Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia de Produção

Arquitetura de software de domínio específico para apoio ao gerenciamento ágil
de projetos distribuídos

Guilherme Delefrate Martins

São Carlos
2011

Guilherme Delefrate Martins

Arquitetura de software de domínio específico para apoio ao
gerenciamento ágil de projetos distribuídos

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Engenharia de Produção.

Área de concentração: “Processos e Gestão de Operações”.

Orientador: Prof. Dr.. Daniel Capaldo Amaral

São Carlos

2011

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.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

M386a Martins, Guilherme Delefrate.
Arquitetura de software de domínio específico para
apoio ao gerenciamento ágil de projetos distribuídos. /
Guilherme Delefrate Martins ; orientador Daniel Capaldo
do Amaral. São Carlos, 2011.

Dissertação (Mestrado - Programa de Pós-Graduação em
Engenharia de Produção e Área de Concentração em
Processos e Gestão de Operações)-- Escola de Engenharia
de São Carlos da Universidade de São Paulo, 2011.

1. Gerenciamento de projetos. 2. Software. 3.
Arquitetura de software. 4. Painel visual. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **GUILHERME DELEFRATE MARTINS**

Título da dissertação: **Arquitetura e solução de painel eletrônico para o gerenciamento ágil de projetos.**

Data da defesa: 21/09/2011:

Comissão Julgadora:

Resultado:

Prof. Associado **Daniel Capaldo Amaral (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

APPROVADO

Prof. Titular **Henrique Rozenfeld**
(Escola de Engenharia de São Carlos/EESC)

APPROVADO

Prof.^a. Dr.^a. **Elisa Yumi Nakagawa**
(Instituto de Ciências Matemáticas e de Computação/ICMC/USP)

APPROVADO

Coordenador do Programa de Pós-Graduação em Engenharia de Produção:
Prof. Titular **Henrique Rozenfeld**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo Cesar Lima Segantine**

AGRADECIMENTOS

Agradeço primeiramente à minha família, pais, irmã, avós, tios e primos, pelo apoio em mais essa etapa da minha vida. Mas principalmente aos meus pais, que sempre me apoiaram em minhas decisões e atitudes.

Ao meu orientador, Prof. Dr. Daniel Capaldo Amaral, pela orientação, ensinamentos, apoio inestimável e paciência durante todo este percurso.

Ao Prof. Tit. Henrique Rozenfeld, pelos ensinamentos e exemplo de trabalho.

Ao sincero e querido amigo Bruno, que me acompanhou e apoiou durante a minha caminhada no mestrado, e que irá me acompanhar em outras etapas da minha vida.

Aos amigos de longa data, Aline, Ana, Bruna, Raphael, Alessandro, Neto e Maysa, que apesar da distância, sempre me apoiaram.

Aos amigos Breno, Leandro, Larissa e Eliane, companheiros de vários episódios de boas risadas.

Aos amigos e companheiros de pesquisa: João, Edivandro, Lucelindo, Samuel, Janaina, Maicon, Mauro, Sayuri, Daniela, Gabriela, Carol, Vitor, Fernando, Rafael e André, pelas experiências e bons momentos compartilhados. E em especial à Camila, amiga e companheira de pesquisa pelo inestimável apoio em meu projeto. Aos demais integrantes do grupo de pesquisa.

À querida amiga Francis, pelo apoio técnico e administrativo.

Aos funcionários do SEP pela ajuda.

À Universidade de São Paulo, pelo ensino gratuito e de excelente qualidade.

À Capes, CNPq e ao Ministério da Ciência e Tecnologia pelo apoio financeiro.

RESUMO

MARTINS, G. D. **Arquitetura de software de domínio específico para apoio ao gerenciamento ágil de projetos distribuídos**. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

O gerenciamento ágil de projetos (APM) é uma abordagem que vem se disseminando rapidamente, em especial para o caso de produtos inovadores. A contribuição desta abordagem está em propor meios para simplificar e fornecer maior flexibilidade nos processos de trabalho, permitindo maior quantidade de mudanças no decorrer do projeto. Uma das estratégias é o uso de gestão visual, por meio de painéis, com adesivos autocolantes. Existem várias inconveniências do uso destes painéis e, paralelamente, vivencia-se a popularização de novas tecnologias de interação humano-computador painéis digital *touchscreen*. O objetivo deste projeto é propor uma nova arquitetura de software de domínio específico para a construção de sistemas de gerenciamento de projetos distribuídos, considerando o planejamento iterativo e visual do gerenciamento ágil de projetos e segundo o paradigma da sinalização digital; ao mesmo tempo em que os dados do projeto são enviados para Sistemas Integrados de Gerenciamento de Projetos tradicionais. Esta pesquisa combina vários procedimentos para o seu completo desenvolvimento, em seis etapas. Duas de levantamento bibliográfico, geral e específico, uma de estudo de tecnologias, duas principais para o desenvolvimento da arquitetura e protótipo e, por fim, a construção de um exemplo de aplicação para testar sua exequibilidade. A viabilidade técnica da proposta foi comprovada com um protótipo utilizando as tecnologias Adobe Flex, PHP e o software de gerenciamento de projetos (SGP) dotProject. Os resultados demonstram que a arquitetura proposta é viável e pode trazer uma série de vantagens frente o uso de softwares tradicionais e os softwares voltados para o APM.

Palavras-chave: gerenciamento de projetos; software; arquitetura de software; painel visual.

ABSTRACT

MARTINS, G. D. **Domain-specific software architecture to support distributed agile project management.** M. Sc. (Dissertation) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2011.

The Agile Project Management (APM) approach was rapidly disseminated, especially in the case of innovative products. The contribution of this approach is to propose methods to simplify and provide flexibility in work processes with considerable amount of change throughout the project. The strategy is use visual management with white panels and stick notes, however, there are several concerns of using these panels. Moreover it is observed the popularization of new technologies for human computer interaction like the touchscreen panels. The main objective of this project is to propose a new Domain-Specific Software Architecture for building distributed project management systems considering iterative and visual planning of agile project management and according to the paradigm of digital signage while the project data is sent to the traditional Integrated Systems Project Management. This research combines various procedures to its development in six phases: two phases for bibliographical research, one phase for study the technologies, another two phases for the development of architecture and prototype, and finally one phase for build an application and test its workability. The prototype was developed using Adobe Flex technologies, PHP and project management software (PMS) dotProject to test the technical feasibility of the proposal. The results show that the proposed architecture is feasible and can bring a number of advantages over the use of traditional software and software aimed at the APM.

Palavras-chave: project management; software; software architecture; visual panels.

LISTA DE QUADROS

Quadro 1. Princípios do APM	6
Quadro 2. Requisitos funcionais para softwares de gerenciamento ágil.	14
Quadro 3. Requisitos não funcionais para softwares de gerenciamento ágil.	15
Quadro 4. Síntese das publicações específicas sobre softwares de gerenciamento ágil de projetos	19
Quadro 5. Elementos e características dos painéis físicos.....	52
Quadro 6. Pacotes de documentação por perfil de usuário da arquitetura.	56
Quadro 7. Requisitos funcionais.....	66
Quadro 8. Requisitos não funcionais.....	67
Quadro 9. Objetivos do PVC e do SIPRO.....	68
Quadro 10. Usuários do sistema	69
Quadro 11. Caso de uso 1.....	70
Quadro 12. Caso de uso 2.....	70
Quadro 13. Caso de uso 3.....	70
Quadro 14. Caso de uso 4.....	70
Quadro 15. Caso de uso 5.....	70
Quadro 16. Caso de uso 6.....	70
Quadro 17. Caso de uso 7.....	71
Quadro 18. Caso de uso 8.....	71
Quadro 19. Relação entre requisitos, módulos e objetivos do software.....	77
Quadro 20. Comparação entre as tecnologias AJAX e Adobe Flex.....	85
Quadro 21. Relação entre os módulos da arquitetura e os componentes do PVC.....	91

LISTA DE FIGURAS

Figura 1. Quadro de atividades para planejamento e controle de atividades	8
Figura 2. Utilização de softwares de gerenciamento de projetos.	20
Figura 3. Publicações com o termo Digital Signage no Web of Science.	33
Figura 4. Painel Visual de Planejamento e Controle de Projetos (PVPCP).	46
Figura 5. Quadro de Planejamento Fino Semanal (QPFS).....	47
Figura 6. Product Backlog. Fonte: SCHWABER, 2004.....	48

Figura 7. Sprint Backlog. Fonte: SCHWABER, 2004.....	49
Figura 8. Action Item and Tasks Tracking Process. Fonte: Chin (2004)	50
Figura 9. Estrutura da documentação da arquitetura para Sistemas de Gerenciamento Ágil de Projetos Distribuídos.....	55
Figura 10. Modelo de referência para o PDP. Fonte: Rozenfeld et al (2006).....	57
Figura 11. Visão geral da solução.	58
Figura 12. Contexto do sistema para gerenciamento ágil de projetos distribuídos	68
Figura 13. Diagrama de Casos de Uso.....	71
Figura 14. Cenário de Planejamento do Projeto.	73
Figura 15. Cenário de Sincronização de Dados com o SIPRO.....	74
Figura 16. Visão de Módulos de Software.....	78
Figura 17. Diagrama de Implantação da Arquitetura de Software de Domínio Específico	80
Figura 18. Visão de Camadas (Geral).....	81
Figura 19. Visão de Camadas (Módulo de Planejamento e Controle).....	82
Figura 20. Diagrama de Implantação do Protótipo	86
Figura 21. Mockup do PVC.....	87
Figura 22. Métodos do SIPRO.....	88
Figura 23. Descrição técnica da chama getProjects do SIPRO.....	89
Figura 24. Código para o método getProjects do SIPRO.	90
Figura 25. Código do módulo ProjetosLista.mxml.....	93
Figura 26. Tela principal do PVC com o painel de seleção de projetos em destaque.	94
Figura 27. Tela de projetos do dotProject.	94
Figura 28. Tela de configuração do projeto.	95
Figura 29. Janela de Planejamento e Controle do PVC.	96
Figura 30. Janela de criação de nova tarefa do PVC.	97
Figura 31. Janela de edição de tarefa do PVC.	98
Figura 32. Cartões autoadesivos utilizados no PVC.....	98
Figura 33. Totem utilizado na pesquisa.	99

LISTA DE TABELAS

Tabela 1. Lista de artigos encontrados.....	12
Tabela 2. Resultado da busca completa (207 artigos) por tipo.	17
Tabela 3. Quantidade de artigos por ano resultante da busca completa.	17

Tabela 4. Quantidade de artigos que tratam de software por tipo.	18
Tabela 5. Quantidade de artigos que tratam de software por ano.....	18
Tabela 6. Principais periódicos que discutem temas relacionados com software para Gerenciamento Ágil.....	19
Tabela 7. Modelos de painéis físicos analisados.	44
Tabela 8. Perfil de usuários da documentação da arquitetura.	53

LISTA DE ABREVIATURAS E SIGLAS

ADL	<i>Architecture Description Languages</i>
API	<i>Application Programming Interface</i>
AJAX	<i>Asynchronous Javascript And XML</i>
APM	<i>Agile Project Management</i>
CPM	<i>Critical Path Method</i>
CSCD	<i>Computer Supported Collaborative Desig</i>
CSCW	<i>Computer supported cooperative work</i>
CVS	<i>Concurrent Version System</i>
DAI	<i>Distributed Artificial Intelligence</i>
DSSA	<i>Domain-Specific Software Architecture</i>
EPM	<i>Enterprise Project Management</i>
GP	Gerenciamento de projetos
HTML	<i>HyperText Markup Language</i>
IHC	Interação Humano-Computador
LCD	<i>Liquid Crystal Display</i>
LED	<i>Light Emitting Diode</i>
MXML	<i>Minimal XML</i>
MFE	Modelo de Fases e Entregas
OLED	<i>Organic Light-Emitting Diode</i>
PDP	Processo de Desenvolvimento de Produtos
PERT	<i>Program Evaluation and Review Technique</i>
PHP	<i>PHP: Hypertext Preprocessor</i>
PM	<i>Project Management</i>
PMS	<i>Project Management Software</i>

PVC	<i>Pro Visual Control</i>
PVPCP	Painel Visual de Planejamento e Controle de Projetos
QPFS	Quadro de Planejamento Fino Semanal
RIA	<i>Rich Internet Application</i>
RPC	<i>Remote Procedure Call</i>
SGBD	Sistema de Gerenciamento de Banco de Dados
SGP	Software para Gerenciamento de Projetos
SOAP	<i>Simple Object Access Protocol</i>
SQL	<i>Structured Query Language</i>
SVG	<i>Scalable Vectorial Graphics</i>
UML	<i>Unified Modeling Language</i>
VML	<i>Vector Markup Language</i>
WBS	<i>Work Breakdown Structure</i>
WSDL	<i>Web Service Definition Language</i>
XML	<i>Extensible Markup Language</i>

Sumário

1. INTRODUÇÃO	1
2. REVISÃO BIBLIOGRÁFICA	5
2.1. GERENCIAMENTO ÁGIL DE PROJETOS	5
2.2. SOFTWARES PARA GERENCIAMENTO DE PROJETOS E AGILIDADE.....	8
2.3. ARQUITETURA DE SOFTWARE	22
2.4. SINALIZAÇÃO DIGITAL E PAINÉIS <i>TOUCHSCREEN</i>	32
2.5. DESAFIOS E OPORTUNIDADES EM SOFTWARES PARA GERENCIAMENTO ÁGIL DE PROJETOS	35
3. MÉTODO DE PESQUISA	39
3.1. OBJETIVO	39
3.2. JUSTIFICATIVA	39
3.3. ETAPAS E MÉTODOS DA PESQUISA	40
4. ESTABELECIMENTO DA ARQUITETURA DE DOMÍNIO ESPECÍFICO PARA SISTEMAS DE GESTÃO ÁGIL DE PROJETOS DISTRIBUÍDOS.....	43
4.1. FUNCIONALIDADES DE UM PAINEL ELETRÔNICO BASEADO EM PAINÉIS FÍSICOS.....	43
4.1.1. Levantamento dos modelos de painéis físicos existentes	44
4.1.2. Elementos dos painéis físicos.....	51
4.2. ESTRUTURA DA DOCUMENTAÇÃO	52
5. ARQUITETURA DE REFERÊNCIA	61
5.1. REQUISITOS DA ARQUITETURA.....	61
5.2. CONTEXTO DO SISTEMA	67
5.2.1. Diagrama de contexto.....	67
5.2.2. Atores do Sistema.....	69
5.3. CASOS DE USO E DIAGRAMA DE CASOS DE USO	69
5.4. CENÁRIO DA SOLUÇÃO	72
6. AMBIENTE E INFRA-ESTRUTURA	75
6.1. TECNOLOGIAS NECESSÁRIAS	75
6.2. VISÃO DOS MÓDULOS	76
6.3. DIAGRAMA DE IMPLANTAÇÃO	79
6.4. DIAGRAMA DE CAMADAS	80
7. AVALIAÇÃO DA ARQUITETURA POR MEIO DE PROTÓTIPO.....	83
7.1. TECNOLOGIAS ADOTADAS.....	83

7.2.	DESENVOLVIMENTO DA INTERFACE E MÓDULOS	86
7.3.	DESENVOLVIMENTO DO SIPRO	87
7.4.	DESENVOLVIMENTO DO PVC.....	91
8.	CONCLUSÕES.....	100
9.	REFERÊNCIAS.....	103

1. INTRODUÇÃO

O gerenciamento por projetos (GP) é o modelo dominante em muitas organizações (WINTER et al, 2006) e há um conjunto significativo de técnicas, ferramentas e conceitos que foram desenvolvidos para apoiar esse tipo de gerenciamento, elaborados a partir dos anos 50 (HIRSCHFELD, 1985; KERZNER, 2002).

O desenvolvimento do gráfico de Gantt, o método do caminho crítico e a análise de rede (PERT), bem como a criação das primeiras associações profissionais na década de 70, são marcos que podem representar o início da teoria de Gerenciamento de Projetos. Essas técnicas, ferramentas e conceitos não sofreram alterações significativas até os dias atuais.

Atualmente, vem surgindo opiniões de que são necessárias mudanças, principalmente nos casos de projetos de produtos inovadores. Pons (2008) afirma que necessidade de novos métodos para produtos complexos e inovadores e Maylor (2001) aponta indícios de projetos aonde as ferramentas são utilizadas de forma burocrática e que não apóiam efetivamente o gerenciamento de projetos.

Um dos problemas é o uso de software. White e Fortune (2002) realizaram um levantamento (*survey*) detalhado sobre GP e identificam os atuais softwares de GP como uma das principais limitações entre os métodos/ferramentas/técnicas do gerenciamento de projetos, pois eles seriam inadequados, principalmente para projetos complexos (CICMIL et al, 2006; WHITE; FORTUNE, 2002). Chin (2004), Highsmith (2004) e Cockburn (2001) são exemplos de autores que vêm se dedicando ao desenvolvimento de novas teorias para suplantar esse problema. Eles foram co-autores do denominado Manifesto Ágil (AGILE ALLIANCE, 2001; ANDERSON et al, 1998; BECK et al, 2001), com proposta de um novo enfoque de GP, chamado de *Agile Project Management* (APM), para as empresas que atuam em mercados inovadores. No Brasil, há o exemplo de Dias (2005). Esses autores têm contribuído com o

desenvolvimento de um conjunto de diretrizes, práticas e métodos diferenciais, voltados para o GP de produtos inovadores.

Entre as novidades propostas, destacam-se duas. O princípio da autodisciplina propõe que as tarefas de controle sejam simplificadas por meio da ampliação da disciplina e auto-organização. Isso é, ao invés do controle realizado pelo gerente, cada profissional seria proativo na detecção do problema e replanejamento. A segunda é a utilização de técnicas de planejamento mais simples e baseadas no conceito de gestão à vista, como painéis e gráficos, públicos, no qual a equipe pode interagir em um mesmo dispositivo.

O principal instrumento para troca de informações são os painéis visuais, normalmente construídos com lousa tradicional branca, nos quais as informações sobre tarefas e prazos são afixadas por meio de etiquetas autoadesivas (*post-its*). O benefício notado é que tais quadros favorecem a interação entre os membros, incentivam o autocontrole e provêm planos em um formato fácil de serem consultados pelo gerente (Amaral et al, 2011). As limitações desse sistema podem ser observadas no trabalho de Conforto & Amaral (2010): dificuldade de atualizar dados, perda do histórico e limitações quanto ao espaço geográfico (os membros do time devem estar fisicamente unidos), dificuldade de analisar a capacidade da empresa.

O objetivo da pesquisa é propor uma nova arquitetura de software de domínio específico para a construção de sistemas de gerenciamento de projetos distribuídos, considerando o planejamento iterativo e visual do gerenciamento ágil de projetos e segundo o paradigma da sinalização digital. Distribuídos refere-se a equipes geograficamente distribuídas e também a informações distribuídas, pois os dados do projeto provêm e são enviados para diferentes Softwares de Gerenciamento de Projetos tradicionais. O projeto envolve a documentação e a proposta de uma arquitetura de domínio específico e um protótipo de um software que comprova a exequibilidade da arquitetura.

O documento está dividido em oito capítulos. Este primeiro introduz o tema e apresenta o trabalho, descrevendo o problema de pesquisa e objetivos. O Capítulo 2 inicia-se com uma breve revisão sobre gerenciamento ágil de projetos e apresenta seus principais conceitos (Seção 2.1). Já na Seção 2.2 é apresentada uma revisão sobre softwares para gerenciamento ágil de projetos e o estado da arte dos sistemas comerciais e da literatura científica na área, demonstrando que há uma oportunidade de contribuição com o advento da abordagem do gerenciamento ágil de projetos. Em seguida, apresentam-se os conceitos fundamentais de duas tecnologias utilizadas para solução do problema proposto: arquiteturas de software (Seção 2.3) e a Sinalização Digital (Seção 2.4). Ao final do capítulo justifica-se a lacuna apresentada no trabalho utilizando as observações da literatura e conceitos apresentados. Trata-se da seção Desafios e oportunidades em softwares para gerenciamento ágil de projetos, Seção 2.5.

O Capítulo 3 apresenta o método de pesquisa e está dividido seguindo a ordem comumente empregada em publicações científicas: objetivo, justificativa e, por fim, etapas e métodos de pesquisa.

O Capítulo 4 estabelece a arquitetura de domínio específico para sistemas de gestão ágil de projetos distribuídos. Inicialmente são apresentadas as funcionalidades e elementos de painéis para gerenciamento ágil de projetos, por meio de uma revisão dos painéis físicos propostos na literatura. Trata-se de uma contribuição teórica importante, a análise teórica dos conteúdos e formas de apresentação dos painéis utilizados no gerenciamento ágil de projetos, e que fundamentaram a arquitetura proposta (Seção 4.1). Em seguida, é apresentada a estrutura da documentação da arquitetura de domínio específico. Seu objetivo foi facilitar a consulta da arquitetura para os diferentes usuários (Seção 4.2).

O Capítulo 5 apresenta a descrição da arquitetura para a construção de sistema para gerenciamento ágil de projetos, proposta principal do trabalho. A arquitetura é apresentada

por meio de uma arquitetura de referência, na qual, serve como guia para o desenvolvimento de uma nova classe de softwares de gerenciamento de projetos.

O Capítulo 6 apresenta o ambiente e infra-estrutura da arquitetura proposta, que corresponde a visão da implantação na arquitetura proposta, isto é, as especificações para o seu funcionamento. Conforme teoria apresentada na seção de revisão bibliográfica.

O Capítulo 7 apresenta o segundo resultado fundamental do trabalho, um protótipo de software desenvolvido segundo a arquitetura proposta e que serve como prova de conceito da arquitetura, demonstrando a viabilidade técnica das especificações propostas.

Finaliza-se com o Capítulo 8, que descreve as considerações finais e conclusões obtidas dos resultados apresentados.

2. REVISÃO BIBLIOGRÁFICA

O objetivo da revisão é apresentar uma síntese sobre os três assuntos que fundamentam o trabalho: os softwares para gerenciamento ágil de projetos, arquitetura de software e a sinalização digital. Cada um deles é apresentado em uma seção separada e, ao final, uma síntese que apoia o desenvolvimento do trabalho.

O conteúdo do Capítulo está dividido em quatro seções. A primeira seção apresenta uma definição para o termo gerenciamento ágil de projetos. A segunda seção apresenta as mudanças que o novo paradigma representa para os softwares destinados a apoiar a atividade de GP. A terceira seção apresenta os conceitos básicos de arquitetura de software que foram utilizados para o desenvolvimento da arquitetura proposta, esses conceitos são necessários, pois para inovar neste campo é necessário uma nova classe de sistemas. A quarta seção apresenta os conceitos de sinalização digital e distribuição de informações, que exploram a utilização de painéis eletrônicos, fundamentais para aplicação dos conceitos propostos na nova arquitetura. A última seção apresenta os desafios e oportunidades identificados pela observação da literatura.

2.1. GERENCIAMENTO ÁGIL DE PROJETOS

O termo Gerenciamento Ágil de Projetos (APM) está em evidência desde 2001, a partir de um manifesto proposto por profissionais e pesquisadores da comunidade internacional de desenvolvimento de softwares. É uma resposta para as pressões por inovação, necessidade de reduzir ciclos de desenvolvimento de produtos, concorrência acirrada e de adaptação a um contexto dinâmico que cada vez mais revelavam a diminuição da eficiência da gestão clássica de projetos quando deparada a esse novo cenário.

O princípio fundamental do APM é transferir a ênfase do planejamento para a execução, fase em que são tomadas grande parte das decisões que determinam o sucesso ou

fracasso do projeto. Highsmith (2004) define o gerenciamento ágil de projetos como “... um conjunto de valores, princípios e práticas que auxiliam a equipe a entregar produtos ou serviços de valor em um ambiente desafiador”.

Diversos autores (ANDERSON et al, 1998; BOEHM, 2002; CHIN, 2004; COCKBURN, 2002; COHN; FORD, 2003; SANJIV; WOODCOCK, 2003) compartilham do mesmo enfoque do APM, definindo-o como uma abordagem que busca a flexibilidade, simplicidade, iterações para agregar valor ao produto de forma incremental. O APM é orientado ao resultado, permitindo adaptar o processo, absorver mudanças de requisitos e escopo do produto (ANGIONI, 2006).

Outros princípios que servem como guia para as práticas do APM foram definidos. Esses princípios, de acordo com Highsmith (2004, p.27), são divididos em duas categorias: uma relacionada ao produto e clientes e a outra relacionada ao gerenciamento (vide Quadro 1).

Quadro 1. Princípios do APM

Categoria	Princípio
Cliente/Produto	Entregar valor ao cliente
	Empregar iterações baseadas em características
	Buscar excelência técnica
Gerenciamento	Encorajar a exploração
	Formar times adaptativos (auto-organizados e autodisciplinados)
	Simplificar

Fonte: (BENASSI & AMARAL, 2007).

Highsmith (2004) apresenta um modelo de gestão ágil de projetos baseado em cinco objetivos essenciais: inovação contínua; adaptabilidade do produto; tempos de entregas reduzidos (ciclos de entrega); adaptabilidade do processo e das pessoas; e resultados confiáveis. Baseado nesses objetivos, o autor destaca as cinco fases do APM:

- *Visão*: determinar a visão do produto, o escopo do projeto, a comunidade do projeto, e a definição de como a equipe irá trabalhar, interagir;
- *Especulação*: definir plano de entregas, cronograma e iterações conforme a visão;
- *Exploração*: entregar os componentes do produto (funcionalidades definidas na fase especulação) em ciclos de entregas, minimizando riscos e incertezas;
- *Adaptação*: rever os resultados entregues, analisar a situação atual e o desempenho do time. Adaptar esses resultados se necessário; e
- *Encerramento*: encerrar o projeto, finalizar tarefas pendentes e transferir as lições aprendidas mais importantes.

As fases de visão, exploração e adaptação repetem-se em ciclos curtos de tempo, chamados iterações, proporcionando flexibilidade para adaptar e absorver mudanças no projeto. Apesar do enfoque na execução e adaptação, isso não significa negligenciar o controle do projeto. Esse deve ser adaptado às necessidades e características do APM, tendo como premissa a flexibilidade e agilidade, e possuir indicadores de desempenho que forneçam informações de valor para os *stakeholders*, equipe de projeto e clientes.

Outro aspecto fundamental no APM é a Gestão da Comunicação. Segundo Coram & Bohner (2005), os métodos ágeis valorizam mais a comunicação informal do que a volumosa documentação tradicional. Para eles, a comunicação rápida facilita a propagação da informação entre toda a equipe e os *stakeholders*.

Um dos diferenciais no quesito comunicação é o conceito de gestão visual. Autores do APM como Highsmith(2004) e Cohn(2005) propõem o uso de quadros visuais para programar a execução de tarefas utilizando pequenos pedaços de papel (cartões) ou até mesmo escrevendo em quadros brancos (Figura 1).

Story	Para Fazer	Testes ok	Em processo	Para verificar	Horas
<div>Como usuário, eu posso... 5</div>	<div>Codificar... 8</div> <div>Codificar... 5</div> <div>Testar... 6</div>	✓	<div>Codificar... 6</div> <div>Codificar... 4</div>	<div>Codificar... 4</div>	33
<div>Como usuário, eu posso... 2</div>	<div>Codificar... 8</div> <div>Codificar... 5</div>				13
<div>Como usuário, eu posso... 3</div>	<div>Codificar... 3</div> <div>Codificar... 6</div>	✓	<div>Codificar... 4</div>		13

Figura 1. Quadro de atividades para planejamento e controle de atividades
Fonte: Cohn (2005, p. 222)

Esses quadros têm como propósito proporcionar à equipe de projeto um mecanismo conveniente, simples e visual para organizar seu trabalho, permitindo uma visão geral do trabalho que precisa ser executado e melhorar a interação entre os membros da equipe e sua participação no planejamento e controle do projeto, descentralizando essa atividade do gerente de projetos.

2.2. SOFTWARES PARA GERENCIAMENTO DE PROJETOS E AGILIDADE

Segundo o PMI(2008), um “*software de gerenciamento de projetos é um tipo de aplicativo de software especialmente projetado para auxiliar a equipe de gerenciamento de projetos no planejamento, monitoramento e controle de projeto, inclusive: estimativa de*

custos, elaboração de cronogramas, comunicação, colaboração, gerenciamento de configuração, controle de documentos, gerenciamento de registros e análise de riscos”. Os softwares de gestão de projetos são fundamentais para o gerenciamento de projetos.

Há ferramentas tradicionais como Microsoft Project¹ e Primavera Professional Project Management² que se tornaram o paradigma na solução desse desafio. Os recursos presentes nestas ferramentas podem ser consultados em Vargas (2011). Eles permitem a edição de WBSs (*Work Breakdown Structure*) de projetos e funcionalidades para alocar recursos, estimar prazos, custos e realizar os diversos tipos de controle de projetos. As funcionalidades clássicas são resumidas em Rozenfeld et al. (2006), sendo as principais listadas a seguir.

- Gestão de calendário e agenda: os sistemas permitem a organização e programação de calendários para o projeto, recursos ou tarefas;
- Gestão de atividades: os sistemas permitem descrever a WBS e registrar as atividades, visualizá-las e organizá-las em níveis e codificação. Permitem, também, estabelecer precedência entre as atividades e representá-las na forma de rede de atividades, de Gráfico de Gantt e geração de WBS. Permitem buscas, filtros e outras facilidades para editar as tarefas e definir dados básicos como esforço, duração, datas de início e fim e recursos alocados;
- Gestão de recursos: permitem o gerenciamento das pessoas e materiais necessários para o projeto, cálculos de disponibilidade, facilitam o acompanhamento, algoritmos para realizar o nivelamento dos recursos e são capazes de gerar diferentes tipos de gráficos para analisar a alocação;

¹ <http://www.microsoft.com/project/en/us/default.aspx>

² <http://www.oracle.com/us/products/applications/042374.htm>

- Gestão de custos: permitem gerar cálculos de orçamento, com as informações dos recursos e atividades, e disponibilizar essas informações em relatórios com diferentes níveis de codificação;
- Ferramentas de monitoramento: os sistemas mais avançados possuem funcionalidades para auxiliar o acompanhamento do projeto. A funcionalidade mais simples é a geração de linhas de acompanhamento em gráfico, gráficos de Gantt, por exemplo, indicadores de desempenho do tipo planejado versus realizado e relatórios de diferentes tipos (como calendário do projeto, lista de tarefas por recurso, listas de tarefas por grupo, etc.), com os mesmos parâmetros armazenados nas linhas de base completas. Os recursos mais avançados de acompanhamento são a gravação de diferentes linhas de base e a geração de indicadores e curva S, segundo a técnica de análise do valor agregado; e
- Gerenciamento de múltiplos projetos: permitem a análise do portfólio da empresa, a realização das análises citadas e dos relatórios considerando mais de um projeto ao mesmo tempo, compartilhamento de dados de recursos entre projetos (*pool* ou grupo de recursos). Esse é um problema que deve ser analisado com cuidado, pois, na grande maioria das empresas, vários projetos estão em andamento ao mesmo tempo, compartilhando um mesmo conjunto de recursos.

Essa classe de sistemas tem sido objeto de interesse da comunidade científica que estuda avanços em softwares e sistemas de informação, relacionada com a engenharia colaborativa, mais recentemente definida como *Computer Supported Collaborative Design (CSCD)* (SHEN et al., 2008).

Conforme Shen, Hao e Li (2008, p.855), a CSCD é o processo de desenhar um produto por meio da colaboração inter e multidisciplinar, apoiada por computador e considerando todo o ciclo de vida completo do produto. Segundo os autores, esse corpo de

conhecimento tem sido formado pela combinação de campos de pesquisa iniciados na década de 80, conhecidos como: Engenharia Simultânea (concorrente ou *concurrent engineering*), *Computer supported cooperative work* (CSCW), Interação Humano-Computador (IHC) e literaturas mais específicas sobre *Blackboard*, *Distributed Artificial Intelligence* (DAI) e agentes de softwares. Dentre os sistemas estudados na área, encontram-se os sistemas de gerenciamento de projetos, que apoiam o trabalho em grupo permitindo o planejamento e controle das atividades e resultados do projeto.

Uma revisão sobre ferramentas de Gerenciamento de Projetos foi realizada por Araujo (2008), entre 2001 e 2008, identificou artigos sobre novas propostas de softwares em gerenciamento de projetos. As definições e principais fontes utilizadas em Shen, Hao e Li (2008) e PMI(2008) foram consideradas para a identificação das principais revistas científicas³ e principais bases de dados⁴, nos campos da CSCD, engenharia de software e de gerenciamento de projetos. As palavras-chave foram retiradas destas áreas também. Empregou-se a palavra-chave “software” combinada com as palavras: “*product data management*”, “*configuration management*”, “*virtual products*” e “*collaborative product development*”, além da palavra “*project management*”. Foram contempladas publicações de pesquisadores da área de Gerenciamento de Projetos e Engenharia Colaborativa. O resultado foram os artigos listados na Tabela 1.

³ Os principais são: Computers in industry, Computers & industrial engineering, Advanced Engineering Informatics, International Project Management Journal; Project Management Journal

⁴ As principais foram: Web of Science, Scopus e Science Direct.

Tabela 1. Lista de artigos encontrados.

Autor (Ano) - Software	Título do sistema: descrição
Rodgers et al. (2001)	WebCADET: a Web-based server/repository for design knowledge that supports decision making during the first stages of design.
Chung and Lee (2002)	An XML and CORBA format on the Web as a system of data-exchange standards to collaborate in the development of mould injection.
Huang and Mak (2002)	CyberCO: a workflow management framework for collaborative product development in Web-based environments.
Huang (2002)	CyberReview: a Web-based system to revise product designing via the collaborative concept.
Liang and Huang (2002)	ICA: a system based on collaborative information agents for a modular-product project.
Qian and Shensheng (2002)	P_PROCE: an architecture and implementation to manage a product development process integrated to a workflow model.
Hameri and Puittinen (2003)	A Web-based project management framework.
Qin et al. (2003)	A prototype of a Web-based conceptual design system that simulates the working of parts for further detailing.
Tay and Roy (2003)	CyberCAD: a Web-based system that combines CAD to videoconference.
Wang, Shen and Ghenniwa (2003)	WebBlow: a software program comprising a combination of technologies such as software agents, Web, XML, and Java to share product information during projects, in particular, projects of auto parts.
Li, Fuh and Wong (2004)	A Web-based system to support collaborative engineering with distributed teams.
Rodriguez and Al-Ashaab (2005)	KdCPD: an architecture for a Web-based collaborative product development system.
Woerner and Woern (2005)	CSCE: a model to manage partners' relationships in engineering projects using the CSCW concept and focusing on trust building among them.
Wu et al. (2006)	An architecture that captures, via agents, collaborative design system users' interests and behaviors.
Mejía, López and Molina (2007)	A collaborative engineering environment by means of existing collaborative engineering tools.

Percorrendo as descrições, é possível identificar os tipos de proposta e também que, até 2008 não há nesta literatura propostas relacionadas com softwares para gerenciamento ágil de projetos. Isso é, nenhum dos artigos relatava o problema. Demonstra também que as

propostas existentes não exploraram os aspectos da gestão visual, uso de painéis e novas tecnologias interação homem-computador.

Assim, a não existência não implica em irrelevância. Parece sim, que os pesquisadores desses campos não tem explorado as novas possibilidades e a nova abordagem do APM neste campo. Essa última análise é corroborada por Shen, Hao e Li (2008). Eles indicam que a área de Gerenciamento de Projetos Distribuídos como uma das oportunidades e desafios principais nesse campo de pesquisa, identificando problemas como a sobrecarga de informações de gerenciamento na equipe de projeto, problemas de programação de atividades, solução de conflitos, controle de custos e tarefas, todos eles próximos às questões principais da abordagem do Gerenciamento Ágil de Projetos.

Ainda nessa época, Araujo (2008) apresentou uma análise dessa revisão conjuntamente com estudos de casos e análises de funcionalidades existentes em softwares comerciais, resultando em um conjunto de requisitos; revistos e aprimorados em Amaral et al. (2011). Os requisitos funcionais e não funcionais desenvolvidos por Araujo (2008), revisados por Amaral et al. (2011), são apresentados nos Quadro 2 e Quadro 3.

Quadro 2. Requisitos funcionais para softwares de gerenciamento ágil.

1) Permitir o registro e controle de usuários	a)	Permitir a inclusão/edição/exclusão do cadastro de organizações;
	b)	Permitir a inclusão/edição/exclusão do cadastro de usuário relacionado à sua organização;
	c)	Permitir a definição de privacidade de informações;
	d)	Possibilitar o cadastro/edição de controle de permissões de acesso e relacioná-lo aos usuários;
	e)	Registrar os acessos e ações dos usuários, gerando um registro de log;
	f)	Permitir o backup de informações pelas organizações, respeitando as restrições de acesso e privacidade de informações;
2) Deve apoiar a elaboração da visão do produto e especulação	g)	Permitir a criação de uma visão do produto, isto é, uma descrição sintética do produto que será desenvolvido, por meio da inserção e combinação de diferentes documentos, como modelos de estrutura de funções e esboços iniciais da concepção do produto final;
	h)	Possibilitar especulação em torno da visão: registro de alternativas propostas por usuários e ferramentas de apoio na seleção das alternativas;
	i)	Armazenar o histórico das alterações na visão e decisões sobre ela;
	j)	Apresentar para o usuário o modelo final da visão de maneira sintética;
3) Planejar o projeto de forma simples	k)	Possibilitar o planejamento por etapas, baseado em iterações e entregas (resultados) ao invés de atividades e fases;
	l)	Permitir a definição de equipes e sua associação às entregas;
	m)	Registrar as atividades principais, sendo essas relacionadas às entregas e usuários responsáveis;
	n)	Permitir o registro do prazo máximo das entregas do projeto;
	o)	Permitir a utilização de informações da visão para gerar o planejamento do projeto, possibilitando: relacionar partes da visão do produto com entregas e relacionar partes do produto por meio da descrição das interfaces;
	p)	Registrar as limitações, restrições e metas de custo do projeto;
	q)	Permitir a identificação e priorização dos riscos do projeto;
	r)	Registrar alterações do planejamento: entregas, visão, restrições, responsáveis e demais elementos;
	s)	Manter os dados do planejamento base e permitir comparações do atual e com o planejamento base;
	t)	Gerar uma forma gráfica de visualização do planejamento;
	u)	Permitir a definição de indicadores de desempenho, utilizando os dados existentes;
	v)	Controlar status das entregas do projeto;
	w)	Controlar e registrar alterações/validações das entregas;
	x)	Possibilitar avaliação das entregas e avaliação parcial do projeto;
4) Acompanhar o andamento do projeto	y)	Registrar novos requisitos;
	z)	Permitir a associação de documentos de vários tipos, à execução das entregas;
	aa)	Gerar relatórios de indicadores de desempenho a partir de dados de entrega e interações;
	bb)	Controlar versões de documentos;
	cc)	Registrar as interações entre os membros do projeto;
	dd)	Possibilitar a visualização detalhada do andamento do projeto;
	ee)	Possibilitar acompanhamento e avaliação pelo cliente, de maneira contínua durante o projeto
	ff)	Criar um painel para o acompanhamento dos riscos envolvidos no projeto.
	2)	Solicitar avaliação final do projeto;
	a)	Gerar indicadores de desempenho finais;
	b)	Gerar forma gráfica de demonstração do resultado final do projeto;
	c)	Guardar validações finais do projeto;
	d)	Permitir o acesso ao histórico do projeto, depois de finalizado, respeitando as restrições de acesso de cada usuário;
	e)	Permitir avaliação final do projeto pelos clientes
6) Comunicação e Integração	f)	Possuir diferentes formas de comunicação entre os membros do projeto, incluindo ferramentas síncronas e assíncronas;
	g)	Registrar as interações síncronas ou assíncronas entre os usuários e seus resultados, de forma que as informações possam ser distribuídas entre os demais membros do projeto ou consultadas posteriormente;
	h)	Deve possibilitar o uso e integração de documentos de diferentes tipos, como por exemplo, textos oriundos de diferentes editores;
	i)	Possibilitar o uso de dados dos sistemas de gerenciamento de projetos tradicionais;
	j)	Permitir a integração com sistemas de workflow para disparar fluxos de atividades;

Fonte: desenvolvidos por Araujo (2008), revisados por Amaral et al. (2011).

Quadro 3. Requisitos não funcionais para softwares de gerenciamento ágil.

1)	A interface deve ser baseada na web. Todas as propostas da literatura empregam interface web, de forma a garantir a distribuição das informações entre os colaboradores, com fácil acesso, não necessitando da instalação de softwares locais. Das 15 propostas de softwares analisadas, todas trocavam dados via web e tinham como interface um browser. Os três softwares de APM também .
2)	Uso de ferramentas de comunicação variadas. Já que o foco é a colaboração no projeto de desenvolvimento de produto, existe a necessidade de várias formas de comunicação e interação entre os colaboradores. Este requisito é apresentado na revisão da literatura, onde nota-se um fator de sucesso do projeto na comunicação efetiva. Também foi observada em estudos de caso realizado pelo nosso grupo. Eles demonstraram a necessidade da integração de comunicação com colaboradores externos, de forma rastreável.
3)	Interface homem-computador simplificada e resumida por meio do uso intensivo de gráficos. O enfoque ágil propõe o uso de relatórios simples e com o máximo de aproveitamento de recursos gráficos. Portanto, o software deve utilizar formas gráficas variadas de modo a facilitar o entendimento de uma entrega ou do próprio produto final, como por exemplo, o uso de modelos gráficos 3D para esboços do produto, demonstrados na análise dos softwares para gerenciamento ágil de desenvolvimento de software. Nas propostas de softwares da literatura esta necessidade está presente e já foi identificada por outros autores . Também podem ser incluídas novas formas de visualização de fases do projeto, como foi visto nos novos SGP para desenvolvimento de software baseados no enfoque ágil.
4)	Integração com funcionalidades de sistemas de engenharia como CAD e Ambientes de desenvolvimento de software. Atender este requisito não é simples, mas acredita-se que seja a tendência. Para que as decisões sejam tomadas de maneira rápida seria importante que os membros da equipe tivessem as ferramentas de controle integradas aos seus ambientes naturais de trabalho. Por exemplo, o desenvolvedor altera o status do documento de projeto e o sistema imediatamente acusa o fim da entrega no sistema de gerenciamento.
5)	Descentralização dos dados. Permitir que os usuários possam exportar os dados e trabalhar nas ferramentas de gerenciamento de projetos que julgarem mais apropriadas. Uma conclusão geral dos estudos de caso é a de que as ferramentas de planejamento e controle não são utilizadas na colaboração com agentes externos. Um problema é o acesso aos sistemas da empresa. Viabilizar a utilização de multi-plataformas poderá facilitar esse compartilhamento, na medida em que usuários externos poderão utilizar o software que mais lhe convier (seja pelo costume, conhecimento ou investimento já realizado).
6)	Rastreabilidade das discussões sobre as entregas e decisões em gates. Na revisão da literatura nota-se que o objetivo do projeto deve ser claramente definido e este é um fator de sucesso nos projetos colaborativos. Nos estudos de caso identificou-se que um dos aspectos importantes é a rastreabilidade. E que isso seria tanto legalmente como contratualmente importante.
7)	Flexibilidade de alteração das entregas/tarefas para atender às mudanças do ambiente. Um dos principais focos da teoria do APM é a criação de métodos de planejamento e controle onde as mudanças nos planos sejam realizadas facilmente, de forma a garantir a flexibilidade necessária no transcorrer do projeto. A falha no controle de mudança do projeto pode impedir o sucesso do projeto, justificando a necessidade. Nas propostas dos softwares de literatura, na avaliação da área de Integração, também foi identificada a ausência de propostas que englobem essa necessidade.
8)	Rastreabilidade dos dados. Todas as discussões, definições e alterações do projeto devem ter a possibilidade de ser rastreadas: estudos de caso realizados no grupo demonstraram a necessidade desta rastreabilidade .
9)	Garantir o sigilo de dados. Na revisão da literatura, e principalmente nos estudos de caso, pode-se notar que o controle de informações é um ponto crítico para as empresas. Uma das empresas estudadas por nós demonstrou possuir ferramentas e deixar de utilizá-las com receio deste aspecto .
10)	Controle de versão e fluxo de documentos. Esta função é reconhecida como importante há tempos. Porém, muitas empresas ainda não possuem procedimentos seguros de versionamento e controle de documentos. Mesmo as que os possuem apresentam problemas de confiabilidade nas informações. Há até mesmo aquelas que não conseguem controlar a configuração. Nos estudos de caso que realizamos, em empresas de ponta em diferentes setores industriais pudemos verificar este quadro. Apresentando os resultados em eventos internacionais, vimos que isso se dá em outros países. Este controle, porém, é fundamental no APM e precisa ser reforçado.
11)	Indicadores de desempenho de forma gráfica e sintética. O capítulo anterior demonstrou como os indicadores de desempenho são importantes para o acompanhamento do projeto. O grande desafio no ágil é conseguir dados, considerando que as práticas envolvem pouca padronização. Assim, os sistemas de informação devem ter um papel fundamental neste campo. A geração automática de indicadores simples é uma meta, como demonstrado na seção 5.4.
12)	Controle de multiprojetos. Os sistemas de informações devem apoiar este controle, integrando as informações sobre recursos disponíveis e ocupados para os projetos, garantindo uma melhor programação de datas e prazos. O caso apresentado no capítulo anterior é um exemplo de como podem apoiar.
13)	Exigir o menor tempo possível durante a execução das atividades rotineiras. Foco no resultado e simplicidade. Esta necessidade é apresentada na revisão da literatura, onde é apontada a burocracia do gerenciamento clássico de um projeto e no surgimento de novas ferramentas comerciais para gerenciamento de projetos com foco na entrega do produto.

Fonte: desenvolvidos por Araujo (2008), revisados por Amaral et al. (2011).

O surgimento das publicações de artigos científicos com softwares especialmente desenvolvidos para a temática do ágil acontece de maneira mais intensa à partir de 2008. Para mostrar o panorama da área, empreendeu-se uma busca na base de dados *Web of Science* empregando os termos “gerenciamento ágil de projetos” (*agile project management*), o

conector AND e um conjunto de termos para identificar proposta de softwares (softwares e *tools*) indicou 207 publicações no período de 2000 até 2011.

O resultado são trabalhos que identificam ferramentas de software e também trabalhos sobre aplicação de métodos ágeis em geral, visto que essa abordagem é bastante utilizada na área de desenvolvimento de software e desenvolvimento de ferramentas. Portanto, o conjunto união são trabalhos que versam sobre softwares para APM (objetivo da dissertação) e trabalhos que versam sobre a teoria de métodos ágeis e a sua aplicação.

Analisando o resultado, mais da metade, 77% foram publicadas em congressos, indicando que as publicações estão em fase de execução, conforme Tabela 2. A publicação em congresso geralmente significa um resultado parcial, de pesquisa em andamento. A conclusão é que se trata de uma área que já atraiu um número expressivo de pesquisadores. É possível observar também a quantidade de trabalhos publicados por ano. Nota-se pela Tabela 3 que há um conjunto de trabalhos entre 2002 e 2007 e outro grupo entre 2008 e 2011, sendo que os últimos quatro anos correspondem a cerca de 58% dos trabalhos.

Em seguida, empreendeu-se uma análise do resumo e título de cada um dos 207 trabalhos para identificar aqueles que trataram de questões relacionadas aos softwares em si, isto é, aplicativos para apoiar a aplicação do gerenciamento ágil. Foram identificados artigos que preenchessem um ou mais dos critérios:

1. apresentar a proposta de um software ou aplicativo para solucionar problema específico ligado ao gerenciamento de projetos;
2. apresentar proposição teórica para apoiar a construção de softwares para gerenciamento de projetos; e
3. apresentar a análise de caso de avaliação de uma ou mais ferramentas para gerenciamento de softwares, em casos reais ou não.
- 4.

Tabela 2. Resultado da busca completa (207 artigos) por tipo.

Tipos de Documentos - Busca Geral	Quantidade de artigos	Porcentagem do total
Anais de congressos	160	77,295
Artigos de revista	47	22,705
Total	207	

Tabela 3. Quantidade de artigos por ano resultante da busca completa.

Ano de Publicação - Total	Quantidade de artigos	Porcentagem de 207
2008	40	19,324
2009	37	17,874
2010	31	14,976
2005	21	10,145
2006	21	10,145
2007	17	8,213
2003	13	6,280
2011	13	6,280
2004	10	4,831
2002	3	1,449
2001	1	0,483

Os critérios apresentados eliminaram, portanto, artigos que tratavam da teoria do Gerenciamento Ágil de Projetos e não contribuíam para o problema da construção de softwares, somando 57 artigos, conforme Tabela 4. O resultado é que cerca de 71% dos artigos passam agora a ser relacionados com os períodos de 2008, 2009 e 2010, conforme Tabela 5, mantendo-se a mesma proporção de artigos de congressos. O ano de 2008 é o período com a maior quantidade de artigos.

Os resultados indicam o ano de 2008 como um marco na literatura de software para gerenciamento ágil de projetos. Indicando que é esse o momento do surgimento das primeiras propostas mais específicas para solucionar o problema de apoiar computacionalmente o desenvolvimento de produtos, empregando a abordagem do gerenciamento ágil.

Tabela 4. Quantidade de artigos que tratam de software por tipo.

Tipos de documentos Filtrados	Quantidade de artigos	Porcentagem de 57
Anais de congressos	36	63,158
Artigos de revistas	21	36,842
Total	57	

Tabela 5. Quantidade de artigos que tratam de software por ano.

Ano de Publicação - Filtrado	Quantidade de artigos	Porcentagem de 57
2008	15	26,316
2009	13	22,807
2010	13	22,807
2003	5	8,772
2005	3	5,263
2007	3	5,263
2011	3	5,263
2006	1	1,754
2004	1	1,754

Os trabalhos que apresentam propostas reais de ferramentas computacionais para APM são apenas cinco, apresentados no Quadro 4. Os trabalhos de Garcia et al. (2010) e Stilli e Succi (2005) tratam de temas bem específicos e são pequenos aprimoramentos em soluções que já existem. Há CVS (*Concurrent Version System*) e relatórios de acompanhamento de CVS, prazo versus realizado, em sistemas comerciais. Eles são passíveis de serem implementados em ferramentas do tipo *Product Data Management*, por exemplo. Já o trabalho de Berger e Rumpe (2010) é de software que não apoia a gestão em si, isso é, as tarefas de gerenciamento propriamente ditas; focam no modelo de processo de desenvolvimento. É mais amplo, mas não traz detalhes dos diferenciais da ferramenta proposta versus as existentes.

Os demais artigos além dos que aparecem no Quadro 4, até formar 57, são artigos que: apresentam revisões bibliográficas e tendências (por exemplo, Spencer et al. (2011)), casos como Procter et al. (2011); Smite et al. (2010); e Cabral et al. (2009). A maior parte são

artigos que descrevem melhores práticas (características, benefícios e processos) de APM para engenharia de software como Rusu et al. (2009) e Concas et al. 2008. Alguns foram retirados por serem referentes a mesma proposta como acontece com Amescua et al. (2010) que é igual ao primeiro item do Quadro 4.

Quadro 4. Síntese das publicações específicas sobre softwares de gerenciamento ágil de projetos

Autor (ANO)	Título	Tipo de Fonte	Objetivos
García et al. (2011)	Design guidelines for software processes knowledge repository development	Revista	Apresenta uma proposta que utiliza wiki para armazenar melhores práticas em engenharia de software
Garcia & Pacheco (2010)	A Web-based Tool for Automatizing the Software Process Improvement Initiatives in Small Software Enterprises	Revista	Ferramenta de software que apresenta um modelo para o processo de desenvolvimento de software, especialmente para empresas pequenas (contém práticas de Gerenciamento Ágil de Projetos no modelo).
Berger & Rumpe (2010)	Supporting Agile Change Management by Scenario-Based Regression Simulation	Revista	Softwares para apoiar o gerenciamento de mudanças, utilizando agentes, de forma a possibilitar o gerenciamento ágil de projetos de novos produtos, aplicado em um projeto de pesquisa avançado da DARPA, de veículo autônomo.
Rychly & Ticha (2008)	A Tool for Supporting Feature-Driven Development	Revista	This tool has been implemented as a multi-user web-based application enabling creation of feature lists, planning a project, supporting, cooperation among members of a feature-team, and tracking project progress in all illustrative way. To support project management and communication with customer representatives, a wide range of reporting features has been provided.
Sillitti & Succi (2005)	Source code repositories and agile methods	Congresso	Propõe uma ferramenta que recupera e sintetiza informações de repositório de sistemas de gestão de configuração (CVS – Concurrent Version Systems) para que analistas possam acompanhar a situação do projeto e propor mudanças.

Outro fato interessante é que o levantamento mostra que a proposta de softwares em si ainda não está sendo tratado nos periódicos da área de gerenciamento de projetos ou de CSCD, estando restrito aos periódicos relacionados com engenharia de software. A Tabela 6 justifica a afirmação.

Tabela 6. Principais periódicos que discutem temas relacionados com software para Gerenciamento Ágil.

Títulos dos periódicos principais	Número de artigos
LECTURE NOTES IN COMPUTER SCIENCE	7
LECTURE NOTES IN BUSINESS INFORMATION PROCESSING	5
AGILE PROCESSES IN SOFTWARE ENGINEERING AND EXTREME PROGRAMMING PROCEEDINGS	2
COMMUNICATIONS IN COMPUTER AND INFORMATION SCIENCE	2
COMPUTER SUPPORTED COOPERATIVE WORK THE JOURNAL OF COLLABORATIVE COMPUTING	2
EMPIRICAL SOFTWARE ENGINEERING	2
IET SOFTWARE	2
INFORMATION AND SOFTWARE TECHNOLOGY	2
INTERNATIONAL SPRING SEMINAR ON ELECTRONICS TECHNOLOGY ISSE	2

Os resultados da busca permitem concluir que a abordagem de gerenciamento ágil de projetos ainda não tem sido explorada pela comunidade de pesquisadores em softwares de apoio ao desenvolvimento de produtos e de gerenciamento de projetos. Como apresentado na seção anterior, a área de conhecimento em gerenciamento está recebendo nova atenção desde o surgimento do APM, a partir dos anos 2000. Trata-se de uma nova abordagem que propõe a simplificação dos métodos e ferramentas empregados em gerenciamento de projetos. Essa simplificação significa um impacto direto para os softwares utilizados. Por exemplo, as novas recomendações de utilização de painéis visuais e controles baseados em resultados, não estavam presentes quando os softwares atuais de gerenciamento de projetos foram propostos.

A situação dos softwares comerciais, porém, a situação é distante. Com os métodos ágeis se popularizando, há uma quantidade grande de propostas de softwares comerciais que se propõem a ser ágeis. Os principais são VersionOne⁵, TargetProcess⁶ e Rally⁷. A Figura 2 apresenta os principais softwares utilizados, segundo um levantamento realizado pela empresa desenvolvedora do software Version One.

Figura 2. Utilização de softwares de gerenciamento de projetos.

Fonte: Pesquisa anual da empresa VersionOne entre empresas - Dados de 2010 .

⁵ <http://www.versionone.com>

⁶ <http://www.targetprocess.com>

⁷ <http://www.rallydev.com>

Araujo (2008) desenvolveu uma análise das funcionalidades de alguns desses softwares. Considerando o VersionOne, TargetProcess e o Rally, a autora aponta como as principais diferenças em termos de funcionalidades:

- utilizam como principal ideia um planejamento inicial simplificado (*Workitem Planning*) e iterações (*Sprints*) dentro do projeto. Dentro de cada iteração, são registradas as entregas que devem ser realizadas e seu prazo;
- o tradicional gráfico de Gantt não é utilizado, mas existem outras ferramentas para acompanhamento visual do andamento do projeto, como status de cada iteração, entrega, subentrega e relatórios “*Burndown*”, isto é, que descrevem progresso de uma equipe e o denominado “*Roadmap*”, relacionado ao escopo do projeto e suas mudanças ao longo do projeto;
- possuem área para guardar as retrospectivas, que seriam as lições aprendidas, além das análises das estimativas iniciais do projeto;
- as mudanças de escopo são gerenciadas e podem ser analisadas em relatórios, permitindo aos colaboradores acompanhar as alterações do projeto; e
- todos os softwares eram específicos para desenvolvimento de produtos do tipo software. Como resultado, há funcionalidades como descrição da visão e de funcionalidades (requisitos) que não se adaptam ao caso de produtos que envolvam hardware.

Como síntese da seção, indica-se que os desafios para a evolução de SGPs envolvem os seguintes aspectos:

- Desenvolver ambientes customizados para cada projeto com funcionalidades que privilegiem a visão do produto final, isto é, a agregação de valor;
- Sistemas que explorem planos baseados em entregas e iterações e que sejam mais “visuais”, explorando os novos dispositivos de mediação como *touchscreen* etc.;

- Sistemas nos quais a alteração e comunicação nos planos seja fácil e o mais automática possível, que pudessem ter a flexibilidade para controlar os projetos e de se realizarem planejamentos. Capazes também de se adequar às mudanças dos projetos;
- Sistemas que apoiem o gerenciamento de recursos entre projetos, isto é, entre equipes ágeis, utilizando os controles simplificados de iteração. Não há sistemas que explorem esta questão.
- Que consumam pouco tempo com documentação, explorando recursos como relatórios em vídeo e som;
- Criar funcionalidades que facilitem as alterações constantes no produto e gerem informações “realistas” e em tempo real sobre o andamento do projeto.
- Necessidade de apoiar a colaboração, com ferramentas e funcionalidades que facilitem a troca rápida de informações, e garantindo também a privacidade das partes interessadas; e
- Possibilidade de utilização de plataformas heterogêneas (promovendo a descentralização da informação e permitindo inclusive que diferentes equipes possam utilizar sistemas distintos conforme sua conveniência).

Interessante é o surgimento de novas tecnologias de IHC e ideias na área de arquitetura de software que podem ajudar nesses problemas, tais como painéis digitais *touchscreen* e outras, apresentadas nas próximas seções.

2.3. ARQUITETURA DE SOFTWARE

Os programas computacionais são formados por um conjunto de elementos interligados, como componentes de software propriamente ditos, hardwares específicos ou outros softwares. A esse conjunto emprega-se o termo Arquitetura de Software.

A arquitetura de software é “*a descrição dos componentes de um determinado sistema computacional e as interações entre esses componentes, incluindo os elementos de hardware e software*” (Rechtin, 1991 *apud* Jones e Konrad, 2011).

Segundo Kruchten et al. (2006), as primeiras citações do termo arquitetura são de 1969, mas o conceito emerge como disciplina na década de noventa. Os autores argumentam que, apesar de ser uma disciplina com um significativo tempo de desenvolvimento, há, paradoxalmente, pouco consenso quanto a uma definição específica de arquitetura, ou seja, em termos de uma definição precisa do conceito e de sua descrição.

Analisando os principais trabalhos da área, Kruchten et al. (2006, p.23) concluem que o consenso existente é o de que uma arquitetura envolve dois aspectos:

- (1) a estrutura e organização pela qual os componentes e subsistemas interagem para dar forma ao sistema; e
- (2) as propriedades do sistema que podem ser melhor projetadas e analisadas no nível de sistema.

A definição utilizada neste trabalho é a apresentada por Nakagawa (2006), que foi baseada na da *Software Engineering Institute*:

“a arquitetura de software é uma estrutura de componentes de um programa/sistema, os relacionamentos entre esses componentes, os princípios e diretrizes que governam os projetos e a evolução dos softwares. Os blocos de construção (ou seja, os componentes) de uma arquitetura são constituídos de unidades que desempenham o comportamento do sistema; são constituídos também de conexões que mostram como essas unidades transmitem ou comunicam informações e de conexões que mostram como as várias operações do programa são atividades sequencial ou concorrentemente”. Nakagawa (2006, p.28),

Optou-se por essa definição por ser a mais detalhada e abrangente dentre as encontradas nos autores referenciados nessa seção e por seguir os dois elementos essenciais identificados na extensa revisão bibliográfica apresentada por Kruchten et al. (2006).

Quanto ao uso e importância das arquiteturas de software, não há dúvidas na literatura especializada. As arquiteturas de software capturam e preservam as intenções dos projetistas do software sobre a estrutura e comportamento do sistema, bem como da interação entre o sistema e o ambiente. Com isso, elas impactam no desempenho geral do sistema e garantem compatibilidade entre sistemas da empresa, facilitando a escalabilidade e reúso (Kruchten et al., 2006).

A descrição de uma arquitetura pode servir a vários propósitos. Segundo Clement et al. (2004, p.22) ela pode servir para guiar os desenvolvedores, para permitir a análise geral do sistema, para a geração automática de código, para a compreensão e tomada de decisões de investimentos ou um roteiro (*blue print*) para o gerenciamento do projeto do software.

Nakagawa (2006, p.29) apresenta um conjunto de indícios, como lista de conferências e várias citações na literatura técnica, de que o conceito de arquitetura de software é uma das bases essenciais da disciplina Engenharia de Software atual.

O projeto adequado da arquitetura é fundamental para obter uma maior qualidade e baixo custo do software, no atual contexto de aumento da complexidade dos softwares e de possibilidades tecnológicas de codificação, hardware e transmissão de dados (Mendes, 2002⁸ apud Nakagawa, 2006). Ela pode orientar a discussão entre *stakeholders* de forma a melhorar a tomada de decisões sobre tecnologias, estrutura, distribuição do trabalho e interface entre partes do sistema e o sistema no seu ambiente de funcionamento.

Segundo Kruchten et al.(2006, p.23), as subáreas, ou tópicos fundamentais, da disciplina de arquitetura de software são cinco, conforme identificação da Federação Internacional de Processamento de Informação, Grupo de Trabalho 2.10. Descritas à seguir.

⁸ MENDES, A. **Arquitetura de Software**: Desenvolvimento Orientado para arquitetura, Rio de Janeiro, Brasil, Campus, 2002.

- **Projeto (de arquiteturas de software).** Como se faz para produzir uma arquitetura de software?
- **Análise (de arquiteturas de software).** Como se faz para responder às questões de qualidade do produto final software, à partir de projetos de arquitetura.
- **Realização (de arquiteturas de software).** Como se faz para produzir um sistema de software tendo-se como base o projeto da arquitetura.
- **Representação (de arquiteturas de software).** Como produzir artefatos duráveis para comunicar a arquitetura para usuários humanos e máquinas.
- **Avaliação econômica (de arquiteturas de software).** Como uma arquitetura direciona as decisões de negócio.

Nakagawa (2006), revisando os trabalhos de Garlan & Perry (1995), Mendes (2002), Shaw & Clements (1997), Eickelmann & Richardson (1996 apud Nakagawa, 2006), Taylor et al. (1995) e Taylor et al. (1998), propõe uma classificação de arquiteturas em três tipos distintos, conforme a abrangência do objetivo:

- **Estilo Arquitetural.** Este é o nível mais abrangente, que permite que um profissional (projetista, arquiteto ou engenheiro) determine a classe a qual pertence à organização de um sistema. Indicam as características dos componentes (subsistemas) e conectores disponíveis, topologia da arquitetura, restrições semânticas e mecanismos de interação entre os componentes que ajudam a identificar o estilo que retrata a arquitetura. Exemplos são (Nakagawa, 2006, p. 31): “...*Pipes e filtros, Camadas, Cliente-Servidor, entre outros, bem como estilos resultantes da combinação de estilos arquiteturais*”. Elas não são ligadas a um domínio de aplicação e podem servir de inspiração para diferentes tipos de software;

- **Arquitetura de Referência.** As arquiteturas de referência são aquelas construídas em um nível mais específico de aplicação. Segundo Eickelmann & Richardson (1996 APUD Nakagawa, 2006)⁹ é uma estrutura que provê uma caracterização das funcionalidades dos sistemas de software de um dado domínio de aplicação. Nakagawa (2006) cita exemplos de arquiteturas desse tipo para diferentes domínios como servidores www, comércio eletrônico e engenharia de software; e
- **Arquitetura de software de domínio específico (do inglês *Domain-Specific Software Architecture – DSSA*).** Fornece uma base estrutural para a interoperabilidade de componentes dentro de um domínio. Uma DSSA pode dar apoio a uma abordagem de desenvolvimento de forma que permita a configuração de componentes reutilizáveis, para um domínio específico de aplicações. Segundo Nakagawa (2006, p.32), baseado em revisão de outros autores, um DSSA compreende um modelo de domínio, requisitos de referência, uma arquitetura de referência, infraestrutura ou ambiente de apoio e um processo de desenvolvimento. Clements et al. (2004) apresentam um exemplo de documentação de arquitetura para um sistema complexo para capturar, gravar, distribuir, processar e tornar disponível altos volumes de dados de uma constelação de estrelas em um satélite orbital. Nele os autores descrevem todos esses elementos, caminhando da referência maior até os detalhes de implantação (visão física) para orientar a equipe de projeto sobre as interfaces entre os

⁹ EICKELMANN, N. S.; RICHARDSON, D. J. An evaluation of software test environment architectures. In: Proc. of the 18th International Conference on Software Engineering, Berlin, Germany: IEEE Computer Society Press, 1996, p. 353–364.

elementos, a manutenibilidade, a escalabilidade e a verificação de desempenho e requisitos não funcionais.

Uma entidade de software é algo complexo e a descrição dos seus componentes não pode ser feita de maneira unidimensional (Clements et al., 2004, p.22). A solução, segundo Clements et al.(2004), é o uso do conceito de visão. Uma visão é a descrição de uma parte do sistema, uma seleção, conforme a perspectiva de um determinado ator que irá desenvolver, testar, manter ou continuar desenvolvendo o sistema.

Na acepção de Clements et al. (2004, p.22), *“documentar uma arquitetura é uma questão de documentar (identificar) as visões relevantes e adicionar documentações que sejam compatíveis com mais do que uma visão”*.

O problema então é encontrar a visão relevante. Clements et al. (2004) afirmam que isso depende do propósito da arquitetura, que pode variar como citado anteriormente. Os autores dizem que uma visão expõe não apenas diferentes componentes do sistema, mas também descreve tais componentes em níveis de detalhe distintos, bem como captam diferentes atributos de qualidade. Dessa forma, a escolha da visão depende também do uso e do usuário da documentação; conforme o propósito inicial definido.

Nakagawa (2006, p.70) propõe o uso de no mínimo três visões para a documentação de arquiteturas de referência: a visão estrutural, comportamental e física. A visão estrutural deve apresentar os módulos, unidades de código que implementam determinada funcionalidade, pacotes, aspectos ou outros. A visão em tempo de execução representa a estrutura do código fonte, indicando o comportamento do sistema em execução. A visão comportamental e física apresenta a estrutura de hardware na qual o sistema será executado. Segundo a autora, essa última visão seria importante em sistemas cuja implementação encontra-se espalhada em mais de um hardware como é comum em sistemas *web*.

Clements et al. (2004, p.23) recomendam um conjunto de diretrizes para a descrição de cada visão e um conjunto de diretrizes para a descrição do conjunto de visões. As recomendações para a documentação de uma visão são:

- Conter uma apresentação inicial, usualmente gráfica, que descreve os elementos primários e as relações da visão com outras;
- Conter um catálogo de elementos que explica e define o elemento como apresentado na visão e lista de suas propriedades;
- Conter uma especificação das interfaces entre os elementos e comportamento;
- Conter um guia explicando os elementos básicos de construção e os mecanismos disponíveis para personalizar a arquitetura; e
- Conter as informações objetivas para o projeto;

As recomendações de Clements et al.(2004) para a descrição da documentação sobre o conjunto total, ou subconjunto específico, de um conjunto de visões da arquitetura são:

- Deve conter uma introdução do pacote completo, incluindo um guia que apoie o leitor e ajude *stakeholders* a encontrarem os pedaços de informações que lhes podem ser útil;
- Deve descrever as relações entre as visões e entre cada visão e o sistema completo;
- Deve mostrar o escopo, restrições e razões, da arquitetura completa; e
- Deve possuir um sistema de informação que garanta a manutenção das informações em todo o pacote.

Essas recomendações são amplas e não orientam de forma precisa como estabelecer diagramas e estruturas textuais, ou mesmo por não identificar as visões principais. Há muitas formas e diferentes aspectos que podem ser representados, em função dos muitos objetivos e usos distintos para uma arquitetura.

Clements et al. (2004, p.25) sugerem o conceito de estilos de arquiteturas. Um estilo é uma forma recorrente, amplamente utilizada, para se descrever diferentes sistemas computacionais. Um estilo permite que um especialista que participa do projeto uma classe especial de sistemas com ferramentas de desenvolvimento, análise e implementações específicas. Nesse caso, a definição de estilos envolve também a ferramenta de descrição da arquitetura. Analisando o conjunto de estilos, os autores identificam três categorias ou grupos de estilo.

1. **Estilos de módulos.** São conjuntos, técnicas e ferramentas de análise, implementação e teste, utilizados para descrever unidades de implementação do sistema, isso é, conjunto coesos de objetos, componentes e/ou demais elementos que possuem algo em comum como funcionalidades, objetivos ou desenvolvimento comuns;
2. **Estilos de componentes e conectores.** São conjuntos de elementos que descrevem os componentes do sistema, e os respectivos comportamentos e interações que apresentam em tempo de execução; e
3. **Estilos de alocação.** São conjuntos de elementos relacionados às estruturas que não são apenas software, isso é, referentes ao ambiente. Hardware, outros componentes, outros softwares, internet etc.. Eles permitem descrever o escopo do software e os elementos que são necessários para o seu funcionamento no contexto do sistema de informação.

Os três estilos listados são, portanto, muito próximos às três visões de referência propostas por Nakagawa (2006) e outros da literatura. Conclui-se que apresentar ao menos uma descrição de cada visão é fundamental para descrever uma arquitetura de software, considerando a última delas para os casos nos quais o software pode ser distribuído.

Segundo Kruchten et al.(2006), o conceito de arquitetura e a descrição de arquiteturas estão difundidos na indústria. Os autores contaram 25 livros importantes, uma quantidade significativa de artigos em revistas especializadas e científicas, dezenas de universidades com cursos regulares de arquitetura de software em seus currículos, bem como várias empresas oferecendo treinamentos na área.

Um dos desafios presentes são as práticas de representação, isto é, as ADLs (do inglês, *Architecture Description Languages*). Em trabalhos teóricos, muitas tentativas parecem ter sido realizadas. Ainda em 2000 Garlan et al. (1997), um dos autores seminais da teoria, apresentou uma proposta e realizou trabalhos no sentido de uma linguagem geral para descrição de arquiteturas, em Garlan et al. (1997). Mevidovic e Taylor (1997) e Mevidovic e Taylor (2000) apresentam, respectivamente, definições para o desenvolvimento de uma ADL e comparam as propostas existentes na época, início da década do novo milênio. Os autores encontraram 10 propostas: ACME, Aesop, C2, Darwin, MetH, Rapide, SADL, UniCon, Weaves e Wright.

Uma das conclusões dos autores é que havia um problema de falta de compreensão da relação entre domínio específico e arquitetura e, enquanto isso não fosse solucionado, permaneceria a dificuldade em construir ADL's gerais, capazes de estruturar o software, conforme o domínio de aplicação (MEDVIDOVIC & ROSENBLUM, 1997). O resultado é que as propostas da época não eram um padrão de fato, útil para a comunidade.

Passados quase dez anos, Kruchten et al. (2006) notam, sobre as ADLs, que permanece a não existência de um padrão de fato. Os que mais se aproximam de uma disseminação prática, segundo os autores, seriam o Koala (Ommering et al., 2000), voltado para o desenvolvimento de softwares embarcados para a indústria de bens de consumo, e a UML (*Unified Modeling Language*). No caso da UML, fazem a ressalva que os teóricos da área questionam a possibilidade de classificá-la definitivamente como uma “ADL pura”. O

que se faz são usos das convenções da linguagem UML, com adaptações específicas para cada caso, tal que não se pode descrever como uma ADL, de forma mais restrita e precisa (purista). É interessante notar que os autores não citam nenhuma das propostas identificadas por Medvidovic e Taylor (2000), o que indica que realmente não se disseminaram para a indústria.

Uma busca na literatura recente (à partir de 2006) indicou que o panorama permanece o mesmo. As propostas de formalismos e métodos de representação de arquitetura são atrelados a domínios específicos de software, como é o caso de Ngoyi et al. (2008) que apresentam uma metodologia de projeto dedicada a sistemas de processamento de vídeo em tempo real e Nakagawa (2006) que apresenta uma proposta para arquitetura de referência utilizando *design patterns*. Sobre a representação de DSSAs, Barber et al. (2002) apresenta um tipo de arquitetura de referência para domínio específico, voltada para a avaliação de software. Ela utiliza diagramas da UML, tabelas e diagramas próprios, podendo ser considerado uma modelagem de UML não bem formada. Utiliza dois diagramas de atores e casos de uso da UML, modificados: um para mostrar funcionalidades e outro para mostrar componentes, fluxograma para apresentar a dinâmica de uso (iteração do processo de avaliação de software), listas e tabelas para mostrar atributos dos componentes e uma figura aberta para mostrar interação entre os componentes principais. Interessante é a diferenciação entre requisitos de domínio, não funcionais e de instalação.

Lankhorst (2004) tenta solucionar o problema de maneira diferente, repete que há um problema do tipo e propõe um modelo de um *workbench*, um software, com o propósito de servir à modelagem de arquiteturas de software voltadas para a integração de empresas (no paradigma do *Business Process Modeling*).

Dessa maneira, essa seção apresentou a definição de arquitetura de software, sua importância, tipos e a situação atual e recomendações para a descrição de arquiteturas de

software. Uma das conclusões importantes é que não há um padrão de fato a ser seguido para documentar arquiteturas do tipo DSSA, como a proposta neste trabalho.

2.4. SINALIZAÇÃO DIGITAL E PAINÉIS *TOUCHSCREEN*

A Sinalização digital (do inglês *Digital Signage*) é definida como a comunicação em espaços públicos utilizando tecnologia digital (CHEN et al. 2009). Kircher (2008) define *digital signage* como “a comunicação externa que aparece em algum tipo de painel (display) digital”. Para o autor, “o propósito do Digital Signage é comunicar algo ao público”.

O avanço recente nas tecnologias de monitores e painéis eletrônicos (displays), incluindo robustez e diminuição de custos associadas a esses equipamentos, é um exemplo e pouco a pouco transforma a comunicação visual. É possível ver sinais no cotidiano da substituição dos tradicionais cartazes em papel por avisos em painéis (displays) digitais. O cardápio, a programação dos cinemas, direções, avisos, propagandas comerciais ou entretenimento a passageiros de transportes coletivos, por exemplo, são encontradas facilmente.

O estudo científico de softwares e as tecnologias em si, porém, estão iniciando. Segundo Chen et al. (2009) e Wang (2010), é uma área de estudos recente e que está avançado de maneira rápida. A busca do termo na base da ISI indica 29 artigos sendo 22 de conferências e 7 de revistas, dispostos conforme Figura 3. Os artigos de revista versam fundamentalmente nas tecnologias, hardware, com exceção de dois: Burke (2009) e Storz et al. (2006).

Figura 3. Publicações com o termo Digital Signage no Web of Science.

Fonte: dados obtidos na base Web of Science, consultado em julho de 2010 (ISI, 2010).

Uma aplicação desta categoria possui três componentes básicos (KIRCHER, 2008):

- **Hardware.** É o primeiro e mais evidente componente. A parte principal é o próprio display digital que atualmente é oferecido com diferentes tecnologias, LCD, plasma, OLED, etc.; com diferentes tamanhos e formatos;
- **Software de interação.** Existem empresas que desenvolvem ferramentas específicas para esse tipo de comunicação. São softwares que envolvem animação, mas de uma maneira diferenciada. Eles permitem a edição da programação e interações com os usuários, bem como a gestão da programação *on line*. Isso é, eles permitem que os usuários finais interajam com o painel utilizando interfaces avançadas como controle de visão (*eye tracking*), voz, manipulação de objetos com toques; e
- **Rede e softwares de controle de rede.** Uma aplicação de *Digital Signage* deve contemplar uma rede de computadores, tal que se possa operar remotamente a programação dos displays. Isso permite o controle de displays em grandes áreas (como uma cidade, um centro de compras inteiro, etc.) e alterar a informação conforme alterações no público ou no contexto. De um lado há o

usuário gestor da campanha (profissional de marketing) que escolhe os conteúdos, a região geográfica que cada conteúdo pode ser apresentado e regras para coordenação do conteúdo (por exemplo, ao apresentar certo material outro deve ser apresentado imediatamente). Na ponta, está o usuário final (público) que podem interagir com o painel e alterar algumas condições, alterando o conteúdo apresentado.

As duas principais tecnologias de painéis (displays) empregadas atualmente são o LCD e o Plasma (KIRCHER, 2008). Os textos da área discutem as vantagens e desvantagens de cada uma das tecnologias. A confiabilidade, desempenho e outros parâmetros são significativamente similares, devido aos avanços nas duas áreas. O assunto não será tratado aqui porque foge ao escopo do trabalho, que versará sobre o software e não o dispositivo em si.

Há várias outras tecnologias que irão impactar a área. Segundo Kircher (2008), as mais promissoras são projeção *front-or-rear* (capacidade de projetar em duas diferentes posições), *Organic LED's* (OLEDs), IPTV (TV's digitais com interatividade), *Field Emission Display* (FED) e o *Electronic Paper*. E há também as tecnologias para televisores e telas 3D. Elas ainda não são tão acessíveis como as duas anteriormente mencionadas.

O importante é que a popularização de dispositivos com essas tecnologias permitirá ao consumidor manipular painéis (displays), com interfaces tipo *touchscreen*, com o reconhecimento de imagem (reconhecimento de sensações), mais do que o reconhecimento de escrita. A tecnologia do *Internet Protocol Television* facilitará a transmissão de programações de vídeo em redes de painéis digitais, com dados dinâmicos e intensificação da imagem em movimento na comunicação. Pesquisar como esse tipo de tecnologia pode ser utilizada para o gerenciamento de projetos é um aspecto interessante.

Engel (2008) apresenta e classifica as aplicações para o Digital Signage em quatro tipos: 1) *in-store advertising*, isso é, a comunicação interna em lojas; 2) marketing ou comunicação externa; 3) entretenimento para passar o tempo, sendo aquelas aplicações que oferecem entretenimento e propaganda para salas de espera ou transporte público; e 3) comunicação corporativa.

Esta pesquisa enquadra-se no último tipo. O autor comenta: “*enquanto os outros três tipos possuem foco no consumidor externo, [a aplicação corporativa] está no foco interno, sendo um natural e importante complemento*”. Afirma ainda que tem se dado pouca atenção à esse tipo de aplicação.

Essa lacuna pode ser verificada também na lista de áreas de aplicação de quiosques eletrônicos, publicada no site Kiosk Marketplace (KIOSK MARKETPLACE, 2008), no qual não é mencionada a aplicação corporativa e menos ainda a possibilidade de utilização de quiosques em áreas de gerenciamento de projetos. O site contém vários estudos de caso de aplicação de quiosques e painéis digitais, as quais foram verificadas exaustivamente em 2008 e não foi encontrado exemplo de aplicação em gerenciamento de projetos.

2.5. DESAFIOS E OPORTUNIDADES EM SOFTWARES PARA GERENCIAMENTO ÁGIL DE PROJETOS

A Seção 2.1 mostra que o Gerenciamento Ágil de Projetos é um tema de interesse e importância crescentes. Trata-se de uma nova abordagem, particularmente útil para projetos de inovação. Um dos diferenciais dessa abordagem está no uso de técnicas de planejamento simplificadas, baseadas em painéis visuais que permitam o planejamento de forma integrada, tal que toda a equipe de projeto participa da atividade de planejamento.

As vantagens da programação em painéis visuais são várias. A interação da equipe na frente do painel pode permitir a discussão do que se espera em cada item planejado (entrega), melhorando o entendimento de todos os membros e oferecendo oportunidade para que se

discutam as interfaces entre as diferentes tarefas (entregas, pacotes de trabalho, produtos etc.). Outra vantagem é a simplicidade obtida no controle e atualização das informações. O uso dos painéis permite que o apontamento seja feito de forma rápida e lúdica, pois as pessoas podem realizá-lo em um período de descanso, como a pausa para o café, sem a necessidade de acessar um sistema de informação, com senha e outras barreiras cognitivas. Por fim, há a vantagem da gestão à vista, dado que os dados ficam a disposição no próprio ambiente de trabalho, com facilidade de acesso.

A Seção 2.2 apresentou uma revisão dos trabalhos de pesquisa de novas ferramentas para gerenciamento de projetos. Os resultados mostram que existem poucas propostas na literatura científica para avaliar e propor ferramentas de gerenciamento de projetos. Um paradoxo frente a prática, na qual há uma série de ferramentas desenvolvidas para o gerenciamento ágil de projetos. O trabalho mostra também que as ferramentas comerciais desenvolvidas para o Gerenciamento Ágil de Projetos, são baseadas nas arquiteturas tradicionais de gerenciamento de projetos. São softwares *web-based*, com servidor central, e que não se comunicam com outras aplicações. Outro ponto importante, é que eles seguem o paradigma do planejador, contendo muitas informações e sendo voltados para a consulta e cadastro de informações, similar ao que acontece nos sistemas tradicionais de Gestão de Projetos.

Na Seção 2.2 mostra também evidências de autores da área de CSCW, como Shen, Hao e Li (2008), de que procurar novas propostas de ferramentas para gerenciamento de projetos, de forma colaborativa e distribuída, são um desafio importante para a pesquisa.

As Seções 2.3 e 2.4 apresentou algumas técnicas e tecnologias que poderiam ser exploradas para a proposição de novas soluções para o problema de softwares de gerenciamento de projetos. Ela iniciou apresentando o conceito de arquitetura de software, suas definições e que a inovação na arquitetura pode gerar uma nova classe de sistemas.

Especialmente mostrou-se o conceito de Arquitetura de Software de Domínio Específico (DSSA), que é a abstração de elementos estruturais que definem uma classe de software. Em seguida, apresentou-se o conceito de sinalização digital, que são sistemas de informação que utilizam hardwares de painéis eletrônicos em uma arquitetura com um servidor central que controla a distribuição de informações. Mostrou-se também que há tecnologias maduras de painéis *touchscreen*.

Resumindo esses resultados, conclui-se que:

1. A teoria de APM propõe o uso de painéis visuais (usando *post-its*, por exemplo) e mecanismos que permitam o planejamento e controles participativos, com para toda a equipe de projeto;
2. Os painéis visuais são uma solução adotada no APM, mas quando implantados na forma física (cartões autoadesivos) apresentam restrições quanto ao nível de detalhe dos dados armazenados, falta de bases de dados históricos e dependência da co-localização física da equipe;
3. Os softwares para gerenciamento ágil existentes não exploram a questão dos painéis e interação com outros softwares e faltam propostas para avançar no planejamento e controle de projetos distribuídos, que possam apoiar o desenvolvimento de produtos colaborativos;
4. Na área de marketing, está surgindo uma nova classe de sistemas cuja arquitetura se diferencia pelo gerenciamento central da distribuição de informações, denominada Sinalização Digital, e que faz uso de painéis digitais interativos; e
5. Desenhar uma arquitetura de software de domínio é um passo importante para desenvolver uma classe de sistemas em um dado domínio. A arquitetura providencia um guia para o desenvolvimento de soluções, visando a interoperabilidade.

Portanto, o resultado da revisão é a identificação de uma oportunidade interessante de pesquisa. Contribuir com o problema da criação de softwares para gerenciamento de projetos distribuídos por meio da exploração conjunta dos conceitos de: Gerenciamento ágil de projetos, sinalização digital, telas *touchscreen* e arquitetura de software.

A oportunidade é a proposição de uma **arquitetura de software de domínio específico**, utilizando o **paradigma da sinalização digital**, de forma a distribuir informações de gerenciamento de projetos para **telas *touchscreen* distribuídas**, na empresa, fornecedores e clientes, permitindo que **equipes ágeis de gerenciamento de projetos** possam interagir na frente de quadros visuais e, ao mesmo tempo, garantir um controle rigoroso e detalhado das informações de projeto.

No próximo capítulo, apresenta-se o método utilizado para o desenvolvimento desta oportunidade.

3. MÉTODO DE PESQUISA

A seção apresenta o método de pesquisa. Inicia-se com a apresentação de objetivos e justificativas, Seções 3.1 e 3.2. Em seguida, na Seção 3.3 apresenta-se a classificação do método e etapas de pesquisa.

3.1. OBJETIVO

O objetivo da pesquisa é propor uma nova arquitetura de software de domínio específico para a construção de sistemas de gerenciamento de projetos distribuídos, considerando o planejamento iterativo e visual do gerenciamento ágil de projetos e segundo o paradigma da sinalização digital.

Os objetivos específicos são:

- **Propor arquitetura de software de domínio específico para gerenciamento ágil de projetos distribuídos.** Elaborar requisitos e documentação da arquitetura empregando a análise da literatura de gerenciamento ágil de projetos e de software para engenharia colaborativa, considerando os conceitos de *Digital Signage*.
- **Desenvolver protótipo para verificar a viabilidade e provar o conceito de arquitetura proposto.** Observar a viabilidade da arquitetura proposta quanto à sua exequibilidade por meio do desenvolvimento de um protótipo.

3.2. JUSTIFICATIVA

A análise da literatura fundamental demonstra que não há soluções apoiando um dos aspectos fundamentais do ágil que é a gestão visual. Os recursos utilizados atualmente como a lousa branca com “*post-its*”, mostra-se inapropriado para projetos que envolvem equipes que não estão próximas geograficamente. Além disso, as lousas não armazenam o histórico do

projeto, impossibilitam a geração de relatórios e gráficos fundamentais para gerentes e o seu uso torna-se inviável para empresas que possuem diversos projetos sendo executados simultaneamente, conforme apresentado na revisão bibliográfica.

Paralelamente, um novo conceito de distribuição de informação, com a utilização de software, a arquitetura de sinalização digital, vem surgindo, fazendo uso de painéis LCD (*Open Whall*) sensíveis ao toque (*touchscreen*), cada vez mais baratos e acessíveis (ELO, 2009).

Portanto, há quatro questões importantes de pesquisa:

1. Seria viável o uso de uma solução computacional de painel eletrônico interativo no gerenciamento de projetos distribuídos, com o uso do conceito de gerenciamento ágil de projetos? A resposta a essa pergunta, depende da resposta de três questões específicas.
2. Quais os requisitos de referência e elementos para o desenvolvimento dessa arquitetura?
3. Como pode ser descrita uma arquitetura que atenda a esses requisitos?
4. A arquitetura proposta é tecnicamente viável?

3.3. ETAPAS E MÉTODOS DA PESQUISA

Esta pesquisa combina vários procedimentos para o seu completo desenvolvimento durante seis etapas, sendo duas de levantamento bibliográfico geral e específico, uma de estudo de tecnologias, duas de desenvolvimento da arquitetura e protótipo e, por fim, a construção de um exemplo de aplicação. Cada uma das etapas está descrita a seguir:

1. **Identificação de requisitos da arquitetura gerais.** A primeira fase consistiu no desenvolvimento dos requisitos da arquitetura para painel no contexto do gerenciamento ágil de projetos (SOMMERVILLE & SAWYER, 1997). Isso foi feito a

partir de uma revisão bibliográfica, consolidação do conhecimento gerado nas disciplinas cursadas e análises de casos já realizados pelo grupo de pesquisa.

2. **Comparação de modelos existentes de painéis visuais para gerenciamento ágil de projetos.** Foi realizada uma comparação específica dos modelos de painéis visuais existentes para o planejamento e controle no Gerenciamento Ágil de Projetos.
3. **Desenvolvimento da arquitetura de domínio de software específico para gerenciamento ágil de projetos distribuídos.** Essa etapa foi realizada de maneira paralela durante toda a pesquisa. Após a identificação dos primeiros requisitos, ainda na fase de revisão bibliográfica, começaram a serem construídos esboços, anotados em artefatos do tipo UML, armazenados ao longo do projeto. Os esboços geraram uma proposta inicial utilizada para o desenvolvimento do protótipo. Atualizações nessa proposta foram sendo realizadas conforme as observações coletadas no decorrer do desenvolvimento do protótipo. O resultado é uma documentação da arquitetura de referência, que serve, portanto, para o desenvolvimento de uma nova classe de softwares de gerenciamento de projetos, sendo: um conjunto de requisitos da arquitetura, os principais componentes (módulos) de uma solução do tipo, a interação entre eles quando em tempo de execução e a relação com outros hardwares e fronteiras do sistema.
4. **Estudo das tecnologias envolvidas.** Conhecer e estudar as tecnologias básicas a serem empregadas no projeto, gerando uma concepção inicial do sistema, ou seja, um modelo conceitual. As tecnologias investigadas foram: a) Monitores LCD *touchscreen*; b) Tecnologia de programação PHP para o *Web service* e c) desenvolvimento da aplicação (*frontend*) em Adobe Flex.
5. **Modelagem e desenvolvimento de um protótipo para testar o conceito da arquitetura.** Nessa etapa foi desenvolvido um protótipo utilizando o desenho inicial

da arquitetura obtido na etapa 3 e os conhecimentos obtidos na etapa 4. Os problemas e oportunidades observados foram registrados e a arquitetura foi atualizada sempre que necessário. O resultado dessa etapa foram os códigos, a preparação do servidor com o sistema e os artefatos da documentação relacionados com a implementação do protótipo: diagrama de implantação.

6. **Verificação de exequibilidade por meio de exemplo de aplicação.** A avaliação do protótipo é inicial e visou apenas verificar a sua exequibilidade. O protótipo foi aplicado nos projetos internos de pesquisas do Grupo de Engenharia Integrada¹⁰.

¹⁰ <http://numa.sc.usp.br/grupoei/>

4. ESTABELECIMENTO DA ARQUITETURA DE DOMÍNIO ESPECÍFICO PARA SISTEMAS DE GESTÃO ÁGIL DE PROJETOS DISTRIBUÍDOS

O capítulo apresenta primeiramente as funcionalidades para um painel eletrônico baseado em painéis físicos, que foram utilizados para a identificação dos requisitos de arquitetura e, posteriormente, apresenta a estrutura da documentação da arquitetura de referência, proposta principal desse trabalho.

4.1. FUNCIONALIDADES DE UM PAINEL ELETRÔNICO BASEADO EM PAINÉIS FÍSICOS

As ferramentas de gestão visual do APM incluem relatórios gráficos, bem como os quadros visuais para planejar e controlar o gerenciamento dos projetos. O principal painel proposto é o de tarefas, ou *task board*, que serve o propósito duplo de proporcionar a equipe um mecanismo conveniente para organizar seu trabalho e um modo visual e simplificado da quantidade de trabalho que resta fazer, de forma que seja possível identificar quais tarefas estão em andamento e quais são as próximas que devem começar. Normalmente esses painéis são quadros brancos ou de cortiça, com divisões em colunas, onde estão distribuídos cartões de papel, do tipo recados autoadesivos com descrição das tarefas planejadas.

Para o presente trabalho, o critério para a escolha dos painéis analisados foi o nível de detalhamento fornecido nos textos. Esses quadros foram definidos e descritos com um nível de detalhamento não comumente encontrado em outros trabalhos, viabilizando o estudo aprofundado de cada característica proposta. Dessa maneira, os quadros analisados estão apresentados na Tabela 7.

Tabela 7. Modelos de painéis físicos analisados.

Autor	Fonte	Título
Mike Cohn	Livro	Agile Estimating and Planning
Edivandro Conforto	Dissertação Mestrado	Método Visual para Gestão Ágil de Escopo e Tempo: Proposta e Aplicação em Empresas de Base Tecnológica
Ken Schwaber	Livro	Agile Project Management with Scrum
Gary Chin	Livro	Agile Project Management: How to Succeed in the Face of Changing Project Requirements

4.1.1. Levantamento dos modelos de painéis físicos existentes

Como o APM foi proposto inicialmente para a área de desenvolvimento de softwares, os modelos de quadros apresentam características e terminologias próprias da engenharia de software. O quadro de tarefas proposto por Cohn (2005), (anteriormente apresentado na Figura 1) apresenta as seguintes colunas:

- *Story*: onde ficam os cartões com os *story cards*¹¹ do projeto de cada iteração planejada;
- *To Do*: com cartões que descrevem as tarefas que devem ser feitas para implementar um determinado *story card*;
- *Tests ready*: com a marcação se existe um teste planejado para o aceite do *story card*;
- *In Process*: mostra os cartões que descrevem as tarefas que estão em andamento no projeto;
- *To Verify*: apresenta os cartões de tarefas que precisam ser verificadas, para depois serem consideradas como realizadas;
- *Hours*: apresenta a soma das horas de trabalho restante para o *story card*.

¹¹ Cartões utilizados no desenvolvimento de software para descrever uma necessidade/funcionalidade do produto que será desenvolvido.

Entende-se que para criar a possibilidade, ao mesmo tempo, de tomadas de decisões participativas e interação entre os membros da equipe é necessário um ambiente colaborativo, que apresente as informações atualizadas. Métodos ágeis de desenvolvimento de software, tal como o SCRUM, incentivam para tanto, o uso desses quadros físicos em locais de fácil acesso aos membros do projeto, com adesivos informativos.

Conforto (2009) também fez uma proposta de quadro visual físico com o uso de recados autoadesivos, que funcionam como cartões de entrega, porém, a aplicação é voltada para projetos inovadores de produtos físicos. Denominado PVPCP (Painel Visual de Planejamento e Controle de Projetos) é composto do conjunto de fases descritos no MFE (modelo de fases e entregas), mais a indicação das avaliações de transição de fases (Phase Gate Review), distribuídos segundo uma escala de tempo, mensal ou anual, dependendo das necessidades e tipos de projetos desenvolvidos.

O PVPCP (Figura 4) contempla o planejamento e controle do escopo do projeto com foco no desenvolvimento por meio de iterações. Dentro de cada fase do MFE, podem ocorrer várias iterações, seguindo um ciclo evolutivo. O interior do painel, no espaço delimitado para cada fase, é onde devem ser afixados os denominados cartões de entregas. Cada cartão de entrega possui os seguintes dados sobre a entrega do projeto:

- apelido do projeto;
- nome da entrega e código (ex.: Plano de projeto, E1);
- data da entrega (planejada e real); e
- responsável pela entrega e tempo estimado de execução.

Para melhor diferenciação entre as entregas dos projetos, recomenda-se o uso de recados autoadesivos coloridos, no qual diferentes cores de recados poderão representar diferentes projetos, ou diferentes áreas ou departamentos envolvidos no projeto. Em outros casos, cada cor pode representar subprojetos ou grupos de entregas.

Figura 4. Painel Visual de Planejamento e Controle de Projetos (PVPCP).
Fonte: CONFORTO, 2009.

Além do PVPCP, Conforto (2009) sugere a utilização de outro painel denominado QPFS (quadro de planejamento fino semanal), apresentado na Figura 5. Ele funciona como uma agenda semanal coletiva, visual e interativa, utilizando os cartões coloridos para a definição de pacotes de trabalho. Esse painel é composto de um quadro onde são inseridos os dias da semana na parte superior e na lateral esquerda, está o período do dia (manhã e tarde).

Cada cartão inserido no QPFS deve conter o nome da entrega relacionada, conforme inserida no PVPCP, o nome da atividade ou pacotes de trabalho, nome do responsável, e data prevista para sua finalização. O QPFS também utiliza o conceito de iteração. Cada semana pode ser considerada uma iteração com resultados rápidos que contribuam para a conclusão de entregas definidas no PVPCP, adicionando valor para o cliente e para a equipe de projeto.

Quadro de Planejamento Fino (Semanal)					
	Segunda	Terça	Quarta	Quinta	Sexta
Período Manhã					
Período Tarde	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira

Figura 5. Quadro de Planejamento Fino Semanal (QPFS).
Fonte: CONFORTO, 2009.

Já Schwaber (2004) sugere que o planejamento e controle do projeto ocorram por meio de uma tabela denominada *Product Backlog*. Nessa tabela, são indicados os requisitos do projeto, a priorização desses requisitos e o seu agrupamento em iterações (de acordo com a priorização). Segundo Schwaber (2004), a *Product Backlog* será constantemente alterada, pois as condições do projeto poderão exigir o re-planejamento das iterações, realocando itens entre elas.

Uma vez que a *Product Backlog* está concluída, a equipe do projeto passa a planejar e controlar as iterações por meio de outra tabela denominada *Sprint Backlog*. Nela são definidas as tarefas que tornam os requisitos da iteração na *Product Backlog* em um conjunto de atividades potencialmente entregáveis. Estas tarefas devem ser divididas de forma que cada uma leve cerca de 4 a 16 horas para terminar (caso a duração seja maior, é considerado que ainda não foram apropriadamente definidas).

Backlog Description	Initial Estimate	Adjustment Factor	Adjusted Estimate	work remaining until completion						
				1	2	3	4	5	6	7
Title Import				256	209	193	140	140	140	140
Project selection or new	3	0.2	3.6	3.6	0	0	0	0	0	0
Template backlog for new projects	2	0.2	2.4	2.4	0	0	0	0	0	0
Create product backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Create sprint backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	0	0	0	0	0	0
Sprint-1	13	0.2	15.6	16	0	0	0	0	0	0
Create a new window containing product backlog template	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Create a new window containing sprint backlog template	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Burndown window of product backlog	5	0.2	6	6	6	0	0	0	0	0
Burndown window of sprint backlog	1	0.2	1.2	1.2	1.2	0	0	0	0	0
Display tree view of product backlog, releases, prints	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Display burndown for selected sprint or release	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Sprint-2	16	0.2	19.2	19	19	1.2	0	0	0	0
Automatic recalculating of values and totals	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
As changes are made to backlog in secondary window, update burndown graph on main page	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Hide/automatic redisplay of burndown window	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
Insert Sprint capability ... adds summing Sprint row	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Insert Release capability ... adds summary row for backlog in Sprint	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Owner/assigned capability and columns optional	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Print burndown graphs	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Sprint-3	14	0.2	16.8	17	17	17	0	0	0	0
Duplicate incomplete backlog without affecting totals	5	0.2	6	6	6	6	6	6	6	6
Note capability	6	0.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2
What-if release capability on burndown graph	15	0.2	18	18	18	18	18	18	18	18
Trend capability on burndown server	2	0.2	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Publish facility for entire project, publishing it as HTML web pages	11	0.2	13.2	0	0	13	13	13	13	13
Future Sprints	39	0.2	46.8	34	34	47	47	47	47	47
Release-1				85	70	65	47	47	47	47

Figura 6. Product Backlog. Fonte: SCHWABER, 2004.

A *Sprint Backlog* é uma imagem em tempo real do trabalho que a equipe pretende realizar durante a iteração. Suas linhas representam as tarefas e as colunas os 30 dias de duração da iteração. Uma vez que uma tarefa é definida, o número estimado de horas restantes para completar a tarefa é colocado na intersecção da tarefa com o dia da iteração pela pessoa responsável pela tarefa (SCHWABER, 2004).

A Figura 7 mostra um exemplo de uma *Sprint Backlog* baseada em uma planilha.

Task Description	Originator	Responsible	Status (Not Started/ In Progress/ Completed)	Hours of work remaining unit											
				1	2	3	4	5	6	7	8	9	10	11	12
Meet to discuss the goals and features for Sprint 3-6	Danielle	Danielle/ Sue	Completed	20	0	0	0	0	0	0	0	0	0	0	0
Move Calculations out of Crystal Reports	Jim	Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Get KEG Data		Tom	Completed	12	0	0	0	0	0	0	0	0	0	0	0
Analyse KEG Data - Title		George	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Analyse KEG Data - Parcel		Tim	Completed	12	12	12	12	12	4	4	4	0	0	0	0
Analyse KEG Data - Encumbrance		Josh	In Progress							12	10	10	10	10	10
Analyse KEG Data - Contact		Danielle	In Progress	24	24	24	24	12	10	8	6	6	6	6	6
Analyse KEG Data - Facilities		Allen	In Progress	24	24	24	24	12	10	10	10	10	10	10	10
Define & build Database		Barry/ Dave	In Progress	80	80	80	80	80	80	60	60	60	60	60	60
Validate the size of the KEG database		Tim	Not Started												
Look at KEG Data on the G:\		Dave	In Progress	3	3	3	3	3	3	3	3	3	3	3	3
Confirm agreement with KEG		Sue	Not Started												
Confirm KEG Staff Availability		Tom	Not Started	1	1	1	1	1	1	1	1	1	1	1	1
Switch JDK to 1.3.1. Run all tests.		Allen	Not Started	8	8	8	8	8	8	8	8	8	8	8	8
Store PDF files in a structure		Jacque	Completed	8	0	0	0	0	0	0	0	0	0	0	0
TopLink. Cannot get rid of netscape parser		Richard	Completed	4	0	0	0	0	0	0	0	0	0	0	0
Buld test data repository		Barry	In Progress	10	10	10	10	10	10	10	10	8	8	8	8
Move application and database to Qual (incl Crystal)		Richard	Completed	4	4	4	4	4	4	4	0	0	0	0	0
Set up Crystal environment		Josh	Completed	2	2	2	2	1	1	1	0	0	0	0	0
Test App in Qual		Sue	In Progress												20
Defining sprint goal required for solution in 2002		Lynne	In Progress	40	40	40	40	40	40	40	38	38	38	38	38
Reference tables for import process		Josh	In Progress												
Build standard import exception process		Josh	In Progress									12	12	12	10
Handle multiple file imports on same page		Jacque	Disregarded												
Migrate CruiseControl Servlet to iWS 6.0 (landcc_7101) server		Allen	Not Started	4	4	4	4	4	4	4	4	4	4	4	4
Create web server for Qual on PF1D8		Allen	Completed	1	0	0	0	0	0	0	0	0	0	0	0
LTCS Disk		Danielle/ George	In Progress	12	12	12	12	8	8	8	8	8	8	8	8
Follow thru with questions about KEG data to Sue/Tom, re: Keg, LTO	Jacque	Danielle	Completed	10	10	10	10	10	8	8	0	0	0	0	0
Map KEG data to Active Tables - see also #14	Jacque	Jacque/ Allen	In Progress	50	50	50	50	50	50	50	50	50	50	50	50
Preparer SQL to import from KEG tables to Active Tables	Jacque	George	In Progress	25	25	25	25	25	25	25	25	25	25	25	25

Figura 7. Sprint Backlog. Fonte: SCHWABER, 2004.

Chin (2004) sugere em seu modelo dois itens que devem ser monitorados: *tasks* e *action items*. Segundo Chin (2004), *task* é uma atividade do projeto que foi definida na fase de planejamento e que aparece como um dos itens do gráfico de Gantt. Já um *action item* é uma atividade definida durante a fase de execução do projeto, geralmente pequena demais para ser adicionada formalmente ao gráfico de Gantt, mas igualmente importante para o andamento do projeto. Eventualmente um *action item* pode tornar-se uma *task* ao ser adicionado ao gráfico de gantt.

Em seu *template*, os *action items* e as *tasks* devem estar separados em duas seções diferentes no mesmo documento. Nele, a primeira coluna deve ser marcada quando uma *action item* ou *task* é concluída. A segunda coluna corresponde ao nome e uma breve descrição do *action item* ou *task*. Na coluna *Priority* deve ser definida a prioridade do *action item* ou *task*, sendo: *High*, *Medium* ou *Low*, facilitando a identificação de elementos mais importantes para o projeto. Já na coluna *Owner* deve ser especificado o responsável pela execução do *action item* ou *task*. As colunas *Date logged* e *Date due* correspondem respectivamente a data em que a *action item* ou *task* foi adicionada a lista e a data final para conclusão da *action item* ou *task*. Por fim a coluna *Notes* é utilizada para qualquer informação relevante ao *action item* ou *task*.

A Figura 8 mostra um exemplo de uma *Action Item and Tasks Tracking Process*.

Done	Action Items	Priority (H-M-L)	Owner	Date Logged	Date Due	Notes
	Action item #2	M	Wilson	May 3	Jun 5	Enter comments here
	Action item #5	M	Price	May 15	Jun 10	Enter comments here
◦	Action item #1	M	Chin	May 3	May 30	Enter comments here
◦	Action item #3	L	Smith	May 5	May 22	Enter comments here
◦	Action item #4	M	Smith	May 12	Jun 20	Enter comments here
Done	Tasks (from timeline)	Priority	Owner	Date Logged	Date Due	Notes
	Task #1		Chin	May 3	May 9	Enter comments here
	Task #2		Kane	May 7	May 25	
◦	Task #3		Smith	May 14	Jun 12	Enter comments here
◦	Task #4		Mac	May 24	Jun 18	
◦	Task #5		Chin	Jun 1	Jun 20	

Project Logo

Figura 8. Action Item and Tasks Tracking Process. Fonte: Chin (2004)

4.1.2. Elementos dos painéis físicos

Por meio da análise dos painéis propostos pelos autores citados, foram identificadas características que poderão ser utilizadas em uma versão eletrônica de um painel para gerenciamento de projetos. Essas características são:

- Divisão dos resultados (chamados entregas) em iterações (chamados também *Sprints*);
- Uso de listas de tarefas (Chamados também de *Backlog List* ou Lista de Entregas);
- Organização visual por meio da metáfora do adesivo autocolante (*Post-its*);
- Definição do responsável pela entrega;
- Status da entrega (iniciada, em andamento ou concluída);
- Estimativa de tempo para conclusão da entrega (em horas ou dias);
- Data de início e fim;
- Validar entrega antes de concluir;
- Divisão do painel em Fases do Projeto; e
- Utilização de *Gates*.

Dessa maneira, o Quadro 5 especifica os elementos e características contidos nas concepções de painéis propostas pelos autores, destacando-as para uma possível aplicação em uma solução computacional.

Alguns desses elementos serão usados posteriormente, na Seção 5.1, para estabelecimento dos requisitos de arquitetura, que servirão como guia para o desenvolvimento da documentação da arquitetura proposta. Para facilitar a compreensão, a próxima seção apresenta a estrutura dessa documentação.

Quadro 5. Elementos e características dos painéis físicos.

	Mike Cohn	Conforto (PVPCP)	Conforto (QPFS)	Ken Schwaber	Gary Chin
Divisão das entregas em <i>Sprints</i> (iterações)		X	X	X	
Uso de <i>Backlog List</i>	X			X	
Organização visual por <i>Post-its</i>	X	X	X		
Definição do responsável pela entrega		X	X	X	X
Status da entrega	X				X
Estimativa de tempo para conclusão da entrega	X	X	X	X	X
Data de início e fim		X	X		X
Validar entrega antes de concluir	X				
Divisão do painel em Fases		X			
Utilização de <i>Gates</i>		X			

4.2. ESTRUTURA DA DOCUMENTAÇÃO

A arquitetura para Sistemas de Gerenciamento de Projetos Distribuídos apresentada possui dois objetivos fundamentais: transmitir uma nova concepção de arquitetura para softwares de gerenciamento de projetos; servir de guia para a evolução de um conjunto de componentes desenvolvidos e que implementam a concepção proposta de maneira física, e que poderá evoluir.

O primeiro objetivo, portanto, é transmitir uma nova concepção de construção de sistemas de gerenciamento de projetos, fazendo uso dos sistemas existentes e painéis digitais para implementar o conceito de Sinalização Digital e apoiar o gerenciamento ágil distribuído.

Os usuários da documentação nesse primeiro caso são pesquisadores na área de softwares para gerenciamento de projetos e CSCD. Eles utilizarão a documentação da arquitetura para encontrar inspiração para construir outros sistemas distintos para gerenciamento de projetos ou áreas correlatas; ou para implementar o conceito geral de forma totalmente distinta, com outra tecnologia.

O segundo objetivo é o de descrever um ambiente de software com componentes, softwares tradicionais de gerenciamento de projetos e painéis eletrônicos digitais de forma a permitir a continuidade do desenvolvimento do sistema (evolução e manutenibilidade). Os usuários, nesse caso, são pesquisadores do próprio grupo de pesquisa ou outros interessados em contribuir para o desenvolvimento desse ambiente. Não precisam entender apenas a filosofia do sistema, mas também, como ele foi construído fisicamente, isso é, precisam de um nível de detalhe maior.

Os usuários da documentação podem então ser sintetizados na Tabela 8.

Tabela 8. Perfil de usuários da documentação da arquitetura.

Stakeholders	Possíveis interesses	Nível de Detalhe de Implementação	Visão Estrutural	Visão de Comportamental	Visão Física
Pesquisadores da área de CSCD	Interesse em entender a proposta de sinalização digital e como ela pode ser incorporada para a criação de sistemas distribuídos de gerenciamento de projetos.	Baixo	ID	IG	x
Profissionais Gestores de Projeto	Compreender o conceito de gestão ágil distribuída e tê-lo como referência para a orientação da implantação de sistemas de Gestão de Projetos (PM Systems) e Enterprise Project Management Systems.	Baixo	IG	IG	x
Profissionais que atuam com o Desenvolvimento Sistemas de Gerenciamento de Projetos	Utilizar conceitos da proposta em casos reais de implementação de sistemas de Gestão de Projetos (PM Systems) e Enterprise Project Management Systems	Alto	ID	ID	X
Pesquisadores interessados em contribuir com o sistema desenvolvido	Compreender os componentes existentes e como podem contribuir desenvolvendo novos componentes para o ambiente	Alto	Alto	ID	IG
Pesquisadores do Grupo de Pesquisa	Realizar a manutenção do protótipo existente e planejar a sua evolução futura.	Alto	Alto	ID	ID

****Legenda:** ID – Informação Detalhada; IG- Informação geral; x – Nenhuma informação;

Considerando as necessidades acima identificadas, enquadra-se a arquitetura proposta como sendo do tipo DSSA, isso é, uma arquitetura de software de domínio específico. O

domínio específico no caso são Sistemas para Gerenciamento Ágil de Projetos Distribuídos. O objetivo da arquitetura é apresentar a estrutura dessa nova classe de sistemas que está sendo proposta (Arquitetura de referência) e descrever um ambiente construído para a implementação e evolução de sistemas do tipo. Deverá orientar diferentes profissionais para a continuidade do desenvolvimento de um sistema como esse, apoiando, portanto, outros trabalhos de pesquisa que poderão ser realizados no futuro.

A documentação seguiu as diretrizes da literatura. Utiliza as três visões para a documentação, conforme proposto por Nakagawa (2006), sendo elas: visão estrutural; visão comportamental; e visão física. Incorpora as recomendações de Clements et al.(2004) na estrutura do documento e detalhamento de cada visão, apresentadas na Seção 2.3.

Para a descrição da visão e construção dos artefatos, a linguagem escolhida foi a UML. Trata-se de uma linguagem padrão para a elaboração da estrutura de projetos de software destinada a visualizar, especificar, construir e documentar os artefatos de um sistema complexo de software (BOOCH et al, 2005). Seguiu-se este padrão tanto quanto possível, isso é, seguem-se as convenções da UML, complementando-as ou modificando-as, conforme a necessidade.

A escolha da UML deve-se ao fato de ser um padrão amplamente conhecido e utilizado na indústria e na academia. É também indicada por Kruchten et al.(2006) como o padrão mais utilizado, o que foi observado nas propostas científicas encontradas.

A documentação é formada por um conjunto de artefatos organizados em pacotes formando dois grandes grupos: a) Arquitetura de Referência; e b) Ambiente e infraestrutura. Dentro de cada grupo há conjuntos de pacotes com artefatos, classificados segundo uma das três visões: estrutura, comportamental e física. Conforme a Figura 9.

Figura 9. Estrutura da documentação da arquitetura para Sistemas de Gerenciamento Ágil de Projetos Distribuídos

A estrutura da documentação deve facilitar a consulta para os diferentes usuários da arquitetura. Pesquisadores e profissionais interessados na estrutura mais geral, no conceito de organização do sistema utilizado poderão consultar apenas o pacote arquitetura de referência. Ela é suficiente para se compreender a organização do sistema, principais componentes e como são os seus comportamentos.

Os pesquisadores interessados em contribuir com o desenvolvimento do ambiente em si, propor novos componentes ou experimentar integrações com outras propostas, podem consultar toda a documentação, incluindo o pacote de alocação (ambiente e infra-estrutura).

O Quadro 6 classifica os usuários da documentação, segundo os pacotes de artefatos de interesse.

Quadro 6. Pacotes de documentação por perfil de usuário da arquitetura.

Stakeholders	Possíveis interesses	Pacote de Interesse
Pesquisadores da área de CSCD	Interesse em entender a proposta de sinalização digital e como ela pode ser incorporada para a criação de sistemas distribuídos de gerenciamento de projetos.	Arquitetura de Referência
Profissionais Gestores de Projeto	Compreender o conceito de gestão ágil distribuída e tê-lo como referência para a orientação da implantação de sistemas de Gestão de Projetos (PM Systems) e Enterprise Project Management Systems.	
Profissionais que atuam com o Desenvolvimento Sistemas de Gerenciamento de Projetos	Utilizar conceitos da proposta em casos reais de implementação de sistemas de Gestão de Projetos (PM Systems) e Enterprise Project Management Systems	Arquitetura de Referência e Ambiente e Infraestrutura
Pesquisadores interessados em contribuir com o sistema desenvolvido	Compreender os componentes existentes e como podem contribuir desenvolvendo novos componentes para o ambiente	
Pesquisadores do Grupo de Pesquisa	Realizar a manutenção do protótipo existente e planejar a sua evolução futura.	

A primeira distinção importante sobre a arquitetura é que ela não tem como objetivo o desenvolvimento de software completo para a gestão de projetos, como é comum nas soluções mais tradicionais (MS Project, Primavera etc.) e nas soluções que vem surgindo na área de gerenciamento de projetos (JIRA, VersionOne, etc..). O objetivo é descrever um conjunto de componentes que juntos apoiem a gestão de projetos. Parte desses componentes são sistemas tradicionais de gerenciamento de projetos como os citados, outra parte são hardwares de interação humano-computador via painéis digitais, e, complementando estes dois conjuntos de componentes disponíveis, um conjunto de componentes especialmente projetados que permitam a sincronização entre os dados atualizados e visualizados nos painéis e as ferramentas de gerenciamento de projetos tradicionais.

Uma solução desse tipo permite que o planejador (*controller*, gerente de programas, especialista em gerenciamento de projetos) continue a utilizar a sua ferramenta de gerenciamento de projetos especializada, obtendo relatórios detalhados e utilizando recursos

avançados. Ao mesmo tempo, permite que os membros da equipe de projeto tenham acesso apenas às informações necessárias para a sua atividade que é a de desenvolver o produto ou software, por meio de interfaces de painéis digitais que são mais amigáveis e permitem o trabalho em conjunto, isto é, a atualização de dados e o planejamento de forma conjunta.

Quanto ao ciclo de vida, o sistema de informação que resulta da arquitetura está inserida na fase de planejamento do projeto, conforme o modelo de desenvolvimento de produtos proposto por Rozenfeld et al (2006), em destaque na Figura 10.

Figura 10. Modelo de referência para o PDP. Fonte: Rozenfeld et al (2006)

O principal diferencial dessa solução em relação às existentes é a capacidade de trabalhar em conjunto com outras ferramentas de Gerenciamento de Projetos, como o Primavera Professional Project Management, Microsoft Project e o dotProject.

Para que isso seja possível, propõe-se um conjunto formado por quatro elementos básicos que são os módulos gerais do sistema, que podem ser consultados na Figura 11.

Os componentes do tipo C (totem, mesas digitais, painéis eletrônicos) são os dispositivos de hardware que apoiam a interação humano computador dentro da equipe de projeto. São dispositivos disponíveis no mercado. Os componentes do tipo D representam

softwares de Gerenciamento de Projetos, também existentes no mercado e que apresentam recursos sofisticados para programação e gerenciamento de recursos do projeto.

(1) Diferentes SGP's são compatíveis com a arquitetura.

(2) Os dados são coletados do SGP Principal e abstraídos em um Web Service.

(3) Aplicações para diferentes dispositivos acessam os dados do SIPRO, de acordo com as permissões e especificações.

(4) Esses dispositivos podem, ou não, estar colocalizados.

***Sistemas para Gerenciamento Ágil de Projetos Distribuídos**

Figura 11. Visão geral da solução.

Dois desses elementos são os componentes inovadores propostos na arquitetura: (A) SIPRO e (B) Pro Visual Control. O módulo SIPRO é o responsável por coletar os dados do(s) sistema(s) de Gerenciamento de Projeto(s) e dos dispositivos de interação com a equipe de

projeto. Ele armazena os dados principais e periodicamente faz a sincronização com todos os softwares por meio de um servidor *Web Service*.

O módulo Pro Visual Control (PVC) contém dados relacionados ao dispositivo específico de hardware (Totem, painel etc.). Ele armazena, por exemplo, a descrição das propriedades do dispositivo (resolução de tela, tamanho etc.) e também do local aonde está instalado, para que a informação possa ser filtrada também conforme o local.

Por exemplo, um diretor pode ter acesso a uma determinada informação, mas pode ser interessante que ele nunca consulte determinados dados em um determinado painel que está, por exemplo, instalado em um cliente. Isso seria um mecanismo que impediria, por exemplo, a divulgação de informação sensível de maneira não intencional. O PVC poderia também incluir regras para alterar o estado do painel em situações específicas, como para transmitir determinada informação urgente, abrir determinado documento relacionado a um item de planejamento e outros.

A meta é que os planejadores, especialistas em gerenciamento de projetos, tenham acesso aos recursos sofisticados (utilizando os softwares de Gerenciamento de Projetos já existentes) e os membros da equipe a recursos adequados ao uso no dia-a-dia de projeto (acessados via painéis visuais).

Softwares como o Microsoft Project ou Primavera são do tipo *desktop*, no qual o principal meio de interação com o usuário ocorre pelo mouse e/ou teclado. Já na solução proposta, a interação ocorre por meio de uma tela *touchscreen*. Essa nova abordagem exige que o sistema seja inovador em sua interface, explorando mais componentes gráficos do que os tradicionais campos de texto comumente utilizados. Está, portanto, adequado aos princípios de simplicidade, planejamento participativo e auto-gestão propostos no paradigma do Gerenciamento Ágil de Projetos, conforme Capítulo 2.

Um sistema de informação construído nos moldes do conceito da Figura 9 pode também ser classificado como um sistema distribuído. As funcionalidades de gestão de projetos estão espalhadas. O planejador utiliza recursos de programação de projetos (nivelamento, PERT-CPM , Monte Carlo, Dados históricos etc.) nos softwares profissionais de gerenciamento de projetos. Membros da equipe acessam funcionalidades de planejamento, edição de planos e alocação de recursos utilizando recursos de painéis digitais.

Cada um deles em diferentes máquinas e componentes. Isso facilitaria a escalabilidade, pois novos componentes poderiam ser adicionados ao ambiente conforme necessidades específicas. As equipes de projeto podem estar geograficamente distribuídas, o que contribui para a utilização da abordagem de gerenciamento ágil de projetos

Dessa forma, obtém-se uma estrutura de *Digital Signage* conforme proposto por Kircher (2008), no qual os três elementos básicos presentes são:

- *Hardware*: O painel LCD *touchscreen* e o computador para processamento dos dados e comunicação;
- *Software para distribuição da informação*: O PVC e o SIPRO que trabalham em conjunto para distribuir as informações para os demais painéis. O PVC contém as funcionalidades para gerenciamento da informação e o SIPRO abstrai as funções de controle e distribuição dos dados; e
- *Rede de painéis eletrônicos*: O sistema ficará localizado em um servidor centralizado sendo acessado por uma rede de computadores ou pela internet.

5. ARQUITETURA DE REFERÊNCIA

O capítulo apresenta a arquitetura de referência, dentro dos objetivos definidos para o projeto. Inicialmente, os requisitos da arquitetura são apresentados, seguido do contexto em que o sistema está inserido. Por fim, são apresentados os casos de uso, diagrama de casos de uso e o cenário da solução.

5.1. REQUISITOS DA ARQUITETURA

O primeiro passo para transferir as funcionalidades dos painéis físicos, da abordagem APM, para um painel digital *touchscreen*, é identificar os requisitos. São eles que devem garantir os princípios propostos na teoria, como a simplicidade do painel físico, a geração de históricos aliada à facilidade de uso, a utilização do conceito de iteração, a adequação ao conceito de sinalização digital.

Os requisitos podem ser divididos em funcionais e não funcionais. Há consenso sobre a definição de requisitos funcionais, como aqueles relacionados a uma função que o sistema deve realizar, “o que o sistema deve fazer” (CHUNG & LEITE, 2007). Os requisitos funcionais são os que expressam funções ou serviços executados por um software. Já os requisitos não funcionais não possuem o mesmo consenso.

Com base nos requisitos gerados por meio da análise de painéis físicos existentes e na identificação de novos requisitos no decorrer deste trabalho, esta etapa visa formar uma síntese dos resultados das análises conduzidas.

A seguir, foram listados os requisitos funcionais, nos quais inicialmente a funcionalidade foi definida seguida do detalhamento do requisito:

RF001: Possuir um módulo para planejamento de projetos inteiramente visual

- a. O planejamento deve ser baseado em iterações;

- b. Deverá ser exibida a iteração atual e a próxima iteração (em maior destaque), seguidas de uma lista com todas as outras iterações em tamanho reduzido. Essa lista inicialmente exibirá as duas seguintes, mas deve possibilitar a navegação entre as outras. Deve-se exibir também a lista de tarefas com as tarefas previstas, mas ainda não planejadas;
- c. A interface deve ser *touchscreen*;
- d. As tarefas serão criadas diretamente nas iterações ou na lista de tarefas, apenas arrastando o ícone de nova tarefa para o local desejado;
- e. Na criação de uma nova tarefa o usuário deve informar: nome da tarefa (obrigatório), datas de início e fim (obrigatórios), esforço (obrigatório), descrição da tarefa (opcional) e alocar os recursos da tarefa (obrigatório);
- f. Quando uma tarefa é criada diretamente em uma iteração, o software deve preencher automaticamente as datas de início e fim com as datas da iteração. Caso o usuário deseje, ele pode alterar essas datas para manter a consistência com o software principal de GP;
- g. O planejamento deve estar sempre disponível, sendo assim uma atividade rotineira e possibilitando o replanejamento; e
- h. As tarefas poderão ser facilmente movimentadas entre as iterações e a lista de tarefas utilizando-se *drag-and-drop*¹².

RF002: Possuir um módulo para acompanhamento de projetos de maneira visual

¹² Nas interfaces gráficas de computadores, *drag-and-drop* (arrastar e largar) é a ação de clicar em um objeto virtual e "arrastá-lo" a uma posição diferente ou sobre um outro objeto virtual.

- a. O acompanhamento do projeto deve utilizar a mesma tela do módulo de planejamento, dispensando a necessidade da navegação entre várias janelas e reutilizando os recursos já oferecidos nesse módulo;
- b. Na iteração atual e na próxima iteração, as tarefas devem ser representadas utilizando a metáfora de cartões autoadesivos;
- c. Nas outras iterações e na lista, as tarefas devem ser representadas apenas pelo Nome da tarefa;
- d. Os cartões devem exibir: Nome da tarefa, Andamento (porcentagem) e os recursos alocados (fotos dos recursos);
- e. Os cartões devem ser de diferentes cores, sendo: amarelo (tarefa não iniciada), azul (tarefa iniciada) e verde (tarefa concluída);
- f. O módulo deve possuir um filtro na iteração atual e na próxima iteração no qual o usuário pode escolher visualizar: todas as tarefas, planejadas, em andamento ou concluídas;
- g. Cada iteração deve exibir o andamento total da iteração (barra de porcentagem) calculado pelo andamento das tarefas; e
- h. A iteração atual e a próxima iteração deve exibir a quantidade de horas trabalhadas naquela iteração.

RF003: Possibilitar registro de plano do projeto baseado em iterações e tarefas

- a. O software deve trabalhar utilizando o conceito de iterações;
- b. Alguns softwares de GP não possuem o conceito de iteração. Nesses casos, quando o projeto for acessado pela primeira vez no PVC, deve ser solicitado a configuração do projeto no qual o usuário informará o tipo de iteração que deseja, sendo elas: semanal, quinzenal ou mensal;

- c. O software deve gerar as iterações automaticamente para o usuário após a configuração, baseando-se na duração total do projeto definida no software principal de GP; e
- d. O software não permite que novas iterações sejam criadas.

RF004: Permitir a associação de usuários às tarefas

- a. Durante a criação ou edição de uma tarefa o software deve permitir que sejam alocados recursos;
- b. O software deve exibir uma lista com os recursos disponíveis e outra com os recursos alocados;
- c. Toda tarefa deve conter pelo menos um recurso alocado;
- d. A dinâmica de alocar ou remover um recurso deve ser por meio de *drag-and-drop*;
- e. As listas exibem o nome e a foto do recurso. A origem da foto pode ser variada, incluindo redes sociais (facebook¹³, twitter¹⁴, etc.); e
- f. Quando um recurso for alocado o software deve permitir que seja configurada a sua dedicação para a tarefa.

RF005: O módulo de acompanhamento de projetos ou tarefas deve permitir que o usuário edite as informações das tarefas, bem como o seu andamento

- a. Quando o usuário clicar na tarefa (toque rápido com o dedo), o software deve abrir uma janela para edição e atualização do seu andamento;

¹³ <http://www.facebook.com/>

¹⁴ <http://twitter.com/>

- b. O usuário pode editar todas as informações da tarefa, bem como alocar ou remover a alocação de algum recurso; e
- c. Na atualização do andamento, o usuário pode alterar o seu progresso (porcentagem), definir as horas trabalhadas até aquele momento e inserir uma breve descrição para esta atualização.

RF006: Apresentar o status das tarefas do projeto com indicadores

- a. Conforme definido no RF002, o cartão deve exibir o andamento da tarefa em porcentagem, alterar sua cor de acordo com seu status (planejada, andamento ou concluída) e a iteração deve exibir uma barra com o seu status (porcentagem).

RF007: Registrar novos requisitos e tarefas durante o acompanhamento de projetos

- a. Conforme definido no RF002, o módulo de planejamento e controle serão os mesmos, dessa maneira, o usuário poderá a qualquer momento no andamento do projeto criar novas tarefas.

RF008: Possibilitar o uso de dados dos sistemas de gerenciamento de projetos (SGP) tradicionais

- a. O software deve se conectar a outros SGPs por meio de *Web Services*.

RF009: Permitir o acesso ao histórico do projeto, depois de finalizado

- a. Mesmo após o término do projeto, o software deve permitir que as iterações e tarefas sejam consultadas; e

- b. Quando a data de fim do projeto for anterior a data atual, as iterações exibidas devem ser as últimas realizadas.

RF010: Quando não utilizado deve exibir notícias da organização

- a. O software deve possuir um módulo para que notícias sejam exibidas quando não estiver sendo utilizado; e
- b. Essas notícias podem originar de uma página da organização ou de redes sociais, como o twitter, por exemplo.

Deve-se ressaltar que para o software desenvolvido não foi diferenciado Tarefa de Entrega ou Pacote de Trabalho (PMI, 2008). Nele, todas as atividades do projeto são tratadas como Tarefa, e essa, por sua vez, pode representar também uma Entrega ou Pacote de Trabalho.

A síntese dos requisitos funcionais e não funcionais é apresentada nos Quadros 7 e 8.

Quadro 7. Requisitos funcionais.

Código	Requisito
RF001	Possuir um módulo para planejamento de projetos inteiramente visual.
RF002	Possuir um módulo para acompanhamento de projetos de maneira visual.
RF003	Possibilitar registro de plano do projeto baseado em iterações e tarefas.
RF004	Permitir a associação de usuários às tarefas.
RF005	O módulo de acompanhamento de projetos ou tarefas deve permitir que o usuário edite as informações das tarefas, bem como o seu andamento.
RF006	Apresentar o status das tarefas do projeto com indicadores.
RF007	Registrar novos requisitos e tarefas durante o acompanhamento de projetos.
RF008	Possibilitar o uso de dados dos sistemas de gerenciamento de projetos (SGP) tradicionais.
RF009	Permitir o acesso ao histórico do projeto, depois de finalizado.
RF010	Quando não utilizado deve exibir notícias da organização.

Quadro 8. Requisitos não funcionais.

Código	Requisito
RNF001	Interface homem-computador simplificada e resumida por meio do uso intensivo de gráficos, podendo simular a interface dos cartões dos painéis físicos para planejamento e acompanhamento de projetos;
RNF002	A interface deve ser simples, com poucos elementos se comparado aos softwares de GP tradicionais.
RNF003	A interface é de fácil compreensão. Significa que é legível porque os elementos são rapidamente identificados e autoexplicativa porque não necessita de manual ou explicação. Segue padrões de acessibilidade.
RNF004	Permitir a sensação de controle tal qual o quadro físico. De forma a garantir a distribuição das informações entre os colaboradores com a mensagem de confiança e controle sobre a informação pela equipe de projeto.
RNF005	Descentralização dos dados, permitindo que os usuários possam exportar os dados e trabalhar nas ferramentas de gerenciamento de projetos que julgarem mais apropriadas;
RNF006	É robusta e confiável. Significa que não há erros na operação e a velocidade das ações do software é compatível com a velocidade de resposta e ações humanas, nas diferentes faixas etárias.
RNF007	Permite a personalização de parâmetros como velocidade e quantidade de informações para adequar-se às diferenças entre os usuários humanos.
RNF008	Flexibilidade de alteração das tarefas para atender às mudanças no projeto.
RNF009	Exigir o menor tempo possível durante a execução das atividades rotineiras.

5.2. CONTEXTO DO SISTEMA

5.2.1. Diagrama de contexto

O contexto do sistema completo é apresentado na Figura 12. Nela, indica-se a relação entre o sistema que é criado e outros elementos de fronteira. No caso, são dois tipos. Hardwares de interação humano computador (Mesa digital, Totem eletrônico e painéis eletrônicos touchscreen) e softwares específicos para o gerenciamento de projetos, que podem ser do tipo: gerais (como o Google Calendar), do tipo desktop (como o MS Project, Tutos e outros) e sistemas corporativos do tipo *Enterprise Project Management* – EPM (como MS Project Server, Primavera, Oracle Project etc..). As integrações com estes sistemas podem ser feitas utilizando as várias tecnologias de integração conhecidas, desde as mais tradicionais como integração via base de dados, até as mais recentes via API's na web, com XML.

Figura 12. Contexto do sistema para gerenciamento ágil de projetos distribuídos

Catálogo de elementos e propriedades

O sistema para gerenciamento ágil de projetos distribuídos é formado por dois elementos principais: (A) Pro Visual Control e (B) SIPRO.

Os objetivos do Pro Visual Control que contém as funcionalidades para armazenamento e distribuição da informação e o SIPRO abstrai as funções de conexão das informações com outros sistemas, podem ser visualizados no Quadro 9.

Quadro 9. Objetivos do PVC e do SIPRO.

Sistema para Gerenciamento de Projetos Ágeis Distribuídos	PVC	Contém as regras para acesso e distribuição das informações para o painel e para a aquisição e sincronização com outros softwares de PM
	SIPRO	Abstrai do PVC as funções de conexão com os softwares e disponibiliza os dados para os sistemas autorizados à atualizá-los.

5.2.2. Atores do Sistema

O escopo do sistema é uma definição importante e o primeiro passo é descrever quem serão os atores. Para Booch et al (2005), um ator representa um papel que um ser humano, um dispositivo de hardware ou até outro sistema desempenha com o sistema.

Os atores do projeto em questão são os usuários dos softwares. Esses atores foram definidos tomando como base um ambiente padrão de desenvolvimento de projetos. Dessa maneira, no Quadro 10, foram definidos três atores.

Quadro 10. Usuários do sistema

Atores		Papel
Chefe de gerentes de projetos		➤ É o gerente do <i>Project Management Office</i> (PMO). É o responsável por acessar as informações do projeto no SGP base e quando necessário, criar um novo projeto.
Gerente de Projeto	Usuário	➤ Usuário responsável pelo projeto.
Membro da equipe de projeto		➤ Usuário diretamente envolvido com o desenvolvimento do projeto.

Nota-se que o Gerente de Projeto e o Membro da equipe estão abstraídos em um mesmo usuário. Isso se deve ao fato que para o software esses usuários tem as mesmas permissões.

5.3. CASOS DE USO E DIAGRAMA DE CASOS DE USO

Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado observável do valor de um ator. Os casos de uso podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento será implementado (BOOCH et al, 2005). Os casos de uso para o PVC são descritos em detalhes nos quadros abaixo.

Quadro 11. Caso de uso 1.

[Caso 1]	Cadastrar projeto
Breve Descrição	O chefe de gerentes de projetos será o responsável por acessar as informações do projeto no sistema de GP base da empresa e cadastrar um novo projeto.
Atores	Chefe de Gerentes de Projetos
Pré-condições	Deve ter acesso ao SGP principal
Pós-condições	O projeto criado no banco de dados do SGP principal

Quadro 12. Caso de uso 2.

[Caso 2]	Configurar projeto
Breve Descrição	Quando um projeto selecionado ainda não foi configurado para ser utilizado no painel, o sistema exibe uma janela de configuração para que as iterações sejam criadas.
Atores	Gerente de Projeto e Membro do Projeto
Pré-condições	O projeto deve ter sido criado no SGP principal.
Pós-condições	Projeto configurado no software e as iterações criadas

Quadro 13. Caso de uso 3.

[Caso 3]	Planejar projeto
Breve Descrição	O ambiente de planejamento e controle é carregado e novas tarefas podem ser adicionadas diretamente nas iterações ou na lista de tarefas.
Atores	Gerente de Projeto e Membro do Projeto

Quadro 14. Caso de uso 4.

[Caso 4]	Cadastrar tarefas
Breve Descrição	Ao arrastar uma tarefa para uma iteração ou para a lista de tarefas, é exibida a janela de cadastro de tarefa.
Atores	Gerente de Projeto e Membro do Projeto
Pós-condições	O sistema salva a nova tarefa no SGP principal

Quadro 15. Caso de uso 5.

[Caso 5]	Excluir tarefa
Breve Descrição	Ao arrastar uma tarefa para o ícone de lixeira, o sistema exclui a tarefa.
Atores	Gerente de Projeto e Membro do Projeto
Pós-condições	A tarefa é excluída do SGP principal

Quadro 16. Caso de uso 6.

[Caso 6]	Atualizar tarefa
Breve Descrição	Ao clicar sobre a tarefa, o sistema exibe a tela de edição e atualização do status da tarefa.
Atores	Gerente de Projeto e Membro do Projeto
Pós-condições	Os dados alterados são sincronizados com o SGP principal

Quadro 17. Caso de uso 7.

[Caso 7]	Visualizar e editar dados da tarefa
Breve Descrição	Ao clicar sobre a tarefa, o sistema exibe a tela de edição e atualização da tarefa, no qual o usuário pode visualizar os dados e editar a tarefa.
Atores	Gerente de Projeto e Membro do Projeto
Pós-condições	Os dados alterados são sincronizados com o SGP principal

Quadro 18. Caso de uso 8.

[Caso 8]	Sincronizar com o SGP principal
Breve Descrição	O SIPRO mantém-se sincronizado ao SGP principal.
Atores	SIPRO

A Figura 13 apresenta o diagrama de casos de uso. Nela, é possível identificar o relacionamento entre o conjunto de casos de uso e atores anteriormente definidos. Dessa maneira, a relação entre os atores e suas possíveis ações no sistema, além de ações que estão vinculadas a outras, podem ser identificadas. Seu objetivo é facilitar a compreensão da dependência das funcionalidades do sistema e as permissões necessárias envolvidas para sua execução.

Figura 13. Diagrama de Casos de Uso.

5.4. CENÁRIO DA SOLUÇÃO

Para este trabalho, foram criados dois diagramas que representam os cenários da solução: um para o PVC e outro para o SIPRO. Neles é possível identificar o fluxo das informações entre os atores e esses sistemas.

Os diagramas apresentam de maneira resumida o comportamento de PVC e SIPRO durante o planejamento e edição de dados do projeto.

O primeiro diagrama, apresentado na Figura 14, representa as sequências de interação entre PVC, SIPRO e demais elementos da arquitetura, durante o Planejamento de Entregas do Projeto. Nele são descrito os atores do software (Chefe de gerende de projetos e usuário) e os sistemas envolvidos (Software de GP, PVC e SIPRO). Cada chamada do diagrama representa um caso de uso envolvendo esses elementos. Nota-se também que todas as funções do PVC dependem do SIPRO, e este, por sua vez, é o responsável pela comunicação com o software de GP.

Já a Figura 15 representa a dinâmica de interação entre outros softwares de Gerenciamento de Projetos e SIPRO para a atualização de dados. Nela é possível identificar como ocorre a troca de dados entre o software de GP e o SIPRO. Essa sincronização ocorre partindo do SIPRO que busca ou envia as informações necessárias para o SGP sempre que necessário. Para que os dados se mantenham atualizados, o SIPRO possui uma *thread*¹⁵ responsável por acessar o SGP de tempos em tempos, definido de acordo com a necessidade do ambiente de utilização.

Em conjunto, os diagramas demonstram como a sincronização dos dados é garantida. Os dados de projeto podem, portanto, ser alterados pelos planejados, direto dos softwares de

¹⁵ Uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

Gerenciamento de Projetos (Figura 15) ou pelos membros da equipe de projeto utilizando os painéis digitais.

Figura 14. Cenário de Planejamento do Projeto.

Figura 15. Cenário de Sincronização de Dados com o SIPRO.

6. AMBIENTE E INFRA-ESTRUTURA

Nesta seção são descritas as tecnologias necessárias para a implantação da arquitetura descrita, a estrutura de hardware e o diagrama de camadas dos softwares envolvidos. Dessa forma, o objetivo dessa seção é apresentar a estrutura de hardware e software na qual o sistema será executado, conforme o propósito da visão física apresentada na revisão bibliográfica.

6.1. TECNOLOGIAS NECESSÁRIAS

Os dados são armazenados no SIPRO utilizando uma base de dados idêntica a do projeto dotProject. A adoção do dotProject evitou retrabalhos desnecessários no desenvolvimento das funcionalidades tradicionais para *desktop*. A sua escolha deve-se ao fato de ser um software *Open Source* para o ambiente web e possuir uma base de dados relacional em MySQL.

As características e as tecnologias escolhidas para a solução são apresentadas a seguir.

- **Pro Visual Control (PVC).**
 - É um software para ambiente web;
 - Sua fonte de dados é um *Web Services*; portanto, a tecnologia escolhida deve facilitar o desenvolvimento de clientes para o seu acesso; e
 - A interface é acessada por meio de uma tela *touchscreen*, sendo assim, a tecnologia de desenvolvimento foi o Adobe Flex.
- **SIPRO**
 - É um software para ambiente web;
 - Utiliza a base de dados do software livre dotProject, mas sem se restringir a apenas esse software; e

- É um servidor *Web Service*; portanto, a tecnologia escolhida foi o PHP com a biblioteca NuSOAP específica para desenvolvimento de *Web Services*;

6.2. VISÃO DOS MÓDULOS

O software foi desenvolvido dividindo suas funcionalidades em módulos. Esses módulos em conjunto são responsáveis para que o software atenda seus objetivos, sendo eles:

- **Módulo de seleção de projeto.** Este módulo é o ponto de partida do sistema, exibindo de forma amigável os projetos em execução para que o Gerente ou Membro do Projeto escolha qual projeto deseja trabalhar.
- **Módulo de configuração de projeto.** Quando um projeto selecionado ainda não foi configurado para o PVC, este módulo é o responsável por coletar as informações adicionais que são necessárias para o seu funcionamento.
- **Módulo de planejamento e controle de projeto.** Após selecionar o projeto, o sistema carrega este módulo que proporciona um ambiente interativo e de acesso intuitivo para o planejamento e controle do projeto. Essa união na interface de planejamento e de controle visa facilitar a utilização do sistema, não sendo necessário a navegação por diferentes janelas para a execução de diferentes ações.
- **Módulo de nova tarefa.**
- **Módulo de excluir tarefa.**
- **Módulo de edição de tarefa.**
- **Módulo de alocação de recursos.** Módulo responsável por exibir os recursos disponíveis e alocá-los a uma tarefa. Este módulo é utilizado juntamente com os módulos de nova tarefa e edição de tarefa.

- **Módulo de notícias.** Quando o sistema não estiver sendo utilizado, funcionará como um painel de informações.

Com base nos requisitos gerados, nos módulos propostos e nos objetivos do PVC, o Quadro 19 foi concebido contextualizando a relação desses elementos.

Quadro 19. Relação entre requisitos, módulos e objetivos do software

Requisitos	Módulos	Objetivos
RF008: Possibilitar o uso de dados dos sistemas de gerenciamento de projetos (SGP) tradicionais	Módulo de seleção de projeto	- (i) Acessar as informações de projetos de outros softwares de GP e apresentá-las em uma nova interface valorizando a gestão visual
RF003: Possibilitar registro de plano do projeto baseado em iterações e tarefas	Módulo de configuração de projeto	
RF001: Possuir um módulo para planejamento de projetos inteiramente visual	Módulo de planejamento e controle de projeto	
RF002: Possuir um módulo para acompanhamento de projetos de maneira visual		
RF006: Apresentar o status das tarefas do projeto com indicadores		
RF009: Permitir o acesso ao histórico do projeto, depois de finalizado		
RF005: O módulo de acompanhamento de projetos ou tarefas deve permitir que o usuário edite as informações das tarefas, bem como o seu andamento	Módulo de edição de tarefa	- (ii) Aumentar o envolvimento da equipe na gestão de projetos simplificando as atividades de planejamento e controle
RF004: Permitir a associação de usuários às tarefas	Módulo de alocação de recursos	
RF007: Registrar novos requisitos e tarefas durante o acompanhamento de projetos	Módulo de nova tarefa	- (iii) Ampliar a disciplina e auto-organização tornando a equipe proativa na detecção de problemas e re-planejamento
	Módulo de excluir tarefa	

Dessa maneira, verifica-se então que os módulos propostos englobam todos os requisitos funcionais e os objetivos do PVC. Já os requisitos não funcionais não necessitam aparecer no quadro, pois se aplicam a todos os módulos.

Nota-se também que o módulo de notícias não foi incluído no quadro. Isso se deve ao fato de esse requisito ter sido identificado posteriormente ao levantamento dos requisitos baseados na literatura de painéis físicos.

Para contextualizar a relação entre os módulos, a Figura 16 representa a visão dos módulos do software desenvolvido. Nela identificam-se os sistemas envolvidos e os módulos correspondentes a cada parte dele.

Figura 16. Visão de Módulos de Software.

Na figura, os módulos de edição de tarefa, alocação de recursos, nova tarefa e exclusão de tarefa estão inseridos no módulo de planejamento e controle, e o de notícias em infraestrutura.

6.3. DIAGRAMA DE IMPLANTAÇÃO

Os diagramas de implantação são utilizados para a modelagem dos aspectos físicos de um sistema. Ele mostra a configuração dos nós de processamento em tempo de execução e os artefatos que nele existem. Esses diagramas são empregados para a modelagem da visão estática da implantação de um sistema. Na maior parte, isso envolve a modelagem da topologia do hardware em que o sistema é executado (BOOCH et al, 2005).

Para o projeto em questão o diagrama de implantação visa fornecer uma visão do comportamento dos elementos de software e hardware envolvidos na aplicação. Ele irá ajudar na compreensão da comunicação e na relação existente entre os artefatos envolvidos no sistema.

Um componente é uma parte encapsulada, reutilizável e substituível do software. São como blocos de construção que combinados constituem o software em si. Por este motivo, os componentes podem variar de tamanho; desde pequenos como uma classe ou arquivo do sistema, até um subsistema ou sistema de grande porte (BOOCH et al, 2005).

Já os nós são elementos físicos que existem em tempo de execução e representam um recurso computacional. Um nó tipicamente representa um processador ou um dispositivo em que os componentes poderão ser instalados (BOOCH et al, 2005).

O Diagrama de Implantação gerado no projeto pode ser visualizado na Figura 17. Nela, o software Pro Visual Control está inserido no nó Servidor de Conteúdo por se tratar de um software para ambiente web, esse servidor será responsável por armazenar o código fonte dessa aplicação. Portanto, o artefato Index.mxml visualizado no nó Estação touchscreen, representa uma instância dessa aplicação.

Dentre as informações que o diagrama oferece, a mais importante é a compreensão de como será a comunicação da Estação *touchscreen* com os dados do banco de dados do dotProject. Como citado anteriormente, ela se realiza por meio de um *Web Service* (SIPRO)

localizado no Servidor de Aplicação, que envia e recebe dados da aplicação principal (PVC) utilizando o protocolo SOAP (padrão para comunicação com *Web Services*).

[online diagramming & design] [createit.com](https://www.createit.com)

Figura 17. Diagrama de Implantação da Arquitetura de Software de Domínio Específico

6.4. DIAGRAMA DE CAMADAS

Foi criado também o diagrama de camadas para melhor entendimento do fluxo da informação interno dos sistemas. A Figura 18 mostra o fluxo geral entre os módulos

principais do sistema e o SIPRO. Nota-se que a ordem com que os módulos aparecem indica como estão relacionados e quando são utilizados. Além disso, é possível identificar também o Cliente e o Servidor *Web Service* que se comunicam através do protocolo SOAP.

No SIPRO, por sua vez, é possível identificar os conectores, que são os responsáveis por acessar os outros SGPs, além de sua base local necessária para o armazenamento dos parâmetros do sistema e as iterações, quando não disponível do SGP principal.

Figura 18. Visão de Camadas (Geral).

Já na Figura 19, foi detalhado o Módulo de Planejamento e Controle de Projetos. Nele, é possível identificar o Módulo de Nova Tarefa, Módulo de Edição de Tarefa e o Módulo de Alocação de Recursos. Estes estão diretamente relacionados e trabalham em conjunto, utilizando recursos em comum do sistema.

Figura 19. Visão de Camadas (Módulo de Planejamento e Controle).

7. AVALIAÇÃO DA ARQUITETURA POR MEIO DE PROTÓTIPO

O desenvolvimento do protótipo a ser apresentado nesse capítulo tem como objetivo permitir a avaliação da arquitetura proposta. A meta foi desenvolver uma solução simples o suficiente para permitir o seu desenvolvimento dentro do prazo do projeto. Dessa maneira, foi desenvolvido apenas o conector para o SIPRO que se comunica com o SGP dotProject, mas possibilitando que futuramente sejam desenvolvidos novos conectores.

Dessa maneira, o capítulo inicia-se apresentando o desenvolvimento descrevendo as tecnologias adotadas (Seção 7.1), seguido do desenvolvimento da interface e módulos (Seção 7.2), desenvolvimento do SIPRO (Seção 7.3) e, por fim, o desenvolvimento do PVC (Seção 7.4).

7.1. TECNOLOGIAS ADOTADAS

A escolha das tecnologias empregadas no desenvolvimento do PVC e do SIPRO foi realizada conforme as principais características de cada um desses elementos da solução completa.

O desenvolvimento do PVC depende de uma interface amigável, portátil em diferentes dispositivos e rica (no sentido de uso de gráficos e diferentes tipos de interação com o usuário). O SIPRO depende do desenvolvimento de uma camada que permita integração com diferentes sistemas em um ambiente web.

As características fundamentais das aplicações são: (i) a interface rica em componentes visuais (PVC) e (ii) implementação de cliente e servidor *Web Service* (PVC e SIPRO). Portanto, a análise será dividida entre esses dois paradigmas.

Sobre o primeiro paradigma, devem-se explorar tecnologias que estão ligadas ao *frontend* da aplicação para obter-se uma interface rica em componentes visuais de software. Linguagens como PHP ou Java utilizam como *frontend* HTML e JavaScript. Essas

tecnologias quanto utilizadas em conjunto são chamadas de AJAX, e são amplamente utilizadas para o desenvolvimento de interfaces interativas. Outra tecnologia para *frontend* de aplicações é o Adobe Flex. Desenvolvido para a criação de aplicações de internet do tipo RIA (*Rich Internet Application*), possui uma coleção de ferramentas e componentes que facilitam o desenvolvimento de interfaces gráficas. Portanto, são tecnologias como essas que devem ser comparadas para o projeto em questão.

- **HTML + JavaScript (AJAX).** O JavaScript é uma linguagem interpretada que pode ser usada junto com o HTML. Ela permite inserir vários efeitos e possibilita a criação de sites com elementos mais dinâmicos. Apesar de ser uma linguagem complexa, atualmente existem diversos *frameworks* que facilitam a criação de códigos JavaScript (W3SCHOOLS, 1999).
- **Adobe Flex.** O Flex é uma estrutura de código aberto para a criação de aplicativos Web expressivos e altamente interativos. Oferece um modelo moderno de linguagem e programação baseado em padrões que suportam modelos comuns de design. Seus principais componentes são o MXML, uma linguagem declarativa baseada em XML, que é usada para descrever comportamentos e layout de interface de usuário, e o ActionScript 3.0, uma linguagem de programação orientada por objetos, que é usada para criar a lógica de cliente (ADOBE, 2009).

Franco (2008) comparou em seu estudo diversos aspectos dessas duas tecnologias. Foram extraídas dessa tabela as informações que são mais relevantes para este projeto, conforme pode ser verificado no Quadro 20.

O trabalho de Franco(2008) indica, portanto, que o Adobe Flex é a mais apropriada para o tipo de software desenvolvido. Sendo, dessa maneira, uma ferramenta robusta para o

desenvolvimento de componentes gráficos e ainda possuir suporte nativo a comunicação por meio de *Web Services*.

Quadro 20. Comparação entre as tecnologias AJAX e Adobe Flex

	Importância	Ajax	Flex
Animações	O suporte a animações possibilita que os desenvolvedores destaquem ou orientem o usuário em uma mudança de estado da aplicação ou para simplesmente entreter e envolver mais o usuário.	Suporte limitado	Amplamente suportado
Gráficos Vetoriais	Os gráficos vetoriais fornecem aos desenvolvedores a habilidade de criar componentes de visualização de dados complexos que podem ser atualizados dinamicamente, animados e redesenhados sem a necessidade de novas requisições ao servidor.	Suportado por meio de VML nos mais novos navegadores e suportado de forma limitada com SVG em alguns outros navegadores	Suporte nativo
Comunicação com servidores	Habilidade de conectar sua aplicação a serviços remotos.	Ambos possibilitam comunicação via Web services, REST ou XML simples.	
Extensibilidade	Permite a criação de ferramentas, plugins, componentes, etc; por terceiros.	Como o AJAX é uma extensão do HTML e do JavaScript, ele herda essa propriedade dessas outras tecnologias	Os componentes do Flex podem facilmente ser criados, e entendidos quando já existentes.

Fonte: Adaptado de Franco (2008)

Já sobre o segundo paradigma, foi escolhida a linguagem PHP para a implementação dos conectores de acesso as fontes de dados de outros sistemas de GP. Juntamente com a biblioteca NuSOAP (para o desenvolvimento do servidor *Web Service*), essas tecnologias apresentavam-se adequadas para subsistema SIPRO, além de serem de domínio do pesquisador.

Para o trabalho em questão, foram desenvolvidos dois conectores:

- **Acesso ao DotProject via base de dados.** Foi desenvolvido apenas um conector visando provar o conceito proposto. Dessa forma, foi escolhido o dotProject como SGP principal. Ele é um software de gerenciamento de projetos livre escrito em PHP de fácil utilização. Possui em sua plataforma um

conjunto básico, mas importante, de funcionalidades e características para a gestão de projetos. Por ser um software para o ambiente web, seu acesso é feito por meio de um navegador, tornando-se independente de sistema operacional e de qualquer instalação na máquina do usuário (DOTPROJECT, 2009).

- **Acesso ao Google docs utilizando a API do Google.** Foi realizado um trabalho publicado em Barata, Amaral, Araujo e Martins (2010), de integração entre o Sistema Proposto e o Google Calendar. Os resultados deste conector podem ser encontrados no artigo citado.

A Figura 20 apresenta o diagrama de implantação para o Protótipo.

Figura 20. Diagrama de Implantação do Protótipo

7.2. DESENVOLVIMENTO DA INTERFACE E MÓDULOS

Inicialmente, foi gerado o *mockup* da tela de planejamento e controle do projeto (Figura 21). Esse serviu como guia no desenvolvimento do PVC e do SIPRO. O protótipo

final possui algumas diferenças com relação a esse *mockup*, devido à identificação de melhorias ao longo do seu desenvolvimento.

É possível identificar quatro iterações, sendo a primeira a atual, seguida das três seguintes. Nota-se também as tarefas representadas por cartões autoadesivos, conforme proposto nos requisitos do projeto.

Figura 21. Mockup do PVC.

Com base nessas informações, iniciou-se o desenvolvimento dos módulos do PVC, em paralelo com o SIPRO. A cada nova necessidade de dados do dotProject, identificada durante o desenvolvimento do PVC, criou-se uma nova chamada no *Web Service*.

7.3. DESENVOLVIMENTO DO SIPRO

A Figura 22 mostra a tela gerada automaticamente pela biblioteca NuSOAP, contendo o resultado final do SIPRO. Nela, os métodos criados para o protótipo estão destacados em azul.

Figura 22. Métodos do SIPRO.

Em sua estrutura interna, quando é realizada uma chamada no SIPRO, acessa-se a classe em PHP responsável por: (i) conectar-se ao banco de dados do dotProject, (ii) coletar os dados necessários e (iii) retorná-los como um vetor. Em seguida, o SIPRO converte esses dados contidos no vetor em um XML, que por sua vez é transmitido para o PVC por meio do protocolo SOAP. Esta dinâmica pode ser verificada na Figura 18, na Seção 6.4 da arquitetura do sistema.

Na chamada `getProjects`, por exemplo, o SIPRO acessa a classe `projects.class.php`, que possui um método também chamado `getProjects` com o código SQL necessário para retornar os dados do projeto para o PVC. Nesse exemplo, os dados necessários são:

- ID do projeto;
- Nome do projeto;
- Apelido do projeto;
- Data de início do projeto;
- Data de término do projeto;
- Descrição do projeto; e
- Criador do projeto.

A descrição técnica, gerada automaticamente pelo NuSOAP, pode ser verificada na Figura 23.

```
Close
Name: getProjects
Binding: siproBinding
Endpoint: http://numa.sc.usp.br/dotproject/webservice/sipro.php
SoapAction: http://numa.sc.usp.br/dotproject/webservice/sipro.php/getProjects
Style: rpc
Input:
  use: encoded
  namespace: http://www.numa.sc.usp.br/dotproject/webservice
  encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
  message: getProjectsRequest
  parts:
Output:
  use: encoded
  namespace: http://www.numa.sc.usp.br/dotproject/webservice
  encodingStyle: http://schemas.xmlsoap.org/soap/encoding/
  message: getProjectsResponse
  parts:
    return: tns:Projects
Namespace: http://www.numa.sc.usp.br/dotproject/webservice
Transport: http://schemas.xmlsoap.org/soap/http
Documentation:
```

Figura 23. Descrição técnica da chama getProjects do SIPRO.

A Figura 24 representa o código gerado para esse exemplo. Nota-se também a criação do vetor de projetos com a função `array_push` do PHP a cada linha retornada na consulta SQL.

```

/**
 * Retorna os projetos do dotProject
 * @return projects[] Array de projetos
 */
public function getProjects() {

    $sql = "SELECT project_id, project_name, project_short_name,
project_start_date, project_end_date, project_description, project_creator
FROM projects WHERE project_status = '3' ORDER BY project_name";
    $result = mysql_query($sql, $this->conn);

    $projects = array();
    while ($project = mysql_fetch_row($result)) {
        $tempArray = array(
            'project_id' => $project[0],
            'project_name' => $project[1],
            'project_short_name' => $project[2],
            'project_start_date' => $project[3],
            'project_end_date' => $project[4],
            'project_description' => $project[5],
            'project_creator' => $project[6]
        );
        array_push($projects, $tempArray);
    }

    return $projects;
}

```

Figura 24. Código para o método getProjects do SIPRO.

No caso de outros sistemas de GP é necessário investigar a possibilidade de acesso direto ao banco de dados, conforme como foi implementado para este conector. Além desta integração já foi testada a integração via API do Google Calendar no trabalho de Barata, Amaral, Araujo e Martins (2010).

Com isso, comprovou-se a possibilidade de dois tipos de integração, dentre os quatro descritos na proposta da arquitetura.

Essas duas integrações podem ser consideradas suficientes como prova de conceito, demonstrando a viabilidade técnica da arquitetura. Entende-se que, funcionando nestes dois casos extremos, uma estratégia de integração mais tradicional (via base de dados) e outra mais atual (Via API do Google), é de se esperar que as demais integrações sugeridas sejam igualmente possíveis.

7.4. DESENVOLVIMENTO DO PVC

Já no PVC foram criados vinte e dois componentes do Adobe Flex e um pacote. Estes componentes, divididos em conjuntos de componentes, estão relacionados pela função que desempenham no sistema. O Quadro 21 apresenta esses componentes e o pacote, relacionando em qual módulo da arquitetura proposta está inserido, o conjunto a que pertence e a sua funcionalidade no protótipo.

Quadro 21. Relação entre os módulos da arquitetura e os componentes do PVC.

Módulos¹⁶	Conjunto de componentes	Função	Componente
Infraestrutura	Aplicação Principal	Módulo principal. Responsável por coordenar o fluxo do sistema.	Index.mxml
	Notícias	Módulo responsável pela exibição das notícias, quando o sistema não estiver sendo utilizado.	TwitterNoticia.mxml
Cliente Web Service	Acesso ao SIPRO	Pacote de componentes de acesso ao SIPRO.	br.usp.sipro
Seleção de Projeto	Lista de Projetos	Módulos responsáveis por exibir os projetos em andamento, e possibilitar a escolha do que se deseja visualizar.	ProjetosItem.mxml ProjetosLista.mxml
Configuração de Projeto	Configuração do Projeto	Módulo responsável pela configuração do projeto, ou seja, exibe suas informações (data de início e término) e as opções para criação de iterações (semanais, quinzenais, mensais). Após a escolha do usuário, cria automaticamente as iterações.	ProjetoConfigForm.mxml

¹⁶ O agrupamento de módulos seguiu a especificação da arquitetura, exibidos na Figura 16.

Quadro 21. Relação entre os módulos da arquitetura e os componentes do PVC. (continuação)

Planejamento e controle	Iteração	Módulos responsáveis pela exibição das iterações, e todas as ações relacionadas a elas, como por exemplo, o <i>drag-and-drop</i> de tarefas entre as iterações.	IteracaoItens.mxml IteracaoJanela.mxml IteracaoJanelinha.mxml IteracoesPlanejadas.mxml
	Recursos	Módulos para alocação de recursos das tarefas.	RecursosAlocados.mxml RecursosListaItem.mxml RecursosSelecionadosItem.mxml RecursosTarefaMiniatura.mxml
	Criação e edição de Tarefas	Módulos para criação, edição e exclusão de novas tarefas. Responsáveis também pela criação de log de tarefas.	NovaTarefaItem.mxml TarefaEditar.mxml TarefaExcluir.mxml TarefaLogForm.mxml TarefaLogItem.mxml TarefaNova.mxml TarefaNovaForm.mxml
	Lista de Tarefas	Módulos responsáveis pela exibição da lista de tarefas.	TarefasLista.mxml TarefasItens.mxml

O pacote (*package*) chamado `br.usp.sipro` é um conjunto de ActionScripts que formam o cliente do servidor *Web Service* (SIPRO). Esse pacote é gerado automaticamente pelo Adobe Flash Builder baseado na WSDL do servidor. Dessa maneira, torna-se simples o acesso ao SIPRO, pois além da conversão automática dos dados em objetos do sistema, não é necessário codificar o tratamento dos XMLs.

A comunicação entre os componentes listados, funções do sistema e do cliente *Web Service* ocorre por meio de eventos. No Flex, a cada ação do usuário são disparados uma série de eventos, como por exemplo, o clique em um botão ou o arrastar do mouse. Cabe ao desenvolvedor dizer ao sistema qual ação deseja que seja tomada quando esses eventos ocorrem.

Um exemplo é a tela de seleção de projetos. O componente responsável pela sua exibição é o `ProjetosLista.mxml`. Quando esse é carregado no sistema, dispara-se um evento chamado `creationComplete`, que faz uma chamada ao `getProjects` do SIPRO (descrito

anteriormente). Esse, por sua vez, retorna um vetor de objetos do tipo Project na variável sipro.getProjects_lastresult. Esse vetor, inserido em um componente do Flex chamado TileList, é o responsável por exibir a lista dos projetos.

A Figura 25 contém a parte do código da função descrita acima. Nela, encontra-se destacado em amarelo o vetor de projetos sipro.getProjects_lastresult.

```
<mx:Script>
    <![CDATA[
        [Bindable] public var sipro:Sipro = new Sipro();    // Variável de acesso ao Webservice Sipro

        /* inicio das funcoes de acesso ao webservice */
        private function getProjetos():void
        {
            // Attach the event fault handler
            sipro.addSiproFaultEventListener(siproFault);
            sipro.addgetProjectsEventListener(kinetic);
            // Invoke the service call
            sipro.getProjects();
        }

        public function siproFault(event:FaultEvent):void
        {
            Alert.show(event.fault.faultString);
        }
        /* fim das funcoes de acesso ao webservice */
    ]]>
</mx:Script>

<mx:Panel id="projetosPanel" fontSize="16" title="Selecione o projeto" width="100%" height="100%">
    <mx:Button width="100%" label="p" fontFamily="WINGDNG3" fontWeight="normal" fontSize="20"
        buttonDown="scrollUp()" autoRepeat="true"/>
    <mx:TileList xmlns:mx="http://www.adobe.com/2006/mxml"
        id="projetosLista" rowHeight="60"
        width="100%" height="100%" verticalScrollPolicy="off" horizontalScrollPolicy="off"
        dataProvider="{sipro.getProjects_lastResult}"
        creationComplete="getProjetos()"
        itemRenderer="ProjetosItem" borderStyle="none"
        columnCount="1"
        selectable="true"
        alternatingItemColors=["#FFFFFF", #B7B7B7]">
    </mx:TileList>
    <mx:Button width="100%" label="q" fontFamily="WINGDNG3" fontWeight="normal" fontSize="20"
        buttonDown="scrollDown()" autoRepeat="true"/>
</mx:Panel>
```

Figura 25. Código do módulo ProjetosLista.mxml

A Figura 27 apresenta o resultado final do código acima. Nela, o componente de seleção de projetos do PVC está destacado em tracejado. Além disso, nota-se o painel de notícias que, apesar de aparecer na mesma tela, é gerado pelo componente TwitterNoticia.mxml.

Figura 26. Tela principal do PVC com o painel de seleção de projetos em destaque.

A sincronização com o dotProject é apresentada na Figura 27. Nela, a tela de projetos do dotProject é exibida com os mesmos projetos da tela principal do PVC. Tais dados são provenientes dos projetos do grupo de pesquisa, conforme descrito na etapa 6 da Seção 3.3.

Figura 27. Tela de projetos do dotProject.

Após selecionar o projeto desejado, caso esse não tenha sido configurado no sistema, o PVC exibe a tela de configuração (Figura 28).

Figura 28. Tela de configuração do projeto.

Nessa tela, o usuário visualiza os dados do projeto, sendo eles: nome do projeto, data de início e data final. Em seguida, o módulo exibe as opções dos tipos de duração das iterações possíveis (semanais, quinzenais ou mensais) para que o usuário possa escolher de acordo com as necessidades do seu projeto. Uma vez salva a configuração, o sistema gera as iterações automaticamente, que são exibidas na Figura 29.

Figura 29. Janela de Planejamento e Controle do PVC.

Esta figura representa a tela de planejamento e controle do PVC. É nela que são executados a maioria dos componentes desenvolvidos. Em sua parte superior, tem-se o nome do projeto, a duração da iteração e o número total de iterações. Logo abaixo estão as duas iterações principais (atual e próxima), seguidas de uma lista de navegação com as demais iterações. Por fim, tem-se a lista de tarefas que é onde se encontram as tarefas planejadas.

É importante ressaltar que qualquer tarefa localizada em um dos locais descritos podem ser movidas (*drag-and-drop*) entre as demais iterações ou lista de tarefas. Assim como para criar uma nova tarefa, basta arrastar o ícone de recado autoadesivo em amarelo para um desses locais. Após arrastar o ícone, a janela representada na Figura 30 é aberta para ser preenchida pelo usuário.

Nova Tarefa

Nome da Tarefa: Tarefa teste

Data de Inicio: 27/08/2011 Data Final: 27/09/2011

Esforço: 32 Dias (Horas de Trabalho Diário: 8h)

Descrição da Tarefa: teste

Recursos Alocados

Guilherme Martins [100%]

0% 25% 50% 75% 100%

Fábio Gaspar de Carvalho

Salvar

Figura 30. Janela de criação de nova tarefa do PVC.

Nesta janela são solicitados os dados necessários para inserir a tarefa no dotProject. Caso ela tenha sido arrastada para uma iteração, as datas de início e fim são preenchidas automaticamente. Além disso, o usuário deve alocar recursos para a tarefa. Mais uma vez basta selecionar o recurso na lista de recursos e arrastá-lo para a lista da tarefa. A barra de porcentagem ao lado do nome do recurso corresponde a dedicação dele para a tarefa.

No caso de o usuário tocar (correspondente a um clique de mouse) uma tarefa já criada, a janela de edição e atualização de acompanhamento é aberta. A Figura 31 corresponde a essa janela.

Figura 31. Janela de edição de tarefa do PVC.

Nela, pode-se editar todos os dados da criação, além de alterar os recursos alocados. Seu diferencial é o painel localizado a direita que permite a criação de *logs* da tarefa. Nele, o usuário pode registrar as horas trabalhadas e alterar o seu status (porcentagem), seguido de um comentário para o gerente de projetos.

Nas duas iterações principais, as tarefas são representadas por recados autoadesivos de três cores diferentes: amarelo quando planejada, azul quando em andamento e verde quando concluída. Além disso, nota-se o nome da tarefa, o seu status (porcentagem) e as fotos dos recursos alocados, conforme ilustrado na Figura 32.

Figura 32. Cartões autoadesivos utilizados no PVC.

Por fim, o usuário ainda pode excluir uma tarefa. Para isto, basta arrastá-la até a lixeira localizada na parte superior da Figura 29.

Em paralelo ao desenvolvimento do protótipo, ocorreu também a montagem do totem onde o PVC deverá rodar. Este totem possui uma tela de 32 polegadas *touchscreen* da marca ELO, e um microcomputador Dell Studio Hybrid, acoplado em sua parte traseira. A Figura 33 apresenta o totem.

Figura 33. Totem utilizado na pesquisa.

8. CONCLUSÕES

A pesquisa iniciou com uma descrição do estado da arte dos softwares de gerenciamento de projetos. A revisão indica uma mudança importante na área de gerenciamento de projetos, o surgimento da abordagem do Gerenciamento Ágil de Projetos.

Essa abordagem propõe a simplificação dos métodos de gerenciamento de projetos por meio de um conjunto de práticas. Uma delas é o conceito de gestão à vista, em que os projetos são planejados pela equipe de projeto, de maneira iterativa e empregando quadros visuais. A segunda é o conceito de iteração, aonde os planos são criados de maneira parcial e atualizados em ciclos de iterações.

Foram apresentados vários benefícios do uso destes conceitos, mas, também dificuldades. O uso de painéis visuais físicos (com cartões autoadesivos, *post-its*) traz limitações para grandes projetos distribuídos. Além disso, os softwares de projeto existentes não apresentam funcionalidades adequadas para o conceito de iteração e planejamento iterativo. Portanto, a utilização plena depende de mudanças nos softwares de gerenciamento de projetos existentes, desenvolvidos segundo a abordagem tradicional de gerenciamento.

A revisão realizada apontou também oportunidades para a solução deste problema. Novas tecnologias como o conceito de Sinalização Digital, hardwares com novas funções de interação homem-computador e a teoria sobre arquitetura de software que facilita a integração e interoperabilidade de vários aplicativos. Optou-se então por investigar o uso dessas tecnologias para propor avanços nos softwares de gerenciamento de projetos, de forma a apoiar a gestão segundo os princípios do gerenciamento ágil.

A pesquisa envolveu a proposição da especificação de uma arquitetura de software de domínio específico, voltada especialmente para o gerenciamento ágil de projetos distribuídos, empregando-se as tecnologias de sinalização digital e painéis *touchscreen*. Para testar a viabilidade técnica da proposta, desenvolveu-se um protótipo utilizando as tecnologias Adobe

Flex, PHP e o SGP dotProject, desenvolvido segundo as especificações propostas e posteriormente utilizado em uma aplicação exemplo, de projeto do próprio grupo de pesquisa, para testar sua exequibilidade.

Os resultados demonstram que a arquitetura proposta é viável e pode trazer uma série de vantagens frente o uso de softwares tradicionais (Microsoft Project, Primavera, etc..) e os softwares voltados para o APM (VersionOne, JIRA, etc...).

As principais vantagens do protótipo desenvolvido, frente aos softwares de GP tradicionais são:

- Implementa o conceito de planejamento iterativo e com gestão à vista, propostos no gerenciamento ágil;
- Não demanda o retrabalho de desenvolver funcionalidades avançadas de gerenciamento de projetos como desenhos de caminho-crítico, análise do valor, relatórios e outras já desenvolvidas nos sistemas existentes; e
- Favorece a integração da equipe de projeto, tanto no planejamento como em seu controle, facilitando atividades rotineiras como a criação de novas tarefas ou atualização do seu andamento;

Já com relação aos softwares de gerenciamento ágil, suas principais vantagens são:

- Interface ainda mais simplificada, pois é desenvolvida especialmente para painéis digitais e não telas de computadores;
 - Permite que o planejador ou membro do escritório de projetos continue a ter a possibilidade de utilizar recursos avançados de gerenciamento de projetos, presentes nos softwares tradicionais e que ainda não estão disponíveis nas ferramentas de APM;
- e

- Oferece grande flexibilidade permitindo que os sistemas de informação gerados com o uso da arquitetura possam atender um espectro amplo de empresas e problemas de projeto, com escalabilidade e considerando equipes distribuídas.

A arquitetura proposta é um resultado importante pois representa um passo inicial para o desenvolvimento de uma nova classe de sistemas. Ela pode orientar o desenvolvimento de uma nova classe de soluções de aplicativos computacionais, reutilizando componentes (softwares, hardwares etc..) existentes, para que atuem em conjunto para apoiar o gerenciamento de projetos com equipes distribuídas e segundo a abordagem do Gerenciamento Ágil. Portanto, contribui para a teoria de Gerenciamento Ágil e para a teoria de Softwares de Apoio ao gerenciamento de projetos. A arquitetura pode servir como uma base inicial para o desenvolvimento de componentes inovadores que utilizem interfaces de IHC avançadas, como as “**interfaces naturais**”, para aprimorar a tarefa de planejamento e controle de projetos.

Dando continuidade a este trabalho são sugeridas as seguintes pesquisas futuras: estudo e desenvolvimento de uma interface gráfica melhorada para o protótipo, análise e desenvolvimento de novas funcionalidades, além do planejamento e controle de tarefas, para o gerenciamento ágil de projetos e estudo empírico com aplicação do protótipo em um ambiente de desenvolvimento de projetos.

9. REFERÊNCIAS

- ADOBE. Disponível em < <http://www.adobe.com> >. Acesso em maio, 2009.
- AGILE ALLIANCE. Manifesto for agile software development. 2001. Disponível em <<http://www.agilemanifesto.org/>>. Acesso em: abril, 2009.
- AMARAL, D. C.; CONFORTO, E. C.; BENASSI, J. L. G.; ARAUJO, C. D. Gerenciamento Ágil de Projetos: Aplicação em Produtos Inovadores. São Paulo: Editora Saraiva, 2011.
- AMESCUA, A. et al. Knowledge repository to improve agile development processes learning. Iet Software, v. 4, n. 6, p. 434-444, Dec 2010.
- ANGIONI, M. et al. Integrating XP project management in development environments. **Journal of Systems Architecture**, v.52, n.11, p. 619-626, 2006.
- ANDERSON, A.; BEATTIE, R.; BECK, K.; ET AL. Chrysler Goes to "Extremes." Distributed Computing, v. 1998, p. 24–28. 1998. Disponível em <<http://www.xprogramming.com/publications/dc9810cs.pdf>>. Acesso em: abril, 2009.
- ARAUJO, C. Softwares de apoio ao gerenciamento ágil de projetos colaborativos de novos produtos: análise teórica e identificação de requisitos. Dissertação (Mestrado em Engenharia de Produção). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.
- BARATA, A. M. ; AMARAL, D. C. ; ARAUJO, C. ; MARTINS, G. D. . Estratégias para utilização das tecnologias do Google Code no gerenciamento ágil de projetos. In: XXX ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 2010, São Carlos. Anais... 2010.
- BARBER, K. S.; HOLT, J.; BAKER, G. Performance evaluation of domain reference architectures. In: Proc. of the 14th International Conference on Software Engineering and Knowledge Engineering, ACM Press, p. 225–232, 2002.
- BECK, K. et al. Manifesto for agile software development. 2001. Disponível em <<http://www.agilemanifesto.org>>. Acesso em abril, 2009.
- BENASSI, J.L.G.; AMARAL, D. C. Gerenciamento ágil de projetos aplicado ao desenvolvimento de produto físico. In: SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO, 14., 2007. Anais... Bauru: UNESP, 2007.

- BERGER, C.; RUMPE, B. Supporting Agile Change Management by Scenario-Based Regression Simulation. **IEEE Transactions on Intelligent Transportation Systems**, v. 11, n. 2, p. 504-509, 2010.
- BOEHM, B. Get ready for agile methods, with care. **IEEE Computer Magazine**, January 2002, [S.l], p. 64-69, 2002.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário**. 2 ed., p.474. Campus, 2005.
- BURKE, RR. Behavioral Effects of Digital Signage. *Journal of advertising research*, v. 49, n. 2, jun., p. 180-185, 2009. doi:10.2501/S0021849909090254.
- CABRAL, A. Y. et al. A Case Study of Knowledge Management Usage in Agile Software Projects. In: FILIPE, J. C. J. (Ed.). *Enterprise Information Systems-Bk*, v.24, 2009.
- CHEN, Q. MALRIC, F. ZHANG, Y. et al. Interacting with digital signage using hand gestures. **Image Analysis and Recognition**, p. 347–358, 2009. Springer. Disponível em: <<http://www.springerlink.com/index/XVL315T180655746.pdf>>.
- CHIN, G. *Agile Project Management: how to succeed in the face of changing project requirements*. 229 p. NY: Amacon, 2004.
- CHUNG, J.; LEE, K. A framework of collaborative design environment for injection molding. **Computers in Industry**, vol. 47, n.3, p. 319-337, março, 2002.
- CHUNG, L.; LEITE, J. C. S. Non-functional requirements in the software development process. In: Glinz, M. *Conceptual Modeling: Foundations and Applications Lecture Notes in Computer Science*, V. 5600, p.363-379, 2009.
- CICMIL, S. et al. Rethinking Project Management: Researching the actuality of projects. *International Journal of Project Management*, v. 24, n. 8, p. 675-686, novembro, 2006.
- CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; MERSON, P.; NORD, R.; STAFFORD, J. *Documenting software architectures: vies and beyond*. Boston -MA: Peason Education, 2ed., 2004. 254p.
- COCKBURN, A.; HIGHSMITH, J. *Agile software development, the people factor*. Computer, vol. 34, nº 11, p.131 – 133, novembro, 2001.

COCKBURN, Alistair. Agile software development joins the “would-be” crowd. Janeiro, 2002. Disponível em: <<http://www.agilealliance.org/system/article/file/782/file.pdf>> Acesso em: abril, 2007.

COHN, M. **Agile Estimating and Planning**. Prentice Hall PTR: New York, 2005.

COHN, M., FORD, D. Introducing an agile process to an organization. IEEE Computer Magazine, June 2003, [S.l.], p. 74-78, 2003.

CONCAS, G. FRANCESCO, M. MARCHESI, M. QUARESIMA, R.; PINNA, S. An Agile Development Process and Its Assessment Using Quantitative Object-Oriented Metrics. p.83-93, 2008.

CONFORTO, E. C.; AMARAL, D. C. Evaluating an agile method for planning and controlling innovative projects. Project Management Journal, v. 41, n. 2, p. 73-80. doi: 10.1002/pmj.20089, 2010.

CONFORTO, E. C. *Gerenciamento ágil de projetos: proposta e avaliação de método para gestão de escopo e tempo*. Dissertação (Mestrado em Engenharia de Produção). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009.

CORAM, M.; BOHNER, S. The Impact of Agile Methods on Software Project Management. **12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)**, p. 363-370. Ieee. doi: 10.1109/ECBS.2005.68, 2005.

DIAS, M. V. B. Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software. Dissertação (Mestrado em Administração). Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, São Paulo, 2005.

DOTPROJECT. Disponível em < <http://www.dotproject.net>>. Acesso em maio, 2009.

ELO. Products. Disponível em: < <http://www.elotouch.com.br/Produtos/detech2.asp>>. Acesso em: junho, 2009.

ENGEL, M. Four purpose for retail signage. In: KIRCHER, T. Digital display technology: learning the basics of digital signage. Network Alliance Media, 2008. Disponível em <http://www.digitalsignagetoday.com/white_paper.php?id=66>. Acesso em: abril, 2009.

FRANCO, A. Flex vs. AJAX Friends or Foes. **White Paper. January**. Disponível em: <<http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Flex+vs+.+AJAX+Friends+or+Foes+?#0>>, 2008. Acesso em: maio, 2010.

GARCIA, I.; PACHECO, C. A Web-based Tool for Automatizing the Software Process Improvement Initiatives in Small Software Enterprises. **IEEE Latin America Transactions**, v. 8, n. 6, p. 685-694, 2010.

GARCÍA, J.; AMESCUA, A.; SÁNCHEZ, M.-I.; BERMÓN, L. Design guidelines for software processes knowledge repository development. **Information and Software Technology**, v. 53, n. 8, p. 834-850, 2011.

GARLAN D.; MONROE R.; WILE D.; Acme: An Architecture Description Interchange Language. In: Johnson Howard, eds. *CASCON '97 Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research*. Canada; 1997:1 -15. Disponível em: <http://portal.acm.org/citation.cfm?id=782017>. Consultado em: julho de 2011.

GARLAN, D.; PERRY, D. Introduction to the special issue on software architecture. *IEEE Transactions on Software Engineering*, v. 21, n. 4, 1995.

HAMERI, A. PUITTINEN, R. WWW-enabled knowledge management for distributed engineering projects. **Computers in Industry**, vol. 50, n. 2, p. 165-177, fevereiro, 2003.

HIGHSMITH, J. **Agile Project Management: Creating Innovative Products**. Boston: Adison-Wesley, 2004.

HIRSCHFELD, H. **Planejamento com PERT-CPM e análise do desempenho: método manual e por computadores eletrônicos aplicados a todos os fins**. Ed.8. São Paulo: Atlas, 1985.

HUANG, G, H. Web-based support for collaborative product design review. **Computers in Industry**, vol. 48, n.1 , p. 71-88, maio, 2002.

HUANG, G. Q.; MAK, K. L. Agent-based Collaboration between Distributed Web Applications: Case Study on “Collaborative Design for X” Using CyberCO. **Concurrent Engineer: Research and Applications**, vol. 10, n. 2, p. 279-290, 2002.

JONES L.; KONRAD M. Capability Maturity Model Integration (CMMI) V1 . 3 and Engineering. 2011; Disponível em: <http://www.sei.cmu.edu/library/abstracts/webinars/Capability-Maturity-Model-Integration-and-Architecture-Centric-Engineering.cfm>.

KERZNER, H. **Gestão de projetos: as melhores práticas**. Tradução de Marco Antonio Viana Borges et al. Porto Alegre: Ed. Bookman, 2002.

KIOSK MARKETPLACE. 12 MAY 2008. Produced by Patrick Avery. European market growing. Disponível em <http://www.kioskmarketplace.com/article.php?id=19861&s=1>. Acesso em: abril, 2009.

KIRCHER, T. Digital display technology: learning the basics of digital signage. Network Alliance Media, 2008. Disponível em <http://www.digitalsignagetoday.com/white_paper.php?id=66>. Acesso em abril 2009.

KRUCHTEN P.; OBBINK H.; STAFFORD J. The Past, Present, and Future for Software Architecture. *IEEE Software*, v.23, n.2, p. 22-30, 2006. Disponível em:: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1605175>.

LANKHORST M. Enterprise architecture modelling—the issue of integration. *Advanced Engineering Informatics*. v.18, n.4, p. 205-216, 2004.

LI, W. D.; FUH, J.Y.H.; WONG, Y.S, An Internet-enabled integrated system for co-design and concurrent engineering, **Computers in Industry**, v.55, n.1, p. 87-103, 2004.

LIANG, W.Y.; HUANG, C.C. The agent-based collaboration information system of product development. **International Journal of Information Management**, v. 22, n. 3, p. 211-224, junho, 2002.

MAYLOR, H. Beyond the Gantt Chart: project management moving on. **Business Management Journal**, v.19, n.1, pp.92-100, 2001.

MENDES, A. Arquitetura de software: Desenvolvimento orientado para arquitetura. Campus, 2002.

MEDVIDOVIC, N.; ROSENBLUM, D. S. Domains of concern in software architectures and architecture description languages. Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL), 1997. **Anais...** p.16–16, 1997. USENIX Association. Disponível em: <<http://dl.acm.org/citation.cfm?id=1267966>>.

MEDVIDOVIC, N.; TAYLOR RN. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*. 2000; v.26, n.1, p.70-93, 2000. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=825767>. Consultado em: jul.,2011.

MEDVIDOVIC N.; TAYLOR RN. A framework for classifying and comparing architecture description languages. In: Jazayeri M, Schauer H, eds. ESEC '97/FSE-5 Proceedings of the

6th European SOFTWARE ENGINEERING conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering. Vol 22. zurich: Springer-Verlag New York, Inc. New York, NY,; 1997:60-76. Available at: <http://portal.acm.org/citation.cfm?doid=267896.267903>.

MEJÍA, R.; LÓPEZ, A.; MOLINA, A. Experiences in developing collaborative engineering environments: An action research approach. **Computers in Industry**, vol. 58, n. 4, p. 329-346, maio, 2007.

NAKAGAWA EY. Uma Contribuição ao Projeto Arquitetural de Ambientes de Engenharia de Software. Tese (Doutorado em Ciência da Computação), Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, 2006, p.252.

NGOYI, G. A. B. et al. Iterative Design Method for Video Processors Based on an Architecture Design Language and its Application to ELA Deinterlacing. 2008 Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference, p. 37-40, 2008.

OMMERING, R. V.; KRAMER, J. The Koala Component model for consumer electronics software. *Computer*. v.33, n.3, p.78-85, 2000. Disponível em: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=825699&tag=1.

PONS, D. Project management for new product development. *Project Management Journal*, v. 39, n. 2, p. 82-97, junho, 2008.

PROCTER, R. et al. Agile Project Management: A Case Study of a Virtual Research Environment Development Project. *Computer Supported Cooperative Work-the Journal of Collaborative Computing*, v. 20, n. 3, p. 197-225, Jun 2011.

PROJECT MANAGEMENT INSTITUTE. PMBOK: A Guide to the Project Management Body of Knowledge. 4th ed. Project Management Institute Inc., 2008.

QIAN, F.; and SHENSHENG, F. Product Development Process Management System Based on P_PROCE Model. **Concurrent Engineer: Research and Applications**, v.10, n.3, p.203-211, 2002.

QIN, S.F. et al., A framework of web-based conceptual design, **Computers in Industry**, v.50, n.2, pp. 153-164, 2003.

RODGERS, P. A. et al. The management of concept design knowledge in modern product development organizations. *International Journal of Computer Integrated Manufacturing*, vol. 14, nº. 1, p. 108-115, 2001.

- RODRIGUEZ, K.; AL-ASHAB, A. Knowledge web-based system architecture for collaborative product development. **Computers in Industry**, v. 56, n.1, p. 125-140, 2005.
- ROZENFELD, H. et al. Gestão do Desenvolvimento de Produtos - uma referência para a melhoria do processo. Saraiva: São Paulo, 2006.
- RUSU, L.; LIN, Y. F.; HODOSI, G. Management Guidelines for Database Developers' Teams in Software Development Projects. Best Practices for the Knowledge Society: Knowledge, Learning, Development and Technology for All, v. 49, p. 377-386, 2009.
- RYCHLÝ, M.; TICHÁ, P. A Tool for Supporting Feature-Driven Development. In: B. Meyer; J. R. Nawrocki; B. Walter (Eds.); **Lecture Notes in Computer Science**. v. 5082, p.196-207, Heidelberg: Springer. 2008.
- SANJIV, A.; WOODCOCK, S. Agile project management: emergent order through visionary leadership. Maio, 2003. Disponível em: <<http://www.ccpace.com/Resources/AgileProjectManagement.pdf>>. Acesso em: abril 2009.
- SHAW, M.; CLEMENTS, P. A field guide to boxology: Preliminary classification of architectural styles for software systems. In: Proc. COMPSAC97, 21st International Computer Software and Applications Conference, Washington, USA, 1997, p. 7-13.
- SHEN, W.; HAO, Q.; LI, W. Computer supported collaborative design: Retrospective and perspective. Computers in Industry, vol 59, n.9, p. 855-862, 2008.
- SCHWABER, Ken. Agile Project Management with Scrum. Microsoft Press, 2004.
- SILLITTI, A.; SUCCI, G. Source code repositories and agile methods. In: BAUMEISTER, H. M. M. H. M. (Ed.). Extreme Programming and Agile Processes in Software Engineering, Proceedings, v.3556, 2005.
- SOMMERVILLE, I.; SAWYER, P. Requirements Engineering – A Good Practice Guide. John Wiley & Sons, 1997.
- SMITE, D. et al. Empirical evidence in global software engineering: a systematic review. Empirical Software Engineering, v. 15, n. 1, p. 91-118, Feb 2010.
- SPENCER, D.; ZIMMERMAN, A.; ABRAMSON, D. Special Theme: Project Management in E-Science: Challenges and Opportunities. Computer Supported Cooperative Work-the Journal of Collaborative Computing, v. 20, n. 3, p. 155-163, Jun 2011.

STORZ, O., FRIDAY, A.; DAVIES, N. Supporting content scheduling on situated public displays. **Computers & Graphics**, v. 30, n. 5, p. 681-691, 2006. ISSN 0097-8493. DOI: 10.1016/j.cag.2006.07.002. Disponível em:

<<http://www.sciencedirect.com/science/article/B6TYG-4KV8TFD-1/2/08e08e0c0d49e4afa09972015b585928>>. Acesso em: 1 jun. 2010.

TAY, F. E.; ROY, A. CyberCAD: a collaborative approach in 3D-CAD technology in a multimedia-supported environment, **Computers in Industry**, v.52, n.2, pp. 127-145, 2003.

TAYLOR, R.; TRACZ, W.; COGLIANESE, L. Software development using domain-specific software architecture: Cdr1 a011a curriculum module in the SEI style. **SIGSOFT Software Engineering Notes**, v. 20, p. 27–37, 1995.

TAYLOR, R. N.; BELZ, F. C.; CLARKE, L. A.; OSTERWEIL, L.; SELBY, R. W.; WILEDEN, J. C.; WOLF, A. L.; YOUNG, M. Foundations for the Arcadia environment architecture. **SIGSOFT Softw. Eng. Notes**, v. 13, n. 5, p. 1–13, 1988.

W3SCHOOLS. 1999. Disponível em < [http:// www.w3schools.com](http://www.w3schools.com)>. Acesso em agosto, 2009.

WANG, P.: Digital signage 101: a quick introduction to those who are new to digital signage. [19--]. Disponível em: <http://digitalsignage.com/tools/articles>. Acesso em: 1 jul. 2010.

WANG, Y.; SHEN, W.; GHENNIWA, H. WebBlow: a Web/agent-based multidisciplinary design optimization environment. **Computers in Industry**, v. 52, n. 1, p. 17-28, 2003.

WHITE, D.; FORTUNE, J. Current practice in project management: an empirical study. **International Journal of Project Management**, v. 20, n. 1, p. 1-11, janeiro, 2002.

WINTER, M. et al. Directions for future research in project management: the main findings of a UK government-funded research network. **International Journal of Project Management**, v. 24, n. 8, p. 638-649, novembro, 2006.

WOERNER, J.; WOERN, H. A security architecture integrated co-operative engineering platform for organised model exchange in a Digital Factory environment. **Computers in Industry**, v. 56, n.4, p. 347-360, maio, 2005.

WU, S. et al. Personal assistant agents for collaborative design environments. **Computers in Industry**, v. 57, n.8-9, p. 732-739, dezembro, 2006.

VARGAS, R. Microsoft project 2010: standard & professional. Rio de Janeiro: Brasport, 2011.