Escola Politécnica da Universidade de São Paulo

Cláudio Luís Pereira Ferreira

**Maestro: Um Middleware para Suporte
a Aplicações Distribuídas Baseadas
em Componentes de Software**

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

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Jorge Luís Risco Becerra

São Paulo
2001

**Aos meus pais, Cláudio e Lizete,
pelo incentivo e apoio.**

Ao Prof. Dr. Jorge Luís Risco Becerra pela amizade, orientação e incentivo constante, responsável pelo desenvolvimento deste trabalho e da minha vida profissional.

À Profª Drª Selma Melnikoff pelo apoio e constantes revisões deste trabalho.

Ao Prof. Dr. Moacyr Martucci Jr. pelo incentivo, conselhos e sugestões oferecidos.

Ao pessoal do Laboratório de Sistemas Abertos (LSA) por toda a ajuda prestada para a conclusão deste trabalho.

Aos colegas do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo pelo apoio e incentivo.

A meus familiares pela compreensão e estímulo.

A todas as pessoas que de alguma forma ajudaram a alcançar esta grande meta na minha vida.

Sumário

Lista de Figuras e Tabelas

Lista de Siglas e Abreviaturas

Resumo

Abstract

CAPÍTULO 1. INTRODUÇÃO	1
1.1 <i>CONSIDERAÇÕES INICIAIS</i>	2
1.2 <i>OBJETIVOS</i>	3
1.3 <i>ABRANGÊNCIA</i>	4
1.4 <i>MOTIVAÇÃO</i>	5
1.5 <i>JUSTIFICATIVAS</i>	6
1.6 <i>METODOLOGIA</i>	8
1.7 <i>HISTÓRICO DA PESQUISA</i>	9
1.8 <i>ESTRUTURA DA DISSERTAÇÃO</i>	10
 CAPÍTULO 2. MIDDLEWARE, COMPONENTES DE SOFTWARE E O MODELO DE REFERÊNCIA ODP.....	11
2.1 <i>MIDDLEWARE</i>	17
2.2 <i>COMPONENTES DE SOFTWARE</i>	18
2.3 <i>O MODELO DE REFERÊNCIA ODP</i>	22
2.3.1 <i>O Ponto de Vista da Empresa</i>	24
2.3.2 <i>O Ponto de Vista da Informação</i>	26
2.3.3 <i>O Ponto de Vista da Computação</i>	27
2.3.4 <i>O Ponto de Vista da Engenharia</i>	30
2.3.5 <i>O Ponto de Vista da Tecnologia</i>	33
 CAPÍTULO 3. ESPECIFICAÇÃO E ARQUITETURA DO MIDDLEWARE MAESTRO.....	34
3.1 <i>ESPECIFICAÇÃO DO MIDDLEWARE MAESTRO UTILIZANDO O RM-ODP</i>	40
3.1.1 <i>O Ponto de Vista da Empresa</i>	41
3.1.2 <i>O Ponto de Vista da Informação</i>	43
3.1.3 <i>O Ponto de Vista da Computação</i>	49
3.1.4 <i>O Ponto de Vista da Engenharia</i>	51
3.1.5 <i>O Ponto de Vista da Tecnologia</i>	53
3.2 <i>ARQUITETURA DO MIDDLEWARE MAESTRO</i>	54

CAPÍTULO 4. UM ESTUDO DE CASO EM AUTOMAÇÃO DO GERENCIAMENTO DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE	58
4.1 O GERENCIAMENTO DE PROJETOS.....	59
4.2 A QUALIDADE DO SOFTWARE.....	61
4.3 O SISTEMA DE AUTOMAÇÃO DE PROJETOS DE SOFTWARE (SAPS)	64
4.4 O SISTEMA SAPS E OS PONTOS DE VISTA DO ODP.....	68
4.4.1 O Ponto de Vista da Empresa.....	68
4.4.2 O Ponto de Vista da Informação	71
4.4.3 O Ponto de Vista da Computação.....	73
4.4.4 O Ponto de Vista da Engenharia	77
4.4.5 O Ponto de Vista da Tecnologia.....	79
4.5 A ARQUITETURA DO SISTEMA SAPS.....	82
4.6 CONCLUSÕES DO CAPÍTULO	87
 CAPÍTULO 5. CONSIDERAÇÕES FINAIS.....	88
5.1 CONCLUSÕES	89
5.2 COMENTÁRIOS GERAIS	92
5.3 CONTINUIDADE DA PEQUISA.....	94
5.4 CONTRIBUIÇÃO ACADÊMICA.....	97
 BIBLIOGRAFIA	99

Lista de Figuras e Tabelas

FIGURA 2.1	UMA VISÃO FUNCIONAL DO MIDDLEWARE	14
FIGURA 2.2	ILUSTRAÇÃO DE UM COMPONENTE QUE SUPORTA TRÊS INTERFACES A, B E C.....	19
FIGURA 2.3	MODELO DE COMUNICAÇÃO ENTRE COMPONENTES DO PADRÃO DCOM.....	21
FIGURA 2.4	PONTOS DE VISTA DO RM-ODP.....	23
FIGURA 2.5	SISTEMA DE COMUNICAÇÃO MULTIMÍDIA QUE ILUSTRA OS CONCEITOS ODP DO PONTO DE VISTA DA COMPUTAÇÃO.....	29
FIGURA 2.6	SISTEMA CLIENTE/SERVIDOR QUE ILUSTRA OS CONCEITOS ODP DO PONTO DE VISTA DA ENGENHARIA.	32
FIGURA 3.1	CENÁRIOS DOS ATUAIS SISTEMAS DISTRIBUÍDOS BASEADO NA ARQUITETURA DO MIDDLEWARE MAESTRO	37
FIGURA 3.2	MECANISMO DE EXECUÇÃO DE UMA APLICAÇÃO DISTRIBUÍDA BASEADA NA ARQUITETURA DO MIDDLEWARE MAESTRO.....	39
FIGURA 3.3	ESPECIFICAÇÃO DO PONTO DE VISTA ODP DA EMPRESA PARA O MIDDLEWARE MAESTRO UTILIZANDO DIAGRAMA DE CLASSES DO UML.....	44
FIGURA 3.4	ESPECIFICAÇÃO DO PONTO DE VISTA ODP DA INFORMAÇÃO PARA O MIDDLEWARE MAESTRO UTILIZANDO DIAGRAMA DE CLASSES DO UML	46
FIGURA 3.5	DIAGRAMA DE COLABORAÇÃO DO PROCESSAMENTO DE UMA APLICAÇÃO REMOTA NO MIDDLEWARE MAESTRO.....	48
FIGURA 3.6	ESPECIFICAÇÃO DO PONTO DE VISTA ODP DA COMPUTAÇÃO PARA O MIDDLEWARE MAESTRO UTILIZANDO DIAGRAMA DE CLASSES DO UML	50
FIGURA 3.7	ESPECIFICAÇÃO DO PONTO DE VISTA ODP DA ENGENHARIA PARA O MIDDLEWARE MAESTRO UTILIZANDO DIAGRAMA DE IMPLANTAÇÃO DO UML	52
FIGURA 3.8	ESPECIFICAÇÃO DO PONTO DE VISTA ODP DA TECNOLOGIA PARA O MIDDLEWARE MAESTRO UTILIZANDO DIAGRAMA DE IMPLANTAÇÃO DO UML	55
FIGURA 3.9	ARQUITETURA DO MIDDLEWARE MAESTRO.....	57

FIGURA 4.1	ESTRATÉGIA PARA O DESENVOLVIMENTO DO SISTEMA SAPS.....	60
FIGURA 4.2	DIAGRAMA IDEF0 (NÍVEL 1) DO SISTEMA SAPS	65
TABELA 4.1	RELAÇÃO DOS MÓDULOS DO SISTEMA SAPS COM AS ÁREAS- CHAVES DO MODELO DE QUALIDADE SW-CMM.....	67
FIGURA 4.3	DIAGRAMA DE CASO DE USO DO SISTEMA SAPS E DO MIDDLEWARE MAESTRO NA VISÃO EMPRESA DO ODP.....	70
FIGURA 4.4	DIAGRAMA DE OBJETOS DO SISTEMA SAPS E DO MIDDLEWARE MAESTRO NA VISÃO INFORMAÇÃO DO ODP.....	72
FIGURA 4.5	DIAGRAMA DE CLASSES DO SISTEMA SAPS E DO MIDDLEWARE MAESTRO NA VISÃO COMPUTAÇÃO DO ODP.....	74
FIGURA 4.6	CLASSES PERSISTENTES DO COMPONENTE GERENCIADOR DE OBJETOS PERSISTENTES.....	75
FIGURA 4.7	DIAGRAMA DE COMPONENTES DO SISTEMA SAPS E DO MIDDLEWARE MAESTRO NA VISÃO ENGENHARIA DO ODP	78
FIGURA 4.8	DIAGRAMA DE COMPONENTES DO SISTEMA SAPS E DO MIDDLEWARE MAESTRO NA VISÃO TECNOLOGIA DO ODP	80
FIGURA 4.9	ARQUITETURA DO SISTEMA SAPS BASEADA NO MODELO DE REFERÊNCIA SAA	83
FIGURA 4.10	ARQUITETURA DO SISTEMA SAPS INTEGRADA COM A ARQUITETURA DO MIDDLEWARE MAESTRO	85
FIGURA 5.1	MÉTODO DE DECOMPOSIÇÃO DO SISTEMA ODP UTILIZADO PARA A ANÁLISE DO MIDDLEWARE MAESTRO.....	91
FIGURA 5.2	ESTRUTURA DE PESQUISA.....	95
FIGURA 5.3	INTEGRAÇÃO DAS TECNOLOGIAS COM E CORBA.....	96

Lista de Siglas e Abreviaturas

ADO	ActiveX Data Objects
ASP	Active Server Pages
CCITT	Comite' Consultatif International de Telegraphique et Telephonique
CMM	Capability Maturity Model
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
EDI	Electronic Data Interchange
ESBC	Engenharia de Software Baseada em Componentes
HTML	Hyper Text Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICAM	Integrated Computer Aided Manufacturing
IDEF	ICAM Definition Method
IDL	Interface Definition Language
IEC	International Electrotechnical Commission
IHM	Interface Homem-Máquina
IIS	Internet Information Server
IPX/SPX	Internetwork Packet Exchange / Sequenced Packet Exchange
ISO	International Organization for Standardization
ITU	International Telecommunications Union
LSA	Laboratório de Sistemas Abertos
MVC	Model-View-Controller
ODBC	Open DataBase Connectivity
ODP	Open Distributed Processing
OLE	Object Linking and Embedding
OMG	Object Management Group
ORB	Object Request Broker
OSF	Open Software Foundation

PAC	Presentation-Abstraction-Control
PSP	Personal Software Process
QoS	Quality of Service
RM-ODP	Reference Model for Open Distributed Processing
RPC	Remote Procedure Call
SAA	Sistema Aberto de Automação
SAPS	Sistema de Automação de Projetos de Software
SEI	Software Engineering Institute
SQL	Structured Query Language
TSP	Team Software Process
UML	Unified Modeling Language
UUID	Universally Unique Identifier
VBA	Visual Basic for Applications
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WOSA	Windows Open Services Architecture
WWW	World Wide Web

Resumo

A atribuição do nome *Maestro* ao título deste trabalho, em substituição a um acrônimo, foi proposital uma vez que representa melhor a finalidade do sistema. Tal como a função de um maestro junto a uma orquestra, é o trabalho de um *middleware* organizar as atividades de seus diversos elementos componentes de forma a serem operados em sintonia com a execução de uma aplicação. O resultado deste trabalho deve ser transparente para quem interage com o sistema, percebendo-o como um único bloco coeso e sincronizado, orquestrado por um agente principal.

Esta é então a proposta principal deste trabalho, a especificação de um *middleware* e seus componentes internos indicando suas principais características e funcionalidades assim como sua operação na execução de aplicações distribuídas. Foi levado em conta também o novo ambiente no qual as aplicações distribuídas estão inseridas tais como a diversidade de dispositivos utilizados pelos seus usuários, a necessidade de constantes alterações nos sistemas, a utilização de novas tecnologias no desenvolvimento de *software* e a necessidade de definição de sistemas abertos.

Para a especificação deste *middleware* foi utilizado o modelo de referência *Open Distributed Processing* (ODP) da ISO/IEC que permite visualizar o sistema segundo cinco pontos de vista distintos. A linguagem *Unified Modeling Language* (UML) da OMG também é utilizada como ferramenta de modelagem para cada ponto de vista. Ao final o sistema é especificado utilizando a tecnologia de componentes de *software*, ilustrando seu uso através de componentes de *software* comerciais.

Ao final do trabalho, um sistema de automação de projetos de *software* é especificado e implementado para ilustrar o funcionamento do *middleware*.

Abstract

The attribution of the name *Maestro* to the title of this work, in substitution to an acronym, was deliberated once it better represents the system's goal. As the duties of a maestro with an orchestra, it's the job of a middleware to organize the activities of its different component elements as to operate in synchrony with the execution of an application. The result of this work should be transparent to whom interact with the system, perceiving it as a single synchronized and cohered block, orchestrated by a master agent.

This is the subject of this work, the specification of a middleware and its internal components indicating its major characteristics and functionalities and also its operation in the execution of distributed applications. It was also taken into account the new environment in which the distributed applications are inserted such as the diversity of devices managed by the users, the necessity for constant system changing, the use of new technologies in software development and the necessity for definition of open systems.

For the specification of this middleware, it was used the reference model of *Open Distributed Processing* (ODP) from ISO/IEC that allows a system to be visualized by five different points of view. The language *Unified Modeling Language* (UML) from OMG is also used as a modeling tool for each point of view. By the end the system is specified using the technology of component software, illustrating its use through commercial component software.

By the end of this work, a software project automation system is specified and implemented to illustrate how the middleware works.

Capítulo 1

Introdução

Este capítulo apresenta os subsídios necessários para a definição preliminar do trabalho de dissertação de mestrado intitulado “*Maestro: Um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software*”. Ao longo do texto, este *middleware* será referenciado pelo nome *Maestro*.

O contexto geral deste capítulo está composto de tópicos referentes aos objetivos, abrangência, motivação, justificativa e a metodologia utilizada.

1.1 Considerações Iniciais

Em quase todas as empresas, existe um certo grau de distribuição geográfica entre os componentes de seu sistema de informação, e alguns requisitos para a comunicação entre as partes. Além disto, em quase todas também existe um certo grau de autonomia entre estes componentes onde cada um se comunica de diferentes maneiras. Estas e outras características inerentes dos sistemas distribuídos tornam extremamente complexa a tarefa de integração de seus serviços de forma que as empresas possam obter resultados consolidados de suas operações de maneira rápida e eficiente.

A inflexibilidade causada pela maneira rígida na qual seus sistemas estão interligados e a heterogeneidade dos elementos do sistema de informação são os principais responsáveis pela demora das empresas em responder aos requisitos do mercado, diminuindo sua competitividade e lucratividade e muitas vezes sendo responsáveis pela sua própria sobrevivência.

Dadas estas considerações iniciais, será analisado neste trabalho a especificação do *middleware Maestro*, cujo principal objetivo é o de integrar os diversos serviços disponíveis num sistema ou entre sistemas de informação corporativos de maneira a consolidar as informações disponíveis em cada um de maneira ágil e flexível. Esta integração deverá se dar através de aplicações corporativas utilizadas por seus usuários, cujos detalhes referentes à distribuição de seu sistema serão abstraídos destes através do suporte oferecido pelo *Maestro*.

1.2 Objetivos

O objetivo principal deste trabalho está relacionado à especificação de uma arquitetura baseada no modelo de referência *Open Distributed Processing* (RM-ODP) da ISO/IEC de um middleware para o suporte a aplicações distribuídas baseadas em componentes de *software*, o qual é referenciado ao longo do texto pelo nome *Maestro*.

O segundo objetivo está relacionado à utilização do ODP na especificação do *middleware Maestro*, possibilitando a verificação de sua eficiência e adequação a este tipo de análise, uma vez que este é um modelo atual e que precisa de comprovações práticas com respeito aos domínios de sua utilização.

No terceiro objetivo está a definição de uma estratégia para obter uma arquitetura aberta utilizando diferentes estilos arquiteturais compondo assim uma arquitetura heterogênea, aumentando assim a flexibilidade de sua utilização.

Através destes três objetivos iniciais deve-se chegar a um modelo prático, inteligível e eficiente de um *middleware* que determine o mecanismo de acesso a seus serviços e defina suas características como sendo aberta, escalável, interoperável, integrável, flexível, modular, portátil, gerenciável e seguro [ISO-98].

O quarto objetivo representa o aspecto prático do mesmo, onde será definida uma especificação da implementação da arquitetura estudada para um caso em sistemas de automação de projetos de *software*. Neste serão estudadas e indicadas as tecnologias e ferramentas, sejam estas abertas ou proprietárias, a serem utilizadas na sua implementação através de componentes de *software*.

Este quarto objetivo visa ainda obter conclusões sobre o uso e integração das tecnologias selecionadas na implementação deste sistema.

O quinto e último objetivo deste trabalho está dentro do contexto acadêmico e de pesquisa, o qual deverá impulsionar as atividades de ensino, pesquisa, desenvolvimento e extensão do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, em particular do Laboratório de Sistemas Abertos, a partir das conclusões e resultados obtidos neste trabalho.

Como resultado deste trabalho, teremos a criação de novos contextos de pesquisa que poderão ser experimentados e verificados em futuros trabalhos acadêmicos através de sua aplicação em diferentes domínios de aplicação, como nas áreas de automação, sistemas distribuídos e abertos, arquitetura de *software*, engenharia de *software* e tecnologias interativas.

1.3 Abrangência

O desenvolvimento da arquitetura *Maestro*, proposta neste trabalho de dissertação, pretende incorporar a ela uma série de características e serviços desejáveis aos atuais sistemas distribuídos de informação. A obtenção destas características e serviços, porém, deverá se limitar ao máximo às atuais possibilidades das tecnologias disponíveis, sejam estas padronizadas *de facto* ou *de jure*, buscando promover sua reutilização e uma maior abertura do sistema e evitando o desenvolvimento de complexas funcionalidades que não encontram (ou torna-se muito complicada a utilização de) uma tecnologia direta para sua implementação.

Para especificar a arquitetura deste sistema, será utilizado o modelo de referência da ISO/IEC para processamento distribuído e aberto (RM-ODP) excetuando-se a sua quarta parte que trata das semânticas arquiteturais e tem por objetivo formalizar os conceitos de modelagem do ODP.

Será feito um projeto de um sistema simples para verificação da funcionalidade da arquitetura *Maestro*. Este sistema não possuirá todas as características e serviços a serem definidos devido à sua complexidade estrutural e ao tempo

necessário para sua implementação. Pretende-se, porém, demonstrar suas funcionalidades básicas através da integração de componentes de *software* com serviços de distribuição da tecnologia WWW.

É bom ressaltar que a aplicação desta arquitetura deverá ser a mais abrangente possível. Sua especificação, porém, não pretende ser completa mas sim funcional para uma ampla gama de casos, deixando questões mais complexas em aberto para futuras pesquisas.

1.4 Motivação

A motivação deste trabalho de dissertação foi encontrada em quatro assuntos que merecem destaque: no estudo da área de sistemas distribuídos; na pesquisa em torno de novos modelos de aplicativos para estes ambientes baseados em modernas tecnologias de desenvolvimento de *software*; na utilização do padrão ODP para especificação destes sistemas; e, finalmente, na preocupação em orientar a comunidade empresarial e acadêmica na utilização de novas tecnologias de informação.

O motivo principal está na pesquisa em torno de um *middleware* que permita o suporte ao desenvolvimento e execução de novas aplicações distribuídas de maneira eficiente e transparente e que aproveite as tecnologias atualmente oferecidas por diferentes fabricantes.

Ocorre hoje uma necessidade por um menor tempo de desenvolvimento na construção de aplicações distribuídas. Isto motiva a pesquisa em torno de soluções que possibilitem esta redução através de um conjunto de serviços de suporte unidos ao uso de modernas tecnologias de desenvolvimento de *software*, em particular às relacionadas à engenharia de *software* baseada em componentes [\[AOYAMA-98\]](#).

Outra fonte de motivação encontra-se no estudo de sistemas distribuídos buscando analisar suas características intrínsecas e propriedades desejáveis.

Esta análise serviria de base para a identificação de tecnologias de informação atuais que permitissem o controle destas características e a implementação destas propriedades. Com isto seria possível o mapeamento de seu esquema conceitual para um esquema interno utilizando estas tecnologias de maneira integrada.

O uso do padrão ODP na especificação do sistema em estudo é também um fator motivante uma vez que este busca estruturar, de uma maneira controlável, a heterogeneidade de um sistema distribuído [ISO-98]. Esta seria uma oportunidade de verificar seu uso e eficiência na descrição de um *middleware* e dos componentes de sua infra-estrutura.

O desenvolvimento de sistemas distribuídos é assunto de interesse na comunidade, seja esta empresarial ou acadêmica. Sua aplicação exige dos usuários profundos conhecimentos sobre diferentes tipos de tecnologias e, neste contexto, o autor motiva-se a propor uma solução que as integre, tornando suas características individuais transparentes aos usuários, facilitando assim seu entendimento e utilização.

Por fim, é também fator de motivação a possibilidade de orientar e contribuir na divulgação de novas tecnologias objetivando o aumento do nível tecnológico de nossas empresas, papel este fundamental na atuação da universidade dentro da comunidade empresarial.

1.5 Justificativas

Existe atualmente uma necessidade, criada pelo mercado, por novas aplicações corporativas (comércio eletrônico, intercâmbio de dados entre empresas, sistemas de automação, gestão empresarial) que sejam mais flexíveis, modulares, de fácil manutenção e que permitam aproximar as empresas de seus clientes. Essas aplicações, de natureza invariavelmente distribuída dada a distribuição geográfica de seus clientes, parceiros de negócio e recursos, são de difícil desenvolvimento e atualmente restringem-se,

em sua maioria, a ambientes isolados e controlados. A necessidade atual por aplicações flexíveis e abertas, ou seja, que se adaptem a uma ampla variedade de cenários possíveis, justifica a pesquisa em torno de um sistema que suporte a integração e a interoperabilidade corporativa.

Isto cria outra justificativa ao estudo e aplicação de novos modelos de especificação e desenvolvimento de software que permitam definir e implementar tanto o *middleware Maestro* como as aplicações que serão executadas no mesmo.

No âmbito de modelos de especificação, merece atenção o modelo de referência ODP que permite a análise de complexos sistemas distribuídos segundo cinco pontos de vista, buscando atender aos requisitos dos sistemas distribuídos e abertos descritos em [\[ISO-98\]](#), e se trata de uma linha de pesquisa nesta área.

No âmbito do desenvolvimento de *software*, merece atenção o campo da *Engenharia de Software Baseada em Componentes* (ESBC), que permite a construção de aplicações através da composição de partes reutilizáveis, conhecidas por componentes de *software*.

O estudo destes dois campos criam, por si só, uma nova justificativa pois estes são ainda muito novos e merecem um estudo aprofundado quanto ao seu uso eficiente em sistemas, para consolidá-los como opções viáveis e fundamentais na construção dos mesmos [\[BECERRA-98\]](#), [\[KIELY-98\]](#).

Outra justificativa está no incentivo ao desenvolvimento de sistemas distribuídos e abertos pelas empresas. Como consequência deste trabalho, será apresentada uma proposta prática de desenvolvimento de um sistema ODP utilizando componentes de *software*, a qual poderá servir de referência básica em outras aplicações acadêmicas e da indústria.

Por fim, os resultados desta dissertação devem gerar novas informações, de referência e de pesquisa, que poderão ser utilizadas no desenvolvimento de pesquisas em diferentes níveis acadêmicos (mestrado, doutorado ou iniciação científica), dentro da linha denominada de sistemas distribuídos e abertos. Esta razão cria outra justificativa ao trabalho.

1.6 Metodologia

Para o desenvolvimento desta dissertação, e com objetivo de gerenciar a evolução do processo visando as metas e a qualidade, está sendo utilizada uma metodologia cujas fases são definidas a seguir:

- ? Pesquisa: fase que engloba a seleção das informações, as quais foram estruturadas em grupos. Cada grupo corresponde a um assunto relevante da dissertação em termos de seus requisitos gerais. Esta atividade é realizada ao longo do processo de desenvolvimento, sendo que esta diminui perto do final do trabalho;
- ? Análise do sistema: fase que engloba as atividades de caracterização do *middleware Maestro*, da especificação e modelagem deste em cinco pontos de vista e de sua especificação, utilizando-se, para tanto, do modelo de referência ODP e da linguagem de modelagem UML;
- ? Definição da arquitetura: fase que engloba a proposta de uma arquitetura que represente o sistema em estudo. Aqui deverão ser analisadas as várias propostas de arquitetura e, segundo a especificação da fase anterior, definir uma arquitetura específica;
- ? Implementação: fase que engloba a especificação da implementação do sistema. Nesta fase serão avaliadas as tecnologias atualmente disponíveis que permitam a implementação dos serviços do sistema. Em alguns casos, onde não houver uma tecnologia disponível, será especificada a implementação do elemento em particular. Uma implementação prática de

um subconjunto do sistema deverá ser feita nesta fase, através de um estudo de caso, visando avaliar parte de suas propriedades.

1.7 Histórico da Pesquisa

Este item tem por objetivo relacionar trabalhos de pesquisa anteriores, realizados no Laboratório de Sistemas Abertos (LSA) do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, que influenciaram no trabalho de dissertação do autor.

O início da pesquisa encontra-se no trabalho de [\[MARTUCCI-92\]](#) que propõe a utilização do conceito de sistemas abertos na automação e apresenta o *modelo de sistema aberto de automação* (SAA), cuja estrutura hierárquica poderia ser aplicada em vários segmentos da indústria. Sua aplicação encontra-se nos trabalhos de [\[TANOMARU-94\]](#) (definição do nível de coordenação do SAA), [\[MATSUYAMA-97\]](#) (aplicação na área de automação bancária), [\[BECERRA-98\]](#) (aplicação na área de automação de projetos), [\[MARTE-00\]](#) (aplicação na área de automação de transportes), [\[ALMEIDA-00\]](#) (aplicação na área de automação de controle sanitário) e [\[FILHO-01\]](#) (aplicação na área de automação de energia elétrica).

O trabalho de [\[BECERRA-98\]](#) propõe uma metodologia de projeto de automação utilizando como referência o modelo SAA proposto em [\[MARTUCCI-92\]](#). O ponto central deste trabalho baseia-se na utilização do padrão ODP na definição de um sistema de informação aplicado na área de automação de projetos.

Estes subsídios criaram a necessidade de se propor uma infra-estrutura de interoperabilidade ou middleware no SAA. Esta infra-estrutura consiste na proposta do *middleware Maestro*, o qual implementa elementos definidos no trabalho de [\[MARTUCCI-92\]](#) e utiliza a metodologia de projeto baseada no padrão ODP definida no trabalho de [\[BECERRA-98\]](#).

1.8 Estrutura da Dissertação

Este trabalho de dissertação abrange basicamente a especificação de um *middleware* para suporte a aplicações distribuídas baseadas em componentes de *software*. A estrutura da dissertação está composta de cinco capítulos, cujos conteúdos são definidos a seguir:

- ? Primeiro capítulo: neste capítulo é fornecida uma visão geral do trabalho desenvolvido, auxiliando o leitor a entender, de maneira geral, o seu conteúdo;
- ? Segundo capítulo: neste capítulo é fornecida uma visão geral ao leitor com referência aos diversos conceitos abordados ao longo do desenvolvimento da dissertação, relacionados ao tema de sistemas distribuídos e abertos. Seu objetivo é fornecer as bases para que o leitor possa compreender o raciocínio utilizado para elaborar o tema principal da pesquisa;
- ? Terceiro capítulo: neste capítulo será desenvolvido o tema principal da dissertação, a especificação do *middleware Maestro*;
- ? Quarto capítulo: o objetivo deste capítulo é apresentar um estudo de caso, selecionado a partir de trabalhos de pesquisa anteriores, que valide a arquitetura proposta;
- ? Quinto capítulo: o objetivo deste capítulo é apresentar o fechamento do trabalho.

Capítulo 2

Middleware,

Componentes de Software e o
Modelo de Referência ODP

O presente capítulo tem por intuito fornecer o embasamento teórico necessário ao entendimento ou à complementação das informações fornecidas nos capítulos restantes desta dissertação. Cada item deste capítulo introduz, de forma concisa, determinado aspecto que se tornou relevante à dissertação, de forma que um conhecimento inicial possa ser oferecido ao leitor.

O primeiro item trata de apresentar o conceito de *middleware*, que é o tema principal desta dissertação. O segundo item apresenta os conceitos de componentes de *software* que também são parte integrante do trabalho. Finalmente o último item fornece uma introdução teórica ao modelo de referência ODP, o qual é utilizado exhaustivamente ao longo dos demais capítulos para a especificação do *middleware Maestro*.

2.1 *Middleware*

Atualmente os sistemas computacionais das empresas necessitam resolver uma questão crucial para a continuidade do desenvolvimento dos seus negócios: a portabilidade e a interoperabilidade entre os mesmos.

O cenário mais comum encontrado nos diversos ambientes de negócio atuais consiste numa variedade de sistemas de informação heterogêneos, tanto no lado dos clientes como no dos servidores. Estes sistemas possuem interfaces e sistemas operacionais diferentes e dependem de arquiteturas de rede distintas. Como resultado, a integração é difícil e onerosa para a empresa.

Esta questão vem sendo discutida no campo dos sistemas abertos, que busca soluções de consenso entre os fornecedores de sistemas de informação. Estas soluções teriam como premissa a utilização de interfaces de programação padronizadas (o que permitiria portar as aplicações para diversas plataformas), e seriam baseadas em protocolos também padronizados (o que possibilita a interoperabilidade entre os sistemas). Isto ofereceria aos consumidores uma certa independência com relação aos seus fornecedores.

Uma solução utilizada para resolver este tipo de problema é conhecida pelo nome de *middleware*. Este consiste em um sistema distribuído de serviços, com interfaces e protocolos padronizados, que busca solucionar problemas de heterogeneidade e interoperabilidade entre produtos de diferentes fabricantes. O *middleware* é genérico, suporta necessidades de uma grande variedade de aplicações e indústrias e é executado em múltiplas plataformas [BERNSTEIN-96].

Devido à importância da padronização de interfaces e de protocolos, diversos *middlewares* tem sofrido esforços para sua padronização, alguns através de organismos de padronização formais tal como a ISO, outros através de consórcios entre indústrias como X/Open, OSF e OMG e outros ainda através do patrocínio de companhias com grande participação no mercado de informática como o *Windows Open Services Architecture* (WOSA) da Microsoft.

Em geral é difícil se definir um *middleware* precisamente. Basicamente trata-se de um software de interoperabilidade que permite que aplicações e usuários interajam um com o outro através de uma rede. Outra definição seria:

? “*Middleware* é um conjunto de serviços, independentes de negócio, que permitem a interação entre aplicações e usuários finais através de uma rede de computadores. Em essência, *middleware* é o *software* que reside acima da rede e abaixo dos *softwares* de aplicações de negócio” [UMAR-97].

O autor define o *middleware* como:

? *Middleware* é um elemento facilitador que permite que aplicações de negócio comuniquem entre si de maneira transparente, independente de sua infraestrutura computacional (interfaces, protocolos de comunicação, sistemas operacionais ou plataformas de *hardware*). Em outras palavras, *middleware* é o elemento que possibilita a interoperabilidade entre estas aplicações.

Os *middlewares* provêem uma grande escala de funcionalidades que são categorizadas em

- ? Serviços de troca de informação;
- ? Serviços específicos de aplicação;
- ? Serviços de gerenciamento e suporte.

conforme ilustra a figura 2.1 [\[UMAR-97\]](#).

Figura 2.1 Uma visão funcional do *middleware*.

A seguir definimos cada um destes serviços: [\[UMAR-97\]](#)

Serviços de troca de informação enviam a informação necessária através da rede em resposta a uma requisição. Em particular, estes serviços podem ser necessários para transferência de dados, envio de comandos, recebimento de respostas, verificação de status, resolver *deadlocks*, estabelecer e terminar conexões entre processos de aplicações localizados remotamente e os usuários finais. Estes serviços incluem tipicamente chamadas síncronas e assíncronas como também facilidades para que o servidor possa manter várias requisições concorrentes. Os serviços de troca de informação são suportados pelo *middleware* através de serviços primitivos (por exemplo transferência de arquivos, emulação de terminais e correio eletrônico) e serviços cliente/servidor de troca de informação (por exemplo chamadas de procedimento remotos ou RPC).

Serviços específicos de aplicação provêm os serviços necessários a diferentes classes de aplicações. Por exemplo, estes serviços acessam e manipulam bancos de dados distribuídos que estão localizados em múltiplas localizações. Outro exemplo é o serviço de processamento de transações distribuídas que mantém a integridade de uma única ou múltiplas cópias de dados para processamento de transações. Ainda outros exemplos destes serviços são os serviços de replicação de dados, serviços de *groupware* para processamento colaborativo, serviços de *workflow*, serviços de objetos distribuídos e serviços especializados necessários para computação móvel e aplicações multimídia distribuídas.

Serviços de gerenciamento e suporte são necessários para localizar recursos distribuídos e administrar estes recursos através da rede. Exemplos destes serviços incluem serviços de diretório, de segurança, de falha e de desempenho.

Segundo [BERNSTEIN-96] e [VERNADAT-96], estas categorias poderiam ser ainda ampliadas, possibilitando os seguintes serviços de *middleware*:

- ✍ Gerenciamento de Apresentação: Gerenciamento de telas e de gráficos, ligação hipermídia e gerenciamento de impressão;
- ✍ Aplicação/Computação: Ordenação, serviços matemáticos, serviços de internacionalização (para manipulação de caracteres e palavras), conversores de dados e serviços de impressão.
- ✍ Gerenciamento de Informação: Serviços de diretório, gerenciamento de *log*, gerenciamento de arquivos distribuídos, gerenciamento de *records*, sistemas de banco de dados relacional e orientado a objetos, e gerenciamento de repositórios;
- ✍ Gerenciamento de Objetos: Serviços de comunicação com objetos remotos, persistência, transação, concorrência, integridade e segurança entre objetos;
- ✍ Comunicação: Mensagem ponto-a-ponto, chamada de procedimentos remotos, enfileiramento de mensagens, correio eletrônico, troca eletrônica de dados (EDI) e protocolos de comunicação;
- ✍ Controle: Gerenciamento de *threads*, gerenciamento de transações, *broker* de recursos, *scheduler* de requisições e de serviços, serviços de notificação de eventos e monitoração de processos;
- ✍ Computação/Programação Distribuída: Serviços de nomes, serviços de *trading* e *Object Request Brokers* (ORBs);
- ✍ Serviços de Horário: Serviços necessários para prover uma maneira padrão de representar a hora e sincronizar os relógios de diferentes computadores no caso de uma federação de sistemas heterogêneos e autônomos;

- ✍ Gerenciamento de Sistema: Serviço de contabilidade, gerenciamento de configuração, gerenciamento de instalação de *software*, detecção de falha, coordenação de recuperação, serviço de segurança (autenticação e autorização), serviço de auditoria, serviço de encriptação e de controle de acesso.

2.2 Componentes de Software

A decisão de se utilizar a tecnologia de componentes de *software* na construção do *middleware Maestro* e de suas aplicações está baseada na estratégia de padronização de interfaces, o que permite uma maior reutilização de seus elementos e uma redução na sua manutenção. Uma vez especificado o componente de *software*, é decisão do desenvolvedor optar pelo seu desenvolvimento ou pela aquisição no mercado. Sua substituição por outro componente de *software* é possível, desde que atenda à sua especificação. Mas como é definido um componente de *software*?

Uma definição clara e precisa de componente de *software* torna-se uma tarefa difícil devido a: diferentes posições tomadas pelos diversos fabricantes de infra-estruturas comerciais de componentes distribuídos; controvérsias existentes entre *software* orientado a objetos e baseado em componentes; e, em parte, à sua primeira implementação industrial através do OLE e do Visual Basic onde havia uma preocupação maior na solução de problemas reais de *software* do que na definição formal de um novo modelo computacional que atendesse a estas necessidades [\[KOZACZYNSKI-98\]](#).

Não obstante, algumas definições de componentes de *software* criadas pela indústria de *software* delineiam seus contornos apesar de não conseguirem caracterizá-lo totalmente [\[HAN-98\]](#). Vejamos algumas destas definições:

- ✍ “Um componente é uma parte não trivial, quase independente e substituível de um sistema que preenche uma função clara no contexto de uma arquitetura bem definida. Um componente conforma com e provê a

realização física de um grupo de interfaces” KRUTCHEN apud [\[BROWN-99\]](#).

- ✍ “Um componente de *software* em tempo de execução é um pacote dinâmico e aglutinável de um ou mais programas gerenciados como uma unidade e acessado através de interfaces documentadas que podem ser descobertas em tempo de execução” GARTNER GROUP apud [\[BROWN-99\]](#).
- ✍ “Um componente de software é uma unidade de composição apenas com interfaces contratualmente especificadas e dependências de contexto explícitas. Um componente de *software* pode ser instalado independentemente e é sujeito à composição por terceiros” [\[SZYPERSKY-98\]](#).
- ✍ “Um componente de negócio representa a implementação em *software* de um conceito “autônomo” de negócio ou processo de negócio. Ele consiste dos artefatos de *software* necessários para expressar, implementar e instalar o conceito como um elemento reutilizável de um sistema de negócio mais amplo” KOZACZYNSKI apud [\[BROWN-99\]](#).

Através destas definições verificamos que estas unidades autônomas, os componentes de *software*, são substituíveis e reutilizáveis, permitindo que um sistema ou aplicação seja construído através da sua composição, inclusive com componentes desenvolvidos por terceiros. Para atingir esta meta, é necessário que as interfaces que permitem o acesso às funcionalidades internas do componente sejam contratualmente especificadas, assim como sua dependência de contexto.

Partiremos então de características básicas dos componentes de *software* que são comuns a qualquer implementação dos mesmos e necessárias ao entendimento.

- ✍ “Um componente é uma parte reutilizável de *software* em forma binária que pode ser conectada a outros componentes de outros fabricantes com certa facilidade” [MICROSOFT-95].
- ✍ “Um componente de *software* tem de aderir a um padrão binário externo, mas sua implementação interna é completamente livre” [MICROSOFT-95].
- ✍ “Um componente não tem estado persistente¹” [SZYPERSKY-98].

Clientes de componentes acessam seus serviços através de interfaces externas, apresentados como *propriedades*, *métodos* e *eventos*. Sua utilização e o comportamento esperado do objeto devem ser definidos em contrato entre o cliente e o desenvolvedor.

Um componente de software possui propriedades não-funcionais que descrevem aspectos de segurança, desempenho e confiabilidade.

Com base nestas definições, utilizaremos a seguinte figura para ilustrar os conceitos mais importantes vistos, o componente e suas interfaces.

Figura 2.2 Ilustração de um componente que suporta três interfaces A, B e C.

¹ Isto significa que um componente não pode salvar seu estado internamente. Apesar desta restrição, outros mecanismos podem ser criados para salvar o estado de um componente (ver *Persistent Objects* e *Monikers* em [MICROSOFT-95])

A *interface* pode ser definida como uma coleção de operações relacionadas, cada qual com uma assinatura definida e possivelmente com o retorno de um tipo de dado. O número de interfaces por objeto pode variar, de uma a várias, dependendo do tipo de padrão de mercado utilizado.

A *localização* dos componentes num sistema se dá através de *identificadores* únicos que, em alguns casos, podem ser gerados através de um algoritmo que garanta sua unicidade. Introduzido pelo DCE (*Distributed Computing Environment*) como UUID (*Universally Unique Identifier*), este conceito é utilizado, com algumas variações, por alguns padrões de componentes do mercado.

A *comunicação* entre componentes através de diferentes processos, locais ou remotos, se dá através da criação de *stubs*¹ e é automatizada através de IDLs (*Interface Definition Languages*), conceito também introduzido pelo DCE e também utilizado com variações pelos demais padrões de componentes. Os *stubs* são criados por um *middleware* de comunicação, mais conhecido por *object request broker* ou ORB. Este possui uma série de funcionalidades no modelo distribuído de componentes, entre elas a de permitir que a comunicação entre componentes seja sempre feita como se estivesse sendo local (transparência da localização), cabendo aos pares de *stubs* a comunicação remota através do ORB. Isto é ilustrado na [figura 2.3](#) para o modelo DCOM da Microsoft, tendo como ORB o COM em processos locais e o DCOM em processos remotos.

Sumarizando, componentes de *software* são partes quase independentes e utilizadas na composição de sistemas, podendo serem substituídos por novas implementações desde que mantenha-se a especificação de suas interfaces, através das quais seus serviços são acessados. Sua identificação, localização, inicialização e comunicação são manipuladas através de um *middleware* de comunicação ou ORB.

¹ Algumas tecnologias de integração de componentes utilizam diferentes nomenclaturas para os *stubs* dos clientes e dos servidores. A tecnologia COM refere-se a estes como *proxy* e *stub* e a tecnologia CORBA como *stub* e *skeleton* respectivamente.

Figura 2.3 Modelo DCOM da Microsoft.

2.3 O Modelo de Referência ODP

A *International Standards Organization* (ISO) e a *International Telecommunications Union* (ITU), anteriormente conhecida como CCITT, iniciaram um esforço conjunto para definição do *Open Distributed Processing* (ODP). O ODP foi aprovado como um novo trabalho da ISO em 1987. O objetivo do ODP tem sido a criação de um modelo de referência, conhecido como RM-ODP, que irá integrar uma ampla gama dos futuros padrões de sistemas distribuídos e manter consistência entre tais sistemas [\[UMAR-97\]](#), [\[BECERRA-98\]](#).

As justificativas e motivações para se propor o modelo de referência ODP basearam-se na necessidade de uma estrutura para suportar as especificações de sistemas distribuídos e abertos e o desenvolvimento de padrões para sistemas distribuídos.

Os principais objetivos do padrão ODP são os de alcançar: [\[RAYMOND-95\]](#)

- ✍ A portabilidade de aplicações entre plataformas heterogêneas;
- ✍ A interoperabilidade entre sistemas ODP;
- ✍ A transparência de distribuição.

Baseado no último objetivo, resulta que um sistema ODP torna-se indicado para prover mecanismos que escondam de seus usuários e aplicações a complexidade interna dos seus componentes. Em particular, estes mecanismos são indicados para endereçar um conjunto de transparências fundamentais. Exemplos de propriedades de transparências chaves são as seguintes:

- ✍ Transparência de localização: esconde as diferenças entre invocações locais e remotas;
- ✍ Transparência de migração: extensão da transparência de localização mascarando a relocação dinâmica de objetos;

- ✍ Transparência de acesso: mascara diferenças entre mecanismos de acesso;
- ✍ Transparência de federação: faz fronteiras entre domínios administrativos federados;
- ✍ Transparência de falha: mascara a falha e possível recuperação dos objetos;
- ✍ Transparência de transação: mascara a coordenação de operações transacionais.

Para alcançar estes e outros objetivos de transparências, o RM-ODP inclui elementos descritivos bem como prescritivos. Os elementos descritivos provêm um vocabulário comum enquanto que os elementos prescritivos restringem o que pode ser construído como requerido por um sistema distribuído e aberto. Em particular, os elementos prescritivos baseiam-se nos cinco pontos de vista do RM-ODP ilustrados na [figura 2.4](#).

Figura 2.4 Pontos de vista do RM-ODP [\[UMAR-97\]](#).

O conceito de ponto de vista aparece com o objetivo de manipular as informações geradas na especificação dos sistemas distribuídos. Estas informações englobam vários aspectos de projeto como por exemplo: comunicação, transparência, segurança, entre outros [\[BECERRA-98\]](#).

Este problema foi uma das motivações do padrão ODP, e como consequência disto, foi proposta a definição do termo ponto de vista, a identificação dos cinco pontos de vista básicos e a estruturação de uma linguagem de especificação para cada um.

O conjunto de especificações referentes aos pontos de vista definidos pelo padrão ODP podem ser utilizados de diferentes formas no desenvolvimento de projetos de sistemas distribuídos. Não existe uma estratégia formal de utilização e, portanto, recomenda-se que esta deva se adequar ao tipo de sistema que se pretende projetar SINDEREN apud [\[BECERRA-98\]](#).

A especificação de cada ponto de vista é constituída de informações em linguagem natural (texto) e diagramas orientados a objetos.

A seguir veremos a descrição de cada ponto de vista do RM-ODP.

2.3.1 O Ponto de Vista da Empresa

O ponto de vista da empresa provê o meio de se capturar requisitos assim como a estrutura e comportamento de uma comunidade em seu ambiente [\[PUTMAN-01\]](#).

Neste ponto de vista são definidos o objetivo e escopo da empresa, as políticas que definem a atuação do sistema e as políticas que definem o inter-relacionamento da empresa com entidades externas.

A estrutura básica do ponto de vista da empresa contém os seguintes aspectos: [\[BECERRA-98\]](#), [\[RAYMOND-95\]](#), [\[PUTMAN-01\]](#)

- ✍ Escopo da empresa: contém as características principais da empresa, enxergando esta como uma entidade de negócios;
- ✍ Funções empresariais: funções realizadas pelos elementos ou objetos dentro da empresa;
- ✍ Políticas: são regras definidas na empresa, internamente ou por agentes externos a ela, as quais estão normalmente constituídas de obrigações, permissões e proibições;
- ✍ Procedimentos: contém um conjunto de ações específicas realizadas pelos objetos empresa;
- ✍ Comunidades: agrupamento de objetos com um propósito comum;
- ✍ Federação: agrupamento de objetos pertencentes a diferentes domínios, sejam estes tecnológicos ou empresariais, e com um propósito comum;
- ✍ Contrato: um acordo entre as partes.

Ao se preparar uma especificação do ponto de vista empresa de uma aplicação ODP, políticas são determinadas pela organização ao invés de serem impostas à organização pelas opções de tecnologia [\[RAYMOND-95\]](#).

2.3.2 O Ponto de Vista da Informação

O ponto de vista da informação tem como foco a semântica das informações, através da definição de: objetos informação; esquemas invariante, estático e dinâmico; e regras associadas.

Os três esquemas utilizados pelo ODP capturam a semântica, a estrutura e o relacionamento junto ao sistema: o metadado. O metadado é uma informação extra sobre o sistema que se relaciona com a existência, as restrições e as regras para uso dos objetos. Ele inclui o tipo, os estados e mudanças de estado permitidos e as regras que dirigem o comportamento do objeto através da sua existência. Obrigações de uma política são capturadas como restrições no esquema invariante e ações que causem uma mudança no estado de um objeto são capturadas no esquema dinâmico [\[PUTMAN-01\]](#).

Em síntese, os três esquemas utilizados pelo ODP podem ser descritos como:

- ✍ Esquema invariante: restringe os estados e a estrutura de um objeto informação ao longo de sua existência;
- ✍ Esquema estático: captura os estados e a estrutura de um objeto informação em ponto particular no tempo;
- ✍ Esquema dinâmico: define as mudanças permitidas nos estados e na estrutura de um objeto.

2.3.3 O Ponto de Vista da Computação

O ponto de vista da computação é utilizado para especificar as funcionalidades de um sistema ODP através da definição de uma estrutura de objetos distribuídos que interagem através de interfaces específicas.

Neste ponto de vista existe a preocupação com a distribuição dos objetos dentro do sistema, mas não com a sua infra-estrutura de comunicação.

A linguagem de especificação deste ponto de vista utiliza um modelo de objeto que define: [\[ISO-98\]](#)

- ✍ A forma de interface que um objeto pode ter;
- ✍ A maneira pelas quais as interfaces podem se conectar e as formas de interação que podem existir entre elas;
- ✍ As ações que um objeto pode executar, em particular a criação de novos objetos e interfaces, e o estabelecimento de conexões.

Existem três formas de interfaces para os objetos computação: [\[BECERRA-98\]](#)

- ✍ Interface de fluxo: suporta interações denominadas de fluxos que representam fluxos contínuos de informação tal como o fluxo de áudio e vídeo numa aplicação multimídia;
- ✍ Interface de operação: suporta interações denominadas operações que são estruturas semelhantes aos procedimentos ou rotinas utilizados em programação de sistemas;
- ✍ Interface de sinal: suporta interações denominadas de sinais que representam invocações tais como eventos e interrupções.

A conexão entre estas interfaces pode se dar de duas maneiras:

- ✍ Conexão explícita: configurada entre interfaces de operação ou de fluxo;
- ✍ Conexão implícita: configurada entre interfaces de operação, especialmente onde não se precisa explicitar a ação de conexão.

Na conexão explícita, interações complexas entre objetos, envolvendo interfaces do mesmo ou de diferentes tipos, são realizadas através de *objetos conexão* [ISO-98]. Este objeto possibilita também a especificação de operações de ligação multiponto ou compostas entre objetos computação, ou ainda operações de controle da qualidade do serviço (QoS) de fluxos contínuos de informação. Através de uma interface de controle, o objeto conexão pode ser informado dos valores limites de QoS que este deverá controlar.

A [figura 2.5](#) ilustra, através de um exemplo, os conceitos ODP apresentados no ponto de vista da computação. Esta figura mostra um sistema de comunicação multimídia, cuja conexão é realizada através de um objeto conexão. Os objetos multimídia do sistema podem comunicar-se entre si através de interfaces de fluxo ou de sinal. Através de sua interface de operação, eles controlam a QoS do fluxo de dados via objeto conexão.

Figura 2.5 Sistema de comunicação multimídia que ilustra os conceitos ODP do ponto de vista da computação.

2.3.4 O Ponto de Vista da Engenharia

O ponto de vista da engenharia tem como preocupação a maneira pela qual a interação entre objetos ocorre e os recursos necessários para que isso se torne possível. Conceitos são definidos para descrever: a infra-estrutura requerida para suportar interações distribuídas entre objetos; as regras para estruturação de sistemas com o propósito de gerenciar recursos; e as regras para estruturação de canais de comunicação entre objetos. [\[ISO-98\]](#)

Existe uma relação direta entre o ponto de vista da computação e o ponto de vista da engenharia: os objetos computação são vistos no ponto de vista da engenharia como *objetos básicos de engenharia* e a conexão entre objetos computação são vistas como canais de comunicação ou conexões locais.

Na estrutura de gerenciamento definida para este ponto de vista, os seguintes conceitos são utilizados: [\[PUTMAN-01\]](#), [\[BECERRA-98\]](#)

- ✍ Grupo (*cluster*): um agrupamento de objetos básicos de engenharia que são tratados como uma unidade, de forma a serem manipulados em conjunto por uma ou mais operações;
- ✍ Cápsula: um agrupamento de objetos engenharia que são tratados como uma unidade de processamento, de maneira semelhante aos módulos de software;
- ✍ Núcleo: encarregado da coordenação das funções de processamento, armazenamento e comunicação, sendo similar ao sistema operacional;
- ✍ Nó: composto por um processador de informação onde são executadas as funções de processamento, armazenamento e comunicação, de maneira similar ao computador e seu sistema operacional.

Na estrutura do canal de comunicação definida para este ponto de vista, os seguintes conceitos são definidos:

- ✍ Adaptador: resolve conflitos quanto ao formato de representação dos dados nos diversos nós envolvidos (*marshalling* e *unmarshalling* de dados);
- ✍ Conector: mantém a integridade ponto-a-ponto do canal, resolvendo problemas de distribuição;
- ✍ Protocolo: provê a comunicação entre os objetos do tipo *conector* servidos, com a qualidade e confiabilidade requeridas;
- ✍ Interceptor: é utilizado para compatibilizar requerimentos técnicos existentes entre dois nós como a conversão de protocolos, o controle de acesso, a compressão dos dados ou a utilização de mecanismos de criptografia.

A [figura 2.6](#) ilustra, através de um exemplo, os conceitos ODP apresentados no ponto de vista da engenharia. Esta figura mostra um sistema cliente/servidor representado através de seus respectivos nós e núcleos. Os objetos básicos de engenharia deste sistema estão agrupados em grupos e cápsulas e se comunicam através de conexões locais ou de canais de comunicação. Existem três canais de comunicação representados nesta figura, um para cada tipo de interface suportada pelo sistema (fluxo, operação e sinal). Cada canal de comunicação é estruturado através de objetos adaptador, conector, protocolo e interceptor.

Figura 2.6 Sistema cliente/servidor que ilustra os conceitos ODP do ponto de vista da engenharia.

2.3.5 O Ponto de Vista da Tecnologia

O ponto de vista da tecnologia preocupa-se com a seleção da tecnologia necessária para implementar o sistema ODP especificado. Assim, deve-se identificar os recursos de *hardware* e de *software* que serão utilizados na implementação, podendo estes serem proprietários, padronizados ou ambos e motivados por critérios de custo e disponibilidade [\[BECERRA-98\]](#).

O RM-ODP provê também quatro pontos de referência em cada um dos pontos de vista para testar a implementação do sistema com relação à sua especificação. Ou seja, estes pontos de referência são pontos de teste de conformidade [\[PUTMAN-01\]](#).

Capítulo 3

Especificação e Arquitetura do
Middleware Maestro

Mudanças fundamentais estão ocorrendo atualmente na natureza e na aplicação da tecnologia de informação nas empresas. Isto se deve em grande parte às alterações nos panoramas político, econômico e social mundiais ocorridas nas últimas décadas, além do surgimento de novos modelos empresariais e de novas tecnologias. Estas alterações provocaram uma mudança no ambiente de negócio das empresas onde a competitividade entre elas alcança níveis nunca antes experimentados [\[TAPSCOTT-93\]](#).

Para manterem-se neste novo ambiente, as empresas tradicionais necessitam reformular constantemente sua estrutura de negócios para poderem competir com as demais. As alterações necessárias à evolução de seus negócios, porém, devem ser realizadas de maneira gradativa e calculada, aproveitando principalmente seus sistemas de informação críticos legados. Em outras palavras, deve haver uma harmonia entre o modelo atual de informação e o futuro durante o período de transição.

Novos modelos de gestão de negócio fundamentados, em grande parte, em novos paradigmas da tecnologia da informação têm demonstrado uma grande eficácia na criação de modelos de negócio mais competitivos para novas empresas ou para a reengenharia de empresas tradicionais [\[DAVIS-99\]](#).

Tendo em vista estas necessidades atuais básicas das empresas, a proposta aqui analisada busca a especificação de um sistema de informação cujo principal objetivo consiste em possibilitar a integração dos atuais e dos novos sistemas de informação das empresas de maneira a torná-los um conjunto coeso, flexível e altamente produtivo [\[BECERRA-98\]](#). Este sistema de informação será referenciado ao longo do texto pelo nome *Maestro*.

O *middleware Maestro* é baseado em um novo campo de desenvolvimento de aplicações distribuídas que vem surgindo graças às transparências possibilitadas pelos diversos sistemas de *middleware* existentes [\[FERREIRA-00a\]](#), [\[FERREIRA-00b\]](#), [\[FERREIRA-00c\]](#). É uma área que tem obtido muito interesse do ambiente acadêmico e das empresas e corresponde ao que se

passou a chamar, em inglês, *Ubiquitous Computing* [WEISER-91], [WEISER-93a], [WEISER-93b] ou ainda *Pervasive Computing* [ARK-99]. Estes dois termos, muitas vezes tratados analogamente, descrevem um novo cenário no campo da computação onde cada vez mais dispositivos heterogêneos são integrados em rede permitindo o acesso transparente do usuário aos recursos computacionais disponíveis no seu universo.

A [figura 3.1](#) ilustra um cenário destes, consistindo de dispositivos utilizados no lazer, no trabalho ou em pesquisa. Todos estes dispositivos necessitam de acesso transparente a serviços disponíveis numa rede global de computadores. Cada dispositivo, porém, possui características próprias de interface com o usuário e velocidade de comunicação, como pode ser visto nesta figura.

O círculo central, na figura, é o responsável por se comunicar com os círculos externos, que identificam as características de cada dispositivo, estabelecendo assim a primeira camada de transparência de acesso do *Maestro*. A segunda camada de transparência é estabelecida com o círculo mais interno na figura, responsável pelo acesso transparente a serviços distribuídos numa rede heterogênea de computadores. Estes três círculos identificados na figura (de fora para dentro) são chamados de *Consumidor*, *Provedor* e *Fornecedor* na arquitetura do *Maestro* e serão vistos em mais detalhes quando estivermos analisando as visões do ODP.

Nesta figura, um exemplo de aplicação é a de um assistente digital com comunicação sem fio que acessa, de maneira assíncrona, as informações disponíveis na rede. A interface com o usuário, neste exemplo, é separada da lógica de negócios da aplicação permitindo que esta se torne dinâmica, ajustando-se ao dispositivo utilizado no acesso ao serviço [EUSTICE-99].

Legenda:

Fibra Óptica

Cabo Coaxial

Rede sem fio

Rede Local

Satélite

Rede Telefônica

Figura 3.1 Cenários dos atuais sistemas distribuídos baseado na arquitetura do middleware Maestro.

A [figura 3.2](#) ilustra o uso do *Maestro* utilizando um telefone celular e um computador como exemplos. Como as interfaces com o usuário são diferentes para cada dispositivo, a camada *Provedor* seleciona a IHM (Interface Homem-Máquina) mais apropriada, com base na informação fornecida pela camada *Consumidor* de cada dispositivo. Uma vez selecionada e executada a IHM obtida do repositório, o dispositivo está pronto para acessar informação da lógica de negócios da aplicação. Esta, por sua vez, é a mesma para ambos dispositivos e acessa serviços, comuns ou específicos de negócio e dispersos na rede, através da camada *Fornecedor*.

Na especificação do *Maestro* são utilizados dois padrões: o RM-ODP (*Reference Model of Open Distributed Processing*) [\[ISO-98\]](#) da ISO (*International Organization for Standardization*) e o UML (*Universal Modeling Language*) da OMG (*Object Management Group*), onde o último servirá de complemento ao primeiro para a definição dos diversos conceitos ODP utilizados na modelagem do sistema *Maestro* [\[MCI-97\]](#), [\[RATIONAL-97\]](#).

O modelo de referência ODP possui cinco visões: a Visão da Empresa, a Visão da Informação, a Visão da Computação, a Visão da Engenharia e a Visão da Tecnologia [\[ISO-96b\]](#). O UML, por sua vez, possui também cinco visões: a Visão de Caso de Uso, a Visão de Projeto, a Visão de Processo, a Visão de Implementação e a Visão de Implantação [\[BOOCH-99\]](#).

Cada visão do UML está relacionada a um determinado diagrama enquanto que as visões do ODP não definem um diagrama específico nem tampouco um padrão a ser utilizado. Assim, relacionaremos as duas visões de forma a padronizarmos o uso dos diagramas no trabalho¹.

¹ Isto não significa que esta seja uma regra adotada pela ISO ou pela OMG para utilização dos dois padrões. Este relacionamento é feito aqui apenas para trabalharmos de maneira organizada, na falta de uma ferramenta para o assunto.

Figura 3.2 Mecanismo de execução de uma aplicação distribuída baseada na arquitetura do middleware Maestro.

No UML, os diagramas de caso de uso, de classes, de objeto, de sequência e de colaboração poderão aparecer na fase de análise (visões empresa, informação e computação); os diagramas de componentes e de implantação na fase de projeto (visões engenharia e tecnologia); e os diagramas de pacotes, de estados e de atividade em ambas as fases.

No item 3.1 será vista a especificação do *Maestro* segundo os cinco pontos de vista do ODP, o que servirá de subsídio para a especificação de sua arquitetura no item 3.2.

3.1 Especificação do *Middleware Maestro* Utilizando o RM-ODP

A principal função de um *middleware* é a de oferecer serviços de maneira transparente aos usuários e aplicações, abstraindo os detalhes de distribuição e heterogeneidade do sistema. Ele pode ser definido como sendo um *software* de interoperabilidade que permite que aplicações e usuários interajam entre si em um ambiente computacional distribuído [UMAR-97].

Pode-se citar como requisitos gerais atribuídos ao *middleware Maestro* os seguintes aspectos: [UMAR-97], [BERNSTEIN-96].

- ✍ Permitir a integração de sistemas legados;
- ✍ Facilitar a portabilidade e a interoperabilidade;
- ✍ Aumentar a confiabilidade;
- ✍ Permitir o desenvolvimento de novas aplicações baseadas em tecnologias de componentes de *software*;

- ✍ Prover uma infra-estrutura de desenvolvimento de componentes de *software* visando a redução do tempo, esforço e gastos necessários ao desenvolvimento e distribuição de aplicações baseadas nesta tecnologia;

Estas são as principais metas do *middleware Maestro*, que busca atingir estes objetivos através da composição de serviços utilizados pelas aplicações de negócio, de maneira a integrá-los num ambiente abrangente e de fácil utilização.

O *middleware Maestro* será especificado no item seguinte segundo os cinco pontos de vista do ODP (empresa, informação, computação, engenharia e tecnologia). Em cada ponto de vista serão analisados os requisitos do *Maestro* relevantes ao nível de abstração especificado pelo ODP, levando em consideração os conceitos que o modelo de referência utiliza em cada visão do sistema [ISO-96a]. Com base nesta especificação obtida, será então definida a arquitetura para o sistema analisado.

3.1.1 O Ponto de Vista da Empresa

Para o ponto de vista da empresa, o ODP define a análise do sistema e de seu ambiente de maneira a determinar seu propósito, escopo e políticas de uso.

Neste ponto de vista, um aspecto importante com relação ao ambiente de atuação do *Maestro* consiste na integração dos diversos processos do negócio das empresas, sejam estes novos ou legados. Estes processos de negócio irão compor serviços, os quais serão utilizados pelos usuários da empresa através de aplicações de negócio.

As políticas de acesso, de uso e de qualidade do serviço devem ser acordadas entre empresa e usuário de maneira a definir um contrato estabelecendo as responsabilidades de cada parte na prestação do serviço.

Três entidades, o *Consumidor*, o *Provedor* e o *Fornecedor*, constituem o *middleware Maestro* e são definidas como objetos empresa deste sistema segundo este ponto de vista do ODP. Estas devem, em conjunto, disponibilizar a aplicação da empresa para seus usuários finais obedecendo aos seguintes critérios:

- ✍ O objeto *Consumidor* deverá interagir com o usuário da aplicação de negócio da empresa. A qualidade do serviço associada à aplicação poderá ser negociada através deste objeto, de acordo com os recursos disponíveis no processador de informação utilizado pelo usuário;
- ✍ O objeto *Provedor* deverá coordenar o acesso dos usuários às aplicações da empresa devendo garantir as políticas de segurança e a qualidade do serviço acordada entre as partes, bem como realizar o registro dos acessos para fins de tarifação ou auditoria;
- ✍ O objeto *Fornecedor* deverá oferecer serviços específicos dos negócios das empresas às aplicações de negócio mantidas pelo objeto *Provedor*. Estes serviços serão disponibilizados por este objeto através da integração dos diversos processos de negócio considerados, sejam estes legados ou não. Se este objeto pertencer a um domínio administrativo diferente daquele do objeto *Provedor*, então ele também deverá garantir as políticas administrativas de seu domínio conforme as estabelecidas para o último.

Esta divisão de responsabilidades entre os três objetos empresa do *middleware Maestro* permite atender aos objetivos definidos para este ponto de vista do ODP, ou seja, o de integração dos diversos processos do negócio das empresas. Como estes objetos podem ser autônomos e pertencerem a domínios administrativos distintos, foram criadas, segundo os conceitos do ODP, uma *comunidade* de usuários, outra de provedores e uma *federação* de fornecedores [ISO-98].

A [figura 3.3](#) ilustra a especificação do *Maestro* do ponto de vista ODP da empresa através de um diagrama de classes do UML. Os usuários administrativos representados neste diagrama deverão manter as políticas da empresa para seu domínio definidas através de *contratos* [\[ISO-98\]](#).

3.1.2 O Ponto de Vista da Informação

Para o ponto de vista da informação, o ODP define a análise do sistema e de seu ambiente de maneira a determinar a semântica da informação e a manipulação realizada com relação a ela.

Neste ponto de vista, analisaremos aquelas informações manipuladas pelo *Maestro* que são trocadas com seu ambiente e processadas internamente.

Os usuários e suas aplicações de negócio trocam informações entre si, através do *Maestro*, de forma a atingirem um determinado objetivo comum. A aplicação, por sua vez, pode trocar informações com outras entidades, também através do *Maestro*, de forma a obter recursos de informação necessários ao seu processamento.

Com relação a esta informação processada pelo *Maestro*, existem quatro categorias a serem consideradas: a primeira trata de informações de contrato que garantem as políticas da empresa consideradas na visão empresa; a segunda trata da informação de registro de acessos ao sistema para fins de tarifação ou auditoria; a terceira refere-se à informação do domínio tecnológico dos usuários devendo identificar as características de seus processadores de informação para a adequação da qualidade do serviço oferecida pela aplicação às características do equipamento¹; e a quarta refere-se à informação que descreve a IHM (interface homem-máquina) da aplicação, determinada com base na informação da categoria anterior.

¹ A qualidade do serviço associada à aplicação é restringida pelas características do equipamento utilizado pelo usuário de acordo com seus recursos disponíveis tais como memória, velocidade de comunicação, recursos visuais, etc.

Figura 3.3 Especificação do ponto de vista ODP da empresa para o middleware *Maestro* utilizando diagrama de classes do UML. *Maestro*.

Estas categorias de informação são processadas pelos seguintes objetos informação do *Maestro* segundo este ponto de vista do ODP:

- ✍ O objeto *Despachante* manipula informações da primeira e da segunda categoria devendo registrar, efetuar e garantir o acesso ao *middleware Maestro* apenas por usuários autorizados e também verificar se a qualidade do serviço prestado¹ por ele está de acordo com a estipulada. O acesso deverá ser garantido aos usuários cujas credenciais estiverem armazenadas neste objeto e a qualidade do serviço deverá ser comparada aos valores limites também armazenados neste objeto. O usuário de cada domínio administrativo deverá manter atualizado este objeto;
- ✍ O objeto *Identificador* manipula informações da terceira categoria devendo identificar as características dos processadores de informação utilizados pelos usuários para acesso ao sistema *Maestro*. Esta informação poderá ser obtida de forma automática, através de verificações realizadas no dispositivo por este objeto;
- ✍ Os objetos *Descritor* e *Apresentador* processam informações da quarta categoria. O primeiro objeto mantém a informação que descreve a IHM da aplicação requisitada pelo usuário, podendo manipular esta antes de enviá-la para poder adequá-la às características do processador de informação utilizado para o acesso. O segundo objeto recebe a informação proveniente do primeiro, manipulando-a para a correta apresentação ao usuário requisitante. O usuário administrativo de cada domínio deverá manter atualizadas as informações armazenadas no objeto *Descritor*.

A [figura 3.4](#) ilustra a especificação do *Maestro* do ponto de vista ODP da informação através de um diagrama de classes do UML.

¹ A qualidade do serviço prestado poderá ser determinada a partir de informações que estipulem a quantidade máxima de falhas, interrupções e atrasos no fornecimento do serviço.

Figura 3.4 Especificação do ponto de vista ODP da informação para o middleware *Maestro* utilizando diagrama de classes do UML.

Os agrupamentos através de pacotes foram utilizados para mostrar a decomposição de alguns objetos empresa em objetos informação ilustrando assim a correspondência desta visão com a anterior.

Com relação aos conceitos ODP para a visão informação, o *esquema invariante* pode ser definido como sendo a estrutura comum a cada contrato entre os diferentes domínios administrativos do *Maestro*, conforme apresentado na [figura 3.3](#) através dos objetos-associação do UML. Conforme definido anteriormente, os contratos da visão empresa do *Maestro* tem suas informações armazenadas nos objetos Despachante desta visão e devem possuir uma estrutura comum que define os predicados que deverão restringir os possíveis estados dos objetos do sistema durante sua operação.

O *esquema estático* é definido como sendo o contrato de ligação (*binding contract*) específico que existe entre os elementos dos diferentes domínios administrativos do *Maestro* em um determinado ponto no tempo. Este conceito pode ser representado no UML através de um ou mais diagramas de objeto que indiquem quais objetos estão ativos em determinado ponto do tempo e suas respectivas ligações.

O *esquema dinâmico* é definido como sendo as regras de interação entre os objetos informação do *Maestro* definindo sua correta sequência no tempo [\[ISO-98\]](#). Este conceito é representado através do diagrama de colaboração do UML da [figura 3.5](#).

Nesta figura é mostrado como uma aplicação remota é executada utilizando o suporte do *middleware Maestro*. Este determina a localização e ativação das aplicações e serviços remotos necessários à execução das transações requisitadas pelo usuário. Para tanto, os acessos e requisições são registrados em cada domínio para fins de tarifação, e é feita também a verificação das permissões de acesso a cada um deles.

Figura 3.5 Diagrama de colaboração do processamento de uma aplicação remota no middleware *Maestro*.

3.1.3 O Ponto de Vista da Computação

Para o ponto de vista da computação, o ODP define a análise do sistema e de seu ambiente de maneira a disponibilizar a distribuição através da decomposição funcional do sistema em objetos computação que interagem através de interfaces.

A interação entre estes objetos é essencialmente assíncrona e pode ocorrer de três forma distintas: através de uma interface de fluxo, através de uma interface de operações ou através de uma interface de sinais [ISO-98].

A [figura 3.6](#) ilustra a especificação do *Maestro* do ponto de vista ODP da computação através de um diagrama de classes do UML onde as classes e pacotes sombreados indicam aqueles objetos computação que pertencem ao *Maestro*.

Os agrupamentos através de pacotes foram utilizados para mostrar a decomposição dos objetos informação do tipo *Despachante* em objetos computação ilustrando assim a correspondência desta visão com a anterior.

O objeto computação do tipo *Comunicador* deste diagrama implementa as funcionalidades de segurança¹ do objeto informação *Despachante* da visão anterior e informa os valores limites de qualidade do serviço armazenados neste objeto para o objeto de conexão através de sua interface de controle, o que é ilustrado no diagrama através das relações de dependência.

O objeto computação denominado *Gerenciador de Eventos* deverá processar os eventos¹ informados pelos demais objetos computação de cada domínio administrativo do sistema de maneira a determinar uma ação correspondente a ele.

¹ A implementação destas funcionalidades pode ser feita utilizando as funções correspondentes do ODP, o que permite a padronização das mesmas.

Este objeto deverá solucionar problemas de comunicação entre entidades onde a mudança de estado de uma deve ser notificada às outras, permitindo assim o desacoplamento da sua comunicação. Os eventos providos pelos objetos computação de cada domínio são enviados a este objeto o qual se encarrega de notificar aqueles objetos computação que se inscreveram para saber da ocorrência de determinado evento específico [BACON-00].

3.1.4 O Ponto de Vista da Engenharia

O ODP define, para o ponto de vista da engenharia, a análise dos mecanismos pelos quais os *objetos básicos de engenharia* (correspondentes aos objetos computação da visão anterior) interagem entre si e os recursos necessários para isso, descrevendo assim a infra-estrutura requerida para o suporte à distribuição transparente dos mesmos [ISO-98].

A [figura 3.7](#) ilustra a especificação do *Maestro* do ponto de vista ODP da engenharia através de um diagrama de distribuição do UML onde os componentes e pacotes sombreados indicam aqueles objetos engenharia que pertencem ao *Maestro*. O agrupamento através de pacote foi utilizado para mostrar a decomposição do objeto computação do tipo *Conexão* em objetos engenharia do canal ODP ilustrando assim a correspondência desta visão com a anterior [ISO-98].

No *Maestro* são identificados três *nós*: *Consumidor*, *Provedor* e *Fornecedor*. Cada um destes contém seu processador e seu sistema operacional individual representado através de seu *núcleo*, o qual é responsável pelo gerenciamento dos recursos locais de cada sistema [ISO-98].

Figura 3.7 Especificação do ponto de vista ODP da engenharia para o middleware *Maestro* utilizando diagrama de implantação do UML.

Neste ponto de vista são também definidos os objetos engenharia que implementam as funções do ODP [ISO-98]. Estas funções do ODP são implementadas no *Maestro* através do objeto engenharia de *Serviços ODP*.

3.1.5 O Ponto de Vista da Tecnologia

O ODP define, para o ponto de vista da tecnologia, a especificação da implementação do sistema em termos de objetos tecnologia que representam os componentes de *hardware* e *software* selecionados a partir de critérios de custo e disponibilidade.

Com referência à disponibilidade da tecnologia, atualmente existem três padrões dominantes na indústria para implementação e comunicação entre objetos distribuídos: CORBA da OMG (padrão *de jure*), ActiveX/COM+/DCOM da Microsoft (padrão *de facto*) e JavaBeans/Enterprise JavaBeans da Sun Microsystems (padrão *de facto*). Estes três padrões implementam os canais de comunicação entre os diferentes nós da visão engenharia, os quais serão chamados de *object request brokers* (ORBs) na visão tecnologia. Não havendo uma compatibilidade na comunicação entre os ORBs dos diferentes padrões mencionados, há a necessidade de se utilizar conversores (*gateways*) para realizar a conversão de formatos.

Os objetos engenharia do tipo *Gerenciador de Eventos*, inicialmente definidos na visão computação do ODP, poderão ser implementados utilizando-se de recursos de objetos interpretadores, capazes de executar programas interpretados escritos em linguagem *script* que determinem a operação deste componente em determinados contextos de utilização. Existem vários produtos no mercado que capazes de executar linguagens interpretadas tais como JavaScript, JScript, VBScript, CORBA Script e Tcl/Tk [SCHNEIDER-98].

Outra preocupação da visão tecnologia está na compatibilização de sistemas legados das empresas com as novas tecnologias baseadas em objetos

[[SUPERNOVA-97](#)]. No *Maestro*, sistemas legados podem ser agentes de *software* deste sistema ou fornecer serviços específicos a ele. Atualmente as técnicas utilizadas para permitir a comunicação entre estes sistemas utilizam o conceito de *empacotadores* (*wrappers*) que realizam a correta conversão de métodos e dados de uma tecnologia para a outra.

A [figura 3.8](#) ilustra uma especificação do *Maestro* no segmento de automação de projetos, do ponto de vista ODP da tecnologia, através de um diagrama de distribuição do UML. As tecnologias utilizadas na sua implementação estão representadas na forma de componentes de *software* [[SZYPERSKI-98](#)].

3.2 Arquitetura do *Middleware Maestro*

A análise ODP realizada anteriormente é de extrema importância na especificação da arquitetura uma vez que ela define as funcionalidades necessárias à arquitetura *Maestro* para que esta possa atender aos diversos requisitos do sistema analisados em cada um dos cinco pontos de vista definidos pelo modelo de referência ODP. Segundo o ODP, atender aos requisitos exigidos pelo modelo é condição para que a arquitetura do sistema seja considerada aberta [[ISO-98](#)].

A arquitetura do *middleware Maestro* é baseada num modelo heterogêneo de arquitetura que envolve uma combinação de diversos estilos arquiteturais [[SHAW-96](#)]. Cada estilo adotado procura solucionar um determinado requisito do sistema procurando torná-lo adaptável ao ambiente dinâmico no qual deverá operar.

Todas estas características permitem que a arquitetura *Maestro* possa ser utilizada numa ampla gama de contextos, devido à sua flexibilidade de configuração.

Figura 3.8 Especificação do ponto de vista ODP da tecnologia para o middleware *Maestro* utilizando diagrama de implantação do UML.

A [figura 3.9](#) ilustra a arquitetura do *middleware Maestro* identificando os domínios de cada sistema integrante (*Consumidor*, *Provedor* e *Fornecedor*). Entre cada domínio existe uma camada de comunicação que se utiliza de *Canais de Comunicação* para estabelecer uma comunicação transparente entre os objetos de cada domínio.

Em cada domínio é implementada uma camada de segurança onde seus objetos devem garantir o acesso seguro a seus recursos.

Numa camada de controle são implementados os objetos que coordenam o comportamento da aplicação e do sistema *Maestro* de acordo com os eventos recebidos por estes objetos. Estes tomam uma ação ou simplesmente notificam aqueles objetos interessados na ocorrência de determinado evento.

Nas camadas de mais alto nível estão os serviços de apresentação da IHM da aplicação (Apresentador, Identificador e Descritor) e de compatibilização com serviços de sistemas legados e ainda as funções ODP de acesso, persistência, (re)localização, migração, falha, transação e replicação.

Alguns padrões (*patterns*) de arquitetura também podem ser definidos para o *middleware Maestro*, especificando assim a arquitetura de cada elemento do metamodelo apresentado na [figura 3.9](#). Alguns exemplos ilustrados nesta figura são o *MVC* (*Model-View-Controller*) ou o *PAC* (*Presentation-Abstraction-Control*) para o modelo de interação com o usuário e o *Broker* para a estruturação dos canais de comunicação [\[BUSCHMANN-96\]](#), [\[OMG-99\]](#).

Figura 3.9 Arquitetura do middleware *Maestro*.

Capítulo 4

Um Estudo de Caso em Automação
do Gerenciamento de Projetos de
Desenvolvimento de Software

O objetivo deste capítulo é experimentar a arquitetura do *middleware Maestro* utilizando uma aplicação na área de gerenciamento de projetos de desenvolvimento de *software* [BECERRA-98].

Inicialmente será fornecida uma apresentação sobre a área de gerenciamento de projetos (em particular a de desenvolvimento de *software*) e da área de qualidade de *software* para identificar o escopo e relacionamento com o experimento.

Em seguida será feita uma descrição geral do projeto intitulado *Sistema de Automação de Projetos de Software* (SAPS), a sua especificação utilizando os cinco pontos de vista do ODP e a definição da sua arquitetura baseada no modelo SAA (Sistema Aberto de Automação) [MARTUCCI-92]. Esta arquitetura baseada no SAA será então integrada à arquitetura do *middleware Maestro* identificando os elementos que pertencem a cada sistema.

A [figura 4.1](#) ilustra a estratégia a ser realizada para o desenvolvimento do experimento. Observa-se nesta figura que em cada ponto de vista do ODP para o sistema SAPS é analisada a conformidade com o ponto de vista ODP correspondente do *middleware Maestro*. Desta forma, toda a especificação desenvolvida ao longo do capítulo 3 é reutilizada para a especificação do sistema SAPS.

4.1 O Gerenciamento de Projetos

As atividades de gerenciamento de projetos tem por interesse manter o escopo, o cronograma e os recursos utilizados de forma estável. Geralmente isto significa planejar, organizar e controlar as tarefas de projeto de forma a identificar e alocar num cronograma os recursos que as executam [WHITTEN-95], [MICROSOFT-99].

Figura 4.1 Estratégia a ser realizada para o desenvolvimento do experimento.

Todos os setores que trabalham com desenvolvimento de produto ou prestação de serviço utilizam-se desta área. Este trabalho focaliza o setor particular de desenvolvimento de *software* baseado em componentes para o qual o sistema SAPS será analisado, projetado e desenvolvido.

Quando o objetivo final destas atividades é atingido, o escopo, o cronograma e os recursos estão em equilíbrio existindo a quantidade correta de recursos para concluir todas as atividades de projeto no prazo final do projeto ou mesmo antes.

No caso particular do gerenciamento de projetos de desenvolvimento de *software*, o interesse maior está no gerenciamento eficiente das atividades e recursos atribuídos a determinado projeto por se tratarem dos principais componentes utilizados no desenvolvimento do produto final¹. Já os sistemas de *automação de projetos* tratam das ferramentas computacionais que permitem agilizar este processo de gerenciamento de maneira a torná-los automatizados para o operador [BECERRA-98], [DEBONI-97].

4.2 A Qualidade de *Software*

A garantia da qualidade de *software* consiste num conjunto de atividades que são aplicadas ao longo de um *processo* de desenvolvimento de *software*. Ela abrange atividades do tipo: [PRESSMAN-97]

- ✍ Técnicas de gerenciamento da qualidade;
- ✍ Tecnologias de engenharia de *software* efetivas (métodos e ferramentas);
- ✍ Técnicas formais de revisão que são aplicadas ao longo do processo de desenvolvimento de *software*;

- ✍ Uma estratégia de testes;
- ✍ Controle da documentação de *software*;
- ✍ Um procedimento para garantir a compatibilidade com padrões de desenvolvimento de *software* e (7) mecanismos de medida e reporte.

As atividades de gerenciamento de projetos de desenvolvimento de *software* relacionam-se com algumas atividades de qualidade de *software*. Para identificar este relacionamento e determinar o escopo do sistema SAPS, será utilizado como referência o modelo de qualidade CMM¹ (Capability Maturity Model) da SEI (Software Engineering Institute)² [SEI-01]. A utilização do CMM servirá também para o propósito do experimento do sistema SAPS através de um estudo de caso que utiliza este modelo de qualidade.

O modelo SW-CMM é baseado em cinco níveis de maturidade, cada nível contendo determinadas áreas-chaves que classificam a empresa em determinado nível. O sistema SAPS não atende a todas as áreas-chaves que definem um determinado nível (uma vez que não é este o seu propósito principal) mas a algumas áreas-chave isoladas, em determinado grau, que são listadas a seguir [BARRETO-98].

¹ Em outros setores, tal como na construção civil, a matéria-prima, os equipamentos de construção e as empreitadas também devem ser considerado no gerenciamento de projetos, além dos recursos de mão-de-obra.

² Existem vários modelos de qualidade baseados no CMM tais como o SW-CMM, o SE-CMM, o IPD-CMM, o TSP e PSP [SPINA-98], [SEI-01]. Estamos interessados aqui no SW-CMM, que trata do modelo de qualidade para desenvolvimento de *software*.

³ Outros modelos de qualidade de *software* existem tais como ISO/IEC 15504 (SPICE), BOOTSTRAP, Software Technology Diagnostic e Trillium [ITABORAHY-99].

✍ Nível 2 (Foco: Processo de Gerenciamento de Projetos)

? Planejamento do projeto:

- Descrição: Estimativas relativas ao *software* são documentadas para uso no planejamento e acompanhamento do projeto de *software*.

? Visão geral e acompanhamento do projeto:

- Descrição: Resultados reais são acompanhados de acordo com o planejamento do *software*.

✍ Nível 3 (Foco: Processos de engenharia e apoio)

? Foco do processo organizacional:

- Descrição: Os pontos fortes e fracos do processo de desenvolvimento de *software* utilizado são identificados.

? Definição do processo organizacional:

- Descrição: O processo padrão de desenvolvimento de *software* da organização é desenvolvido e mantido. A informação relacionada ao uso do processo padrão de desenvolvimento de *software* é coletada, revisada e disponibilizada.

? Gerenciamento de *software* integrado:

- Descrição: O processo de *software* definido para o projeto é uma versão adaptada do processo padrão de desenvolvimento de *software* da organização.

✍ Nível 4 (Foco: Qualidade do produto e do processo)

? Gerenciamento quantitativo dos processos:

- Descrição: A performance do processo de desenvolvimento de *software* definido para o projeto é controlada quantitativamente.

Quando o sistema SAPS for apresentado, no próximo item, retornaremos a este tópico identificando as áreas-chaves que cada módulo do sistema atende.

4.3 O Sistema de Automação de Projetos de *Software* (SAPS)

O sistema SAPS tem por intuito facilitar a área de gerenciamento de projetos de *software* automatizando determinadas tarefas necessárias a ela. Seu escopo também foi elaborado de forma a atender a requisitos de outras áreas como a de qualidade de *software*, o que será demonstrado através da associação de suas funções à determinadas áreas-chaves do modelo de qualidade SW-CMM, discutido no item anterior.

Este sistema está composto de cinco módulos representados no diagrama IDEF0¹ da [figura 4.2 \[VERNADAT-96\]](#). Cada módulo possui as seguintes responsabilidades:

- ✍ Gerenciador de Configuração: Este módulo é responsável pelo:
 - ? Cadastro das atividades de projeto determinadas pelo processo definido pela corporação;
 - ? Cadastro dos recursos de projeto;
 - ? Cadastro do calendário de cada recurso determinando seus dias de trabalho e de folga.
- ✍ Gerenciador de Planejamento: este módulo é responsável pela definição do cronograma de atividades de um projeto, utilizando os elementos de projeto cadastrados no módulo anterior;

¹ Um diagrama IDEF0 expressa um processo da mesma forma que naturalmente pensamos sobre ele. Seu primeiro nível A0 representa o processo como um todo e seus sub-níveis (A1, A2, A3, etc.) os sub-processos. Cada parte do processo é representada no diagrama por um retângulo que possui uma flecha à esquerda indicando sua entrada, uma à direita indicando sua saída, uma acima indicando seu controle e outra abaixo indicando o mecanismo utilizado.

Figura 4.2 Diagrama IDEF0 (nível 1) do sistema SAPS.

- ✍ Gerenciador de Atividades: este módulo é responsável pela coleta das informações de execução das atividades alocadas a cada recurso de desenvolvimento;
- ✍ Gerenciador de Controle: este módulo é responsável pelo controle da execução das atividades de projeto, comparando-a com o planejado. A informação necessária para esse controle é obtida do módulo anterior e permite a realização de ajustes no planejamento original, caso necessário;
- ✍ Gerenciador de Coordenação: este módulo é responsável pela obtenção de estatísticas do processo de gerenciamento de projetos de *software* indicando o desempenho de seus elementos.

A [tabela 4.1](#) mostra a relação dos módulos do sistema SAPS citados acima com as áreas-chaves do modelo de qualidade CMM. Na tabela são relacionados os módulos do sistema SAPS que oferecem suporte à automação de determinadas áreas-chaves do CMM, especificando desta maneira seu escopo para este estudo de caso.

CAPABILITY MATURITY MODEL (CMM)						
<div style="text-align: center;"> <div>Area-Chave</div> <div>Módulo</div> </div>		NÍVEL 2		NÍVEL 3		NÍVEL 4
		Planejamento do Projeto	Visão Geral e Acompanhamento do Projeto	Foco do Processo Organizacional	Definição do Processo Organizacional	Gerenciamento de Software Integrado
S	Ger. de Configuração				X	
A	Ger. de Planejamento	X				X
P	Ger. de Atividades		X			X
P	Ger. de Controle		X			X
S	Ger. de Coordenação		X	X	X	X

Tabela 4.1 Relação dos módulos do sistema SAPS com as áreas-chaves do modelo de qualidade SW-CMM.

4.4 O Sistema SAPS e os Pontos de Vista do ODP

Neste item o sistema SAPS será modelado segundo os cinco pontos de vista do ODP resultando numa especificação do projeto. Esta especificação servirá então para a definição no próximo item da arquitetura do projeto, a qual será integrada à arquitetura do *middleware Maestro*, definindo assim a arquitetura geral do sistema. A informação sobre a implementação do projeto e os diagramas adicionais gerados farão parte do anexo desta dissertação.

4.4.1 O Ponto de Vista da Empresa

O ponto de vista da empresa do ODP é analisado, para o SAPS, através do diagrama de caso de uso do UML apresentado na [figura 4.3](#). Neste diagrama constam os atores responsáveis pela interação com o sistema e os cinco casos de uso equivalentes aos cinco módulos do SAPS discutidos anteriormente.

É também salientado neste diagrama que o SAPS apóia-se nos serviços fornecidos pelo *middleware Maestro*, cuja integração será analisada em mais detalhes ao verificarmos a arquitetura geral do sistema. Desta forma será reutilizada a especificação dos serviços fornecidos pelo *middleware Maestro* que são necessários na especificação do SAPS. Tais serviços englobam serviços de distribuição, de apresentação, de controle de acesso e de comunicação.

Os atores e casos de uso do diagrama apresentado equivalem a objetos empresa neste ponto de vista do ODP. São as seguintes as responsabilidades de cada objeto empresa pertencente à comunidade de usuários do sistema SAPS¹:

¹ Os objetos empresa equivalentes aos casos de uso do diagrama da [figura 4.3](#) foram descritos anteriormente no item 4.3.

- ✍ Gerente de Desenvolvimento de *Software*: responsável pela definição do processo de *software* da corporação e verificação de suas métricas;
- ✍ Gerente de Projeto: responsável pelo planejamento de um projeto e controle de sua execução;
- ✍ Recurso de Desenvolvimento de *Software*: responsável pela execução de cada etapa do planejamento do projeto ao qual foi alocado;
- ✍ Administrador de Sistemas de Informação: responsável pela gerência dos recursos oferecidos pelo *middleware Maestro* ao sistema SAPS;

Quanto aos requisitos de segurança que se aplicam a estes objetos empresa e constituem um contrato entre as partes, apenas o objeto *Recurso de Desenvolvimento de Software* requer regras de segurança para acessar o sistema uma vez que, por definição, é o único objeto que poderá se encontrar fora dos domínios da corporação e ainda assim acessar o SAPS remotamente. Os demais objetos encontram-se dentro do domínio da corporação e possuem sempre acesso local ao sistema¹.

Verifica-se que a análise no ponto de vista do ODP fica bastante simplificada uma vez que os requisitos do sistema relativos à sua distribuição, segurança, comunicação e acesso remoto já foram examinados durante a especificação do *middleware Maestro* e podem ser reutilizados aqui, possibilitando assim um maior esforço de análise para o negócio ao qual o sistema SAPS pretende oferecer uma solução.

¹ Supõe-se, para efeito de simplificação da análise, que não existam requisitos de segurança para o acesso ao sistema SAPS dentro dos domínios da corporação.

Figura 4.3 Diagrama de caso de uso do sistema SAPS e do middleware Maestro na visão empresa do ODP.

4.4.2 O Ponto de Vista da Informação

O ponto de vista da informação do ODP é analisado, para o SAPS, através do diagrama de objetos do UML apresentado na [figura 4.4](#). Neste diagrama constam os objetos informação do sistema SAPS e do *middleware Maestro*, responsáveis pela troca de informação no sistema e com seus usuários.

Os objetos informação do sistema SAPS são responsáveis pela troca de informação que permite o gerenciamento da configuração, do planejamento, das atividades, do controle e da coordenação do seu sistema. Esta informação é armazenada em um serviço comum responsável pelo gerenciamento de dados de projeto e apresentada aos usuários através do serviço do *middleware Maestro* de apresentação. O serviço gerenciador de arquivos de projeto, é utilizado para apoio aos objetos informação do SAPS que necessitem armazenar em arquivos informações relacionadas ao seu estado.

Para o SAPS necessita-se especificar a seguinte informação:

- ✍ Esquema Invariante: neste esquema é importante salientar a informação referente ao acesso remoto ao sistema, identificada no contrato do ponto de vista da empresa. A informação de acesso remota é sempre válida e consiste no envio de um *login* e de uma senha do SAPS, necessários para a autenticação do usuário no sistema.
- ✍ Esquema Estático: ao iniciar, o sistema SAPS inicializa todos os *objetos informação persistentes* do sistema, ou seja, recupera os dados armazenados no gerenciador de dados de projeto e disponibiliza os objetos informação criados para acesso pelos demais objetos informação do sistema.

Figura 4.4 Diagrama de Objetos do sistema SAPS e do middleware Maestro na visão informação do ODP.

Esquema Dinâmico: durante a operação do SAPS, os objetos persistentes podem ser acessados pelos demais objetos informação podendo ser alterados, apagados, criados e persistidos. A ação de persistência só é realizada via comando, sendo os dados perdidos caso o objeto seja destruído antes que esta ação seja realizada.

Neste ponto de vista definiu-se os serviços do *middleware Maestro* e os serviços comuns que serão utilizados para a apresentação e o armazenamento e da informação do sistema SAPS bem como sua maneira de armazenamento utilizando objetos persistentes.

4.4.3 O Ponto de Vista da Computação

O ponto de vista da computação do ODP é analisado, para o SAPS, através do diagrama de classes do UML apresentado na [figura 4.5](#). Neste diagrama constam os objetos computação do sistema SAPS e do *middleware Maestro*, responsáveis pela operação do sistema e interação com seus usuários.

Percebe-se neste diagrama o papel central do *middleware Maestro* como elemento de interoperabilidade entre os diversos objetos computação do sistema SAPS. Através dos elementos *Consumidor*, *Provedor* e *Fornecedor* do *middleware Maestro*, o processamento relacionado à interoperabilidade deste sistema ficam sob o encargo do *middleware Maestro*, não havendo a necessidade de uma nova análise sobre esta questão.

Também foi definido um objeto computação *Gerenciador de Objetos Persistentes* de forma a atender os requisitos definidos no ponto de vista da informação para o serviço de armazenamento de dados. Este objeto computação é apresentado no diagrama de classes do UML da [figura 4.6](#) mostrando suas interfaces.

Figura 4.5 Diagrama de classes do sistema SAPS e do middleware Maestro na visão computação do ODP.

Figura 4.6 Classes persistentes do componente *Gerenciador de Objetos Persistentes*.

Nesta figura, observa-se que este objeto computação é composto por quatro classes: *Serviço*, *Trabalho*, *Equipe* e *Calendário*. Cada uma destas classes, por sua vez, compõem subclasses: *Projeto*, *Atividade*, *Recurso* e *Dia*, respectivamente. As primeiras classes oferecem quatro interfaces de acesso, cada uma permitindo o acesso a diferentes coleções de objetos internos.

A primeira interface, fornecida pela classe *Serviço*, possibilita o acesso a uma coleção de objetos que representam os diversos projetos cadastrados no banco de dados de projetos. Através da associação que existe entre esta classe e a classe *Trabalho*, é possível acessar outra coleção de objetos contendo as diversas atividades associadas a determinado projeto cadastrado no SAPS. Em seguida, através de outra associação, tem-se acesso a outra coleção de objetos contendo os recursos associados a cada uma das atividades de determinado projeto do SAPS. E finalmente, a última associação permite acesso à coleção de objetos contendo os dias de folga ou de trabalho de cada um dos recursos de projeto descritos acima.

A segunda interface, fornecida pela classe *Trabalho*, possibilita o acesso a uma coleção de objetos que representam as diversas atividades obrigatórias a todos os projetos, ou seja, que definem o processo da corporação (as demais associações possuem descrição similar àquela feita para a primeira interface).

A terceira interface, fornecida pela classe *Equipe*, possibilita o acesso a uma coleção de objetos que representam todos os recursos cadastrados no SAPS (as demais associações possuem descrição similar àquela feita para a primeira interface).

A quarta interface, fornecida pela classe *Calendário*, possibilita o acesso a uma coleção de objetos que representam os dias, de trabalho ou de folga, que representam o calendário de trabalho para todos os recursos de projeto.

A utilização do *middleware Maestro* neste ponto de vista permitiu que o esforço de especificação fosse concentrado nos objetos computação do sistema SAPS

ao invés de se concentrar no processamento relacionado à sua interoperabilidade, facilitando assim a sua especificação neste ponto de vista.

4.4.4 O Ponto de Vista da Engenharia

O ponto de vista da engenharia do ODP é analisado, para o SAPS, através do diagrama de componentes do UML apresentado na [figura 4.7](#). Neste diagrama constam os objetos engenharia do sistema SAPS e do *middleware Maestro*, distribuídos entre os diversos nós do sistema.

Os usuários do SAPS (Gerente de Desenvolvimento de *Software*, Gerente de Projeto e Projetista) localizam-se em um nó individual do sistema. Os gerentes apóiam-se no serviço remoto de um nó servidor de arquivos, através de um canal de comunicação estabelecido entre eles. Já os projetistas apóiam-se no serviço remoto de um nó servidor Web, através, também, de um canal de comunicação.

Uma vez que os objetos engenharia *Gerenciador de Configuração*, *Gerenciador de Planejamento*, *Gerenciador de Controle* e *Gerenciador de Coordenação* estejam carregados no nó de seus usuários, estes estabelecem uma ligação através de um canal de comunicação com o objeto engenharia *Gerenciador de Objetos Persistentes*, requisitando serviços a ele. O objeto engenharia *Gerenciador de Atividades*, por ser carregado no nó servidor Web ao invés de no nó do projetista, não precisa estabelecer um canal de comunicação com o objeto engenharia *Gerenciador de Objetos Persistentes* uma vez que ambos encontram-se no mesmo nó.

Figura 4.7 Diagrama de componentes do sistema SAPS e do middleware Maestro na visão engenharia do ODP.

O objeto engenharia *Gerenciador de Objetos Persistentes*, ao ser carregado, estabelece uma ligação local com o objeto engenharia *Gerenciador de Dados de Projeto* para requisição de serviços. Este objeto, por sua vez, estabelece uma ligação remota, através de um canal de comunicação, com o objeto engenharia *Banco de Dados de Projeto* para obter acesso aos dados de projeto mantidos por este objeto no nó servidor de dados.

Foi definido, neste ponto de vista, a infra-estrutura requerida para suportar a interação entre os objetos engenharia do sistema SAPS. Também foram definidos os canais de comunicação entre os objetos engenharia, cuja especificação reutiliza a definida para o *middleware Maestro* com respeito ao seu serviço de comunicação.

4.4.5 O Ponto de Vista da Tecnologia

O ponto de vista da tecnologia do ODP é analisado, para o SAPS, através do diagrama de componentes do UML apresentado na [figura 4.8](#). Neste diagrama constam os objetos tecnologia do sistema SAPS e do *middleware Maestro*, distribuídos entre os diversos nós do sistema.

Na visão tecnologia, analisamos os componentes apresentados na visão engenharia segundo a tecnologia utilizada para sua implementação. Alguns componentes foram desenvolvidos utilizando alguma linguagem de programação do mercado; para outros foram aproveitados componentes comerciais disponibilizados por fabricantes; e para outros ainda foram aproveitados pacotes de mercado com alguma configuração necessária.

Os seguintes objetos tecnologia foram desenvolvidos utilizando uma linguagem de programação de mercado:

Figura 4.8 Diagrama de componentes do sistema SAPS e do middleware Maestro na visão tecnologia do ODP.

- ✍ *Gerenciador de Configuração e Gerenciador de Coordenação* implementados como uma aplicação utilizando a linguagem *Visual Basic 6.0* da Microsoft;
- ✍ *Gerenciador de Objetos Persistentes* implementado como um componente ActiveX utilizando a linguagem *Visual Basic 6.0* da Microsoft;
- ✍ *Gerenciador de Atividades* implementado como um componente Web utilizando a linguagem *Active Server Pages (ASP)* da Microsoft;

O seguinte objeto tecnologia utilizou um componente pronto de mercado:

- ✍ *Gerenciador de Dados de Projeto* utilizou o componente *ActiveX Data Objects 2.5 (ADO)* da Microsoft;

Os seguintes componentes utilizaram pacotes prontos de mercado:

- ✍ *Gerenciador de Planejamento e Gerenciador de Controle* utilizaram o pacote de mercado *Project 2000* da Microsoft com algumas configurações realizadas utilizando-se a linguagem de programação *Visual Basic for Application (VBA)*;
- ✍ *Servidor Web* utilizou o pacote de mercado *Internet Information Server (IIS)* da Microsoft;
- ✍ *Banco de Dados de Projeto* utilizou o pacote de mercado *Oracle 8i* da Oracle;
- ✍ *Gerenciador de Telas Remoto* utilizou o *Communicator 4.73* da Netscape;
- ✍ *Gerenciador de Telas Local* utilizou o *Gerenciador de Desktop* do Windows NT Workstation 4.0 da Microsoft;

Os canais de comunicação definidos utilizaram os seguintes protocolos:

- ✍ *Distributed Component Object Model* (DCOM) da Microsoft para comunicação entre os componentes dos nós *Servidor de Arquivo* e *Servidor Web*;
- ✍ *Hyper Text Transfer Protocol* (HTTP) da W3C para comunicação entre os componentes dos nós *Estação do Projetista* e *Servidor Web*;
- ✍ *Open DataBase Connectivity* (ODBC) da Microsoft para comunicação entre os componentes dos nós *Servidor Web* e *Servidor de Dados*;
- ✍ *Internetwork Packet Exchange / Sequenced Packet Exchange* (IPX/SPX) da Novell para comunicação entre os componentes dos nós *Gerente de Desenvolvimento de Software* e *Gerente de Projeto* com o nó *Servidor de Arquivo*.

Como a especificação ODP obtida para o SAPS é aberta, outras implementações com tecnologias de outros fabricantes são também possíveis, tanto para o sistema SAPS como para o *middleware Maestro*. Aqui analisou-se apenas a interoperabilidade entre produtos e linguagens de programação de um mesmo fabricante de maneira a demonstrar uma possível implementação deste sistema¹.

4.5 A Arquitetura do Sistema SAPS

A [figura 4.9](#) ilustra a arquitetura para o sistema SAPS. Esta arquitetura é baseada no modelo de referência SAA (Sistema Aberto de Automação) [\[MARTUCCI-92\]](#). Nesta arquitetura existem três camadas justapostas: a camada de coordenação mais acima, a camada de controle ao meio e a camada de instrumentação abaixo.

¹ No item 5.3 do próximo capítulo serão propostos novos sistemas que tratam da interoperabilidade entre tecnologias de objetos distribuídos de diferentes fabricantes.

Nível de Coordenação

Nível de Controle

Nível de Instrumentação

Figura 4.9 Arquitetura do sistema SAPS baseada no modelo de referência SAA.

A camada de instrumentação do modelo SAA é responsável pela aquisição de dados para o sistema. No SAPS, os módulos *Gerenciador de Planejamento* e *Gerenciador de Atividades* são responsáveis pela aquisição de dados para o planejamento das atividades determinando seu cronograma e pela aquisição de dados das atividades executadas pelos recursos de projeto, respectivamente.

A camada de controle do modelo SAA é responsável pelo controle dos dados obtidos na camada inferior de instrumentação. No SAPS, os módulos *Gerenciador de Configuração* e *Gerenciador de Controle* são responsáveis por estabelecer as atividades de processo e recursos de projeto a serem utilizados pelos módulos da camada inferior e por controlar e ajustar a execução do cronograma definido na camada inferior, respectivamente.

A camada de coordenação do modelo SAA é responsável pela obtenção de estatísticas que demonstrem a eficiência do sistema, podendo alterá-lo se necessário. No SAPS, o módulo *Gerenciador de Coordenação* é responsável por obter estatísticas que demonstrem a eficiência do processo utilizado pela empresa e a determinação do desempenho da equipe de projeto. Caso necessário, podem ocorrer ajustes no nível inferior de controle para que a eficiência e o desempenho do sistema tornem-se aqueles almejados pela corporação.

A integração da arquitetura do SAPS à do *Maestro* pode ser vista na [figura 4.10](#). Esta figura mostra, de maneira tridimensional, a arquitetura do SAPS sobre a arquitetura do *Maestro*, utilizando os serviços fornecidos por este.

Figura 4.10 Arquitetura do sistema SAPS integrada com a arquitetura do middleware Maestro.

Os seguintes módulos foram utilizados para executar as funções do *middleware Maestro*:

- ✍ Apresentador: foram utilizados os módulos de gerenciamento de *desktop* do Windows e um *browser* HTML para esta função;
- ✍ Descritor/Produtor: foram utilizados um servidor de arquivos (Netware), um servidor Web (IIS) e um servidor de dados (UNIX);
- ✍ Gerenciador de eventos: foi utilizado o sistema de mensagens do Windows;
- ✍ Comunicador: foi desenvolvido um componente de acesso que permite a autenticação de usuários do sistema SAPS;
- ✍ Serviço de apoio: foi desenvolvido um componente que permite o gerenciamento de objetos persistentes, facilitando assim o desenvolvimento das aplicações.

Para o sistema SAPS foram desenvolvidas as seguintes aplicações baseadas no modelo de referência SAA:

- ✍ Instrumentação: foram desenvolvidas as aplicações *Gerenciador de Planejamento* e *Gerenciador de Atividade*;
- ✍ Controle: foram desenvolvidas as aplicações *Gerenciador de Configuração* e *Gerenciador de Controle*;
- ✍ Coordenação: foi desenvolvida a aplicação *Gerenciador de Coordenação*.

4.6 Conclusões do Capítulo

O objetivo principal deste capítulo foi experimentar a arquitetura do *middleware Maestro* através de um estudo de caso. O experimento realizado utilizando um estudo de caso na área de gerenciamento de projetos de desenvolvimento de *software* obteve êxito na sua especificação e implementação demonstrando a viabilidade da utilização do *middleware Maestro* neste tipo de contexto.

Através deste experimento demonstrou-se também um método de especificação ODP através do qual este referencia, em cada um dos seus pontos de vista, objetos utilizados em sua especificação e definidos em outro projeto ODP. É o caso da especificação do sistema SAPS que referencia objetos do *middleware Maestro* em seus pontos de vista, conforme ilustrado na [figura 4.1](#).

Verificou-se também a possibilidade de reutilização da especificação dos objetos referenciados em outro projeto ODP, diminuindo assim o tempo necessário para a análise de um novo projeto.

Ao final do experimento obteve-se uma arquitetura aberta que permite sua implementação utilizando padrões e tecnologias de diferentes fabricantes. Demonstrou-se ainda que esta arquitetura pode ser integrada à de outro sistema ODP utilizado em sua especificação, de forma a obter uma arquitetura geral para todo o sistema.

Capítulo 5

Considerações Finais

Neste capítulo final são apresentadas as principais conclusões deste trabalho, sendo apresentados também os comentários gerais, a continuidade da pesquisa para impulsionar o desenvolvimento de novos trabalhos e a contribuição acadêmica obtida.

5.1 Conclusões

A utilização do modelo de referência ODP na especificação do middleware *Maestro* mostrou-se adequada à análise de sistemas complexos como este, analisando seus requisitos segundo cinco pontos de vista distintos do sistema. A análise dos requisitos inerentes aos sistemas distribuídos fica também simplificada através da utilização dos conceitos descritivos e prescritivos do ODP para este tipo de ambiente.

Alguns requisitos novos, criados devido à mobilidade dos usuários, a novos dispositivos de acesso, a novas tecnologias e conceitos para construção de aplicativos distribuídos também foram considerados na definição da arquitetura do middleware *Maestro*, tornando-o flexível o suficiente para o desenvolvimento de aplicativos de última geração e sua integração com sistemas legados.

O método adotado de decomposição do sistema em cada ponto de vista do ODP mostrou-se adequado para identificar a correlação entre os objetos de cada ponto de vista permitindo assim um melhor controle das funcionalidades do sistema em cada nível conforme ilustrado na [figura 5.1](#). Este mesmo método também parece ser coerente para a análise do sistema em um contexto particular ou para detalhar melhor um determinado objeto de um ponto de vista.

Figura 5.1 Método de decomposição do sistema ODP utilizado para a análise do middleware *Maestro*.

Como o ODP não especifica um método ou uma linguagem para serem usados na modelagem dos seus pontos de vista, foi utilizado o padrão UML para esta finalidade. Através das correspondências entre ODP e UML especificadas em [MCI-97] e [RATIONAL-97], foi possível modelar todos os pontos de vista do ODP utilizando os diagramas e os conceitos do UML. Também foi possível estabelecer uma correspondência entre os tipos de diagramas UML mais apropriados para modelagem de determinado ponto de vista. Esta correspondência foi baseada no estudo do relacionamento das cinco visões do UML e as cinco visões do ODP.

5.2 Comentários Gerais

Um aspecto importante da dissertação foi o projeto e a implementação de um serviço de objetos persistentes para o middleware *Maestro*. Objetos persistentes são classes especiais de objetos capazes de salvar seu estado atual (no caso seus atributos) de maneira a recuperá-los na próxima vez que forem instanciados. Sua implementação (descrita no capítulo 4 e no anexo) criou a possibilidade de desenvolver as aplicações totalmente orientadas a objetos, sem a noção de estarem na realidade manipulando dados de um banco de dados relacional, tarefa esta realizada pelo componente de objetos persistentes.

5.3 Continuidade da Pesquisa

Por se tratar de um sistema aberto, o middleware *Maestro* e as aplicações desenvolvidas para ele podem ser experimentadas utilizando novas tecnologias de mercado. No experimento realizado no capítulo quatro foram utilizadas somente tecnologias da Microsoft relacionadas ao seu padrão COM. Numa continuidade da pesquisa o mesmo experimento ou outros podem ser realizados utilizando-se tecnologias baseadas em CORBA e Enterprise JavaBeans.

Outro experimento que pode ser realizado através dos resultados obtidos neste trabalho é a integração das tecnologias COM e CORBA ou COM e Enterprise JavaBeans. A [figura 5.2](#) ilustra um exemplo onde as tecnologias COM e CORBA são integradas através do desenvolvimento de um stub COM, desenvolvido na linguagem Java, e um skeleton CORBA, também desenvolvido na linguagem Java. Observe que o stub COM consiste em um COM Wrapper, ou seja, o objeto escrito em Java, que acessa as funções do CORBA, é encapsulado na tecnologia COM. Mais detalhes a respeito de integração COM/CORBA pode ser obtido em [\[CHANG-00\]](#).

Figura 5.2 Integração das tecnologias COM e CORBA.

5.4 Contribuição Acadêmica

A principal contribuição acadêmica obtida neste trabalho está relacionada à arquitetura desenvolvida para o middleware *Maestro*. Mostrou-se aqui que, além de servir como arquitetura para o desenvolvimento de um middleware comercial, a mesma arquitetura pode ser utilizada como um modelo de referência para o desenvolvimento de aplicações distribuídas e abertas, conforme demonstrado no capítulo quatro.

Esta mesma arquitetura pode servir também como um metamodelo de forma a descrever módulos que podem ser desenvolvidos utilizando-se de patterns de arquitetura [BUSCHMANN-96] ou patterns de projeto [GAMMA-98], auxiliando, desta maneira, no desenvolvimento de novos módulos para aplicações específicas. Esta relação foi demonstrada no capítulo três.

Ainda com relação à arquitetura do middleware *Maestro*, um avanço foi obtido na estruturação dos módulos *Consumidor*, *Provedor* e *Fornecedor*, conforme visto na visão empresa do ODP. Esta estruturação tem um paralelo com a estrutura cliente/servidor de três camadas (three-tier) conforme apresentada em [UMAR-97]. Este modelo, porém, é aplicado a aplicações distribuídas, cujo conceito foi estendido à arquitetura de middlewares. Outra diferença também ocorre na distinção entre a funcionalidade de acesso a dados do modelo de três camadas com relação a uma funcionalidade mais ampla do módulo *Fornecedor* do middleware *Maestro*. Esta funcionalidade, orientada a serviços ao invés de dados, permite ao middleware *Maestro* uma maior abrangência para execução de aplicações federadas, tão necessárias em nossos dias.

Bibliografia

- [ALMEIDA-00] ALMEIDA, J. **Automação dos Processos de Controle de Qualidade da Água e Esgoto dos Laboratórios de Controle Sanitário**. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 2000.
- [AOYAMA-98] AOYAMA, M. **New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development**. 1998 International Workshop on Component-Based Software Engineering. URL: <http://www.sei.cmu.edu/cbs/icse98/papers/p14.html>
- [ARK-99] ARK, W. S.; LEHMAN, T. J.; MORALES, A.; MUNSON, M. C.; EDLUND, S.; GUILLEN, M. **A Universal Information Appliance**. IBM Systems Journal, Vol. 38, N?4, 1999, pgs. 504 – 507.
- [BACON-00] BACON, J.; MOODY, K.; BATES, J.; HAYTON, R.; MA, C.; McNEIL, A; SEIDEL, O. SPITERI, M. **Generic Support for Distributed Applications**. IEEE Computer – Vol. 33, N?3, Mar/2000.
- [BARRETO-98] BARRETO, A. **Qualidade de Software**. URL: <http://www.barreto.com.br/qualidade>
- [BECERRA-98] BECERRA, J. L. R. **Aplicabilidade do Padrão de Processamento Distribuído e Aberto nos Projetos de Sistemas Abertos de Automação**. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 1998.
- [BERNSTEIN-96] BERNSTEIN, P. **Middleware: A Model for Distributed System Services**. Communications of the ACM – Vol. 39, N? 2, Fev/1996, pgs. 86 – 98.

- [BOOCH-99] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. ***The Unified Modeling Language User Guide***. Addison-Wesley, 1999.
- [BROWN-99] BROWN, A. W. ***Mastering the Middleware Muddle***. IEEE Software – Vol. 16, N°4, Jul/Ago 1999, pgs. 18 – 21.
- [BUSCHMANN-96] BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. ***Pattern-Oriented Software Architecture – A System of Patterns***. John Wiley & Sons Ltd, 1996.
- [CHANG-00] CHANG, E.; ANNAL, D.; GRUNTA, F. ***A Large Scale Distributed Object Achitecture – CORBA & COM for Real Time Systems***. ICC Software Development. Philips Public Communication Limited, 2000
- [DAVIS-99] DAVIS, J. ***Mall Rats***. Business 2.0, Jan/1999, pgs. 41 – 50.
- [DEBONI-97] DEBONI, J.; MARTINI, J. ***Um Método para Desenvolvimento de Sistemas Computacionais de Apoio a Projetos de Engenharia***. Boletim Técnico da Escola Politécnica da Universidade de São Paulo, 1997.
- [EUSTICE-99] EUSTICE, K. F.; LEHMAN, T. J.; MORALES, A.; MUNSON, M. C.; EDLUND, S.; GUILLEN, M. ***A Universal Information Appliance***. IBM Systems Journal, Vol. 38, N°4, 1999, pgs. 575 – 598.
- [FERREIRA-00a] FERREIRA, C.; BECERRA, J. ***Maestro: Um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software***. SBES'2000 - V Workshop de Teses e Dissertações em Engenharia de Software. SBC - Sociedade Brasileira de Computação, Out/2000.

- [FERREIRA-00b] FERREIRA, C.; BECERRA, J. **Especificação ODP de um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software.** WORKCOMP'2000 - Workshop de Computação. ITA - Instituto Tecnológico de Aeronáutica (Divisão de Ciência da Computação), Out/2000.
- [FERREIRA-00c] FERREIRA, C.; BECERRA, J. **Especificação de um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software.** OD'2000 - Objetos Distribuídos'2000. CITS - Centro Internacional de Tecnologia de Software / Visionnaire, Nov/2000.
- [FILHO-01] FILHO, A. **Especificação de um Sistema de Informação Integrado para o Planejamento Operacional em Empresas de Energia.** Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 2001.
- [GAMMA-95] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software.** Addison-Wesley, 1995.
- [HAN-98] HAN, J. **Characterization of Components.** 1998 International Workshop on Component-Based Software Engineering, 1998.
URL: <http://www.sei.cmu.edu/cbs/icse98/papers>
- [ISO-96a] ISO Recommendation X.902/ISO/IEC **10746-2: Information technology - Open Distributed Processing - Reference Model: Foundations.** 1996.

- [ISO-96b] ISO Recommendation X.903/ISO/IEC **10746-3: Information technology - Open Distributed Processing - Reference Model: Architecture.** 1996.
- [ISO-98] ISO Recommendation X.901/ISO/IEC **10746-1: Information technology - Open Distributed Processing - Reference Model: Overview.** 1998.
- [ITABORAHY-99] ITABORAHY, A. **Avaliação do Processo de Software Utilizando SW-CMM e ISO/IEC 15504.** MBA - Engenharia de Software. Escola Politécnica da Universidade de São Paulo, 1999.
- [KIELY-98] KIELY, D. **Are Components the Future of Software?** IEEE Computer, Vol. 31, No. 2, Fev/1998, pgs. 10 – 11.
- [KOZACZYNSKI-98] KOZACZYNSKI, W.; BOOCH, G. **Component-Based Software Engineering.** IEEE Software. Vol. 15, N? 5, Set/Out 1998, pgs. 34 - 36.
- [MARTE-00] MARTE, C. **Sistemas Computacionais Distribuídos Aplicados em Automação dos Transportes.** Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 2000.
- [MARTUCCI-92] MARTUCCI Jr., M. **Estudos de Estruturas de Sistemas de Automação.** Trabalho (Professor Livre Docente) – Escola Politécnica da Universidade de São Paulo, 1992.
- [MATSUYAMA-97] MATSUYAMA, F. **Generalização dos Níveis de Controle e Instrumentação de um Sistema Aberto de Automação.** Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 1997.

- [MCI-97] MCI Systemhouse Corporation. ***Relationship of the Unified Modeling Language to the Reference Model of Open Distributed Computing - V 1.1.2.*** 1997.
URL: <http://enterprise.Systemhouse.MCI.com/UML-ODP>
- [MICROSOFT-95] Microsoft Corporation and Digital Equipment Corp. ***The Component Object Model Specification - V. 0.9.*** 1995.
URL: <http://msdn.microsoft.com/library/specs/S1D137.HTM>
- [MICROSOFT-99] Microsoft Corporation. ***User's Guide for Microsoft Project 2000 – Project Management for Business Results.*** 1999.
- [OMG-99] Object Management Group. ***OMA Component Definitions.*** 1999. URL: <http://www.omg.org/oma/>
- [PRESSMAN-97] PRESSMAN, R. ***Software Engineering – A Practitioner's Approach (4th Edition).*** MacGraw-Hill, 1997.
- [PUTMAN-01] PUTMAN, J. ***Architecting with RM-ODP.*** Prentice-Hall PTR, 2001.
- [RATIONAL-97] Rational Software Corporation. ***UML Semantics Appendix M5 – Relationship to Reference Model of Open Distributed Computing – V 1.0.*** 1997.
URL: <http://www.rational.com>
- [RAYMOND-95] RAYMOND, K. ***Reference Model of Open Distributed Processing (RM-ODP): Introduction.*** Queensland University, 1995. URL : http://archive.dstc.edu.au/AU/research_news/odp/ref_model/papers/icodp95.ps

- [SCHNEIDER-98] SCHNEIDER, J.; NIERSTRASZ, O. ***Scripting: Higher-level Programming for Component-Based Systems***. OOPSLA'98 Conference, 1998.
URL: <http://www.iam.unibe.ch/%7Escg/OOPSLA98>
- [SEI-01] Software Engineering Institute. ***CMM-Capability Maturity Model***. Carnegie Mellon University, 2001.
URL: <http://www.sei.cmu.edu>
- [SHAW-96] SHAW, M.; GARLAN, D. ***Software Architecture***. Prentice Hall, 1996.
- [SPINA-98] SPINA, E. MARTUCCI, M. ***Um Estudo Comparativo dos Sistemas de Qualidade***. Boletim Técnico da Escola Politécnica da Universidade de São Paulo, 1998.
- [SUPERNOVA-97] SuperNova. ***Component-Based Development And Integration of Legacy Systems***. White Paper, Set/1997.
URL: <http://www.supernova.com>
- [SZYPERSKY-98] SZYPERSKY, C. ***Component Software: Beyond Object Oriented Programming***. Addison-Wesley Longman, 1998.
- [TANOMARU-94] TANOMARU, N. ***Projeto de Concepção e Projeto Básico do Nível de Coordenação de um Sistema Aberto de Automação Utilizando Conceitos de Orientação a Objetos***. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, 1994.
- [TAPSCOTT-93] TAPSCOTT, D.; CASTON, A. ***Paradigm Shift - The New Promise of Information Technology***. McGraw-Hill, 1993.

- [UMAR-97] UMAR, A. ***Object-Oriented Client/Server Internet Environments***. Prentice-Hall, 1997.
- [VERNADAT-96] VERNADAT, F. B. ***Enterprise Modeling and Integration: Principles and Applications***. Chapman & Hall, 1996.
- [WEISER-91] WEISER, M. ***The Computer for the Twenty-First Century***. Scientific American, Vol. 265, N°3, Set/91, pgs. 66 – 75.
- [WEISER-93a] WEISER, M. ***Some Computer Science Problems in Ubiquitous Computing***. Communications of the ACM, Vol. 36, N°7, Jul/93, pgs. 74 – 84.
- [WEISER-93b] WEISER, M. ***Hot Topics: Ubiquitous Computing***. IEEE Computer, Vol. 26, N°10, Out/93, pgs. 71 – 72.
- [WHITTEN-95] WHITTEN, N. ***Managing Software Development Projects (2nd Edition)***. John Wiley & Sons, 1995.