades de cada perfil de usuário. Assim, a composição de ambas peças de software é denominada HM^{2}AT Framework. A Figura 61 apresenta a relação entre ambos componentes da ferramenta e os perfis de usuários.

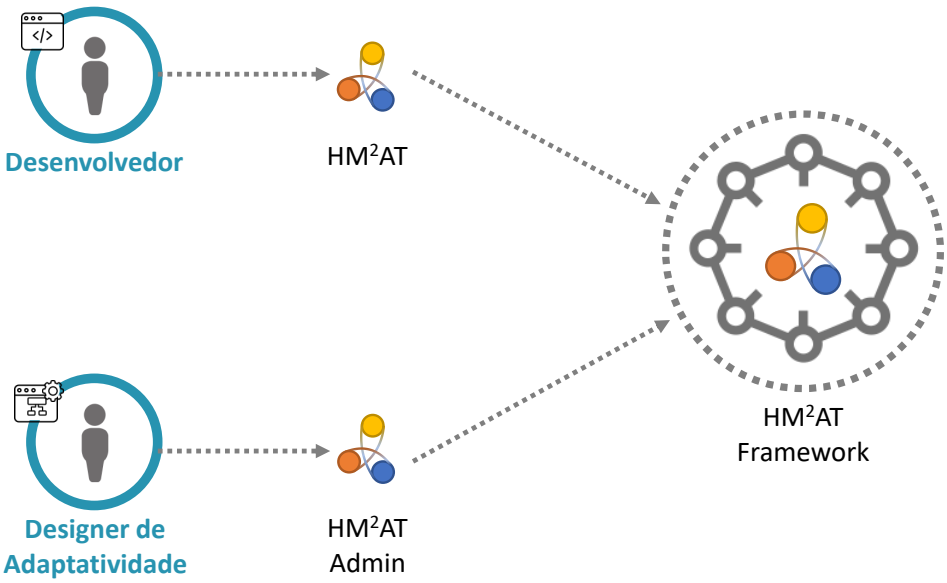


Figura 61 – Relação entre os componentes da ferramenta e os perfis de usuário.
O Autor (2022).

O Designer de Adaptatividade através do HM^{2}AT-Admin cria recursos relacionados aos modelos de adaptatividade e suas implementações, os quais são salvos no repositório da do framework. O Desenvolvedor de Sistemas acessa esses recursos através do HM^{2}AT para utilizar e especializá-los.

A implementação da ferramenta foi realizada com as seguintes especificações e recursos técnicos principais:

- Linguagem de programação Java, com o JDK 17.0.1 LTS.
- Eclipse IDE 2022-09 como plataforma de desenvolvimento
- Maven 3.8.0 como gerenciador de projeto
- GitHub como repositório de código
- Jackson para o gerenciamento de arquivos JSON e XML
- Python 3.10.2

O desenvolvimento foi realizado em um computador com sistema operacional Windows 11 nativo, processador Intel(R) Core(TM) i7 de 11a. geração com 2.80GHz de velocidade e RAM de 16GB.O código-fonte de ambos componentes da ferramenta está disponível em https://github.com/stars/rosedth/lists/hm2at-framework, repositório de implementações que acompanha este trabalho.

O framework foi projetado como dois componentes de software independentes conectados via um repositório comum. Os projetos de software de ambos componentes foram implementados no Eclipse IDE como projetos Maven. A interação entre as tecnologias utilizadas está representada na Figura 62 e a estrutura geral dos projetos para é apresentada na Figura 63.

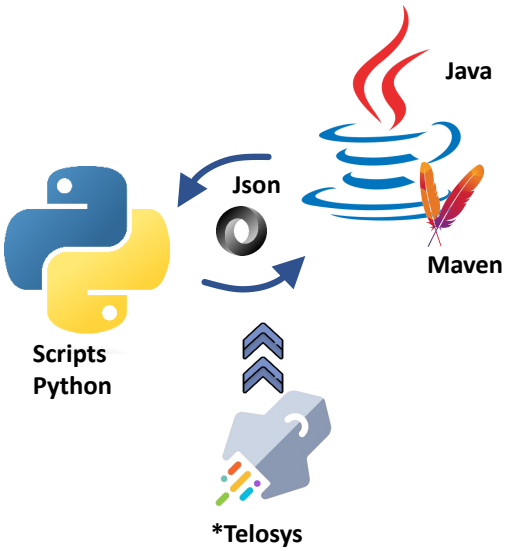


Figura 62 – Tecnologias utilizadas na implementação do framework HM^{2}AT.
O Autor (2022).

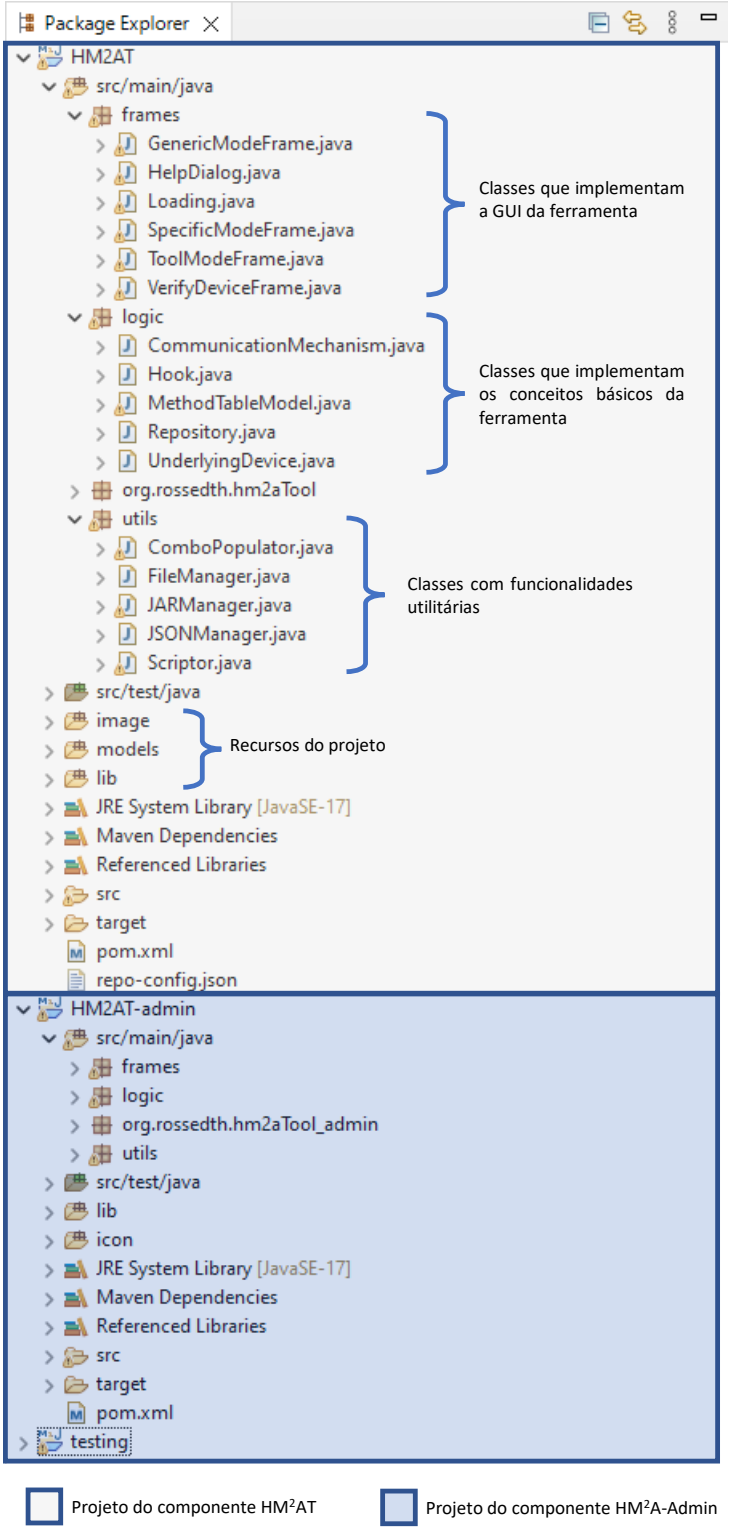


Figura 63 – Estrutura de projetos em Eclipse IDE de ambos componentes do framework HM²AT.

O Autor (2022).

A estrutura do repositório comum aos dois componentes do framework está descrita na Figura 64. Essa estrutura é criada automaticamente através das funcionalidades de configurações do componente HM^{2}AT-Admin.

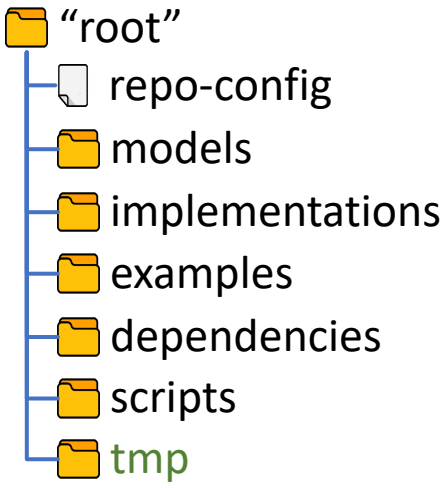


Figura 64 – Estrutura do repositório comum do framework HM^{2}AT.
O Autor (2022).

A seguir detalhamos ambas componentes do HM^{2}AT Framework e descrevemos como os usuários de cada uma podem utilizar as funcionalidades oferecidas, bem como a implementação dos componentes.

7.1 HM^{2}AT-Admin

Este componente do HM^{2}AT Framework apresenta diversas opções para o gerenciamento dos recursos disponibilizados associados com o desenvolvimento de sistemas adaptativos. Os recursos gerenciados pelo HM^{2}AT-Admin são armazenados no repositório unificado e acessados pelo componente HM^{2}AT do framework com a finalidade de atender as necessidades dos desenvolvedores.

A Figura 65 apresenta a tela inicial do HM^{2}AT-Admin a partir da qual é possível acessar um menu de navegação do tipo *drawer* que mostra as funcionalidades disponíveis tanto para o gerenciamento dos elementos básicos da lógica do framework quanto das configurações da ferramenta.

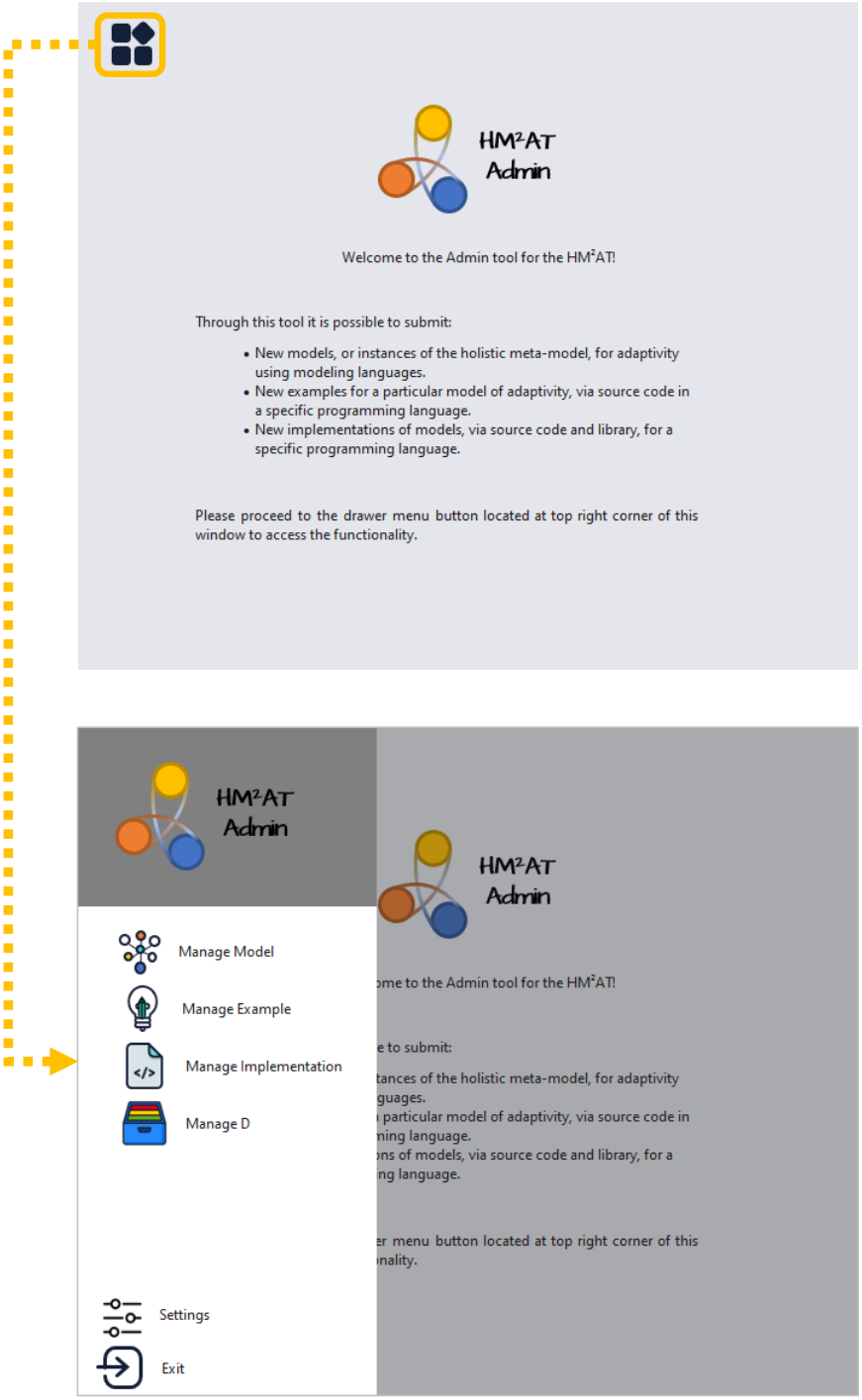


Figura 65 – Tela de apresentação do HM^{2}AT-Admin.
O Autor (2022).

A partir dessa tela o sistema fornece suporte para a realização dos casos de uso mostrados na Figura 66, a maioria dessas funcionalidades está associada ao gerenciamento das entidades principais da ferramenta: modelos, implementações, exemplos e dependências. Adicionalmente, o componente apresenta uma funcionalidade para estabelecer as configurações relacionadas ao repositório comum e atualização de índices e recursos.

O detalhe de cada um dos casos de uso da ferramenta está especificado no “Documento de Especificação de Casos de Uso” disponível no repositório científico que acompanha este trabalho [1], na página de “Contribuições”, Seção “Contribuições Técnicas”.

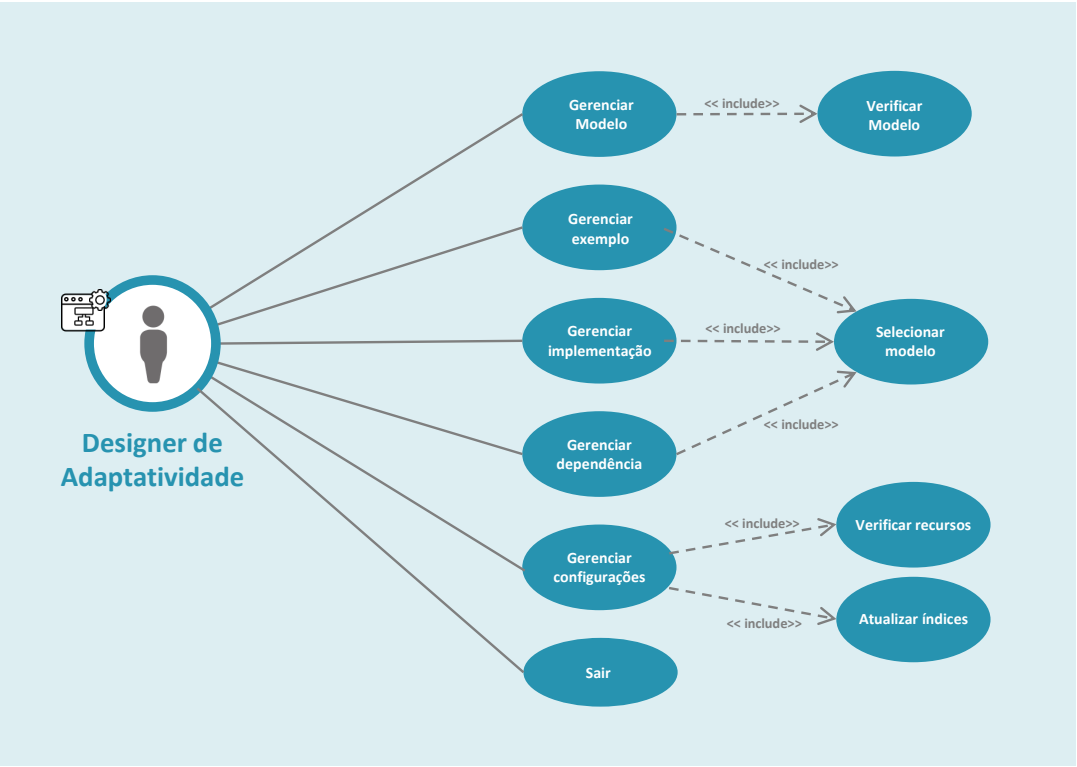


Figura 66 – Casos de uso relacionadas ao HM^{2}AT-Admin.
O Autor (2022).

A continuação apresentamos as telas da ferramenta HM^{2}AT-Admin que implementam os casos de uso descritos. A Figura 67 apresenta a tela de incorporação de um novo modelo de adaptatividade. Os modelos são a entidade base da ferramenta, é a partir dela que as implementações, dependências e exemplos podem ser organizadas.

A Figura 68 apresenta a tela para o registro de novas implementações para um modelo determinado. Cabe ressaltar que a ferramenta solicita ao designer de adaptatividade o cadastro de um script associado com a implementação do modelo. Esse script contém a sequência de ações necessárias para especializar a implementação e será utilizada pelo componente HM^{2}AT quando o desenvolvedor solicitar a especialização da implementação para a construção de uma lógica adaptativa particular.

A Figura 69 apresenta a tela para o registro de dependências associadas a uma implementação. Dependendo do gerenciador de dependências utilizado, o designer de adaptatividade pode registrar dependências para uma implementação em várias linguagens de programação.

[1] https://sites.google.com/view/rosalia-caya-thesis-document/contribui%C3%A7%C3%B5es

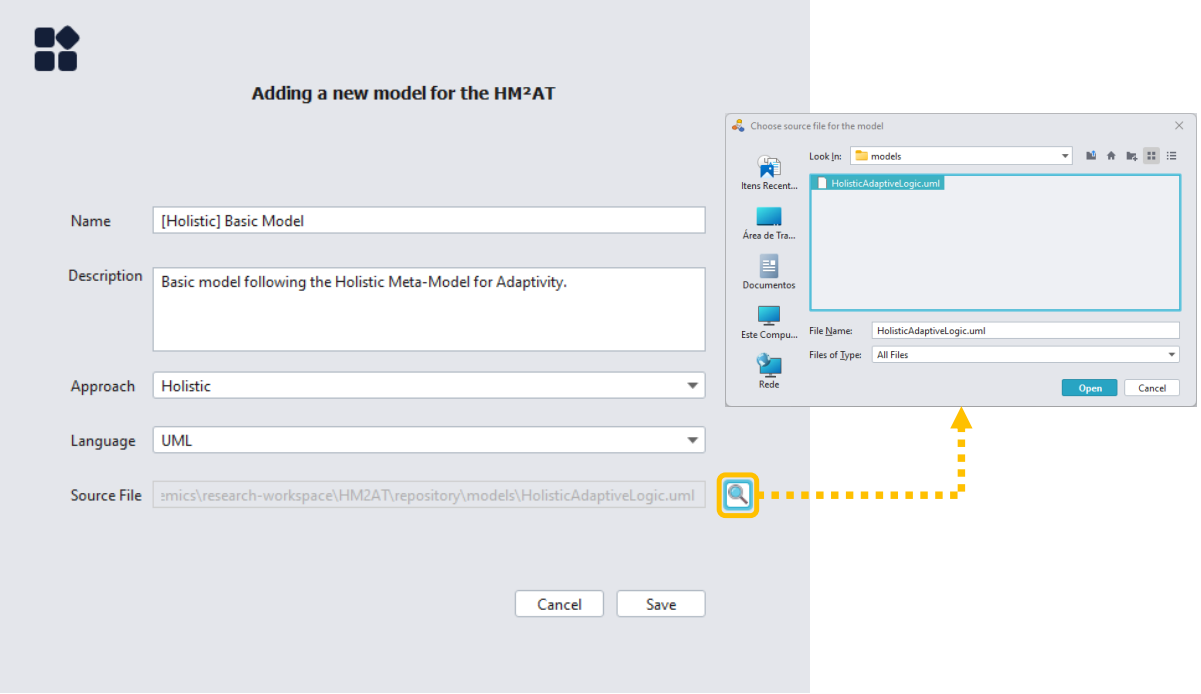


Figura 67 – Tela de incorporação de modelos no HM²AT-Admin.
O Autor (2022).

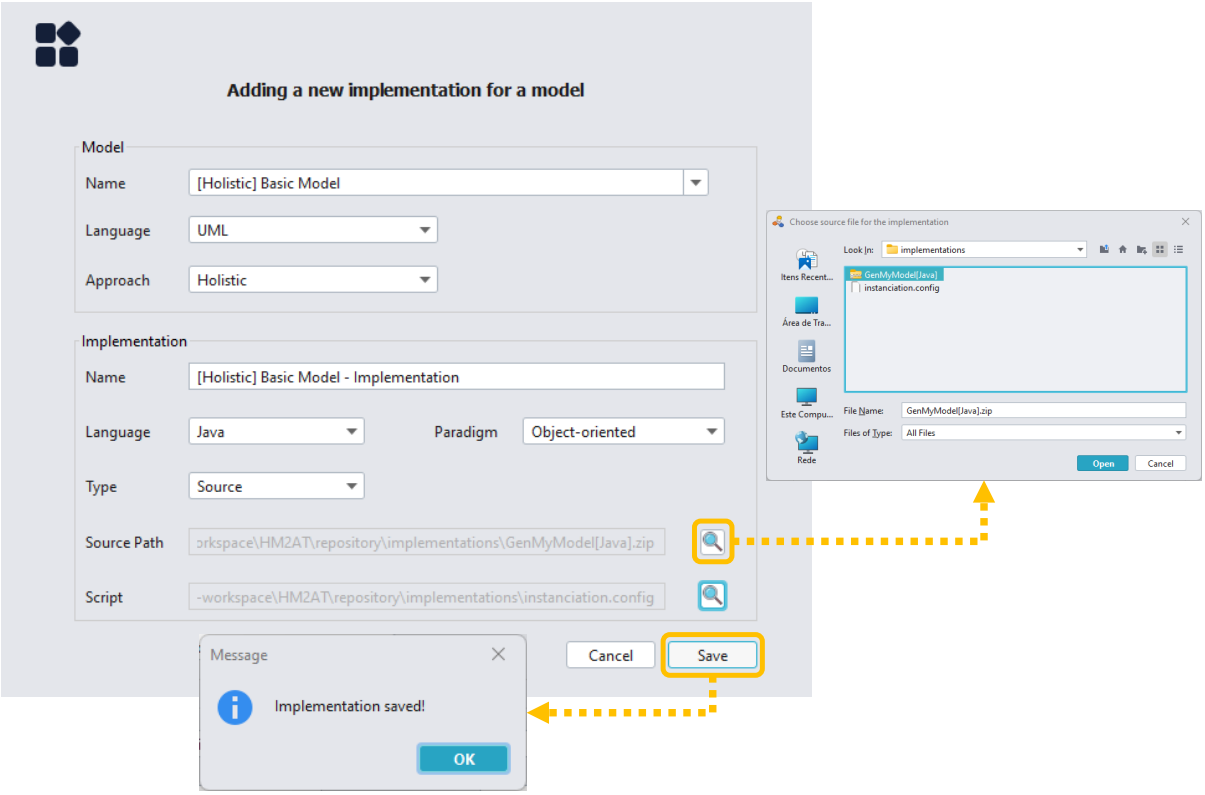


Figura 68 – Tela de incorporação de implementações no HM²AT-Admin.
O Autor (2022).

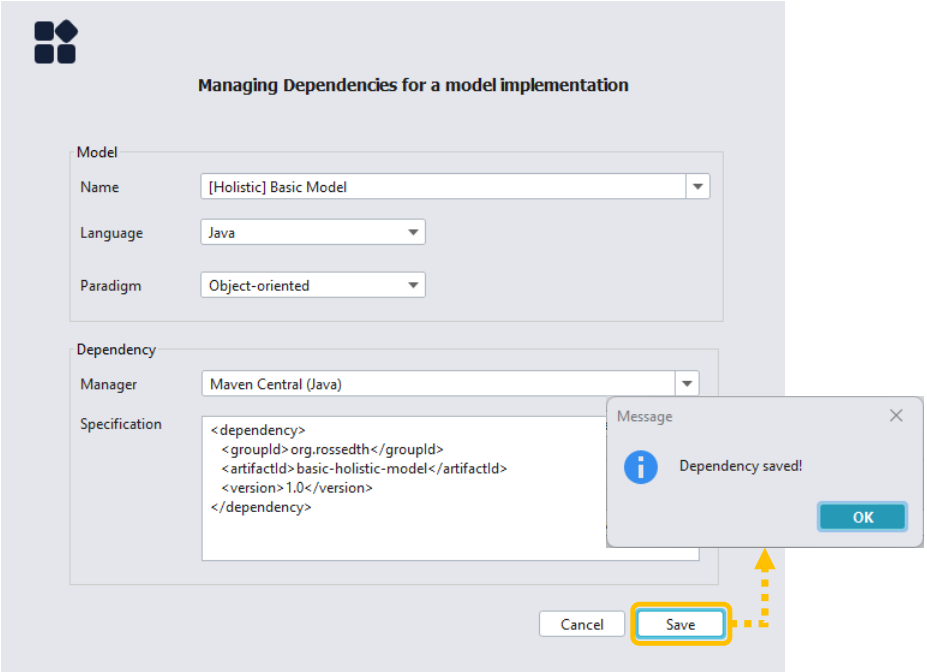


Figura 69 – Tela de incorporação de dependências no HM²AT-Admin.
O Autor (2022).

Finalmente, a Figura 70 apresenta a tela de incorporação de exemplos de aplicação de um modelo determinado.

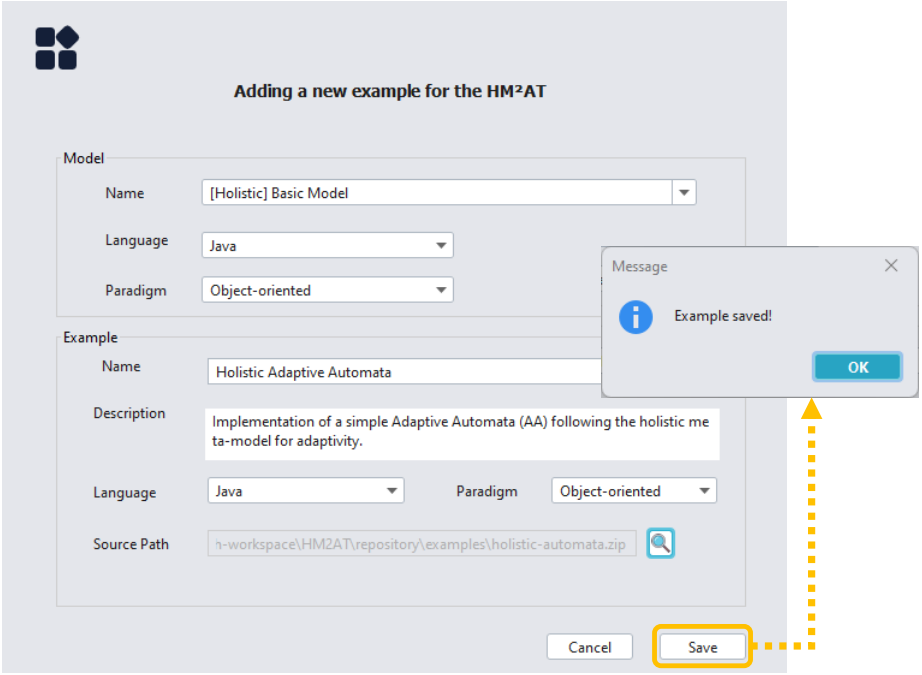


Figura 70 – Tela de incorporação de exemplos no HM²AT-Admin.
O Autor (2022).

7.2 HM^{2}AT

Este componente do HM^{2}AT Framework permite que o desenvolvedor de software, o perfil de usuário do componente, acesse aos diferentes recursos dos modelos de adaptatividade gerenciados pela ferramenta. A tela de entrada do componente é apresentada na Figura 71.
O carregamento deste componente realiza duas operações: a busca do arquivo de configuração do repositório de trabalho e a atualização dos índices dos recursos disponíveis. A ferramenta precisa da existência do arquivo "repo-config.json" para carregar, caso o arquivo não seja encontrado a ferramenta informa ao desenvolvedor o erro encontrado.

A configuração do repositório pode ser realizada desde o componente HM^{2}AT-Admin ou diretamente especificada em um arquivo de configurações em Json com o formato correspondente. Um exemplo de arquivo de configurações é apresentado no Código 7.1.

```
{
  "type": "local",
  "path": "file:///D:/Academics/PhD/new-workspaces/tool/testing/",
  "config": "D:\\Academics\\PhD\\new-workspaces\\tool\\holisticAdaptiveTool\\repo
    -config.json",
  "directories": [
    {
      "name": "models",
      "path": "\\models"
    },
    {
      "name": "implementations",
      "path": "\\implementations"
    },
    {
      "name": "examples",
      "path": "\\examples"
    },
    {
      "name": "dependencies",
      "path": "\\dependencies"
    }
  ]
}
```

Código 7.1 – Exemplo de arquivo de configuração do repositório

A partir dessa tela o desenvolvedor pode escolher entre duas modalidades de trabalho, como mostram os casos de uso da Figura 72: a Modalidade Genérica e a Modalidade Específica. Em ambos casos, o HM^{2}AT oferece mais informações sobre a modalidade via uma tela de ajuda como mostra a Figura 73.

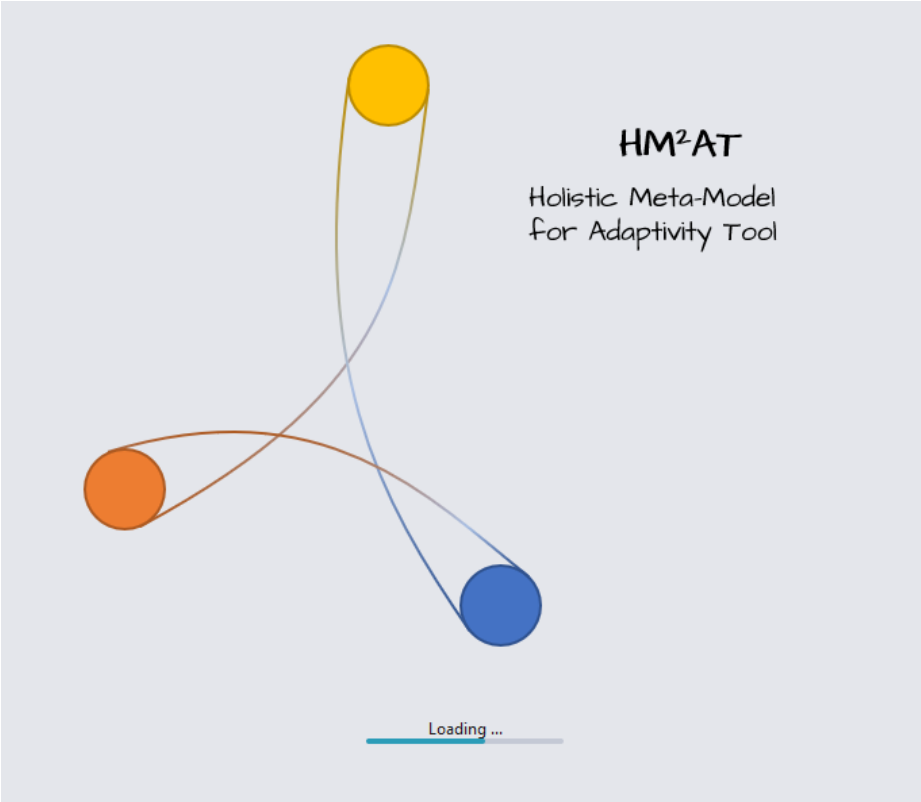


Figura 71 – Tela de apresentação do HM^{2}AT.
O Autor (2022).

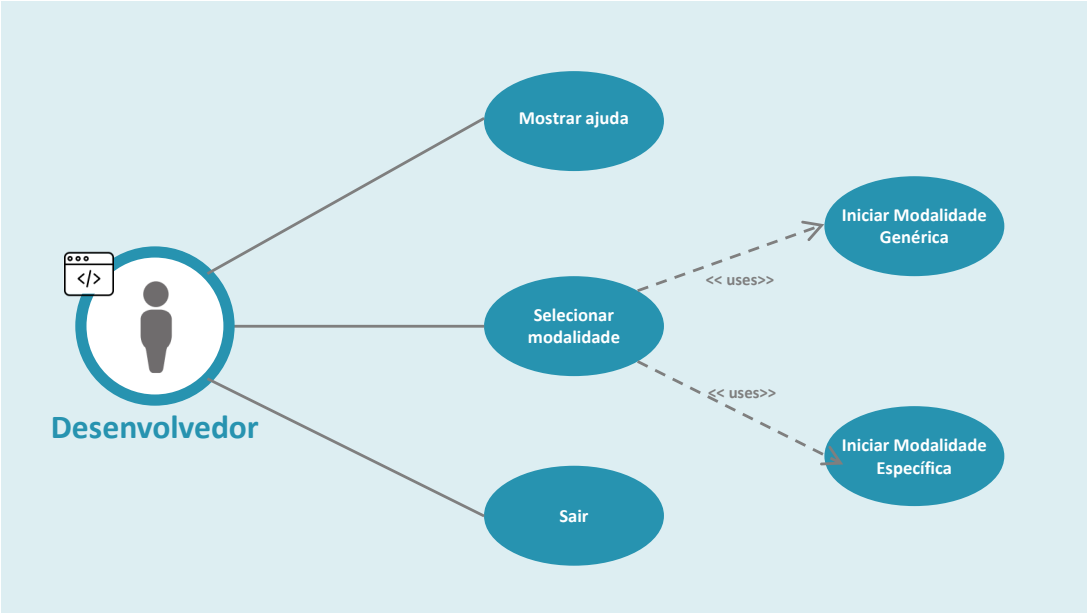


Figura 72 – Casos de uso relacionadas à tela de apresentação do HM^{2}AT.
O Autor (2022).

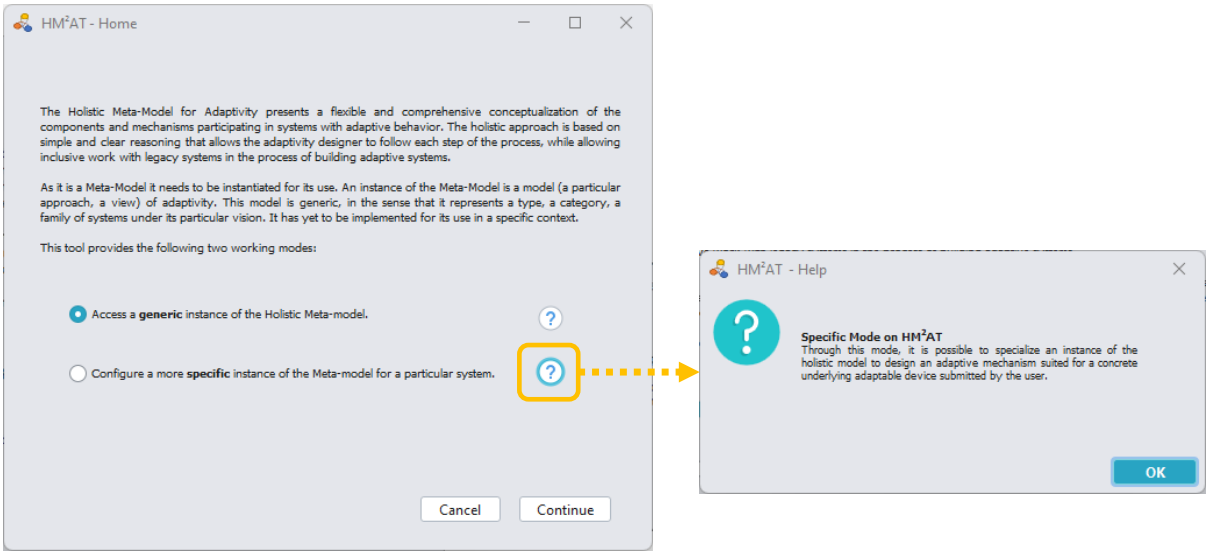


Figura 73 – Seleção de Modalidade do HM^{2}AT.
O Autor (2022).

A seguir apresentaremos maiores detalhes do funcionamento de cada modalidade e os recursos utilizados para sua implementação.

7.2.0.1 Modalidade Genérica

A modalidade genérica procura fornecer aceso aos recursos de natureza abstrata gerenciados pela ferramenta. Isso significa que o usuário poderá descarregar recursos, e não especializar ou atualizar informações, assim, essa funcionalidade age basicamente como uma interface de busca organizada para o desenvolvedor. Os principais casos de uso relacionados com esta modalidades são apresentados na Figura 74.

Na tela inicial da modalidade genérica na Figura 75 é apresentada a organização geral da modalidade: uma seção para selecionar um modelo de adaptatividade, e uma seção para explorar os recursos de implementação associados com tal modelo.

Como decisão de projeto, todos os componentes seletores de dados (*combo box*) da ferramenta apresentam listas preenchidas com valores já cadastrados nos elementos do repositório, e funcionam como filtros compostos para auxiliar o usuário na escolha de uma opção. Assim, o usuário só pode escolher combinações existentes no repositório. De maneira semelhantes, a ferramenta só habilita novas opções quando as pré-condições foram atendidas, por exemplo: o botão de descarga e as abas na seção de implementações só são habilitados após a escolha de um modelo específico como mostra a Figura 76.

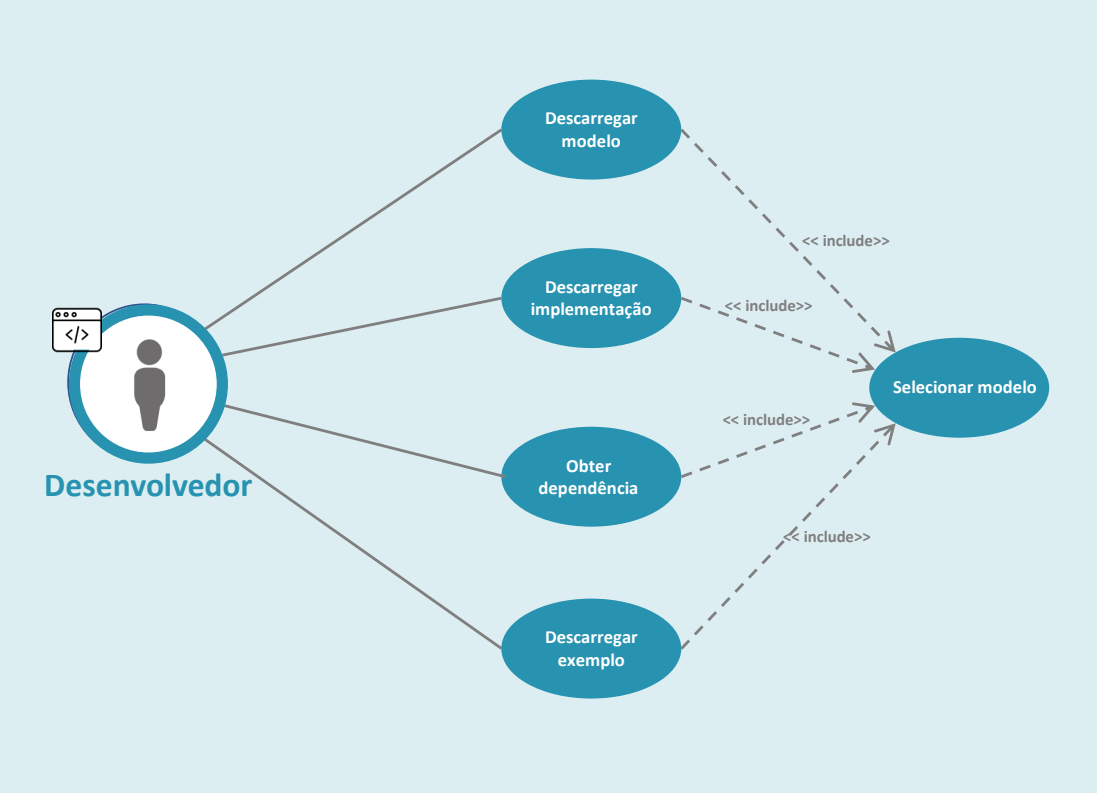


Figura 74 – Casos de uso relacionados à Tela de Modalidade Genérica do HM^{2}AT. O Autor (2022).

Após a seleção de um modelo de adaptatividade, o desenvolvedor pode descarregar o arquivo que especifica o modelo e pode consultar os recursos de implementação associados.

A recuperação dos recursos solicitados pelo usuário é realizada através do acesso ao repositório comum configurado inicialmente. Assim, os recursos registrados na plataforma através da ferramenta HM^{2}AT-Admin estão disponíveis para os desenvolvedores.

As Figuras 77, 78 79 e apresentam os casos de uso Descarregar Implementação, Obter Dependência e Descarregar Exemplo correspondentemente.

Adicionalmente, uma sequência passo a passo de cada funcionalidade pode ser encontrada no "Manual Rápido de Usuário" disponível no repositório científico que acompanha este trabalho [2], na página de "Contribuições" Seção "Contribuições Técnicas".

[2] https://sites.google.com/view/rosalia-caya-thesis-document/contribui%C3%A7%C3%B5es

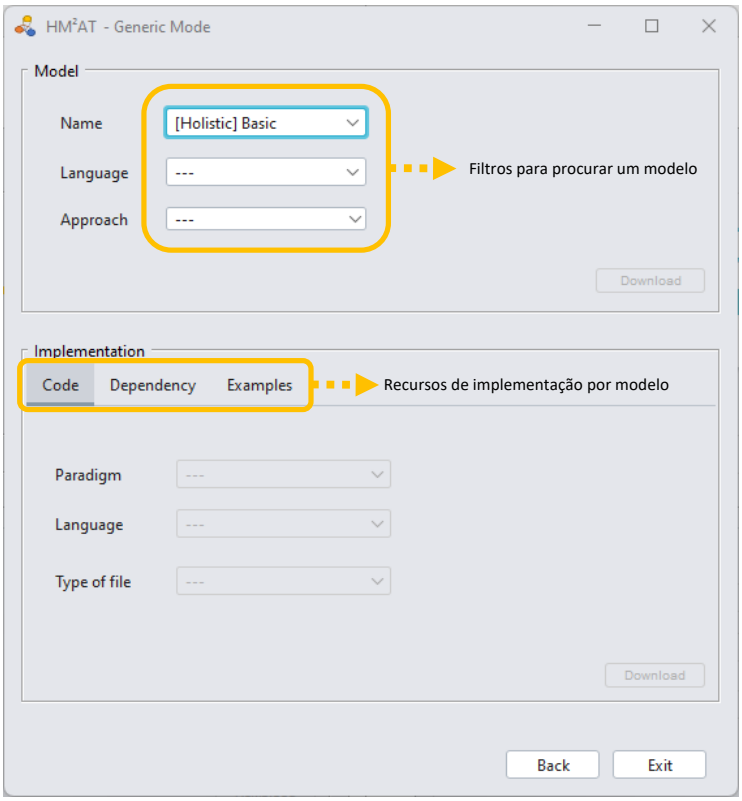


Figura 75 – Tela de Modalidade Genérica do HM²AT.
O Autor (2022).

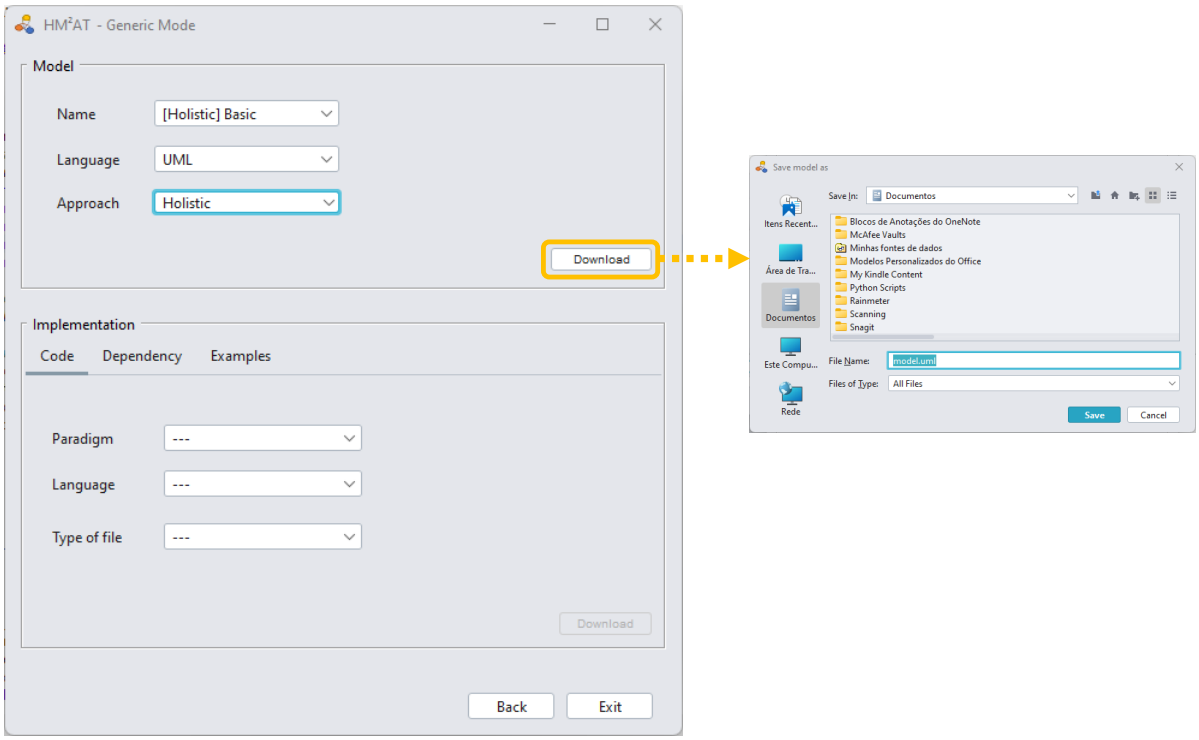


Figura 76 – Descarregar Modelo no HM²AT.
O Autor (2022).

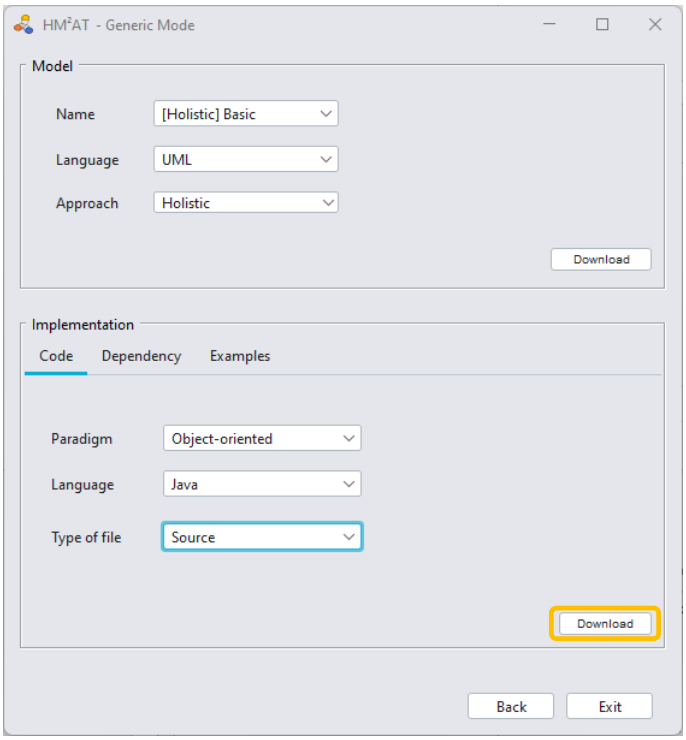


Figura 77 – Descarregar Implementação no HM^{2}AT.
O Autor (2022).

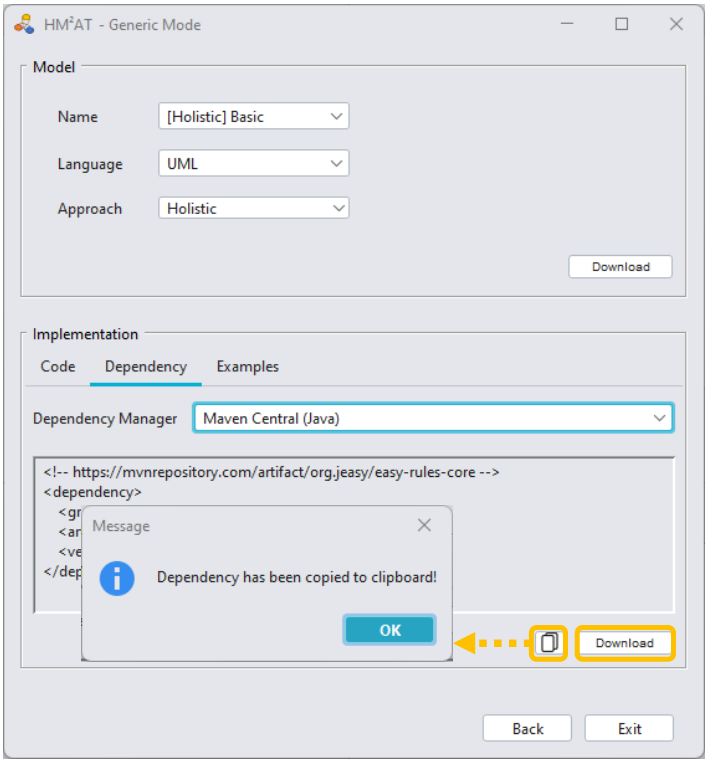


Figura 78 – Descarregar Dependência no HM^{2}AT.
O Autor (2022).

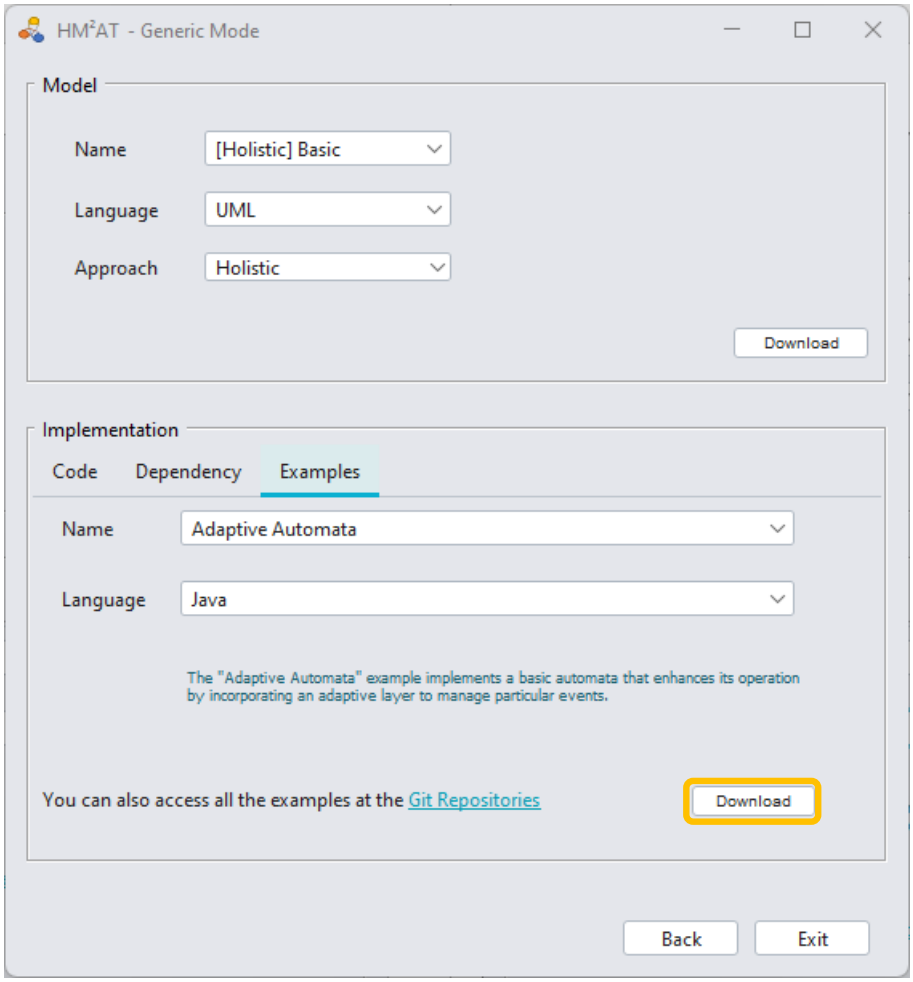


Figura 79 – Descarregar Exemplo de implementação no HM^{2}AT.
O Autor (2022).

7.2.0.2 Modalidade Específica

Como mencionado anteriormente, a modalidade específica acompanha ao desenvolvedor de software no desenvolvimento de uma lógica adaptativa especializada para um dispositivo subjacente concreto.
Os casos de uso principais para esta modalidade são apresentados na Figura 80. Nela observamos que os três passos principais do processo são: (i) a indicação do dispositivo subjacente, (ii) a verificação da adaptabilidade, e (iii) a configuração das entidades que compõem a lógica adaptativa.

A tela de entrada desta modalidade, apresentada na Figura 81, solicita que o usuário inicie a interação com a ferramenta apontando a localização no sistema de arquivos do diretório raiz do dispositivo subjacente. Seguidamente, solicita as informações correspondentes com a linguagem de programação do dispositivo e o tipo de arquivo sendo submetido: biblioteca ou códigos fonte. Após essas informações serem fornecidas,

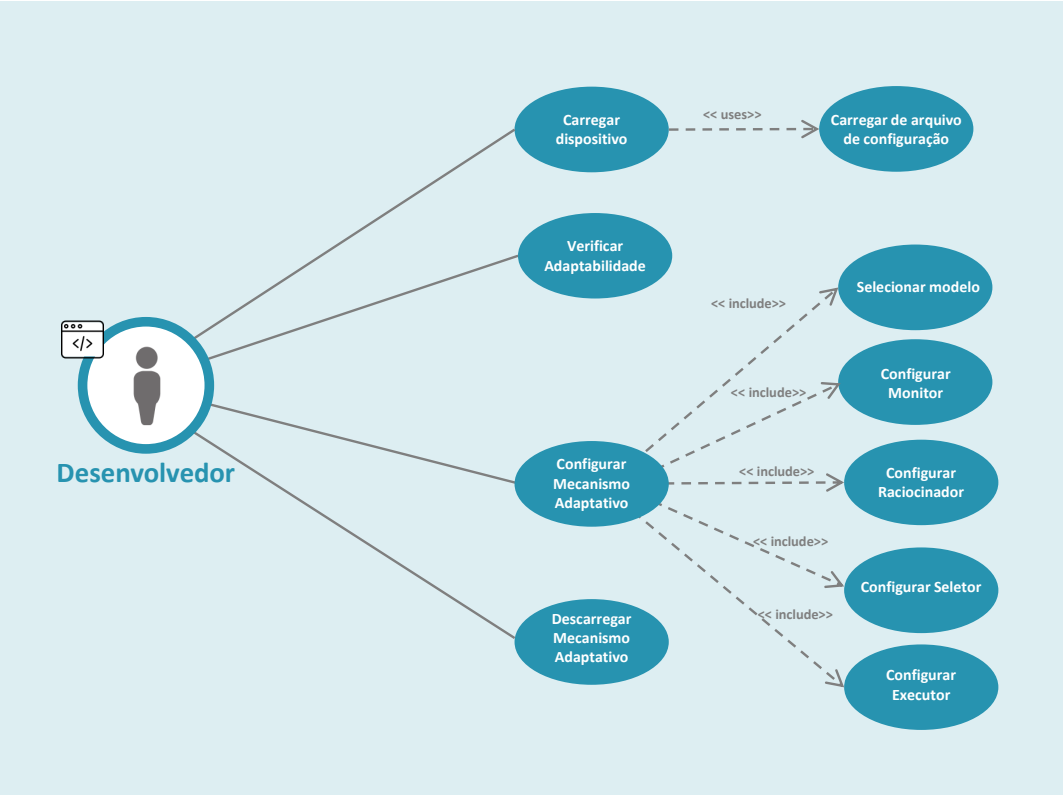


Figura 80 – Casos de uso relacionados à Tela de Modalidade Específica do HM^{2}AT. O Autor (2022).

a ferramenta habilita a opção de "Carregar" dispositivo para verificar as condições de adaptabilidade.

Para decidir se o dispositivo apresenta adaptabilidade, a ferramenta se suporta no instrumento desenvolvido como parte da elaboração do meta-modelo. Nele as condições principais de adaptabilidade são:

- apresentar métodos acessíveis que agem como ganchos de adaptatividade, do tipo sensores, expondo informações do dispositivo para a lógica adaptativa.
- apresentar métodos acessíveis que agem como ganchos de adaptatividade, do tipo atuadores, que realizam mudanças nas informações do dispositivo.
- um método, estratégia ou protocolo de comunicação através do qual o dispositivo e a lógica podem trocar informações.

A tela na Figura 82 é a encarregada de recolher essas informações e verificar se as condições são cumpridas. As informações gerais da tela podem ser preenchidas manualmente ou através de arquivo de configuração (.json) localizado no diretório raiz do dispositivo. As informações da seção correspondente aos ganchos de adaptatividade são carregadas automaticamente após identificada a entidade principal do dispositivo.

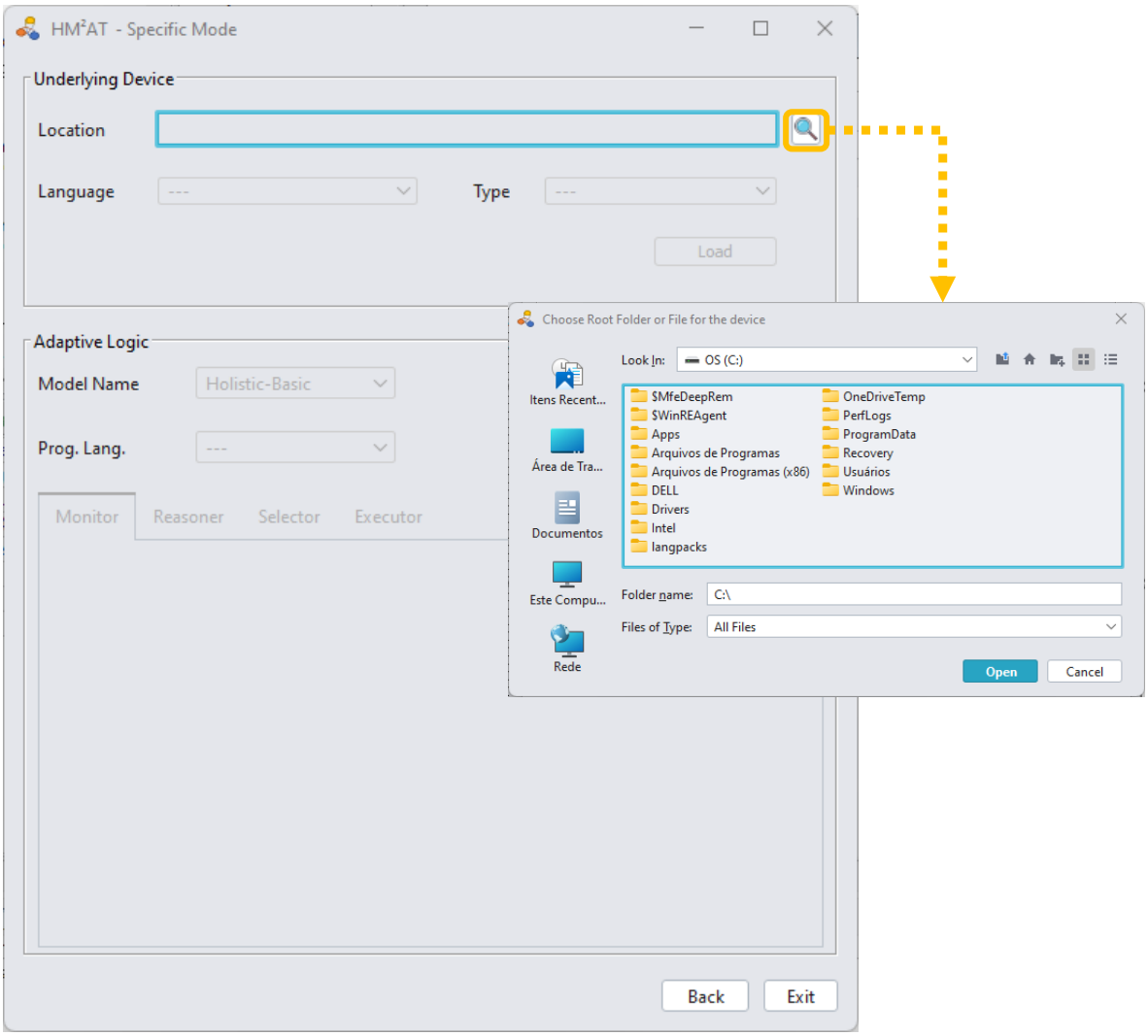


Figura 81 – Procurar Dispositivo no HM²AT.
O Autor (2022).

É importante ressaltar que as verificações realizadas são aplicadas às informações fornecidas pelo desenvolvedor e não de uma análise de adaptabilidade mais profunda do dispositivo. A análise e verificação de adaptabilidade são assuntos de pesquisa que envolvem diferentes tipos de critérios e desenvolvimento de métricas, os trabalhos em Pimentel, Franch e Castro (2011), Lenhard, Geiger e Wirtz (2015) e Merelli, Paoletti e Tesei (2016) apresentam os resultados de algumas dessas pesquisas. Sua incorporação no processo da ferramenta é considerada um trabalho futuro.

Após a verificação das informações a ferramenta informa ao usuário do resultado e caso ele cumpra com os critérios o processo continua. Caso contrário, o usuário deve realizar uma nova escolha de dispositivo para submeter na ferramenta, como apresentado na Figura 83.

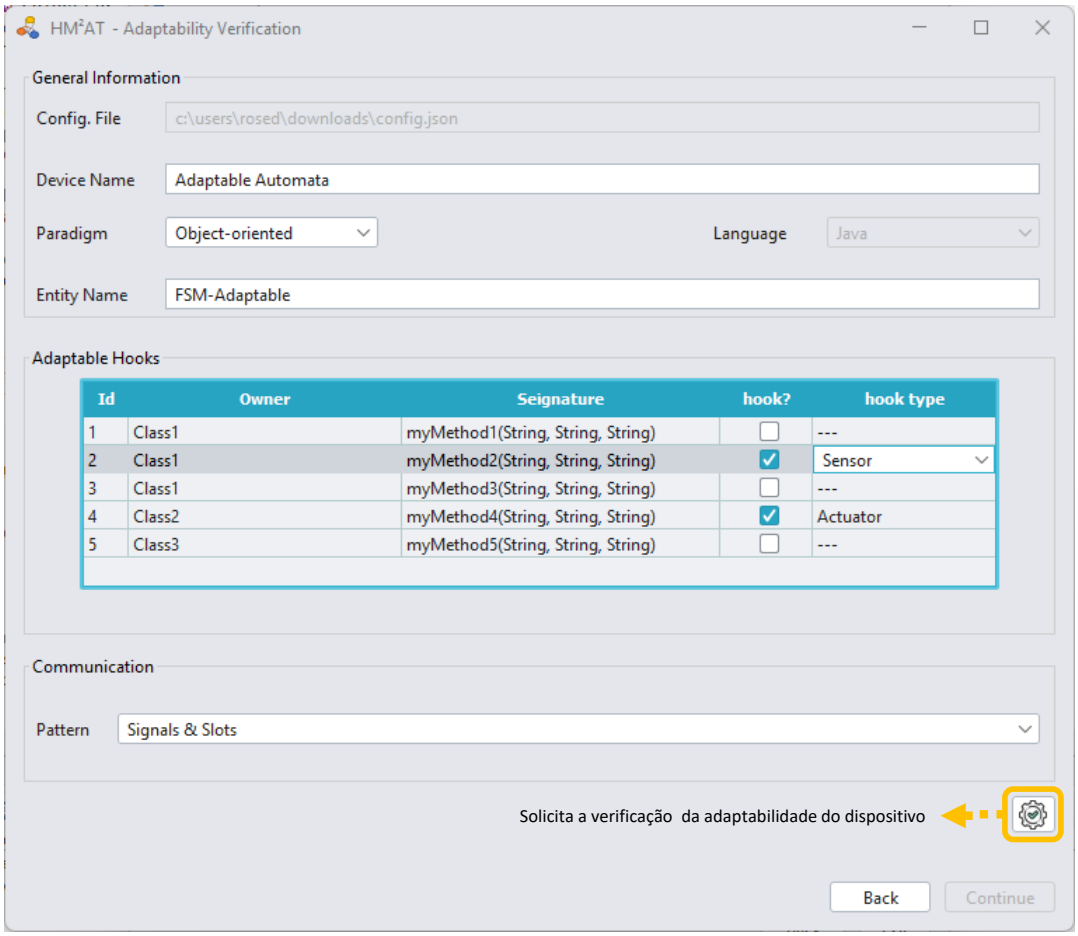


Figura 82 – Verificar adaptabilidade do Dispositivo no HM^{2}AT.
O Autor (2022).

Para projetar uma lógica adaptativa especializada é necessário escolher uma implementação concreta, correspondente a uma linguagem de programação particular, de algum modelo de adaptatividade. Só depois dessa escolha será ativado o painel de configurações para cada um dos elementos da lógica adaptativa como mostra a Figura 84.

A especialização dos elementos da lógica adaptativa é realizada combinando scripts e anotações no código-fonte. Para começar, a implementação do modelo é acompanhada por um script que descreve os passos a serem realizados para especializar o modelo. Depois, o próprio código-fonte inclui anotações (*annotations*) personalizadas que permitem identificar as partes do código que realizam as funções descritas no meta-modelo de adaptatividade. A ferramenta procura essas anotações e apresenta na tela as seções de código correspondentes com cada processo da lógica adaptativa para serem modificadas pelo desenvolvedor, mostra a Figura 85. Se bem essas anotações possuem diferentes nomes dependendo da linguagem de programação, por exemplo: *annotations* em Java e *attributes* em C#, uma grande parte das linguagens de programação oferece recursos para o trabalho com metadados através de tags no código-fonte.

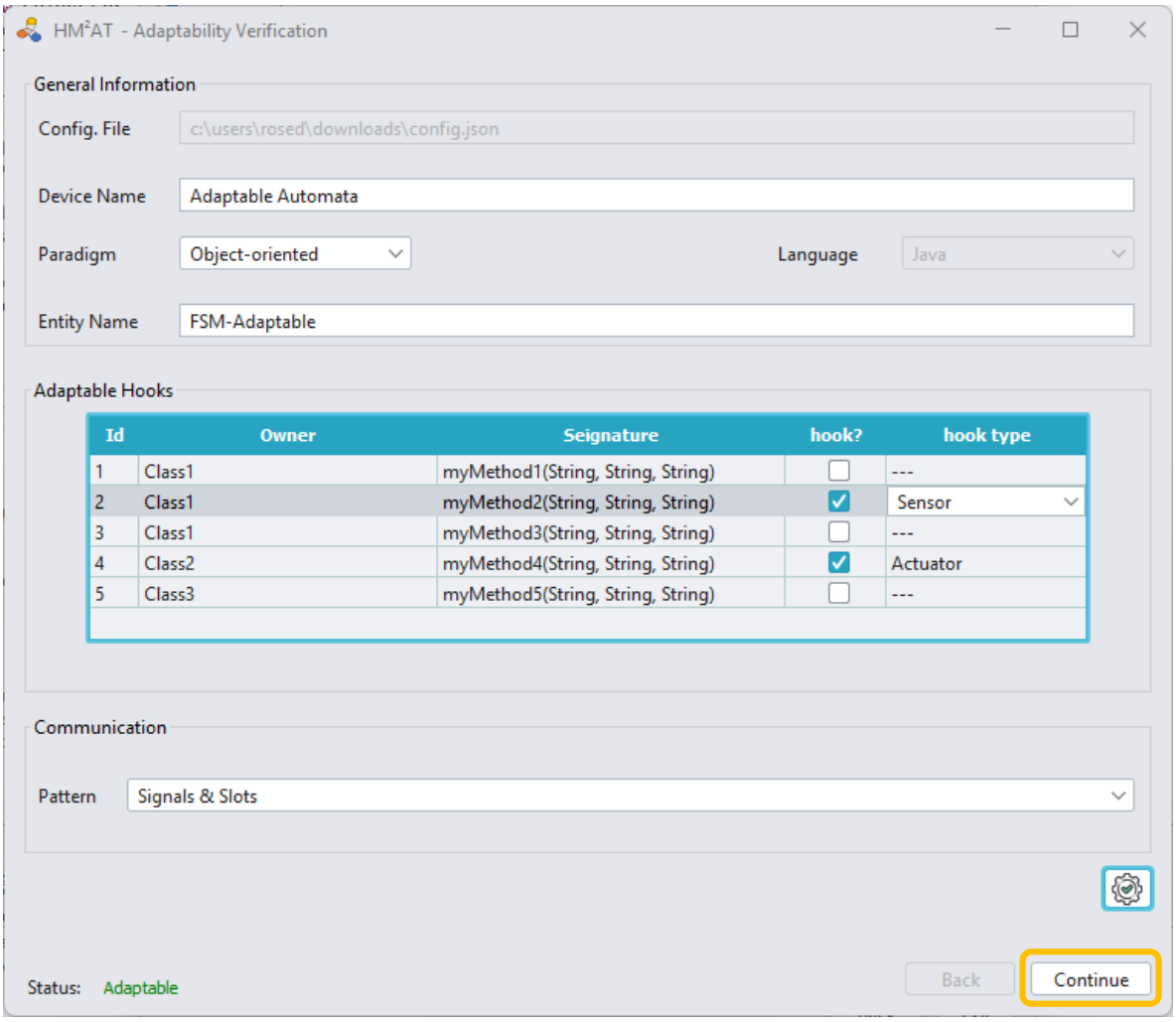


Figura 83 – Dispositivo Verificado no HM^{2}AT.
O Autor (2022).

Finalmente o desenvolvedor pode descarregar o código-fonte correspondente com a lógica adaptativa que especializou. A ferramenta não realiza verificação, compilação ou verificação de código, ela funciona como um editor de texto que ajuda ao desenvolvedor a identificar os processos da lógica adaptativa. Nesse sentido, o desenvolvedor é responsável pela corretude do código incorporado no processo de edição.

As funcionalidades que descrevemos nesta seção são as principais características do Framework HM^{2}AT, e servem como base para novas funcionalidades que podem ser desenvolvidas como trabalhos futuros. A continuação analisaremos os fatores que fazem desta proposta se diferenciar das outras propostas de solução.

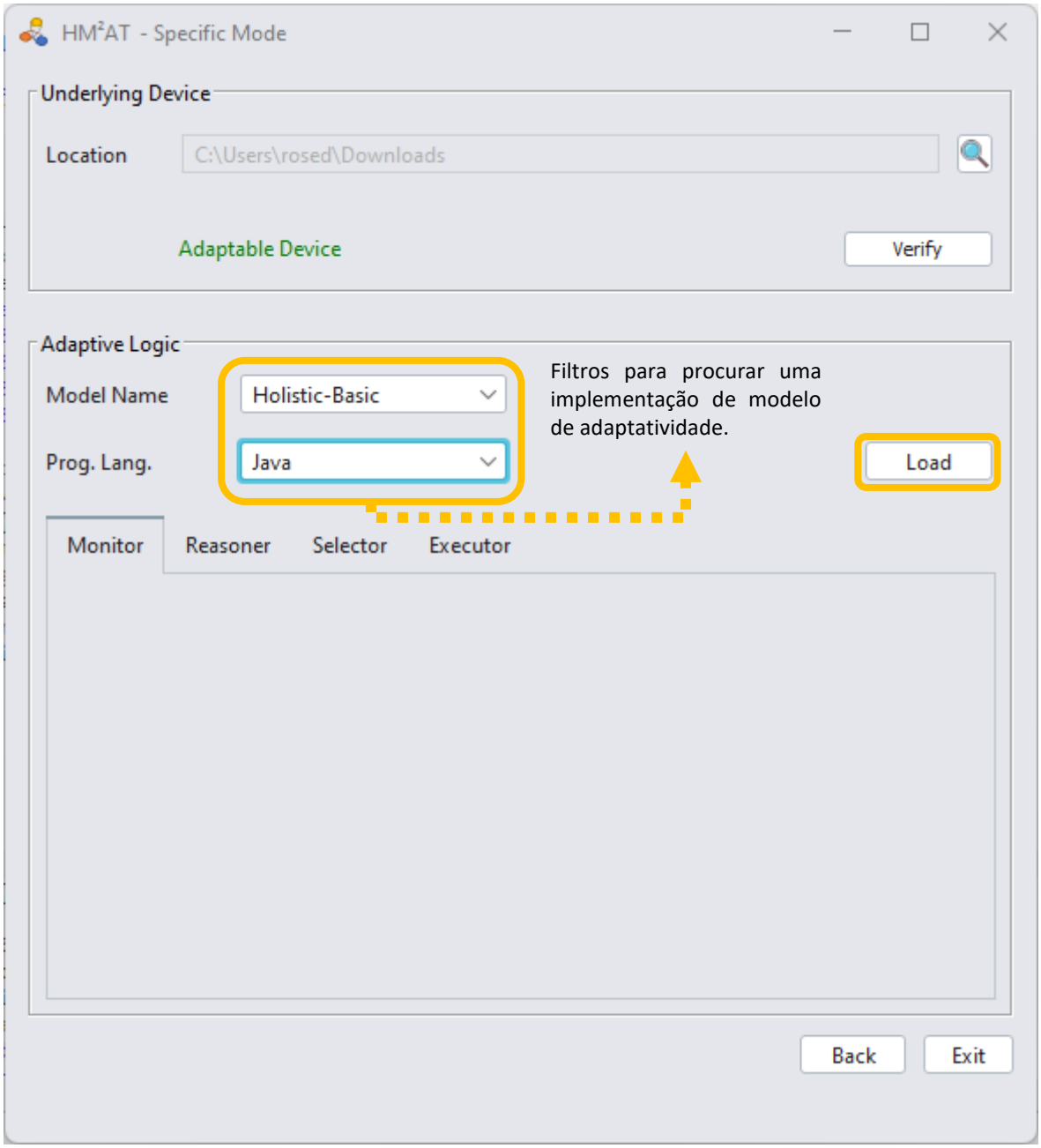


Figura 84 – Carregar Modelo no HM²AT.
O Autor (2022).

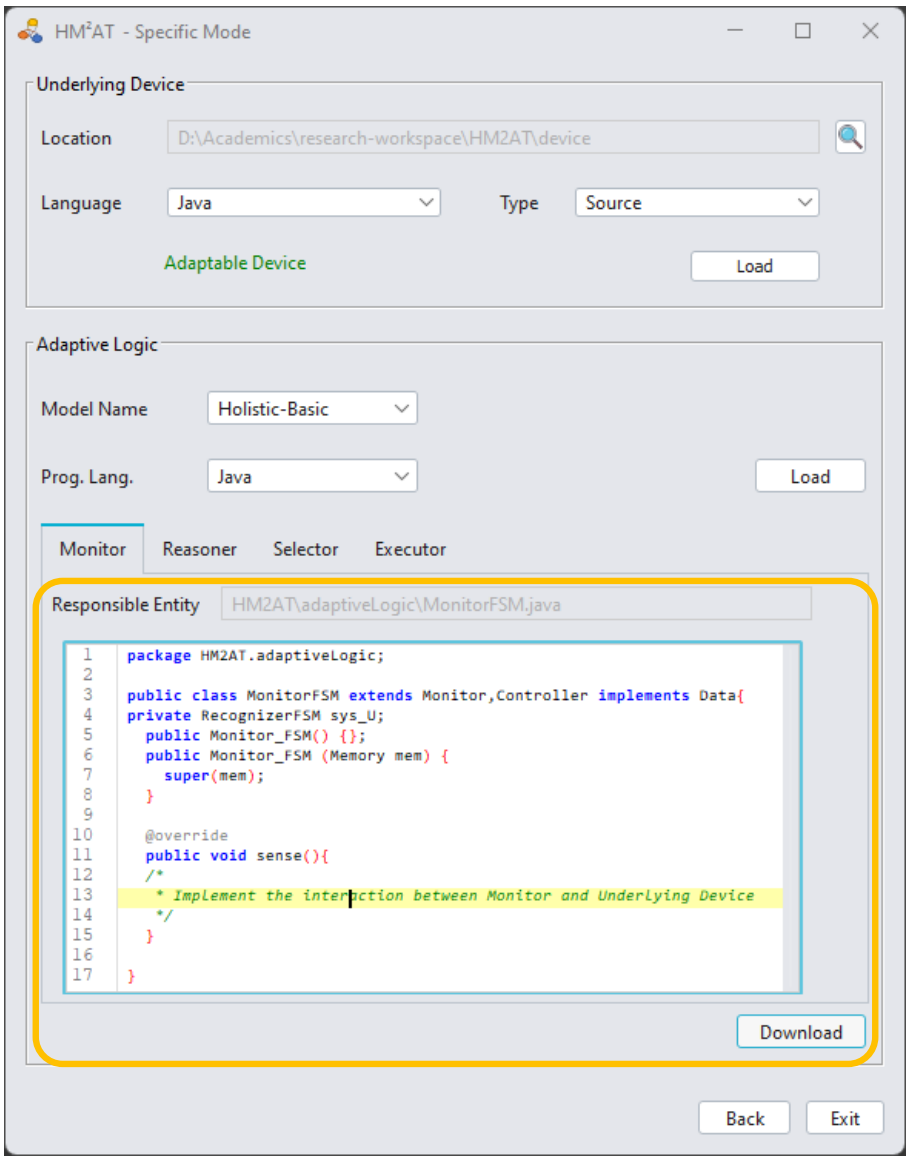


Figura 85 – Descarregar Lógica Adaptativa no HM^{2}AT.
O Autor (2022).

7.3 Comparação com outras ferramentas

Nesta seção apresentamos uma comparação detalhada entre as características principais do framework HM^{2}AT e as características das outras ferramentas que suportam o desenvolvimento de sistemas adaptativos obtidas através do Mapeamento Sistemático na Seção 4.1.2. A Figura 86 apresenta um resumo dessa comparação.

Ferramenta	Última atualização	Fácil de compreender e usar?	Suporta Sistemas Legados como parte dos sistemas adaptativos?	Proposta de aplicação genérica de adaptatividade em software?	Disponível?
Rainbow	2019	Não	Aqueles desenvolvidos com os componentes associados à ferramenta.	Sim	Sim: github
Fractal	2009	Não	Aqueles desenvolvidos com os componentes associados à ferramenta.	Sim	Sim: site oficial do projeto
MOCAS	2008	Sim	É possível através de desenvolvimento de wraps para integrar os sistemas.	Sim	Parcialmente: SourceForge
FESAS	2020	Sim	Em teoria poderia ser realizada a integração com sistemas legados.	Sim	Parcialmente: 1 de 2 componentes está disponível no github
REACT	2022	Não muito	Em teoria poderia ser realizada a integração com sistemas legados.	Possivelmente, porém ele foi planejado especificamente para sistemas de comunicação.	Sim : github
HM^2AT	2023	Sim	Sim	Sim	Sim: github e repositório acadêmico

Figura 86 – Quadro Comparativo entre as ferramentas.
O Autor (2022).

A respeito da vanguarda das tecnologias utilizadas e a vigência da ferramenta podemos observar que embora o framework HM^{2}AT é a proposta mais recente, ele não incorpora recursos tecnológicos de ponta ou inovadores. Ao contrário de REACT que incorpora diversos recursos como Clafer, ZeroC Ice e Docker, o framework HM^{2}AT emprega tecnologias estáveis e muito bem constituídas, especialmente com duas finalidades: (i) fornecer uma curva de aprendizagem menor para os dois tipos de usuários e assim facilitar a compreensão dos conceitos associados, e (ii) incrementar a compatibilidade com recursos tecnológicos legados que podem interagir com o framework. No entanto, o desenvolvimento do framework foi realizado utilizando as versões mais atualizadas (2022-2023) das tecnologias.

A facilidade de uso da ferramenta e aplicação dos conceitos relacionados são outro critério de comparação, neste sentido o framework HM^{2}AT fornece maior suporte que outras propostas porque disponibiliza não só o código-fonte da ferramenta, mas também documentos como a Especificação de Casos de Uso e o Manual Rápido de Usuário acompanhados por exemplos de aplicação e vídeos com demonstrações. Adicionalmente, os conceitos por trás do framework se encontram detalhados na Seção 6.3 de maneira detalhada e usando uma linguagem clara e acessível que pode ser compreendida por público geral além dos especialistas no estudo de comportamento adaptativo.

Quando analisamos o suporte que as ferramentas oferecem para a incorporação de sistemas legados como parte do desenvolvimento de sistemas adaptativos podemos perceber que as outras ferramentas fornecem este suporte parcialmente, no entanto, o framework HM^{2}AT foi projetado com esse suporte como apresenta a Figura 87.

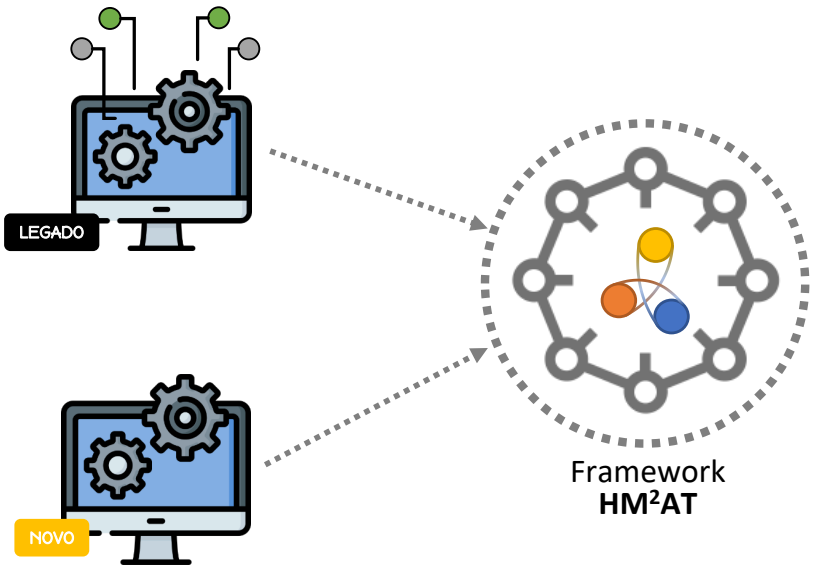


Figura 87 – Quadro Comparativo entre as ferramentas.
O Autor (2022).

Da mesma maneira, ao observar o critério de generalidade para a aplicação de adaptatividade em diferentes domínios, o framework HM^{2}AT se diferencia das outras propostas porque foi desenvolvido com base na separação de interesses entre sistema subjacente e lógica adaptativa. Desta maneira, o framework gerencia e observa os recursos associados exclusivamente com o comportamento adaptativo. Assim, o desenvolvedor de sistemas pode incorporar os recursos criados pelo framework em qualquer domínio com a mesma facilidade. O framework também não depende de recursos ad-hoc para o seu funcionamento ou para a criação dos recursos submetidos no repositório. Adicionalmente, o HM^{2}AT foi desenvolvido sob os princípios do meta-modelo holístico de adaptatividade e, em consequência, é agnóstico de abordagem, podendo gerenciar recursos de diferentes abordagens adaptativas concretas como, por exemplo, MAPE-K, SaS, Sistemas Complexos ou Sistemas Multiagentes.

Finalmente, como descrito previamente, desenvolvemos o framework sob princípio de reprodutibilidade, assim, tanto o código-fonte quanto exemplos de aplicação estão disponíveis nos repositórios de código e acadêmicos que acompanham este trabalho. Adicionalmente, também é disponibilizada a informação necessária para a instalação, compilação e reprodução dos exemplos fornecidos.

8 TESTES E RESULTADOS

Neste capítulo detalhamos as técnicas através das quais realizamos os testes da proposta tanto no nível conceitual quanto tecnológico, assim como especificamos as condições nas quais tais testes foram aplicados. Em seguida, reportamos os resultados obtidos e realizamos a análise correspondente. O tipo de testes aplicados difere em cada caso, pelo que apresentamos primeiro os testes no nível conceitual para depois prosseguir com os testes no nível tecnológico.

8.1 Provas no Nível Conceitual

Nesta seção detalhamos as provas que aplicamos no Meta-modelo Holístico de Adaptatividade para verificar a medida na qual ele serve ao propósito de fornecer um marco conceitual geral e abrangente para diversas perspectivas de estudo do comportamento adaptativo.

8.1.1 *Tipo de prova: Cobertura conceitual*

Consiste na verificação que o meta-modelo é um marco conceitual geral e age como guarda-chuva para diferentes abordagens mais concretas de comportamento adaptativo em diversas áreas do conhecimento.

Como esse objetivo elaboramos um instrumento simples para assistir na realização de análises comparativas com base no meta-modelo holístico de adaptatividade apresentado no Capítulo 6.

O instrumento apresentado na Tabela 2 mostra de maneira muito resumida quais são as condições necessárias para a manifestação de comportamento adaptativo em um sistema. Assim, através dele é possível verificar as correlações, no nível conceitual, entre outras abordagens para adaptatividade e o meta-modelo holístico.

O instrumento é de fácil aplicação, devendo-se preencher cada linha com duas informações: a presença ou ausência da respectiva condição (obrigatória) e, se presente, o identificador do elemento (no sistema em observação) que preenche tal condição (opcional). Com ajuda desse instrumento realizamos uma análise comparativa entre o meta-modelo holístico de adaptatividade e as abordagens descritas na Seção 6.2: Tecnologias Adaptativas, Computação Autonômica, Sistemas autoadaptativos (SaS) e Sistemas Complexos Adaptativos (CAS). A seguir examinamos com maior detalhe. cada uma dessas comparações, ressaltando os pontos de maior diferença entre as outras abordagens e o Meta-modelo Holístico para adaptatividade.

HOLÍSTICO	Modelo A
CONDIÇÕES NECESSÁRIAS	CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	■
Modelos	□
Consciência Individual	□
Contextualização	□
Resposta característica	□
Objetivo do sistema	□
Comportamento regido por regras	□
II. SER ADAPTÁVEL	■
Sensores	□
Atuadores	□
Mecanismo de comunicação/conexão	□
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	■
Autonomia Básica	□
Mecanismos de interação com o ambiente	□
Medida de desempenho	□
Procedimento de decisão	□
Mecanismos homeostáticos	□
Modificadores estruturais	□
Mecanismo regulador	□
Monitoramento	□
Raciocínio	□
Seleção	□
Execução	□
Feedback	□
Memória	□

Tabela 2 – Instrumento para análise comparativa do meta-modelo holístico e outras abordagens.

O Autor (2022).

Em primeiro lugar realizamos a comparação do meta-modelo holístico com as Tecnologias Adaptativas (TA) descritas em Neto (2002) apresentado na Tabela 3. A respeito do sistema subjacente é possível observar que a própria definição dos dispositivos adaptativos, dispositivos desenvolvidos sob a abordagem das Tecnologias Adaptativas, coloca de maneira explícita o modelo do sistema na forma de conjunto de configurações e regras de transição. Dessa maneira, a interação entre o sistema subjacente e o ambiente fica especificada através do alfabeto de entradas e o alfabeto de saídas. O objetivo do sistema subjacente está representado ou codificado de maneira implícita nas próprias regras que regem seu comportamento. No modelo dos dispositivos adaptativos os sensores são representados através das funcionalidades que realizam a inspeção das regras em busca de ações adaptativas em dois momentos diferentes: antes (*before action*) e depois (*after action*) da execução da regra do dispositivo subjacente. Da mesma forma, os atuadores no modelo são os mecanismos de execução das ações adaptativas identificadas, seja por

aplicação direta, por chamada a função ou alguma outra estratégia de execução.

HOLÍSTICO		Tecnologias Adaptativas [Neto2002]
CONDIÇÕES NECESSÁRIAS		CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	■	*Dispositivo Subjacente : Dispositivo dirigido por regras*
Modelos		
Consciência Individual	□	Conjunto de configurações, configuração inicial e configurações finais, memoria para processos intermediários
Contextualização	□	Eventos que representam um estímulo de entrada
Resposta característica		
Objetivo do sistema	□	Executar as regras associadas aos estímulos de entrada válidos percebidos
Comportamento regido por regras	□	Conjunto de regras particulares do dispositivo
II. SER ADAPTÁVEL	■	
Sensores	□	Inspeção de ações adaptativas antes (*before* (B•)) e depois (*after* (•A)) da ativação de uma regra do dispositivo subjacente.
Atuadores	□	Execução das ações adaptativas associadas às regras compatíveis segundo o caso (aplicação direta, chamada de função, etc.).
Mecanismo de comunicação/conexão	□	Mecanismo de incorporação das ações adaptativas às regras de transição compatíveis com o contexto corrente (i.e. Mapeamento, Merge)
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	■	Mecanismo Adaptativo / Camada Adaptativa
Autonomia Básica		
Mecanismos de interação com o ambiente	□	Mecanismo para detecção de entradas
Medida de desempenho	□	Não existe. As propostas não realizam seleção de regras adaptativas pois, segundo o caso, aplicam 0 (sem escolha), 1 (sem escolha) ou todas as possíveis (sem escolha).
Procedimento de decisão	□	Para cada entrada e estado corrente é procurado um conjunto de possíveis regras a serem aplicadas.
Mecanismos homeostáticos	□	Verificações de coerência, i.e. eliminação de regra corrente
Modificadores estruturais	□	Ações adaptativas elementares : (?),(-),(+)
Mecanismo regulador		
Monitoramento	□	Dado um estímulo é verificado se existe alguma regra adaptativa associada à regra do dispositivo subjacente compatível com a situação corrente do dispositivo.
Raciocínio	□	Coleção das regras adaptativas compatíveis
Seleção	□	Não existe mecanismo de seleção nesta proposta.
Execução	□	Aplicação da regra adaptativa
Feedback	□	Nova configuração do dispositivo subjacente
Memória	□	Sequência de configurações do dispositivo desde a configuração inicial até a configuração corrente e as experiências prévias (configurações intermediárias e mudanças no conjunto de regras)

Tabela 3 – Comparação entre o meta-modelo holístico e a abordagem das Tecnologias Adaptativas.

O Autor (2022).

O mecanismo de comunicação entre o dispositivo subjacente e a camada adaptativa está representado pelo mecanismo de incorporação de ações adaptativas nas regras de transição do dispositivo subjacente, tal incorporação pode ser realizada de várias maneiras, por exemplo: manipulação direta das regras, ou mapeamento entre o conjunto de regras e um conjunto auxiliar que especifique as ações associadas a cada regra. No que concerne à autonomia básica, os dispositivos adaptativos cumprem com todos os pontos da tabela, porém uma observação interessante é a ausência de medida de desempenho, justificada no caso pelo tratamento de não-determinismo no próprio sistema subjacente.

Finalmente, o funcionamento da camada adaptativa realiza um monitoramento das

ações adaptativas associadas para cada regra de transição dada a configuração atual do dispositivo subjacente e uma entrada. Se existem regras possíveis para essas condições, elas são coletadas e tanto a regra subjacente quanto as ações adaptativas associadas são aplicadas. Como consequência dessa aplicação, o dispositivo muda sua configuração e isso fecha o ciclo de feedback. A memória do modelo está representada tanto pela sequência de configurações do dispositivo assim como as informações sobre experiências prévias, por exemplo, mudanças no conjunto de regras.

Outra comparação foi realizada com a abordagem de Computação Autonômica (AC) proposta pela IBM em IBM Corporation (2005), Horn (2001) e Kephart e Chess (2003) e apresentada na Tabela 4. Na AC o sistema subjacente, nomeado recurso gerenciado, possui tanto conhecimento de si mesmo quanto conhecimento do ambiente no qual se desenvolve, por tanto, podemos assumir a existência de modelos que forneçam essas informações. Da mesma maneira, é identificado que o recurso possui um comportamento padrão alinhado com um objetivo particular e o seu comportamento está especificado por mecanismos dirigidos por metas, objetivos ou políticas expressas em sentenças com formato evento-condição-ação ou ECA.

A respeito de adaptabilidade, em Kephart e Chess (2003) e Horn (2001) é mencionado de maneira explícita que os recursos gerenciados devem ser adaptáveis, porém sem maior detalhe sobre as condições que isso impõe. No entanto, AC cria o conceito de *touchpoint* para funcionar como um ponto de interação entre o recurso gerenciado e o gerenciador autonômico. Os *touchpoints* disponibilizam sensores, efetores e interfaces de gerenciamento dos recursos que os gerenciadores autonômicos podem utilizar.

No que respeita à autonomia básica, a AC se baseia em políticas, medidas de eficiência e otimização, restrições e funções de utilidade como métricas para calcular o desempenho do recurso gerenciado e analisar a necessidade de comportamentos adaptativos dadas as condições correntes. Os procedimentos de decisão que auxiliam a resolução de conflitos e na seleção das estratégias a serem executadas são especificados em forma de políticas com base no cumprimento de metas ou objetivos. De maneira particular, a AC especifica o uso de técnicas especializadas, como *forecasting* e até a intervenção do gerenciador manual como possíveis mecanismos homeostáticos que permitem equilibrar as consequências das ações do gerenciador autonômico.

A abordagem de AC estabelece o ciclo MAPE-K onde os processos de monitoramento, analise (raciocínio e seleção), planejamento e execução implementam os mecanismos necessários para o comportamento adaptativo, e a interação e atualização das informações do sistema na Base de Conhecimento fecham o ciclo.

É interessante observar que a proposta da AC considera dois pontos interessantes: (a) o requerimento de sensores e efetores no nível do gerenciador autonômico (de *touch-points*) para permitir a construção de hierarquias de adaptatividade, e (b) a possibilidade de implementar parcialmente o ciclo MAPE-K, isto é, não é necessária a realização de todas as fases, pois eventualmente a hierarquia de adaptatividade pode completar o ciclo e garantir o comportamento adaptativo.

HOLÍSTICO		**Computação autonômica[IBM2005][Horn2001][Kephsrt2003]**
CONDIÇÕES NECESSÁRIAS		CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	■	*Recurso gerenciado*
Modelos		
Consciência Individual	□	Conhecimento de si [Horn2001]
Contextualização	□	Conhecimento do ambiente [Horn2001]
Resposta característica		
Objetivo do sistema	□	Comportamento padrão do recurso
Comportamento regido por regras	□	Mecanismos dirigido por metas ou objetivos e políticas [Kephart2003] (ECA)
II. SER ADAPTÁVEL	■	Requisito mencionado em [Horn2001][Kephart2003]
Sensores	□	Touchpoint - sensor
Atuadores	□	Touchpoint - effector
Mecanismo de comunicação/conexão	□	Touchpoint- Interface de gerenciamento
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	■	Gerenciador Autonômico. Também expõe Sensors e Effector e pode ser agregado
Autonomia Básica		
Mecanismos de interação com o ambiente	□	Próprio do recurso gerenciado
Medida de desempenho	□	Políticas e medidas de eficiência e optimização, restrições, regras ou funcões de utilidade[Kephart2003]
Procedimento de decisão	□	Políticas(metas ou objetivos)
Mecanismos homeostáticos	□	Técnicas especializadas (forecasting) + intervenção do gerenciador manual
Modificadores estruturais	□	Conjunto de operações no Effector. Mecanismos e políticas de negociação [Kephart2003].
Mecanismo regulador		Ciclo Autonômico + Gerenciador Manual. Não precisa implementar o ciclo completo
Monitoramento	□	Monitor
Raciocínio	□	Analise
Seleção	□	Analise
Execução	□	Planejamento e Execução
Feedback	□	Através da Base de Conhecimento
Memória	□	Base de Conhecimento

Tabela 4 – Comparação entre o meta-modelo holístico e a Computação Autonômica. O Autor (2022).

Duas comparações foram realizadas entre o meta-modelo holístico e abordagens de *Self-adaptive Systems*(SaS) para verificar sua concordância. Num primeiro momento realizou-se a comparação com o framework MUSIC apresentado em Rouvoy *et al.* (2009),Geihs (2013),Kakousis *et al.* (2010) e Floch *et al.* (2013) apresentada na Tabela 5. A proposta de MUSIC é assistir o desenvolvimento de sistemas SaS para aplicativos móveis que interagem em ambientes inteligentes e precisam adaptar a sua função às

condições do ambiente. Assim, para MUSIC o sistema subjacente são aplicativos na forma de componentes de software, que fornecem o seu modelo de domínio e de variabilidade (as pautas para gerir as mudanças) e o modelo de contexto que faz parte essencial do foco do trabalho. O objetivo dos aplicativos é intrínseco a cada um deles e a especificação está detalhada nos planos de configuração dos componentes e no plano de composição do aplicativo, assim, a composição e comunicação de vários componentes de software criam o aplicativo.

HOLÍSTICO		**MUSIC (Self-Adaptive) [Rouvoy2009,Geihs2013,Kakousis2010, Floch2012]**
CONDIÇÕES NECESSÁRIAS		CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	■	*Componentes de software (aplicação)*
Modelos		
Consciência Individual	□	modelo de variabilidade, modelo de domínio
Contextualização	□	Modelos de contexto
Resposta característica		
Objetivo do sistema	□	Intrínseco à aplicação
Comportamento regido por regras	□	Plano de configuração dos componentes (Role, Port, Type), plano de composição
II. SER ADAPTÁVEL	■	componentes autocontidos, aplicações adaptáveis[Floch2012]
Sensores	□	Sensores de contexto e raciocinadores
Atuadores	□	Não menciona
Mecanismo de comunicação/conexão	□	portas(interna) e protocolos (externa)
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	■	Controlador de Adaptações
Autonomia Básica		
Mecanismos de interação com o ambiente	□	Sensores de contexto, listeners
Medida de desempenho	□	Funções de utilidade
Procedimento de decisão	□	Maximização de utilidade para as diferentes configurações possíveis
Mecanismos homeostáticos	□	Restrições, acordos de serviço e negocioações
Modificadores estruturais	□	Serviços do sistema e Kernel [Hallsteinsen2012]
Mecanismo regulador		
Monitoramento	□	Gerenciador de contexto
Raciocínio	□	Processo de Planejamento - Raciocinador de Adaptações
Seleção	□	Processo de Planejamento - Planejador de Configurações
Execução	□	Processo de Planejamento - Executor de Configurações
Feedback	□	Asumido pela natureza dos aplicativos
Memória	□	Repositório de Planos, *QoS-aware model*, Modelo de Adaptações

Tabela 5 – Comparação entre o meta-modelo holístico e MUSIC(SaS).
O Autor (2022).

A respeito da natureza dos componentes é dito que eles são autocontidos e em particular criam aplicativos adaptáveis (FLOCH *et al.*, 2013) apresentando sensores de contexto e raciocinadores, e propriedades que permitem a mudança de configuração dos componentes e a arquitetura e funcionalidade do aplicativo em geral através do modelo de variabilidade e um mecanismo de comunicação entre componentes baseado em portas e protocolos.

A autonomia básica é estabelecida pelas funções de utilidade como medida de

desempenho e a maximização dos seus parâmetros também guia a tomada de decisão sobre as estratégias adaptativas. As restrições, os acordos de serviço e as estratégias de negociação são os possíveis mecanismos homeostáticos considerados na proposta.

Já o mecanismo adaptativo é implementado pelo Gerenciador de contexto que monitora o ambiente de execução do aplicativo e transmite informações sobre a mudança no contexto para o Raciocinador de Adaptações que dadas as condições correntes procura a melhor variação para o sistema segundo o indicado no modelo de variabilidade e os recursos disponíveis no repositório de planos. Logo o Planejador de Configurações realiza o cálculo de utilidade para variante de configuração do sistema e escolhe a de maior valor. O Executor de Configurações realiza as mudanças especificadas pelo plano de reconfiguração e atualiza o modelo. No caso de MUSIC tanto o repositório de planos quanto o modelo de qualidade de serviço (QoS) e o modelo de adaptações atuam como memória e base de conhecimento e mata data para os processos do comportamento adaptativo.

A segunda comparação com um SaS foi realizada com o framework Rainbow descrito em Garlan *et al.* (2004) e Garlan, Schmerl e Cheng (2009) e apresentada na Tabela 6. Para Rainbow o sistema subjacente é denominado sistema alvo e fornece informações sobre ele mesmo por um modelo arquitetônico, sobre o ambiente de execução no qual se encontra inserido, e sobre o seu objetivo através das preocupações do sistema. O comportamento do sistema alvo está detalhado no próprio modelo do sistema através de regras, restrições e propriedades do sistema.

A adaptabilidade do sistema alvo está suportada tanto pelos *probes* que monitoram o contexto e os descobridores de recursos que permitem a atualização dos recursos disponíveis para o sistema quanto pelos efetores que realizam as modificações solicitadas. Na arquitetura de Rainbow existe uma camada dedicada para a gestão de tradução entre os níveis de abstração dos modelos e as ações a serem realizadas no sistema alvo, i.e., a API do sistema. Assim, existem também interfaces de conexão e protocolos de comunicação entre os componentes do sistema detalhadas como parte de Rainbow que podem ser reutilizadas.

A autonomia está representada pela medida de desempenho implementada nos *probes*, *gauges* e no avaliador de restrições, e pelo procedimento de decisão especificado no motor de adaptações e enriquecido pelas funções de utilidade composta para gerenciar a resolução de conflitos. É mencionado também que embora existam possibilidades para a implementação de mecanismos homeostáticos através das próprias estratégias de adaptação, eles não são realizados de maneira muito intuitiva devida a que o framework está orientado à ativação de comportamento adaptativo como consequência da infração de restrições.
No que respeito ao mecanismo adaptativo, Rainbow fornece o monitoramento das condições

HOLÍSTICO	Rainbow (SaS) [Garlan2004][Garlan2009]
CONDIÇÕES NECESSÁRIAS	CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	*Sistema Alvo*
Modelos	
Consciência Individual	Modelo operacional do sistema através do modelo arquitetónico
Contextualização	Modelo do ambiente
Resposta característica	
Objetivo do sistema	Preocupação do sistema
Comportamento regido por regras	Regras, restrições e propriedades do sistema, especificadas no modelo do sistema alvo
II. SER ADAPTÁVEL	
Sensores	Probes e descobridores de recursos, API do sistema alvo
Atuadores	Efetores
Mecanismo de comunicação/conexão	Infrastrutura de tradução, interfaces de conexão e protocolos de comunicação entre componentes
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	
Autonomia Básica	
Mecanismos de interação com o ambiente	Probes, propriedades do sistema
Medida de desempenho	Inclusa nos probes, nos gauges e no avaliador de restrições
Procedimento de decisão	Engine de adaptações e função de utilidade composta
Mecanismos homeostáticos	Possível através de estrategias de adaptação porém não muito intuitivo
Modificadores estruturais	Operadores e efetores do sistema alvo
Mecanismo regulador	
Monitoramento	Probes e gauges
Raciocínio	Avaliador de restrições e Engine de adaptações (através de estratégias de adaptações)
Seleção	Engine de adaptações: através de restrições, estilos de adaptações, funções de utilidade, dimensões de qualidade e atributos custo-beneficio
Execução	Executor de adaptações
Feedback	Através dos modelos do sistema e do ambiente atualizados pelo Gerenciador de Modelos
Memória	Base de Conhecimento de adaptações e modelo do sistema alvo

Tabela 6 – Comparação entre o meta-modelo holístico e Rainbow(SaS).
O Autor (2022).

do sistema via *probes* e *gauges*, as informações colhidas são enviadas para o Avaliador de Restrições, se existe alguma infração este ativa o Motor de Adaptações que consulta as estratégias de adaptação para obter as possíveis respostas para resolver a situação. As diversas opções de resposta são avaliadas com funções de utilidade, preferências, acordos de qualidade e atributos de custo-benefício, para selecionar a estratégia a ser aplicada pelo Executor de Adaptações. O ciclo é fechado com a atualização dos modelos do sistema realizados pelo Gerenciador de Modelos. Para Rainbow tanto a base de conhecimento dedicada quanto o modelo do sistema alvo são elementos de metainformações e memória.

Por último realizamos uma comparação com o Modelo baseado em Agentes, proposta mais utilizada para o projeto de Sistemas Adaptativos Complexos (CAS) apresentada na Tabela 7.

HOLÍSTICO		Modelo baseado em Agentes (CAS) [Holland1992, Holland1992Daedalus]
CONDIÇÕES NECESSÁRIAS		CONDIÇÕES NECESSÁRIAS
I. SER UM SISTEMA *	■	*O sistema está composto por elementos semiautônomos organizados de maneira decentralizada que interagem para a realização de objetivos comuns.*
Modelos		
Consciência Individual	□	i) Modelo interno de cada agente e suas funcionalidades. ii) Modelo organizacional do sistema geral gerenciando regras de interação e objetivos entre os seus elementos.
Contextualização	□	Os agentes possuem mecanismos que permitem interações independentes com o ambiente e com outros agentes do sistema
Resposta característica		
Objetivo do sistema	□	O objetivo geral do sistema é estabelecido como uma série de objetivos (*goals*) que dirigem o comportamento dos agentes
Comportamento regido por regras	□	Comportamento do sistema: regras simples (agentes) + interação (agentes e ambiente) + realização dos objetivos gerais.
II. SER ADAPTÁVEL	■	Cada agente que conforma o sistema é uma entidade adaptativa
Sensores	□	Cada agente possui sensores que permitem a coleta de informações
Atuadores	□	Cada agente possui mecanismos que permitem que realize ações como resposta a estímulos
Mecanismo de comunicação/conexão	□	Integrado no próprio agente, não existe separação entre comportamento padrão e comportamento adaptativo. O comportamento está sempre em transformação.
III. CONEXÃO COM UMA LÓGICA ADAPTATIVA	■	
Autonomia Básica		
Mecanismos de interação com o ambiente	□	Recursos incorporados em cada agente (sensores de temperatura, feromônios, etc)
Medida de desempenho	□	Podem ser especificadas como políticas, medidas de aptidão, utilidade ou preferências associadas a cada agente e/ou ao sistema em geral.
Procedimento de decisão	□	Cada agente é autônomo e realiza seu própria tomada de decisão. Como consequência disso emergem no sistema propriedades gerais.
Mecanismos homeostáticos	□	Possíveis através de estratégias de resolução de conflitos e negociações
Modificadores estruturais	□	Denominadas building blocks, são unidades de construção que podem ser reutilizadas no sistema
Mecanismo regulador		
Monitoramento	□	Cada agente está equipado com recursos para obter informações de interesse do seu ambiente e da interação com outros agentes.
Raciocínio	□	A lógica do comportamento de cada agente está relacionada com os seguintes elementos: (a) às regras básicas (simples) que modelam o seu comportamento, (b) o comportamento dos outros agentes com os quais interage, (c) das condições do ambiente no qual está imerso, e (d) os objetivos que precisa atingir.
Seleção	□	Em diferentes políticas de níveis organizacionais é possível incluir funções de utilidade, e avaliação
Execução	□	O próprio agente é o encarregado de executar os comportamentos adaptativos necessários
Feedback	□	A interação entre agente e ambiente gera um feedback que o próprio agente é capaz de perceber. Além disso podem existir outros mecanismos de comunicação (*broadcasting*) que permitem que esse feedback se propague para o sistema geral.
Memória	□	O sistema possui uma memória ou história: cada agente integra sua própria história e evolução, assim a história do sistema é também uma composição da história contada pelos agentes.

Tabela 7 – Comparação entre o meta-modelo holístico e o Modelo baseado em agentes dos CAS.

O Autor (2022).

O trabalho com CAS exige uma mudança no design e organização dos sistemas, pois eles são compostos por muitos elementos heterogêneos e semi-autônomos denominados agentes que interagem tanto com o ambiente quanto com outros agentes para criar o comportamento geral do sistema. Assim, um CAS não descreve um sistema subjacente desacoplado, nem possui uma lógica padrão centralizada que governa o sistema. No entanto, os agentes podem ser entendidos como pequenos sistemas que existem e co-evoluem no domínio do CAS. Assim, eles apresentam modelos internos que ajudam na representação da identidade dos agentes e modelos organizacionais que regulam as interações entre eles e o ambiente.

Da mesma maneira, os agentes estão providos de mecanismos que lhes permitem reunir informações do contexto de execução em que se encontram. O comportamento geral do CAS é resultado da especificação de regras simples que guiam o comportamento dos agentes, as interações entre agentes e com o ambiente, e dos objetivos comuns estabelecidos para o sistema em geral. A respeito da adaptabilidade subjacente podemos mencionar que os agentes são entidades que possuem não só adaptabilidade, mas também adaptatividade, portanto, conseguem incorporar mudanças e evoluir segundo as necessidades. A lógica adaptativa nos CAS está presente em dois níveis: (i) em cada agente, como mencionamos previamente, e (ii) no sistema como uma agregação e propagação do comportamento dos agentes. É interessante notar que mesmo no caso de agentes com capacidade adaptativa é necessária a inclusão de certos elementos que permitam coordenar alguns esforços. Um desses elementos são as medidas de desempenho que podem ser especificadas como pontos de atenção para os agentes de maneira que podem reagir a ameças ou identificar oportunidades. Outro elemento de coordenação são as estratégias de solução de conflito e negociação, elas agem como mecanismos homeostáticos que cuidam de manter o comportamento do sistema dentro do "controle". Os elementos do mecanismo regulador estão contidos maioritariamente nos próprios agentes, complementados em alguns casos por políticas gerais.

Finalmente, todo CAS apresenta uma memória ou histórico que existe tanto no nível de agentes através de sua própria evolução quanto no nível geral do sistema. Essas experiências prévias são corresponsáveis pelo comportamento no presente.

8.1.2 *Tipo de prova: Prova de conceito*

Para verificar a aplicabilidade da proposta desenvolvemos um exemplo, a modo de prova de conceito, que embora simples permite ilustrar as vantagens de uma instância do meta-meta-modelo holístico de adaptatividade apresentado na seção anterior.

Com a finalidade de fornecer reprodutibilidade, o código-fonte de cada projeto que faz parte da prova de conceito está disponível em https://github.com/stars/rosedth/lists/proof-of-concept, repositório de implementações que acompanha este trabalho. As implementações da prova de conceito foram realizadas utilizando a linguagem de programação Java, com o JDK 17.0.1 LTS, com Maven 3.8.0 e GitHub para o gerenciamento da distribuição de código e a biblioteca Jackson para o compartilhamento de dados. Utilizamos o IDE Eclipse 2021-09 como plataforma para o desenvolvimento em um computador com sistema operacional Windows 11 nativo, processador Intel(R) Core(TM) i7 de 11a. Geração com 2.80GHz de velocidade e RAM de 16GB.

O exemplo desenvolvido representa o comportamento de um Autômato Adaptativo (AA) e segue as diretrizes apontadas em Neto (2002). O AA tem como dispositivo subjacente um autômato finito que reconhece uma linguagem particular e segue as regras estabelecidas por esse autômato até a aplicação de regras adaptativas em algum momento futuro.

Seguindo o princípio de reutilização utilizamos como base para a implementação do autômato finito um software básico desenvolvido por Mahmoud Ben Hassine para criar máquinas de estado finitas na linguagem Java chamado Easy State[1] com o qual implementamos a funcionalidade padrão do nosso autômato, representada pela Figura 88.

Como podemos observar, o autômato reconhece a linguagem $L = b(a^*b)^*(b|c)$ começando no estado inicial “s”, o seu alfabeto de entradas é $\Sigma = a, b, c$, e possui apenas um estado final “r”. A implementação do nosso reconhecedor é simples e está contida no projeto denominado “normal-fsm” do repositório.

O projeto implementa as classes AEvent, BEvent e CEvent para representar os eventos gerados pelas entradas a, b e c, e a classe RecognizerFSM que, de fato, cria o autômato FSMNormal. Um exemplo da implementação de AEvent está no Código8.1, os casos de BEvent e CEvent são correspondentes. Um extrato do código utilizado para a implementação do RecognizerFSM está no Código 8.2 onde podemos observar que o autômato identifica a entrada recebida e a associa com o evento correspondente para logo verificar a existência de uma transição desde o estado atual com a entrada corrente e a chegada no estado de aceitação “r”.

[1] Disponível em https://github.com/j-easy/easy-states

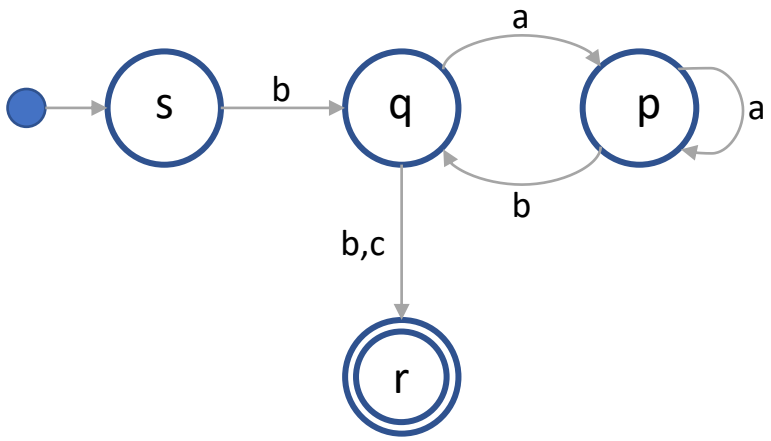


Figura 88 – Configuração Padrão do Autômato ANormal
O Autor (2022)

Da mesma maneira, podemos observar que o autômato distingue duas situações excepcionais: (1) entrada não suportada: a leitura de uma entrada fora do alfabeto, e (2) entrada inválida: a leitura de uma entrada que pertence ao alfabeto, porém que não apresenta nenhuma transição a partir do estado atual.

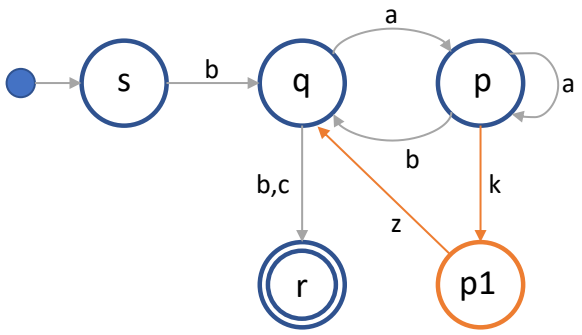
```
1  public class AEvent extends AbstractEvent {
2
3      public AEvent() {
4          super("AEvent");
5      }
6
7      protected AEvent(String name) {
8          super(name);
9      }
10
11 }
```

Código 8.1 – Código de implementação de AEvent

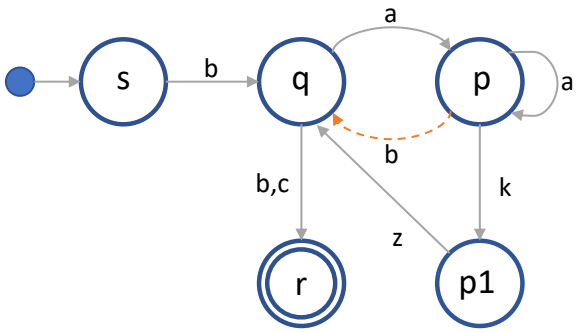
```
1  public class RecognizerFSM {
2      private FiniteStateMachine fsm;
3      private String path;
4
5      public RecognizerFSM() {
6          /*
7           * Define FSM states
8           */
9          State s_State = new State("S");
10         State q_State = new State("Q");
11         State p_State = new State("P");
12         State r_State = new State("R");
```

```
13
14          Set<State> states = new HashSet<>();
15          states.add(s_State);
16          states.add(p_State);
17          states.add(q_State);
18          states.add(r_State);
19
20          /*
21           * Define FSM transitions
22           */
23          Transition s_q = new TransitionBuilder()
24                  .name("S to Q with b")
25                  .sourceState(s_State)
26                  .eventType(BEvent.class)
27                  .targetState(q_State)
28                  .build();
29
30          // Other transitions here.
31
32          /*
33           * Build FSM instance
34           */
35          fsm = new FiniteStateMachineBuilder(states, s_State)
36                  .registerTransition(s_q)
37                  .registerTransition(q_p)
38                  .registerTransition(p_p)
39                  .registerTransition(p_q)
40                  .registerTransition(q_r_b)
41                  .registerTransition(q_r_c)
42                  .registerFinalState(r_State)
43                  .build();
44          path="";
45      }
46
47      public void processInput(String input) throws FiniteStateMachineException {
48          AbstractEvent event=null;
49          switch(input.trim().toUpperCase()) {
50            case "A": event=new AEvent(); break;
51            case "B": event=new BEvent(); break;
52            case "C": event=new CEvent(); break;
53            default:  System.out.println("Unsupported entry!");
54          }
55
56          if(event!=null) {
57              if (transitionExists(fsm.getCurrentState(), event)) {
58                  fsm.fire(event);
59                  path=buildPath(event);
60                  System.out.println("Processed sequence: "+ path);
61              }
62              else
63              { System.out.println("Invalid input for current state " + fsm.
    getCurrentState().getName());}
64          }
65      }
66
67      // Getters, Setters and printing methods
68      ...
69  }
```

Código 8.2 – Código de implementação de RecognizerFSM.



(a) Depois da entrada “k” no estado p.



(b) 30 segundos depois de (a).

Figura 89 – Mudanças em FSMNormal.
O Autor (2022).

O autômato FSMNormal representa o ao sistema subjacente no meta-modelo holístico, então, ele deve cumprir com as condições de ser adaptável, assim ele deve expor alguns sensores e atuadores além de fornecer um mecanismo de comunicação. Lembrando que os sensores e atuadores determinam as características sobre as quais pode ser implementado o comportamento adaptativo. No caso do nosso exemplo, as Figuras 89a e 89b representam o comportamento desejado para o autômato FSMNormal na sua versão adaptativa. Delas podemos observar que as mudanças envolvem a criação e remoção de transições e estados como resposta a novos tipos de entradas, por tanto, devemos garantir que nosso autômato possua sensores para identificar o evento de novos tipos de entradas e que existam atuadores para a inclusão e remoção de estados e transições.

Por essa razão, um novo autômato denominado FSMAdaptavel é criado no projeto “adaptavel-fsm”, onde é possível identificar tanto os sensores quanto os atuadores e o mecanismo de comunicação por eventos que permitem a passagem de informações desde e para o autômato. No exemplo, o nosso autômato irá reagir ao identifica o símbolo K, uma entrada que até o momento não estava dentro do seu alfabeto de entrada e que passará a

ser reconhecida por FSMAdaptavel. Para facilitar a tarefa de gerenciamento do alfabeto de entradas do autômato utilizamos a biblioteca Jackson[2] que permite interagir com arquivos na linguagem Json através da classe FSMEntry que representa os dados que uma entrada do autômato deve ter: o símbolo, a classe de evento que trata essa entrada e o nome do evento em particular a ser reconhecido. A classe AListener é a encarregada de processar os eventos que representam a identificação de entradas desconhecidas e passagem de tempo. A classe NNEvent permite o reconhecimento de entradas com símbolos fora do alfabeto do autômato, sua implementação é semelhante à dos outros eventos. A implementação de AListener está no Código 8.3. No Código 8.4 ressaltamos as partes significativas da implementação de FSMAdaptavel.

```
1  public class AListener {
2      private RecognizerFSM sys_U;
3
4      public void setSys_U(RecognizerFSM recognizer) {
5          this.sys_U=recognizer;
6          sys_U.setListener(new RecognizerFSM.IListener() {
7
8              @Override
9              public void onInvalidEntry(AbstractEvent event) {
10                 // code to process an invalid entry
11             }
12
13             @Override
14             public void onUnidentifiedEntry(AbstractEvent event) {
15                 // code to process an unidentified entry
16                 onTimer(5000);
17             }
18
19             public void onTimer(final int delay) {
20                 // timer configuration
21                 Timer timer=sys_U.getTimer();
22                 if (timer==null){
23                     timer=new Timer(delay,null);
24                     sys_U.setTimer(timer);
25                 }
26                 timer.setDelay(delay);
27                 timer.addActionListener(new ActionListener() {
28                     public void actionPerformed(ActionEvent evt) {
29                         // code to process reaching the time limit
30                         sys_U.getTimer().stop();
31                     }
32                 });
33                 timer.start();
34             }
35         });
36     }
37 }
```

Código 8.3 – Código de AListener.

[2] https://github.com/FasterXML/jackson-databind/

```java
public class RecognizerFSM {
   /*
    *  Some code containing members, getters, setters, constructor and some code
    for visualization and file management
    */

    public void processInput(String input) throws FiniteStateMachineException {
        AbstractEvent event=null;
        input=input.trim().toUpperCase();

        switch(input) {
        case "A":   event=new AEvent();     break;
        case "B":   event=new BEvent();     break;
        case "C":   event=new CEvent();     break;
        default:
                    if(!acceptedFSMEntry(input)) {
                        unidentifiedEntry(new NNEvent(input));
                    }
                    else {
                        event=new NNEvent(input);
                    }

        }

        if(event!=null) {
            if (transitionExists(fsm.getCurrentState(), event)) {
                fsm.fire(event);
                buildPath(event);
                System.out.println("Processed sequence: "+ path);
            }
            else
            {
                invalidEntry(event);
            }
        }
    }

    public  boolean acceptedFSMEntry(String entry) {
        boolean isAccepted=false;
        for(FSMEntry e:this.acceptedEntries) {
            if(e.getEntry().equalsIgnoreCase(entry)) {
                isAccepted=true;
            }
        }
        return isAccepted;
    }


    /*
     * Additional sensors for required events
     */

    private Timer timer;

    public Timer getTimer() {
        return timer;
    }
    public void setTimer(Timer timer) {
        this.timer= timer;
```

```
59      }
60
61
62      /*
63       * The communication mechanism is implemented through events and listeners
64       */
65
66      private IListener listener = null;
67
68      public IListener getListener() {
69          return this.listener;
70      }
71      public void setListener(IListener listener) {
72          this.listener = listener;
73      }
74
75      public interface IListener {
76          public void onInvalidEntry(AbstractEvent event);
77          public void onUnidentifiedEntry(AbstractEvent event);
78          public void onTimer(int delay);
79      }
80
81      public void invalidEntry( AbstractEvent e){
82          if (listener != null)
83              listener.onInvalidEntry(e);
84      }
85
86      public void unidentifiedEntry(AbstractEvent e){
87          if (listener != null)
88              listener.onUnidentifiedEntry(e);
89      }
90
91  }
```

Código 8.4 – Mudanças na classe RecognizerFSM.

Uma vez que nosso sistema subjacente possui as condições necessárias, é possível construir a lógica adaptativa. Para isso desenvolvemos o modelo de classes da Figura 90 utilizando a ferramenta Papyrus do Eclipse[3] e o projeto adaptiveLogic que o implementa, fornecendo reutilização dessa estrutura básica. Cabe mencionar que esta implementação é só uma maneira de realizar (ou instanciar) a lógica adaptativa especificada no meta-modelo holístico, considerando as características do problema em questão, e diversas outras podem ser implementadas de acordo com preferências do projetista ou as condições do problema.

[3] https://www.eclipse.org/papyrus/

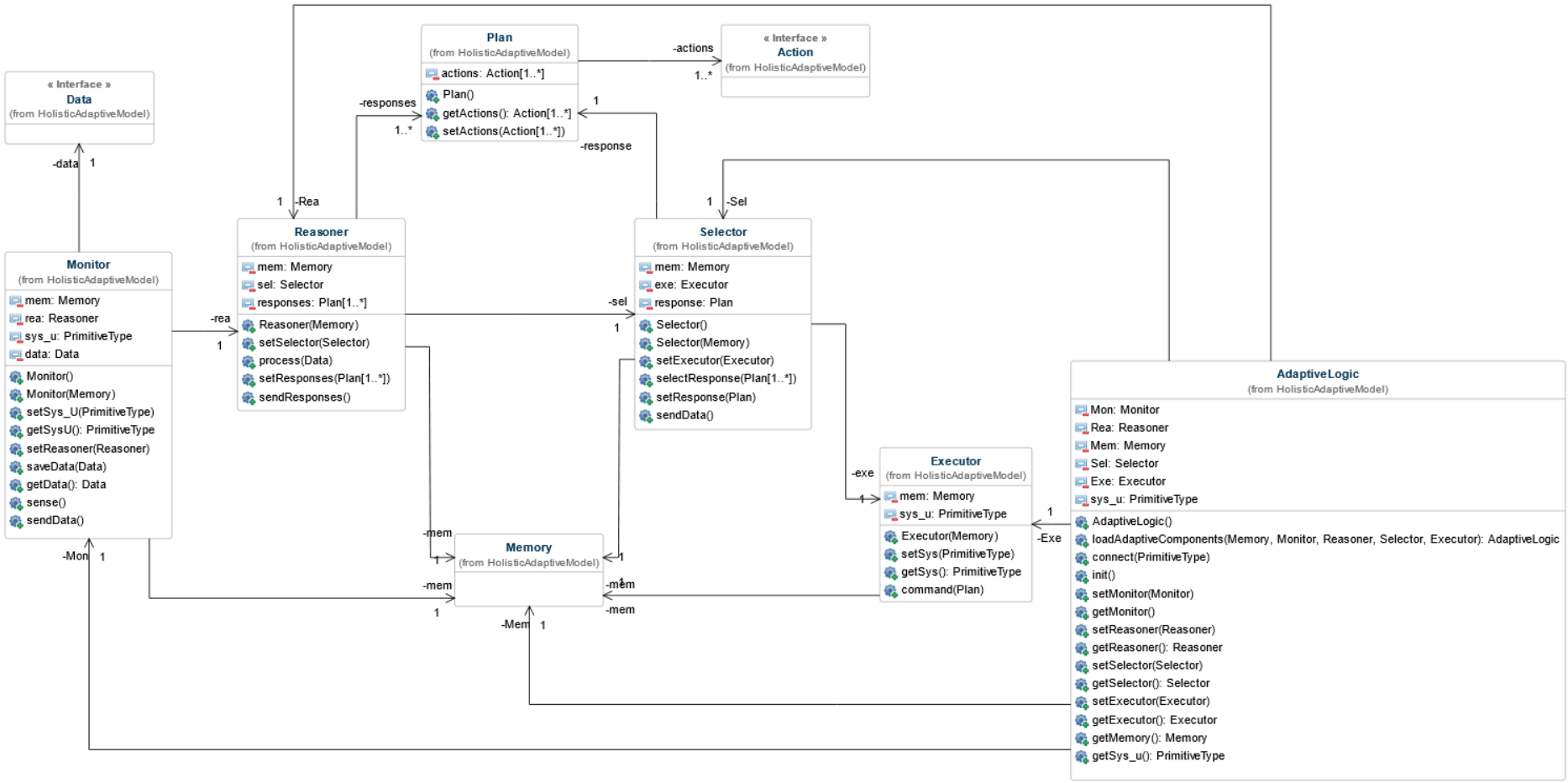


Figura 90 – Diagrama da Lógica Adaptativa realizada para a prova de conceito.
O Autor (2022).

As implementações das classes Monitor, Reasoner, Selector e Executor são apresentadas nos Códigos 8.5, 8.6, 8.7 e 8.8 correspondentemente. As implementações são simples e pretendem cobrir de maneira geral a funcionalidade da lógica adaptativa.

```
1  public class Monitor {
2      private Memory mem;
3      private Object sys_u;
4      private Reasoner rea;
5      private Data data;
6
7       public Monitor(Memory mem) {
8          this.mem=mem;
9       }
10
11
12     // Establishes the reference between the monitor and the system under
       observation
13     public void setSys_U(Object sys) {
14         this.sys_u=sys;
15     }
16
17     public Object getSysU() {
18         return this.sys_u;
19     }
20
21
22     // Connects the monitor and the next process
23     public void setReasoner(Reasoner rea) {
24         this.rea=rea;
25     }
26
27     public void saveData(Data data) {
28         this.data=data;
29     }
30
31     public Data getData() {
32         return this.data;
33     }
34
35     public void sense() {
36     }
37
38     public void sendData() {
39         rea.process(data);
40     }
41 }
```

Código 8.5 – Classe Monitor.

```
1  public class Reasoner {
2      private Memory Mem;
3      private Selector sel;
4      private List<Plan> responses;
5
6      public Reasoner(Memory mem) {
```

```
7           this.Mem=mem;
8       }
9
10      public void setSelector(Selector sel) {
11          this.sel=sel;
12      }
13
14      public void process(Data data) {
15      }
16
17      public void setResponses(List<Plan> responses) {
18          this.responses=responses;
19      }
20
21      public void sendResponses(){
22          sel.selectResponse(responses);
23      }
24
25  }
```

Código 8.6 – Classe Reasoner.

```
1   public class Selector {
2       private Memory Mem;
3       private Executor ex;
4       private Plan response;
5
6       public Selector(Memory mem) {
7           this.Mem=mem;
8       }
9
10      public void setExecutor(Executor ex) {
11          this.ex=ex;
12      }
13
14      public void selectResponse(List<Plan> responses) {
15      }
16
17      public void setResponse(Plan  response) {
18          this.response=response;
19      }
20
21      public void sendData(){
22          ex.command(response);
23      }
24
25  }
```

Código 8.7 – Classe Selector.

```
1   public class Executor {
2       private Memory Mem;
```

```
3       private Object sys_u;
4
5
6       public Executor(Memory mem) {
7           this.Mem=mem;
8       }
9
10      // Establishes the reference between the monitor and the system under
        observation
11      public void setSys_U(Object sys) {
12          this.sys_u=sys;
13      }
14
15      public Object getSysU() {
16          return this.sys_u;
17      }
18
19      public void command(Plan response) {
20      }
21  }
```

Código 8.8 – Classe Executor

```
1   public class AdaptiveLogic {
2       private Monitor Mon;
3       private Reasoner Rea;
4       private Selector Sel;
5       private Executor Ex;
6       private Memory Mem;
7       private Object sys_u; // Underlying System
8
9
10      public AdaptiveLogic(){}
11
12      // Structure is loaded with proper components
13      public AdaptiveLogic loadAdaptiveComponents(Memory mem, Monitor mon, Reasoner
         rea, Selector sel, Executor ex) {
14          this.Mem=mem;
15          this.Mon=mon;
16          this.Rea=rea;
17          this.Sel=sel;
18          this.Ex=ex;
19
20      // Connection in between adaptive processes
21          this.Mon.setReasoner(Rea);
22          this.Rea.setSelector(Sel);
23          this.Sel.setExecutor(Ex);
24
25          return this;
26      }
27
28
29      // Adaptive Logic connects with the underlying system
30      public boolean connect(Object sys) {
31          boolean result=false;
32
33          Mon.setSys_U(sys);
```

```
34          Ex.setSys_U(sys);
35          return result;
36      }
37
38      public void init() {
39          Mon.sense();
40      }
```

Código 8.9 – Classe AdaptiveLogic.

Para criar a camada adaptativa para o autômato FSMAdaptavel, precisamos criar os elementos da lógica adaptativa para o caso do sistema subjacente, no caso um autômato finito, e especificar o conteúdo de cada método através da capacidade de sobrescrita de Java, com essa finalidade criamos o projeto FSMAdaptativo. Assim, a classe Monitor_FSM estende a classe Monitor da lógica adaptativa e sobrescreve o método *sense*, que a sua vez cria uma classe anônima que herda da interface IListener do RecognizerFSM e trata os eventos de interesse. No nosso caso, o monitor processa os eventos de detecção de símbolos fora do alfabeto e tempo como mostra o Código 8.10.

```
1   public class Monitor_FSM extends Monitor {
2       private IListener listener;
3
4       public void sense() {
5           final RecognizerFSM sys_U=(RecognizerFSM)this.getSysU();
6
7           listener=new RecognizerFSM.IListener() {
8
9               @Override
10              public void onInvalidEntry(AbstractEvent event) {
11                  System.out.println("Invalid input for current state "+ sys_U.
    getFSM().getCurrentState().getName() +" from Monitor");
12                  saveData(new FSMData(sys_U.getFSM().getCurrentState(),event));
13                  saveDataToFile();
14                  sendData();
15              }
16
17              @Override
18              public void onUnidentifiedEntry(AbstractEvent event) {
19                  System.out.println("Unidentified input " +event.getName()+"
    detected at state "+ sys_U.getFSM().getCurrentState().getName()+" reported
    from Monitor");
20                  saveData(new FSMData(sys_U.getFSM().getCurrentState(),event));
21                  saveDataToFile();
22                  sendData();
23              }
24
25              public void onTimer(final int delay) {
26                  Timer timer=sys_U.getTimer();
27                  if (timer==null){
28                      timer=new Timer(delay,null);
29                      sys_U.setTimer(timer);
30                  }
```

```
31                  timer.setDelay(delay);
32                  timer.addActionListener(new ActionListener() {
33                      public void actionPerformed(ActionEvent evt) {
34                          System.out.println(delay+" seconds have passed since K
    was detected as input");
35                          sys_U.getTimer().stop();
36                          saveData(new FSMData(sys_U.getFSM().getCurrentState(),new
     NNEvent("TimeEvent") {
37                          }));
38                          saveDataToFile();
39                          sendData();
40                      }
41                  });
42
43                  timer.start();
44              }
45          };
46
47          sys_U.setListener(listener);
48      }
49
50      public void saveDataToFile() {
51           ObjectMapper mapper = new ObjectMapper();
52          mapper.enable(SerializationFeature.INDENT_OUTPUT);
53           try {
54              mapper.writeValue(Paths.get("Monitor_Data.json").toFile(), this.
    getData());
55          } catch (IOException e) {
56              e.printStackTrace();
57          }
58
59      }
60
61  }
```

Código 8.10 – Classe Monitor_FSM.

Procedemos da mesma maneira com a classe Reasoner_FSM, ela herda da classe Reasoner da lógica adaptativa e sobrescreve o método que realiza o processamento dos dados enviador pelo monitor. A implementação para tanto para a identificação da entrada do símbolo k no estado P, ilustrado na Figura 89a quanto para a passagem de um tempo determinado, como indicado na Figura 89b estão expressas no Código 8.11.

```
1   public class Reasoner_FSM extends Reasoner{
2
3       public Reasoner_FSM(Memory mem) {
4           super(mem);
5       }
6
7       public void process(Data data) {
8           FSMData fsm_data=(FSMData)data;
9           State s=fsm_data.getState();
10          List<Plan> responses=new ArrayList<Plan>();;
11
```

```
12          AbstractEvent e=fsm_data.getEvent();
13          System.out.println("Processing Data with reasoner");
14          if(s.getName().equalsIgnoreCase("P") &&
15              e.getName().equalsIgnoreCase("K")) {
16              Plan plan=new Plan();
17              List<Action> actions=new ArrayList<Action>();
18
19              actions.add(new FSMAction("new", "state", "P1"));
20              actions.add(new FSMAction("new", "transition", "P to P1", "P", "P1",
    "KEvent"));
21              actions.add(new FSMAction("new", "transition", "P1 to Q", "P1", "Q",
    "ZEvent"));
22              plan.setActions(actions);
23              responses.add(plan);
24
25          }
26          if(e.getName().equalsIgnoreCase("TimeEvent")) {
27              Plan plan=new Plan();
28              List<Action> actions=new ArrayList<Action>();
29              actions.add(new FSMAction("remove", "transition", "P to Q","P","Q","
    BEvent"));
30              plan.setActions(actions);
31              responses.add(plan);
32          }
33          setResponses(responses);
34          sendResponses();
35
36      }
37  }
```

Código 8.11 – Classe Reasoner_FSM.

Para o processo de seleção, implementamos a classe Selector_FSM que estende a classe Selector da lógica adaptativa. Esta classe é a encarregada de selecionar o plano de ação ou estratégia adequada para responder à situação corrente. No caso do nosso exemplo, o autômato finito é determinístico, o que significa que se existe alguma resposta para a dupla <estado, entrada> esta é única, e por tanto não existe a necessidade de um mecanismo de seleção: ou existe um único plano, ou não existe estratégia. Porém, é possível implementar diferentes abordagens aplicando a medida de desempenho e critérios ou políticas de decisão. O código mostra a implementação básica de Selector_FSM.

```
1   public class Selector_FSM extends Selector {
2
3       public Selector_FSM(Memory mem) {
4           super(mem);
5       }
6
7       public void selectResponse(List<Plan> responses) {
8           System.out.println("Selecting Response with Selector");
9           if (!responses.isEmpty()){
10              if(responses.size()>1) {
11                  // There are more than 1 option to select from
```

```
12                }
13                else {
14                    this.setResponse(responses.get(0));
15                }
16                sendData();
17            }
18            else {
19                // behavior when there are no responses from the reasoner
20            }
21        }
22    }
```

Código 8.12 – Classe Selector_FSM.

O último passo do comportamento adaptativo é desenvolvido pelo Executor_FSM. Ele herda da classe Executor da lógica adaptativa e sobrescreve o método *command*, além de implementar alguns métodos intermediários para traduzir as ações do plano selecionado no processo anterior. Assim, o Código 8.13 apresenta o conteúdo mais importante implementado na classe Executor_FSM para produzir o resultado esperado segundo para nosso exemplo.

```
1   public class Executor_FSM extends Executor {
2
3       public Executor_FSM(Memory mem) {
4           super(mem);
5       }
6
7
8       public void command(Plan response) {
9           System.out.println("Commanding Actions from Executor");
10          RecognizerFSM recognizer=(RecognizerFSM)this.getSysU();
11          FiniteStateMachine fsm= (FiniteStateMachine)recognizer.getFSM();
12
13          if (response!=null) {
14              System.out.println("Executor needs to translate the plan and modify
    the FSM");
15              FSMAction action=(FSMAction)response.getActions().get(0);
16              if(action.getType().equalsIgnoreCase("remove")) {
17                  applyRemoveAction(action, recognizer, fsm);
18              }
19              else if(action.getType().equalsIgnoreCase("new")) {
20                  applyNewAction(action, recognizer, fsm);
21              }
22
23          }
24
25      }
26
27      @SuppressWarnings("rawtypes")
28      private void applyRemoveAction(FSMAction action,RecognizerFSM recognizer,
    FiniteStateMachine fsm) {
29          if(action.getElement().equalsIgnoreCase("transition")) {
30              String source=action.getSource();
31              Class eventTypeClass=getEventClassType(action.getEventType());
32              Set<Transition> transitions= fsm.getTransitions();
```

```
33              for (Transition t: transitions) {
34                  //looking for transition "P to Q with B"
35                  if (t.getSourceState().getName().equalsIgnoreCase(source)&&
36                          (t.getEventType()==eventTypeClass)){
37                      transitions.remove(t);
38                      break;
39                  }
40              }
41
42              recognizer.printCurrentState(Launcher.viewer);
43
44          }else {
45              // code to remove a state
46          }
47      }
48
49      private void applyNewAction(FSMAction action,RecognizerFSM recognizer,
        FiniteStateMachine fsm) {
50          // Create new state
51          State p1_State = new State("P1");
52          fsm.getStates().add(p1_State);
53
54          // Look for source state
55
56          Transition p_p1 = new TransitionBuilder()
57                  .name("P to P1 with k")
58                  .sourceState(fsm.getCurrentState())
59                  .eventType(NNEvent.class)
60                  .targetState(p1_State)
61                  .build();
62
63          Set<State> states=fsm.getStates();
64          State q_State=null;
65
66          // Look for target state
67          for(State s:states) {
68              if (s.getName().equalsIgnoreCase("Q")) {
69                  q_State=s;
70              }
71          }
72
73          //Build transition
74          Transition p1_q = new TransitionBuilder()
75                  .name("P1 to Q with z")
76                  .sourceState(p1_State)
77                  .eventType(NNEvent.class)
78                  .targetState(q_State)
79                  .build();
80
81          // Add transition to transitions set
82          Set<Transition> transitions= fsm.getTransitions();
83          transitions.add(p_p1);
84          transitions.add(p1_q);
85
86          // Modify accepted entries for FSM
87          recognizer.addFSMAcceptedEntry("K");
88          recognizer.addFSMAcceptedEntry("Z");
89
90          int delay=3000;
91          recognizer.getListener().onTimer(delay);
```

```
92
93          try {
94              AbstractEvent event=new NNEvent("KEvent");
95              fsm.fire(event);
96              recognizer.buildPath(event);
97              System.out.println("Processed sequence: "+ recognizer.getPath());
98          } catch (FiniteStateMachineException e) {
99              // TODO Auto-generated catch block
100             e.printStackTrace();
101         }
102     }
103
104 }
```

Código 8.13 – Classe Executor_FSM.

Podemos observar que as implementações do FSMAdaptavel e os elementos da lógica adaptativa Monitor_FSM, Reasoner_FSM, Selector_FSM e Executor_FSM estão desacopladas, de fato o projeto FSMAdaptativo só contém a referência para FSMAdaptavel de maneira que não modifica as funcionalidades do sistema subjacente e mantém, no nível de implementação, a separação de interesses. O Código 8.14 expõe o uso e conexão entre o sistema subjacente e a lógica adaptativa correspondente.

```
1     public static void main(String[] args) throws FiniteStateMachineException,
      IOException {
2
3
4         /*
5          * Create a RecognizerFSM instance
6          */
7         RecognizerFSM recognizer=new RecognizerFSM();
8
9         /*
10         * Setup GraphViz
11         */
12
13        viewer.addln(viewer.start_graph());
14        viewer.setup_graph("GraphViz.config");
15
16        /*
17         * Create a AdaptiveLogic instance
18         */
19
20        AdaptiveLogic AL=new AdaptiveLogic();
21        createAdaptiveLogic(AL,recognizer);
22        AL.init();
23
24
25        /*
26         * Fire some events and print FSM state
27         */
28
29
```

```
30          recognizer.printCurrentState(viewer);
31
32          Scanner scanner = new Scanner(System.in);
33          System.out.println("Insert an input or Press [q] to quit tutorial.");
34
35          while (true) {
36              String input = scanner.nextLine();
37
38              recognizer.processInput(input);
39              recognizer.printCurrentState(viewer);
40
41              if (recognizer.atFinalState()) {
42                  System.out.println("Recognizer has reach final state ");
43                  System.exit(0);
44                  scanner.close();
45              }
46
47              if (input.trim().equalsIgnoreCase("q")) {
48                  System.out.println("input = " + input.trim());
49                  System.out.println("Bye!");
50                  System.exit(0);
51                  scanner.close();
52              }
53
54          }
55
56      }
57
58
59      public static void createAdaptiveLogic(final AdaptiveLogic AL,RecognizerFSM
        sys_U) {
60          Memory mem= new Memory();
61          Monitor mon=new Monitor_FSM(mem);
62          Reasoner rea=new Reasoner_FSM(mem);
63          Selector sel=new Selector_FSM(mem);
64          Executor ex=new Executor_FSM(mem);
65
66          AL.loadAdaptiveComponents(mem, mon, rea, sel, ex);
67          AL.connect(sys_U);
68      }
69
70  }
```

Código 8.14 – Classe App do projeto FSMAdaptativo.

Dessa maneira, o autômato FSMAdaptavel passa a desenvolver o comportamento adaptativo descrito nas Figuras 89a e 89b.

8.1.3 *Meta-adaptatividade*

Ampliamos o escopo do exemplo apresentado na seção anterior para incluir na prova de conceito a implementação de meta-adaptatividade como uma segunda aplicação do meta-modelo holístico. A lógica a implementar no nível de meta-adaptatividade é seguinte: se o dispositivo adaptativo perceber que uma entrada não identificada foi detectada o máximo

número de vezes permitido (informado previamente) ele deve bloquear o processamento dessa entrada. Assim, o dispositivo adaptativo deve mudar o mecanismo de monitoramento das entradas não identificadas para refletir o comportamento desejado. Com essa finalidade criamos 2 novos projetos no repositório:

- Adaptable-FSM-Adaptive, que inclui novas classes e mecanismos que funcionam como sensores e atuadores, e um listener para estabelecer a comunicação entre o autômato adaptativo e a lógica meta-adaptativa.
- Meta-Adaptive-FSM, que representa uma nova instância da lógica adaptativa apresentada anteriormente.

O nosso dispositivo subjacente, que nesse caso é um dispositivo adaptativo, precisa fornecer métodos que funcionem como sensores e atuadores que permitam o comportamento desejado. Assim, a classe Monitor_FSM deve fornecer novas propriedades e métodos que permitam boquear uma entrada, identificar as entradas bloqueadas, o número máximo de vezes que uma entrada não identificada é tolerada e acompanhar a lista de entradas bloqueadas. O Código 8.15 apresenta as mudanças da classe Monitor_FSM com respeito à implementação anterior.

```
1  public class Monitor_FSM extends Monitor {
2      private RecognizerFSM sys_U;
3
4      /*
5       * New properties to represent the adaptable behavior
6       * */
7
8      private EntryTracker_FSM tracker;
9      private List<String> blockedEntries;
10
11     private ILimitReachedEvent limitReachedListener;
12     private UndefinedEntryListener undefinedEntryListener;
13
14
15     public Monitor_FSM (Memory mem, int max) {
16         super(mem);
17         this.tracker=new EntryTracker_FSM();
18         this.tracker.setLimit(max);
19         this.blockedEntries=new ArrayList<String>();
20         this.limitReachedListener=null;
21         undefinedEntryListener=null;
22     }
23
24     // Code for Setters and Getters
25
26     /*
27      * New sensor to identify when an unidentified entry has reach the max. num
       of appearances
28      * */
29
```

```
30      public interface ILimitReachedEvent{
31          public void onLimitReached(String entry, EntryTracker_FSM tracker);
32      }
33
34      public void sense() {
35      // same as before
36      }
37
38      public void saveDataToFile() {
39      // same as before
40      }
41
42      /*
43       * Functionality to track the number of tries an undefined entry has
44       * */
45
46      public void trackUnidentifiedEntries(String entry) {
47          tracker.addEntry(entry);
48          if (tracker.verifyLimit(entry)) {
49                  this.limitReachedListener.onLimitReached(entry, tracker);
50          }
51      }
52
53      /*
54       * Functionality to verify if a particular entry has reached the max. num of
        appearances allowed
55       * */
56
57      public boolean entryBlocked(String entry) {
58          boolean result=false;
59          for (String blocked: blockedEntries) {
60              if(blocked.equalsIgnoreCase(entry)) {
61                  result=true;
62              }
63          }
64          return result;
65      }
66
67  }
```

Código 8.15 – Classe EntryTracker_FSM.

O dispositivo adaptável Adaptable-FSM-Adaptive implementa 2 novas classes: EntryTracker_FSM.java e AdaptiveLogicListener.java. A classe EntryTracker_FSM implementa um rastreador simples, apresentada no Código 8.16, que salva o número de vezes que uma entrada foi identificada.

```
1  public class EntryTracker_FSM {
2      private HashMap<String,Integer> dictionary;
3      private Integer limit;
4
5      public EntryTracker_FSM() {
6          dictionary=new HashMap<String, Integer>();
7      }
```

```
8
9       public Integer getLimit() {
10          return limit;
11      }
12
13      public void setLimit(Integer limit) {
14          this.limit = limit;
15      }
16
17      public void addEntry(String entry) {
18          if (dictionary.containsKey(entry)){
19              dictionary.replace(entry, dictionary.get(entry)+1);
20          }else {
21              dictionary.put(entry, 1);
22          }
23      }
24
25      public boolean verifyLimit(String entry) {
26          return dictionary.get(entry)==limit;
27      }
28  }
```

Código 8.16 – Classe EntryTracker_FSM.

A classe AdaptiveLogicListener, apresentada no Código 8.16, processa o evento lançado quando uma entrada não identificada atinge o limite máximo

```
1   public class AdaptiveLogicListener{
2
3       private AdaptiveLogic sys_U;
4       public AdaptiveLogicListener() {
5       }
6
7       public void setSys_U(AdaptiveLogic AL) {
8           this.sys_U=AL;
9           Monitor_FSM monitor= (Monitor_FSM)sys_U.getMonitor();
10          monitor.setLimitReachedListener(new ILimitReachedEvent() {
11
12              @Override
13              public void onLimitReached(String entry, EntryTracker_FSM tracker) {
14                  monitor.getBlockedEntries().add(entry);
15              }
16          });
17      }
18  }
```

Código 8.17 – Classe AdaptiveLogicListener.

Pela sua vez, o projeto Meta-Adaptive-FSM, implementa as seguintes classes:

- Meta_Monitor_FSM: representa o elemento Monitor da lógica adaptativa no nível

meta-adaptativo.

- Meta_Reasoner_FSM: representa o elemento Reasoner da lógica adaptativa no nível meta-adaptativo.
- Meta_Selector_FSM: representa o elemento Selector da lógica adaptativa no nível meta-adaptativo.
- Meta_Executor_FSM: representa o elemento Executor da lógica adaptativa no nível meta-adaptativo.
- ALData: representa um dado identificado pelo Monitor e transferido para o Reasoner na lógica adaptativa.
- ALAction: representa uma ação, e serve para a transferência de informações entre o Reasoner e o Selector da lógica adaptativa.

As classes Meta_Reasoner_FSM e Meta_Selector_FSM apresentam poucas mudanças com respeito às implementações anteriores, por essa razão não entraremos em maiores detalhes. A classe Meta_Monitor_FSM é de nosso interesse, essa classe implementa o listener necessário para iniciar o processo: onLimitReached. O Código 8.18 apresenta a implementação dessa classe.

```
1  public class Meta_Monitor_FSM extends Monitor{
2      private ILimitReachedEvent limitReachedlistener;
3      private AdaptiveLogic sys_U;
4
5      public Meta_Monitor_FSM () {};
6      public Meta_Monitor_FSM  (Memory mem) {
7          super(mem);
8      }
9
10     public void sense() {
11         sys_U=(AdaptiveLogic)this.getSysU();
12         Monitor_FSM monitor= (Monitor_FSM)sys_U.getMonitor();
13         limitReachedlistener=new Monitor_FSM.ILimitReachedEvent(){
14
15             @Override
16             public void onLimitReached(String entry, EntryTracker_FSM tracker) {
17                 System.out.println("A frequent Unidentified entry detected from
    Meta-Monitor");
18                 saveData(new ALData("onLimitReached",entry));
19                 saveDataToFile();
20                 sendData();
21
22             }
23
24         };
25         monitor.setLimitReachedListener(limitReachedlistener);
26
27     }
```

```
28
29      public void saveDataToFile() {
30      // save data from Meta_Monitor_FSM to file
31      }
32
33  }
```

Código 8.18 – Classe Meta_Monitor_FSM.

A classe Meta_Executor_FSM implementa a última fase da lógica adaptativa, em particular substitui o listener encarregado de processar as entradas não identificadas no dispositivo subjacente (o Monitor_FSM) para incluir no processo a verificação de entradas bloqueadas. O Código 8.19 apresenta essa implementação.

```
1   public class Meta_Executor_FSM extends Executor{
2
3       private Monitor_FSM monitor;
4
5       public Meta_Executor_FSM(Memory mem) {
6           super(mem);
7       }
8
9       public void command(Plan response) {
10          System.out.println("Commanding Actions from Meta-Executor");
11          AdaptiveLogic AL=(AdaptiveLogic)this.getSysU();
12          if (response!=null) {
13              System.out.println("Meta-Executor needs to translate the plan and
    modify the AL");
14              ALAction action=(ALAction)response.getActions().get(0);
15              if(action.getType().equalsIgnoreCase("block")) {
16                  applyBlockEntryAction(action, AL);
17              }
18
19          }
20
21      }
22
23      private void applyBlockEntryAction(ALAction action, AdaptiveLogic AL) {
24          System.out.println("Meta-Executor blocking (ignoring) the use of entry "+
     action.getElement());
25          monitor=(Monitor_FSM)((AdaptiveLogic)this.getSysU()).getMonitor();
26          monitor.getBlockedEntries().add(action.getElement());
27          monitor.setUndefinedEntryListener(
28                  (event)->{
29                      System.out.println("Unidentified input " +event.getName()+"
    detected at state "+ ((RecognizerFSM)monitor.getSysU()).getFSM().
    getCurrentState().getName()+" reported from NEW UnidentifiedEntryListener");
30                      monitor.trackUnidentifiedEntries(event.getName());
31                      if (!monitor.entryBlocked(event.getName())) {
32                          monitor.saveData(new FSMData(((RecognizerFSM)monitor.
    getSysU()).getFSM().getCurrentState(),event));
33                          monitor.saveDataToFile();
34                          monitor.sendData();
```

```
35                      }else {
36                          System.out.println("Blocked entry");

37                      }
38                  }
39                  );
40      }
41  }
```

Código 8.19 – Classe Meta_Executor_FSM

Finalmente, na classe principal do projeto, App.java, é realizada a criação da lógica meta-adaptativa da maneira análoga à criação da lógica adaptativa. O Código 8.20 apresenta a implementação dessa classe.

```
1
2       public static void main(String[] args) throws FiniteStateMachineException,
        IOException {
3
4           /*
5            * The max. num of appearances allowed for unidentified entries can be
        loaded from a configuration file. In this case we indicate it will be 3.
6            */
7           max_app_entry=3;
8
9           /*
10           * Create a AdaptiveLogic instance
11           */
12
13          AdaptiveLogic AL=new AdaptiveLogic();
14          createAdaptiveLogic(AL,recognizer);
15          AL.init();
16
17          /*
18           * Create a second AdaptiveLogic instance (meta-adaptive logic)
19           */
20          AdaptiveLogic metaAL=new AdaptiveLogic();
21          createMetaAdaptiveLogic(metaAL, AL);
22          metaAL.init();
23
24
25          // Code to use FSM
26
27      }
28
29
30      public static void createAdaptiveLogic(final AdaptiveLogic AL,RecognizerFSM
        sys_U) {
31          Memory mem= new Memory();
32          Monitor mon=new Monitor_FSM(mem,max_app_entry);
33          Reasoner rea=new Reasoner_FSM(mem);
34          Selector sel=new Selector_FSM(mem);
35          Executor ex=new Executor_FSM(mem);
36
37          AL.loadAdaptiveComponents(mem, mon, rea, sel, ex);
```

```
38          AL.connect(sys_U);
39
40      }
41
42      public static void createMetaAdaptiveLogic(final AdaptiveLogic metaAL,
        AdaptiveLogic sys_U) {
43          Memory meta_mem= new Memory();
44          Monitor meta_mon=new Meta_Monitor_FSM(meta_mem);
45          Reasoner meta_rea=new Meta_Reasoner_FSM(meta_mem);
46          Selector meta_sel=new Meta_Selector_FSM(meta_mem);
47          Executor meta_ex=new Meta_Executor_FSM(meta_mem);
48
49          metaAL.loadAdaptiveComponents(meta_mem, meta_mon, meta_rea, meta_sel,
        meta_ex);
50          metaAL.connect(sys_U);
51      }
52
53  }
```

Código 8.20 – Classe App do projeto Meta-Adaptive-FSM

Ao acompanhar com maior detalhe a implementação da prova de conceito é possível observar algumas sutilezas nos diferentes níveis. Num primeiro momento, no Nível 0, temos o sistema subjacente equipado com os mecanismos necessários para incorporar a adaptatividade. No caso do exemplo, o dispositivo subjacente permite a escuta dos eventos unidentifiedEntry e invalidEntry como apresenta a Figura 91.

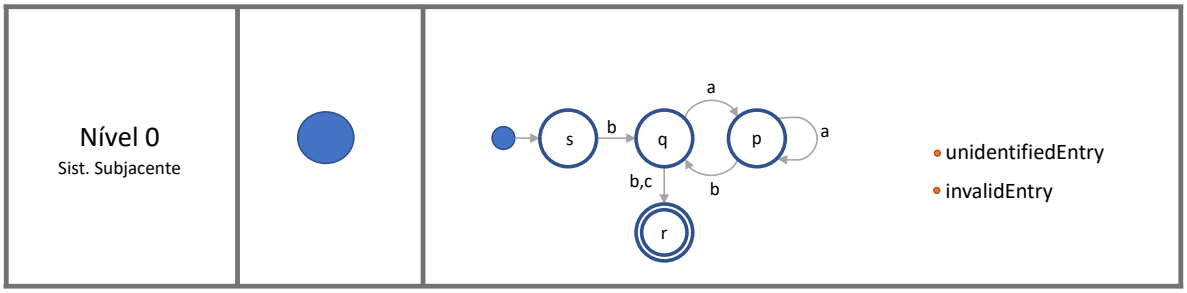


Figura 91 – Configuração Padrão do Autômato FSMNormal.
O Autor (2022).

O Nível 1 corresponde com a lógica adaptativa e contém os elementos próprios dessa camada: Mon, Rea, Sel, Exe. Essa camada é ativada quando algum dos eventos sendo monitorados acontece, no caso do exemplo a identificação da entrada k no estado p dispara o evento unidentifiedEntry. Em seguida, uma pilha entre níveis salva as informações da configuração corrente do dispositivo subjacente e a entrada que disparou a chamada para a lógica adaptativa. Pela sua parte, a lógica adaptativa inicia o seu fluxo de ação normalmente para fornecer uma resposta à entrada k. Essa dinâmica é apresentada na Figura 92.

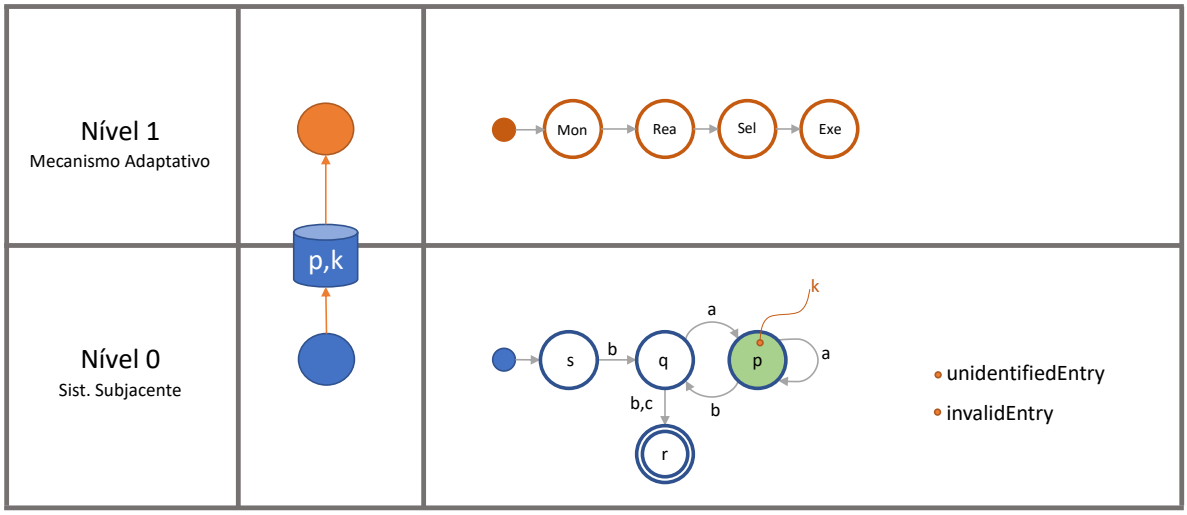


Figura 92 – Configuração inicial do Autômato FSMAdaptativo.
O Autor (2022).

Ao finalizar suas operações, a lógica adaptativa retorna o controle para o sistema subjacente juntamente com as ações a serem realizadas para responder ao evento identificado. No caso do exemplo, as ações são a criação do estado p1, a criação das transições: (i) p → p1 com k, e (ii) p1 → q com z, e a ativação do monitoramento de um novo evento de tempo com alguns segundos de espera. Após realizar as ações propostas o dispositivo subjacente retoma suas funções, neste caso realiza a transição de p a p1 com k que é a entrada atual. A Figura 93 representa essas condições de trabalho.

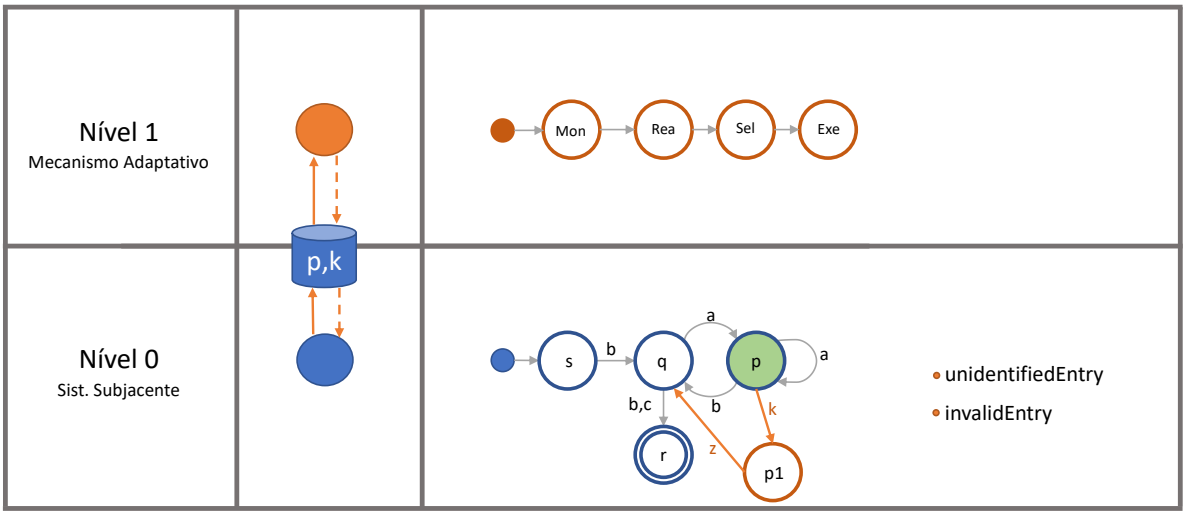


Figura 93 – Retono do Autómato FSMAdaptativo.
O Autor (2022).

O Nível 2 corresponde como a lógica meta-adaptativa, que em particular é uma lógica adaptativa, e também contém os elementos próprios dessa camada. Essa camada é ativada quando algum dos eventos sendo monitorados acontece, no caso do exemplo a leitura da entrada não identificada “U” o número máximo de vezes permitido (3x) através

do evento onLimitReached. Nesse momento uma segunda pilha entre níveis é criada e salva as mesmas informações: a configuração corrente do dispositivo subjacente (lógica adaptativa) e a entrada que disparou a chamada para a lógica adaptativa (lógica meta-adaptativa). Em seguida, a lógica meta-adaptativa inicia o seu fluxo de ação normalmente para fornecer uma resposta ao evento onLimitReached. Essa dinâmica é apresentada na Figura 94.

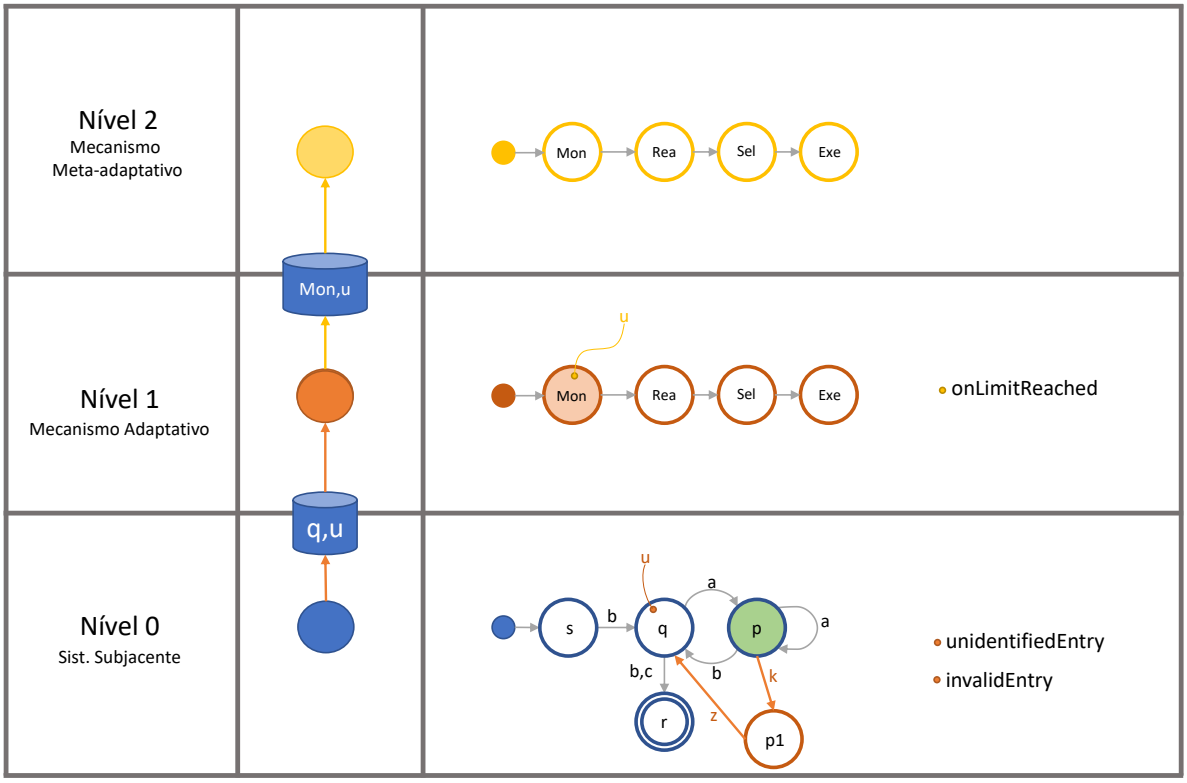


Figura 94 – Configuração Padrão do Autómato FSM_MetaAdaptativo.
O Autor (2022).

Ao finalizar suas operações, a lógica meta-adaptativa retorna o controle para o sistema subjacente, neste caso a lógica adaptativa, juntamente com as ações a serem realizadas para responder ao evento identificado. No caso do exemplo é a substituição do processo de monitoramento por uma nova implementação que realiza a verificação das entradas bloqueadas. Esse novo monitor é passa a fazer parte do processo da lógica adaptativa, porém só será ativado no próximo ciclo de execução, pois a referência ao elemento anterior, ao qual chamaremos de monitor antigo, ficou salva na pilha e foi o ponto no qual o controle voltou para a lógica adaptativa, preservando a coerência do ciclo de execução. A Figura 95 reflete essa situação.

A prova de conceito nos permite observar tanto um exemplo de instância do meta-modelo holístico quanto a implementação do mesmo para uma linguagem de programação particular. Dessa maneira podemos comprovar que a aplicação dos seus conceitos e construções conferem as vantagens de flexibilidade e reutilização desejadas.

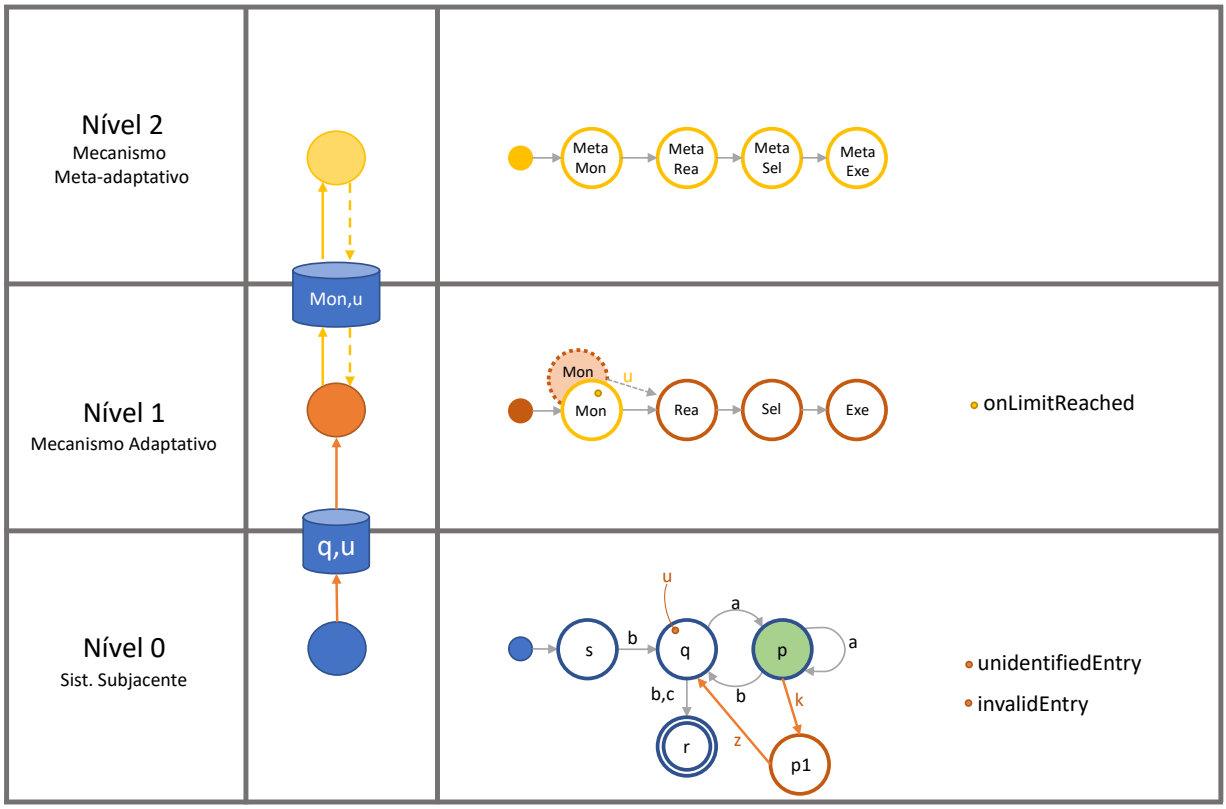


Figura 95 – Retono do Autómato FSMA_MetaAdaptativo.
O Autor (2022).

8.2 Provas no Nível Tecnológico

Para verificar o funcionamento do framework aplicamos 2 estratégias de testes: testes de unidade e testes de sistema. Por um lado, os testes de unidade são responsáveis de avaliar o funcionamento específico de pequenas unidades de código de maneira isolada do restante do sistema. No nosso caso, os testes de unidade foram especificamente desenvolvidos para:

(i) verificar as funcionalidades básicas dos componentes GUI nos componentes da ferramenta;

(ii) verificar o correto gerenciamento dos recursos no sistema de arquivos e os arquivos de configurações, e

(iii) verificar a lógica dos casos de uso.

Os testes de unidade são realizados através do desenvolvimento de classes de teste na própria estrutura de ambos componentes da ferramenta, como mostra a Figura 96, os quais podem ser executados de maneira manual quanto automatizada. As ferramentas que suportam a execução dos testes de unidade estão detalhadas na Seção 8.2.1.2.

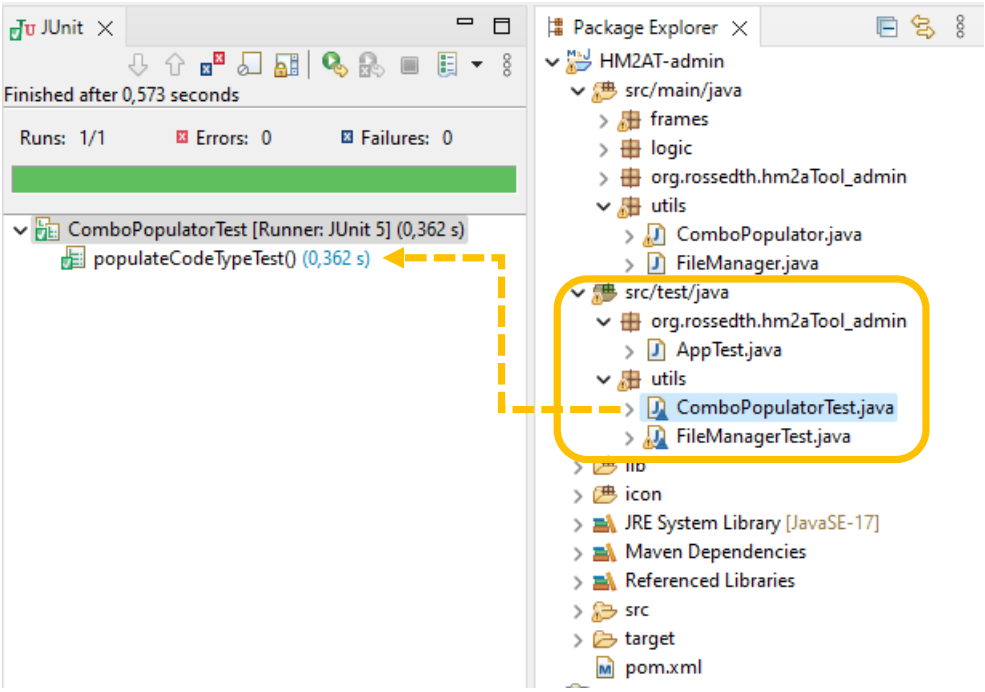


Figura 96 – Exemplo da estrutura dos casos de teste de unidade nos projetos da ferramenta. O Autor (2022)

Por outro lado, os testes de sistema buscam replicar o comportamento de um usuário com o software em um ambiente de sistema completo. Eles tratam o comportamento de todo o sistema definido pelo escopo do projeto através de vários fluxos ou casos de teste. No nosso caso, os testes do sistema executados consistem na aplicação de um fluxo completo de interação com cada componente do framework, o HM^{2}AT-Admin e o HM^{2}AT, detalhados na Seção 8.2.1.4 deste capítulo.

Os testes não funcionais, tais como testes de desempenho, testes de estresse, embora muito recomendados, estão fora do escopo deste projeto por razões de tempo. No entanto, são considerados parte de trabalhos futuros.

8.2.1 Configuração dos Casos de Testes

Nesta seção são apresentados os detalhes tecnológicos e técnicos dos testes aplicados para verificar o funcionamento das ferramentas que compõem o framework HM^{2}AT. Os detalhes tecnológico descrevem o equipamento e o software utilizado para a realização dos testes, enquanto os detalhes técnicos descrevem a maneira em que os testes verificam o comportamento esperado das ferramentas do framework.

8.2.1.1 Equipamento utilizado

Tanto os testes de unidade quanto os testes de sistema foram aplicados sob as seguintes especificações tecnológicas:

- Sistema Operacional Windows 11 de 64 bits
- Processador 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 1.69 GHz
- RAM 16,0 GB

8.2.1.2 Ferramentas utilizadas

A aplicação dos testes está subdividida em duas atividades: aplicação de testes de unidade e aplicação de testes de sistema. Existem algumas diferenças importantes entre ambas atividades. Por um lado, a aplicação de testes de unidade foi desenvolvida e aplicada de maneira progressiva ao longo do desenvolvimento da ferramenta. Por outro lado, os testes de sistema foram projetados e aplicados após a finalização do protótipo da ferramenta com base nas especificações de casos de uso. Adicionalmente, os testes de unidade foram realizados através do ambiente de desenvolvimento e tendo acesso ao código-fonte, enquanto os testes de sistema foram aplicados sobre os arquivos JAR gerados. Finalmente, os testes de unidade foram automatizados com ajuda de plug-ins no ambiente de desenvolvimento, enquanto os testes de sistema foram realizados de forma manual.

As ferramentas utilizadas para a automação dos testes de unidade foram:

- Plugin Surefire do Maven versão 2.22.1
- Eclipse IDE Version: 2022-09 (4.25.0)
- JUnit versão 5 5.8.2
- JUnit HTML Report versão 2.22.0

8.2.1.3 Suíte de testes

A maioria dos testes de unidade estão relacionados à verificação de funcionalidades básicas de GUI e ao gerenciamento de recursos de sistema. Esses testes, embora necessários desde a visão de desenvolvimento de sistema, não são de especial interesse para assunto de pesquisa neste documento. A maneira informativa, as Tabelas 8 e 9 apresentam um resumo dos casos de teste realizados por cada componente da ferramenta desenvolvida. Os resultados de todos os testes de unidade foram disponibilizados no repositório que acompanha o documento no formato de relatório de teste elaborado pelo JUnit Report.

Por outro lado, existem testes que avaliam a aplicação dos conceitos desenvolvidos neste trabalho na ferramenta, eles são:

- Teste de avaliação de adaptabilidade.
- Teste de instanciação do meta-modelo de adaptatividade.

Pacote	Classe	Método	Resultado
frames	SettingsPanel	createRepository(Path path)	sucesso
frames	SettingsPanel	updateIndexes()	sucesso
frames	ModelPanel	submitModel(Model model)	sucesso
frames	ImplementationPanel	validateModel()	sucesso
frames	ImplementationPanel	saveImplementation(Implementation imp)	sucesso
frames	DependencyPanel	validateImplementation()	sucesso
frames	DependencyPanel	saveDependency(Dependency dep)	sucesso
frames	ExamplePanel	validateImplementation()	sucesso
frames	ExamplePanel	saveExample(Example ex)	sucesso
logic	AdaptivityModel	gets & sets	sucesso
logic	AdaptivityModel	loadFromJSON(File file)	sucesso
logic	AdaptivityModel	updateAdaptivityModel(...)	sucesso
logic	AdaptivityExample	gets & sets	sucesso
logic	AdaptivityExample	updateExample(...)	sucesso
logic	Dependency	gets & sets	sucesso
logic	Dependency	update(List<String> dep)	sucesso
logic	AdaptivityImplementation	gets & sets	sucesso
logic	AdaptivityImplementation	loadScript(File f)	sucesso
logic	AdaptivityImplementation	executeScript()	sucesso
logic	Repository	gets & sets	sucesso
logic	Repository	updateIndexes()	sucesso
logic	Repository	createStructure(Path path)	sucesso
utils	ComboPopulator	popoulate methods	sucesso
utils	FileManagerTest	save, update, read, copy files	sucesso

Tabela 8 – Resumo de Testes unitários aplicados em HM^{2}AT-Admin.
O Autor (2022).

Pacote	Classe	Método	Resultado
frames	LoadingFrame	verifyRepository()	sucesso
frames	GenericModeFrame	checkModelCompleteOptions()	sucesso
frames	GenericModeFrame	loadModel(...)	sucesso
frames	GenericModeFrame	loadImplementation(...)	sucesso
frames	GenericModeFrame	loadDependency(...)	sucesso
frames	GenericModeFrame	copyToClipboad(String dep)	sucesso
frames	GenericModeFrame	loadExample(...)	sucesso
frames	GenericModeFrame	saveModel(...)	sucesso
frames	GenericModeFrame	saveImplementation(...)	sucesso
frames	GenericModeFrame	saveDependency(...)	sucesso
frames	GenericModeFrame	saveExample(...)	sucesso
frames	SpecificModeFrame	checkDeviceCompleteOptions()	sucesso
frames	SpecificModeFrame	checkImplementationCompleteOptions()	sucesso
frames	SpecificModeFrame	createAdaptiveLogic(...)	sucesso
frames	VerifyDeviceFrame	loadConfiguration(Path path)	sucesso
frames	VerifyDeviceFrame	loadClasses(Path path, String mainEntity)	sucesso
frames	VerifyDeviceFrame	loadAccessibleMethods(Class entity)	sucesso
frames	VerifyDeviceFrame	populateTable(...)	sucesso
frames	VerifyDeviceFrame	validateAdaptabilityInformation()	sucesso
utils	JARManager	jar managing functionality	sucesso
utils	JSONManager	json managing functionality	sucesso
utils	Scriptor	implementation script interpreter	sucesso

Tabela 9 – Resumo de Testes unitários aplicados em HM^{2}AT.
O Autor (2022).

A Tabela 10 apresenta a configuração dos testes para ambos os casos.

Nome	Descrição	Entrada	Resultado esperado	Resultado obtido
Teste de avaliação de adaptabilidade	O teste observa a capacidade da ferramenta de identificar os critérios de adaptabilidade no dispositivo submetido.	• Dispositivo subjacente fechado.	adaptável=*false*	adaptável=*false*
		• Dispositivo subjacente com métodos acessíveis. • Sensores e atuadores não identificados.	adaptável=*false*	adaptável=*false*
		• Dispositivo subjacente com métodos acessíveis. • Sensores e atuadores não identificados. • Mecanismo de comunicação ausente.	adaptável=*false*	adaptável=*false*
		• Dispositivo subjacente com métodos acessíveis. • Sensores e atuadores não identificados. • Mecanismo de comunicação presente.	adaptável=*true*	adaptável=*true*
Teste de especialização de uma implementação do modelo	O teste observa a capacidade da ferramenta de criar uma lógica adaptativa com base numa implementação prévia.	• Seleção de um modelo sem implementação.	não é habilitada a criação da lógica adaptativa	não é habilitada a criação da lógica adaptativa
		• O usuário apaga as informações das entidades da lógica.	Notificar o usuário. Não é permitida a criação da lógica adaptativa ou sua descarga	Notificar o usuário. Não é permitida a criação da lógica adaptativa ou sua descarga

Tabela 10 – Teste unitários particulares de HM^{2}AT.
O Autor (2022).

8.2.1.4 Especificação dos casos (fluxos)

Foram configurados casos de teste por componente para verificar o funcionamento integral das ferramentas. A Tabela 11 apresenta o fluxo de interações entre o Designer de adaptatividade e o componente HM^{2}AT-Admin. Por sua parte, as Tabelas 12 e 13 apresentam os fluxos de interação do Desenvolvedor de sistemas e as modalidades Genérica e Específica para o componente HM^{2}AT da ferramenta.

8.3 Resultados

Como mencionado anteriormente, os resultados detalhados da aplicação dos testes unitários desenvolvidos na própria ferramenta e executados pelo JUnit foram disponibilizados em formato de relatório de testes no repositório para consulta. No entanto, podemos informar que a ferramenta teve um desempenho satisfatório em todos os testes unitários aplicados.
Da mesma forma, todos os fluxos dos casos de teste especificados foram verificados, obtendo também resultados satisfatórios.

HM²AT-Admin	
Pré-condições	O usuário abre a ferramenta HM²AT-Admin.
Pós-condições	O usuário configurou o repositório da ferramenta e registrou um modelo, uma implementação, um exemplo e uma dependência.
1. Fluxo: Configurar ferramenta	1.1 O usuário entra na ferramenta e seleciona a opção de Configurações. 1.2 A ferramenta apresenta a tela para configurações com as informações sugeridas pré-selecionadas. 1.3 O usuário realiza as modificações convenientes na tela e seleciona "Salvar". 1.4 A ferramenta verifica as configurações selecionadas e cria/atualiza o repositório.
2. Fluxo: Registrar Modelo	2.1 O usuário seleciona a opção de registrar um novo modelo. 2.2 A ferramenta apresenta a tela para registro de modelo. 2.3 O usuário preenche o formulário com as informações do novo modelo e seleciona "Salvar". 2.4 A ferramenta verifica as informações do modelo e registra suas informações no repositório.
3. Fluxo: Registrar Implementação	3.1 O usuário seleciona a opção de registrar uma implementação do modelo registrado previamente. 3.2 A ferramenta apresenta a tela para registro de implementações. 3.3 O usuário preenche as implementações para o registro da implementação e seleciona "Salvar". 3.4 A ferramenta verifica as informações da implementação e registra suas informações no repositório.
4. Fluxo: Registrar Dependência	4.1 O usuário seleciona a opção de registrar uma dependência associada a uma implementação do modelo registrado previamente. 4.2 A ferramenta apresenta a tela para registro de dependências. 4.3 O usuário preenche as implementações para o registro da dependência e seleciona "Salvar". 4.4 A ferramenta verifica as informações da dependência e registra suas informações no repositório.
5. Fluxo: Registrar Exemplo	5.1 O usuário seleciona a opção de registrar um exemplo associado a uma implementação do modelo registrado previamente. 5.2 A ferramenta apresenta a tela para registro de exemplos. 5.3 O usuário preenche as implementações para o registro do exemplo e seleciona "Salvar". 5.4 A ferramenta verifica as informações do exemplo e registra suas informações no repositório.
6. Fluxo: Sair	6.1 O usuário seleciona a opção "Sair" da ferramenta. 6.2 A ferramenta atualiza os índices do repositório.

Tabela 11 – Casos de teste para HM²AT-Admin.
O Autor (2022).

HM²AT - Modalidade Genérica	
Pré-condições	O usuário abre a ferramenta HM²AT.
Pós-condições	O usuário descarregou um modelo, uma implementação, uma dependência, e um exemplo.
1. Fluxo: Selecionar Modalidade Genérica	1.1 O usuário seleciona a modalidade Genérica. 1.2 A ferramenta carrega a tela de modalidade Genérica..
2. Fluxo: Descarregar Modelo	2.1 O usuário seleciona um modelo. 2.2 A ferramenta carrega as informações do modelo selecionado. 2.3 O usuário seleciona a opção descarregar modelo e um local no sistema de arquivos para salvar o modelo. 2.4 A ferramenta salva o modelo no local indicado.
3. Fluxo: Descarregar Implementação	3.1 Desde o painel de implementações, o usuário seleciona as configurações correspondentes com uma implementação do modelo indicado previamente. 3.2 A ferramenta carrega as informações da implementação selecionada. 3.3 O usuário seleciona a opção descarregar implementação e um local no sistema de arquivos para salvar a implementação. 3.4 A ferramenta salva a implementação no local indicado.
4. Fluxo: Descarregar Dependência	4.1 O usuário seleciona a aba de Dependências. 4.2 A ferramenta apresenta a aba de Dependências. 4.3 O usuário seleciona as configurações correspondentes com uma dependência da implementação indicada previamente. 4.4 A ferramenta carrega as informações da dependência correspondente. 4.5 O usuário seleciona a opção descarregar dependência e um local no sistema de arquivos para salvar a dependência. 4.6 A ferramenta salva a dependência no local indicado.
5. Fluxo: Descarregar Exemplo	5.1 O usuário seleciona a aba de Exemplos. 5.2 A ferramenta apresenta a aba de Exemplos. 5.3 O usuário seleciona as configurações correspondentes com um exemplo da implementação indicada previamente. 5.4 A ferramenta carrega as informações do exemplo correspondente. 5.5 O usuário seleciona a opção descarregar exemplo e um local no sistema de arquivos para salvar o exemplo. 5.6 A ferramenta salva o exemplo no local indicado.
6. Fluxo: Sair	6.1 O usuário seleciona "Voltar" desde a tela da modalidade Genérica. 6.2 A ferramenta volta para a tela principal de escolha de modalidades.

Tabela 12 – Casos de teste para HM²AT-Modalidade Genérica.
O Autor (2022).

HM²AT - Modalidade Específica	
Pré-condições	O usuário abre a ferramenta HM²AT.
Pós-condições	O usuário especializou uma lógica adaptativa para um dispositivo particular.
1. Fluxo: Selecionar Modalidade Específica	1.1 O usuário seleciona a modalidade Específica. 1.2 A ferramenta carrega a tela de modalidade Específica..
2. Fluxo: Especializar Lógica Adaptativa	2.1 O usuário preenche as informações do dispositivo e seleciona a opção "Verificar". 2.2 A ferramenta carrega a tela de verificação com as informações correspondentes. Caso exista um arquivo de configurações suas informações são apresentadas na tela. 2.3 O usuário preenche as informações necessárias para o dispositivo e seleciona a opção "Avaliar". 2.4 A ferramenta avalia a adaptabilidade do dispositivo. Se o dispositivo for adaptável a ferramenta ativa a opção "Continuar", caso contrário aguarda pelas correções do usuário. 2.5 O usuário seleciona "Continuar" e preenche as informações necessárias para selecionar a implementação de um modelo de adaptatividade. 2.6 A ferramenta carrega as informações da implementação, executa o script associado e apresenta as informações correspondentes a cada entidade da lógica adaptativa. 2.7 O usuário realiza as modificações no editor de código para cada entidade. Ao finalizar seleciona a opção descarregar lógica e um local no sistema de arquivos para salvar a lógica adaptativa especializada. 2.8 A ferramenta salva a lógica adaptativa no local indicado.
3. Fluxo: Sair	6.1 O usuário seleciona "Voltar" desde a tela da modalidade Específica. 6.2 A ferramenta volta para a tela principal de escolha de modalidades.

Tabela 13 – Casos de teste para HM²AT-Modalidade Específica.
O Autor (2022).

8.4 Análise dos Resultados

Ao respeito do Meta-modelo holístico, a aplicação dos testes nos permitiu formular os seguintes argumentos:

- O Meta-modelo holístico cobre diferentes abordagens, tanto das ciências naturais quanto das áreas tecnológicas, agindo como guarda-chuva para os estudos sobre comportamento adaptativo. Em vários casos, as abordagens concretas apresentam informações mais detalhadas sobre infraestrutura, decisões de conectividade, e recursos fornecidos. Porém, isso é possível graças a que elas estão delimitadas pela perspectiva que apresentam. Da mesma forma, uma instância do meta-modelo holístico, por tanto um modelo holístico, pode referenciar e especificar recursos mais concretos e diversos, funcionando como estrutura a ser preenchida segundo a necessidade do

projetista do sistema.

- A implementação de uma instância do meta-modelo holístico foi possível com poucos recursos e técnicas de programação básicas. Em particular, o uso de ferramentas como Papyrus e o aplicativo GenMyModel [4] permitem inclusive a geração automática de código em diversas linguagens a partir do modelo UML.

- O meta-modelo holístico permitiu a representação de meta-adaptatividade como uma segunda instância do meta-modelo de adaptatividade, mantendo de maneira transparente a relação teórica entre os conceitos e sua passagem para o sistema computacional.

- Os exemplos de aplicação demonstram a versatilidade de cenários nos quais pode ser instanciado o meta-modelo holístico, e coloca em evidência a vantagem de ser uma visão agnóstica de abordagem, pois não compromete as decisões de projeto e implementação do caso de estudo.

A respeito dos testes realizados com a ferramenta podemos realizar as seguintes observações:

- Ao trabalhar com arquivos do tipo JAR, a ferramenta depende da correta estruturação e declaração do manifesto para ter acesso às entidades nele contidas.

- A estrutura dos arquivos de configurações não é verificada, por tanto a ferramenta é vulnerável aos formatos utilizados. A criação de verificadores para este tipo de recurso pode ser um ponto de maior flexibilidade para a ferramenta, por tanto, esse desenvolvimento é proposto como trabalho futuro.

- Existe um delay na carga dos componentes devido à atualização dos índices, porém essa demora é compensada na velocidade transacional das funcionalidades dos componentes.

- A diferente diagramação e estilo dos componentes, HM^{2}AT e HM^{2}AT-Admin, gera uma certa discordância para o usuário que usa ambos componentes. Porém, é esperado que os perfis de usuário sejam diferentes a maioria das vezes, pelo que esse contraste afeta um mínimo do público alvo da ferramenta.

[4] https://app.genmymodel.com/

9 CONTRIBUIÇÕES E CONCLUSÕES

Neste capítulo são apresentadas as contribuições realizadas ao longo da evolução do projeto de pesquisa, essas contribuições são divididas em três categorias: contribuições científicas, contribuições técnicas e tecnológicas, e publicações. Da mesma maneira, são detalhadas as conclusões obtidas a partir do desenvolvimento deste trabalho e expostas algumas ideias de trabalhos futuros.

9.1 Contribuições Científicas

Nesta seção são detalhadas as contribuições científicas realizadas como resultado do desenvolvimento deste trabalho. Para uma maior organização, as contribuições são agrupadas em: (a) contribuições de suporte à pesquisa, e (b) contribuições à Teoria de Adaptatividade. Os documentos elaborados detalhados nesta seção estão disponíveis na página de "Contribuições", seção "Contribuições Científicas" do repositório que acompanha o trabalho [1].

9.1.1 Suporte à pesquisa

As contribuições de suporte à pesquisa procuram criar recursos que facilitem as atividades dos pesquisadores, tanto no contexto de tarefas de pesquisa em geral quanto no assunto de pesquisas sobre adaptatividade.

- *Revisão da literatura sobre Métodos e Metodologias para pesquisas interdisciplinares*
  Particularmente no caso de assuntos de pesquisa de natureza multidisciplinar, o levantamento de informações deve ser realizado considerando as terminologias e conceitos das diferentes áreas de interesse. No entanto, é necessário desenvolver uma estratégia que ajude a equilibrar o foco da pesquisa e a quantidade de recursos investidos no levantamento de informações. Elaboramos uma revisão básica da literatura que procura esclarecer os conceitos básicos e apresentar a bibliografia mais referenciadas sobre o assunto.

- *Manual para realização Estudos Sistemáticos*
  Um enfoque particular do levantamento de informações é a elaboração de revisões sistemáticas. Existem uma série de vantagens ao elaborar este tipo de análise, alguns deles são: organizam o foco da revisão, fornecem dados comprováveis que falam sobre o tema de pesquisa e que podem ajudar na tomada de decisões e na percepção do estado da arte do assunto, e são reproduzíveis e extensíveis devido ao rigor exigido para sua elaboração.

[1] https://sites.google.com/view/rosalia-caya-thesis-document/home

Existem diversas fontes que fornecem informações detalhadas e acompanham o pesquisador no processo de realização da revisão. No entanto, consideramos que seria útil desenvolver uma guia rápida para apresentar as técnicas de estudos sistemáticos para pesquisadores. Assim, elaboramos um manual curto e prático para a elaboração de alguns dos estudos sistemáticos mais conhecidos, que é acompanhado de bibliografia básica e especializada. Em particular, são detalhados os passos, ferramentas de suporte e exemplos para Revisões Sistemáticas e Mapeamentos Sistemáticos. Da mesma maneira, apresentamos um exemplo da aplicação dessas técnicas no contexto deste projeto.

- *Manual para aplicação e análise de indicadores bibliométricos*
  A aplicação de índices bibliométricos é uma das análises que podem se beneficiar das revisões sistemáticas. No nosso trabalho a aplicação de bibliometria permitiu a descoberta de novas abordagens, novas palavras-chave e uma análise da evolução do assunto em estudo. Assim mesmo, identificamos muitos pontos de aplicação de bibliometria que podem ter resultados interessantes. Dessa maneira, a bibliometria auxilia o pesquisador a ter uma visão diferente sobre a atividade de pesquisa e sobre o trabalho de outros pesquisadores, no lugar de observar unicamente os dados científicos. Com a finalidade de resumir as informações correspondentes aos indicadores e sua aplicação, elaboramos tanto um manual prático quanto documentamos a aplicação dos indicadores para um caso particular.

- *Meta-Knowledge para adaptatividade*
  O documento é um *toolbox* de informações de pesquisa relacionada à adaptatividade e contém informações como: veículos de publicação, projetos de pequisa, ferramentas desenvolvidas e disponíveis, principais palavras-chave da área, catálogo de pesquisadores, principal bibliografia, áreas relacionadas e índices bibliométricos.

- *Bibliografia categorizada por assunto*
  Desenvolvemos uma bibliografia categorizada segundo os diferentes focos que fazem parte do nosso trabalho de pesquisa. A finalidade dessa bibliografia é apresentar uma ferramenta que permitia o pesquisador encontrar trabalhos científicos alinhados ao seu interesse, de qualidade, mas em muita menor quantidade que fazendo uma consulta em bases de dados de pesquisa. O intuito da bibliografia categorizada é fornecer um ponto de partida para pesquisas relacionadas. Assim mesmo, esperamos que esta ferramenta seja reutilizada e enriquecida para incrementar o seu valor.

9.1.2 Teoria da Adaptatividade

As contribuições listadas nesta seção centram-se no estudo e análise do comportamento adaptativo como objeto de pesquisa.

- *Condições necessárias e suficientes para adaptatividade*
  O estudo do comportamento adaptativo usando diferentes abordagens nos permite a identificação dos elementos envolvidos. Partindo da base de uma descrição clara, precisa e o mais formal possível, podemos abstrair as condições necessárias e suficientes para identificar como adaptativo um comportamento que gerencia mudanças. Tais condições são especificadas via linguagens formais de maneira que possam ser utilizadas como pré-condições para a migração de dispositivos adaptáveis para dispositivos adaptativos. Esta contribuição está contida na própria elaboração do meta-modelo holístico na Seção 6.3.

- *Meta-modelo Holístico de Adaptatividade*
  Uma das maiores contribuições do presente trabalho é a elaboração do Meta-modelo de Adaptatividade com abordagem holística, detalhada no Capítulo 6 deste documento. Esta abordagem permite identificar, de maneira geral, os elementos envolvidos no comportamento adaptativo, as interações entre eles, suas responsabilidades, e sua dinâmica principal. Desta maneira é possível identificar claramente os pontos de contato adequados para a aplicação eficiente de técnicas, mecanismos, estratégias e métodos das diferentes áreas. Esta visão holística visa fornecer uma ferramenta para a superação de problemas de terminologia e abordagens de domínio com uma homologação de termos correspondente.

- *Transformação de dispositivos Adaptáveis*
  O meta-modelo adaptativo explicita os componentes e mecanismos necessários para realizar comportamentos adaptativos, assim, com base nele é possível identificar sistemas com potencial para ser transformados em sistemas adaptativos adicionando as peças faltantes. O processo passo a passo descrito na Seção 6.4 deste documento permite acompanhar uma ordem simples que garante a construção de sistemas adaptativos preservando a lógica e dependências conceituais. A partir desse processo é possível analisar questões como compatibilidade, garantias de funcionalidade, e propriedades de propagação. Como demonstrado no Capítulo 7 a aplicação deste processo é independente do nível de abstração, assim, o mesmo processo guia tanto o desenvolvimento de sistemas adaptativos quanto sistemas meta-adaptativos.

- *Coleção e homologação de termos relacionados com adaptatividade*
  O comportamento adaptativo tem sido observado há muito tempo na comunidade científica desde diversas áreas de interesse, e, em consequência, existem diferentes abordagens, nomenclaturas e abstrações utilizadas para este fenômeno. Muitas vezes essas nomenclaturas estão consagradas dentro das áreas, sub-áreas e campos de pesquisa, onde são compreendidas com clareza. No entanto, quando se sai desses limites a terminologia torna-se ambígua, pouco explícita, ou até confusa. Nesses casos, o uso de um dicionário ajuda a compreensão e cooperação de diferentes

áreas que observam o mesmo fenômeno. Assim, elaboramos um dicionário básico contendo os termos relacionados a adaptatividade com os quais interagimos ao longo do desenvolvimento deste trabalho. O dicionário está disponível no repositório do trabalho mencionado anteriormente, ele tem por objetivo ser a base para um trabalho contínuo da comunidade científica.

9.2 Contribuições Técnicas e Tecnológicas

As contribuições citadas nesta seção também se encontram disponíveis na página de "Contribuições", seção "Contribuições Técnicas" e seção "Contribuições Tecnológicas" correspondentemente, do repositório que acompanha o trabalho [2].

9.2.1 Contribuições Técnicas adicionais

A maioria das produções técnicas já foram mencionadas como parte das contribuições, no entanto, existem alguns produtos técnicos que fazem parte da solução e podem ser utilizados para extensão da plataforma. Alguns deles são:

- *Especificação de Casos de Uso da ferramenta HM^{2}T*
  Documento que detalha as funcionalidades principais da ferramenta HM^{2}T assim como os usuários que interagem com a plataforma em cada um dos modos implementados.

- *Criação do Manual Rápido de Usuário da ferramenta HM^{2}T*
  É um documento autodescritivo que permite que o usuário possa criar, com base nele, novos componentes que definem mecanismos adaptativos. Se o componente já existe, este documento permite observar quais os requisitos que deve cumprir para ser importado para a plataforma.

- *Criação de documentação para incorporação de novas entradas no repositório bibliográfico*
  Documento simples e descritivo que permite guiar o usuário na incorporação de novas referências bibliográficas no repositório disponibilizado. O documento também fornece assistência básica para a categorização da fonte.

- *Elaboração de vídeos de demostração de uso da ferramenta HM^{2}T*
  Os vídeos apresentam casos de aplicação de cada funcionalidade principal da ferramenta, assim como a integração da lógica adaptativa obtida da HM^{2}T num projeto de desenvolvimento terceiro.

[2] https://sites.google.com/view/rosalia-caya-thesis-document/home

9.2.2 Contribuições Tecnológicas

Os elementos a seguir são sub-produtos de desenvolvimento da solução proposta, mas podem ser reutilizados, tanto para aplicação direta quanto para modificações. Todos esses produtos tecnológicos estão disponíveis no repositório associado a este trabalho mencionado anteriormente.

- *Ferramenta para agilização de bibliografias em LaTeX*
  A ferramenta, Sum&CiteBib, auxilia na criação blocos de citações em LaTeX a partir de um diretório contendo arquivos de bibliografias parciais (.bib) e cria uma bibliografia totalizada.

- *Funções de Análise bibliométrico*
  Consiste em um grupo de funções em Python que processam informações bibliométricas e realizam gráficos ao respeito de histórico de publicações e distribuição de áreas de estudo das publicações submetidas.

- *Autômato Adaptativo sob a abordagem do Meta-Modelo Holístico*
  Desenvolvido como parte da prova de conceito, implementa tanto o dispositivo subjacente quando um modelo de lógica adaptativa que pode ser reutilizado para diferentes componentes.

- *Modelo Holístico de Adaptatividade*
  Modelo que instancia o Meta-Modelo Holístico de Adaptatividade. O modelo especifica a lógica adaptativa e seus componentes detalhadamente. O recurso está disponível em UML 2.0 e pode ser utilizado como entrada para geração de código em várias plataformas.

- *Formatos de configurações para o* HM^{2}AT
  Os documentos expõem os diferentes formatos de arquivos .json e de script requeridos para o funcionamento da ferramenta.

- HM^{2}AT
  Uma ferramenta que acompanha o projeto e desenvolvimento de lógicas adaptativas especializadas para diferentes dispositivos subjacentes. A ferramenta também facilita o acesso a recursos genéricos relacionados ao meta-modelo holístico de adaptatividade.

9.3 Publicações

Nesta seção estão inclusos os trabalhos e publicações científicas associadas à elaboração teórica e prática do presente trabalho.

Literature review of formal testing on Adaptive Systems

Submetido e apresentado no *X Workshop de Tecnologias Adaptativas* (WTA2017) em janeiro de 2017. Este trabalho visa explorar o uso de técnicas formais para testes (verificação e validação) de software com comportamento adaptativo. São apresentadas as diferentes linguagens e modelos usados para desenvolver e descrever testes em condições de mudança dinâmica. Assim mesmo, são exploradas as características monitoradas nos casos de sistemas adaptativos. Como resultado deste trabalho, foi possível pensar em usar um enfoque misto para o design de testes, com definição de zonas no espaço de estados do sistema e selecionando algumas características para monitorar as métricas correspondentes. Para a implementação e geração dos casos de testes a proposta mais atraente é a de testes baseados em modelos devido a que parte do projeto é o desenvolvimento de tal modelo.

An holistic vision on Adaptivity

Trabalho apresentado no *VI Workshop de Pós-graduação de Engenharia de Computação* (WPGEC2017) em novembro de 2017. O artigo apresenta uma abordagem holística para análise de comportamentos adaptativos. São detalhados 3 enfoques tecnológicos que estudam os comportamentos adaptativos: Cibernética Computacional, Sistemas Complexos, e Computação Autonômica, suas respectivas características e modelos. Assim mesmo, são descritas as características principais tanto do comportamento adaptativo quanto dos problemas que se beneficiam de abordagens que consideram adaptatividade no momento de planejar uma proposta de solução.

Adaptivity: One phenomenom, multiple perspectives of study

Apresentado no *XI Workshop de Tecnologia Adaptativa* (WTA2018) realizado em fevereiro de 2018. O trabalho aprofunda a pesquisa começada no artigo de WPGEC2017 de criar uma abordagem holística para a análise e caracterização da adaptatividade. O objetivo é criar uma base teórica que permita identificar analogias e equivalências entre abordagens desenvolvidos em diferentes áreas do conhecimento para trabalhar problemas com comportamentos adaptativos. Dessa maneira é possível resgatar características da adaptatividade observadas por umas áreas e ocultas para outras, e, sobretudo, compartilhar o conhecimento desenvolvido em cada domínio (metodologias, técnicas, ferramentas, propostas de solução). No presente trabalho são apresentadas quatro abordagens computacionais: Sistemas Self-*, Cibernética, Sistemas Complexos e Computação Autonômica. Em cada caso é feita a descrição de cada enfoque, sua relação com adaptatividade, terminologia básica da área correspondente e os modelos básicos e representativos para facilitar a percepção gráfica de analogias. Finalmente, é apresentada a visão holística que descreve as condições necessárias, as propriedades dos sistemas adaptativos e o tipo de problema que se espera resolver de maneira mais confortável com o auxílio das técnicas adaptativas.

A Bibliometric review about adaptivity

Apresentado no *Second Workshop of Adaptive Technology* (WAT2018) como parte do *9th International Conference on Ambient Systems, Networks and Technologies (ANT-2018)* em maio de 2018. O trabalho apresenta um estudo bibliográfico sobre o comportamento adaptativo nas diferentes áreas do conhecimento. Usando as técnicas indicadas pela Bibliometria são trabalhadas duas bases de dados de trabalhos científicos: Web of Science e Scopus. Com um processo sistemático, documentado e disponibilizado via um site, são extraídas informações sobre a evolução e frequência de publicações sobre adaptatividade, dentro e fora das áreas tecnológicas. Da mesma maneira, são identificadas as áreas de maior interesse, os tipos de publicações mais frequentes, e os intervalos de tempo significativos, todos eles parâmetros para a definição de uma busca com filtros especializados sobre adaptatividade necessários para a realização de uma revisão sistemática. Além disso, são coletados termos relacionados com adaptatividade, conforme empregados nas publicações originárias de diferentes áreas, os veículos mais utilizados e as instituições com maior índice de publicações. Como resultado deste trabalho, foram obtidos indicadores específicos da frequência e do tipo de trabalhos científicos desenvolvidos sobre o tema de adaptatividade. Assim mesmo, foram coletados parâmetros que permitem configurar buscas especializadas, revisões bibliográficas e uma melhor procura de trabalhos para poder situar melhor nossa pesquisa na comunidade científica.

A Conceptual Framework for characterizing Self-modifying Systems

Artigo colaborativo desenvolvido no LTA com a finalidade de analisar as taxonomias que descrevem os sistemas automodificáveis e fornecer uma taxonomia que permita sua clarificação de maneira mais geral. O artigo segue a abordagem 5W1H para estabelecer os principais critérios de classificação, assim como fornece um instrumento de comparação entre abordagens relacionadas. O artigo foi apresentado para o CIBSE2020.

Adaptive Systems and Soft-Computing: an holistic model to inspire their synergy

Artigo apresentando para o periódico Soft-Computing da Springer contendo o modelo holístico de adaptatividade e explorando a relação com os métodos e técnicas de Soft-Computing. O artigo também expõe o potencial sinérgico entre ambas áreas.

9.4 Conclusões

Considerando as problemáticas apresentadas no Capítulo 1 é possível verificar que o problema de visibilidade científica decorrente da individualização de abordagens pode ser mitigado, pelo menos, desde duas iniciativas: a primeira é através da elaboração do que

vamos denominar como “mineração de conhecimento”, e a segunda é através da construção de uma base teórica unificada para a análise e desenvolvimento de Sistemas Adaptativos.

Por um lado, observamos que a mineração de conhecimentos sobre adaptatividade está intimamente relacionada com os conceitos e técnicas de Bibliometria, Revisões da Literatura e Estudos Sistemáticos. Assim, os resultados dessas pesquisas dependem da descrição clara e detalhada dos dados da pesquisa, seu foco, recursos utilizados, técnicas aplicadas e seus metadados associados. Por essa razão é importante enfatizar que escolha de termos precisos para descrever os trabalhos científicos é uma tarefa de grande impacto tanto para a visibilidade do trabalho em particular quanto para sua contribuição nas análises de literatura sobre uma área ou assunto.

Da mesma maneira, observamos que ao desenvolver pesquisas de mineração de conhecimentos é necessário que os pesquisadores exercitem habilidades, tanto científicas quanto técnicas, que permitam superar as dificuldades descritas. No contexto científico é importante que os pesquisadores se familiarizem com os conceitos gerais que regem a pesquisa, tanto no que se refere à elaboração e desenvolvimento de revisões científicas e bibliometria quanto no objeto de estudo. Particularmente, ao trabalhar com comportamento adaptativo, percebemos que essa familiaridade foi responsável por criar sensibilidade com suas características, permitindo o reconhecimento de padrões inclusive em contextos distantes e pouco claros. No contexto tecnológico, percebemos que o estudo de técnicas bibliométricas e revisões sistemáticas não é um recurso oferecido de maneira cotidiana na formação de pesquisadores das áreas tecnológicas. No entanto, em geral, observamos que existe uma tendência para incorporar essas técnicas no toolbox científico através do desenvolvimento de ferramentas de suporte para suas atividades e a criação de novos veículos e linhas de publicação para comunicação dos seus resultados.

Por outro lado, a elaboração de uma base unificada sobre adaptatividade significa a convergência de recursos tecnológicos e científicos disponíveis sobre o comportamento adaptativo. A disponibilidade de uma base unificada sobre adaptatividade pode se traduzir no empoderamento de pesquisas científicas, possibilitar a construção de tecnologias mais eficientes e elaboração de respostas inovadoras para assuntos que, de outro modo, poderiam ficar temporalmente fora de alcance. No entanto, ao analisar as dimensões desta tarefa podemos afirmar que a construção dessa teoria depende de um esforço comunitário, sendo necessária a colaboração e compromisso dos pesquisadores assim como sua participação ativa para dar continuidade ao processo. Assim, um ponto-chave neste desafio é a criação de um senso de comunidade: a identificação dos próprios pesquisadores como integrantes da mesma comunidade de pesquisa, na qual cada um tem o potencial de contribuir desde diferentes perspectivas segundo suas áreas de origem.

Como primeiro passo para a construção dessa base unificada elaboramos o Meta-

modelo Holístico de Adaptatividade, cujas provas de conceito permitiram analisar com maior detalhe o potencial e versatilidade da proposta. De maneira particular, sua implementação nos permitiu atingir um dos objetivos do trabalho: a incorporação de sistemas legados no desenvolvimento de sistemas adaptativos. O trabalho com sistemas desenvolvidos por terceiros, que englobam decisões de design e implementação alheias, nos deixou duas reflexões. A primeira diz sobre a inclusão deste tipo de sistemas nas propostas de desenvolvimento de sistemas adaptativos, como apontamos no Capítulo 4, dos 164 trabalhos analisados só o 2% menciona a possibilidade de trabalho com sistemas legados. Esse fator contribui no desenvolvimento de abordagens de criação integral ao invés da elaboração de propostas que permitam a integração, reutilização e transformação de sistemas adaptáveis em adaptativos. A segunda reflexão se refere à dificuldade de acesso a ferramentas desenvolvidas como parte de projetos científicos. Dos estudos analisados, o 26% expõe algum recurso tecnológico ou ferramenta para concretizar a contribuição realizada na pesquisa, porém aqueles trabalhos que disponibilizam esses recursos são minoria, representando só o 6% do total. Ainda menor é a quantidade de trabalhos que, de fato, mantém esses recursos disponíveis, de livre acesso e em repositórios existentes, chegando a ser o 3% dos trabalhos. Assim, é necessário reforçar a adoção do princípio de reprodutibilidade nos trabalhos científicos e estimular a divulgação dos recursos tecnológicos desenvolvidos (instrumentos, ferramentas, especificações e inclusive subprodutos) nas pesquisas com o intuito de recriar provas e analisar os resultados obtidos, e, simultaneamente, suportar o trabalho contínuo sobre abordagens estabelecidas para desenvolver contribuições complementares e mais aprofundadas.

Finalmente, o desenvolvimento deste trabalho permitiu identificar que o maior potencial do estudo de comportamentos adaptativos está associado à sinergia de recursos tecnológicos e parcerias científicas que possibilita para a resolução de problemas. A adaptatividade viabiliza a própria adaptação dos recursos computacionais e científicos, permite sua modificação, abrindo portas para novas abordagens, estabelecendo pontes e conexões que antes eram difíceis de transitar, tudo com base num processo familiar e intuitivo. Em resumo, o maior potencial da adaptatividade está na reinterpretação de complexidade e incerteza como novidade, tanto no nível teórico quanto prático, permitindo a reintrodução de conceitos como mudança, flexibilidade, e aprendizagem em contextos que tinham permanecido rígidos. Essa adaptatividade conceitual tem como resultado uma explosão combinatória de possibilidades sinérgicas que impulsionam o avanço das ciências em várias direções.

9.5 Trabalhos Futuros

Nesta seção mencionamos alguns dos trabalhos futuros que podem ser desenvolvidos a partir das contribuições realizadas neste trabalho. As propostas de trabalhos futuros

estão agrupadas em: trabalhos com revisões sistemáticas, trabalhos com o meta-modelo e trabalhos com a ferramenta.

9.5.1 Trabalho com Revisões Sistemáticas

Como comprovamos no Capítulo 1, as pesquisas relacionadas com adaptatividade representam um grande volumem de trabalhos pertencentes a uma diversidade de áreas e objetos de estudo. Portanto, é possível analisar essa bibliografia com diferentes objetivos e nível de detalhe. Por exemplo, é possível realizar uma revisão sistemática para obter o universo de trabalhos cujo objetivo é o uso de comportamento adaptativo como viabilizador da personalização e individualização de aprendizado (no contexto de educação).

Assim, possíveis trabalhos futuros são:

- *Revisão sistemática sobre trabalhos com abordagem de evolução de software.* Revisão sistemática sobre os trabalhos acadêmicos que proponham mecanismos para o exercício de adaptatividade em dispositivos já existentes, com o intuito de aproveitar o seu potencial.

- *Exploração e construção de meta-research em adaptatividade.* Aplicação de índices bibliométricos especializados para identificação de informações relacionadas com meta-knowledge tanto do assunto quanto das pesquisas sendo realizadas(meta-pesquisa).

- *Estudo bibliométrico sobre palavras-chave e assuntos emergentes em adaptatividade* Usando técnicas de bibliometria é possível realizar um levantamento e uma análise das palavras-chave mais usadas pelas principais áreas nas quais se desenvolvem pesquisas sobre adaptatividade. Com essas palavras-chave é possível analisar os tópicos científicos emergentes para o assunto com índices, tais como o *Latent Dirichlet Allocation (LDA)* (MCLEVEY; MCILROY-YOUNG, 2017) com processamento de linguagem natural com ferramentas apropriadas, por exemplo, *metaknowledge.*

- Elaboração de roteiro de pesquisa entre assuntos relacionados.

9.5.2 Trabalhos com o meta-modelo

Os seguintes elementos são ideias centrais para o desenvolvimento de trabalhos de pesquisa com base no meta-modelo holístico de adaptatividade.

- *Taxonomia extensa e detalhada das tecnologias adaptativas*
  Artigo que apresenta a taxonomia usada no trabalho e examina tanto os critérios conceituais quanto os modelos, técnicas, e tecnologias usadas para o desenvolvimento de software com comportamento adaptativo.

- *Implementação de testes baseados em modelos*
  A partir do modelo holístico é possível pesquisar a aplicação de técnicas da área de Testes baseados em modelos que descobrir os critérios de interesse e as técnicas apropriadas para assistir na avaliação de sistemas com comportamento adaptativo. Da mesma maneira, é interessante pesquisar sobre o desenvolvimento de benchmarks e testbeds associados à adaptatividade.

- *Ampliação do escopo de comparações do modelo*
  Realizar a aplicação do instrumento para a comparação do modelo holístico com outras abordagens que descrevem comportamentos adaptativos. Por questões de tempo diversas abordagens não foram analisadas neste trabalho, porém são muito recomendadas para uma próxima etapa, por exemplo: Revisão de Crenças, abordagens das áreas de Sistemas Dinâmicos, Aprendizagem de máquina, e algumas técnicas de Soft-Computing.

- *Experimentos com desenvolvedores sobre a facilidade de compreensão e aplicação do modelo holístico*
  Aplicação de técnicas de engenharia de software para analisar a usabilidade nas implementações do modelo holístico, assim como elaboração/seleção de critérios que permitam medir o nível de complexidade conceitual que requer planejar e implementar uma instância do modelo.

- *Desenvolvimento de técnicas para verificação de compatibilidade entre modelos e meta-modelos.*
  O trabalho com modelos em diferentes níveis de abstração implica que devem ser desenvolvidas técnicas formais que dirijam os trabalhos de verificação e validação desses recursos de maneira individual, assim como a compatibilidade entre níveis.

- *Simulação de diferentes paradigmas com base no meta-modelo holístico*
  Com base no meta-modelo holístico é possível recriar implementações dos outros paradigmas que modelam o comportamento adaptativo, fornecendo um recurso acessível para a incorporação de adaptatividade em diferentes sub-áreas e campos de conhecimento.

9.5.3 Trabalhos com a ferramenta

Existe diversidade de oportunidades para enriquecer e melhorar a ferramenta dependendo do foco e nível de pesquisa a ser desenvolvida, a seguir fornecemos algumas sugestões interessantes como trabalhos futuros com foco na ferramenta.

- Plataforma online: O funcionamento total da plataforma será desenvolvido online. As principais razões para escolher essa abordagem são:

(i) permite maior independência da configuração tecnológica local e um maior acesso a recursos. Esta configuração não restringe a possibilidade de oferecer uma versão "lite"que suporte alguns casos;

(ii) é acessível em qualquer computador que cumpra os requisitos mínimos para o funcionamento da plataforma online;

(iii) várias das implementações do DBC, como os sistemas multiagentes ou software baseado em serviços, já oferecem plataformas que estão equipadas com técnicas, métodos e estratégias que suportam e facilitam várias atividades de comunicação, seleção e avaliação que são complementares ao assunto de estudo deste trabalho.

- Aplicação de testes de usabilidade com um conjunto de usuário diverso para verificar a facilidade de uso da ferramenta.

- *Feedback para a ferramenta*
  A camada adaptativa pode conter mecanismos que armazenam informações correspondentes ao desempenho do componente adaptativo para realizar um feedback posteriormente na plataforma online sobre o funcionamento.

REFERÊNCIAS

ABBAS, N.; ANDERSSON, J.; LÖWE, W. **Towards Autonomic Software Product Lines (ASPL) - A Technical Report**. [*S.l.*], 2011. 20 p.

ABEYWICKRAMA, D. B.; OVASKA, E. A survey of autonomic computing methods in digital service ecosystems. **Serv. Oriented Comput. Appl.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 11, n. 1, p. 1–31, mar. 2017. ISSN 1863-2386. Disponível em: https://doi.org/10.1007/s11761-016-0203-8.

ABUSETA, Y.; SWESI, K. Design patterns for self adaptive systems engineering. **CoRR**, abs/1508.01330, 2015. Disponível em: http://arxiv.org/abs/1508.01330.

AKEN, J. E. v. Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules. **Journal of Management Studies**, v. 41, n. 2, p. 219–246, 2004. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6486.2004.00430.x.

ALEGRE, U.; AUGUSTO, J. C.; CLARK, T. Engineering context-aware systems and applications: A survey. **Journal of Systems and Software**, v. 117, n. Supplement C, p. 55 – 83, 2016. ISSN 0164-1212. Disponível em: http://www.sciencedirect.com/science/article/pii/S0164121216000467.

ALTURKI, A.; BANDARA, W.; GABLE, G. G. Design science research and the core of information systems. *In*: PEFFERS, K.; ROTHENBERGER, M.; KUECHLER, B. (ed.). **Design Science Research in Information Systems. Advances in Theory and Practice**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 309–327. ISBN 978-3-642-29863-9.

AMARAL, J. N. **About Computing Science Research Methodology. With significant contributions from Michael Buro, Renee Elio, Jim Hoover, Ioanis Nikolaidis, Mohammad Salavatipour, Lorna Stewart, and Ken Wong**. 2006. https://webdocs.cs.ualberta.ca/~c603/readings/research-methods.pdf. Accessed: 16-03-2021.

AMOUI, M. *et al.* Achieving dynamic adaptation via management and interpretation of runtime models. **Journal of Systems and Software**, v. 85, n. 12, p. 2720–2737, 2012. ISSN 0164-1212. Self-Adaptive Systems. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121212001458.

ANDERSSON, J. *et al.* Reflecting on self-adaptive software systems. *In*: IEEE. **2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems**. [*S.l.: s.n.*]: IEEE, 2009. p. 38–47.

ANDIAPPAN, V.; WAN, Y. K. Distinguishing approach, methodology, method, procedure and technique in process systems engineering. **Clean Technologies and Environmental Policy**, v. 22, n. 3, p. 547–555, 2020. ISSN 1618-9558.

ARNOLD, R. D.; WADE, J. P. A definition of systems thinking: A systems approach. **Procedia Computer Science**, v. 44, p. 669 – 678, 2015. ISSN

1877-0509. 2015 Conference on Systems Engineering Research. Disponível em: http://www.sciencedirect.com/science/article/pii/S1877050915002860.

ASHBY, W. R. **Design for a Brain: The origin of adaptive behaviour**. [*S.l.: s.n.*]: Springer, Dordrecht, 1960. ISBN 978-0-412-20090-8.

ATKINSON, C.; KUHNE, T. Model-driven development: a metamodeling foundation. **IEEE Software**, v. 20, n. 5, p. 36–41, 2003.

BALLAGNY, C.; HAMEURLAIN, N.; BARBIER, F. Dynamic adaptive software components: The mocas approach. *In*: **Proceedings of the 5th International Conference on Soft Computing as Transdisciplinary Science and Technology**. New York, NY, USA: Association for Computing Machinery, 2008. (CSTST '08), p. 517–524. ISBN 9781605580463. Disponível em: https://doi.org/10.1145/1456223.1456327.

BALLAGNY, C.; HAMEURLAIN, N.; BARBIER, F. Mocas: A state-based component model for self-adaptation. *In*: **2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems**. [*S.l.: s.n.*], 2009. p. 206–215.

BARANDIARAN, X.; MORENO, A. Adaptivity: From metabolism to behavior. **Adaptive Behavior**, v. 16, n. 5, p. 325–344, 2008. Disponível em: https://doi.org/10.1177/1059712308093868.

BARBIER, F. *et al.* Software adaptation: Classification and a case study with state chart xml. **IEEE Software**, v. 32, n. 5, p. 68–76, Sep. 2015. ISSN 0740-7459.

BERNS, A.; GHOSH, S. Dissecting self-* properties. *In*: **2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems**. [*S.l.: s.n.*], 2009. p. 10–19. ISSN 1949-3673.

BETTINI, C. *et al.* A survey of context modelling and reasoning techniques. **Pervasive Mob. Comput.**, Elsevier Science Publishers B. V., NLD, v. 6, n. 2, p. 161–180, apr 2010. ISSN 1574-1192. Disponível em: https://doi.org/10.1016/j.pmcj.2009.06.002.

BÉZIVIN, J. Model engineering for software modernization. *In*: . [*S.l.: s.n.*], 2004. p. 4–4. ISBN 0-7695-2243-2.

BÉZIVIN, J. On the unification power of models. **Software & Systems Modeling**, v. 4, p. 171–188, 2005.

BONET, B.; GEFFNER, H. Planning as heuristic search. **Artificial Intelligence**, v. 129, n. 1, p. 5 – 33, 2001. ISSN 0004-3702. Disponível em: http://www.sciencedirect.com/science/article/pii/S0004370201001084.

BORREGO, M.; DOUGLAS, E. P.; AMELINK, C. T. Quantitative, qualitative, and mixed research methods in engineering education. **Journal of Engineering Education**, v. 98, n. 1, p. 53–66, 2009. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2009.tb01005.x.

BRADBURY, J. S. *et al.* A survey of self-management in dynamic software architecture specifications. *In*: **Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems**. New York, NY, USA: ACM, 2004. (WOSS '04), p. 28–33. ISBN 1-58113-989-6. Disponível em: http://doi.acm.org/10.1145/1075405.1075411.

BRATMAN, H.; COURT, T. The Software Factory. **Computer**, v. 8, n. 5, p. 28–37, May 1975. ISSN 0018-9162.

BREM, A.-K.; RAN, K.; PASCUAL-LEONE, A. Learning and memory. **Handbook of clinical neurology**, v. 116, p. 693–737, 2013. ISSN 0072-9752.

BROCK, J. **The Evolution of Adaptive Systems: The General Theory of Evolution**. [*S.l.: s.n.*]: Elsevier Science, 2000. ISBN 9780080542461.

BRUN, Y. *et al.* Engineering self-adaptive systems through feedback loops. *In*: CHENG, B. H. C. *et al.* (ed.). **Software Engineering for Self-Adaptive Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 48–70. ISBN 978-3-642-02161-9. Disponível em: https://doi.org/10.1007/978-3-642-02161-9_3.

BRUNI, R. *et al.* A conceptual framework for adaptation. *In*: LARA, J. de; ZISMAN, A. (ed.). **Fundamental Approaches to Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 240–254. ISBN 978-3-642-28872-2.

BRUNI, R. *et al.* A white box perspective on behavioural adaptation. *In*: NICOLA, R. D.; HENNICKER, R. (ed.). **Software, Services, and Systems: Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering**. Cham: Springer International Publishing, 2015. p. 552–581. ISBN 978-3-319-15545-6. Disponível em: https://doi.org/10.1007/978-3-319-15545-6_32.

BUCKLEY, J. *et al.* Towards a taxonomy of software change. **Journal of Software Maintenance and Evolution: Research and Practice**, v. 17, n. 5, p. 309–332, 2005. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.319.

BURNS, D. D. R. *et al.* Ant colony nest networks adapt to resource disruption. **Journal of Animal Ecology**, n/a, n. n/a, 2020. Disponível em: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2656.13198.

CAMBRIDGE ENGLISH DICTIONARY. **Adaptability**. 2017. https://dictionary.cambridge.org/us/dictionary/english/adaptability#dataset-british. Accessed: 2018-04-23.

CANTOR, C. **Why forgetting is good for your memory**. 2021. https://www.columbiapsychiatry.org/news/why-forgetting-good-your-memory. Accessed: 2021-08-23.

CARDOZO, N. *et al.* Modeling and analyzing self-adaptive systems with context petri nets. *In*: **2013 International Symposium on Theoretical Aspects of Software Engineering**. [*S.l.: s.n.*], 2013. p. 191–198.

CARNEIRO, L. E. d. S.; ALMEIDA, M. B. Design science: Representação de um campo teórico. **Informação & Sociedade: Estudos**, v. 29, n. 1, 2019. Copyright - © 2019. This work is published under http://creativecommons.org/licenses/by-nc/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2019-04-25. Disponível em: https://search.proquest.com/scholarly-journals/design-science-representaĂğĂčo-de-um-campo-teĂşrico/docview/2212830963/se-2?accountid=14643.

CAYA, R.; NETO, J. Personalização, customização, adaptatividade e adaptabilidade. *In*: **Memórias do X Workshop de Tecnologia Adaptativa - WTA 2016**. São Paulo: EPUSP, 2016. p. 52–59.

CAYA, R.; NETO, J. J. A bibliometric review about adaptivity. **Procedia Computer Science**, v. 130, p. 1114 – 1119, 2018. ISSN 1877-0509. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops. Disponível em: http://www.sciencedirect.com/science/article/pii/S1877050918305295.

CHEN, C.-C.; CRILLY, N. **From modularity to emergence: a primer on the design and science of complex systems**. Cambridge, UK, 2016. Disponível em: https://doi.org/10.17863/CAM.4503.

CHENG, B. H. C. *et al.* (ed.). **Software Engineering for Self-Adaptive Systems**. New York: Springer Berlin Heidelberg, 2009. (Lecture Notes in Computer Science). ISBN 978-3-642-02160-2.

CHENG, B. H. C. *et al.* Software engineering for self-adaptive systems: A research roadmap. *In*: CHENG, B. H. C. *et al.* (ed.). **Software Engineering for Self-Adaptive Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–26. ISBN 978-3-642-02161-9. Disponível em: https://doi.org/10.1007/978-3-642-02161-9_1.

CHENG, S.-W.; GARLAN, D. Stitch: A language for architecture-based self-adaptation. **Journal of Systems and Software**, v. 85, n. 12, p. 2860 – 2875, 2012. ISSN 0164-1212. Self-Adaptive Systems. Disponível em: http://www.sciencedirect.com/science/article/pii/S0164121212000714.

CILLIERS, P. **Complexity and postmodernism: Understanding complex systems**. [*S.l.: s.n.*]: Routledge, 2002.

CONBOY, K.; GLEASURE, R.; CULLINA, E. Agile design science research. *In*: DONNELLAN, B. *et al.* (ed.). **New Horizons in Design Science: Broadening the Research Agenda**. Cham: Springer International Publishing, 2015. (Lecture Notes in Computer Science, 9073), p. 168–180. ISBN 9783319187136.

COSTANZA, P.; HIRSCHFELD, R.; NIERSTRASZ, O. Context-oriented programming. **Journal of Object Technology**, v. 7, n. 3, p. 125–151, 2008.

CUSUMANO, M. A. The software factory: a historical interpretation. **IEEE Software**, v. 6, n. 2, p. 23–30, March 1989. ISSN 0740-7459.

DEMEYER, S. Research methods in computer science. *In*: **2011 27th IEEE International Conference on Software Maintenance (ICSM)**. Williamsburg, VA, USA: IEEE Computer Society, 2011. (ICSM '11), p. 600. ISBN 978-1-4577-0664-6.

DOBZHANSKY, T. On some fundamental concepts of darwinian biology. *In*: DOBZHANSKY, T.; HECHT, M. K.; STEERE, W. C. (ed.). **Evolutionary Biology: Volume 2**. Boston, MA: Springer US, 1968. p. 1–34. ISBN 978-1-4684-8094-8. Disponível em: https://doi.org/10.1007/978-1-4684-8094-8_1.

DOBZHANSKY, T.; DOBZHANSKY, T. **Genetics of the Evolutionary Process**. Columbia University Press, 1970. (A Columbia paperback). ISBN 9780231083065. Disponível em: https://books.google.ca/books?id=f7su1R-NYwsC.

DODIG-CRNKOVIC, G. Scientific methods in computer science. *In*: **Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia**. [*S.l.: s.n.*], 2002. p. 126–130.

DONGES, J. F. *et al.* The technosphere in earth system analysis: A coevolutionary perspective. **The Anthropocene Review**, v. 4, n. 1, p. 23–33, 2017. Disponível em: https://doi.org/10.1177/2053019616676608.

DRESCH, A.; LACERDA, D.; JÚNIOR, J. **Design Science Research: Método de Pesquisa para Avanço da Ciência e Tecnologia**. [*S.l.: s.n.*]: Bookman Editora, 2015. (Métodos de Pesquisa). ISBN 9788582602997.

DRESCH, A.; LACERDA, D. P.; MIGUEL, P. A. C. Uma análise distintiva entre o estudo de caso, a pesquisa-ação e a design science research. **Revista Brasileira de Gestão de Negócios**, scielo, v. 17, p. 1116 – 1133, 06 2015. ISSN 1806-4892. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-48922015000301116&nrm=iso.

ELLIS, B.; HERBERT, S. I. Complex adaptive systems (cas): an overview of key elements, characteristics and application to management theory. **Journal of Innovation in Health Informatics**, v. 19, n. 1, p. 33–37, 2011. ISSN 2058-4563. Disponível em: https://hijournal.bcs.org/index.php/jhi/article/view/791.

ENGLISH OXFORD DICTIONARY. **Adaptability**. 2017. https://en.oxforddictionaries.com/definition/adaptability. Accessed: 2018-04-23.

FATEHAH, M.; MEZHUYEV, V.; AL-EMRAN, M. A systematic review of metamodelling in software engineering. *In*: _____. **Recent Advances in Intelligent Systems and Smart Applications**. Cham: Springer International Publishing, 2021. p. 3–27. ISBN 978-3-030-47411-9. Disponível em: https://doi.org/10.1007/978-3-030-47411-9_1.

FERREIRA, H. J. S. L. **Adaptive Object-Modeling : Patterns, Tools and Applications**. December 2010. Tese (Doutorado) — Faculty of Engineering, University of Porto, Porto, Portugal, December 2010.

FLOCH, J. *et al.* Playing music — building context-aware and self-adaptive mobile applications. **Software: Practice and Experience**, v. 43, n. 3, p. 359–388, 2013. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2116.

FLORIAN, R. V. [**Technical Report Coneural-03-01**] **Using Bibliometrics: A guide to evaluating research performance with citation data**. [*S.l.*], 2003. Disponível em: https://coneural.org/reports/Coneural-03-01.pdf.

FORGETFULNESS - 7 types of normal memory problems. 2021. https://www.health.harvard.edu/mind-and-mood/forgetfulness-7-types-of-normal-memory-problems.

FORREST, S.; MITCHELL, M. Adaptive computation: The multidisciplinary legacy of john h. holland. **Commun. ACM**, ACM, New York, NY, USA, v. 59, n. 8, p. 58–63, jul. 2016. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/2964342.

GAINES, B. Axioms for adaptive behaviour. **International Journal of Man-Machine Studies**, v. 4, n. 2, p. 169 – 199, 1972. ISSN 0020-7373. Disponível em: http://www.sciencedirect.com/science/article/pii/S0020737372800300.

GAMUKAMA, E. A.; POPOV, O. The level of scientific methods use in computing research programs. *In*: **31st International Convention on Information and Communication Technology**. [*S.l.: s.n.*], 2008. p. 176–183. ISBN 978-953-233-039-7.

GARLAN, D. *et al.* Rainbow: Architecture-based self-adaptation with reusable infrastructure. **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 37, n. 10, p. 46–54, out. 2004. ISSN 0018-9162. Disponível em: http://dx.doi.org/10.1109/MC.2004.175.

GARLAN, D.; SCHMERL, B.; CHENG, S.-W. Software architecture-based self-adaptation. *In*: ZHANG, Y.; YANG, L. T.; DENKO, M. K. (ed.). **Autonomic Computing and Networking**. Boston, MA: Springer US, 2009. p. 31–55. ISBN 978-0-387-89828-5. Disponível em: https://doi.org/10.1007/978-0-387-89828-5_2.

GEERTS, G. L. A design science research methodology and its application to accounting information systems research. **International Journal of Accounting Information Systems**, v. 12, n. 2, p. 142–151, 2011. ISSN 1467-0895. Special Issue on Methodologies in AIS Research. Disponível em: https://www.sciencedirect.com/science/article/pii/S1467089511000200.

GEIHS, K. Self-adaptivity from different application perspectives. *In*: LEMOS, R. de *et al.* (ed.). **Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 376–392. ISBN 978-3-642-35813-5. Disponível em: https://doi.org/10.1007/978-3-642-35813-5_15.

GELL-MANN, M. Complex adaptive systems. *In*: COWAN, G. A.; PINES, D.; MELTZER, D. (ed.). **Complexity: Metaphors, Models, and Reality**. Cambridge, MA, USA: Perseus Books, 1999. p. 17–45. ISBN 0-7382-0232-0. Disponível em: http://dl.acm.org/citation.cfm?id=336600.336616.

GEROSTATHOPOULOS, I. *et al.* Meta-adaptation strategies for adaptation in cyber-physical systems. *In*: WEYNS, D.; MIRANDOLA, R.; CRNKOVIC, I. (ed.). **Software Architecture**. Cham: Springer International Publishing, 2015. p. 45–52. ISBN 978-3-319-23727-5.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: theory and practice**. [*S.l.: s.n.*]: Elsevier, 2004.

GHARAJEDAGHI, J. **Systems Thinking (Third Edition)**. Third edition. Boston: Morgan Kaufmann, 2012. ISBN 978-0-12-385915-0.

GLASS, R. L.; VESSEY, I.; RAMESH, V. Research in software engineering: an analysis of the literature. **Information and Software technology**, Elsevier, v. 44, n. 8, p. 491–506, 2002.

GOLDSTEIN, D. S.; KOPIN, I. J. Homeostatic systems, biocybernetics, and autonomic neuroscience. **Autonomic Neuroscience**, v. 208, p. 15–28, 2017. ISSN 1566-0702. Disponível em: https://www.sciencedirect.com/science/article/pii/S1566070217300711.

GORDON, D. M. **An ant colony has memories that its individual members don't have**. 2018. https://aeon.co/ideas/an-ant-colony-has-memories-that-its-individual-members-dont-have. Accessed: 2019-10-05.

GRABOWSKI, F.; STRZALKA, D. Simple, complicated and complex systems—the brief introduction. *In*: IEEE. **2008 Conference On Human System Interactions**. [*S.l.: s.n.*], 2008. p. 570–573.

GREENFIELD, J. **The Case of Software Factories**. 2004. https://msdn.microsoft.com/en-us/library/aa480032(d=printer).aspx. Accessed: 2018-05-23.

GREENFIELD, J.; SHORT, K. Software factories: Assembling applications with patterns, models, frameworks and tools. *In*: **Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications**. New York, NY, USA: ACM, 2003. (OOPSLA '03), p. 16–27. ISBN 1-58113-751-6. Disponível em: http://doi.acm.org/10.1145/949344.949348.

HALLSTEINSEN, S. *et al.* A development framework and methodology for self-adapting applications in ubiquitous computing environments. **Journal of Systems and Software**, v. 85, n. 12, p. 2840–2859, 2012. ISSN 0164-1212. Self-Adaptive Systems. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121212002245.

HASSANI, H. Research methods in computer science: The challenges and issues. **CoRR**, abs/1703.04080, 2017. Disponível em: http://arxiv.org/abs/1703.04080.

HENDERSON-SELLERS, B. Bridging metamodels and ontologies in software engineering. **Journal of Systems and Software**, v. 84, n. 2, p. 301–313, 2011. ISSN 0164-1212. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121210002906.

HEVNER, A. R.; CHATTERJEE, S. **Design Research in Information Systems: Theory and Practice**. [*S.l.: s.n.*]: Springer, 2010. (Integrated Series in Information Systems). ISBN 978-1-4419-5653-8.

HEVNER, A. R. *et al.* Design science in information systems research. **MIS Quarterly**, Management Information Systems Research Center, University of Minnesota, USA, v. 28, n. 1, p. 75–105, mar. 2004. ISSN 0276-7783. Disponível em: https://www.jstor.org/stable/25148625.

HEYLIGHEN, F. Complexity and self-organization. *In*: BATES, M.; MAACK, M. (ed.). **Encyclopedia of Library and Information Science, 3rd. edition**. United States: Taylor and Francis, 2009. ISBN 978-0-8493-9712-7. Marcia J. Bates and Mary Niles Maack.

HEYLIGHEN, F.; JOSLYN, C. Cybernetics and second-order cybernetics. **Encyclopedia of physical science & technology**, v. 4, p. 155–170, 2001.

HITCHINS, D. K. **Advanced Systems Thinking, Engineering, and Management**. Boston, MA: Artech House INC, 2003. ISBN 9781580536196.

HOFLER, D. B. Approach, method, technique a clarification. **Reading World**, Routledge, v. 23, n. 1, p. 71–72, 1983. Disponível em: https://doi.org/10.1080/19388078309557742.

HOLANDA, A. B. D. **Dicionário Aurélio da Língua Portuguesa, versão online. Adaptabilidade**. 2017. https://dicionariodoaurelio.com/adaptabilidade. Accessed: 2015-10-18.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence**. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262082136.

HOLLAND, J. H. **Hidden Order: How Adaptation Builds Complexity**. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-40793-0.

HOLLAND, J. H. **Complexity : a very short introduction**. [*S.l.: s.n.*]: Oxford University Press, 2014. (Very short introductions). ISBN 9780199662548.

HOLZ, H. J. *et al.* Research methods in computing: What are they, and how should we teach them? **SIGCSE Bull.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 4, p. 96–114, jun. 2006. ISSN 0097-8418. Disponível em: https://doi.org/10.1145/1189136.1189180.

HONG, J. yi; SUH, E. ho; KIM, S.-J. Context-aware systems: A literature review and classification. **Expert Systems with Applications**, v. 36, n. 4, p. 8509 – 8522, 2009. ISSN 0957-4174. Disponível em: http://www.sciencedirect.com/science/article/pii/S0957417408007574.

HORN, P. **Autonomic computing: IBM's Perspective on the State of Information Technology**. [*S.l.: s.n.*]: IBM, 2001.

HUEBSCHER, M. C.; MCCANN, J. A. A survey of autonomic computing—degrees, models, and applications. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 40, n. 3, p. 7:1–7:28, ago. 2008. ISSN 0360-0300. Disponível em: http://doi.acm.org/10.1145/1380584.1380585.

IBM Corporation. **An Architectural Blueprint for Autonomic Computing**. [*S.l.*], 2005.

INOJOSA, R. **Uma nova formulação algébrica para o autômato finito adaptativo de segunda ordem aplicada a um modelo de inferência indutiva**. 3 2012. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 3 2012.

JAIN, H.; SINHA, A. P.; VITHARANA, P. (ed.). **DESRIST'11: Proceedings of the 6th International Conference on Service-Oriented Perspectives in Design Science Research**, v. 6629 de **Lecture Notes in Computer Science**, (Lecture Notes in Computer Science, v. 6629). Berlin, Heidelberg: Springer-Verlag, 2011. ISBN 9783642206320.

JEUSFELD, M. A. Metamodel. *In*: _____. **Encyclopedia of Database Systems**. Boston, MA: Springer US, 2009. p. 1727–1730. ISBN 978-0-387-39940-9. Disponível em: https://doi.org/10.1007/978-0-387-39940-9_898.

JR, J. F. N.; CHEN, M.; PURDIN, T. D. Systems development in information systems research. **Journal of management information systems**, Taylor & Francis, v. 7, n. 3, p. 89–106, 1991.

KAKOUSIS, K. *et al.* Testing self-adaptive applications with simulation of context events. **Electron. Commun. Eur. Assoc. Softw. Sci. Technol.**, v. 28, 2010. ISSN 1863-2122. Disponível em: https://doi.org/10.14279/tuj.eceasst.28.394.

KELL, S. A survey of practical software adaptation techniques. **j-jucs**, v. 14, n. 13, p. 2110–2157, jul 2008.

KEPHART, J. O. Research challenges of autonomic computing. *In*: **Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.** [*S.l.: s.n.*], 2005. p. 15–22. ISSN 0270-5257.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **Computer**, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 1, p. 41–50, jan. 2003. ISSN 0018-9162. Disponível em: http://dx.doi.org/10.1109/MC.2003.1160055.

KING, T. M. *et al.* An integrated self-testing framework for autonomic computing systems. **JCP**, v. 2, p. 37–49, 2007.

KOKAR, M. M.; BACLAWSKI, K.; ERACAR, Y. A. Control theory-based foundations of self-controlling software. **IEEE Intelligent Systems**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 14, n. 3, p. 37–45, maio 1999. ISSN 1541-1672. Disponível em: http://dx.doi.org/10.1109/5254.769883.

KOUNEV, S. *et al.* **Self-Aware Computing Systems**. 1st. ed. [*S.l.: s.n.*]: Springer Publishing Company, Incorporated, 2017. ISBN 3319474723, 9783319474724.

KRAMER, J.; MAGEE, J. Self-managed systems: An architectural challenge. *In*: **2007 Future of Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2007. (FOSE '07), p. 259–268. ISBN 0-7695-2829-5. Disponível em: http://dx.doi.org/10.1109/FOSE.2007.19.

KRUPITZER, C. *et al.* A survey on engineering approaches for self-adaptive systems. **Pervasive and Mobile Computing**, v. 17, p. 184 – 206, 2015. ISSN 1574-1192. 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian. Disponível em: http://www.sciencedirect.com/science/article/pii/S157411921400162X.

KURKA, D. B.; PITT, J. The principled violation of policy: Norm flexibilization in open self-organising systems. *In*: **2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)**. Tucson, AZ, USA: IEEE, 2017. p. 33–38.

LACERDA, D. P. *et al.* Design science research: método de pesquisa para a engenharia de produção. **Gestão & Produção**, scielo, v. 20, p. 741 – 761, 00 2013. ISSN 0104-530X. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2013000400001&nrm=iso.

LADDAGA, R. **Self Adaptive Software BAA 98-12 DUE 1/27/98**. [*S.l.*], 1998.

LADDAGA, R. Self adaptive software problems and projects. *In*: **Proceedings of the Second International IEEE Workshop on Software Evolvability**. Washington, DC, USA: IEEE Computer Society, 2006. (SOFTWARE-EVOLVABILITY '06), p. 3–10. ISBN 0-7695-2698-5. Disponível em: https://doi.org/10.1109/SOFTWARE-EVOLVABILITY.2006.10.

LADDAGA, R.; ROBERTSON, P. Self adaptive software: A position paper. *In*: CITESEER. **SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems**. [*S.l.: s.n.*], 2004. v. 31, p. 19.

LALANDA, P.; MCCANN, J. A.; DIACONESCU, A. **Autonomic Computing: Principles, Design and Implementation**. [*S.l.: s.n.*]: Springer Publishing Company, Incorporated, 2013. ISBN 1447150066, 9781447150060.

LANESE, I.; BUCCHIARONE, A.; MONTESI, F. A framework for rule-based dynamic adaptation. *In*: WIRSING, M.; HOFMANN, M.; RAUSCHMAYER, A. (ed.). **Trustworthly Global Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 284–300. ISBN 978-3-642-15640-3.

LEMOS, R. de *et al.* Software engineering for self-adaptive systems: A second research roadmap. *In*: LEMOS, R. de *et al.* (ed.). **Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 1–32. ISBN 978-3-642-35813-5. Disponível em: https://doi.org/10.1007/978-3-642-35813-5_1.

LEMOS, R. de *et al.* (ed.). **Software Engineering for Self-Adaptive Systems II. International Seminar, Dadstuhl, Germany, October 24-29, 2010. Revised and Invited Papers**. New York: Springer Berlin Heidelberg, 2013. (Lecture Notes in Computer Science). ISBN 978-3-642-35812-8.

LENHARD, J.; GEIGER, M.; WIRTZ, G. On the measurement of design-time adaptability for process-based systems. *In*: **2015 IEEE Symposium on Service-Oriented System Engineering**. [*S.l.: s.n.*], 2015. p. 1–10.

LEROUX-MARTIN, P.; O'CONNOR, V. **Systems Thinking for Peacebuilding and Rule of Law: Supporting Complex Reforms in Conflict-Affected Environments**. [*S.l.*], 2017.

LEWONTIN, R. C. Adaptation. **Scientific American**, Scientific American, a division of Nature America, Inc., v. 239, n. 3, p. 212–231, 1978. ISSN 00368733, 19467087. Disponível em: http://www.jstor.org/stable/24955807.

LINTS, T. The essentials of defining adaptation. *In*: **2010 IEEE International Systems Conference**. [*S.l.: s.n.*], 2010. p. 113–116.

LIU, L.; SCHMECK, H. A roadmap towards autonomic service-oriented architectures. **International Transactions on Systems Science and Applications**, Glasgow, Xiaglow Press, Volume 2, n. 3, p. 245–255, September 2006.

MACíAS-ESCRIVá, F. D. *et al.* Self-adaptive systems: A survey of current approaches, research challenges and applications. **Expert Systems with Applications**, v. 40, n. 18, p. 7267 – 7279, 2013. ISSN 0957-4174. Disponível em: http://www.sciencedirect.com/science/article/pii/S0957417413005125.

MACKENZIE, N.; KNIPE, S. Research dilemmas: Paradigms, methods and methodology. **Issues In Educational Research**, v. 16, n. 2, p. 193–205, 2006.

MARCH, S. T.; SMITH, G. F. Design and natural science research on information technology. **Decis. Support Syst.**, Elsevier Science Publishers B. V., NLD, v. 15, n. 4, p. 251–266, dez. 1995. ISSN 0167-9236.

MARTÍNEZ-GARCÍA, M.; HERNÁNDEZ-LEMUS, E. Health systems as complex systems. **American Journal of Operations Research**, Scientific Research, v. 3, n. 1A, p. 113–126, jan. 2013. Disponível em: http://dx.doi.org/10.4236/ajor.2013.31A011.

MATURANA, H. R.; VARELA, F. J. On machines, living and otherwise. *In*: **Autopoiesis and Cognition: The Realization of the Living**. Dordrecht: Springer Netherlands, 1980. p. 77–84. ISBN 978-94-009-8947-4. Disponível em: https://doi.org/10.1007/978-94-009-8947-4_9.

MCLEVEY, J.; MCILROY-YOUNG, R. Introducing metaknowledge: Software for computational research in information science, network analysis, and science of science. **Journal of Informetrics**, v. 11, n. 1, p. 176 – 197, 2017. ISSN 1751-1577. Disponível em: http://www.sciencedirect.com/science/article/pii/S1751157716302000.

MERELLI, E.; PAOLETTI, N.; TESEI, L. Adaptability checking in complex systems. **Science of Computer Programming**, v. 115-116, p. 23–46, 2016. ISSN 0167-6423. Special Section on Foundations of Coordination Languages and Software (FOCLASA 2012) Special Section on Foundations of Coordination Languages and Software (FOCLASA 2013). Disponível em: https://www.sciencedirect.com/science/article/pii/S0167642315000659.

MERRIAM-WEBSTER DICTIONARY. **Adaptable**. 2017. https://www.merriam-webster.com/dictionary/adaptable. Accessed: 2018-04-18.

MICROSOFT DOCS. **Software Factories**. 2005. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff699235(v=pandp.10). Accessed: 2018-04-10.

MICROSOFT DOCS. **Metadados e componentes autodescritivos**. 2017. https://docs.microsoft.com/pt-br/dotnet/standard/metadata-and-self-describing-components. Accessed: 2018-01-02.

MILLICAN, P.; WOOLDRIDGE, M. Them and us: Autonomous agents in vivo and in silico. *In*: BALTAG, A.; SMETS, S. (ed.). **Johan van Benthem on Logic and Information Dynamics**. Cham: Springer International Publishing, 2014. p. 547–567. ISBN 978-3-319-06025-5. Disponível em: https://doi.org/10.1007/978-3-319-06025-5_20.

MITCHELL, M. **Complexity: A Guided Tour**. [*S.l.: s.n.*]: Oxford University Press, USA, 2009. ISBN 9780691127026.

MITCHELL, M.; NEWMAN, M. Complex systems theory and evolution. *In*: PAGEL, M. D. (ed.). **Encyclopedia of Evolution**. New York, NY, USA: Oxford University Press, 2002. p. 1–5. ISBN 9780195122008. Disponível em: https://books.google.com.br/books?id=7MV9AQAACAAJ.

NAMI, M. R.; BERTELS, K. A survey of autonomic computing systems. *In*: **Third International Conference on Autonomic and Autonomous Systems (ICAS'07)**. [*S.l.: s.n.*], 2007. p. 26–26.

NETO, J. J. Adaptive automata for context-dependent languages. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 29, n. 9, p. 115–124, set. 1994. ISSN 0362-1340. Disponível em: http://doi.acm.org/10.1145/185009.185033.

NETO, J. J. Adaptive rule-driven devices - general formulation and case study. *In*: **Revised Papers from the 6th International Conference on Implementation and Application of Automata**. London, UK, UK: Springer-Verlag, 2002. (CIAA '01), p. 234–250. ISBN 3-540-00400-9. Disponível em: http://dl.acm.org/citation.cfm?id=647268.721714.

NETO, J. J. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. **Revista IEEE América Latina**, v. 5, n. 7, p. 496–505, 2007. Disponível em: http://www.ewh.ieee.org/reg/9/etrans/ieee/issues/vol05/vol5issue7Nov.2007/5TLA7_02Neto.pdf.

NETO, J. J. Um levantamento da pesquisa em técnicas adaptativas na epusp. **Revista de Sistemas e Computação-RSC**, UNIFACS, Salvador, Fortaleza, Brazil, v. 1, n. 1, p. 23–47, 2011. ISSN 2237-2903. Disponível em: http://doi.acm.org/10.1145/185009.185033.

NETO, J. J. Simulation and adaptivity. *In*: KOGLER, J. E. J.; ADAMS, F.; PESSOA, O. J. (ed.). **Cognitive Science: Recent Advances and Recurring Problems**. [*S.l.: s.n.*]: Vernon Press, 2017, (Cognitive Science and Psychology). p. 113–150. ISBN 978-1-62273-100-8.

NETO, J. J.; BRAVO, C. Adaptive automata - a revisited proposal. *In*: CHAMPARNAUD, J.-M.; MAUREL, D. (ed.). **Implementation and Application of Automata**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 158–168. ISBN 978-3-540-44977-5.

NEWMAN, M. E. J. Resource letter cs–1: Complex systems. **American Journal of Physics**, American Association of Physics Teachers (AAPT), v. 79, n. 8, p. 800–810, Aug 2011. ISSN 1943-2909. Disponível em: http://dx.doi.org/10.1119/1.3590372.

NIELSEN, J. **Usability 101: Introduction to Usability**. [*S.l.: s.n.*]: Nielsen Norman Group. World Leaders in Research-Based User Experience, 2012. https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

NIELSEN, J. **10 Usability Heuristics for User Interface Design**. [*S.l.: s.n.*]: Nielsen Norman Group. World Leaders in Research-Based User Experience, 2020. https://www.nngroup.com/articles/ten-usability-heuristics/.

NORTHROP, L. M.; CLEMENTS, P. C. **[Whitepaper] A Framework for Software Product Line Practice, Version 5.0**. [*S.l.*], 2012.

OKANO, H.; HIRANO, T.; BALABAN, E. Learning and memory. **Proceedings of the National Academy of Sciences**, v. 97, n. 23, p. 12403–12404, 2000. Disponível em: https://www.pnas.org/doi/abs/10.1073/pnas.210381897.

OREIZY, P. *et al.* An architecture-based approach to self-adaptive software. **IEEE Intelligent Systems and their Applications**, v. 14, n. 3, p. 54–62, May 1999. ISSN 1094-7167.

PAGE, S. E. **Diversity and Complexity**. 1st. ed. Princeton, NJ, USA: Princeton University Press, 2010. ISBN 0691137676, 9780691137674.

PANDEY, A. *et al.* Towards a formal framework for hybrid planning in self-adaptation. *In*: **Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. Piscataway, NJ, USA: IEEE Press, 2017. (SEAMS '17), p. 109–115. ISBN 978-1-5386-1550-8. Disponível em: https://doi.org/10.1109/SEAMS.2017.14.

PAOLO, E. D. Autopoiesis, adaptivity, teleolog agency. **Phenomenol. Cogn. Sci**, v. 4, p. 429–452, 12 2005.

PATIKIRIKORALA, T. *et al.* A systematic survey on the design of self-adaptive software systems using control engineering approaches. *In*: **Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. Piscataway, NJ, USA: IEEE Press, 2012. (SEAMS '12), p. 33–42. ISBN 978-1-4673-1787-0. Disponível em: http://dl.acm.org/citation.cfm?id=2666795.2666803.

PEFFERS, K. *et al.* A design science research methodology for information systems research. **Journal of Management Information Systems**, Taylor & Francis, Ltd., v. 24, n. 3, p. 45–77, 2007. ISSN 07421222. Disponível em: http://www.jstor.org/stable/40398896.

PFANNEMüLLER, M. *et al.* React-ion: A model-based runtime environment for situation-aware adaptations. **ACM Trans. Auton. Adapt. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 4, dec 2021. ISSN 1556-4665. Disponível em: https://doi.org/10.1145/3487919.

PFANNEMüLLER, M. *et al.* Enhancing a communication system with adaptive behavior using react. *In*: **2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)**. [*S.l.: s.n.*], 2020. p. 228–229.

PFANNEMüLLER, M. *et al.* React: A model-based runtime environment for adapting communication systems. *In*: **2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)**. [*S.l.: s.n.*], 2020. p. 65–74.

PIMENTEL, J.; FRANCH, J.; CASTRO, J. Measuring architectural adaptability in i* models. *In*: _____. **CIbSE 2011: XIV Congreso iberoamericano en software engineering: 27/04 al 29/04, 2011 Rio de Janeiro, Brasil**. [*S.l.: s.n.*], 2011. p. 115–128. ISBN 9788580060324. Disponível em: http://hdl.handle.net/2117/18014.

PISTORI, H. **Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações**. 12 2003. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil, 12 2003.

PISTORI, H.; NETO, J. J.; PEREIRA, M. Adaptive non-deterministic decision trees: General formulation and case study. **INFOCOMP**, v. 5, n. 1, p. 35–40, 2006. ISSN 1982-3363. Disponível em: http://infocomp.dcc.ufla.br/index.php/INFOCOMP/article/view/120.

POSLAD, S. Autonomous systems and artificial life. *In*: _____. **Ubiquitous Computing**. John Wiley and Sons, Ltd, 2009. p. 317–341. ISBN 9780470779446. Disponível em: http://dx.doi.org/10.1002/9780470779446.ch10.

PRINCIPE, J. C.; EULIANO, N. R.; LEFEBVRE, W. C. **Neural and Adaptive Systems: Fundamentals Through Simulations with CD-ROM**. 1st. ed. Berlin, Germany: John Wiley & Sons, Inc., 1999. ISBN 0471351679.

RAHMAN, M. *et al.* A taxonomy and survey on autonomic management of applications in grid computing environments. **Concurrency and Computation: Practice and Experience**, v. 23, n. 16, p. 1990–2019, 2011. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1734.

RAIBULET, C. Towards a taxonomy for the evaluation of self-* software. *In*: **2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)**. [*S.l.: s.n.*], 2018. p. 22–23.

RAMESH, V.; GLASS, R. L.; VESSEY, I. Research in computer science: an empirical study. **Journal of Systems and Software**, v. 70, n. 1, p. 165–176, 2004. ISSN 0164-1212. Disponível em: https://www.sciencedirect.com/science/article/pii/S0164121203000153.

RAMIREZ, A. J. *et al.* Plato: a genetic algorithm approach to run-time reconfiguration in autonomic computing systems. **Cluster Computing**, v. 14, n. 3, p. 229–244, Sep 2011. ISSN 1573-7543. Disponível em: https://doi.org/10.1007/s10586-010-0122-y.

REEVE, H. K.; SHERMAN, P. W. Adaptation and the goals of evolutionary research. **The Quarterly Review of Biology**, University of Chicago Press, v. 68, n. 1, p. 1–32, 1993. ISSN 00335770, 15397718. Disponível em: http://www.jstor.org/stable/2832133.

ROBERTSON, P.; LADDAGA, R.; SHROBE, H. Introduction: the first international workshop on self-adaptive software. *In*: **Self-Adaptive Software**. [*S.l.: s.n.*]: Springer, 2000. p. 1–10.

RODRIGUES, D. D. Design science research como caminho metodológico para disciplinas e projetos de design da informação | design science research as methodological path for information design subjects and projects. **InfoDesign - Revista Brasileira de Design da Informação**, v. 15, n. 1, p. 111–124, ago. 2018. Disponível em: https://infodesign.org.br/infodesign/article/view/564.

ROHR, M. *et al.* A Classification Scheme for Self-adaptation Research. *In*: **Proceedings of the International Conference on Self-Organization and Autonomous Systems In Computing and Communications (SOAS'2006)**. [*S.l.: s.n.*], 2006.

ROMME, A. G. L. Making a difference: Organization as design. **Organization Science**, v. 14, n. 5, p. 558–573, 2003. Disponível em: https://doi.org/10.1287/orsc.14.5.558.16769.

ROSOL, C.; NELSON, S.; RENN, J. Introduction: In the machine room of the anthropocene. **The Anthropocene Review**, v. 4, n. 1, p. 2–8, 2017. Disponível em: https://doi.org/10.1177/2053019617701165.

ROTHENBERG, J. The nature of modeling. *In*: _____. **Artificial Intelligence, Simulation & Modeling**. USA: John Wiley & Sons, Inc., 1989. p. 75–92. ISBN 0471605999.

ROUVOY, R. *et al.* Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. *In*: CHENG, B. H. C. *et al.* (ed.). **Software Engineering for Self-Adaptive Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 164–182. ISBN 978-3-642-02161-9. Disponível em: https://doi.org/10.1007/978-3-642-02161-9_9.

RUIZ-MIRAZO, K.; MORENO, A. Basic autonomy as a fundamental step in the synthesis of life. **Artif. Life**, MIT Press, Cambridge, MA, USA, v. 10, n. 3, p. 235–259, jun 2004. ISSN 1064-5462. Disponível em: https://doi.org/10.1162/1064546041255584.

RUIZ-MIRAZO, K.; PERETó, J.; MORENO, A. A universal definition of life: Autonomy and open-ended evolution. **Origins of life and evolution of the biosphere**, Kluwer Academic Publishers, Cambridge, MA, USA, v. 34, n. 3, p. 323–346, jun 2004. ISSN 1573-0875. Disponível em: https://doi.org/10.1023/B:ORIG.0000016440.53346.dc.

RUSE, M.; TRAVIS, J. (ed.). **Evolution: The First Four Billion Years**. [*S.l.: s.n.*]: Belknap Press, 2009. ISBN 9780674031753.

SABATUCCI, L.; SEIDITA, V.; COSSENTINO, M. The four types of self-adaptive systems: A metamodel. *In*: . [*S.l.: s.n.*], 2018. p. 440–450. ISBN 978-3-319-59479-8.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. **ACM Trans. Auton. Adapt. Syst.**, ACM, New York, NY, USA, v. 4, n. 2, p. 14:1–14:42, maio 2009. ISSN 1556-4665. Disponível em: http://doi.acm.org/10.1145/1516533.1516538.

SALVANESCHI, G.; GHEZZI, C.; PRADELLA, M. Context-oriented programming: A software engineering perspective. **Journal of Systems and Software**, v. 85, n. 8, p. 1801–1817, 2012. ISSN 0164-1212. Disponível em: https://www.sciencedirect.com/science/article/pii/S016412121200074X.

SCHMECK, H. *et al.* Adaptivity and self-organization in organic computing systems. **ACM Trans. Auton. Adapt. Syst.**, ACM, New York, NY, USA, v. 5, n. 3, p. 10:1–10:32, set. 2010. ISSN 1556-4665. Disponível em: http://doi.acm.org/10.1145/1837909.1837911.

SENGE, P. M. **The Fifth Discipline: The Art & Practice of The Learning Organization**. New York: Doubleday/Currency, 1990. (A Currency book). ISBN 9780385260947.

SERRAL, E. *et al.* Context-adaptive petri nets: Supporting adaptation for the execution context. **Expert Systems with Applications**, Elsevier, n. 42, p. 9307–9317, jul. 2015. ISSN 1556-4665. Disponível em: http://dx.doi.org/10.1016/j.eswa.2015.08.004.

SERUGENDO, G. D. M. *et al.* A generic framework for the engineering of self-adaptive and self-organising systems. *In*: BELLMAN, K. *et al.* (ed.). **Organic Computing - Controlled Self-organization**. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008. (Dagstuhl Seminar Proceedings, 08141). ISSN 1862-4405. Disponível em: http://drops.dagstuhl.de/opus/volltexte/2008/1563.

SHALIZI, C. R. Methods and techniques of complex systems science: An overview. *In*: DEISBOECK, T. S.; KRESH, J. Y. (ed.). **Complex Systems Science in Biomedicine**. Boston, MA: Springer US, 2006. cap. 1, p. 33–114. ISBN 978-0-387-33532-2. Disponível em: https://doi.org/10.1007/978-0-387-33532-2_2.

SHANAHAN, T. **The Evolution of Darwinism: Selection, Adaptation and Progress in Evolutionary Biology**. [*S.l.: s.n.*]: Cambridge University Press, 2004.

SHIN, Y.-J.; BAE, J.-Y.; BAE, D.-H. Concepts and models of environment of self-adaptive systems: A systematic literature review. *In*: **2021 28th Asia-Pacific Software Engineering Conference (APSEC)**. [*S.l.: s.n.*], 2021. p. 296–305.

SIMON, H. A. **The Sciences of the Artificial**. 3rd. ed. Cambridge, MA, USA: MIT Press, 1996. ISBN 0262691914.

SINGH, P. K. *et al.* Autonomic computing: A revolutionary paradigm for implementing self-managing systems. *In*: **2011 International Conference on Recent Trends in Information Systems**. [*S.l.: s.n.*], 2011. p. 7–12.

SNOWDEN, D. Strategy in the context of uncertainty. **Handbook of Business Strategy**, v. 6, n. 1, p. 47–54, 2005. Disponível em: https://doi.org/10.1108/08944310510556955.

SNOWDEN, D. J.; BOONE, M. E. A leader's framework for decision making. **Harvard Business review**, Harvard Business Review, v. 85, n. 11, p. 68–76, Nov. 2007. ISSN 0017-8012. Disponível em: https://hbr.org/2007/11/a-leaders-framework-for-decision-making.

SOMMERVILLE, I. *et al.* Large-scale complex it systems. **Commun. ACM**, ACM, New York, NY, USA, v. 55, n. 7, p. 71–77, jul. 2012. ISSN 0001-0782. Disponível em: http://doi.acm.org/10.1145/2209249.2209268.

STAHL, T.; VOELTER, M.; CZARNECKI, K. **Model-Driven Software Development: Technology, Engineering, Management**. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2006. ISBN 0470025700.

STEELS, L. When are robots intelligent autonomous agents? **Robotics and Autonomous Systems**, v. 15, n. 1, p. 3 – 9, 1995. ISSN 0921-8890. The Biology and Technology of Intelligent Autonomous Agents. Disponível em: http://www.sciencedirect.com/science/article/pii/0921889095000114.

STERRITT, R. *et al.* A concise introduction to autonomic computing. **Adv. Eng. Inform.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 19, n. 3, p. 181–187, jul. 2005. ISSN 1474-0346. Disponível em: http://dx.doi.org/10.1016/j.aei.2005.05.012.

STERRITT, R. *et al.* Self-* properties in nasa missions. *In*: **In 4th International Workshop on System/Software Architectures (IWSSA'05) in Proc. 2005 International Conference on Software Engineering Research and Practice (SERP'05)**. [*S.l.: s.n.*]: CSREA Press, 2005.

STRANG, T.; POPIEN, C. L. A context modeling survey. *In*: **UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management**. Nottingham: [*S.l.: s.n.*], 2004. p. 31–41. Disponível em: http://citeseer.ist.psu.edu/712164.html.

STURMBERG, J.; MARTIN, C. Complexity and health - yesterday's traditions, tomorrow's future. v. 15, p. 543 – 548, 05 2009.

SUCIPTO, S.; WAHONO, R. S. A systematic literature review of requirements engineering for self-adaptive systems. **Journal of Software Engineering**, v. 1, n. 1, 2015. Disponível em: http://journal.ilmukomputer.org/index.php/jse/article/view/33.

TEYSSIER, J. *et al.* Photonic crystals cause active colour change in chameleons. **Nature Communications**, Macmillan Publishers Limited, v. 6368, n. 6, 2015. ISSN 2041-1723. Disponível em: http://www.jstor.org/stable/2832133.

THE HEALTH FOUNDATION. **Evidence scan: Complex adaptive systems**. [*S.l.: s.n.*]: The Health Foundation, 2010. 1–33 p.

TOMFORDE, S.; SICK, B.; MÜLLER-SCHLOER, C. Organic computing in the spotlight. **CoRR**, abs/1701.08125, 2017. Disponível em: http://arxiv.org/abs/1701.08125.

TRITCHKOV, I.; GOETZ, H. Verification and validation of decentralized, self-organizing cyber-physical production systems: A blueprint process for testing cyber-physical production systems with self-properties. *In*: **2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)**. [*S.l.: s.n.*], 2016. p. 112–117. ISBN 978-1-5090-3651-6.

VAISHNAVI, V. K.; KUECHLER, W. **Design Science Research Methods and Patterns: Innovating Information and Communication Technology, 2nd Edition**. 2nd. ed. USA: CRC Press, Inc., 2015. ISBN 1498715257.

VARELA, F.; MATURANA, H.; URIBE, R. Autopoiesis: The organization of living systems, its characterization and a model. **Biosystems**, v. 5, n. 4, p. 187–196, 1974. ISSN 0303-2647. Disponível em: https://www.sciencedirect.com/science/article/pii/0303264774900318.

VARIOUS. **International Encyclopedia of Systems and Cybernetics**. K.G. Saur, 2005. v. 61. 672-675 p. ISBN 978-3598116308. Disponível em: https://doi.org/10.1108/00220410510625877.

VARIOUS. **Chambers's Twentieth Century Dictionary (part 1 of 4: A-D)**. Library of Alexandria, 2017. (Library of Alexandria). ISBN 9781465562883. Disponível em: http://www.gutenberg.org/files/37683/37683-h/37683-h.htm.

VARIOUS. **Wikitionary - adapt**. 2017. https://en.wiktionary.org/wiki/adapt#English. Accessed: 2018-04-18.

VIROLI, M. *et al.* Combining self-organisation and autonomic computing in cass with aggregate-mape. *In*: **2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)**. [*S.l.: s.n.*], 2016. p. 186–191.

VROMANT, P. *et al.* On interacting control loops in self-adaptive systems. *In*: **Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. New York, NY, USA: ACM, 2011. (SEAMS '11), p. 202–207. ISBN 978-1-4503-0575-4. Disponível em: http://doi.acm.org/10.1145/1988008.1988037.

WAN, K. A brief history of context. **CoRR**, abs/0912.1838, 2009. Disponível em: http://arxiv.org/abs/0912.1838.

WARRIACH, E.; ÖZçELEBI, T.; LUKKIEN, J. Self-* properties in smart environments: Requirements and performance metrics. *In*: AUGUSTO, J.; ZHANG, T. (ed.). **Workshop Proceedings of the 10th International Conference on Intelligent Environments (IE'14, Shanghai, China, June 30-July 4, 2014)**. Netherlands: IOS Press, 2014. (Ambient Intelligence and Smart Environments), p. 194–205. ISBN 978-1-61499-410-7.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. [*S.l.: s.n.*]: Elsevier, 2009. v. 2.

WEBER, S. Design science research: Paradigm or approach? *In*: **Proceedings of the Sixteenth Americas Conference on Information Systems**. Lima, Peru: Association for Information Systems, AIS e-Library, 2010. p. 214.

WEYNS, D. Software engineering of self-adaptive systems: An organised tour and future challenges. *In*: TAYLOR, R.; KANG, K. C.; CHA, S. (ed.). **Handbook of Software Engineering**. [*S.l.: s.n.*]: Springer, 2017.

WEYNS, D. *et al.* Claims and supporting evidence for self-adaptive systems: A literature study. *In*: **Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. Piscataway, NJ, USA: IEEE Press, 2012. (SEAMS '12), p. 89–98. ISBN 978-1-4673-1787-0. Disponível em: http://dl.acm.org/citation.cfm?id=2666795.2666811.

WEYNS, D.; MALEK, S.; ANDERSSON, J. Forms: Unifying reference model for formal specification of distributed self-adaptive systems. **ACM Trans. Auton. Adapt. Syst.**, ACM, New York, NY, USA, v. 7, n. 1, p. 8:1–8:61, maio 2012. ISSN 1556-4665. Disponível em: http://doi.acm.org/10.1145/2168260.2168268.

WHITTLE, J. *et al.* Relax: Incorporating uncertainty into the specification of self-adaptive systems. *In*: **Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE**. Washington, DC, USA: IEEE Computer Society, 2009. (RE '09), p. 79–88. ISBN 978-0-7695-3761-0. Disponível em: http://dx.doi.org/10.1109/RE.2009.36.

WIERINGA, R. J. Design science as nested problem solving. *In*: **Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology**. New York, United States: Association for Computing Machinery (ACM), 2009. p. 1–12. ISBN 978-1-60558-408-9.

WILLIAMS, M. *et al.* The anthropocene biosphere. **The Anthropocene Review**, v. 2, n. 3, p. 196–219, 2015. Disponível em: https://doi.org/10.1177/2053019615591020.

YANG, Z. *et al.* Review on requirements modeling and analysis for self-adaptive systems: A ten-year perspective. **CoRR**, abs/1704.00421, 2017. Disponível em: http://arxiv.org/abs/1704.00421.

YATES, F. E. Biological views of adaptation — some historical accounts. *In*: _____. **Adaptive Control of Ill-Defined Systems**. Boston, MA: Springer US, 1984. p. 39–50. ISBN 978-1-4684-8941-5. Disponível em: https://doi.org/10.1007/978-1-4684-8941-5_5.

ZADEH, L. A. On the definition of adaptivity. **Proceedings of the IEEE**, v. 51, n. 3, p. 469–470, March 1963. ISSN 0018-9219.

ZIEMKE, T. Adaptive behavior in autonomous agents. **Presence: Teleoperators and Virtual Environments**, v. 7, n. 6, p. 564–587, 1998. Disponível em: https://doi.org/10.1162/105474698565947.

APÊNDICES

APÊNDICE A – TERMINOLOGIA RELACIONADA

Segundo o Dicionário Chamber's do Século Vinte (VARIOUS, 2017a) o verbo **adaptar** vem do Latim *adaptāre* e significa na sua forma raiz: tornar apto/adequado/próprio ou acomodar/ajustar/encaixar com ou para alguma coisa.

Existem vários termos derivados desta forma raiz, estreitamente relacionados, mas que carregam significados com diferenças sensíveis no nível semântico.

A consequência de dar pouca atenção a esse assunto tem gerado muitas confusões, que ampliam o grau de dificuldade na análise dos fundamentos empregados para trabalhar com o comportamento adaptativo.

A.1 Terminologia próxima

A Figura 97 apresenta de maneira resumida o esquema morfológico do verbo *adapt* e termos relacionados, no inglês, elaborado com base nas informações de (VARIOUS, 2017b).

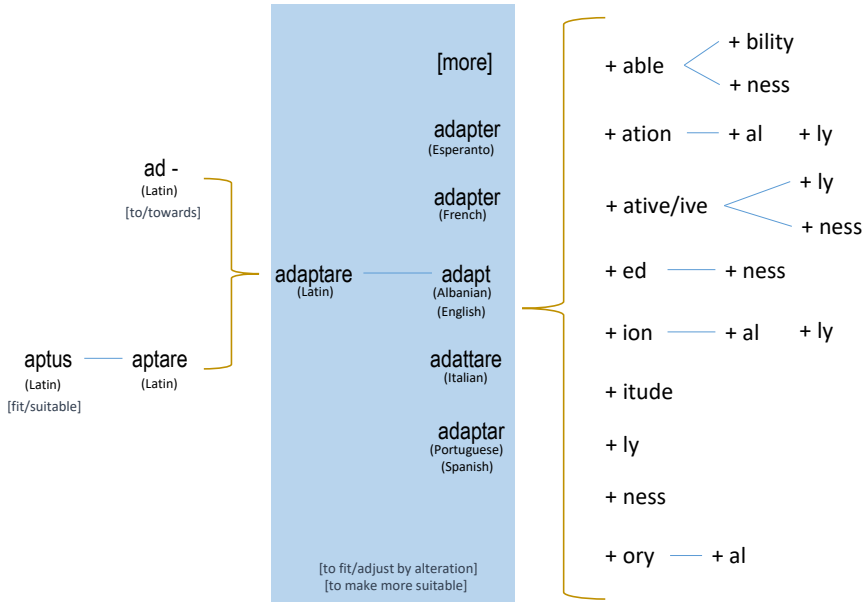


Figura 97 – Espaço morfológico do verbo *adapt* e termos relacionados.
O Autor (2018).

No nível linguístico, tanto o dicionário Aurélio (HOLANDA, 2017)(português) quanto o Merriam-Webster (MERRIAM-WEBSTER DICTIONARY, 2017), Oxford (ENGLISH OXFORD DICTIONARY, 2017) e Cambridge (CAMBRIDGE ENGLISH DICTIONARY, 2017) (inglês) definem a adaptabilidade como a qualidade ou capacidade de uma

entidade de ser adaptável, o que a sua vez significa que pode ser objeto de adaptação ou mudança, em alguma das suas caraterísticas, com a finalidade de ajustá-la ou torná-la mais adequada para um novo propósito ou situação. Assim, uma entidade que possui adaptabilidade admite a possibilidade de aplicar mudanças nas suas caraterísticas-padrão. As mesmas fontes definem adaptatividade ou adaptativo como a capacidade de uma entidade de se adaptar, de ajustar as suas próprias caraterísticas às novas condições e mudanças no seu entorno. Assim, de maneira geral, é possível definir a adaptatividade como a capacidade que possui alguma entidade de efetuar alguma mudança nela mesma, a fim de se adequar a uma nova situação. A diferença desses termos, como apontado em Caya e Neto (2016), é a entidade executora da mudança ou adaptação: no caso da adaptabilidade, pode ser a própria entidade ou um agente externo, enquanto na adaptatividade, a própria entidade é a responsável direta, tanto da decisão de mudança quanto da sua execução.

No nível de teoria de sistemas, outras várias terminologias são criadas, definidas e exploradas, estabelecendo novas categorizações. Neste sentido, alguns critérios adotados para a classificação dos comportamentos adaptativos são: a origem do estímulo que ativa a adaptação, ou do objeto sobre o qual serão aplicadas as mudanças, a finalidade da mudança, ou a instância de adaptação que estiver sendo avaliada. No entanto, a diferença básica entre adaptabilidade e adaptatividade reside na origem da decisão que provoca a realização das mudanças necessárias, como apontado pelas entradas correspondentes na Enciclopédia Internacional de Sistema e Cibernética (VARIOUS, 2005).

A importância de observar esta sutil, porém sensível diferença é destacada na elaboração do modelo holístico do comportamento adaptativo. A terminologia em diferentes áreas apresenta particularidades nem sempre conciliáveis e só o conhecimento semântico permite diferenciar entre uma entidade com a capacidade de aceitar mudanças nas suas caraterísticas, doravante denominada *adaptável*, e outra com a capacidade de tomar a decisão de mudar suas configurações para otimizar o seu funcionamento, doravante denominada *adaptativa*. No nosso caso estamos interessados em observar os elementos presentes no comportamento do segundo tipo de entidade.

A Figura 98 apresenta um mapa conceitual dos tipos de dispositivos com respeito às suas capacidades de desenvolver comportamento adaptativo.

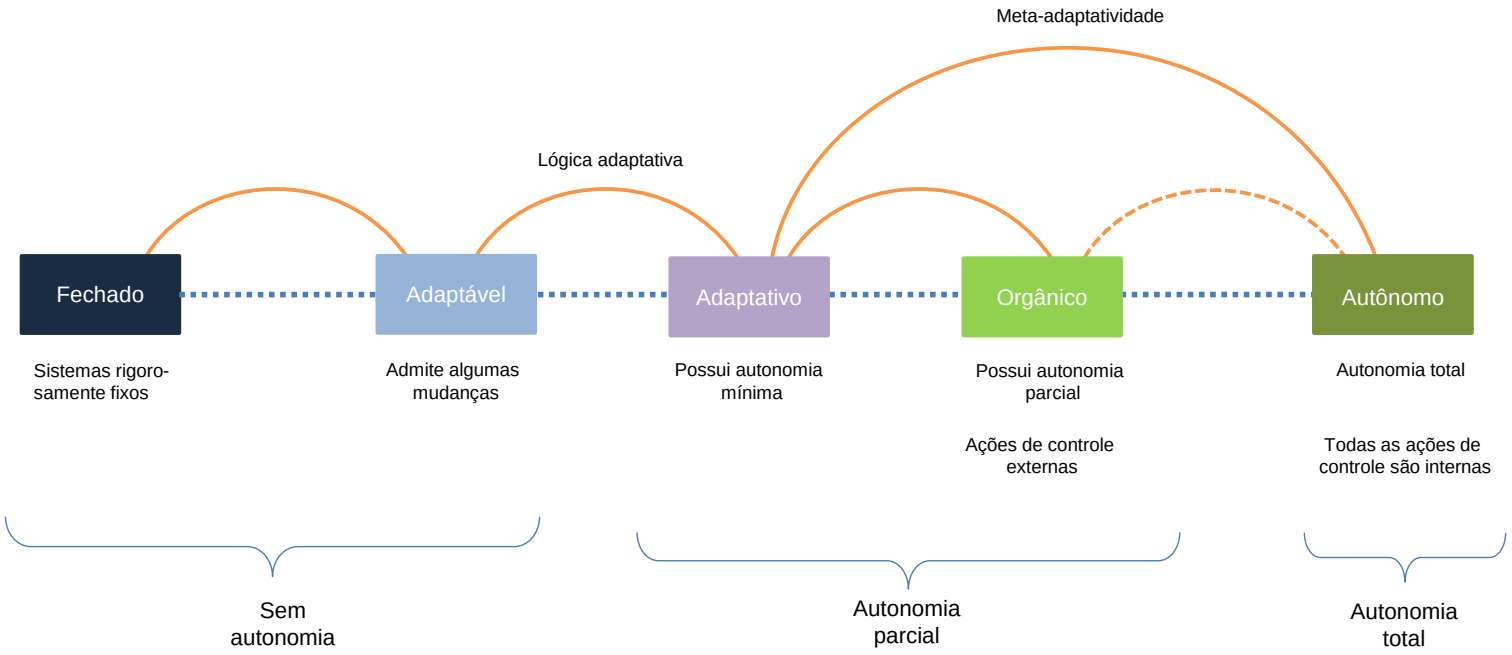


Figura 98 – Tipos de dispositivos com diferentes capacidades de comportamento adaptativo.
O Autor (2018).

A.2 Outros nomes

Os sistemas adaptativos também podem ser nomeados de diferentes maneiras dependendo das áreas nas quais são desenvolvidas as análise e/ou abordagens. No entanto, a maioria destes termos são "quase" equivalentes, alguns deles indicam uma abordagem mais específica, enquanto outros preferem manter uma descrição mais geral.

Alguns termos mais frequentemente encontrados na literatura são:

- Sistemas Autoadaptativos: do inglês *Self-adaptive Systems*(SaS). No contexto de estudo dos sistemas autogerenciáveis (self-managed systems) são denominados SaS os sistemas capazes de manter automaticamente um correto funcionamento em diferentes contextos. Neste caso é importante notar que o prefixo "self-" é utilizado para enfatizar que a adaptatividade está contida no grupo das chamadas propriedades self-*, mas que no caso do comportamento adaptativo ele só existe se é decidido e aplicado pela própria entidade.

- Autogestão: do inglês *Self-management* é o processo pelo qual os sistemas de computador conseguem gerenciar sua própria operação sem intervenção humana. A tecnologia de autogestão requer capacidades de adaptação, introspecção e autocura.

- Auto-Ajustável: também conhecido em inglês como *Self-tuning* ou *autotuning* é um termo originário da Teoria de Controle, que define um sistema capaz de otimizar seus próprios parâmetros internos de execução para maximizar ou minimizar o cumprimento de uma função objetivo. O termo também é definido como reajuste contínuo ou *continuous re-tuning*.

- Auto-(re)configuração: do inglês *Self-reconfiguring* se refere à capacidade pela qual sistemas ou redes de computadores aplicam, automaticamente, mudanças em sua própria configuração de componentes, em resposta a ocorrências internas ou no seu ambiente de execução, sem intervenção humana direta.

- Autoadaptativo em tempo real: do inglês *real-time self-adaptive*

- Adaptatividade de tempo de execução: do inglês *real-time adaptation*

- Reconfiguração dinâmica: do inglês *Dynamic reconfiguration*

- Sistemas Adaptativos Dinâmicos: do inglês *Dynamic(ally) Adaptive Systems* são sistemas que monitoram propriedades selecionadas de seu ambiente e raciocinam sobre os valores medidos para determinar quais comportamentos o sistema deve adotar para melhor operar no ambiente detectado.

- Software de personalização automática: do inglês *Automatic/Automated software personalization* são sistemas que selecionam e configuram, principalmente, as caraterísticas que governam a interação homem-máquina para fornecer alternativas mais alinhadas com o perfil do usuário. O termo está fortemente associado com acessibilidade, usabilidade e usuários com deficiência.

- Computação Autonômica: é a visão da IBM que visa desenvolver sistemas computacionais capazes de autogerenciar tarefas relacionadas à manutenção e otimização de forma independente (com pouca ou nenhuma interação humana). O sistema toma decisões adaptativas, usando políticas de alto nível dadas pelo gerente humano para superar os diferentes tipos de mudanças. Ele foi desenvolvido, principalmente, para superar a crescente complexidade do gerenciamento de sistemas computacionais.

- Auto-organização: mais próximo de sistemas adaptativos complexos do que sistemas adaptativos, em geral.

Uma coleção de palavras-chave relacionadas com o comportamento adaptativo está disponível na página Contribuições do repositório que acompanha este trabalho [1].

[1] https://sites.google.com/view/rosalia-caya-thesis-document/contribui%C3%A7%C3%B5es

APÊNDICE B – PENSAMENTO SISTÊMICO: SISTEMAS SIMPLES, COMPLICADOS E COMPLEXOS

O Pensamento Sistêmico ou Visão de Sistema, é uma abordagem holística (ARNOLD; WADE, 2015) para a análise se problemas focada na maneira como as partes constituintes do sistema se inter-relacionam, e no funcionamento do sistema ao longo do tempo no contexto de execução. Em contraste com as abordagens de análise tradicionais, as quais observam e estudam os sistemas através da separação dos seus componentes, o Pensamento Sistêmico fornece uma estrutura, tanto conceitual quanto técnica, para ter uma visão geral do problema, sua dinâmica e os inter-relacionamentos entre seus componentes no lugar de observar elementos individuais, estudando os padrões de mudança que o sistema desenvolve ao longo do seu funcionamento ao invés de representações instantâneas e estáticas (SENGE, 1990). Assim, o Pensamento Sistêmico trabalha como uma metodologia para melhorar a capacidade de identificar e entender os sistemas, prever os seus comportamentos, e planejar modificações para produzir os efeitos desejados (ARNOLD; WADE, 2015).

Na literatura existem diferentes definições que tentam descrever de maneira formal e também prática os conceitos principais relacionados com o Pensamento sistêmico, assim como ferramentas e técnicas que os suportam. Da mesma maneira, existem diversas propostas de classificação de sistemas, por exemplo, em Gharajedaghi (2012), o autor examina os diferentes sistemas segundo seus princípios, dimensões, e modelos socioculturais. No entanto, uma primeira classificação interessante é aquela que observa a quantidade e o tipo de relacionamento de um componente com os outros dentro do próprio sistema, assim como entre os componentes e o ambiente de execução. Baseado neste critério, o trabalho em Snowden (2005) identifica cinco domínios possíveis para os sistemas:

- Simples: representa sistemas onde as relações são conhecidas.
- Complicado: representa sistemas onde as relações podem ser descobertas com as ferramentas apropriadas.
- Complexo: representa sistemas onde as relações não podem ser antecipadas, só é possível tentar prever alguns comportamentos com base em informações anteriores.
- Caótico: representa sistemas nos quais não existe relação causa-efeito entre os elementos do sistema. Não existe controle nem previsibilidade.
- Desordenado: representa a impossibilidade de identificar o tipo de relacionamento entre os elementos.

A Figura 99 mostra as principais diferenças para distinguir entre os sistemas simples, complicados, complexos e caóticos.

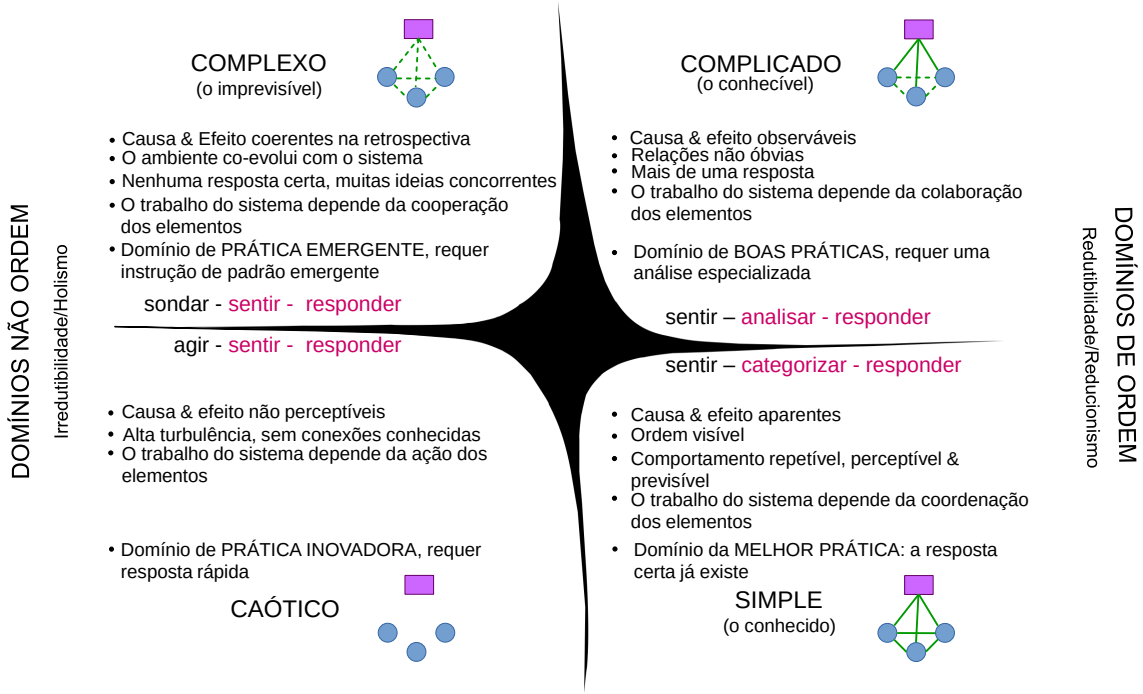


Figura 99 – Cynefin Framework para a diferenciação dos diferentes tipos de sistemas. Adaptado de Snowden (2005), Snowden e Boone (2007), Sturmberg e Martin (2009).

A seguir aprofundaremos na diferença entre os sistemas simples, complicados e complexos.

Um sistema é considerado simples se, como mencionado antes, as relações entre os elementos são conhecidas. Normalmente, um sistema simples está conformado por um número relativamente pequeno de elementos que atuam juntos de acordo com regras bem definidas e compreensíveis (GRABOWSKI; STRZALKA, 2008). Assim, um sistema simples pode ser completamente descrito de maneira determinística (SOMMERVILLE *et al.*, 2012) com base na funcionalidade dos seus elementos e as relações entre eles.

Um sistema é considerado complicado se possui muitos elementos, cada um com funcionalidades bem definidas e compreensíveis, mas que precisam colaborar de diferentes maneiras para atingir o objetivo global. Assim, um sistema complicado é aquele cuja funcionalidade pode ser descrito de maneira bem definida (GRABOWSKI; STRZALKA, 2008), mesmo que as relações entre suas partes não sejam fáceis de identificar.

Ambos sistemas, simples e complicados, são também denominados sistemas lineares, pois o comportamento deles é consequência direta da aplicação das regras que os definem.

Um sistema é denominado complexo se está conformado de muitos elementos, de natureza não necessariamente idêntica, que cooperam para atingir um objetivo comum. As relações de esses elementos, internamente e com o ambiente, não podem ser explicados

por completo só pela análise independente das partes devido à sua natureza dinâmica e de constante mudança (GRABOWSKI; STRZALKA, 2008; SOMMERVILLE *et al.*, 2012). Assim, devido à incerteza do seu comportamento, os sistemas complexos são considerados não determinísticos. O próprio termo complexo, derivado de complexidade, acarreta um sentido de interconectividade profunda e interações dinâmicas resultantes na "emergência"(entendido como evidência) de novas características, tanto dentro dos próprios Sistemas Adaptativos Complexos quanto através deles. Alguns exemplos de sistemas complexos são o sistema imunológico, sistemas neurais, e inclusive comportamentos animais, como os mecanismos de nado dos peixes, as estruturas nos bandos, e os padrões de movimentos dos animais. Exemplos ao nível macro são a floresta amazônica Brasileira, os sistemas sociais, o aprendizado ou inclusive o próprio desenvolvimento da humanidade.

Um sistema caótico descreve também um comportamento complexo decorrente de um sistema determinístico dinâmico não-linear, que exibe duas propriedades especiais: (1) dependências sensíveis nas condições iniciais, (2) estruturas características. A natureza determinista desses sistemas não os torna previsíveis, pelo contrário os sistemas caóticos possuem um comportamento considerado imprevisível.

Para consolidar esses conceitos, a Figura 100 permite uma comparação visual entre exemplos de sistemas de cada um dos domínios: simples, complicado, complexo e caótico.

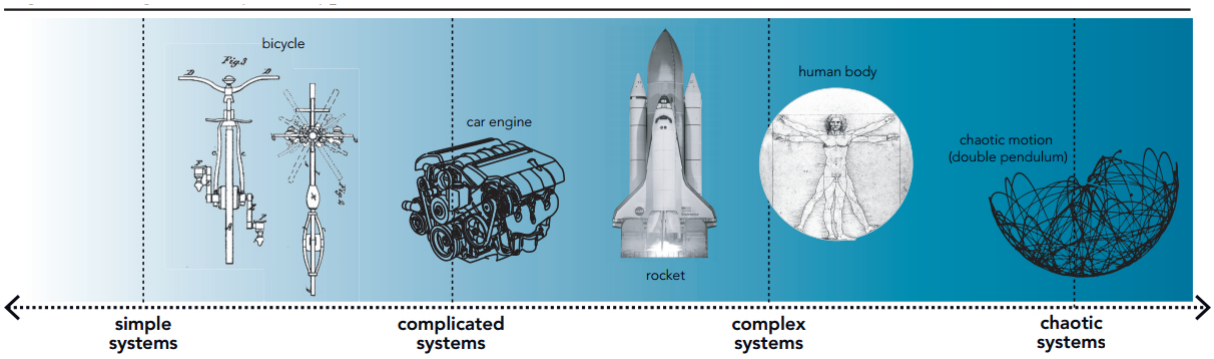


Figura 100 – Categorização de tipos de sistema.
Obtido de Leroux-Martin e O'Connor (2017).

Na literatura existem estudos de diferentes áreas que avaliam as diversas técnicas, abordagens e padrões que ajudam a lidar com problemas em cada um dos tipos de sistemas. A Tabela 101 apresenta um roteiro para ajudar na análise e caracterização. Este tipo de recurso geral cobra vital importância quando observamos que é possível identificar sistemas de diferentes naturezas.

Table I

	Chaotic un-order	Complex un-order	Hidden order	Visible order
Decision model	Act-sense-respond	Probe-sense-respond	Sense-analyse-respond	Sense-categorise-respond
Practice	Novel practice	Worst practice	Good practice	Best practice
	Avoidance of failure		Imitation of success	
Related techniques	Cynefin development area: sense making, narrative and network mangement		Systems thinking Scenario planning Market intelligence	Process re-engineering Standard operating Procedures
Nature of knowing	Emergent, experimental, uncertain, knowledge of conditions but not outcomes		Empirical, hypothesis based, objective validation or refutation	
Nature of management	Decisive-directive	Information-consensual	Oligarchic-consensual	Hierarchical-directive
Network	No connections	Tightly connected network, loosely connected centre	Tightly connected centre, tightly connected network	Tightly connected centre, network connections irrelevant
Diagnostics	Every diagnostic action is an intervention and every intervention a diagostic		Diagnostic can be isolated from intervention and do not impact on system behaviour	
Illustrative legitimate domain use	As a source of innovation and new perspective	To enable self organisation	Use of expert knowledge	Process control to for safety
Illustrative illegitimate domain use	Accidental precipitation into a crisis	Anarchy without control of any time	Refusal of experts to accept new ideas	Over control preventing progress
Focus	Effectiveness: allowing a degree of redundancy to create adaptive capacity		Efficiency: stripping away superfluous functions to maximise productity	
Nature	EXPLORATION		EXPLOITATION	

Figura 101 – Análise de características dos diferentes tipos de sistema.
Obtido de Snowden (2005).

Em particular, a terceira geração do Pensamento Sistêmico visa responder ao triplo desafio fixado por problemas de natureza interdependente, auto-organizada, e mudanças dinâmicas, presentes em sistemas mecânicos, vivos e sociais (GHARAJEDAGHI, 2012).

Independentemente do problema particular analisado, o uso do Pensamento Sistêmico, nos permite um melhor entendimento do mundo ao nosso redor. Como consequência, é possível executar melhores tomadas de decisão, desenvolver melhores designs e elaborar soluções significativas, estratégicas e duradouras.

O pensamento sistêmico tem sido aplicado em diferentes áreas, tais como estudos de sistemas médicos, sistemas ambientais, sistemas políticos e econômicos, recursos humanos, e sistemas educacionais.

Para uma descrição mais aprofundada do Pensamento sistêmico consultar:

- *Systems thinking coping with 21st century problems*, que proporciona duas maneiras de exercitar o Pensamento Sistêmico: aplicando as técnicas dos sistemas aos problemas, ou modelando os problemas como sistemas. O livro também apresenta as principais contribuições do Pensamento Sistêmico nas áreas da indústria e engenharia.
- *Advanced systems thinking, engineering, and management*, o livro que contém tanto filosofia do pensamento sistêmico quanto metodologia para sua aplicação e diversidade de exemplos de resolução de problemas.
- *Systems Thinking, Systems Practice: Includes a 30-Year Retrospective*, que oferece uma contextualização histórica sobre o pensamento sistêmico e sua evolução, detalhes sobre a teoria e metodologia que pose ser utilizada para sua aplicação em contextos organizacionais.

Para uma descrição mais aprofundada das diferenças básicas entre sistemas simples e complexos consultar os trabalhos de J. Snowden (SNOWDEN; BOONE, 2007; SNOWDEN, 2005) e Grabowski (GRABOWSKI; STRZALKA, 2008).

APÊNDICE C – PROPOSTA DE TAXONOMIA PARA COMPORTAMENTOS ADAPTATIVOS

A adaptatividade é entendida de maneira geral como um mecanismo abstrato que permite a mudança de uma entidade, em um momento determinado de sua execução, com a finalidade de evitar algum perigo ou aproveitar alguma vantagem. No entanto, a materialização deste conceito apresenta uma variedade de formas, dependendo, por exemplo, a abordagem de observação, a entidade sujeita à mudança, o ambiente de execução, entre outros. A seguir apresentamos uma taxonomia, no formato de mapas conceituais nas Figuras 102, 103, 104, 105 e 106. Esta taxonomia procura oferecer um mapa para analisar as características de alguma proposta de implementação de comportamento adaptativo dentro da computação, tecnologia e áreas relacionadas. É importante ressaltar que a taxonomia não é exaustiva nem definitiva, pelo contrário pretende ser só uma organização intermediária, que pode e deve evoluir com o desenvolvimento e fortalecimento de estudos sobre a teoria da adaptatividade. Assim, motivamos o uso e modificação desta taxonomia com a finalidade de melhorar a classificação e análise dos trabalhos científicos sobre sistemas adaptativos.

Esta taxonomia foi elaborada com base em trabalhos científicos de grande abrangência, tais como: surveys, revisões sistemáticas, taxonomias, e categorizações. Esses trabalhos analisam o comportamento adaptativo desde diferentes áreas e com diferentes terminologias, gerando uma ferramenta holística. A seguir apresentamos as referências, em formato de lista e de maneira geral, com a finalidade de manter a legibilidade e praticidade da taxonomia na sua representação gráfica:

- OREIZY, P. *et al.* An architecture-based approach to self-adaptive software. **IEEE Intelligent Systems and their Applications**, v. 14, n. 3, p. 54–62, May 1999. ISSN 1094-7167.

- SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. **ACM Trans. Auton. Adapt. Syst.**, ACM, New York, NY, USA, v. 4, n. 2, p. 14:1–14:42, maio 2009. ISSN 1556-4665. Disponível em: http://doi.acm.org/10.1145/1516533.1516538.

- BRADBURY, J. S. *et al.* A survey of self-management in dynamic software architecture specifications. *In*: **Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems**. New York, NY, USA: ACM, 2004. (WOSS '04), p. 28–33. ISBN 1-58113-989-6. Disponível em: http://doi.acm.org/10.1145/1075405.1075411.

- BUCKLEY, J. *et al.* Towards a taxonomy of software change. **Journal of Software Maintenance and Evolution: Research and Practice**, v. 17, n. 5, p. 309–332, 2005. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.319.

- MACíAS-ESCRIVá, F. D. *et al.* Self-adaptive systems: A survey of current approaches, research challenges and applications. **Expert Systems with Applications**, v. 40, n. 18, p. 7267 – 7279, 2013. ISSN 0957-4174. Disponível em: http://www.sciencedirect.com/science/article/pii/S0957417413005125.

- SERRAL, E. *et al.* Context-adaptive petri nets: Supporting adaptation for the execution context. **Expert Systems with Applications**, Elsevier, n. 42, p. 9307–9317, jul. 2015. ISSN 1556-4665. Disponível em: http://dx.doi.org/10.1016/j.eswa.2015.08.004.

- CARDOZO, N. *et al.* Modeling and analyzing self-adaptive systems with context petri nets. *In*: **2013 International Symposium on Theoretical Aspects of Software Engineering**. [*S.l.: s.n.*], 2013. p. 191–198.

- BRUNI, R. *et al.* A conceptual framework for adaptation. *In*: LARA, J. de; ZISMAN, A. (ed.). **Fundamental Approaches to Software Engineering**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 240–254. ISBN 978-3-642-28872-2.

- BARBIER, F. *et al.* Software adaptation: Classification and a case study with state chart xml. **IEEE Software**, v. 32, n. 5, p. 68–76, Sep. 2015. ISSN 0740-7459.

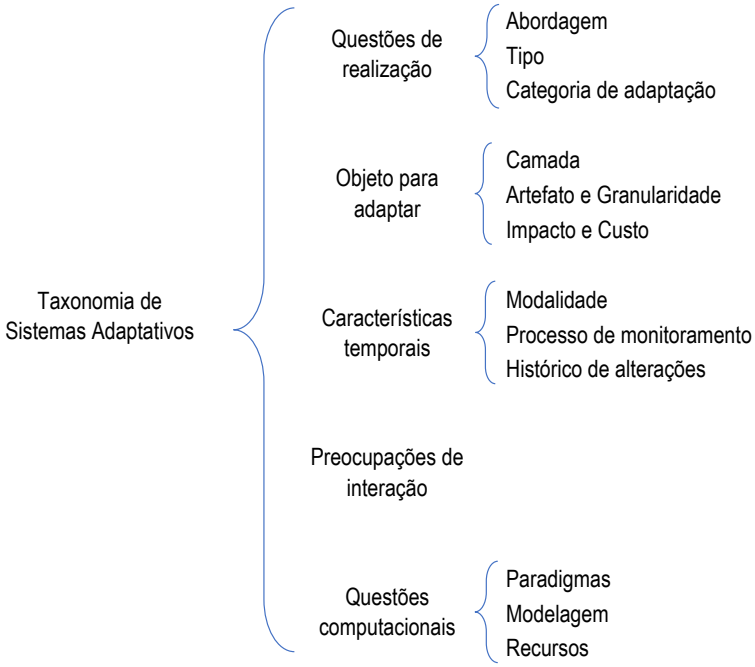


Figura 102 – Taxonomia de Sistemas Adaptativos no nível geral.
O Autor (2018).

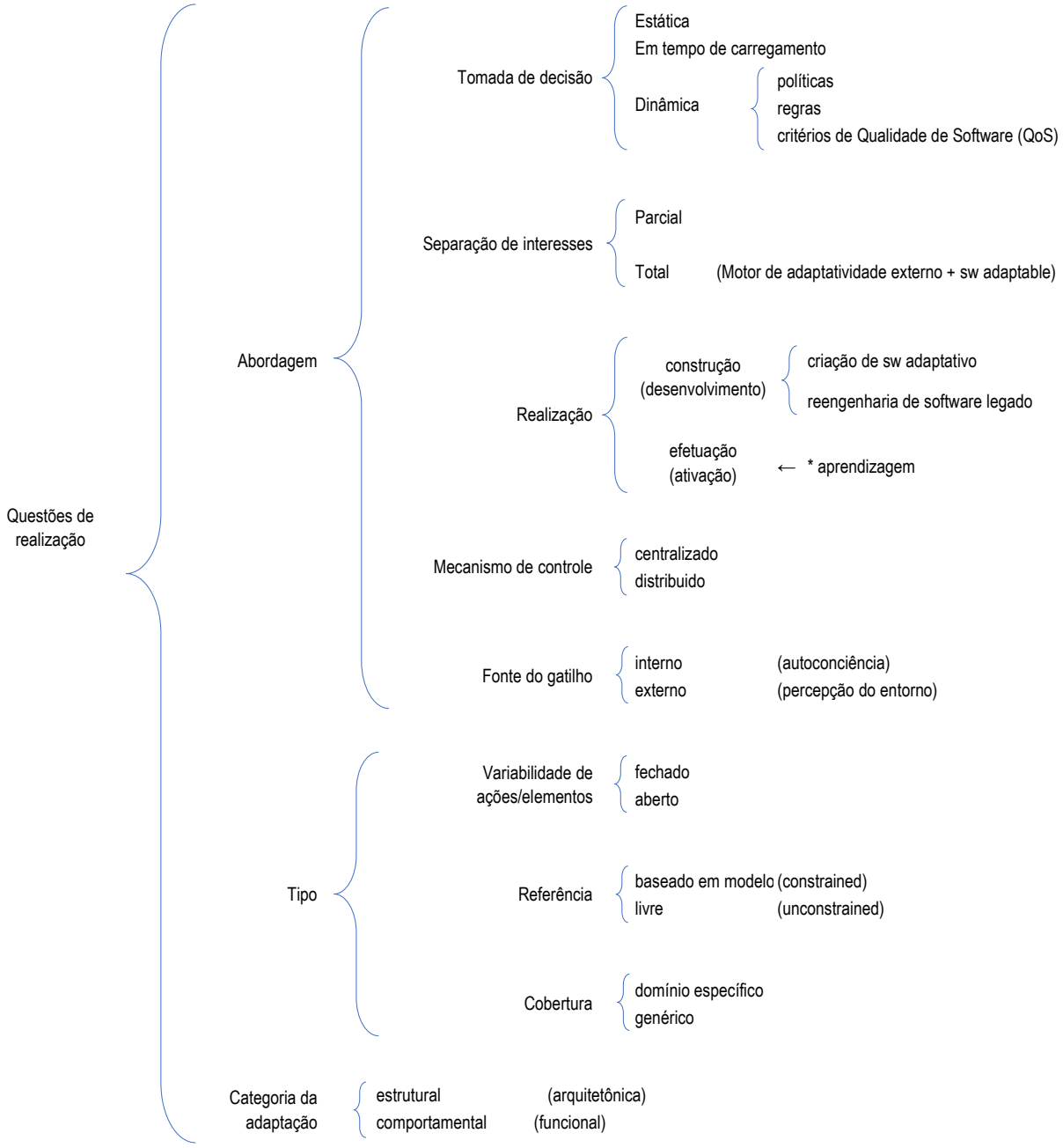


Figura 103 – Taxonomia de Sistemas Adaptativos no sub-nível considerações de realização. O Autor (2018).

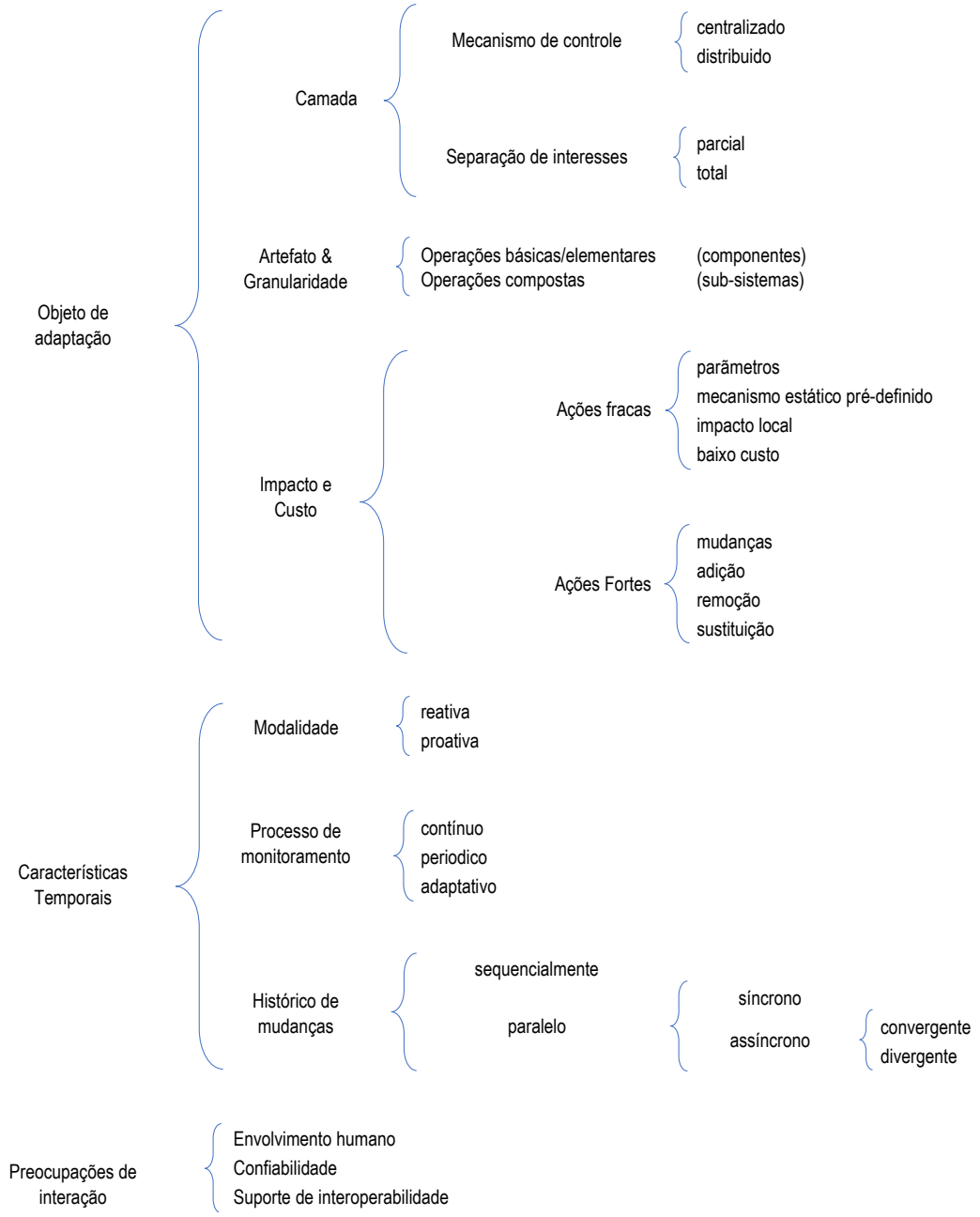
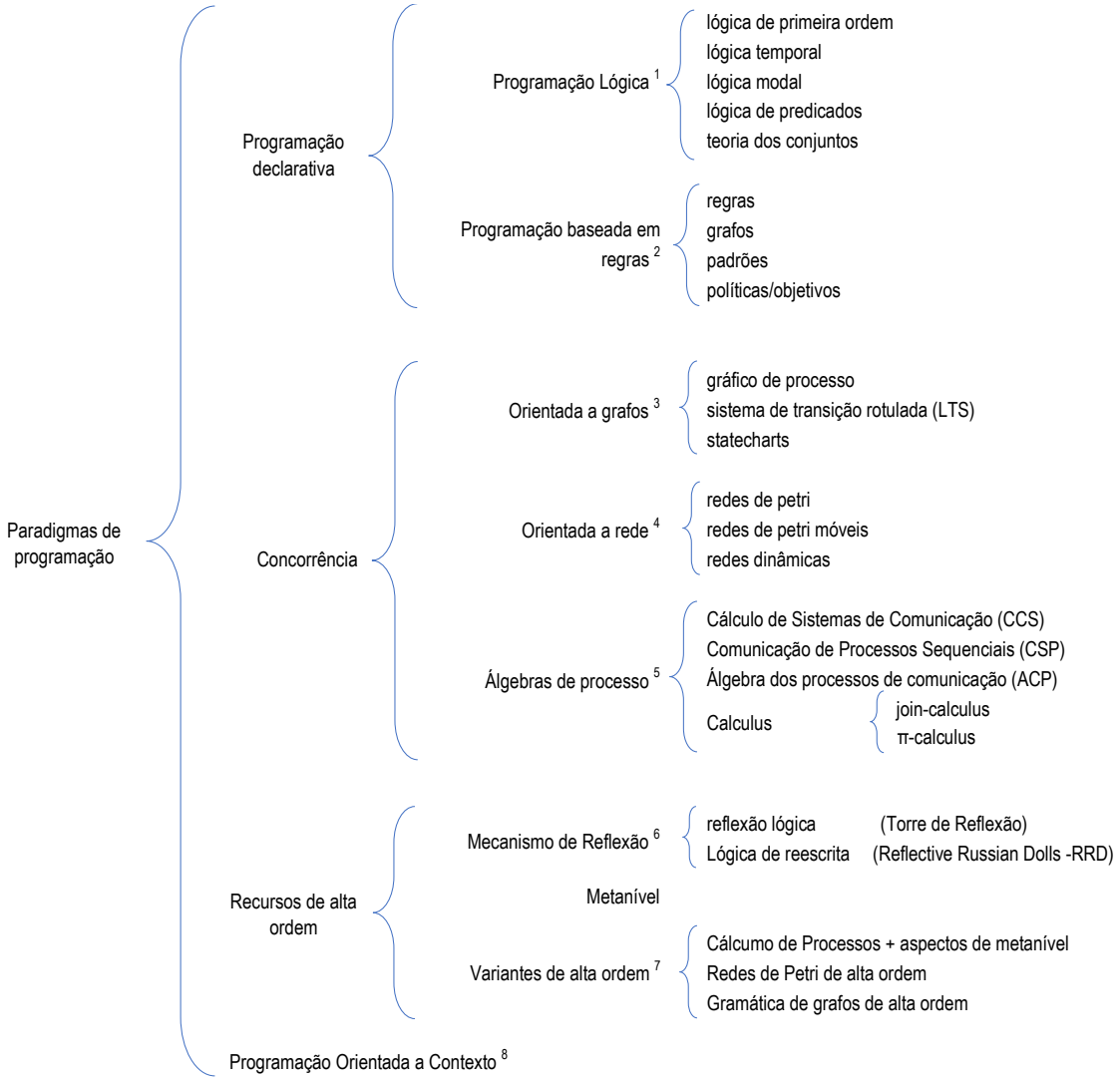


Figura 104 – Taxonomia de Sistemas Adaptativos nos sub-níveis de objeto a adaptar, características temporárias e considerações da interação.
O Autor (2018).



Adaptatividade está representa através de:

[1] manipulação de cláusulas
[2] regras de adaptação
[3] regras de reescrita e transformações de gráficos
[4] informações agregadas à estrutura das redes
[5] comportamento dos processos nos canais de adaptação. Os processos provavelmente irão misturar adaptação com computação
[6] manipulação de regras através de reflexão em metaníveis
[7] modificação do programa (como cidadão de primeira classe) durante a execução
[8] reações às mudanças/eventos do contexto

Figura 105 – Taxonomia de Sistemas Adaptativos no sub-nível de considerações computacionais focando em paradigmas de computação.
O Autor (2018).

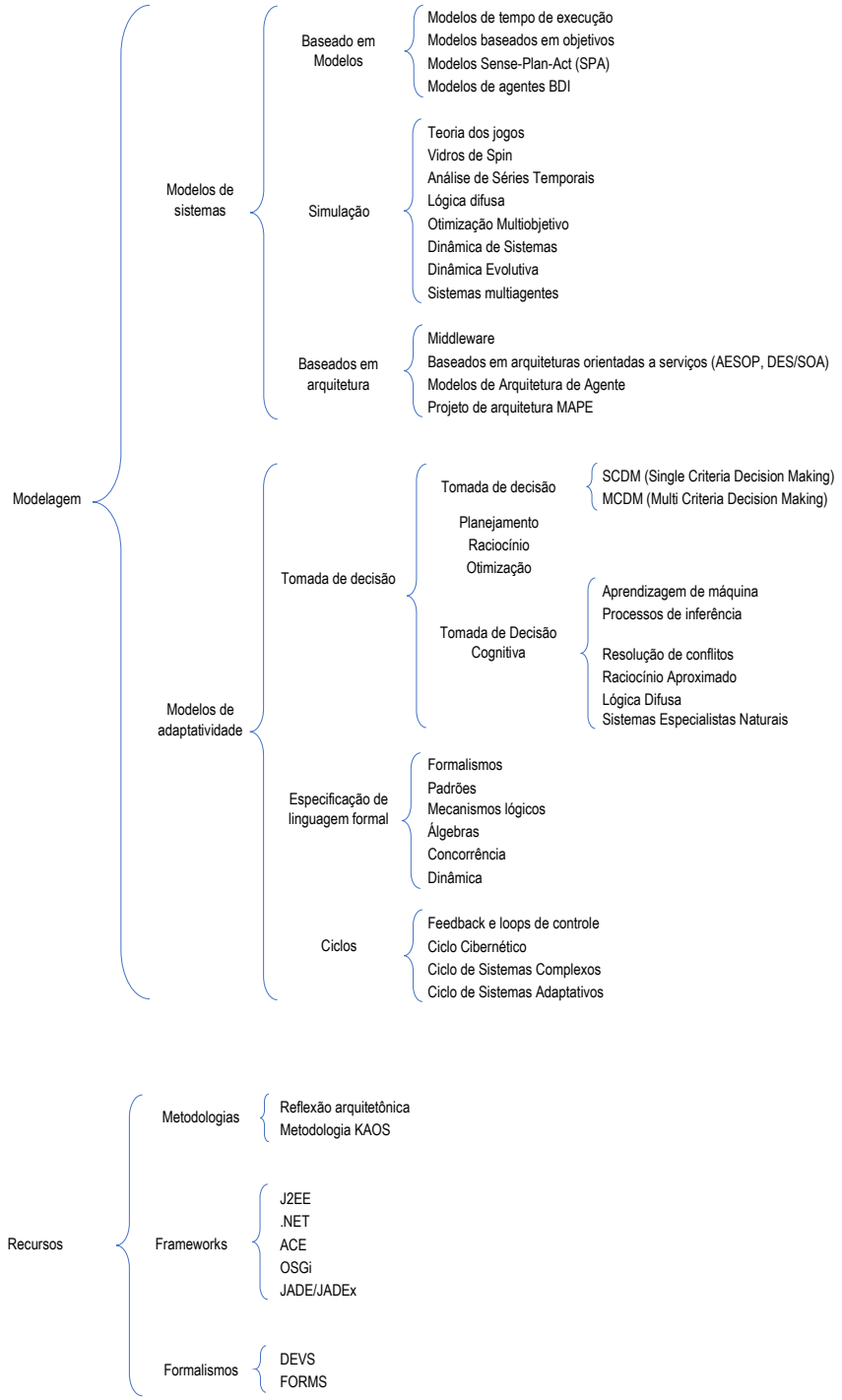


Figura 106 – Taxonomia de Sistemas Adaptativos no sub-nível de considerações computacionais focando em modelos e recursos.
O Autor (2018).

APÊNDICE D – PROPOSTA DE BIBLIOGRAFIA CATEGORIZADA

A bibliografia relacionada com nosso estudo do comportamento adaptativo e a suas implementações é ampla devido à natureza multidisciplinar do assunto. Assim, existem diferentes abordagens que procuram analisar o comportamento adaptativo, diferentes métodos e técnicas para implementar seus mecanismos, e diferentes ferramentas que auxiliam na tarefa de incorporar adaptatividade em plataformas específicas em casos concretos. Por praticidade a bibliografia é apresentada dividida em três categorias que facilitam o uso dela dependendo do tipo de informações que se procuram. As categorias usadas são Adaptatividade, Assuntos Técnicos, Recursos para revisões da Literatura.

A categoria de Adaptatividade está dividida nos seguintes tipos de trabalhos:

- *Aplicações*: coleção de trabalhos que relatam aplicações de Tecnologias Adaptativa em diferentes áreas de conhecimento, com diferentes abordagens, em diferentes níveis de abstração e com diferentes objetivos.
- *Definições*: coleção de trabalhos que propõem definições para o comportamento adaptativo de maneira geral, intuitiva e com terminologia básica.
- *Modelagens e Especificações*: coleção de diferentes abordagens nas áreas tecnológicas que visam modelar, especificar e implementar comportamento adaptativo em diferentes recursos e ferramentas. Os trabalhos nesta categoria são agrupados segundo a principal abordagem que desenvolvem.
- *Verificação, Validação, avaliação e testes*: coleção de trabalhos que apresentam métricas, métodos e estratégias de avaliação, tanto teórica (verificação) quanto técnica (validação), que permitem a comparação e aplicação de critérios de qualidade em software adaptativo.

A categoria de Assuntos Técnicos apresenta os conceitos próprios das áreas tecnológicas que permitem a modelagem, design e implementação da plataforma proposta. Os itens contidos nesta categoria estão relacionados com:

- *Fábrica de software*: definição, modelagem e implementação de fábricas de software voltadas para economias de escopo.
- *Componentes*: definição, modelagem, implementação e avaliação de componentes de software. Particularmente, os casos de extração de componentes com base em peças de software criadas em diferentes paradigmas, e a autodescrição de componentes são analisados com maior interesse.

Esta categoria da bibliografia será a mais enriquecida durante a etapa de desenvolvimento e implementação da proposta, pois estabelecerá o suporte teórico e prático das decisões tecnológicas.

A categoria de Recursos para revisões da literatura apresenta duas subcategorias: Estudos Bibliométricos e Revisões Sistemáticas da literatura. Os trabalhos indicados dentro da subcategoria de Estudos bibliométricos são divididos em:

- *Definições*: literatura básica sobre bibliometria, terminologia da área, indicadores e teorias sedimentadas.
- *Tutoriais*: recursos acadêmicos elaborados por instituições de pesquisa que orientam no uso de indicadores bibliométricos para a elaboração de diferentes tipos de análise.
- *Ferramentas*: recursos tecnológicos usados para suportar as atividades da elaboração de um estudo bibliométrico.
- *Exemplos*: artigos que realizam estudos bibliométricos sobre diferentes assuntos e apresentam exemplos de estrutura para a elaboração deste tipo de trabalho, assim como os principais índices bibliométricos utilizados.

Por seu lado, a subcategoria de Revisão Sistemática da Literatura apresenta trabalhos divididos em:

- *Definições*: literatura básica sobre revisão sistemática, definições, terminologia, propriedades, usos e vantagens.
- *Métodos*: literatura específica sobre diferentes metodologias para elaboração de revisões sistemáticas em diferentes áreas do conhecimento, os seus componentes gerais e particularidades no caso de Software.
- *Ferramentas*: recursos científicos e acadêmicos usados para suportar as atividades de revisão da literatura.
- *Exemplos*: artigos da área tecnológica que apresentam revisões sistemáticas.

Além da versão em formato de citação bibliográfica, uma versão em formato digital, e com processamento para cada item acompanhará este trabalho. É esperado que essa bibliografia processada evolua concomitantemente ao avanço no projeto.

D.1 Base de dados bibliográfica em formato digital

Durante a elaboração deste trabalho foi necessário revisar abundantes trabalhos, relacionados em diferente grau ao comportamento adaptativo. Nesse contexto, observamos

que a categorização dessas referências não só respondia a uma necessidade, mas também trazia uma série de vantagens e suporte para o trabalho de pesquisa. Algumas das vantagens identificadas são:

- Permite um trabalho mais eficiente e um mapeamento rápido para resgatar trabalhos de interesse segundo os critérios.
- Fornece fontes confiáveis sobre adaptatividade em diferentes categorias.
- Destaca trabalhos nas diversas categorias que podem ser pontos de partida para iniciantes no assunto.
- Permite a busca por literatura mais especializada segundo o assunto de interesse da pesquisa.

A bibliografia em formato digital seguirá um modelo de base de sados científica, procurando classificar os trabalhos consultados durante a elaboração desta tese de maneira expressiva. A bibliografia será elaborada em inglês, tanto para estar alinhada com a terminologia das bases de dados, quanto para fornecer uma maior visibilidade.

Assim, são utilizados quatro grupos de classificação: *Type of publication*, *Format of publication*, *Focus of the study* e *Approach*. A seguir, a Figura 107 apresenta as classificações para cada um dos grupos.

A Figura 108 é um extrato da bibliografia em formato digital consultada neste trabalho, apresentado aqui a maneira de exemplo de uso das classificações definidas.

A totalidade dos trabalhos revisados para a elaboração deste trabalho se encontra disponível no formato de bibliografia categorizada em https://sites.google.com/view/categorizedbibliography/home

Type of publication	
BO	Book
CH	Chapter
AR	Article in Journal or Proceedings
TR	Technical Report/ White Paper
AW	Academic Work (Dissertation and Theses)
PT	Presentation
IA	Internet Article

Focus	
AP	Application
EV	Evaluation
DF	Definition
CP	Concepts
SP	Specification and Modeling
OT	Other
VA	Various

Format of publication	
BO	Book
CH	Chapter
AR	Article
RW	Review or Study
SR	SLR or Roadmap
SY	Survey
TX	Taxonomy
CS	Case Study
OT	Other

Approach	
AC	Autonomic Computing
CAS	Complex Systems
CO	Concurrency
CW	Context-Aware
CT	Control Theory
CB	Cybernetics
CPS	Cyber-Physical Systems
DC	Dynamic Change
DR	Dynamic Reconfiguration
FM	Formal Method
LS	Legacy System
MB	Model-based
NN	Neural Networks
PN	Planning
AB	Architecture-based
RT	Run-time
SS	Self-* Systems
SoS	Systems of Systems
DB	Design-based
BI	Bio-inspired

Figura 107 – Grupos e Categorias utilizadas para a classificação de trabalhos relacionados com adaptatividade.

O Autor (2018).

ID	Title	Type	Form.	Focus	App.	Comment
1	Simulation and Adaptivity	CH	RW	VA, AP	FM	Rule-driven devices
2	Software Technology for Self-Adaptive Systems	PT	RW	SP, AP	SS	Smart Grids, IoT, Automatic maintenance
3	A development framework and methodology for self-adapting applications in ubiquitous computing environments	AR	AR	AP	SS	Ubiquitous Computing
4	AI-WSN: Adaptive and Intelligent Wireless Sensor Network	AR	AR	AP	CAS	Wireless Sensor Network
5	An overview of self-adaptive technologies within virtual reality training	AR	RW	VA, AP	SS	Virtual Reality Training
6	A self-adaptive hardware architecture with fault tolerance capabilities"	AR	AR	AP	SS	Hardware architecture
7	Challenges of Governance in Complex Adaptive Systems: A Case Study of U.S. Public Education	AR	CS	AP	CAS	USA Public Education
8	Complex Adaptive Systems Drive Innovations in Synthetic Biology	AR	CS	AP	CAS	Synthetic Biology
9	DeltaIoT: A Self-adaptive Internet of Things Exemplar	AR	AR	AP	SS	IoT
10	Hospitals as complex adaptive systems: A case study of factors influencing priority setting practices at the hospital level in Kenya	AR	CS	AP	CAS	Priority settings practices at the hospital level in Kenya
11	NeuralSens: A neural network based framework to allow dynamic adaptation in wireless sensor and actor networks	AR	AR	AP	NN	Wireless Sensor and actor Networks
12	Research Challenges for the Internet of Things: What Role Can OR Play?	AR	RW	VA, AP	CAS	IoT
13	SmartApps: Middle-ware for Adaptive Applications on Reconfigurable Platforms	AR	AR	SP, AP	RT	Smart apps
14	Web services management: a survey	AR	RW	VA,CP	AB	Self-organization of Web services
15	Self-managed Cells for Ubiquitous Systems	AR	AR	VA, AP	SoS	Ubiquitous Computing
16	Node-Adapt, Path-Adapt and Tree-Adapt: Model-Transfer Domain Adaptation for Random Forest	AR	AR	VA, AP	FM	Domain Adaption for learning classifiers
17	Context specific adaptation grammars for climate adaptation in urban areas	AR	AR	VA, SP	FM	Specification of grammar for adaptations in the design of urban areas for climate changes
18	The three stages of Intelligent Organization	IA	AR	DF, AP	BI	Adaptivity as part of organizations and management
19	Adaptive Leadership vs. Adapting	IA	AR	DF, AP	BI	Adaptivity as part of leadership
20	An Adaptive Framework for Discovery and Mining of User Profiles from Social Web-Based Interest Communities	CH	CH	VA, AP	AB	Adaptive framework for data minig
21	Self-Adaptive Software: Applications	BO	BO	VA, AP	SS	Application of Self-Adaptive Software
22	Adaptive Automa for Mapping Unknown Environments by Mobile Robots	AR	AR	SP, AP	MB	Application of Adaptivity to Mapping Problem
23	Adaptive Non-Deterministic Decision Trees: General Formulation and Case Study	AR	AR	DF, AP	FM	Definition of Adaptive Decision Trees
24	Using Adaptive Automata in a Multi-paradigm Programming Environment	AR	AR	VA, AP	FM	Application of Adaptive Automata
25	What Is Adaptivity? Does It Improve Performance?	AR	AR	VA, AP	MB	Using Adaptivity in e-Learning
26	Adaptive Computation: The Multidisciplinary Legacy of John H. Holland	AR	RW	VA, DF	CAS	Holland Legacy
27	Part I: Adaptive Systems	AR	RW	VA, DF	CAS	CAS Conference
28	Terminology for adaptive behavior	AR	OT	DF	CB	International Encyclopedia of Systems and Cybernetics
29	Computer Science: Artificial Intelligence and Economics: The Rise of Formalism and Behaviorism	IA	AR	DF		Formalism and Behaviorism
30	An introduction to the philosophy of mathematics	BO	BO	DF		Formalism in Philosophy of Mathematics
31	Thinking about Mathematics: The Philosophy of Mathematics	BO	BO	DF		Formalism in Philosophy of Mathematics
32	Remarks on the definition and nature of mathematics	CH	CH	DF		Formalism as definition on Mathematics
33	Intuitionism and Formalism	CH	CH	DF		Definitions in Mathematics
34	The formalism foundations of Mathematics	CH	CH	DF		Definitions in Mathematics
35	Philosophy of Mathematics	CH	CH	DF		Stanford Encyclopedia of Philosophy

ID	Title	Type	Form.	Focus	App.	Comment
36	Formalism vs. Hermeneutics	IA	AR	DF		Hermeneutics and Formalism
37	What is the meaning of “formalism” in a scientific paper? What is the structure of such paper?	IA	AR	DF		Formalism in scientific work
38	A Case Study on Formal Verification of Self-Adaptive Behaviors in a Decentralized System	CH	CS	EV	SS	Formal Verification
39	Cost-effective regression testing through Adaptive Test Prioritization strategies	AR	AR	EV	FM	Adaptive regression testing strategy. Adaptivity in the Method of evaluation
40	Adaptive Runtime Verification	AR	AR	EV	FM	Adaptive Runtime Verification. Adaptivity in the Method of evaluation
41	An evaluation of multi-model self-managing control schemes for adaptive performance management of software systems	AR	AR	EV	CT	Self-managing control system. Adaptivity in the Method of evaluation
42	Evaluating the adaptivity of computing systems	AR	AR	EV, SP	FM	Formal framework to measure and evaluate adaptivity in software
43	Formal Specification and Analysis of Autonomous Systems under Partial Compliance	AR	AR	EV, SP	FM	Formal specification, design and evaluation of systems that deal with uncertainty
44	Layered evaluation of interactive adaptive systems: framework and formative methods	AR	AR	EV	MB	Evaluation of interactive adaptive systems through layers
45	Perpetual Assurances in Self-Adaptive Systems	AR	AR	EV	SS	Verification, Validation and Proofs of Self-Adaptive Systems
46	Self-adaptive Software Needs Quantitative Verification at Runtime	AR	AR	EV	SS	Quantitative Verification at Runtime
47	Software Engineering for Self-Adaptive Systems: Assurances	TR	AR	EV	SS	Various approaches and techniques to solve different aspects of assurances in Self-adaptive systems
48	Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications	AR	RW	EV	CPS	Specification-Based (formal methods) monitoring
49	Testing the robustness of controllers for self-adaptive systems	AR	AR	EV	SS	Testing robustness of controllers within Rainbow framework
50	Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems	CH	RW	EV	SS	Software validation and verification (V&V) for SAS
51	Verification Points for Self-adaptive Systems	AR	AR	EV	CAS	Verification of Self-Adaptive Systems
52	Verifying Stochastic Behaviors of Decentralized Self-Adaptive Systems: A Formal Modeling and Simulation Based Approach	AR	AR	EV	FM	Verification of Self-Adaptive Systems
53	Towards Run-time Adaptation of Test Cases for Self-adaptive Systems in the Face of Uncertainty	AR	AR	EV	FM	Adaptation of test-cases for SAS
54	Towards systematic model-based testing of self-adaptive software	AR	AR	EV	MB	Failure Mode and Effects Analysis on a generic perspective on SAS
55	Towards Testing Self-organizing, Adaptive Systems	AR	AR	EV	DB	The Corridor Enforcing Infrastructure (CEI) pattern for Self-Organizing Self-Adaptive sw
56	Self-adaptive systems - A survey of current approaches, research challenges and applications	AR	TX	VA	SS	
57	Towards a Taxonomy of Software Change: Research Articles	AR	TX	CP	DC	
58	A Comprehensive Technological Survey on Dependable Self-Managing CPS: The Decade of Researches on Correctness and Dependability	AR	SY	VA,CP	SS	Some Taxonomies for Cyber-Physical Systems
59	Review on Requirements Modeling and Analysis for Self-Adaptive Systems - A Ten-Year Perspective	AR	RW	VA	SS	Mix between an SLR and a Taxonomy article. Big revision of work and comparative data
60	A Survey of Formal Methods in Self-adaptive Systems	AR	SR	SP	FM	Several criteria for modeling and testing adaptivity
61	A Systematic Literature Review of Requirements Engineering for Self-Adaptive Systems	AR	SR	SP	SS	
62	Analysing monitoring and switching problems for adaptive systems	AR	AR	DF	FM	Definitions of monitoring and switch on formal language
63	A Systematic Literature Review of Requirements Modeling and Analysis for Self-adaptive Systems	AR	SR	VA	SS	Mapping between topics and Modeling methods
64	A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems	AR	SR	VA	AB	Identification of common paradigms, architectural models, and tools
65	Claims and supporting evidence for self-adaptive systems - a literature study	AR	SR	VA	SS	Analysis of characteristics of publications about adaptivity and experients
66	Software Engineering of Self-Adaptive Systems - An Organised Tour and Future Challenges	AR	SY	VA	SS	Guided tour in 6 waves about self-adaptive systems, the different approaches, evolution and challenges
67	Self-adaptive concurrent components	AR	AR	SP,VA	CO	Composition of concurrent adaptive components within a platform

Figura 108 – Exemplo de uso da categorização bibliográfica.
O Autor (2018).