CRIAÇÃO E INSTANCIAÇÃO DE ARQUITETURAS DE SOFTWARE ESPECÍFICAS DE DOMÍNIO NO CONTEXTO DE UMA INFRA-ESTRUTURA DE REUTILIZAÇÃO

José Ricardo Xavier

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____

Profª. Cláudia Maria Lima Werner, D.Sc.

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Profª. Fernanda Claudia Alves Campos, D.Sc.

_____

Prof. Eber Assis Schmitz, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2001

JOSÉ RICARDO XAVIER

Criação e Instanciação de Arquiteturas de Software Específicas Domínio no Contexto de uma Infra-estrutura de Reutilização [Rio de Janeiro] 2001

IX, 123 p. 29,7 cm (COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 2001)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Arquiteturas de Software
2. Reutilização
3. Processos de desenvolvimento de software
4. Infra-estrutura de suporte ao desenvolvimento de software
5. Projeto Odyssey

I. COPPE/UFRJ II. Título (série)

Aos meus pais
e a minha esposa.

Agradecimentos

À Professora Cláudia Werner por me apresentar a reutilização de software, por me guiar pelos caminhos da vida acadêmica, por orientar minha tese, por participar da banca examinadora da minha tese, por sua amizade e atenção, e, principalmente, por sua compreensão e confiança em relação a minha capacidade para realizar este trabalho.

Ao Professor Guilherme Horta Travassos por me apresentar a arquitetura de software, por me guiar pelos caminhos da vida acadêmica, por co-orientar a minha tese, por participar da banca examinadora da minha tese, por sua amizade a atenção, e por me ensinar a pesquisar.

À Professora da área de Engenharia de Software da COPPE, Ana Regina da Rocha, pela contribuição ao meu aprendizado no decorrer do curso.

Aos Professores Eber Assis Schmitz e Fernanda Claudia Alves Campos por participarem da banca examinadora da minha tese.

À Regina Braga, Nélson Miller e Márcio Barros por suas contribuições na extensão dos processos e na definição de abordagem de seleção igualmente contidos neste trabalho e pela constante disposição em ajudar em que fosse necessário.

Aos amigos da COPPE, Alexandre Correa, Alexandre Dantas, Róbson Pinheiro, Leonardo Murta, Denis, Gustavo, Marcelo e Alessandréia, pelo convívio gratificante que temos e por terem me ajudado a crescer como desenvolvedor de software.

Aos demais colegas da COPPE pelo incentivo e motivação.

À CAPES pelo apoio financeiro.

E, finalmente, mas não menos importante, a Deus por permitir a conclusão de mais esta etapa importante da minha vida.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CRIAÇÃO E INSTANCIAÇÃO DE ARQUITETURAS DE SOFTWARE ESPECÍFICAS DE DOMÍNIO NO CONTEXTO DE UMA INFRA-ESTRUTURA DE REUTILIZAÇÃO

José Ricardo Xavier

Junho/2001

Orientadores: Cláudia Maria Lima Werner
Guilherme Horta Travassos
Programa: Engenharia de Sistemas e Computação.

Uma Arquitetura de Software Específica de Domínio tem por objetivo tornar mais efetivo o compartilhamento e a reutilização de idéias, métodos, componentes e produtos dentro de uma comunidade de desenvolvedores de aplicações similares. Os processos necessários para sua criação e instanciação são tão importantes quanto sua importância para a arquitetura das aplicações construídas.

Para realizar estes processos torna-se necessário organizá-los de tal forma que seja possível a identificação e a descrição de suas principais fases e atividades e o apoio necessário em alguns dos pontos mais sujeitos às dificuldades específicas ao contexto do projeto arquitetural de software.

Este trabalho foi realizado no contexto do Projeto Odyssey, em desenvolvimento na COPPE/UFRJ, e descreve uma proposta de extensão de uma infra-estrutura de reutilização baseada em Engenharias de Domínio e de Aplicação para que seja possível a adoção do enfoque de desenvolvimento centrado em arquiteturas de software.

‘Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

CREATION AND INSTANCIATION OF DOMAIN-SPECIFIC SOFTWARE ARCHITECTURE IN THE CONTEXT OF A REUSE INFRASTRUCTURE

José Ricardo Xavier

June / 2001

Advisors: Cláudia Maria Lima Werner
Guilherme Horta Travassos
Department: Computer and System Engineering

A Domain-Specific Software Architecture intends to make more effective the sharing and reusing of ideas, methods, components and products among a community of developers of similar applications. The processes to its creation and instantiation are as important as the development of application architectures.

In order to perform those processes, we need to organize them in such a way that it can be possible to identify and describe their main phases and activities and to support those ones more influenced by specific difficulties, which do exist in the context of a software architectural project.

This work was accomplished within the context of the Odyssey Project, which is under development at COPPE/UFRJ, and describes a proposal for the extension of a reuse infrastructure based on Domain and Application Engineering to allow an effective software architecture-centered development.

Índice

Capítulo 1: Introdução

1.1 Motivação

A sociedade mundial está cada vez mais sustentada por uma tecnologia baseada em software. Contemplamos a participação do software em grande parte das ações humanas. Desde a utilização de navegadores sobre informações na Internet através de telefones celulares, até sistemas de controle de geração e distribuição de energia, os sistemas baseados em software vêm atingindo complexidade e alcance inimagináveis. Ao longo dos anos, o elemento software sobrepôs o elemento hardware em relação à importância exercida pela sua ação sobre a automação de serviços e processos disponibilizados por muitas empresas. Hoje, percebemos que o sucesso de um sistema computacional depende muito mais das características do software produzido do que do hardware sob o qual este irá rodar. Mas, em contrapartida, vemos que o aumento dos custos de desenvolvimento, bem como a crescente complexidade dos sistemas e a pouca sistematização para a produção em larga-escala, entre outras, são questões ainda muito presentes em nossa realidade.

Um fator crítico de nossa dependência a esta tecnologia é a constatação de que falhas decorrentes de produtos de software mal produzidos podem gerar efeitos imprevisíveis sobre nossa sociedade, desde um simples mal-funcionamento de um sistema operacional até a perda de vidas. Porém, eliminar ou mesmo diminuir esta dependência tecnológica mostra-se impossível. Fundamentalmente, isso ocorre porque os crescimentos econômicos, científicos e sociais foram e continuam a ser impulsionados pela tecnologia baseada em software. Hoje, por exemplo, podemos perceber que muitas das relações entre organizações só são possíveis graças a uma infra-estrutura tecnológica fortemente centrada em software. Embora identificados os possíveis danos pelo mal funcionamento de um software, os benefícios encontrados pela sua utilização são muito maiores.

No entanto, desenvolver aplicações que efetivamente sejam a realização de um conjunto de funções e características desejadas, para a solução de determinado problema ou automação de uma ou mais atividades, é uma tarefa difícil. Além do produto, o processo de desenvolvimento envolve a gerência de prazos, custos, recursos humanos e técnicos, e influências organizacionais e políticas. Com a competição entre empresas, o software produzido passou a ter grande importância e sobre este investimento vêm também as pressões para a produção de aplicações em prazos e custos cada vez menores, sem implicar na perda de sua qualidade. No

entanto, percebemos que ainda são produzidos produtos de software que extrapolam em muito o tempo necessário para o seu desenvolvimento, bem como os valores monetários estabelecidos, em geral, no início de cada projeto, o que gera inevitavelmente insatisfação e grandes prejuízos.

A principal dificuldade vem do fato de lidarmos com domínios computacionais que ao longo do tempo vêm crescendo em complexidade. Uma forma de se administrar esta complexidade pode ser realizada pela decomposição do problema em uma estrutura hierárquica que permita a sua compreensão e descrição a partir de suas partes. Booch, em (BOOCH, 1994), afirmou que a estruturação de um sistema complexo é uma função tanto de seus componentes quanto da relação hierárquica entre estas partes. Esta visão hierárquica de decomposição do problema possibilitou o desenvolvimento de métodos de construção de software a partir de componentes ou módulos funcionalmente coesos, e que, quando combinados, colaboram para a realização da solução esperada. Esta forma de produção de software ficou conhecida como decomposição modular.

A técnica de modularização de software já existe há algum tempo (STAA, 2000). No entanto, com o desenvolvimento desta abordagem, a representação estrutural de inter-relação entre módulos possibilitou a identificação de algumas características importantes. A primeira delas refere-se a observação de que muitos sistemas complexos possuem um padrão comum de organização de seus módulos. Pôde-se também observar que a diagramação da estrutura modular era insuficiente para informar a natureza dos módulos, o significado de suas ligações, a forma com que os módulos interagem em execução e quais as influencias da organização modular sobre as características observáveis do software.

O aprofundamento dos trabalhos nesta área permitiu a evolução e formalização do conceito que atualmente denominamos *arquitetura de software*. A representação explícita da arquitetura da aplicação contribuiu, sobretudo, para a identificação e avaliação da relação existente entre características de qualidade de um produto de software e o projeto arquitetural que lhe dá fundamento (BASS *et al.*, 1998, BOSCH, 2000).

Da mesma forma que construções podem seguir linhas comuns, as arquiteturas de software também podem seguir alguns estilos ou padrões específicos. A observância a determinado estilo arquitetural possibilita que soluções conhecidas sejam reaplicadas a muitos projetos, a partir da reutilização de modelos de composição e regras de aplicação (SHAW e GARLAN, 1996).

A possibilidade de reutilização deste conhecimento traz à tona uma discussão mais relacionada a capacidade de reutilização de software no nível de projeto arquitetural. A reutilização de soluções arquiteturais acabou sendo uma das principais conseqüências da formalização da área de projeto arquitetural. Como citou Prieto-Díaz, em (PRIETO-DÍAZ e FREEMAN, 1987), as arquiteturas de software podem ser vistas como as principais unidades reutilizáveis no nível de projeto, de acordo com a taxonomia que se baseia no reuso de produtos de software. Convém observarmos, porém, que uma arquitetura de software por si só não é facilmente reutilizada, como pode ser imaginado inicialmente. Arquiteturas reutilizáveis exigem um alto grau de esforço, técnicas apropriadas e níveis de abstração que permitam a preparação de estruturas genéricas o suficiente para serem reutilizadas.

No contexto de reutilização em um domínio específico de aplicação, as arquiteturas de software mostram ser de uma importância significativa. As arquiteturas especialmente preparadas para a reutilização, também conhecidas como arquiteturas de referência de domínio, são os artefatos que efetivamente realizam, através da inter-relação de seus componentes, as funções essenciais do domínio, e dão sustentação às características de qualidade que advêm do projeto arquitetural realizado (CHUNG *et al.,*1994). Observamos, no entanto, que o desenvolvimento de aplicações através da reutilização de arquiteturas implica em conseqüências na natureza dos produtos de software, sendo estas muito negativas se o projeto arquitetural de referência tiver sido mal conduzido (BOSCH, 2000).

As iniciativas de apoio à reutilização vertical de software são caracterizadas pela identificação e desenvolvimento de artefatos de software que apoiarão a construção de aplicações neste mesmo domínio (GACEK, 1995, MEEKEL *et al.*, 1997). Pela complexidade envolvida, o processo utilizado para se chegar às arquiteturas de domínio deve ser sistematizado, envolvendo atividades específicas e procedimentos que apóiem a construção de uma representação arquitetural que possibilite a produção de software em larga escala.

A atividade de projeto arquitetural, tanto no contexto de aplicações específicas quanto para a produção de arquiteturas de referência, baseia-se na conversão de um conjunto de requisitos na arquitetura de software que os preenche, ou ao menos facilita o seu preenchimento. No entanto, verificamos que esta atividade por muito tempo vem se baseando em um processo de construção focado principalmente na obtenção das funções que devem ser suportadas pelo software (CHUNG *et al.,* 1999). Percebemos, contudo, que na prática a atividade de projeto arquitetural baseado apenas na função não é suficiente para garantir que um software será bem avaliado

pelos seus patrocinadores. Produtos de software que sejam lentos, difíceis de operar ou que não tenham níveis de robustez aceitáveis dificilmente serão aceitos, mesmo que façam exatamente o que foi especificado nos requisitos funcionais. Isto se dá em grande parte porque a arquitetura planejada não foi preparada para atender algumas das características de qualidade de natureza não funcional. Tal problema assume proporções maiores quando relacionados ao reuso de arquiteturas de referência. Com o reuso, as características de qualidade relacionadas à determinada arquitetura de referência são incorporadas às aplicações produzidas por esse processo, representando um impacto maior sobre os níveis de aceitação, agora não apenas sobre um produto, mas de todo um conjunto de aplicações.

Tal constatação apenas fundamenta a importância da arquitetura para a relação existente entre qualidade e projeto de software. No entanto, é difícil preparar soluções arquiteturais que suportem requisitos explícitos de qualidade. Talvez isso ocorra porque os desenvolvedores em grande parte desconhecem a importância desta relação e as possíveis formas de exploração de alternativas arquiteturais para um problema em mãos.

Consideramos a iniciativa dos padrões arquiteturais[1] (BUSCHMANN *et al.*, 1996) como uma das possibilidades atuais de minimização das dificuldades em alcançarmos bons projetos arquiteturais. Mesmo com os benefícios associados à reutilização deste conhecimento, constatamos que muito poucos trabalhos exploram a possibilidade de aplicá-los em contextos de infra-estruturas de reutilização de software, e, mesmo mencionando-a, não a descrevem com maior nível de detalhe (TRACZ e HAYES, 1994, GOMAA e FARRUKH, 1999, MANNION *et al.,* 1999). Acreditamos que isso ocorre porque é difícil utilizar os padrões para apoiar uma visão qualitativa dos objetivos de um projeto de software. Ou seja, de que forma um determinado conjunto de características de qualidade poderá ser melhor suportado tendo como referência algumas soluções arquiteturais recorrentes.

1.2 Objetivos da Tese

O objetivo desta tese é prover um detalhamento das principais atividades relacionadas aos processos de criação e instanciação de arquiteturas de referência para a reutilização de software em domínios de aplicação particulares, explorando a possibilidade de apoio automatizado a alguma destas atividades. Neste contexto,

[1] Um padrão arquitetural nomeia, abstrai e identifica os principais aspectos comuns a um projeto arquitetural particular, tornando-os disponíveis para serem utilizados durante a criação de novos projetos.

percebemos que a definição da arquitetura de referência é fundamental para que os requisitos funcionais e de qualidade sejam estendidos às aplicações construídas em um domínio particular. Os processos de criação e a instanciação da arquitetura de referência estarão apoiados na aplicação de padrões arquiteturais visando a modelagem da arquitetura de referência e a adequação desta arquitetura no contexto da aplicação, respectivamente.

Nosso trabalho busca atender a infra-estruturas de reutilização de software que possuam alguma ênfase no projeto arquitetural de domínio e de aplicações. Porém, para que seja concretizado este detalhamento, utilizamos especificamente a infra-estrutura Odyssey, em desenvolvimento na COPPE/UFRJ.

Este trabalho explora particularmente a possibilidade de incorporar tanto uma atividade de especificação de características arquiteturais de referência quanto a utilização de padrões arquiteturais, reconhecidos pelos seus benefícios (BUSCHAMNN *et al.*, 1996), no contexto de infra-estruturas de reutilização.

Propomos a utilização de uma abordagem de seleção dos padrões arquiteturais que destaca as potenciais alternativas de projeto arquitetural para o suporte à obtenção de características arquiteturais desejadas. A definição desta abordagem possibilitou o desenvolvimento de uma ferramenta que lhe dá suporte.

Apresentamos, por fim, a elaboração de um processo de avaliação dos padrões arquiteturais, visando extrair, junto a desenvolvedores de software, um nível de conhecimento que possibilite a calibração contínua do conhecimento utilizado pela abordagem de seleção, visando apoiar a sugestão de alternativas de projeto condizentes com a experiência em projeto arquitetural.

1.3 Organização da Tese

Esta tese está dividida em seis capítulos e três anexos, incluindo este capítulo de introdução, a saber:

Capítulo 2 - Arquiteturas de Software – onde são apresentadas as características relacionadas às arquiteturas de software. Em seguida, são apresentados os principais estilos e padrões arquiteturais encontrados na literatura. Por fim, são descritas as principais áreas de pesquisa que influenciaram este trabalho.

Capítulo 3 - Arquiteturas de Software Específicas de Domínios no Contexto de uma Infra-estrutura de Reutilização – onde descrevemos o processo de suporte a criação e instanciação de arquiteturas de referência em domínios de aplicação particulares para o contexto da infra-estrutura Odyssey.

Capítulo 4 - A Abordagem de Seleção dos Padrões Arquiteturais – onde são apresentadas a abordagem que permite a indicação e seleção dos padrões arquiteturais e a ferramenta que suporta a sua realização.

Capítulo 5 - O Processo de Avaliação Contínua dos Padrões Arquiteturais – onde apresentamos a organização dada ao estudo que procurou viabilizar o processo de avaliação contínua dos padrões arquiteturais e os resultados obtidos com a sua realização.

Capítulo 6 - Conclusão – onde são apresentadas as principais contribuições da tese, comparações com outras abordagens, suas restrições, e ainda, os trabalhos futuros.

Anexo I - Questionário de Avaliação de Soluções Arquiteturais – onde são apresentados os formulários utilizados durante a primeira organização dada ao processo de avaliação contínua dos padrões arquiteturais.

Anexo II - Exercício de Avaliação de Soluções Arquiteturais – onde são apresentados os formulários utilizados durante a segunda organização dada ao processo de avaliação contínua dos padrões arquiteturais.

Anexo III - Descrição das características e sub-características de qualidade – onde são descritos os elementos da ISO/IEC 9126 (ISO9126, 1992) que são utilizados na abordagem.

Capítulo 2: Arquiteturas de Software

2.1 Introdução

O Brasil comemorou os 500 anos de descobrimento em 22 de abril de 2000. Para coroar esta festa, o governo federal patrocinou o projeto de construção de uma réplica da nau do descobrimento. A réplica em tamanho natural da embarcação usada pela esquadra portuguesa teve seu projeto orçado em dois milhões de dólares e deveria estar totalmente pronta na data do descobrimento, onde faria sua viagem inaugural de Valença, onde foi construída, até Porto Seguro, ambas cidades do estado da Bahia. Com capacidade para até 30 pessoas e 16 canhões, a nau foi projetada para navegar com velocidade máxima de 10 nós/h (18,5Km/h) e, apesar da aparência antiga, seu interior foi planejado para utilizar muitos equipamentos da moderna tecnologia náutica.

No entanto, constatamos que, em situação inversa, o Brasil dificilmente descobriria Portugal. Com quatro meses de atraso, a nau apresentou uma série de problemas, que, em grande, foram atribuídos a erros no projeto inicial. O que seria o símbolo das comemorações custou quase o dobro do previsto e quase afundou depois de apenas 8 horas de viagem inaugural.

O exemplo da nau só reforça a importância de um bom projeto para a engenharia de qualquer produto. A construção de produtos que efetivamente funcionem e apresentem o nível de qualidade esperado não é uma tarefa fácil. Mesmo sendo um elemento de sistema lógico, e não físico, o software também depende de um bom processo de construção para ter qualidade.

O processo de desenvolvimento de software está sujeito à aplicação de métodos, ferramentas e procedimentos (PRESSMAN, 1995) e é, fundamentalmente, uma atividade que envolve muitas pessoas com diferentes habilidades e interesses. O desenvolvimento da maioria dos sistemas baseados em software envolve a criação e evolução de vários artefatos, tais como diagramas, arquivos de código-fonte, de teste, manuais de uso, documentos de projeto e de usuário. Cada um destes artefatos descreve um aspecto diferente do sistema e atende aos interesses dos diferentes membros envolvidos no desenvolvimento, aquisição e uso do software.

Independentemente do ciclo de vida adotado, o processo de desenvolvimento de software é caracterizado por seis fases principais (também chamados subprocessos) que são a captura de requisitos, a análise, o projeto, implementação, teste e implantação. Ao longo deste ciclo de vida, a equipe desenvolvedora amplia a

compreensão do sistema e descreve suas funções e a estrutura principal do software em crescentes níveis de detalhes. Das atividades citadas acima, é durante a fase de projeto do software que há a transformação do problema em uma solução (PFLEEGER, 1998), ou seja, quando a equipe desenvolvedora (o projetista do software em particular) define um sistema com detalhes suficientes para permitir sua realização física. Porém, uma dificuldade comum ao desenvolvimento de software é a realização da transformação dos requisitos identificados na fase de análise na solução proposta no projeto da aplicação. Bosch, em (BOSCH, 2000), afirma que esta atividade é a mais complexa de ser realizada durante o desenvolvimento de aplicações, porque envolve a passagem do contexto do problema para a realização da solução. Com o aumento da complexidade dos problemas, os engenheiros de software depararam-se com soluções não menos difíceis de serem realizadas.

Vimos que uma forma de se minimizar a problemática de administração da complexidade é realizada através da formalização de um nível de abstração que represente o contexto global da aplicação através das interligações existente entre os seus principais artefatos. A *arquitetura de software* de um programa, ou sistema de computação, é a estrutura (ou estruturas) do sistema que define preliminarmente uma solução particular e orienta as fases de projeto detalhado e de implementação do software (PFLEEGER, 1998). A arquitetura está relacionada com os componentes do software, as propriedades externamente visíveis destes componentes e o relacionamento existente entre os mesmos (BASS *et al.*, 1998).

O objetivo deste capítulo é discutir os aspectos relativos a área de pesquisa sobre arquiteturas de software. O capítulo está organizado da seguinte forma: no item 2.2, são descritas as principais características das arquiteturas de software, ressaltando sua origem, definições e o papel do arquiteto de software; no item 2.3, são apresentados os principais benefícios encontrados com a sua utilização; no item 2.4, descrevemos um enfoque arquitetural para processos de desenvolvimento; no item 2.5, descrevemos o conceito de estilo arquitetural e, em seguida no item 2.6, analisamos a aplicação de padrões arquiteturais; no item 2.7, apresentamos algumas heurísticas para seleção de estilos e padrões arquiteturais encontradas na literatura; no item 2.8, são apresentados os principais trabalhos referentes a arquiteturas de software; por fim, o item 2.9, conclui o capítulo.

2.2 As Características das Arquiteturas de Software

Como foi dito anteriormente, o projeto de software é a atividade realizada durante o desenvolvimento de software onde ocorre a definição de uma solução computacional com detalhes suficientes para permitir sua realização. Fundamentalmente, o projeto de software é uma atividade que envolve uma sistemática de decomposição da solução, a começar pela descrição em mais alto nível dos principais elementos do sistema e, em seguida, criando uma visão mais detalhada de como as características e funções deste sistema deverão estar integradas (PFLEEGER, 1998).

Projetistas de sistemas de software vêm, ao longo dos anos, reconhecendo a importância de se representar e explorar o conhecimento obtido durante a construção de novos sistemas. Desde a década de 70, esta atividade vem sendo impulsionada pela comunidade desenvolvedora, em resposta aos problemas identificados durante a produção de sistemas de complexidade e escala maiores que os anteriormente produzidos (BROOKS, 1975). As premissas identificadas foram as de que o projeto de software deveria ser uma atividade separada da implementação, requerendo ainda notações, técnicas e ferramentas especiais (BERGLAND, 1981).

Várias abordagens distintas, porém relacionadas, almejavam suportar o conceito de "*programming-in-the-large*"[2] (PARNAS, 1972). Uma das dificuldades encontradas na fase de projeto, para o tratamento da complexidade de sistemas, era lidar com aspectos macros, tendo um enfoque de solução centrado na especificação e escolha de algoritmos e estruturas de dados. Identificou-se, então, a necessidade de se especificar um nível que tratasse principalmente da topologia dos componentes, formalizando uma representação uniforme do conjunto (SHAW, 1989). Neste nível, estaria descrita a *arquitetura do software*.

A arquitetura de um software identifica um conjunto de componentes que colaboram para atingir os propósitos do sistema. A arquitetura especifica as propriedades externamente visíveis dos componentes e define as ligações entre os mesmos – ou seja, que "noção ou conclusão" outros componentes podem ter de um componente, tais como os serviços fornecidos, e de que forma estas interligações estão restritas por uma "forma ou topologia" particular (BASS *et al.*, 1998).

Encontramos na literatura algumas definições relacionadas ao termo arquitetura de software. Em (SHAW e GARLAN, 1993), os autores definem a arquitetura de um software como a descrição de elementos que construirão o sistema,

[2] *Programming-in-the-large* é a expressão utilizada para designar o desenvolvimento de software de tamanho elevado, onde é necessária a participação de muitas pessoas por um longo período de tempo.

interações entre estes elementos, padrões que guiarão suas composições, e limitações destes padrões. Em geral, um sistema é definido em termos de uma coleção de componentes e interações entre estes componentes. Tal sistema pode, ainda, ser usado como um elemento de composição em um projeto de sistema maior.

Muito próximo disto, CLEMENTS (1996) define a arquitetura de software como "a estrutura dos componentes do sistema, seus inter-relacionamentos, princípios e diretrizes que governam seu projeto e evolução ao longo do tempo".

Em (GACEK *et al.,* 1995), encontramos uma definição de arquitetura de software que preza os interesses de quem a utiliza. Diferentes interesses geram diferentes pontos de vista sobre a mesma arquitetura. Os autores sintetizam bem o termo quando descrevem a arquitetura do software como:

1. Uma coleção de componentes de software e de sistemas, e restrições;
2. Uma coleção de declarações sobre os interesses envolvidos; e
3. Um conjunto de razões que demonstra o contexto que motivou determinada definição do sistema pela utilização de componentes, conexões, e restrições particulares.

Em (STAA, 2000), o autor descreve que arquitetura é o processo de organização de um programa em termos dos módulos[3] que os constituem e o seu resultado é a arquitetura do programa. Staa ressalta a idéia de que a arquitetura não deixa de ser uma forma de projeto, utilizada para estabelecer a organização global do programa. Na visão do autor, a arquitetura de um programa define uma estrutura de dependência entre módulos, especificando para cada um dos módulos o seu objetivo, a sua interface e a qualidade requerida. O processo de organização arquitetural deve almejar o desenvolvimento de uma solução que minimize a dependência entre os módulos, atendendo as funções especificadas para o programa, e deve ainda se preocupar em produzir módulos que possam ser reutilizados em diversos programas.

É comum encontrarmos representações da arquitetura de sistemas através do uso de diagramas que privilegiam a aplicação de símbolos gráficos, mesmo que isso se faça de maneira informal. A notação gráfica adotada baseia-se na utilização de símbolos pré-definidos, onde caixas descrevem os principais componentes dos sistemas, e linhas representam alguma comunicação, controle, ou relacionamento de dados entre estes componentes. Infelizmente, descrições e diagramas como estes são

[3] Um módulo é uma unidade de compilação, formado por um ou mais arquivos de texto fonte necessários para que possam ser compilados. O código fonte de um módulo é formado por diversos elementos, tais como: classes, funções, declarações de tipos, variáveis globais, tabelas (STAA, 2000).

ambíguos, pois permitem interpretações conflitantes e representações pessoais (DEWAYNE e ALEXANDER, 1992). Além disso, não havia até muito pouco tempo atrás conceitos adequados, ferramentas e critérios de decisão que permitissem a seleção e descrição de estruturas que se adequassem ao problema em mão (SHAW e GARLAN, 1996).

A necessidade da aplicação de um maior formalismo das representações estruturais de sistemas e o estudo dos possíveis benefícios alcançados e conseqüências diretas de sua aplicação levaram a arquitetura de software ao status de disciplina, sendo uma área de pesquisa que tem recebido grande atenção recentemente. Esta área de estudo engloba alguns fatores estruturais tais como a organização e decomposição macroscópica, a estrutura de controle, os protocolos para comunicação, sincronização e acesso aos dados, atribuição de funcionalidade e composição dos elementos de projeto, distribuição física, propriedades globais do sistema e qualidades arquiteturais e seleção de alternativas de projeto (SHAW *et al.*, 1995).

Com a ampliação do interesse sobre o projeto arquitetural de software, os centros de pesquisas e as organizações perceberam que uma série de habilidades específicas eram exigidas dos desenvolvedores para o trabalho com as arquiteturas de software. A figura do arquiteto de software é uma conseqüência direta desta comprovação, mas as funções, habilidades e responsabilidades deste novo papel ainda é motivo de debate (MALAN e BREDEMEYER, 2000). A primeira visão do papel exercido pelo arquiteto de software é a de que este cria arquiteturas e suas responsabilidades englobam a articulação da visão arquitetural e das possíveis alternativas para o problema, a criação de documentos que representem a arquitetura através de modelos, componentes e interfaces, e a validação do projeto arquitetural em função dos requisitos esperados para a aplicação. Porém, há uma série de quesitos que estão acima das questões técnicas e que também fazem parte das responsabilidades do arquiteto, tais como habilidades de negociação, estratégia e consultoria, e tais responsabilidades são importantes para que as arquiteturas possam ser compreendidas e aceitas pelos diversos papéis envolvidos no processo de desenvolvimento de software.

2.3 Benefícios Encontrados

O projeto arquitetural de sistemas complexos possui um papel fundamental para o sucesso de todo o processo de desenvolvimento porque seu principal objetivo é

realizar a associação entre as capacidades identificadas do sistema aos componentes arquiteturais que irão implementá-las (PLEEGER, 1998). Mesmo assim, o projeto arquitetônico tradicional de um sistema tem sido muitas vezes tratado informalmente.

O uso de uma formalização de princípios para descrição da arquitetura traz vantagens significativas para o processo de desenvolvimento de software. Diversos autores (DEWAYNE e ALEXANDER, 1992, ABOWD *et al.*, 1993, CLEMENTS, 1996) citam algumas destas vantagens:

- *Compreensão:* A arquitetura do software simplifica o entendimento de sistemas complexos através de níveis mais abstratos, de maneira que todo o sistema possa ser compreendido, reduzindo também a distância semântica existente entre a especificação dos requisitos e a programação;
- *Reutilização:* A descrição arquitetural permite a reutilização em múltiplos níveis. As arquiteturas de software baseadas em domínios específicos de aplicação (DSSA)[4] e os *frameworks*[5] são algumas evidências dos níveis de reutilização encontrados.
- *Evolução:* Através da arquitetura do software podemos encontrar a dimensão que uma aplicação pode evoluir, sem que ocorram os problemas de violação e instabilidade da arquitetura.

A principal vantagem da especificação de uma arquitetura de software, descrita em (ABOWD *et al.,* 1993), é que esta descreve o sistema através de um nível mais alto de abstração, permitindo que sejam visualizadas as decisões referentes ao projeto o mais cedo possível, ajudando o desenvolvedor a verificar se os requisitos foram realmente atingidos. Através da abstração dos detalhes de implementação, uma boa descrição arquitetural facilita a gerência do projeto do sistema e expõe as propriedades mais cruciais para o seu sucesso.

2.4 O Processo de Desenvolvimento e a Arquitetura do Software

Sistemas surgem devido a inúmeras necessidades: oportunidades de negócios, suporte automatizado a tarefas, acesso e disponibilização de informações. Seja qual for o motivo, o desenvolvimento do software passa por etapas bem definidas e em geral segue algum tipo de orientação.

[4] Um DSSA representa uma abordagem de reutilização de software que objetiva o suporte a geração de aplicações em um domínio particular (GACEK, 1995).

[5] Um *framework* é um projeto reutilizável de um programa ou parte de um programa expresso como um conjunto de classes (JOHNSON, 1988).

Cada projeto de desenvolvimento pode estar centrado em um ou mais fatores que o influenciam diretamente. Existem projetos que sofrem sérias restrições de prazos e, por isso, em geral seu enfoque é direcionado na obtenção de um produto no tempo acordado. Outros possuem uma ênfase maior na documentação a ser produzida, orientando desta forma o esforço de desenvolvimento a esta necessidade.

Projetos centrados na arquitetura do sistema são uma forma de orientação do desenvolvimento de software (BOOCH, 1996). Estes projetos orientam um desenvolvimento de software caracterizado pelo foco na criação de uma estrutura de componentes que satisfaça os requisitos do sistema mais significativos de serem alcançados e que seja suficientemente flexível para se adaptar aos requisitos não conhecidos ou não bem compreendidos da aplicação.

Em (DEWAYNE e ALEXANDER, 1992, HP, 2000), encontramos idéias gerais da caracterização de um enfoque arquitetural dentro do contexto de desenvolvimento de software. Para a simplificação desta análise, os autores caracterizam as fases do desenvolvimento do software a partir de uma abordagem seqüencial inspirada no modelo cascata:

- A *captura dos requisitos* está relacionada à determinação e caracterização das funções do sistema, além do contexto em que a aplicação estará inserida e dos requisitos não funcionais inerentes ao problema. Os requisitos esperados nunca são fixos, e um dos desafios desta fase é justamente identificar os requisitos mais significativos que poderão influenciar diretamente e que devem ser suportados pela arquitetura do software;
- A *arquitetura do software* se relaciona com a seleção dos elementos arquiteturais, suas interações e limitações, para a produção de uma estrutura que satisfaça os requisitos e sirva como base para a fase de projeto. Uma vez definida uma arquitetura, esta deve ser clara e sem ambigüidades para que possa ser entendida por todos os seus usuários.
- O *projeto* abrange aspectos de modularização e detalhamento das interfaces dos elementos de projeto, seus algoritmos e procedimentos, e os tipos de dados necessários para suportar a arquitetura e satisfazer os requisitos. Nesta fase, é que ocorre a translação dos requisitos e da arquitetura em níveis de projeto mais próximos da implementação;
- A *implementação (codificação/ integração)* relaciona-se com as representações dos algoritmos e dos tipos de dados que satisfazem o projeto, a arquitetura e os requisitos do software;

- Os *testes* concentram-se nos aspectos lógicos internos do software e nos aspectos funcionais externos; e
- A *manutenção*, que reaplica cada uma das etapas precedentes do ciclo de vida ao programa existente, e não a um novo. Porém, deve-se manter ao máximo a arquitetura existente do software, para que não se desfigure os elementos da configuração estabelecida para o sistema.

Segundo (GACEK *et al.*,1995), o ciclo de vida em espiral é o que melhor se adequou ao processo de desenvolvimento com ênfase na arquitetura, pois permite a identificação incremental dos requisitos da aplicação, dos riscos, restrições e objetivos. No modelo em espiral, na medida em que requisitos são inseridos, existe a possibilidade de atualização do projeto, evitando que a arquitetura não atenda aos requisitos da aplicação.

Cabe ainda aqui a discussão sobre o ciclo de desenvolvimento de um software com ênfase na arquitetura e os papéis envolvidos neste processo. É importante compreender de que forma a arquitetura pode atender a tantos interesses. Normalmente, muitos papéis são assumidos por uma ou mais pessoas neste processo, sendo comum encontrarmos pessoas que exerçam mais de um desses papéis durante o desenvolvimento. Para que possamos diferenciar bem estes posicionamentos, é importante caracterizá-los através dos interesses em relação ao software e, para o nosso propósito, em relação a sua arquitetura.

Os principais envolvidos no ciclo de desenvolvimento de uma aplicação e seus respectivos interesses em relação à arquitetura são:

- *Cliente* – O cliente é a pessoa ou empresa que contrata uma equipe de desenvolvimento para construção de um sistema de sua necessidade. O cliente não necessariamente é o usuário deste software, porém nem por isso seu interesse é menor. O cliente espera uma estimativa de certos fatores, normalmente econômicos, uma vez que a estrutura principal do software está definida; por exemplo, o cliente tem interesse em estimativas de custo, confiabilidade e manutenibilidade do software baseado nesta arquitetura. Procura ainda um controle dos orçamentos e prazos envolvidos. Para o cliente, é muito importante que a arquitetura esteja fortemente associada aos requisitos do software, de forma a representar suas reais expectativas em relação ao software.
- *Usuário* – O usuário de um software é aquele que efetivamente utilizará o sistema a ser desenvolvido. Questões de desempenho, usabilidade, e

interoperabilidade entre sistemas são as mais relevantes e devem de alguma forma ser representadas pela arquitetura.

- *Engenheiro e Arquiteto de Sistemas* – O principal interesse aqui é a translação dos requisitos no projeto preliminar da aplicação. Tanto o arquiteto quanto o engenheiro da aplicação podem usar a arquitetura como elemento de entendimento e negociação dos requisitos do sistema.
- *Desenvolvedor* – Da arquitetura de um software, o desenvolvedor busca uma especificação que seja suficiente em detalhes e que satisfaça os requisitos do cliente, mas não tão restritiva que impeça abordagens diferentes para a sua implementação. Os desenvolvedores então usam a arquitetura como uma referência para a composição e desenvolvimento dos componentes do sistema, e para a reutilização de componentes existentes.
- *Mantenedor* – Os aspectos mais importantes de seu interesse são o grau de facilidade de extensão e modificação do software, dada sua estrutura de alto nível. A descrição arquitetural do software fornece aos mantenedores uma estrutura central da aplicação que, idealmente, não deve ser violada. Qualquer mudança deve preservá-la, buscando, se possível, uma modificação puramente dos componentes da aplicação.

2.5 Estilos Arquiteturais

As arquiteturas são formadas por elementos e formas que as caracterizam particularmente. Cada elemento possui um tipo e cada forma representa um relacionamento entre estes elementos. Ao definirmos arquitetura como uma combinação formal de elementos arquiteturais, então um estilo arquitetural será visto como um modelo, composto de elementos abstratos e aspectos formais, oriundos de várias arquiteturas. Em (SHAW e GARLAN, 1996), encontramos a definição de estilo arquitetural como um conjunto de regras que identificam os tipos de componentes e conectores que podem ser usados para compor um sistema ou subsistema, junto a restrições locais ou globais à forma que a composição é feita. Um estilo arquitetural é menos rígido e completo do que uma arquitetura específica.

A vantagem de se trabalhar com os estilos arquiteturais é que estes encapsulam decisões importantes sobre os elementos da arquitetura, seus relacionamentos e suas restrições. Sua utilidade também se estende à coordenação entre os desenvolvedores responsáveis pela arquitetura do software.

Um número de estilos mais abrangentes tem sido identificado em descrições de idiomas arquiteturais (SHAW e GARLAN, 1996, GACEK, 1998). A tabela 1.3 representa os estilos arquiteturais mais comumente utilizados.

Pipes & Filters Transformadores incrementais de cadeias. Ex. Pipes do Unix, processamento de sinais, compiladores tradicionais.
Cliente-Servidor Serviços compartilhados, fornecidos através da solicitação dos clientes distribuídos; Ex. Servidores de arquivos, banco de dados distribuídos.
Camadas hierárquicas Sistemas particionados em camadas, que atuam como máquinas virtuais. Ex. Núcleos de S.O., ISO OSI.
Processos de comunicação Sistema é composto de processos independentes e concorrentes. Ex. Muitos sistemas distribuídos.
Interpretadores Ligações entre programas abstratos e as máquinas sobre as quais devem rodar. Ex. Sistemas baseados em regras.
Baseado em eventos e Invocação implícita A interação entre os componentes é realizada através do anúncio de um ou mais eventos. Ex. Mecanismos de invocação.
Repositório Sistemas que possuem um repositório de informações compartilhadas. Ex. Ambientes de programação com repositório de programas.
Organização em programa principal e subrotinas Estabelecimento de hierarquias de dados e procedimentos. Ex. Programas estruturados
Orientados a um domínio específico Customização de uma estrutura para uma família de aplicações. Ex. Arquiteturas desenvolvidas para domínios de aviação.
Organização orientada a objetos ou de abstração de dados Encapsulação de dados e operações em um objeto ou ADT. Ex. ORB, OLE, OpenDoc.

Tabela 1.3 - Estilos arquiteturais mais conhecidos

As principais contribuições dos estilos arquiteturais para o processo de desenvolvimento, em particular para o projeto de software são:

1. Os estilos possibilitam a utilização de um vocabulário de elementos de projeto (tipos de componentes e conexões), tais como “pipes”, repositório de dados, chamadas de subrotinas, “sockets”, etc;

2. Definem regras de configuração (ou restrições topológicas) que determinam as composições pertinentes dos elementos;

3. Definem uma semântica de interpretação, onde composições de elementos de projeto, apropriadamente restritos pelas regras de configuração, possuem significados bem definidos; e

4. Definem análises que podem ser realizadas nos sistemas que lhes são baseados. Porém, a natureza da análise está muito ligada ao estilo arquitetural em questão. Por exemplo, é comum a análise de “throughput”[6] e “deadlock”[7] de arquiteturas baseadas no estilo “Pipe-&-Filter”, porém este tipo de análise é de pouca utilidade em arquiteturas construídas na maiorias dos estilos restantes.

De forma complementar a estas características, cada estilo traz consigo ainda informações sobre as especializações mais comuns e suas vantagens e desvantagens em relação a determinadas propriedades, tais como reutilização, desempenho e manutenção. O que percebemos, porém, é que não há uma definição comum de propriedades que pudesse ser utilizada na avaliação mais apurada da aplicação dos diferentes estilos. Por exemplo, se as vantagens e desvantagens de se aplicar um estilo fossem descritas tendo como referência um conjunto de características de qualidade desejadas, poder-se-ia ter uma melhor avaliação dos estilos para o alcance de determinados atributos das aplicações.

2.6 Padrões Arquiteturais

Um padrão arquitetural é uma abordagem descrita por (BUSCHMANN *et al.,*1996) que tem o objetivo de apoiar o desenvolvimento de sistemas orientado a objetos, através da ênfase na organização estrutural do software. Cada padrão fornece um conjunto de subsistemas pré-definidos, especifica suas responsabilidades,

---

[6] “Throughput” representa o volume de trabalho que um computador pode realizar em um dado período de tempo (TANENBAUM, 1995).

[7] Um processo é dito em “deadlock” quando espera por um evento que nunca ocorrerá. Esta situação é conseqüência, na maioria das vezes, do compartilhamento de recursos do sistema entre vários processos, sendo que cada processo deve ter acesso ao recurso de forma exclusiva. (TANENBAUM, 1995).

e inclui regras e orientações para a organização dos relacionamentos entre os subsistemas.

Podemos pensar em um padrão como a reutilização da essência de uma solução para determinados problemas similares. O par problema-solução tende a categorizar famílias de problemas e soluções, onde o padrão torna-se o elo de ligação entre os dois contextos. Um padrão arquitetural lida com problemas de projeto estrutural recorrentes e que aparece em situações de projeto específicas, apresentando ainda uma solução para isso.

Embora um padrão determine uma estrutura básica da solução para um problema de projeto particular, este não especifica a solução em um nível de detalhe profundo. O que este fornece é um esquema de solução genérica para uma família de problemas, cabendo ao projetista a adaptação da solução ao problema específico. Eles ajudam a solução do problema, mas não oferecem a solução completa.

Embora alguns trabalhos apresentem os estilos e os padrões arquiteturais como sinônimos (BASS *et al.*, 1998, BOSCH, 2000), na medida que ambos enfocam a representação arquitetural do software através de seus principais componentes, ligações e topologias, encaramos os padrões arquiteturais apresentados por (BUSCHMANN *et al.*, 1996) como um nível mais específico da aplicação destes elementos arquiteturais ao projeto de software. Segundo (SHAW e GARLAN, 1996), a organização orientada a objetos é um estilo arquitetural possível de ser aplicado à representação arquitetural do software. Com essa definição, percebemos que todos os padrões arquiteturais por (BUSCHMANN *et al.*, 1996) são representações particulares deste estilo arquitetural. Desta forma, sob o ponto de vista da generalidade, consideramos que os padrões arquiteturais têm o escopo de aplicação reduzido ao desenvolvimento de software OO, e esta diferença deve ser levada em conta, por exemplo, durante a seleção de determinado estilo ou padrão arquitetural.

Os principais padrões arquiteturais encontrados em (BUSCHMANN *et al.*, 1996) são:

1. *Camadas* (*Layers*): Padrão arquitetural que propõe a decomposição de aplicações em grupos de subtarefas, onde cada grupo se apresenta em um nível particular de abstração;
2. *Pipes & Filters*: Este padrão arquitetural suporta a divisão de uma função ou processo em várias etapas de processamento seqüenciais. Cada etapa recebe, processa e disponibiliza uma cadeia de dados e o fluxo de saída de uma etapa corresponde ao fluxo de entrada da etapa seguinte;

3. *Blackboard*: Neste padrão, vários subsistemas organizam o conhecimento para a construção de uma possível solução aproximada ou parcial através do uso de uma memória compartilhada. Cada subsistema é especializado para solucionar uma fase particular da tarefa completa, e todos os subsistemas trabalham juntos para a aquisição da solução. Tais subsistemas não interagem diretamente entre si e nem há uma seqüência determinada para as suas ativações. Em vez disto, a direção que o sistema toma é, principalmente, determinada pelo estado corrente da aplicação. A principal variação deste padrão é o Repositório, uma generalização onde não há a especificação de um componente responsável pelo controle interno da aplicação. Sistemas que utilizam banco de dados tradicionais podem ser considerados como exemplos de repositórios;

4. *Broker*: Pelo uso deste padrão, uma aplicação pode acessar serviços de outras aplicações simplesmente pelo envio de mensagens a objetos mediadores sem se preocupar com questões específicas relacionadas à comunicação entre processos (acoplamento, localização, transmissão de dados, etc.);

5. *Model-View-Controler* (MVC): Propõe a divisão da aplicação em três categorias principais de componentes. O componente “modelo” encapsula dados e funcionalidade principais da aplicação. O componente “visão” mostra informações ao usuário obtidas de um modelo. Cada visão apresenta um componente “controlador” que recebe as entradas do usuário (na forma de eventos disparados) e os traduz em solicitações de serviços ao modelo e/ou visão. O modelo é independente de suas possíveis representações, bem como do comportamento de entrada, e pode apresentar múltiplas visões;

6. *Presentation-Abstraction-Control* (PAC): Define uma estrutura para sistemas na forma de uma hierarquia de agentes cooperativos. Cada agente é responsável por um aspecto específico da funcionalidade da aplicação e é composto por três componentes: apresentação, abstração e controle;

7. *Microkernel*: Propõe a separação de um conjunto de funcionalidade mínimo das funcionalidades estendidas e partes específicas de clientes. O encapsulamento dos serviços fundamentais da aplicação deve ser realizado no componente “microkernel”. As funcionalidades estendidas e específicas devem ser distribuídas entre os componentes restantes da arquitetura;

8. *Reflection*: Uma arquitetura que é dividida em duas partes principais: o nível meta, que fornece uma representação do software para o autoconhecimento da estrutura e comportamento do sistema através de componentes chamados “metaobjetos”, e o nível base, que define a lógica da aplicação, onde a implementação

do sistema utiliza os metaobjetos. Uma interface é especificada para a manipulação dos metaobjetos através de um protocolo de metaobjetos;

Todo padrão arquitetural é descrito por um esquema de três partes gerais que segue uma orientação similar à documentação de padrões em geral (APPLETON, 1997):

1. Contexto: Descreve as situações em que um problema ocorre. A escolha do contexto correto de um padrão é tarefa difícil. Em (BUSCHMANN *et al.,*1996) os autores acham quase impossível determinar todas as situações, tanto genéricas quanto específicas, em que um padrão pode ser aplicado. Uma forma mais prática seria listar as situações mais relevantes onde um problema que é referenciado por um padrão pode ocorrer. Todavia, esta abordagem não garante que serão cobertas todas as situações relevantes, visto a larga amplitude de estados que um contexto pode assumir, mas ao menos fornece uma orientação valiosa das possibilidades de utilização do padrão.

2. Problema: O problema recorrente que aparece em um determinado contexto. Descreve uma especificação geral do problema, capturando sua essência. A declaração geral do problema é composta também pelo conjunto de *forças*. O termo força é descrito pela comunidade que utiliza os padrões como sendo "todo aspecto do problema que deve ser considerado durante sua solução" (APPLETON, 1997), tais como os requisitos de solução, restrições que devem ser consideradas e propriedades desejáveis que a solução deve possuir. Em geral, os itens das forças discutem o problema de pontos de vista variados e permitem a sua compreensão mais detalhadamente.

3. Solução: Aqui encontramos a descrição de como solucionar o problema recorrente, ou em uma forma mais efetiva, como balancear as forças associadas ao problema. Na solução de questões arquiteturais, cada padrão especifica uma certa estrutura, uma configuração espacial dos elementos. Esta estrutura enfoca aspectos estáticos da solução. Um padrão arquitetural deve especificar ainda um comportamento dinâmico da solução, pela descrição da colaboração e comunicação de seus componentes.

BUSCHMANN e MEUNIER (1994) salientam que cada padrão pode ser classificado de acordo com uma visão orientada à natureza dos problemas específicos que podem surgir durante o desenvolvimento do projeto do software, tais como a gerência do paralelismo do processamento, a distribuição das partes que compõem o

software e a apresentação de uma interface gráfica adequada às necessidades de uma categoria de usuários. Sob esta classificação, temos os padrões organizados em categorias, cada uma representando uma característica central do projeto. Estas categorias e seus padrões são representados na tabela 1.4. Alguns padrões não estão vinculados exclusivamente a uma única categoria de problema. Na visão dos autores, cada padrão pode ser aplicado a diversas categorias, contudo, apresenta apenas uma categoria de problema principal, sendo as demais consideradas categorias secundárias.

CATEGORIA DE PROBLEMAS	PADRÕES ARQUITETURAIS
"From Mud to Structure"	Layers Pipes e Filters Blackboard
Sistema Distribuídos	Broker Pipes e Filters Microkernel
Sistemas Interativos	MVC PAC
Sistemas Adaptáveis	Microkernel Reflection

Tabela 1.4: Categoria de problemas e padrões arquiteturais

2.7 Heurísticas para a Seleção de Estilos e Padrões Arquiteturais

A principal dificuldade relacionada à construção de arquiteturas baseadas em estilos e padrões arquiteturais deve-se a falta de uma base de conhecimentos suficientemente maduros para o seu exercício (FOX, 1996). O conhecimento que o arquiteto utiliza para tomar suas decisões de projeto apóia-se na maioria das vezes na subjetividade. Neste contexto, existem abordagens que se propõem a apoiar o processo de seleção de estilos e padrões arquiteturais para as suas aplicações (CLEMENTS, 1995, SHAW e GARLAN, 1996, KAZMAN e KLEIN, 1999). As principais abordagens que dão sustentação às heurísticas de seleção baseiam-se no complemento do conhecimento do projetista e na avaliação dos estilos e padrões arquiteturais em relação à obtenção de certas propriedades (normalmente, características relacionadas à função da aplicação ou aos seus aspectos gerais, tais como desempenho, reutilização, etc.).

Um exemplo de heurística pode ser encontrado em (SHAW e GARLAN, 1996). Os autores mostram que a seleção de estilos arquiteturais em geral requer dois tipos de informações principais:

1. Uma discriminação cuidadosa entre as arquiteturas candidatas, e
2. Um guia de projeto para a realização de uma escolha apropriada.

A forma com que esta discriminação é realizada baseia-se em elementos classificadores relacionados à natureza dos componentes e conectores de cada estilo, ao compartilhamento, alocação e transferência do controle entre os componentes, ao tráfego dos dados através do sistema, à interação existente entre dados e controle e aos tipos de análise (por ex., teoria de máquina de estados não-determinística, composição funcional, etc.) que cada estilo arquitetural está sujeito.

A estrutura utilizada para a escolha do estilo apropriado ao projeto baseia-se na forma geral: “Se o seu problema possui a característica X, considere arquiteturas com características Y”. Tal estrutura recebeu influência do trabalho de (LANE, 1990) com componentes de interface para sistemas interativos. Baseado nesta forma geral, encontramos um exemplo real da aplicação de um guia de projeto:

“*Se o seu problema envolve a transformação de cadeias de dados contínuos, considere a arquitetura Pipes & Filters*”.

Porém, o que nos chama a atenção nesta abordagem é o fato de ser utilizado um critério para a discriminação baseado em informações muito específicas e que estão distantes da fase de projeto preliminar de uma aplicação. Acreditamos ser mais propício e natural que o projetista arquitetural pense em características mais genéricas que a arquitetura busca atender durante a passagem da fase de análise para o projeto preliminar, do que pensar em processos, chamadas de procedimentos e topologias. Esta abordagem mostra-se frágil, ainda, ao buscar resolver certos conflitos quando estão envolvidas mais de uma característica do problema. Ao criar uma descrição de problema semelhante a anterior, adicionando uma característica que o complemente, tal como:

“*Se o seu problema envolve a transformação de cadeias de dados contínuos e a manutenção da funcionalidade em situações de erros*”.

Considerar o estilo ‘Pipe & Filter” como o mais indicado para atender a ambas as características vai de encontro ao que pode ser encontrado na análise deste mesmo estilo em (BUSCHMANN *et al.*, 1996). Isto mostra que, em situações em que tenhamos que analisar problemas mais complexos, esta abordagem é limitada porque

exigiria uma descrição complexa e longa das diversas possibilidades que um problema particular pode se manifestar.

O esquema de classificação adotado em BUSCHMANN *et al.* (1996) relaciona as categorias e problemas para formar uma classificação bi-dimensional de padrões que engloba as categorias arquiteturais. Os autores realçam, ainda, a possibilidade de aplicação de um padrão em categorias distintas de sua principal classificação. Este esquema de classificação baseia-se na descrição e análise de cada padrão arquitetural e na avaliação dos benefícios e dificuldades encontrados com a sua aplicação. Particularmente, percebemos que este processo é longo e envolve um volume alto de informações que acabam mais confundindo do que apoiando um projetista sem experiência (encontramos uma certa dificuldade em identificar critérios comuns de avaliação dos padrões).

Encontramos, ainda em (CLEMENTS, 1995), outros fatores relacionados a influência e decisões tomadas durante a especificação da arquitetura de projetos de sistemas em larga escala. Ampliando o contexto das influências descritas em (SHAW e GARLAN, 1996), aqui o autor relaciona algumas questões referentes à cultura organizacional e a experiência dos projetistas com determinadas decisões tomadas durante a fase de projeto arquitetural. Compreender que muitas arquiteturas são o reflexo da aplicação destas influências extra-projeto permite que se associe ao problema computacional práticas adquiridas com o tempo, visando reaproveitar as experiências adquiridas ao longo dos projetos executados, principalmente quando certa estrutura arquitetural é bem aplicada. Tais influências, em geral, complementam as atividades de especificação de arquiteturas das aplicações. Contudo, isto por si só não garante a produção de arquiteturas adequadas, visto que a aplicação de algumas soluções representadas pelos estilos, em geral, não são bem conhecidas pelas equipes desenvolvedoras e por não haver uma única organização arquitetural que atenda bem a todas as situações de projeto.

Mais recentemente, encontramos o trabalho de levantamento de estilos arquiteturais baseados em atributos (ABAS) promovido por (KAZMAN e KLEIN, 1999). O objetivo deste enfoque é tornar mais explícitos alguns dos fatores que influenciam o processo de projeto arquitetural. A estrutura utilizada baseia-se em modelos de atributos de qualidade e seu enfoque é específico em relação a como determinada característica não funcional é atendida por um estilo arquitetural particular. Ou seja, em situações em que um estilo arquitetural favorece a disponibilidade de determinada característica como eficiência ou confiabilidade, isto estimularia a criação de um ABAS respectivo, para a descrição da relação entre o estilo e a característica identificada.

Um catálogo de ABAS tende a se tornar uma realidade incorporada às referências de projetistas de software, mas a sua utilização impõe certas dificuldades à especificação arquitetural em determinadas situações, onde exige-se a composição de soluções baseadas em atributos mínimos de qualidade esperados. Como o centro do enfoque é a característica não funcional e de que forma um padrão arquitetural atende a esta característica, percebemos nesta abordagem a contribuição de trazer a atenção do projetista para questões relacionadas ao problema ao invés do processo inverso, semelhante ao descrito em (SHAW e CLEMENTS, 1996). Porém, a idéia proposta não deixa claro como podemos privilegiar um conjunto de características não funcionais quando se pretende especificar uma solução baseada em um determinado estilo.

2.8 Trabalhos Encontrados na Literatura

Nesta seção buscamos descrever os principais trabalhos referentes a arquiteturas de software descritos na literatura, organizando-os pelas áreas de pesquisa que são relacionadas, e que estão associados a este trabalho, sendo elas: o suporte automatizado ao projeto arquitetural, a análise de arquiteturas de software e a reutilização de arquiteturas.

2.8.1 Suporte Automatizado ao Projeto Arquitetural

Com o surgimento da arquitetura de software como disciplina, houve o desenvolvimento de ferramentas e ambientes que pretendem apoiar o desenvolvimento de software com ênfase em suas arquiteturas. Duas abordagens principais nortearam a proliferação destes trabalhos: a primeira abordagem orientou o desenvolvimento de ferramentas e ambientes que suportam um estilo arquitetural particular e seus respectivos métodos de desenvolvimento. São exemplos destas abordagens os suportes a arquiteturas baseadas em fluxo de dados (MAK, 1992) e no estilo *Blackboard* (NII, 1986). Tal abordagem mostrou-se desfavorável devido aos problemas relacionados ao custo de desenvolvimento em relação aos benefícios alcançados (SHAW e GARLAN, 1996).

A segunda abordagem baseia-se na possibilidade de suporte arquitetural a mais de um estilo. Isto significa que as ferramentas e ambientes baseados nestas propostas abrem um leque de possibilidades quanto a escolha de estilos arquiteturais (um número finito suportado pela ferramenta) e permitem a aplicação deste estilo para a definição da arquitetura da aplicação. Esta abordagem mostra-se mais generalista do que a anterior, principalmente no que se refere a seleção de um estilo arquitetural

para a orientação do projeto da aplicação. Os principais exemplos deste tipo de abordagem são o AESOP (MOROE e GARLAN, 1996) e o ARGO (ROBBINS e REDMILES, 1998)

2.8.1.1 AESOP

O AESOP é um ambiente construído para suportar o desenvolvimento rápido de "ambientes de projeto arquitetural de software baseados em um estilo" (MOROE e GARLAN, 1996). O kit de ferramentas do AESOP inclui ainda um repositório de elementos de projeto arquitetural (componentes, conectores e configurações) que é basicamente utilizado para suportar a classificação, armazenamento, e recuperação destes elementos arquiteturais. O ambiente AESOP mostra-se fortemente baseado na reutilização dos componentes gerais de estilos arquiteturais para a geração dos ambientes instanciados. A facilidade de sua expansão também é favorável porque o repositório de elementos de projeto pode ser alimentado sem que o conjunto de ferramentas que o utilizam seja afetado.

No entanto, o aspecto crítico do AESOP é, justamente, a instanciação de ambientes para arquiteturas homogêneas, permitindo apenas o uso de um estilo arquitetural por vez no momento da instanciação. O fator desfavorável desta abordagem é que alguns produtos de software pré-existentes dificilmente poderiam ser tratados por um ambiente instanciado, pelo simples fato de não terem sido desenvolvidos em um único estilo arquitetural.

2.8.1.2 ARGO

O ambiente de projeto arquitetural ARGO representa o esforço de pesquisa na área de suporte a alguns dos desafios cognitivos existentes no projeto de software (ROBBINS e REDMILES, 1998). O propósito principal deste ambiente é apoiar as ações do projetista no momento em que este toma suas decisões em relação as possíveis alternativas de projeto. O ambiente apóia o desenvolvimento de software orientado a objetos e possui um mecanismo de crítica que pretende informar os problemas potenciais envolvidos e sugerir possíveis ações corretivas.

O ambiente ARGO apóia-se em um processo onde seus mecanismos de controle baseiam-se em predicados que coordenam a execução de críticas. As críticas são agrupadas em tipos principais, de acordo com a semântica que cada uma oferece. Desta forma, críticas sobre a corretude, completude e consistência, entre outras, do projeto são realizadas ao longo do desenvolvimento. O modelo de decisão

que coordena as ações de crítica do ARGO pode ser diretamente manipulado pelo projetista, através da avaliação de cada um dos tipos de decisões aplicáveis ao projeto.

Porém, até o momento que esta tese foi escrita o ambiente ARGO apenas apresenta o suporte à aplicação do estilo arquitetural C2 no desenvolvimento de arquiteturas (MEDVIDOVIC *et al.*, 1997). Acreditamos que, por apoiar o desenvolvimento de software orientado a objetos com ênfase no projeto arquitetural, a aplicação de padrões arquiteturais (BUSCHMANN *et al.,* 1996) e de projeto (GAMMA *et al.*, 1995) torna-se uma características quase que obrigatória às ferramentas e ambientes que se intitulam "*favoráveis ao projeto arquitetural"*.

2.8.2 Análise de Arquiteturas de Software

A experiência prática tem mostrado que algumas características de qualidade de um software baseiam-se principalmente na sua arquitetura (CHUNG *et al.*, 1999, BOSCH e MOLIN, 1999). Isto permite dizer que um bom projeto arquitetural irá facilitar em muito o alcance de níveis satisfatórios de aceitação da aplicação produzida. Porém, a qualidade do software não pode ser adicionada ao final do projeto arquitetural; deve ser buscada desde o início, construída durante o projeto. Cientes desta relação, existem trabalhos que buscam a análise das arquiteturas de software para avaliação de suas características ainda na fase de projeto. A principal abordagem é permitir que seja possível, durante o projeto de software, a avaliação de modelos de arquiteturas candidatas antes que estas sejam de fato construídas. Desta forma, a apresentação de técnicas efetivas de avaliação de arquiteturas candidatas mostram-se de grande valor.

Embora existam métodos que avaliam sistemas de software em relação a atributos específicos, tais como desempenho, estamos interessados em métodos que permitam uma avaliação de arquiteturas que se baseiem em diferentes características de qualidade esperadas para o sistema. Neste contexto, encontramos os métodos SAAM (KAZMAN *et al.*, 1994) e ATA (BARBACCI *et al.,* 1998).

2.8.2.1 SAAM

O método de análise arquitetural SAAM (Software Architecture Analysis Method) é um método baseado em cenários para análise de arquiteturas. Seu principal objetivo é fornecer meios de caracterizar o quanto um projeto arquitetural corresponde às demandas estabelecidas por um conjunto de cenários, onde cada cenário

representa uma seqüência especificada de passos (KAZMAN *et al.*, 1994). Os cenários são utilizados para ilustrar os tipos de atividades que a aplicação deve suportar e os tipos de modificações antecipadamente vislumbrados. O método se preocupa em capturar os casos de uso mais importantes, os usuários do sistema e as qualidades que o sistema deve satisfazer. Em geral, os cenários são organizados em conjuntos que apresentam uma semântica comum, como por exemplo cenários usados para avaliar a capacidade do software em tolerar falhas.

O processo de avaliação do método SAAM mede, entre outras coisas, quantos cenários de avaliação causam potenciais mudanças a um único componente arquitetural. Desta forma, o critério de avaliação baseia-se nesta premissa: se um grupo de cenários são similares em seus objetivos e todos afetam o mesmo componente, ou poucos componentes relacionados da arquitetura, então isto significa que a funcionalidade do sistema foi modularizada de uma forma que reflete apropriadamente o esforço de se alcançar as características representadas pelos cenários. De forma análoga, se um grupo de cenários similares afeta diferentes componentes por toda a arquitetura, então esta arquitetura apresenta problemas.

Um dos aspectos positivos deste método é o suporte dado à avaliação de alternativas arquiteturais para o conjunto de cenários. Neste caso, o método adota a atribuição de pesos aos cenários para representar suas importâncias no contexto do sistema e com isso pode-se criar uma escala de arquiteturas que suportam bem os cenários em análise. A atribuição de pesos aos cenários busca resolver as situações onde uma determinada arquitetura aplica-se bem a uma metade de cenários enquanto que uma segunda arquitetura relaciona-se melhor com a outra metade. A atribuição de pesos é um processo subjetivo que envolve todos os papéis presentes no desenvolvimento do software.

2.8.2.2 ATA

O método ATA (*Architecture Tradeoff Analysis*) baseia-se em um conjunto de medições de características específicas do sistema, algumas analíticas, baseadas em modelos formais (por exemplo, desempenho e disponibilidade), e outras qualitativas, baseadas em inspeções formais (manutenibilidade e confiabilidade) (BARBACCI *et al.,* 1998). Associado ao método, existe o processo que prescreve uma série de passos e que utiliza um conjunto de estilos arquiteturais e modelos analíticos definidos pela proposta ABAS (KAZMAN e KLEIN, 1999). Os benefícios almejados são a descoberta e o entendimento das principais características de qualidade, a melhoria da documentação arquitetural com a respectiva melhoria da comunicação entre os papéis

com interesse na arquitetura e apoio à identificação de riscos e tomada de decisões ao nível de projeto preliminar.

O enfoque deste método de avaliação apóia-se na extração de conjuntos de características de qualidade em múltiplas dimensões, analisando os efeitos de cada requisito isoladamente, e então, compreendendo as interações entre estes requisitos. Ambos os processos envolvem aspectos técnicos e sociais. Os aspectos técnicos pretendem mostrar como as informações arquiteturais são coletadas, representadas e analisadas, enquanto os aspectos sociais lidam com interações entre os papéis diretamente interessados na arquitetura do software, permitindo que se estabeleça um processo de comunicação que chegue a avaliações das análises realizadas.

A principal vantagem deste método é a representação dos aspectos sociais como fonte de influências sobre a avaliação das características de qualidade da arquitetura e da possibilidade de aplicação do método antes mesmo de se ter qualquer representação de uma arquitetura, permitindo que algumas análises realizadas sobre o problema (como por exemplo a identificação de riscos potenciais) possam guiar o desenvolvimento do projeto arquitetural da aplicação.

2.8.3 Reutilização de Arquiteturas

Construir software sem reutilizar o que já foi feito mostra-se economicamente inviável para se atingir níveis de qualidade, prazos e custos competitivos. Com a complexidade do software, a reutilização de arquiteturas de sistemas apresenta-se como uma opção de desenvolvimento de aplicações que possuam o mesmo propósito geral. Isto se dá porque as organizações têm considerado suas arquiteturas como recursos que devem ser mantidos e reutilizados tanto quanto for possível. Para elas, cada arquitetura representa um investimento significativo de tempo e esforço especializado para a sua obtenção, justificando com isso o interesse de se maximizar este investimento através da reutilização desta arquitetura em sistemas similares. Uma das principais premissas identificadas por essas organizações é a de que as arquiteturas influenciam as organizações tanto quanto as organizações influenciam as arquiteturas (BASS *et al.*, 1998). No contexto de reutilização arquitetural, apresentamos a iniciativa de reutilização de arquiteturas de software específicas de domínio, também conhecido como DSSA (*Domain-Specific Software Architecture*) (GACEK, 1995).

2.8.3.1 DSSA

Um DSSA representa mais do que uma arquitetura para um determinado domínio de aplicação. Um DSSA é uma coleção de componentes de software, especializados para um determinado tipo de tarefa (domínio), generalizados para que seja possível seu uso efetivo através do domínio, e compostos em uma estrutura padronizada (topologia) efetiva para a construção de aplicações (HAYES, 1994). Além do aspecto arquitetural que este representa, é importante salientar a infra-estrutura que suporta a sua geração e o processo de instanciação e refinamento do mesmo, tendo como objetivo a criação de aplicações no domínio referenciado pelo DSSA. Seu enfoque é o desenvolvimento de software para reutilização (PRIETO-DÍAZ e FREEMAN, 1987), com ênfase na reutilização de informações de domínio e de projeto arquitetural.

A arquitetura de referência de um domínio de aplicação pretende tornar mais efetivo o compartilhamento e a reutilização de idéias, métodos, componentes e produtos dentro de uma comunidade de desenvolvedores de aplicações similares. Por conseguinte, atinge-se níveis mais aceitáveis de qualidade para os produtos desenvolvidos e reduz-se os riscos inerentes ao processo de criação de um software e redução do tempo associado ao ciclo de vida da aplicação.

Por apresentar uma característica de generalização que busca atender uma família de aplicações, o custo de criação de um DSSA pode chegar até a três vezes mais do que uma arquitetura de aplicação específica (WENTZEL, 1994). Porém, a expectativa na redução dos gastos em desenvolvimento pode ser observada através da diminuição dos custos de desenvolvimento de uma aplicação (em aproximadamente um quarto), e dos custos em manutenção (cerca de um terço), baseados em relatos de experiência de programas de reutilização de software descritos na literatura (JONES, 1986, TRACZ, 1987, WENTZEL, 1994).

Encontramos em (TRACZ e HAYES, 1994, HAYES, 1994) uma descrição de processos de criação e instanciação de um DSSA, representando o resultado do esforço de pesquisa do projeto ARPA nesta área (METTALA e GRAHAM, 1992).

O processo de criação de um DSSA apóia-se na aquisição de três elementos de informação principais, que são o modelo de domínio, os requisitos e a arquitetura de referência.

O *modelo de domínio* busca fornecer aos indivíduos uma compreensão sem ambigüidades dos vários aspectos de um domínio em particular. Um domínio é definido por um conjunto comum de problemas ou funções que aplicações neste domínio devem resolver. Além disso, um domínio é caracterizado por um jargão

comum (ou ontologia) que serve para descrever os problemas que as aplicações buscam solucionar. A principal diferença entre a análise dos requisitos de um DSSA e a análise tradicional dos requisitos de um software está na ênfase que é dada na separação dos requisitos funcionais dos requisitos de projeto e implementação, muito maior no primeiro caso. O modelo de domínio descreve o contexto em que aplicações serão construídas e caracteriza os principais objetos, seus atributos e relacionamentos. Inclui os requisitos de referência que caracterizam todas as aplicações deste domínio.

Os *requisitos de referência* são usados pelos arquitetos de domínio para a criação da arquitetura de referência. Neste sentido, busca-se identificar a porção do espaço da solução para onde o modelo de domínio será mapeado. Os requisitos de referência são divididos em requisitos funcionais, que são as características de definição do espaço do problema, e requisitos não-funcionais, de projeto e de implementação, que representam as características restritivas do espaço-solução (PRIETO-DÍAZ, 1987).

Pelo menos uma *arquitetura de referência* que satisfaz as capacidades funcionais e não-funcionais identificadas nos requisitos de referência. Esta idéia não implica que todos os requisitos funcionais e não funcionais sejam satisfeitos por uma única arquitetura, e sim permitir que exista mais de uma arquitetura de referência para um domínio de aplicação. Esta flexibilidade de opções de arquiteturas de referência permite que o engenheiro de domínio caracterize múltiplas soluções a partir de arquiteturas de referência distintas.

A arquitetura de referência é mais genérica do que um projeto arquitetural de uma aplicação única porque é construída para ser reutilizada, estendida e configurada. Além disto, associa-se à arquitetura uma documentação mais detalhada, de maneira que se forneça aos engenheiros de aplicações e aos programadores uma quantidade de informação suficiente à geração de novas aplicações e à modificação das já existentes que são baseadas na arquitetura de referência.

A iniciativa de construção de arquiteturas de referência enfoca ainda a importância da escolha do estilo ou padrão arquitetural que irá guiar a especificação arquitetural, porém não discute a informa sobre como fazê-lo. Percebemos que a relação existente entre o modelo proposto por um padrão arquitetural e o suporte aos requisitos de referência, especialmente dos de natureza não-funcional, é difícil de ser identificada e realizada pela arquitetura de referência. Alguns trabalhos (TRACZ, 1994, MEEKEL *et al.*, 1997, GOMAA e FARRUKH, 1999) ilustram a situação onde determinada arquitetura de referência é apresentada sem que tenha havido uma única

discussão sobre os critérios utilizados para justificar a utilização do padrão adotado. Vemos nesta situação uma oportunidade de contribuição para o problema de escolha de padrões arquiteturais no contexto da especificação da arquitetura de referência.

O processo de instanciação e refinamento de um DSSA, descrito em (TRACZ e HAYES, 1994), baseia-se na utilização dos elementos identificados no processo de sua criação e em uma sistemática de tratamento destas informações, almejando com isso a construção de uma aplicação.

Os passos principais deste processo são a análise dos requisitos da aplicação e o projeto e desenvolvimento da aplicação.

A *análise dos requisitos da aplicação* está relacionada à caracterização do sistema desejado. Os requisitos de um software específico podem tanto especializar os requisitos de referência do DSSA quanto adicionar novos objetivos, restrições e critérios de avaliação. O que o autor não explora é a possibilidade de serem definidos requisitos específicos da aplicação que venham de encontro aos identificados na especificação dos requisitos de referência e suportados pela arquitetura de referência. Entendemos que na maioria das situações este conflito não deva ocorrer porque uma família de aplicações em geral compartilha requisitos comuns de função e de qualidade. Porém, esta possibilidade deveria ao menos ser citada e de alguma forma minimamente tratada.

O *projeto e o desenvolvimento da aplicação* ocorre quando se deseja projetar e implementar a aplicação. Este processo especifica a arquitetura de referência de forma a satisfazer os requisitos da aplicação. Esta especialização, configuração, ou particularização, produz uma arquitetura mais detalhada do sistema. E, por fim, os componentes da arquitetura, inicialmente caracterizados abstratamente, são instanciados com uma implementação particular, e as características da plataforma destino para a execução do sistema são explicitadas. Em todos os casos, o sistema deve satisfazer de maneira consistente ao contexto assumido pelo modelo de domínio.

2.9 Conclusão

O recente interesse na área de estudo das arquiteturas de software foi impulsionado pelo caráter que estas representam no desenvolvimento de produtos de qualidade (BOSCH e MOLIN, 1999, CHUNG *et al.*, 1999), pelo seu valor econômico agregado (BASS *et al.*, 1998), e pela dificuldade de se desenvolver produtos de software complexos (PARNAS, 1972).

As inúmeras arquiteturas de software existentes alimentaram o estudo que permitiu que fossem identificados os principais estilos e padrões arquiteturais, formando uma base de conhecimento que vem possibilitando a representação, comunicação, construção de novas arquiteturas e a reutilização de arquiteturas existentes em novos projetos. As arquiteturas de software que seguem determinado estilo, ou padrão, apresentam características semelhantes, principalmente no que se refere aos benefícios alcançados e responsabilidades assumidas (como por exemplo, a possibilidade de reutilização de componentes, a facilidade de manutenção, os problemas com desempenho, etc.).

A descrição dos principais trabalhos relacionados ao tema podem dar uma dimensão das áreas de pesquisa e de suas principais frentes de exploração. Através desta descrição podemos realizar algumas observações que consideramos importantes:

1. Um dos aspectos críticos de se desenvolver software com ênfase arquitetural é a seleção de um estilo, ou padrão. As abordagens apresentadas buscam a seu modo ampliar o conhecimento do projetista. Neste sentido, identificamos a oportunidade de apoiar a seleção de padrões arquiteturais utilizando uma abordagem orientada pela necessidade de se obter determinadas características de qualidade para o software a ser produzido;
2. Existe uma carência de suporte à aplicação dos padrões arquiteturais em ambientes que se dizem orientados ao desenvolvimento arquitetural (ou que reconhecem a importância da arquitetura de software);
3. A reutilização de arquiteturas de software objetiva a redução dos investimentos feitos no desenvolvimento de aplicações construídas em um mesmo domínio computacional. As arquiteturas de referência agregam uma série de requisitos comuns às aplicações do domínio computacional e sua importância para a construção de aplicações com qualidade é uma de suas principais propriedades. Porém, na maioria dos casos estudados, percebemos uma representação da arquitetura de referência com pouca ênfase na descrição do porquê determinado estilo ou padrão arquitetural foi escolhido e de que maneira os requisitos de referência estariam sendo privilegiados com esta arquitetura particular.

Capítulo 3: Arquiteturas de Software Específicas de Domínios no Contexto de uma Infra-estrutura de Reutilização

3.1 Introdução

Conforme dito no capítulo 1, na nova economia digital, o software exerce influência direta sobre as formas como os negócios são realizados. A demanda pela disponibilização de novos serviços exige que sistemas que os apóiem ou os realizem sejam produzidos e implantados no menor tempo possível, sendo muitas vezes um critério crucial para a manutenção e prosperidade das organizações em mercados altamente competitivos. A mesma situação pode ser constatada em vários outros domínios. Desta forma, uma maneira de se conseguir diminuir os custos e o tempo de desenvolvimento de aplicações seria a utilização de componentes de software pré-fabricados (SZYPERSKI, 1998).

Segundo (BOOCH, 1996), o desenvolvimento de aplicações baseado em componentes torna-se mais efetivo quando está centrado na arquitetura do software. A arquitetura de software de uma aplicação é um artefato que exerce grande impacto sobre a estruturação do esforço de desenvolvimento que a produz (a arquitetura prescreve os componentes de software que devem ser integrados para formar o sistema, e estes componentes são por sua vez o resultado de uma divisão de atribuições nas equipes desenvolvedoras, que por natureza formam a base para a estrutura de desenvolvimento de um projeto) (BASS *et al.*, 1998). Por apresentar tais características, a reutilização de arquiteturas em determinados contextos tem recebido especial atenção, particularmente na situação em que uma arquitetura orienta a produção de um conjunto de sistemas que compartilham um mesmo domínio de aplicação.

A abordagem dos DSSA's representa a concretização de uma proposta de reutilização de software com ênfase arquitetural. O principal objetivo desta abordagem reside na estruturação de componentes para que sirva de referência ao desenvolvimento de aplicações. Ao buscar a produção de aplicações em determinado domínio, um DSSA torna-se mais um elemento de ligação entre a Engenharia de Domínio (ED) (GRISS, 1999) e a Engenharia de Aplicações (EA) (MANNION *et al.*, 1999). Segundo o *Army Reuse Center* (SIMOS e ANTHONY, 1998), a ED é o processo de busca, análise, manipulação e formalização das informações relevantes do domínio para a implementação de um programa de reutilização, que promova o desenvolvimento de aplicações do domínio a um custo reduzido. A ED tem como

principal objetivo a utilização de uma abordagem baseada em modelos de domínio para facilitar a instituição de reutilização e sua utilização em cadeia no ciclo de desenvolvimento de software. Em contrapartida, a EA se dedica ao estudo das melhores técnicas, processos e métodos para a produção de aplicações, no contexto do desenvolvimento baseado em reutilização (GRISS *et al.,* 1998). Assim sendo, a EA atua de forma paralela à ED, utilizando os componentes produzidos reutilizáveis que representam o domínio (conhecido como desenvolvimento *para* reuso) para a construção de um produto de software (desenvolvimento *com* reuso).

A aplicabilidade de um DSSA pode ser percebida quando há a intenção de se enfocar mais explicitamente uma organização arquitetural que sirva como referência a um determinado domínio de aplicações e quando há a necessidade do desenvolvimento de aplicações neste domínio, a partir da reutilização desta arquitetura.

Baseado nestes enfoques diferentes de reutilização de software e a partir da constatação da extensão do DSSA nestas duas visões da reutilização, cabe realçar que para a ED a arquitetura de referência exerce o papel da organização lógica que representa uma possível solução para as questões especificadas durante a fase de entendimento do domínio, e que para a EA esta mesma arquitetura serve de alimentador para o projeto da aplicação, na medida que possibilita uma melhor compreensão da solução de referência, além do seu uso e subseqüente refinamento (figura 3.1).

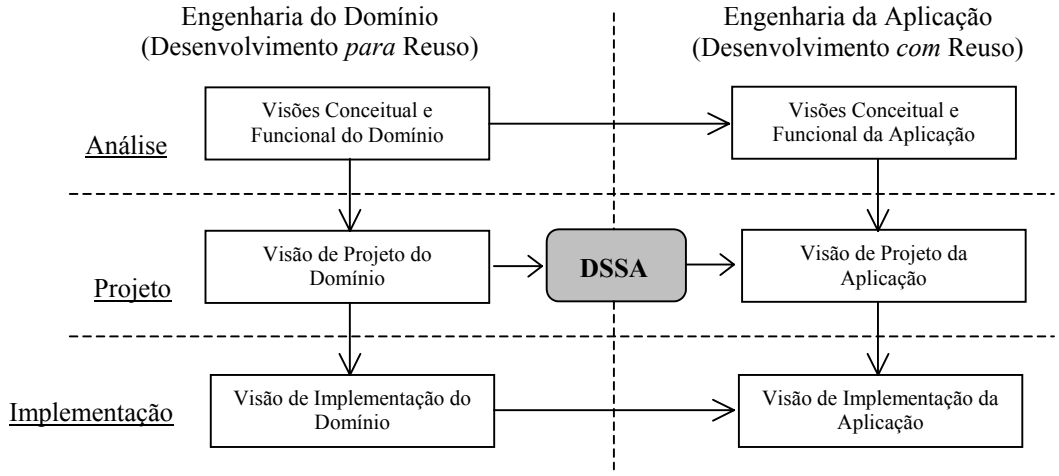


Figura 3.1: O contexto das DSSA's na ED e EA

Dois tipos de profissionais são especialmente interessados na arquitetura de referência de um domínio, de acordo com o tipo de engenharia adotado: o projetista do

domínio, cuja responsabilidade principal é construir a arquitetura em si e o projetista da aplicação, cuja função é adaptar a arquitetura para a realidade de uma aplicação.

Conforme dito anteriormente, este trabalho considerou o contexto do projeto Odyssey, atualmente em desenvolvimento na COPPE/UFRJ, que tem como objetivo constituir uma infra-estrutura de suporte ao desenvolvimento baseado em modelos de domínio e que permite a construção de aplicações em domínios particulares (BRAGA e WERNER, 1999). Neste capítulo propomos a adaptação dos processos de ED e EA suportados atualmente pelo Odyssey para que as atividades relacionadas à geração e instanciação de arquiteturas de software específicas de domínio sejam detalhadas suficientemente para sua efetiva realização.

O capítulo está dividido em 5 seções (incluindo esta introdução). Na seção 3.2 descrevemos os processos de Engenharia de Domínio e de Aplicações adotados pelo Odyssey, caracterizando suas principais etapas, e apresentamos algumas considerações gerais sobre os processos de criação e instanciação de arquiteturas de software específicas de domínio. Na seção 3.3, buscamos estender o processo Odyssey-ED para o suporte à geração de DSSAs e propomos uma abordagem de seleção e aplicação de padrões arquiteturais para a criação de arquiteturas de referência. Na seção 3.4, detalhamos o processo Odyssey-EA no que diz respeito ao suporte à instanciação de DSSAs em aplicações particulares do domínio. E, finalmente na seção 3.5, realizamos uma conclusão deste capítulo em relação aos objetivos almejados.

3.2 A Infra-estrutura Odyssey e seus Processos de Engenharia de Domínio e de Aplicação

A infra-estrutura Odyssey pode ser vista como um arcabouço onde modelos conceituais, arquiteturas de software e modelos implementacionais são especificados para domínios de aplicação previamente selecionados.

Sua infra-estrutura de suporte ao processo de engenharia de domínio busca criar mecanismos que permitam a geração de aplicações em um domínio do conhecimento através do maior entendimento do mesmo, tendo como base a tecnologia de desenvolvimento baseado em componentes (BRAGA e WERNER, 1999). Para que o desenvolvimento de software baseado em componentes seja efetivo, o Odyssey utiliza um processo de engenharia de domínio (ED), que almeja a criação de componentes de qualidade para domínios de aplicações, e um processo de engenharia de aplicação (EA), centrado na produção de aplicações a partir dos componentes reutilizáveis de um domínio particular.

Os tipos de usuários da infra-estrutura Odyssey podem ser divididos conforme o enfoque dado sobre os processos de ED e EA:

- Engenheiros de domínio e especialistas do domínio utilizam o ambiente principalmente para especificar e evoluir os conceitos de domínio (papéis centrados na ED);
- Engenheiros de software utilizam a infra-estrutura para obter conhecimento sobre o domínio da aplicação e reutilizar este conhecimento na especificação de sua aplicação (papel centrado na EA).

3.2.1 O Processo de Engenharia de Domínio no Odyssey

O processo de engenharia de domínio do Odyssey é chamado *Odyssey-ED* e tem como principal objetivo a criação de componentes reutilizáveis em um dado domínio, para que estes componentes venham a atender as necessidades dos desenvolvedores de aplicações naquele domínio, em todos os níveis de abstração requeridos. Um domínio é definido por um conjunto comum de problemas ou funções que aplicações neste domínio podem resolver. Além disso, um domínio é caracterizado por um jargão comum (ou ontologia) que serve para descrever os problemas que as aplicações buscam solucionar (GUARINO, 1998). A figura central desta fase é o especialista do domínio e seu objetivo é levantar as características comuns à família de sistemas do domínio em questão através da formalização dos seus requisitos mais gerais.

Para que a criação de componentes de qualidade se torne efetiva, o Odyssey-ED utiliza uma forma incremental e evolutiva de disponibilização de componentes, ou seja, na medida que novas informações vão sendo agregadas ao domínio, os componentes reutilizáveis vão sendo evoluídos com base nas novas informações do domínio adquiridas (figura 3.2). Desta forma, o Odyssey-ED adota um modelo de ciclo de vida em espiral (PRESSMAN, 1995), uma vez que os componentes disponibilizados em todas as etapas do processo são melhorados continuamente. O desenvolvimento dos componentes reutilizáveis passa pela fase de análise até a fase de testes, completando um ciclo inteiro do desenvolvimento de software.

O processo Odyssey-ED está baseado nas seguintes etapas: Estudo de Viabilidade, Planejamento e Análise de Risco, Análise do Domínio, Projeto do Domínio e Implementação do domínio (BRAGA *et al.*, 1999).

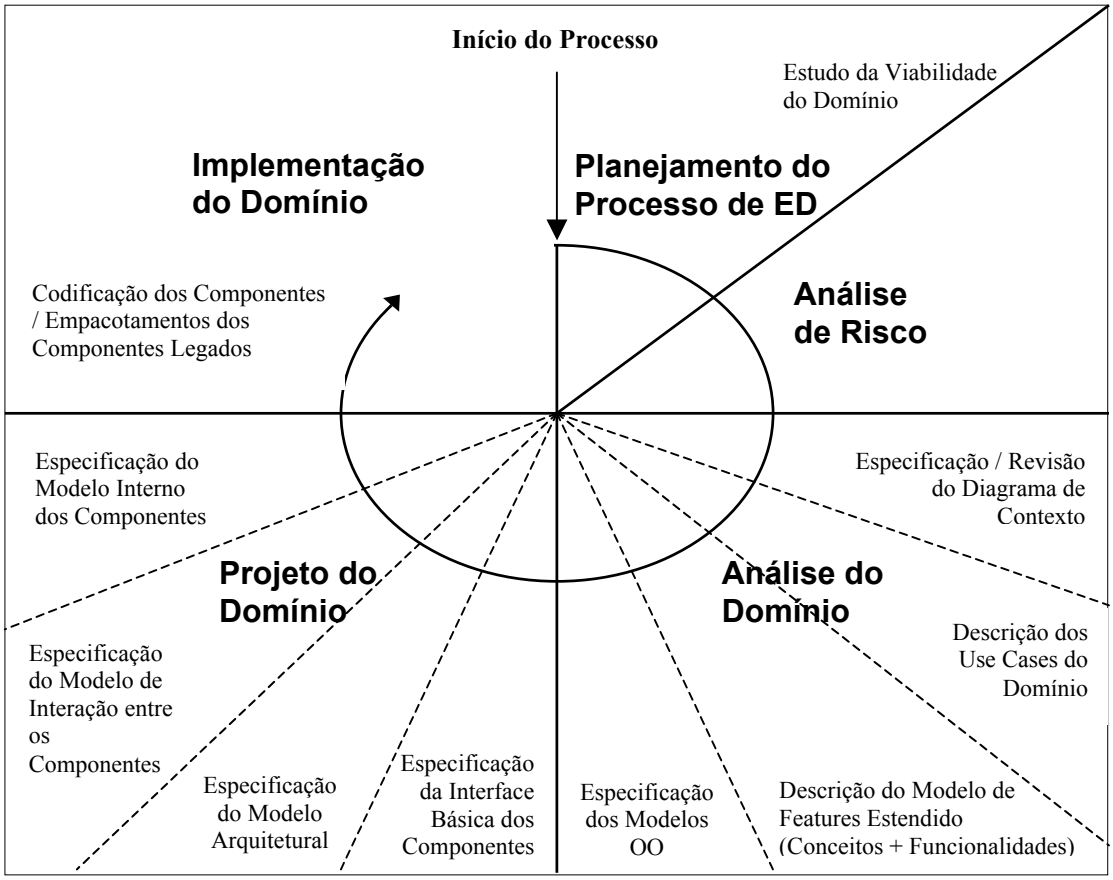


Figura 3.2: Ciclo de vida do processo do Odyssey-ED

3.2.1.1 A Etapa de Estudo de Viabilidade do Domínio

A etapa de estudo de viabilidade do domínio tem como principal objetivo a identificação de um domínio que tem grande chance de ser analisado com sucesso, dado o tempo e recursos disponíveis. Esta etapa de estudo de viabilidade do domínio é uma das mais importantes, pois ela ajudará a decidir se vale ou não a pena a realização da Engenharia de Domínio (ED). Sem esta etapa, as conseqüências da realização da ED podem ser bastante desastrosas em termos de custos e resultados com pouco ou nenhum benefício.

3.2.1.2 A Etapa de Planejamento e Análise de Risco

A etapa de planejamento e análise de risco busca identificar os riscos pertinentes aos projetos de um domínio de aplicação, organizando as informações sobre estes riscos. No contexto do Odyssey, risco é a possibilidade da ocorrência de um evento que cause transtornos ao processo de desenvolvimento de software (BARROS *et al*., 1999).

3.2.1.3 A Etapa de Análise do Domínio

A etapa de análise do domínio consiste na definição dos principais conceitos relacionados ao domínio, ressaltando suas similaridades e diferenças em um nível de abstração que seja de fácil entendimento para os seus diversos usuários. É nesta etapa que são descritos também os limites do domínio e os seus relacionamentos com os demais domínios relevantes ao contexto da análise. Para facilitar a seleção de atividades e conceitos do domínio relevantes a uma dada aplicação, os produtos desta fase do Odyssey-ED são particionados de acordo com as principais características (funcionalidades) do domínio. As principais fases realizadas na análise de domínios no Odyssey-ED são:

- *A especificação / revisão do diagrama de contexto do domínio*: Esta fase tem como principal função a contextualização do domínio em relação ao seu escopo, limites, relacionamentos com outros domínios e o envolvimento dos seus principais atores;
- *A descrição dos casos de uso do domínio e modelos OO relacionados*: Por adotar como paradigma de desenvolvimento a orientação a objetos (BOOCH, 1994) e por seguir abordagens consagradas para a elicitação do conhecimento, a descrição dos modelos de casos de uso e de classes é realizada no Odyssey-ED para a representação dos comportamentos e conceitos relacionados ao domínio, respectivamente. Os casos de uso são utilizados principalmente para capturar as principais funcionalidades do domínio, apoiando ainda a construção de outros modelos OO, principalmente o diagrama de classes, para que haja a avaliação e formalização das principais entidades que sustentarão as funções de domínio. Os casos de uso podem ainda ajudar na identificação de componentes reutilizáveis que são descritos em maior detalhe nos outros modelos OO (diagrama de colaboração, diagrama de atividades, etc);
- *A descrição do modelo de features estendido*: Objetiva a apresentação, em um nível abstrato, do relacionamento entre as funcionalidades e os conceitos do domínio. É a fase que se busca explicar quais os significados dos conceitos do domínio e seus respectivos relacionamentos, para que seja facilitado o entendimento do domínio como um todo. As características gerais do domínio nesta fase da ED são capturadas pelo modelo de *features*[8], para que tanto as similaridades quanto as diferenças identificadas durante sua análise sejam conhecidas. As *features* propostas para o uso no Odyssey nesta fase são as funcionais e conceituais (MILLER, 2000).

[8] *Features* são características essenciais do domínio e representam uma forma de captura da informação semântica do mesmo (SIMOS e ANTHONY, 1998).

3.2.1.4 A Etapa de Projeto do Domínio

Na etapa de projeto do domínio, os conceitos e características considerados importantes no domínio são projetados, adicionando-se considerações arquiteturais e de projeto aos componentes conceituais identificados na etapa anterior. As principais fases realizadas na etapa de projeto de domínio são:

- *A definição mais precisa dos componentes e suas interfaces*: Nesta fase há a definição dos serviços que os casos de uso irão disponibilizar para fora de seus limites, ou seja, quais as possíveis mensagens que os *use cases* podem trocar para que as funções de domínio sejam realizadas.

A definição de um modelo geral de colaboração entre componentes: Onde são levados em consideração os componentes identificados na etapa de análise e a interação entre estes componentes para a formação de uma arquitetura de referência do domínio. Este modelo de colaboração surge como uma descrição arquitetural para uma família de aplicações do domínio e descreve componentes funcionais, conexões, protocolos e controle. Além disso, geralmente é constituído de um sistema parcialmente especificado, composto de componentes abstratos ou genéricos, que são substituídos por componentes reais quando a arquitetura é instanciada para uma aplicação. Nesta etapa, é analisada a possibilidade de utilização de componentes legados que disponibilizam alguma funcionalidade requerida pelo domínio.

- *A definição do projeto interno dos componentes*: Objetiva a definição em maior detalhe da estrutura interna de cada componente.

3.2.1.5 A Etapa de Implementação do Domínio

Finalmente, na etapa de implementação do domínio, os componentes especificados na fase de projeto são codificados, definindo-se precisamente suas interfaces para comunicação com outros componentes.

3.2.2 O Processo de Engenharia de Aplicações no Odyssey

O processo de EA do Odyssey é chamado *Odyssey-EA* e tem como um de seus principais objetivos a completa integração com o Odyssey-ED. O Odyssey-EA serve como um processo de referência na construção de aplicações no projeto Odyssey, porém outros processos podem ser propostos para satisfazer necessidades de projeto específicas (MURTA, 2001).

No contexto de um processo de EA, a aplicação da reutilização nas fases iniciais do desenvolvimento deve ter como resultado a reutilização de componentes na

implementação da aplicação, ou seja, deve haver uma relação estreita entre os conceitos de um dado domínio nas fases iniciais do processo de desenvolvimento e os componentes codificados utilizados na implementação da aplicação.

O processo de EA do Odyssey também apresenta etapas dispostas em um ciclo de vida em espiral (figura 3.3), organizado da seguinte forma:

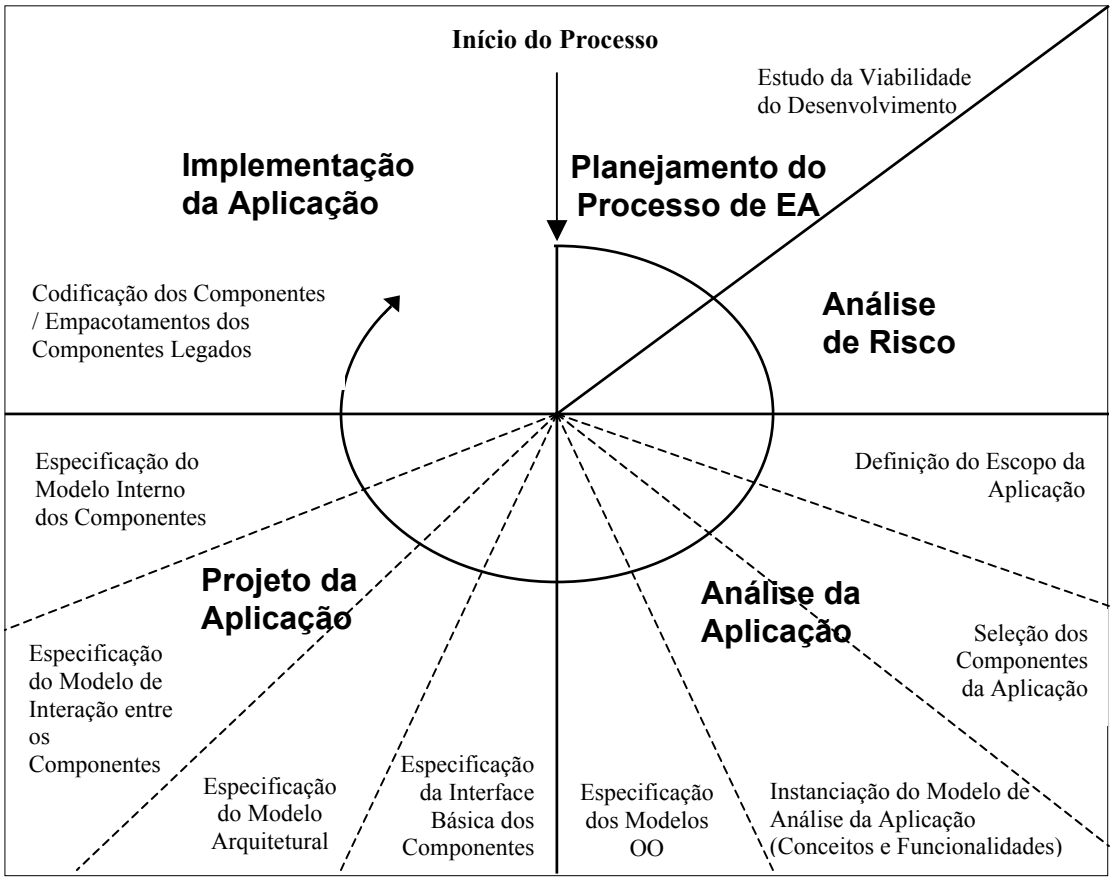


Figura 3.3: Ciclo de vida do processo do Odyssey-EA

3.2.1.1 A Etapa de Planejamento e Análise de Risco

A etapa de planejamento e análise de risco busca identificar e avaliar os riscos envolvidos durante o desenvolvimento de aplicações e planejar medidas que permitam a captura, contenção e contingência dos mesmos. Esta etapa pretende apoiar o Engenheiro de Software na avaliação da possibilidade de desenvolvimento da aplicação através da utilização dos componentes disponíveis do domínio.

3.2.1.2 A Etapa de Análise da Aplicação

A etapa de análise da aplicação tem como objetivo a identificação dos principais conceitos e funcionalidades de uma aplicação. Nesta etapa, o Engenheiro

de Software leva em consideração apenas os aspectos conceituais e funcionais da aplicação, em um nível de abstração que permita a definição do seu escopo em relação ao domínio em que está inserido.

As principais fases realizadas na análise de aplicações no Odyssey-EA são:

- A *definição do escopo da aplicação*: É durante esta fase que o Engenheiro de Software avalia os subdomínios, as entidades e interfaces externas mais relevantes ao escopo da aplicação. Com isso, é possível realizar um primeiro corte sobre o domínio da aplicação, e esta definição de escopo reflete sobre a atividade de seleção dos componentes da aplicação. Em paralelo a esta atividade ocorre a *definição das interfaces com outras aplicações*, e, em conjunto, definem o macro-modelo da aplicação, contendo seu escopo e as suas interfaces com outras aplicações. As definições de escopo e das interfaces com outras aplicações podem representar uma diminuição no número de componentes exibidos para seleção.
- A *seleção dos componentes da aplicação:* Onde ocorre a definição das funções especificadas no modelo de *features* do domínio que serão utilizadas no desenvolvimento da aplicação. Esta seleção de componentes é realizada sobre o subconjunto do domínio estabelecido na atividade de definição de escopo da aplicação. Cada componente funcional disponível para seleção é descrito a partir da definição de casos de uso do domínio. É através destes casos de uso que se compreende as seqüências de funcionalidades e atores neles envolvidos, bem como os componentes estruturais que compõem cada uma das funcionalidades, e é ainda através desta informação que é feita posteriormente a ligação entre as visões conceituais e funcionais. O Odyssey-EA possui uma Política de Seleção de Componentes que sugere algumas opções de escolha dadas pelos componentes selecionados. Isto significa que a escolha de determinada *feature* pode significar a inclusão, ou a exclusão, de outras que possuam um eventual relacionamento ou restrição com a escolhida (MILLER, 2000).
- A *instanciação do modelo de análise da aplicação*: É realizada tendo como objetivo a verificação das inconsistências no conjunto de componentes selecionados para a aplicação (problema causado, principalmente, por escolhas feitas pelo desenvolvedor que violaram restrições ou relacionamentos mandatórios ou por componentes não definidos) e a instanciação das funcionalidades e conceitos do domínio, vindos dos componentes selecionados, para que possam vir a ser adaptados para a aplicação. É importante ressaltar que esta instanciação é feita a partir de uma cópia das funções e conceitos elicitados no domínio para a aplicação e que a ligação entre os componentes disponibilizados na aplicação e os que lhe deram origem é

mantida para que seja possível a rastreabilidade entre os elementos da aplicação e as referências no domínio.

- A e*specificação dos modelos OO:* Fase onde o desenvolvedor busca a adequação dos conceitos e funcionalidades instanciados do domínio para a realidade da aplicação, e que novos componentes, específicos da aplicação, são definidos, caso necessário. A definição dos novos componentes deve ser feita a partir do nível mais alto de abstração (features e casos de uso), utilizando o padrão de definição de componentes do Odyssey (BRAGA, 2000), que utiliza o modelo de features estendido (MILLER, 2000) e os conceitos da notação UML (BOOCH *et al.,* 1998). Os novos componentes são então marcados como incompletos e devem ser detalhados no decorrer do processo de desenvolvimento. Ao final desta fase, a etapa de análise da aplicação terá terminado e o desenvolvedor terá definido os elementos necessários para a definição completa do modelo de análise da aplicação. Este modelo servirá de base para a etapa de projeto da aplicação.

3.2.1.3 A Etapa de Projeto da Aplicação

Na etapa de projeto da aplicação, o enfoque está no refinamento dos conceitos abstratos provenientes da fase de análise, agregando a eles considerações específicas de solução. É nesta fase que se define a arquitetura da aplicação, bem como a adaptação dos componentes da aplicação à arquitetura instanciada.

3.2.1.4 A Etapa de Implementação

E, finalmente na *etapa de implementação*, o desenvolvedor estabelece características totalmente dependentes de máquina tal como sistema operacional, interface com o usuário, armazenamento de dados, etc. O intuito é estabelecer o código executável da aplicação.

3.2.2 Considerações Gerais sobre DSSA's no Odyssey

Uma das dificuldades encontradas durante a reutilização de software baseada em modelos é a transição dos modelos conceituais para os modelos arquiteturais (SIMOS, 1996, CHAN e LAMMERS, 1998, BOSCH, 2000). O mapeamento dos modelos de domínio em uma representação arquitetural é feita, em geral, através da associação dos grupos funcionais, identificados durante a análise do domínio, a uma hierarquia de subsistemas (na maioria dos casos organizados segundo o padrão "Camadas"). Acreditamos que este processo se dá desta forma porque são poucas as

abordagens que detalham as atividades necessárias para apoiar esta transição, e mesmo as que a realizam, carecem de mecanismos que ofereçam opções arquiteturais para a realização deste mapeamento.

Nosso trabalho pretende detalhar as atividades que consideramos necessárias para apoiar esta transição entre modelos e baseia-se nos trabalhos de (BRAGA e WERNER, 1999, MILLER, 2000) no que diz respeito ao contexto da infra-estrutura de reutilização do Odyssey. Estendemos ambos os trabalhos ao definirmos etapas e atividades que estão inseridas nos processos Odyssey-ED e Odyssey-EA e que visam apoiar os processos de criação e instanciação de arquiteturas de referência no contexto desta infra-estrutura. Além disso, propomos a utilização de uma abordagem que relaciona os requisitos de referência com as possibilidades de solução arquiteturais conhecidas e incorporadas ao processo de ED na forma de uma base de padrões para dar apoio à criação do DSSA, e buscamos estabelecer uma diretriz que permita estender o modelo arquitetural do domínio com as características particulares de uma aplicação, no que diz respeito à instanciação do DSSA. A figura 3.4 contextualiza os pontos principais deste trabalho em relação às etapas da ED e EA do Odyssey.

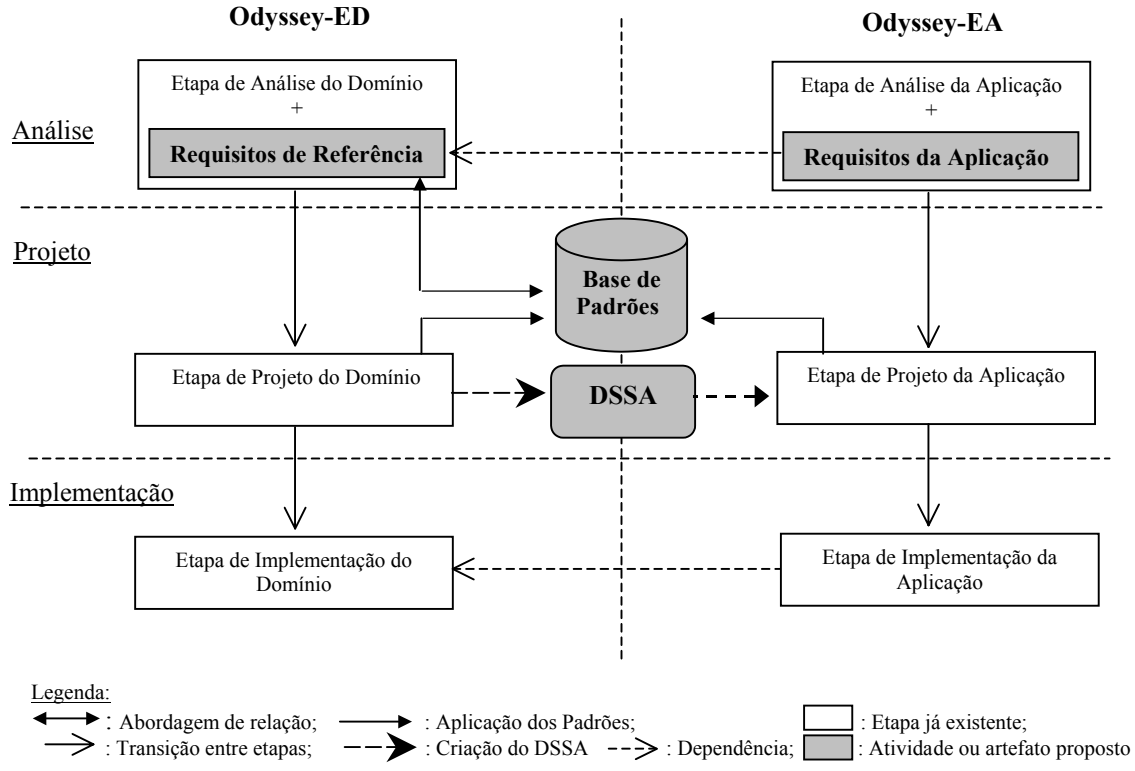


Figura 3.4 Contexto de criação e instanciação de DSSAs no Odyssey

3.3 O Processo de Geração de um DSSA no Contexto do Odyssey - DE

Descrevemos um DSSA como o resultado de um processo de aquisição e modelagem dos elementos de um domínio de aplicação, sob uma perspectiva arquitetural do software. Para que o processo de geração de um DSSA na fase de ED se torne efetivo, é importante especificar as atividades que fazem parte desta estratégia de desenvolvimento, bem como os elementos gerados ao longo deste processo. A adaptação dos conceitos encontrados em (TRACZ, 1994, HAYES, 1994) ao contexto do Odyssey foi realizada no sentido de disponibilizar um processo de ED que enfoque e apoie a aquisição de arquiteturas de software específicas de domínio. Propomos a adição da *fase de especificação dos requisitos arquiteturais de referência de domínio* à etapa de análise e das atividades de seleção e aplicação de padrões arquiteturais OO à fase de definição do modelo geral da arquitetura de domínio no Odyssey-DE.

3.3.1 A Etapa de Análise do Domínio

O processo de criação de um DSSA no Odyssey começa com a etapa de análise de um certo domínio de aplicações. O propósito desta etapa é fornecer aos indivíduos uma compreensão, sem ambigüidades, dos vários aspectos relativos ao domínio. Para que o propósito de criação da arquitetura de referência seja alcançado, necessitamos explicitar quais e de que maneira os elementos identificados na etapa de análise do domínio poderão contribuir para a obtenção deste objetivo. Dos modelos conceituais que advêm das etapas descritas no item 3.2.1.3, são o modelo de casos de uso e os modelos OO relacionados os que influenciarão a produção dos componentes que constituirão a arquitetura de domínio.

A maneira como isso ocorre inicia-se através dos casos de uso genéricos identificados na análise funcional do domínio, a partir das seguintes atividades:

1. *Elicitação dos atores e casos de uso*: O resultado desta atividade é a construção de um modelo de casos de uso com a definição das responsabilidades dos atores envolvidos, a descrição inicial dos casos de uso identificados e a representação do modelo de casos de uso como um todo (relacionamentos, generalizações, etc.);

2. *Priorização dos casos de uso*: Como o ciclo de vida do processo do Odyssey-ED adotado é o espiral, a priorização dos casos de uso permitirá que as subseqüentes fases do desenvolvimento sejam orientadas inicialmente pelos casos de uso de maior prioridade até que toda a funcionalidade do domínio esteja disponibilizada;

3. *Detalhamento dos casos de uso*: a descrição dos fluxos de eventos, dos desvios de execução e dos estados para cada caso de uso de domínio permite uma

representação mais detalhada das funções e dos conceitos envolvidos durante cada instanciação de um caso de uso particular.

Os casos de uso levantados na visão funcional influenciam a identificação dos vários conceitos presentes no modelo conceitual do domínio, que por sua vez, através de interações entre estes conceitos, dão sustentação às funções e/ou processos identificados pelos casos de uso de domínio. Tais conceitos de domínio são agrupados em um modelo de *classes* utilizado, para a representação de entidades do domínio e os possíveis tipos de relacionamentos existentes (herança, associação, composição, dependência, etc). A natureza das colaborações entre classes influi na definição da arquitetura do domínio a ser estabelecida em fase posterior à análise conceitual.

A criação do modelo de análise do domínio é iniciada pela definição dos principais subsistemas, de suas entidades relevantes e de requisitos comuns, para que o "todo" possa ser gerenciado a partir de suas "partes". A identificação inicial dos subsistemas do domínio é a tarefa mais significativa para a definição da arquitetura do domínio e é realizada baseando-se nos requisitos funcionais e no domínio do problema.

A utilização dos casos de uso para a definição de subsistemas do domínio caracteriza-se como uma maneira de organizar as funcionalidades do domínio, pois uma das formas de estruturação dos casos de uso em modelos OO é agrupar os processos característicos do domínio em unidades (ou subsistemas) funcionais (JACOBSON *et al.*, 1999). Esta organização funcional em subsistemas mostra-se favorável à obtenção de componentes funcionalmente mais coesos.

A etapa de análise do domínio deverá fornecer ao projetista do domínio o arcabouço para o início do projeto de domínio. De posse dos modelos de casos de uso, dos conceitos de domínio e dos principais subsistemas identificados na fase de especificação de domínio, identificamos a necessidade de extensão desta etapa para a representação dos principais requisitos de referência do domínio, conjunto de requisitos típicos que as aplicações construídas neste domínio devem satisfazer, assunto este discutido a seguir.

3.3.1.1 A Fase de Especificação dos Requisitos Arquiteturais de Referência no Odyssey - ED

Durante a fase de análise de domínio, é comum serem identificadas algumas características que são genéricas a todo o domínio. Tais características são

consideradas requisitos de referência, porque estão relacionadas ao todo representado pelo domínio e não especificamente a algum de seus casos de uso. Por exemplo, um domínio de aplicação pode ser inerentemente distribuído ou apresentar requisitos funcionais que mudam constantemente, exigindo-se que tal natureza seja considerada na fase de especificação do problema e aplicada, principalmente, na fase de projeto de soluções que atendam a estas características.

Os requisitos arquiteturais de referência influenciam diretamente a forma com que um domínio é representado e, principalmente, estruturado. Tais requisitos são importantes porque representam uma diretriz para o projeto do domínio, na medida que influenciam a criação de um modelo de solução que os suportem. Esta arquitetura servirá como referência a todas as aplicações que são construídas sob este contexto.

A adição da fase de especificação dos requisitos arquiteturais de referência à etapa de análise do domínio permite que sejam explicitados alguns dos critérios que contextualizam e caracterizam domínios de aplicações e que, normalmente, são pouco destacados pelos modelos funcionais e conceituais. No Odyssey, estes requisitos têm grande destaque porque é através deste conjunto que se pretende apoiar a passagem do modelo conceitual do domínio para o modelo de projeto, representada pela arquitetura de referência do domínio. A figura 3.5 mostra, resumidamente, as atividades do processo de Engenharia de Domínio do Odyssey com a adição da fase de especificação dos requisitos de referência.

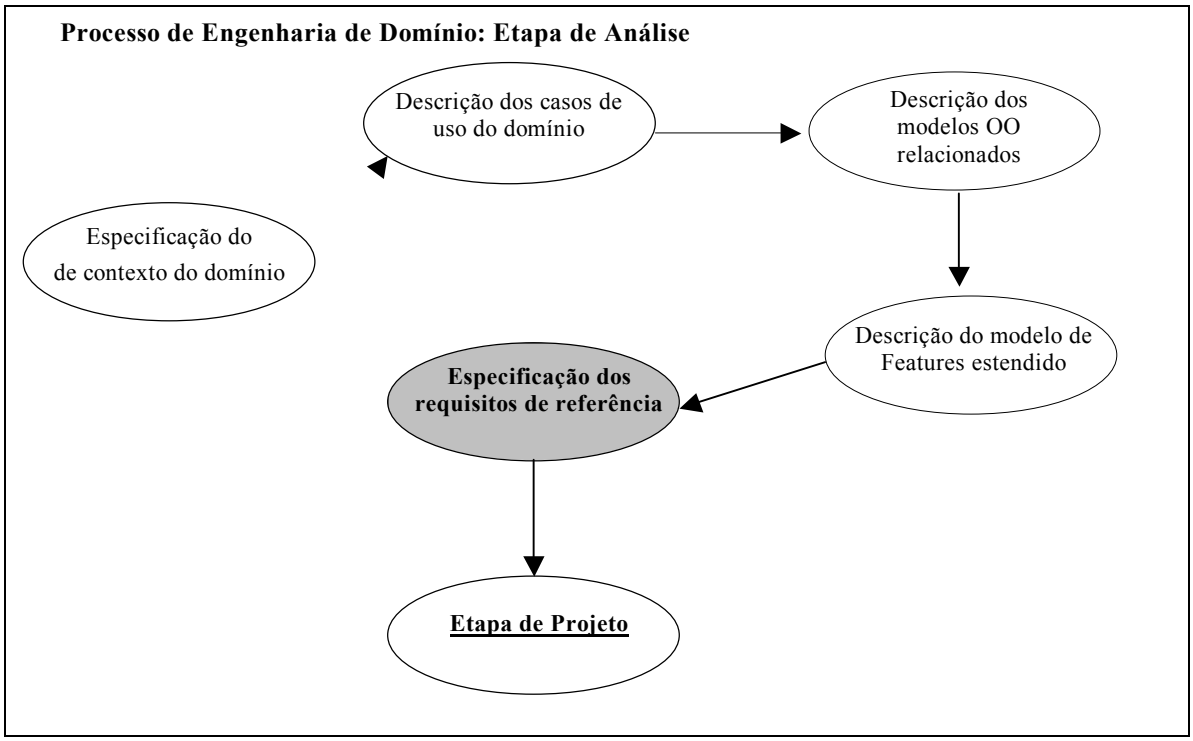


Figura 3.5: Atividades da etapa de análise de domínio do Odyssey-ED

Ao propormos a incorporação de uma fase específica de especificação de características arquiteturais de referência, seguimos uma linha de trabalhos que reconhecem a influência das características de qualidade para o projeto de software (CHUNG *et al.*, 1999, CYSNEIROS, 2001). Nossa abordagem, no entanto, não trata da questão sobre a representação dessas características em grafos para o tratamento de suas interdependências (CHUNG e NIXON, 1995). Nossa intenção está centrada apenas na utilização de um conjunto de características arquiteturais de referência que permita a representação da natureza não funcional do domínio. Para isso, utilizamos uma maneira de caracterização que se baseia na avaliação do grau de relevância que cada item deste conjunto tem para o domínio sob análise.

Para que esta representação se torne efetiva no processo Odyssey-ED, é importante especificar quais as características arquiteturais de referência que desejamos avaliar e em que momento da etapa de análise estas características podem ser identificadas.

A identificação das características mais significativas do domínio em geral encontra correspondência nas características de qualidade identificadas nos produtos desenvolvidos neste mesmo domínio. Para que haja efetividade nesta identificação, sem que a cada domínio diferentes especialistas representem conjuntos distintos de características, propomos a utilização de um conjunto mínimo baseado nas características de qualidade descritas na norma ISO/IEC 9126 (ISO9126, 1992). Justificamos mais detalhadamente nossa escolha no próximo capítulo, mas, fundamentalmente, esta se deve ao fato de ser uma iniciativa amplamente aceita de padronização das principais características relacionadas à qualidade de um produto de software. A lista das características de qualidade da ISO/IEC 9126 pode ser encontrada no anexo 3 desta tese.

É ao final da etapa de análise de domínio que a fase de especificação das características de referência pode ser realizada. Durante esta fase, o especialista de domínio poderá representar os elementos que irão permitir uma melhor definição de um modelo de solução para o domínio. As características arquiteturais de referência, bem como os subsistemas identificados durante a fase de modelagem do domínio, alimentarão a fase de projeto do domínio, fornecendo ao projetista os elementos que serão refinados a ponto de estarem suficientemente detalhados para que possam ser implementados.

3.3.2 A Etapa de Projeto do Domínio

Conforme visto anteriormente, é na etapa de projeto do domínio que as principais funcionalidades e conceitos identificados na etapa de análise começam a receber as considerações arquiteturais e de projeto detalhado. O principal artefato definido nesta etapa é a arquitetura de referência. É ela que dá forma à topologia da ligação entre os componentes e estabelece uma base para o refinamento do projeto da solução.

A decisão mais significativa da etapa de projeto de domínio é a escolha de um ou mais estilos que têm a função de orientar a composição dos componentes, para dar origem à arquitetura de referência. A especificação desta arquitetura inicia-se, justamente, por um modelo de abstração baseado nestes estilos arquiteturais. É importante que não haja confusão entre um modelo da arquitetura de referência e a própria arquitetura. O primeiro representa a natureza dos componentes, das conexões e restrições de combinações e a segunda representa um artefato em um nível de abstração mais próximo da implementação.

A decisão de se utilizar os modelos de composição propostos pelos estilos e padrões arquiteturais precisa ser apoiada por atividades que auxiliem tanto a seleção de um ou mais padrões arquiteturais quanto a sua utilização. Propomos a incorporação deste suporte à fase de definição de um modelo geral da arquitetura de referência do Odyssey-DE detalhando estas atividades. A figura 3.6 mostra, resumidamente, as atividades da etapa de projeto da Engenharia de Domínio do Odyssey.

3.3.2.1 A Atividade de Seleção e Utilização dos Padrões Arquiteturais

A utilização dos padrões arquiteturais OO durante a estruturação da arquitetura de referência em infra-estruturas de reutilização mostra-se adequado e com implicações positivas sobre a elaboração de arquiteturas de domínio. Contudo, a dificuldade desta utilização passa necessariamente pela seleção de um padrão a ser aplicado durante a etapa de projeto da arquitetura do domínio. Algumas heurísticas são sugeridas na literatura (CLEMENTS, 1995, SHAW e CLEMENTS, 1996). Porém, estas abordagens não representam contribuições efetivas para a escolha de um padrão arquitetural, sendo muito abstratas e subjetivas para apoiar, por exemplo, usuários com pouca experiência. Tal deficiência pode ser comprovada pelo pequeno número de trabalhos que utilizam variações destas heurísticas para a identificação de estilos ou padrões arquiteturais (KAZMAN e KLEIN, 1999).

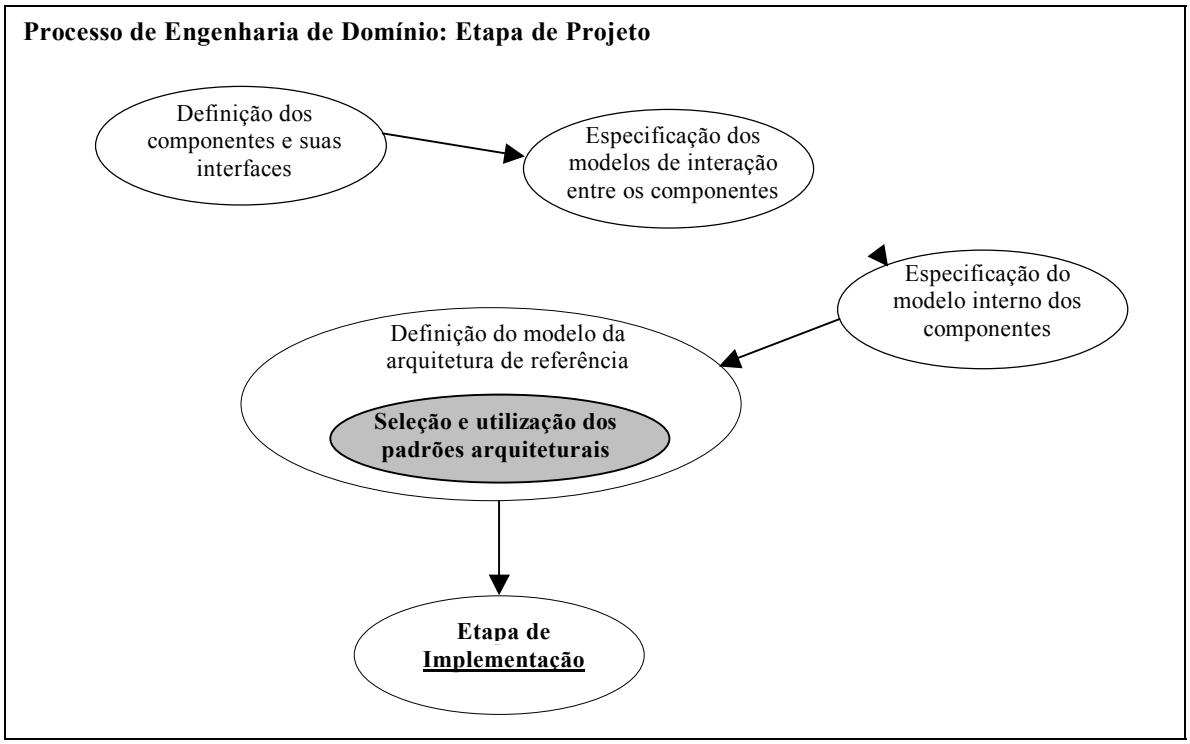


Figura 3.6: Atividades da etapa de projeto de domínio do Odyssey-ED

Para que possamos realizar a fase de modelagem da arquitetura de referência utilizando os padrões arquiteturais OO, é preciso primeiro que haja a definição de um critério de seleção que se baseie em alguma característica do domínio, de maneira que a solução decorrente da utilização de um padrão esteja afim com as expectativas do domínio. A abordagem que propomos utiliza a especificação dos requisitos arquiteturais de referência. Esta abordagem tem se mostrado útil em muitos aspectos, porque enfatiza critérios importantes para o sucesso do software, além dos seus requisitos funcionais (CHUNG e NIXON, 1995). Uma conseqüência importante e direta desta abordagem é a representação explícita das características que influenciaram a construção de determinada solução para o domínio. É muito comum que depois de algum tempo não se tenha o registro do porquê determinada decisão foi tomada e quais os motivos que levaram a uma certa implementação. A utilização de uma representação consistente para a arquitetura do domínio possibilita ainda que as pessoas envolvidas durante este processo tenham um entendimento comum da solução.

Para que os requisitos de referência apóiem a transição entre os modelos conceituais e arquiteturais e que, além disso, o projetista possa utilizar durante a criação da arquitetura de referência o conhecimento descrito pelos padrões arquiteturais, é imprescindível que se possua uma visão relativa às deficiências e

aplicabilidades sobre todos os padrões em função destes requisitos. A dificuldade encontrada neste ponto é, justamente, obter e representar o conhecimento da utilização de um padrão em situações que se espera um conjunto de requisitos arquiteturais de domínio e como disponibilizar estas informações ao projetista para que possam ser utilizadas durante a especificação arquitetural do domínio. Surge desta forma a necessidade de se criar uma base de padrões arquiteturais que possa ser consultada pelo engenheiro de domínio para a análise, escolha e descrição de um ou mais padrões que fazem sentido para o domínio em questão. Esta base cresceria com o tempo, através da adição de mais padrões arquiteturais que fossem reconhecidos como direcionadores eficientes na tarefa de construção de uma arquitetura, de variações dos padrões existentes e por arquiteturas formadas em aplicações do domínio que mereçam ser referenciadas em outros desenvolvimentos.

Outro aspecto importante da criação desta base de padrões está na forma de visualizarmos os padrões arquiteturais como entidades reutilizáveis e independentes dos componentes que fazem parte do domínio. Ou seja, cada padrão arquitetural representa uma decisão de projeto em maior nível, aplicável a inúmeros contextos e que prescreve apenas a natureza dos seus componentes (pipes, camadas, modelos, etc) sem que com isso estejam amarrados a nenhum componente de domínios específicos (como por exemplo, a um analizador léxico, a *business layers,* uma entidade “nota fiscal”, etc). Existe uma expectativa em relação aos benefícios que esta estratégia proporciona, tais como:

a. Modificações na estrutura da arquitetura não afetariam a implementação interna dos componentes de domínio, desde que a interface dos componentes se mantenha a mesma;

b. As arquiteturas poderiam ser consultadas, salvas e recuperadas da mesma forma que um componente de domínio;

c. Poderiam ser usadas algumas ferramentas de análise arquitetural (BARBACCI *et al.,* 1998, KAZMAN *et al.*, 1999) antes que se tenha especificado completamente o mapeamento entre os componentes de domínio e os da arquitetura;

d. Compartilhamento de padrões arquiteturais para domínios próximos.

Através da especificação dos requisitos de referência desejados e do conhecimento representado na base de padrões arquiteturais, podemos realizar uma comparação entre os requisitos identificados durante a análise e os obtidos pela utilização de um padrão, com a intenção de se chegar a um indicador de padrão que melhor represente uma solução para o atendimento aos requisitos esperados.

Esta indicação é possível porque a abordagem de seleção de padrões arquiteturais no contexto do Odyssey utiliza uma base de conhecimento fundamentada na relação existente entre os requisitos arquiteturais de referência de domínio e os padrões arquiteturais OO. Com isso, a atividade de seleção de padrões arquiteturais pode se apoiar em uma abordagem de indicação que busca identificar os padrões que, através de sua aplicação, mais adequadamente suportam os requisitos arquiteturais de referência especificados para o domínio. No capítulo a seguir, podem ser encontrados maiores detalhes sobre esta abordagem.

A atividade de seleção só pode ter início após a conclusão da especificação dos requisitos de referência do domínio, na forma da avaliação do grau de relevância de cada um dos seus itens para o domínio. A indicação de possíveis padrões arquiteturais permitirá que sejam avaliadas alternativas de aplicação de determinadas soluções arquiteturais ao domínio. O resultado desta indicação não tem como objetivo indicar o melhor padrão arquitetural para o domínio em questão, mas sim as possibilidades arquiteturais coerentes com as características levantadas durante a avaliação do problema que podem ser utilizadas durante a fase de projeto. A indicação dos padrões também é influenciada pelas questões funcionais do domínio, mas esta relação é melhor identificada pelo projetista da arquitetura tendo em vista a própria organização do conhecimento associado a estrutura dos padrões. Neste ponto, é possível ter acesso às informações do padrão arquitetural possibilitando, desta forma, uma visão mais completa sobre o escopo de sua aplicação.

Nos casos em que, após o processo de indicação, não se consiga chegar a um consenso sobre o possível padrão arquitetural para o domínio, o projetista de domínio pode tomar sua decisão baseando-se na sua experiência para estabelecer a arquitetura de referência, ou pode utilizar a documentação existente na base de padrões para apoiar os critérios de sua decisão. Nesta situação, o projetista assume a responsabilidade pela obtenção das características de qualidade do domínio através da arquitetura que melhor lhe entenda.

Após ter sido realizada a indicação dos padrões e considerando a situação em que pelo menos um padrão arquitetural OO foi selecionado, em seguida deve ter início a fase de modelagem da arquitetura de referência. A diagramação desta arquitetura deve seguir o modelo proposto pelo padrão selecionado, através da utilização dos componentes do padrão, bem como pelas restrições para a composição destes mesmos componentes. Cada tipo de componente traz consigo uma interface previamente definida, para que um conceito de domínio particular implemente determinada responsabilidade na arquitetura de referência. Uma vez que, na fase de

análise de domínio, os principais conceitos do domínio foram identificados, podemos agora, na diagramação da arquitetura do domínio, estabelecer a ligação entre estes elementos e seus correspondentes na arquitetura, de maneira que, por exemplo, no contexto do padrão arquitetural Modelo-Visão-Controle (MVC), determinado conceito deva assumir um papel de '*Modelo*', ou que dependa de um componente '*Controle'* para a realização de suas responsabilidades (figura 3.7), de forma semelhante a que é suportada pela ferramenta "Framework Studio" (BLUEPRINT, 2001). Além de compor a arquitetura, o projetista estará utilizando um mecanismo que realiza a junção entre o domínio do problema e o da solução, para que a funcionalidade esperada esteja mapeada nos elementos que constituem a arquitetura.

Para que haja independência entre os elementos de domínio e os componentes da arquitetura, o processo de utilização dos padrões deve preservar os conceitos sobre os quais são aplicados. Isso é importante porque pode haver situações em que seja necessário reaplicar um outro padrão na modelagem da arquitetura de referência ou que se deseje ter duas ou mais arquiteturas para o domínio.

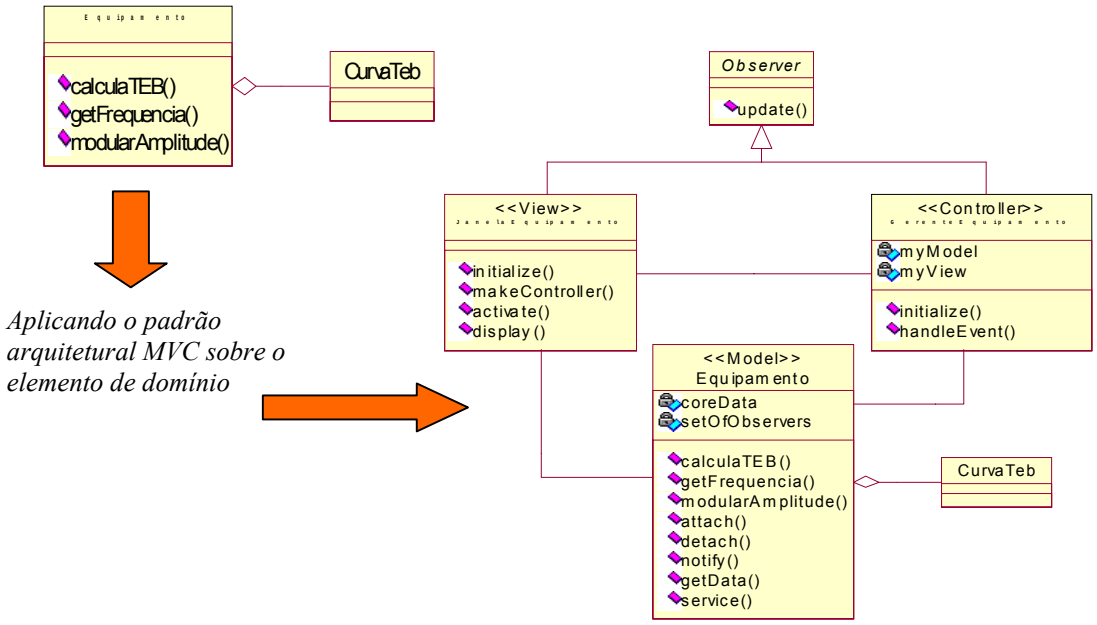


Figura 3.7: Modelo da aplicação de um padrão arquitetural

Esta característica de retorno ao ponto de origem da solução arquitetural é possível de ser feita com a aplicação do padrão sobre uma cópia do elemento de domínio. A cópia sobre a qual o padrão é aplicado mantém uma referência a sua

origem, possibilitando a rastreabilidade dos conceitos de domínio e o suporte as demais situações apresentadas (figura 3.8).

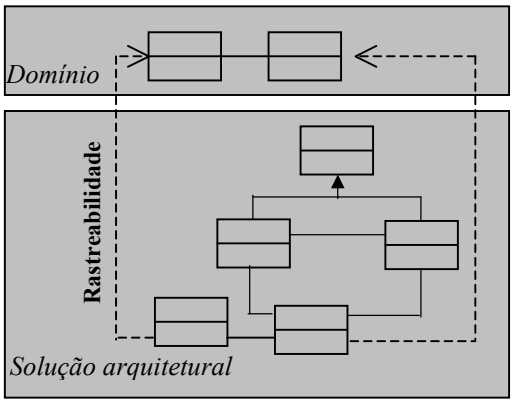


Figura 3.8: Rastreabilidade entre componentes

É importante ressaltar que nada impede que determinado domínio apresente múltiplas arquiteturas de referência. Tal situação pode ser necessária quando o projetista do domínio vislumbra antecipadamente a necessidade de suportar cenários de projeto arquitetural que dificilmente podem ser contemplados com o suporte de uma única arquitetura. Esta possibilidade de escolha fortalece a aplicabilidade das arquiteturas de referência durante a instanciação de aplicações neste domínio, porque amplia o leque de opções arquiteturais condizentes com o contexto de projeto de cada aplicação.

Ao término desta tarefa, teremos especificado o modelo da arquitetura de referência e iniciado a especificação da interface básica dos componentes. Em seguida, o projetista deve refinar o projeto interno de cada componente até que seja possível a implementação da solução arquitetural.

3.4 O Processo de Instanciação de um DSSA no Contexto do Odyssey - EA

O processo complementar à criação de uma arquitetura de referência é a sua instanciação. Aplicações produzidas em um domínio espelham-se na seleção de funcionalidades e entidades identificadas e suas respectivas arquiteturas são customizadas através da arquitetura de domínio.

A adaptação desta sistemática ao processo Odyssey-EA permite que sejam identificadas as fases envolvidas na criação de aplicações através da reutilização da arquitetura de software específica de domínio, bem como as possibilidades de uso de ferramentas automatizadas ou semi-automatizadas que apóiem este processo de

instanciação. Das etapas identificadas na seção 3.2.2 sobre o processo de EA do Odyssey, enfocamos as que diretamente dependem dos artefatos produzidos na ED e que são relacionadas ao processo de criação de DSSAs. Propomos ainda a incorporação de uma fase na etapa de análise da aplicação para a especificação de diretrizes de projeto e a fase de instanciação da arquitetura de referência no contexto da aplicação.

3.4.1 A Etapa de Análise da Aplicação

A etapa de análise do processo de EA tem como objetivo a identificação dos principais conceitos e funcionalidades de uma aplicação. Nesta etapa, o engenheiro de software leva em consideração apenas os aspectos conceituais e funcionais da aplicação, em um nível de abstração que permita a definição do seu escopo em relação ao domínio em que está inserido. Além dos modelos de análise da aplicação propostos para o Odyssey-EA, identificamos a necessidade de mais uma fase na etapa de análise. Esta fase deve dar o apoio inicial à instanciação da arquitetura de referência.

3.4.1.1 A Fase de Especificação das Diretrizes de Projeto

Analogamente à fase de especificação dos requisitos arquiteturais de referência na etapa da análise de domínio do Odyssey-ED, propomos a fase de especificação das diretrizes de projeto, cujo objetivo principal é apoiar a instanciação da arquitetura de referência do domínio no contexto da aplicação. Esta fase se realiza após a especificação dos componentes do modelo, momento onde os elementos necessários para a definição completa do modelo de análise da aplicação estão suficientemente detalhados (MILLER, 2000).

Durante a fase de especificação das diretrizes de projeto, o Engenheiro de Software terá acesso à caracterização do domínio de sua aplicação em função dos requisitos arquiteturais de referência e poderá reavaliar este conjunto em função de adaptações realizadas nos conceitos e funcionalidades extraídos do domínio, caso seja necessário. O desenvolvedor deve, porém, estar ciente que modificações na reavaliação dos requisitos arquiteturais de referência do domínio poderão não mais encontrar suporte da arquitetura de software específica de domínio, sendo necessário a reformulação do DSSA ou construção da uma arquitetura de software específica para a aplicação.

A fase de especificação das diretrizes de projeto inicia-se quando o Engenheiro de Software solicita o resultado da avaliação dos requisitos arquiteturais de referência do domínio que está reutilizando. Este resultado permitirá que sejam identificadas as características mais importantes do domínio e, principalmente, explicitadas as informações que influenciam a criação da arquitetura de software específica de domínio. Após o entendimento do resultado da avaliação dos requisitos de referência, o desenvolvedor poderá aceitá-los, estabelecendo assim uma correspondência entre os requisitos do domínio e da aplicação, ou não, por identificar que por algum motivo a aplicação não se encaixa nesta descrição do domínio, possivelmente por ter introduzido modificações na aplicação que a descaracterizou dos requisitos arquiteturais do domínio. Na situação em que o resultado da avaliação é aceito pelo Engenheiro de Software, os requisitos arquiteturais de referência do domínio servirão de modelo para a criação do conjunto de requisitos da aplicação. Esta criação está baseada na cópia da especificação das características do domínio para o contexto da aplicação. Caso não haja a aceitação do resultado da avaliação, um conjunto de novos requisitos específicos da aplicação deverá ser utilizado.

3.4.2 A Etapa de Projeto da Aplicação

A etapa de projeto de um processo de Engenharia de Aplicações tem como objetivo principal o refinamento dos conceitos abstratos provenientes da fase de análise e a reutilização dos componentes no nível de projeto oriundos da Engenharia de Domínio, principalmente através da instanciação da arquitetura de software específica de domínio. Esta, por sinal, é a primeira fase realizada nesta etapa, estando a arquitetura de referência já previamente definida pelo processo de ED. Assim, uma ou mais possibilidades arquiteturais podem ser fornecidas ao desenvolvedor como modelos de instanciação da arquitetura da aplicação, apoiando a conclusão desta fase.

3.4.2.1 A Fase de Instanciação da Arquitetura de Referência

O principal cenário de realização da fase de instanciação da arquitetura ocorre quando o engenheiro de software concorda com a arquitetura de referência estabelecida no projeto arquitetural do domínio (ou seleciona uma dentre possíveis soluções arquiteturais). Este acordo é dependente de outra aceitação, anteriormente realizada, e que condiciona a utilização do DSSA: a conformidade entre os requisitos arquiteturais de referência do domínio e os requisitos da aplicação. Nesta fase, há a

necessidade de informar ao engenheiro de software se houve a aceitação dos requisitos de domínio na fase de especificação das diretrizes de projeto, sem que este tenha que sair do contexto do projeto e retorne ao da análise da aplicação. Esta conformidade indica que a arquitetura da aplicação será uma instanciação (completa ou parcial) da arquitetura de referência, que por sua vez busca suportar os requisitos arquiteturais de referência do domínio, estando estes completamente de acordo com os requisitos da aplicação.

Devemos lembrar que a arquitetura de referência é composta por uma lista de componentes (*features* de projeto) individualmente ligados, através de um mecanismo de rastreabilidade, aos modelos de projeto do domínio que lhes deram origem (MILLER, 2000). Esta rastreabilidade permitirá que sejam identificados os componentes do DSSA que farão parte da arquitetura da aplicação, a partir dos componentes selecionados para a aplicação (atividade realizada na análise da aplicação).

A instanciação da arquitetura segue uma seqüência de ações representadas na figura 3.9 pelos passos 1, 2 , 3 e 4.

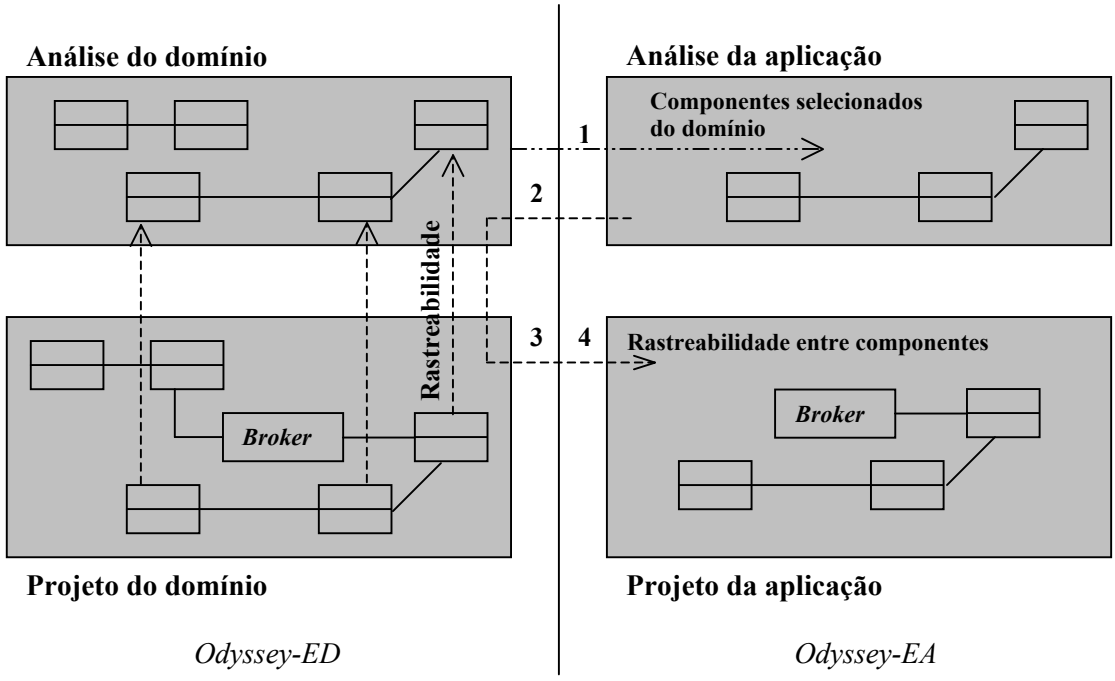


Figura 3.9: Cenário de instanciação da arquitetura da aplicação

A fase de instanciação inicia-se com a seleção do conjunto de componentes na análise da aplicação (1). Esta seleção permite que sejam identificados os componentes correspondentes no contexto do domínio (2). O modelo arquitetural de referência do domínio, com a sua ligação de rastreabilidade com os componentes

conceituais do domínio (3), pode então ser instanciado, ou seja, trazido para o contexto da aplicação (4) onde as atividades de detalhamento subseqüentes podem ocorrer. A instanciação se dá pela cópia dos componentes da arquitetura de domínio para o contexto da aplicação e apenas os componentes da arquitetura de referência que encontram correspondentes no modelo de análise da aplicação serão instanciados.

Na situação em que existam componentes específicos ao contexto da aplicação (ou seja, que não há correspondentes no modelo de análise do domínio), estes talvez precisem ser adaptados de forma a preservar o seu modelo arquitetural original. Esta adaptação é possível quando, por exemplo, um componente específico da aplicação segue a interface de um dos componentes prescritos pelo padrão arquitetural utilizado na arquitetura de referência, tal como mostrado na figura 3.7.

O outro cenário de instanciação ocorre quando o desenvolvedor não aceita a avaliação do conjunto de requisitos de referência especificado para o domínio da aplicação. Com isso, a instanciação da arquitetura dependerá de uma reavaliação da possibilidade de aplicação de padrão arquitetural e de uma verificação sobre o resultado proposto, validando a manutenção do DSSA na aplicação. Caso seja mantido o DSSA para a instanciação da arquitetura de software para aplicação, então teremos o mesmo cenário de instanciação descrito anteriormente; caso contrário, deixa de haver o processo de instanciação do DSSA e esta fase passa a ser apenas a utilização de padrões arquiteturais no contexto do projeto da aplicação (figura 3.10).

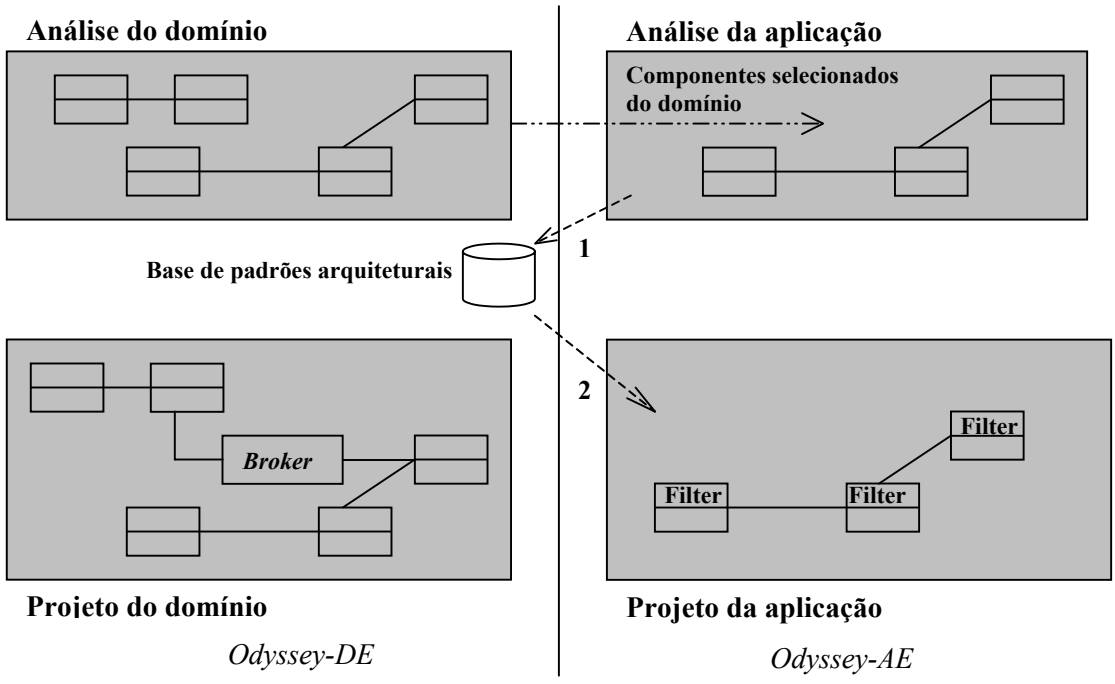


Figura 3.10: Cenário de modelagem da arquitetura da aplicação

Para completar este cenário, o desenvolvedor deve perceber que a arquitetura da aplicação seguirá um padrão arquitetural determinado por ele (1), ficando livre para formalizar as ligações entre os componentes da aplicação (2), de forma idêntica ao descrito no cenário de aplicação de um padrão arquitetural. Nesta situação, a arquitetura da aplicação deixa de ter ligação com o DSSA de seu domínio, sendo um artefato completamente independente.

Ao final desta fase, teremos alcançado um modelo geral da arquitetura da aplicação, onde poderá ser realizado o projeto detalhado dos componentes da aplicação, preparando-os para a etapa de implementação da aplicação (MILLER, 2000). Umas das principais funcionalidades desta etapa é a possibilidade de verificação de padrões e anti-padrões de projeto existentes no modelo de aplicações (CORREA *et al.,* 2000), permitindo que se detecte eventuais construções ruins e que seja possível a realização de mudanças que possibilitem uma construção de software OO mais eficiente.

3.5 Conclusão

As Engenharias de Domínio e de Aplicações carecem de sistemáticas que explorem os benefícios encontrados no desenvolvimento centrado em arquiteturas de software (BOOCH, 1996, BRAGA *et al.*, 1999). Um dos principais desafios destas propostas de desenvolvimento de software é a convergência das idéias de construção de software baseadas em componentes e a proposta de “Domain-Specific Software Architectures (DSSAs)”. A união destas abordagens permite explorar a reutilização das características identificadas em domínios de aplicações nos diversos níveis de abstração do processo de desenvolvimento, principalmente no que diz respeito a reutilização de soluções arquiteturais aplicáveis ao domínio.

A iniciativa do detalhamento das fases e atividades necessárias para a realização da criação e instanciação de arquiteturas de referência, tal como realizamos no contexto do Odyssey, contribui para a diminuição da dificuldade encontrada durante a transição dos modelos conceituais para os modelos arquiteturais. Esperamos, assim, contribuir com este trabalho através da discussão e do estabelecimento de diretrizes para o projeto arquitetural.

É importante ressaltar que estas atividades e seus mecanismos de suporte correspondentes podem ser aplicáveis com maior ou menor esforço a outras infra-estruturas de reutilização que se baseiam em processos de engenharia de domínio e de aplicação (GOMAA e FARRUKH, 1999, MANNION *et al.*, 1999).

Entretanto, estamos também conscientes da necessidade de utilização de todas as fases envolvidas nos processos de ED e EA, seja no contexto do Odyssey ou de outra infra-estrutura, para que possamos verificar a efetividade da realização dessas atividades.

Capítulo 4: A Abordagem de Seleção de Padrões Arquiteturais

4.1 Introdução

Vimos que os processos de criação e instanciação de arquiteturas de referência em infra-estruturas de reutilização podem ser apoiados, entre outras coisas, por uma abordagem que favoreça a seleção de determinado padrão arquitetural visando sua aplicação.

Uma das possíveis abordagens utilizadas para realizar esta seleção é a que explora o relacionamento entre os requisitos de referência e os padrões arquiteturais. Com isso, podemos avaliar que influências poderão ter no projeto arquitetural, tendo em vista os requisitos de qualidade especificados, as soluções arquiteturais conhecidas.

Baseado neste relacionamento, uma abordagem para a seleção de padrões arquiteturais é proposta. O objetivo deste capítulo é apresentar os principais pontos da proposta que fundamenta a seleção de padrões arquiteturais no contexto de infra-estruturas de reutilização, bem como o desenvolvimento de uma ferramenta semi-automatizada para suportá-la. Este capítulo está organizado da seguinte forma: na seção 4.2, discutimos a preparação necessária para a avaliação dos padrões arquiteturais; na seção 4.3, apresentamos o resultado obtido com a avaliação dos padrões arquiteturais; na seção 4.4, detalhamos a regra de seleção dos padrões arquiteturais utilizada; na seção 4.5, mostramos os modelos e a interface da ferramenta de seleção arquitetural Mentor que implementa a regra de seleção; por fim, na seção 4.6, finalizamos o capítulo com algumas conclusões.

4.2 A Organização da Avaliação dos Padrões Arquiteturais

Antes de considerarmos qualquer abordagem de seleção, precisamos primeiro verificar se existem indícios de que arquiteturas de software influenciam ou são influenciadas por características específicas. Encontramos na literatura estudos sobre as principais influências existentes na maior parte dos projetos arquiteturais (CHUNG *et al.*,1994, CLEMENTS, 1995, BASS *et al.*, 1998). Em geral, estas influências podem ser classificadas em três categorias principais, sendo relacionadas à natureza social, organizacional e técnica da atividade de projeto de software. Sob este ponto de vista, uma arquitetura representaria o resultado destas influências particulares e sua existência, em contrapartida, influencia o ambiente que a cerca.

Estudando as diferentes influências que uma solução arquitetural traz para o projeto da arquitetura, vimos que as que mais tem relação com a fase de especificação do software e exercem grande impacto sobre a aceitação da solução são as características de qualidade da aplicação. As características de qualidade representam os atributos que, em geral, são desejáveis a uma aplicação. Existem alguns modelos de qualidade de produtos de software (BOEHM *et al.*, 1978, ROCHA, 1983, ISO9126, 1992) que formalizam conjuntos de características de qualidade que contribuem para a avaliação dos produtos de software.

Podemos encontrar na literatura especializada (BUSCHMANN *et al.*, 1996, SHAW e GARLAN, 1996, CLEMENTS, 1996, BOSCH, 2000) descrições sobre os benefícios e dificuldades que freqüentemente ocorrem durante a utilização de determinado estilo ou padrão arquitetural em relação a algumas das características de qualidade. No entanto, tais descrições seguem pontos de vista particulares, de acordo com a perspectiva de cada autor. Muitas vezes um mesmo padrão arquitetural possui descrições semelhantes, mas pouco afins, gerando dúvidas sobre a aplicabilidade destes detalhamentos. Em (BUSCHMANN *et al.*, 1996), por exemplo, o enfoque principal está na apresentação dos benefícios e dificuldades encontrados com a utilização de determinado padrão, independentemente do número e da natureza das características de qualidade envolvidas em cada descrição. Já (BOSCH, 2000), apresenta uma descrição que se baseia em um número fixo de características de qualidade, onde cada padrão é analisado em relação a estas características. Percebemos, também, que não há um critério que apóie a escolha de um conjunto mínimo de características que possibilite a avaliação consistente da utilização dos estilos e padrões arquiteturais.

Para que possamos explorar o conhecimento sobre os benefícios e dificuldades comuns à utilização de estilos e padrões arquiteturais, precisamos analisar e optar por um modelo de qualidade que oriente o processo de formalização deste conhecimento. No contexto deste trabalho escolhemos como ponto de partida o modelo definido pela ISO/IEC 9126-1 (ISO9126, 1992). Esta escolha deveu-se a ser esta uma iniciativa que propõe a consolidação de um modelo de qualidade padrão para a avaliação de produtos de software. Contudo, selecionamos apenas as características de qualidade que julgamos depender mais diretamente da arquitetura do software para a sua realização. Nossa escolha recebeu a influência de trabalhos que relacionam algumas características de qualidade ao projeto arquitetural de software (BASS *et al.,* 1998*,* CHUNG *et al.,* 1999). Este conjunto de características de qualidade pode ser encontrado no anexo 3 deste trabalho.

Após termos definido o conjunto inicial de características arquiteturais de qualidade, precisamos estabelecer as organizações arquiteturais que devem ser os objetos sob avaliação. Em contextos gerais de infra-estruturas de reutilização, podemos utilizar os principais estilos e padrões arquiteturais descritos na literatura (vide itens 2.5 e 2.6 para uma descrição detalhada dos mesmos). No entanto, optamos por utilizar apenas os padrões arquiteturais OO descritos em (BUSCHMANN *et al.*, 1996), visto que o Odyssey adota o paradigma OO.

Uma vez estabelecidas as características de avaliação (características arquiteturais de qualidade baseadas na ISO/IEC 9126-1) e os objetos sob avaliação (padrões arquiteturais OO), precisamos escolher um critério que permita a avaliação desta relação. A maneira como os padrões arquiteturais OO são descritos por si só já sugere um critério de avaliação a ser utilizado. Poderíamos avaliar, por exemplo, se um determinado padrão traz algum benefício ou é desfavorável quando procuramos privilegiar, por exemplo, o desempenho ou a testabilidade do software. Percebemos, no entanto, que esta avaliação está restrita a apenas duas possibilidades de valores, sendo de caráter limitado para ser utilizada como um suporte adicional à atividade de projeto.

A escolha de um critério de avaliação deve estar relacionada ao contexto de projeto de aplicações e que de alguma forma represente a ação direta da utilização de uma solução arquitetural baseada em um padrão arquitetural OO para que as características arquiteturais de qualidade sejam alcançadas. O critério utilizado para alcançar este objetivo é o que se baseia na observação do aparente grau de esforço de desenvolvimento aplicado à fase de projeto arquitetural. Ou seja, em algumas situações, a aplicação de um padrão pode exigir um esforço em projeto bastante alto para que uma ou mais características arquiteturais de qualidade sejam privilegiadas pela arquitetura, podendo inviabilizar o seu uso no contexto do projeto. Com isso, seria melhor o uso de um padrão que se aplique mais adequadamente ao contexto dado pelas características arquiteturais que se deseja privilegiar.

A avaliação deste esforço é feita por elementos classificatórios que dão a idéia da dimensão de sua dificuldade. No contexto deste trabalho, a classificação do esforço é feita pelos elementos “muito alto, alto, médio, baixo, muito baixo e desconhecido[9]”. Esta classificação pretende apoiar a compreensão da influência, baseada na distância

[9] O relacionamento denominado "desconhecido" representa os casos em que para determinada característica arquitetural de qualidade não há informação suficiente que permita uma avaliação segura de sua relação.

conceitual, da distância existente entre a aplicação de um padrão arquitetural e a conseqüente obtenção de características arquiteturais de qualidade.

4.3 A Avaliação dos Padrões Arquiteturais

Para realizarmos as avaliações dos padrões arquiteturais baseadas nas características arquiteturais de qualidade e no critério de esforço em projeto, utilizamos a própria definição dos padrões arquiteturais (BUSCHMANN *et al.*, 1996), complementada com algumas das descrições obtidas em (SHAW *et al.*, 1995, CLEMENTS, 1996, BOSCH, 2000).

A legenda utilizada é uma proposta de classificação inicial do critério de avaliação e organiza as faixas classificatórias relativas à aplicação do padrão para a obtenção de determinada característica arquitetural, da seguinte maneira:

1. Muito Bom (MB): Característica principal do padrão utilizado, sendo sua aplicação completamente favorável;
2. Bom (B): A utilização do padrão é favorável, mas exige algum esforço de projeto para alcançar o benefício almejado;
3. Médio (M): A utilização do padrão aplica-se a situações em que o esforço desprendido se justifica perante o benefício encontrado;
4. Ruim (R): A utilização do padrão é injustificável na maioria das situações;
5. Muito Ruim (MR): A utilização do padrão é altamente desfavorável.
6. Desconhecido (D): Quando não há menção sobre esta relação de esforço;

O resultado desta avaliação pode ser visto na tabela 4.1.

Padrões Arquiteturais / Características arquiteturais	Camadas	Pipes & Filters	Blackboard	Broker	MVC	PAC	Microkernel	Reflection
Interoperabilidade	**B**	**D**	**D**	**B**	**D**	**M**	**M**	**D**
Segurança de acesso	**MB**	**MB**	**M**	**B**	**B**	**M**	**B**	**D**
Maturidade	**M**	**R**	**MR**	**M**	**M**	**M**	**B**	**D**
Tolerância a falhas	**M**	**R**	**MB**	**R**	**M**	**M**	**R**	**R**
Recuperabilidade	**M**	**MR**	**R**	**R**	**D**	**R**	**R**	**D**
Operacionalidade	**B**	**D**	**D**	**D**	**MB**	**MB**	**D**	**D**
Comportamento em relação ao tempo	**M**	**B**	**MR**	**MR**	**M**	**R**	**M**	**R**
Comportamento em relação aos recursos	**R**	**R**	**MR**	**MR**	**M**	**R**	**M**	**R**
Modificabilidade	**B**	**M**	**MB**	**B**	**B**	**R**	**MB**	**MB**
Testabilidade	**MB**	**R**	**MR**	**R**	**B**	**MR**	**M**	**D**
Adaptabilidade	**MB**	**MR**	**D**	**B**	**B**	**D**	**B**	**B**

Tabela 4.1: Avaliação preliminar dos padrões arquiteturais utilizados.

As características arquiteturais de qualidade estão ordenadas na seqüência definidas pelo modelo ISO/IEC 9126-1 e iremos mantê-la ao longo deste trabalho. Todavia, não acreditamos que qualquer mudança neste sentido cause algum impacto significativo na regra de seleção de padrões arquiteturais utilizada, porque, para efeito de simplificação da abordagem, consideramos que as características utilizadas durante a avaliação são independentes umas das outras.

4.4 A Regra para Seleção de Padrões

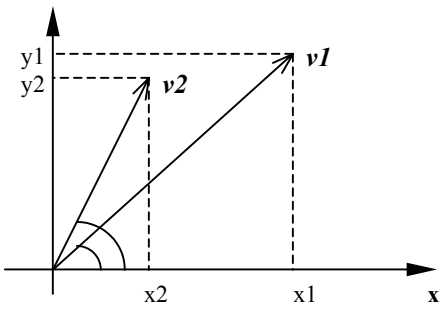
Uma das condições necessárias para que a abordagem de reutilização de software seja realizada é a que está relacionada ao processo de recuperação de componentes. Esta busca de componentes objetiva a classificação de possíveis soluções para que determinada necessidade seja atendida, a partir de um conjunto características de referência.

Existem algumas formas de se realizar estas buscas. Organizações hierárquicas, como o esquema de classificação de livros em bibliotecas, e a classificação facetada (PRIETO-DÍAZ, 1991) de software, são mecanismos utilizados neste sentido.

A organização hierárquica dos padrões arquiteturais mostrou-se inflexível porque o relacionamento existente entre os padrões e as características arquiteturais de qualidade não se dá desta forma, o que tornaria a seleção de padrões impraticável. O esquema de classificação facetada é eficiente, vem sendo utilizado em inúmeros trabalhos e relaciona um componente às facetas[10]. O sistema de classificação facetada depende da automação de um sistema de recuperação que fundamentalmente busca e recupera um componente de acordo com uma especificação facetada qualquer. O problema em se utilizar facetas para apoiar a recuperação de padrões arquiteturais que possivelmente atendam a um conjunto de características arquiteturais de qualidade reside no fato de não estarmos lidando com um problema de busca exata, e sim com uma aproximação entre conceitos. Ou seja, há situações em que podemos procurar um padrão que tenha sido classificado como “esforço muito baixo” para determinada característica e este padrão não existir. Mesmo assim, é importante que a recuperação possa indicar os padrões que estão mais próximos deste objetivo. Trabalhos como o de (BARROS, 1995) lidam com questões relacionadas à busca facetada por aproximação, mas, no contexto deste trabalho, optamos por uma outra forma para a realização deste tipo de busca.

[10] Uma faceta é um tipo de descritor que auxilia a identificação de um componente (PRIETO-DÍAZ, 1991).

Uma forma possível de realização desta busca por aproximação se baseia no conceito matemático de distância euclidiana (BOLDRINI *et al.*, 1980). Este mecanismo de busca explora a possibilidade de avaliação de distâncias conceituais a partir da comparação de elementos em um espaço vetorial multidimensional. A noção de distância conceitual é realizada, matematicamente, pela norma da diferença entre dois vetores ***v1*** e ***v2*** (figura 4.1).



Cálculo da distância:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Figura 4.1: distância entre vetores

O cálculo da distância é realizado após a normalização dos vetores. Esta normalização é importante porque estamos interessados apenas na direção que cada vetor apresenta no espaço vetorial correspondente, minimizando a influência das dimensões dos vetores no cálculo da distância. Com a normalização, temos a equalização da importância das dimensões vetoriais e o enfoque fica apenas direcionado na direção que estes vetores podem assumir (KONTIO, 1995).

A norma, ou comprimento, de um vetor *v* é dado por $\|v\| = \sqrt{\langle v, v \rangle}$. Ou seja, sendo $\langle v, v \rangle$ o produto interno (ou escalar) usual, se *v* = (x, y, z) $\in$ R³ , então o valor encontrado em $\|v\| = \sqrt{\langle v, v \rangle}$ = $\sqrt{x^2 + y^2 + z^2}$ representa a norma do vetor *v.* É importante observarmos que todo vetor não nulo *v* pode ser normalizado, da forma $v/\|v\|$.

Alguns trabalhos mostram a possibilidade de utilização da abordagem vetorial para a avaliação de distâncias conceituais (MONETA *et al.*, 1990, ASADA *et al.*, 1992). Porém, precisamos identificar de que forma podemos utilizar a abordagem vetorial para apoiar o processo de seleção. Ou seja, de que maneira esta abordagem pode viabilizar a utilização do conhecimento obtido com a avaliação dos padrões arquiteturais para que a indicação de possíveis soluções possa atender da melhor forma possível ao conjunto de características arquiteturais almejadas.

Percebemos que a própria organização dada à avaliação dos padrões permitiu sua representação vetorial. Consideramos que as características arquiteturais de qualidade são as dimensões do espaço vetorial e que cada padrão arquitetural é um elemento neste espaço dimensional.

Partindo da observação de que todos os padrões arquiteturais OO podem ser representados através do seu relacionamento com as características arquiteturais de qualidade, iremos formalizá-los através de um espaço vetorial com um número de dimensões igual ao conjunto de características utilizado em nossa abordagem, da seguinte maneira:

$$V = \mathbf{R}^{11} = (X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11})$$

onde V é o vetor que representa um padrão arquitetural no espaço das características arquiteturais de qualidade e Xi representa a avaliação de esforço dada ao padrão para a característica arquitetural de qualidade de índice i, sendo a característica de índice 1 igual a “interoperabilidade”, de índice 2 igual a “segurança de acesso”, e assim sucessivamente, na seqüência de características definidas na seção 4.3.

Para efeito de exemplificação da abordagem de avaliação da distância conceitual baseado na representação vetorial dos padrões arquiteturais, escolhemos um exemplo que será utilizado ao longo desta seção para a sua melhor compreensão. Se considerarmos, por exemplo, os padrões “Camadas” e “Pipes & Filters” como elementos no espaço vetorial dos padrões arquiteturais, suas representações vetoriais são dadas da seguinte maneira:

Camadas = (B, MB, M, M, M, B, M, R, B, MB, MB); e

Pipes & Filters = (D, MB, R, R, MR, D, B, R, M, R, MR)

Após termos especificado a representação vetorial dos padrões arquiteturais, precisamos transformar os conceitos subjetivos utilizados nesta representação em valores numéricos, de forma a ser possível o cálculo vetorial das distâncias. Uma forma de tratamento desta quantificação de valores é a que atribui pesos a classificação de conceitos subjetivos. Para efeito de validação da proposta, estabelecemos que a faixa de valores utilizados em qualquer quantificação de classificações subjetivas está definida pelo intervalo [0..1].

Especificamente para os padrões arquiteturais, a subjetividade dos conceitos localiza-se na aparente relação de esforço existente entre o padrão e determinada característica de qualidade. Sua classificação é feita pelos elementos “muito alto, alto, médio, baixo e desconhecido”, como foi visto na seção anterior.

A atribuição dos valores para o grau de esforço foi realizada respeitando a faixa pré-estabelecida e estabelecendo que os itens “muito baixo” e “muito alto” estariam

nos extremos da faixa de valores possíveis. Buscando, ainda, estabelecer distâncias bem próximas entre os itens do conjunto para representar proporcionalmente o crescimento do grau de esforço dentro do intervalo de valores utilizado. Atribuímos os pesos descritos na tabela 4.2 aos elementos classificatórios utilizados na avaliação dos padrões:

Grau de esforço	Valores atribuídos
Muito bom	1
Bom	0.7
Médio	0.5
Ruim	0.3
Muito ruim	0

Tabela 4.2: Quantificação dos graus de esforço especificados na avaliação dos padrões

Este mapeamento quando realizado gera a representação quantitativa para cada uma das dimensões do padrão. Ou seja, no exemplo da utilização da abordagem, os padrões arquiteturais “Camadas” e “Pipes & Filters” passam a ser representados da seguinte maneira:

Camadas = (0.7, 1, 0.5, 0.5, 0.5, 0.7, 0.5, 0.3, 0.7, 1, 1); e

Pipes & Filters = (D, 1, 0.3, 0.3, 0, D, 0.7, 0.3, 0.5, 0.3, 0)

Propomos ainda que o valor a ser atribuído ao nível de esforço "desconhecido" seja estabelecido a partir de uma simulação desta relação que permite ao projetista a avaliação do impacto destas características desconhecidas de três maneiras distintas:

- Otimista: Subentende-se que o padrão possivelmente atenderia bem a característica. Pressupõe-se que os relacionamentos desconhecidos contribuem em um grau que esteja entre o muito bom e o bom;
- Neutra: Aposta-se que a aquisição da característica exigiria um grau razoável de adaptação da arquitetura. Pressupõe-se que os relacionamentos desconhecidos contribuem em um grau que seja igual ao valor médio;
- Pessimista: O projetista não considera que a utilização do padrão possa atender favoravelmente a característica. Pressupõe-se que os relacionamentos desconhecidos contribuem em um grau que esteja entre o ruim e o muito ruim.

Nestas simulações, o projetista pode confrontar os resultados obtidos nas três visões e inferir quais das perspectivas podem ser aplicadas nas situações de especificação arquitetural. A idéia representada nesta simulação é a de que mesmo não conhecendo como uma determinada característica arquitetural desejada é

suportada por um padrão arquitetural, o impacto destas características sob as visões especificadas anteriormente pode ser calculado.

A quantificação de um grau para a representação do tipo de simulação adotado deve pertencer a mesma faixa de valores condizentes com os graus atribuídos nos relacionamentos conhecidos. Ou seja, uma abordagem otimista deve atribuir ao relacionamento "desconhecido" um valor que represente um relacionamento que esteja entre o muito bom e o bom, sendo realizado de forma semelhante para os outros dois tipos possíveis de simulações. Atribuímos os valores representados pela tabela 4.3 para a representação e utilização na construção do espaço vetorial de padrões arquiteturais.

Simulação	Valores atribuídos
Otimista	0,8
Neutra	0,5
Pessimista	0,2

Tabela 4.3: Quantificação das possibilidades de simulação

Supondo que tenhamos escolhido uma simulação neutra para o tratamento do grau de esforço desconhecido, temos então os padrões arquiteturais “Camadas” e “Pipes & Filters” utilizados no exemplo da seguinte maneira:

Camadas = (0.7, 1, 0.5, 0.5, 0.5, 0.7, 0.5, 0.3, 0.7, 1, 1); e

Pipes & Filters = (0.5, 1, 0.3, 0.3, 0, 0.5, 0.7, 0.3, 0.5, 0.3, 0)

É importante perceber que somente após a escolha de uma abordagem de simulação é que realizamos a quantificação completa de toda a avaliação dos padrões. Somente quando houver conhecimento suficiente para garantir uma avaliação completa dos padrões é que este tipo de simulação deixa de ser importante para o contexto desta abordagem seleção.

Analogamente à avaliação dos padrões, precisamos definir de que forma iremos realizar a representação vetorial da especificação das características arquiteturais de referência. Esta especificação representa o objetivo de projeto a ser atingido com a utilização dos padrões arquiteturais.

Propomos, então, a especificação deste conjunto a partir de uma classificação que demonstre o valor de cada característica arquitetural no contexto específico de desenvolvimento arquitetural. Em nosso trabalho, o papel de cada característica pode ser avaliado como:

1. Importante (IM): É fundamental que a característica esteja presente no software;
2. Desejável (DJ): É bom ter esta característica, mas sua ausência não é crítica para o software;
3. Irrelevante (IR): Não é uma preocupação do software suportar a característica[11];

Por exemplo, em uma situação de avaliação de características arquiteturais, podemos ter a avaliação representada pela tabela 4.4.

Aplicação	Expectativa
Interoperabilidade	IM
Segurança de acesso	DJ
Maturidade	IR
Tolerância a falhas	IR
Recuperabilidade	IR
Operacionalidade	IR
Comportamento (tempo)	IM
Comportamento (recursos)	IR
Modificabilidade	DJ
Testabilidade	IR
Adaptabilidade	IR

Tabela 4.4: Exemplificação da classificação das características arquiteturais

A classificação das características arquiteturais possibilita a geração de um vetor no espaço multidimensional destas características. A geração deste vetor é semelhante a que foi feita para os padrões arquiteturais, como pode ser vista abaixo:

Objetivo = (IM, DJ, IR, IR, IR, IR, IM, IR, DJ, IR, IR)

É a partir da classificação do conjunto de características arquiteturais que podemos atribuir os valores que representam a significância de cada característica. Atribuímos inicialmente os valores apresentados na tabela 4.5 para a representação e construção do vetor que traduz a especificação das características arquiteturais de

[11] A priori pode ser considerado que, em determinadas circunstâncias, quando o vetor objetivo possui muitas dimensões irrelevantes, um componente cujas dimensões sejam descritas como muito desfavoráveis seja apresentado como o mais indicado, visto o efeito destas dimensões sobre o cálculo da distância. Esse efeito, porém, tende a ser minimizado com o processo de normalização, devido ao ajuste sobre o efeito negativo das dimensões desfavoráveis. Entretanto, este é um aspecto que necessita um estudo mais elaborado.

qualidade que se busca alcançar. A atribuição destes pesos foi feita também buscando estabelecer distâncias proporcionais, estabelecendo que os itens “importante” e “irrelevante” estariam nos extremos da faixa de valores possíveis para a quantificação das classificações realizadas.

Papel da característica	Valores atribuídos
Importante	1
Desejável	0,5
Irrelevante	0

Tabela 4.5: Quantificação dos papéis atribuídos ao conjunto de características de referência

A representação quantitativa do vetor *Objetivo* correspondente à especificação das características arquiteturais exemplificada na tabela 4.4 tem a seguinte configuração:

Objetivo = (1, 0.5, 0, 0, 0, 0, 1, 0, 0.5, 0, 0)

A normalização e o cálculo da distância entre vetores seguem as funções descritas no começo desta seção e sua realização pode ser melhor visualizada quando utilizamos os exemplos dos vetores correspondentes ao padrão arquitetural “Camadas” e “Pipes & Filters”, denominados respectivamente *C* e *P*, e a especificação das características arquiteturais previamente estabelecida. A simulação escolhida é a neutra.

O cálculo da norma dos vetores *C* e *P* é dado pelas seguintes equações:

$$\|C\| = \sqrt{0.7^2 + 1^2 + 0.5^2 + 0.5^2 + 0.5^2 + 0.7^2 + 0.5^2 + 0.3^2 + 0.7^2 + 1^2 + 1^2} = 2.36$$

$$\|P\| = \sqrt{0.5^2 + 1^2 + 0.3^2 + 0.3^2 + 0^2 + 0.5^2 + 0.7^2 + 0.3 \ + 0.5^2 + 0.3^2 + 0^2} = 1.61$$

Com o cálculo dos valores das normas dos vetores correspondentes aos padrões “Camadas” e “Pipes & Filters”, podemos normalizá-los, para que estes passem a ser vetores unitários. Os novos vetores *C’* e *P’* são os resultados desta normalização e seus valores são os seguintes:

C’ = (0.30, 0.42, 0.21, 0.21, 0.21, 0.30, 0.21, 0.13, 0.30, 0.42, 0.42); e

P' = (0.31, 0.62, 0.19, 0.19, 0.00, 0.31, 0.43, 0.19, 0.31, 0.19, 0.00)

O vetor correspondente à especificação das características arquiteturais passa pelo mesmo processo de normalização e é representado pelo vetor denominado *O’*, já normalizado, da seguinte maneira:

O' = (0.63, 0.32, 0.00, 0.00, 0.00, 0.00, 0.63, 0.00, 0.32, 0.00, 0.00)

O cálculo das distâncias entre cada vetor correspondente a um padrão arquitetural e a especificação das características arquiteturais permite que ao fim deste processo tenhamos uma classificação dos resultados, sendo possível a criação de uma ordenação que informa quais os padrões arquiteturais que estão mais ou menos próximos da especificação desejada. Esta ordenação sugere a possibilidade de aplicação dos padrões no contexto desejado, orientando o processo de escolha e aplicação de determinada solução arquitetural.

Em nosso exemplo, apresentamos na tabela 4.6 os valores das distâncias vetoriais entre os padrões arquiteturais "Camadas" e "Pipes & Filters" e a especificação das características arquiteturais de referência.

Padrão Arquitetural	Distância
Camadas	0.95
Pipes & Filters	0.69

Tabela 4.6: Quantificação das distâncias entre vetores

De acordo com os resultados apresentados, podemos inferir que, para o contexto especificado pelo conjunto de características arquiteturais de qualidade de referência, o padrão "Pipes & Filters" é mais indicado do que o padrão "Camadas", indicando que o nível de esforço para se desenvolver uma solução arquitetural que privilegie nos graus especificados para as características arquiteturais deve ser menor quando nos baseamos no padrão "Pipes & Filters" ao invés do padrão "Camadas".

4.5 A Ferramenta de Indicação de Padrões Arquiteturais

A iniciativa de seleção de padrões arquiteturais materializou-se com a construção de uma ferramenta denominada Mentor[12]. Esta ferramenta está inserida no contexto da infra-estrutura do Odyssey e foi construída para apoiar as fases de criação e instanciação de arquiteturas de referência, especificamente no apoio a seleção dos padrões arquiteturais OO. A arquitetura central desta ferramenta segue o padrão arquitetural definido para o Odyssey (padrão arquitetural OO Model-View-Controller). O padrão MVC favoreceu a operacionalidade da ferramenta bem como facilitou o trabalho de modificação e teste dos seus principais componentes, principalmente os

[12] Mentor é o grande amigo de Ulisses, na obra Odisseía de Homero (ROCHA, 2000).

componentes de interface e controle de eventos dos usuários. A linguagem de programação utilizada foi Java.

A ferramenta pode ser dividida em dois principais grupos funcionais. O primeiro grupo se refere à configuração dos pesos definidos pela regra de seleção e o suporte ao gerenciamento dos resultados obtidos pela análise dos padrões arquiteturais. O principal ator desta parte da ferramenta é o especialista em projeto arquitetural. Neste ponto, é importante ressaltar que, mesmo inserida no contexto da infra-estrutura do Odyssey, este grupo funcional não exige que toda a infra-estrutura esteja em funcionamento para que as configurações e as avaliações sejam realizadas.

O segundo grupo funcional está relacionado às funções de especificação das características arquiteturais almejadas e ao apoio a seleção de soluções arquiteturais que possam melhor suportá-las. O principal ator destas funcionalidades da ferramenta é o projetista de software que irá especificar a arquitetura de domínio ou de uma aplicação.

A figura 4.2 apresenta o modelo global das interações suportadas através dos seus principais casos de uso.

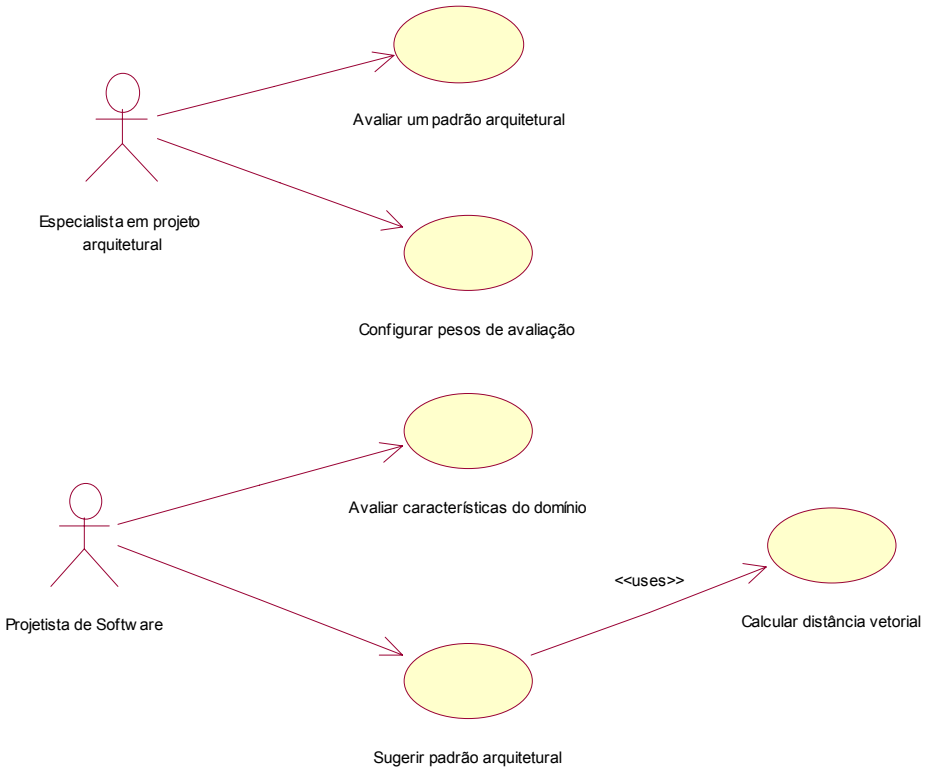


Figura 4.2: Diagrama global de casos de uso da ferramenta Mentor

Além de buscarmos a implementação dos casos de uso identificados, procuramos também organizar o relacionamento de classe para que fosse possível a implementação do suporte a instanciação dos padrões no contexto do projeto do Odyssey, de forma paralela e independente, visto que esta atividade está fora do escopo de nosso trabalho.

A figura 4.3 exibe o modelo de classes da ferramenta, utilizando a notação UML (BOOCH *et al.*, 1998). Apresentamos as classes que em nossa arquitetura possuem o esteriótipo de "Modelo", porque a semântica principal da aplicação está localizada sobre estas classes.

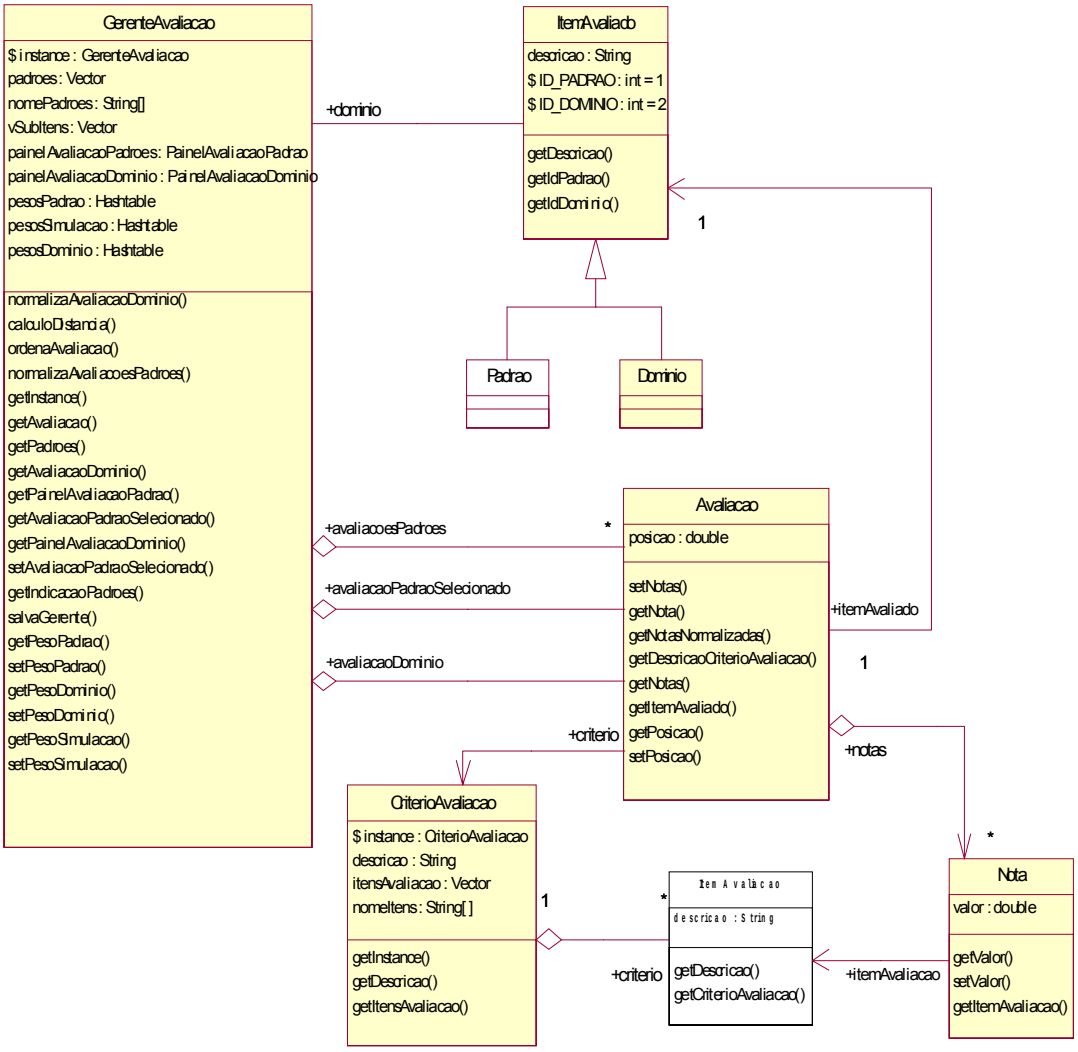


Figura 4.3: Modelo de classes da ferramenta Mentor

4.5.1 A Interface Gráfica da Ferramenta

A interface gráfica da ferramenta Mentor é ilustrada através de cenários de utilização das suas principais janelas. Os dados apresentados na janela de avaliação dos padrões seguem os valores que foram mostrados na seção 4.3 e a especificação das características arquiteturais é a mesma que foi utilizada na exemplificação da regra de seleção na seção 4.4.

A figura 4.4 apresenta a janela de configuração dos pesos atribuídos aos conceitos subjetivos envolvidos na abordagem (nível de esforço, de relevância e de simulação) e que representam a quantificação dos valores utilizados.

Os pesos utilizados pela ferramenta são os mesmos descritos na seção 4.4 e o principal objetivo desta interface é possibilitar o ajuste desses pesos, caso se verifique a necessidade de ajuste dos limites definidos. O botão de salvamento das alterações somente é habilitado quando há qualquer modificação dos componentes de ajuste dos pesos, retornando a situação de desabilitado após o último salvamento. O botão de saída fecha a janela de configuração, apresentando um diálogo com o usuário caso se verifique a solicitação de saída do contexto de configuração sem que a última mudança dos dados tenha sido salva.

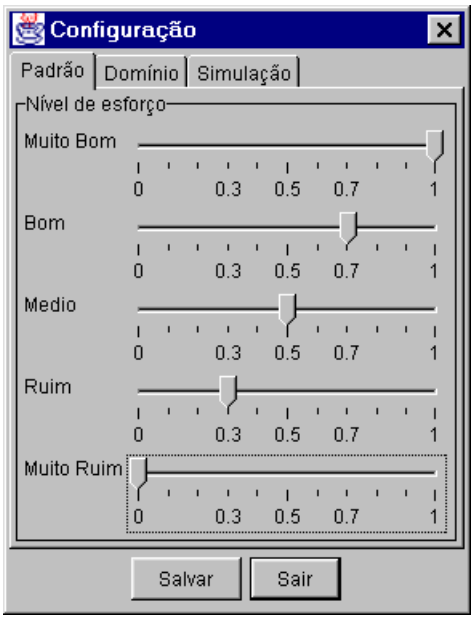


Figura 4.4: Janela de configuração dos pesos utilizados para a indicação dos padrões

A figura 4.5 apresenta a janela de edição das avaliações associadas aos padrões arquiteturais utilizados na abordagem. Em caráter de exemplificação, a janela da figura 4.5 mostra a avaliação do padrão arquitetural “Camadas” apresentada na tabela 4.1. Os símbolos utilizados para representação do grau de esforço apresentam uma ajuda quanto ao seu significado quando se passa o cursor do “mouse” sobre a

área que cada um se localiza. Cada item do conjunto de características utilizadas também apresenta a mesma facilidade, tendo as suas definições sido extraídas da referência apropriada (ISO9126, 1992).

No topo da janela temos o componente visual de seleção de um entre os possíveis padrões arquiteturais sob análise. A ferramenta garante que a avaliação de uma característica dentro do espectro de valores apresentado seja exclusiva, de maneira que não seja possível a seleção de mais de um valor de nível de esforço para um único atributo de qualidade. O evento de seleção de um outro padrão implica da mudança dos valores atribuídos à avaliação corrente, trazendo ao novo contexto a avaliação pré-existente do novo padrão selecionado, ou os valores de uma avaliação que ainda não se realizou. Tal avaliação é caracterizada quando todos os valores estão definidos como “desconhecidos”. Os botões de salvamento e de saída trabalham da mesma forma descrita na janela de configuração.

Figura 4.5: Janela de análise dos padrões arquiteturais

As funções apresentadas pelas janelas de configuração e de análise representam a implementação do grupo funcional responsável em disponibilizar as funções utilizadas pelo ator “Especialista em projeto arquitetural”.

A figura 4.6 apresenta a janela de representação da especificação das características arquiteturais. Nesta janela o ator “Projetista de Software” poderá realizar a entrada dos graus atribuídos ao nível de relevância de cada atributo de qualidade do conjunto utilizado no contexto de desenvolvimento de projeto arquitetural que este está inserido. Para efeito de ilustração, a janela da figura 4.6 mostra a especificação do conjunto de características arquiteturais ilustrado no exemplo do item 4.4, no contexto do domínio de “sistemas de informação”. Esta atribuição de níveis de relevância faz parte do histórico de projeto da arquitetura e contribui para indicar os fatores de natureza qualitativa que influenciaram a escolha de determinado padrão arquitetural.

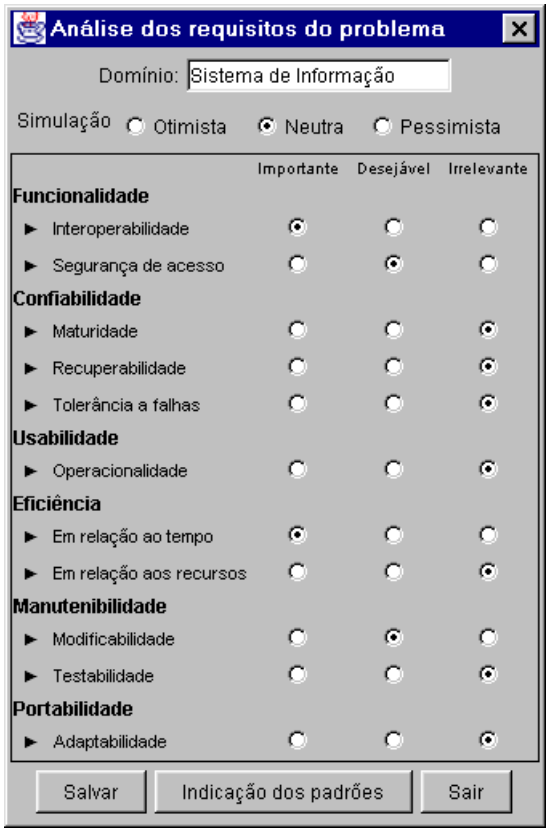


Figura 4.6: Janela de representação e avaliação dos atributos de referência almejados para a arquitetura de domínio

A solicitação de indicação dos padrões arquiteturais é feita através da seleção da abordagem de simulação (isto é, pessimista, neutra e otimista) e do envio do evento de indicação, representado pelo botão “Indicação dos Padrões”. É importante que a abordagem de simulação seja definida para que o problema do tratamento das relações desconhecidas seja realizado, conforme descrito na seção 4.4. Caso não

ocorra seleção de uma abordagem, a ferramenta utiliza por *default* a abordagem neutra. Os botões de salvamento e de saída trabalham da mesma forma descrita na janela de configuração (figura 4.4).

A solicitação de indicação de padrões dispara o mecanismo de cálculo e construção da janela de indicação apresentada na figura 4.7. O principal objetivo da janela é apresentar o *ranking* dos padrões arquiteturais que potencialmente poderiam ser utilizados durante a fase de projeto arquitetural do domínio. Esta lista de padrões implicitamente representa a ordem crescente das distâncias euclidianas calculadas, sendo o primeiro padrão apresentado o que menor distância obteve da especificação dos requisitos de referência registrados.

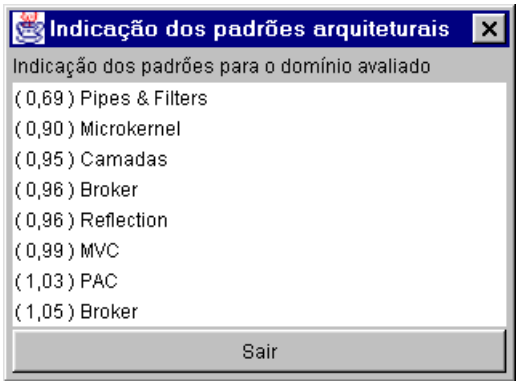


Figura 4.7: Janela de indicação dos padrões arquiteturais.

Para efeito de ilustração a janela de indicação apresenta o padrão “Pipes & Filters” como o mais indicado ao contexto que foi representado pela especificação das características arquiteturais. No contexto desta janela, o projetista de software também tem acesso aos valores das distâncias que possibilitaram a construção deste *ranking* (representado especificamente pelos valores entre parênteses que antecedem os nomes dos padrões). A seleção de um padrão arquitetural deverá disparar o uso da ferramenta de instanciação de padrões arquiteturais, também necessária ao contexto do Odyssey. Contudo, como não há ainda o suporte a instanciação dos padrões arquiteturais no Odyssey, a utilização desta ferramenta não pode ser realizada neste momento.

4.6 Conclusão

A utilização da distância entre vetores mostra-se uma forma adequada em situações em que se precisa comparar entidades representadas por suas características (MONETA *et al.*, 1990, ASADA *et al.*, 1992). Sua utilização mostra-se frágil na situação em que em teoria nenhum padrão atenderia bem a especificação de projeto (como, por exemplo, a necessidade de privilegiar todos os itens do conjunto de características arquiteturais). Na prática, o cálculo da distância entre vetores irá apresentar os mais próximos desta especificação alvo. Contudo, é difícil estabelecer um valor limite para a distância, onde este valor representa a situação em que não faz mais sentido indicar padrões arquiteturais que apresentam uma distância maior do que este valor de exclusão. Um tratamento mais refinado se faz necessário neste ponto para que as indicações sejam realmente confiáveis.

KONTIO (1995) afirma, ainda, que a utilização da abordagem vetorial para a avaliação de distâncias conceituais só é válida quando o número de critérios avaliados é baixo e todos os critérios envolvidos são independentes uns dos outros. Contudo, para o contexto apresentado neste capítulo, acreditamos que estes requisitos são respeitados e a utilização desta abordagem se mantém viável.

Um dos pontos que podem ampliar a compreensão sobre a distribuição dos padrões arquiteturais no espaço vetorial que os contém é o que avalia a distância vetorial existente entre os padrões, na medida que representa regiões do espaço onde os padrões estariam mais ou menos próximos uns dos outros. A identificação destas regiões possibilitaria o estudo da afinidade entre os padrões, mas não consideramos esta análise neste trabalho.

A avaliação dos padrões arquiteturais permitiu que identificássemos, para o conjunto de características arquiteturais de qualidade particular, aspectos relacionados ao aparente nível de esforço despendido nos projetos arquiteturais. Contudo, a eficácia da seleção dos padrões arquiteturais depende muito da qualidade deste levantamento. Percebemos que o conhecimento descrito na literatura é importante, mas não é suficiente para sustentar de forma confiável a abordagem de seleção. Vemos, portanto, a necessidade de estabelecermos uma abordagem que possibilite o ajuste, extensão e validação deste conhecimento.

Capítulo 5: O Processo de Avaliação Contínuo dos Padrões Arquiteturais

5.1 Introdução

A abordagem de seleção dos padrões arquiteturais está baseada em dois aspectos principais. O primeiro deles refere-se a utilização de um mecanismo que permite avaliar a distância conceitual entre os padrões arquiteturais e determinada especificação de características arquiteturais de qualidade que serve como referência. Este cálculo age sobre uma base de avaliações e a qualidade dos resultados apresentados depende, substancialmente, do processo de análise utilizado para obter este conhecimento.

Cientes de que a organização da base de avaliações é o segundo aspecto importante da abordagem de seleção e que a utilização de apenas algumas poucas referências encontradas na literatura pode implicar em uma baixa aplicabilidade dos resultados, buscamos formalizar a base de um processo que possa ser aplicado continuamente e que pretende apoiar a evolução das avaliações associadas aos padrões arquiteturais. Para viabilizar este processo, decidimos elaborar um estudo que pudesse, ao seu final, contribuir para a avaliação contínua e realimentação das informações nesta base.

O objetivo deste capítulo é discutir a natureza e a viabilidade deste estudo e de que forma os resultados obtidos podem influenciar a evolução das avaliações dos padrões arquiteturais previamente existente. O capítulo está organizado da seguinte forma: no item 5.2, descrevemos a organização dada ao estudo e os resultados que levaram a mudança do planejamento de sua aplicação; no item 5.3, mostramos o estudo sob o novo enfoque, apresentando a análise dos novos resultados obtidos; no item 5.4, realizamos a evolução da base de avaliação previamente existente, baseando-nos nos resultados obtidos com a realização do estudo; o item 5.5 finaliza o capítulo.

5.2 Estudo 1: Avaliação Baseada no Esforço em Projeto

O principal objetivo deste estudo foi a realização do levantamento, junto a desenvolvedores de software, da relação identificada entre arquiteturas de software representadas através de padrões arquiteturais OO e as características arquiteturais de qualidade definidas para o escopo deste trabalho (vide seção 4.2).

5.2.1 Planejamento

Este estudo baseou-se na idéia de que desenvolvedores de software, com experiência em projeto arquitetural e detalhado, possuem conhecimento suficiente para contribuir com algumas indicações sobre a utilização dos padrões arquiteturais. Contamos com a participação voluntária dos desenvolvedores e o estudo foi planejado para avaliar a aparente relação de esforço em projeto para que determinada característica de qualidade seja privilegiada, quando o projeto arquitetural baseia-se em um padrão específico.

Para melhorar a medição dos resultados, o estudo foi dividido em duas partes:

1. Caracterização do perfil do desenvolvedor: O perfil do desenvolvedor foi utilizado para complementar a análise dos resultados, principalmente na situação em que há conflito de respostas. Esta caracterização procurou indicar também se existiu qualquer tipo de influência nas respostas dadas, principalmente em relação à natureza dos sistemas previamente desenvolvidos e pela diversidade de domínios de aplicações conhecidos;
2. Avaliação dos padrões arquiteturais: Manteve o objetivo estabelecido para a representação do conhecimento relacionado à utilização dos padrões arquiteturais.

5.2.2 Operação

Este estudo se deu através do preenchimento de um questionário[13]. Demos liberdade ao desenvolvedor para realizá-lo a sós ou de forma supervisionada, como uma entrevista. Contamos com a participação voluntária de 10 pessoas, sendo este um grupo formado, em sua maioria, por alunos do curso de graduação e pós-graduação em informática da UFRJ que prestavam ou já haviam prestado uma disciplina relacionada ao desenvolvimento de software com ênfase em projeto. Tivemos a preocupação em formar um grupo com um nível de experiência em desenvolvimento condizente com os objetivos que buscamos alcançar.

5.2.3 Análise dos Dados

Apenas dois especialistas responderam ao questionário. Os demais participantes deram um retorno verbal sobre as dificuldades encontradas durante o seu preenchimento.

[13] O modelo do questionário utilizado no estudo pode ser visto no anexo I desta tese.

5.2.4 Interpretação dos Resultados

O questionário mostrou-se muito difícil de ser respondido. A maioria dos participantes reclamou das dificuldades para avaliar a relação entre os padrões arquiteturais e as características arquiteturais de qualidade utilizadas. Percebemos que a organização dada para a avaliação pudesse estar distante da natureza dos problemas que os desenvolvedores estão acostumados a lidar.

5.2.5 Conclusão

Notamos que, para a maioria dos participantes, a relação existente entre as características arquiteturais de qualidade e o projeto arquitetural de um software não é evidente. Existem algumas hipóteses para esta questão. O resultado obtido pode indicar que as pessoas desconhecem a aplicação dos padrões arquiteturais visando especificamente a obtenção de características arquiteturais específicas. Acreditamos, porém, que a orientação dada ao planejamento do estudo dificultou o entendimento dos objetivos. Procuramos, então, realizar uma reformulação sobre a forma de avaliação dos padrões arquiteturais.

5.3 Estudo 2: Avaliação Baseada em Cenários de Projeto

Preservamos o objetivo de realizar o levantamento da experiência em projeto junto aos desenvolvedores de software. Porém, o resultado obtido com o primeiro estudo alertou-nos sobre a dificuldade de utilizarmos elementos de avaliação que talvez estejam distantes da prática do desenvolvimento de software.

5.3.1 Planejamento

O estudo, baseado agora em cenários de projeto, manteve a idéia de que desenvolvedores de software com experiência em projeto arquitetural e detalhado podem contribuir com um conhecimento prático sobre a utilização de padrões arquiteturais. Porém, a avaliação deste conhecimento foi realizada através de um exemplo que procurou estar mais próximo de uma situação real de desenvolvimento.

Organizamos, então, o experimento na forma de um exercício[14], onde um sistema hipotético, com requisitos arquiteturais de qualidade específicos, deveria ser analisado quanto à utilização de padrões arquiteturais. Mantivemos a caracterização

[14] O modelo do exercício aplicado no estudo pode ser visto no anexo II desta tese

necessária do entrevistado para a identificação do seu perfil com o objetivo de melhorar a avaliação dos resultados.

5.3.2 Operação

A organização dos cenários utilizados para dar sustentação ao exercício está representada pela tabela 5.1. Preocupamo-nos com o equilíbrio na distribuição das características arquiteturais de qualidade para que pudéssemos extrair algumas conclusões sobre os resultados obtidos, principalmente porque consideramos existir um certo grau de dificuldade para se lidar com as características envolvidas em cada cenário.

Cenários de Projeto / Características Arquiteturais	CENÁRIO 1	CENÁRIO 2	CENÁRIO 3	CENÁRIO 4	CENÁRIO 5
Interoperabilidade		X			
Segurança de acesso	X				
Maturidade				X	
Tolerância a falhas			X		
Recuperabilidade	X				
Operacionalidade			X		
Comportamento em relação ao tempo		X			
Comportamento em relação aos recursos					X
Modificabilidade				X	
Testabilidade				X	
Adaptabilidade					X

Tabela 5.1: Distribuição das características arquiteturais entre cenários de projeto

Escolhemos um grupo inicial de 10 participantes e pedimos a cada um o esforço voluntário para a submissão da pesquisa já que este deveria ser realizado como um exercício. Este grupo foi preliminarmente escolhido junto aos alunos de mestrado e doutorado da COPPE/UFRJ pela característica técnica de cada um e pela proximidade física do grupo, para que situações de dúvidas ou dificuldades no preenchimento das respostas pudessem ser atendidas o mais rápido possível. Para que o resultado fosse mais efetivo, procuramos garantir que todos os participantes já haviam tido contato, pelo menos teórico, com a aplicação de padrões arquiteturais.

5.3.3 Análise dos dados

O estudo baseado em cenários de projeto apresentou um resultado mais expressivo de respostas (50 % dos entrevistados retornaram o exercício). Acreditamos que este número foi suficiente para que pudéssemos realizar algumas considerações sobre os resultados obtidos. Evitamos a nominação dos especialistas neste estudo e iremos referenciá-los, a partir de agora, pelas letras A, B, C, D e E.

O quadro da atividade profissional de cada especialista pode ser vista na tabela 5.2. Todos os desenvolvedores são graduados em informática, têm alguma experiência em desenvolvimento de software na indústria e estão envolvidos em atividades acadêmicas. Em relação ao conhecimento relacionado à utilização dos padrões arquiteturais, este grupo de desenvolvedores mostrou que o assunto não era desconhecido, embora sem conhecerem profundamente o assunto.

		Tempo (em anos) de atuação em:		Principal atividade	
Participante	**Formação**	**Empresa**	**Universidade**	**Empresa**	**Universidade**
A	Doutor em ES	5	7	Analista	Professor
B	Mestre em ES	9	4	Programador	Professor
C	Mestre em ES	12	5	Consultor	Professor
D	Graduação	2	5	Programador	Pesquisador
E	Graduação	1	4	Programador	Pesquisador

ES – Engenharia de Software

Tabela 5.2: Perfil dos participantes

A principal natureza de sistemas que os especialistas têm experiência é o de informação (todos os especialistas). Os domínios de aplicações em que este grupo trabalhou compreendem principalmente as ferramentas de desenvolvimento de software (todos os especialistas), contando ainda com experiências pontuais no domínio agropecuário (especialista A), educacional (especialistas A, C e D), administrativo (especialista C), software básico (especialista C), teleinformática / telecomunicações (especialistas C e D) e de transportes (especialistas A e C).

Solicitamos aos participantes uma auto-avaliação de suas experiências nas principais etapas relacionadas ao desenvolvimento de software. Esta avaliação pôde ser feita a partir da classificação em 5 níveis, a saber:

- Nenhuma experiência (NE);
- Estudei em classe ou em livros (CL);
- Pratiquei em um projeto em classe (PC);
- Utilizei em um projeto na indústria (PI); e
- Utilizei em mais de um projeto na indústria (MP).

O principal objetivo desta avaliação é contornar os possíveis conflitos entre as respostas dadas em cada cenário de projeto. A tabela 5.3 sintetiza a avaliação de cada participante.

Para a execução do exercício, solicitamos ao especialista que, para cada cenário de projeto, indicasse, e justificasse, os padrões arquiteturais mais e menos aplicáveis ao contexto descrito, em um número de pelo menos dois padrões para cada situação. O especialista deveria ainda descrever algumas observações em relação aos motivos que o levaram a excluir os restantes dos padrões das situações anteriores. Com isso, buscamos ter uma visão da aplicação destes padrões, em uma relação de aplicabilidade parcial no cenário (ou seja, quando um determinado padrão pode ser utilizado, apesar de não ser considerado a solução ideal).

Experiência em Desenvolvimento de Software	A	B	C	D	E
Elicitação de requisitos	MP	MP	MP	NE	PC
Projeto de sistemas	PI	MP	MP	PC	PC
Criação de projetos OO	PI	MP	MP	PI	PI
Modificação de projetos p/ manutenção	NE	MP	MP	NE	PI
Codificação, baseado em projeto OO	PI	MP	MP	PI	MP
Manutenção de código	PI	MP	MP	NE	PI
Projeto utilizando algum padrão arquitetural	NE	PI	MP	MP	PI

Tabela 5.3: Avaliação da experiência em desenvolvimento

As sínteses dos resultados são mostradas a seguir em forma de tabelas, cada qual relacionada a um cenário particular. Utilizamos os símbolos +, - e +/- para representar, respectivamente, os padrões arquiteturais mais aplicáveis, os menos adequados e os não considerados nas situações anteriores. A primeira coluna representa os padrões arquiteturais avaliados, e as colunas designadas pelas letras A, B, C, D e E representam os especialistas.

Consideramos igualmente que devíamos submeter Mentor aos cenários propostos no estudo, para que os resultados obtidos com o seu uso venham a ser comparados às respostas dadas pelos especialistas. A indicação dos padrões arquiteturais sugeridos pela ferramenta pode ser vista na coluna representada pela letra M em cada uma das tabelas correspondentes aos cenários de projeto.

É importante lembrar que Mentor apresenta as indicações na ordem decrescente das possibilidades, variando do mais ao menos indicado à solução do problema. Para a apresentação desta ordenação, indicamos a posição que cada padrão ocupa no “ranking” de sugestões (assumindo as posições de 1° a 8°). Completamos o “ranking” dos padrões com os valores obtidos durante o cálculo das

distâncias entre os padrões arquiteturais e as características arquiteturais referenciadas no cenário de projeto, para que tenhamos uma melhor compreensão das distâncias que os separam. Como a realização da indicação dos padrões depende da definição de uma abordagem de simulação, escolhemos a abordagem “neutra” para a obtenção dos resultados aqui utilizados porque acreditamos ser esta a representação média das possibilidades de avaliação dos padrões na maioria das situações de utilização desta ferramenta.

Informamos, ainda, que nem todos os especialistas responderam aos questionários por completo, sendo principalmente descartada as respostas para os padrões não considerados nas situações de mais e menos aplicáveis. Tivemos também a situação em que o especialista apenas indicou os padrões de maior possibilidade de uso.

5.3.3.1 Cenário 1: Segurança de Acesso e Recuperabilidade

A tabela 5.4 apresenta os resultados obtidos para o primeiro cenário de projeto. Como podemos ver, os especialistas são unânimes em afirmar que o padrão “Camadas” é o mais adequado, seguido pelo padrão “Blackboard”. Como menos indicado temos o padrão “Reflection”.

Especialistas / Padrões	A	B	C	D	E	M
Camadas	**+**	**+**	**+**	**+**	**+**	**1° (1,05)**
Pipes&Filters	**+/-**	**-**				**2° (1,06)**
Blackboard		**-**	**+**	**+**	**+**	**8° (1,16)**
Broker	**+/-**	**-**	**+**		**-**	**3° (1,07)**
MVC	**+**			**+/-**		**4° (1,09)**
PAC	**+**			**+/-**		**7° (1,14)**
Microkernel	**-**					**6° (1,13)**
Reflection	**-**			**-**	**-**	**5° (1,10)**

Tabela 5.4: Resultado do primeiro cenário de projeto.

Concordamos com os especialistas quando indicam os padrões "Camadas" e "Reflection" como o mais e o menos indicado ao contexto do cenário, respectivamente, mas discordamos do resultado apresentado pela maioria ao indicar o padrão "Blackboard" como um dos mais apropriados. De acordo com a bibliografia pesquisada (BUSCHMANN *et al.*, 1996, SHAW e GARLAN, 1996, BASS *et al.*, 1998, BOSCH, 2000), a característica da recuperabilidade é muito desfavorável quando este padrão é

utilizado. Desta forma, consideramos a resposta dada pelo especialista B como a mais afinada com as referências obtidas sobre este padrão.

A análise das justificativas referentes ao padrão “Pipes & Filters” nos mostrou que não há indícios que a característica de segurança de acesso é valorizado por esse padrão. Em relação a recuperabilidade do software, esta ficaria muito prejudicada porque eventuais falhas em um de seus componentes repercutem sobre a integridade do processamento, sendo muito difícil recuperar a aplicação para o estado anterior à falha ocorrida. Desta forma, concordamos com a análise do especialista B quando considera o padrão “Pipes & Filters” um dos menos indicados para este cenário.

Quanto ao padrão “Broker”, o especialista A considerou-o uma infra-estrutura de suporte a aplicação, como por exemplo CORBA e DCOM (SZYPERSKI, 1998), visão diferente do que é dada ao padrão por Buschmann (BUSCHMANN *et al.*, 1996), onde o componente que trata da distribuição da funcionalidade é construído e incorporado ao projeto, sendo um de seus principais elementos. Por isso, neste cenário, consideramos que as respostas dadas pelos especialistas A, B e E são mais apropriadas porque descrevem que eventuais falhas no componente responsável pela distribuição afetam diretamente a todas as aplicações que lhes são dependentes, dificultando assim a sua utilização no contexto do cenário.

Mentor também apresenta o padrão Camadas como o mais apropriado ao contexto do cenário mas diverge dos especialistas, principalmente ao colocar o padrão “Pipes & Filters” e “Broker” como os mais indicados e o padrão “Reflection” como um intermediário.

Identificamos que a diferença de resultados apresentada para este cenário, e de forma semelhante nos cenários subseqüentes, pode ser causada ou pela utilização por parte da ferramenta de avaliações imprecisas, no sentido de não expressar corretamente a relação de utilização dos padrões em relação às características envolvidas no cenário, ou pela falta de certeza ou maturidade em relação às respostas dadas pelos especialistas. Desta forma, somente com a avaliação qualitativa das respostas dadas pelos especialistas poderemos fundamentar qualquer aceitação e mudança nas avaliações que dão sustentação aos resultados apresentados por Mentor.

Aqui a análise dos resultados obtidos neste cenário nos mostrou que os resultados apresentados pelos especialistas são suficientemente válidos para serem usados no ajustes da avaliação dos padrões "Pipes & Filters", "Broker" e "Reflection". Em relação aos demais resultados, acreditamos que não há subsídios suficientes que justifiquem quaisquer modificações nas avaliações.

5.3.3.2 Cenário 2: Interoperabilidade e Comportamento em Relação ao Tempo

A tabela 5.5 apresenta os resultados obtidos para o segundo cenário de projeto. Aqui os especialistas são unânimes em afirmar que o padrão “Broker” é o mais adequado. Como menos indicado temos o padrão “Camadas”. Os padrões “Pipes & Filters”, “Blackboard” e “Microkernel” precisam ter os seus resultados analisados.

Segundo os especialistas A e D o padrão “Pipes & Filters” privilegia a comunicação entre processos e o paralelismo de execução dos componentes, o que o tornaria adequado ao cenário. Porém, o especialista E considerou este padrão como desfavorável por não considerar o contexto do sistema em questão um problema de natureza seqüencial de processamento. Como o objetivo desta avaliação busca transcender o contexto de sistemas pontuais, mesmo que reconheçamos a importância de suas influências para a análise dos resultados, consideramos mais apropriadas as justificativas que enquadram este padrão como um dos favoráveis ao cenário.

Especialistas / Padrões	A	B	C	D	E	M
Camadas	-			-	-	**2° (1,13)**
Pipes&Filters	+			+	-	**1° (0,97)**
Blackboard	-	+		+	-	**8° (1,26)**
Broker	+	+	+	+	+	**7° (1,18)**
MVC		-				**5° (1,15)**
PAC		-			+	**4° (1,14)**
Microkernel			+	+	-	**2° (1,13)**
Reflection				+/-		**6° (1,17)**

Tabela 5.5: Resultado do segundo cenário de projeto.

O padrão “Blackboard” inspira maiores cuidados, porque alguns especialistas o vêem como um repositório central de dados, na linha de um SGBD, onde seria possível a interoperabilidade das aplicações através deste repositório e a otimização das consultas através do armazenamento de procedimentos no servidor de dados. Esta visão, no entanto, está mais voltada para uma variação do padrão “Blackboard”, onde não há o problema da dificuldade de se estabelecer as estratégias de controle da execução do software. Desta forma, as respostas mais apropriadas são as que consideram o padrão inadequado, principalmente em função do comportamento do software em relação ao tempo, muito prejudicado por questões específicas deste padrão.

E no caso do padrão “Microkernel”, sua avaliação foi considerada positiva porque este pode ser aplicado de modo a permitir a adaptação do sistema a diferentes aplicações, como definido no cenário, com a possibilidade de otimização da execução das funcionalidades do kernel.

Mentor apresentou resultados diferentes dos especialistas, principalmente ao não indicar o padrão “Broker” como o mais apropriado. Isto ocorre porque a base de avaliações utilizada pela ferramenta considera que o aspecto de comportamento em relação ao tempo é muito prejudicado em arquiteturas que utilizam mecanismos de distribuição. Nenhum dos especialistas considerou este aspecto em suas respostas, talvez porque o aspecto da interoperabilidade tenha encoberto a característica de desempenho em relação ao tempo. Deste modo, acreditamos que as respostas obtidas para os padrões “Broker”, “MVC”, “PAC” e “Reflection” não são suficientes para justificar a adaptação dos resultados apresentados pela ferramenta. Em relação a avaliação dada ao padrão "Camadas", os especialistas apresentam a interoperabilidade como um dos seus aspectos desfavoráveis. No entanto, acreditamos que neste ponto devemos aguardar um maior número de resultados para que seja possível uma melhor interpretação das respostas, visto que a bibliografia consultada para a formalização da base de avaliações vem de encontro às respostas obtidas.

5.3.3.3 Cenário 3: Tolerância a Falhas e Operacionalidade;

A tabela 5.6 apresenta os resultados para o terceiro cenário de projeto. Aqui os especialistas, em sua maioria, em afirmar que os padrões “MVC” e “PAC” são os mais adequados ao cenário. Como menos indicado temos os padrões “Reflection” e “Pipes & Filters”. O restante dos padrões precisam ter os seus resultados analisados.

O padrão “Camadas” apresenta, na visão do especialista D, a possibilidade de inclusão de uma camada de tratamento das falhas que venham ocorrer na aplicação. Por outro lado, o especialista E lembra que a utilização deste padrão pode gerar uma certa dificuldade para a obtenção dos objetivos do cenário, por não possuir uma estrutura especializada que trate as várias visões de dados, comuns aos sistemas que privilegiam a característica de operacionalidade. No entanto, ainda assim consideramos este padrão como um dos indicados.

Padrões \ Especialistas	A	B	C	D	E	M
Camadas				+	+/-	**4° (1,13)**
Pipes&Filters				-	-	**5° (1,14)**
Blackboard		-		+		**2° (0,89)**
Broker		-	+	+/-	+	**6° (1,15)**
MVC	+		+	+	+	**3° (1,00)**
PAC	+	+		+		**1° (0,83)**
Microkernel	-			+/-	+/-	**8° (1,19)**
Reflection	-			-	-	**7° (1,17)**

Tabela 5.6: Resultado do terceiro cenário de projeto.

O padrão “Blackboard” é descrito pelo especialista B como um modelo estrutural onde os resultados decorrentes do processamento da aplicação são apenas hipóteses, condição esta que praticamente invalida os objetivos específicos do cenário, no momento que possibilita a indicação de resultados que não retratam, por exemplo, a política correta de descontos. Desta forma, concordamos que sua utilização neste cenário torna-se inapropriada.

Os especialistas têm visões antagônicas sobre a aplicabilidade do padrão “Broker”. Visto como uma infra-estrutura de suporte à aplicação, a tolerância a falhas é facilmente realizada porque as aplicações que as utilizam se beneficiam do suporte à esta característica arquitetural, comum na maioria das infra-estruturas deste tipo, representando um esforço de desenvolvimento praticamente nulo. No entanto, considerando-o como um componente do software, este esforço de tolerância à falhas do padrão fica mais difícil, exigindo a replicação do componente responsável pela distribuição da funcionalidade do software. Como estamos considerando apenas a segunda visão deste padrão, acreditamos que o padrão “Broker” deve ser um dos que está na faixa dos mais ou menos adequados.

Os especialistas mostraram certa dificuldade em apresentar suas justificativas em relação à aplicabilidade do padrão Microkernel. Acreditamos que isto seja, em parte, o reflexo do baixo número de evidências na literatura que apóiem uma melhor análise do padrão para este cenário.

Em relação aos padrões mais adequados, Mentor apresentou padrões que foram considerados pelos especialistas (“MVC” e “PAC”), mas diverge, principalmente, do resultado que consideramos apropriado ao indicar o padrão “Blackboard” como um dos mais adequados ao contexto do cenário. A indicação do padrão “Microkernel” reflete a escolha da abordagem de simulação e como não houve uma análise mais

segura deste padrão, optamos por manter a avaliação de grau “desconhecido” para as características do cenário.

5.3.3.4 Cenário 4: Maturidade, Modificabilidade e Testabilidade

A tabela 5.7 apresenta os resultados para o quarto cenário de projeto. Aqui os especialistas, em sua maioria, afirmam que o padrão “Microkernel” é o mais adequado, sendo igualmente possível a aplicação do padrão “MVC” no contexto do cenário analisado. Como menos indicado temos o padrão “Blackboard”. Todos os padrões restantes precisam ser analisados com mais atenção.

Na visão do especialista B, os padrões “Camadas” e “Pipes & Filters” sempre criam uma dependência funcional entre os componentes adjacentes, o que poderia dificultar em muito a modificabilidade e testabilidade do software. Seu ponto de vista é válido, mas, conforme foi lembrado pelos especialistas D e E, estes padrões privilegiam o aspecto da modularização do software, o que facilitaria as características arquiteturais descritas para este cenário. Neste sentido, concordamos com esta análise e igualmente acreditamos que estes padrões, quando utilizados, atendem bem aos objetivos do cenário de projeto.

Especialistas / Padrões	A	B	C	D	E	M
Camadas		-		+	+	**3° (0.96)**
Pipes&Filters		-		+		**6° (1,10)**
Blackboard	-			-	-	**7° (1,16)**
Broker			+	-	-	**4° (0,98)**
MVC		+		+	+/-	**4° (0,98)**
PAC				+		**8° (1,20)**
Microkernel	+	+	+	+		**1° (0,84)**
Reflection	+			-	+	**2° (0,85)**

Tabela 5.7: Resultado do quarto cenário de projeto.

No caso do padrão “Broker”, os especialistas D e E argumentam que não há ênfase na modularidade dos componentes do sistema, o que dificultaria o acompanhamento da troca de mensagens no ambiente distribuído. Com isso, concordamos que o padrão seria inapropriado, principalmente em situações que exigem a facilidade de teste da aplicação. A visão de infra-estrutura de aplicação dada pelo especialista C compromete a sua avaliação e por isso não foi considerada.

O padrão “Reflection” foi escolhido como favorável pela sua facilidade em suportar sistemas altamente modificáveis (especialistas A e E). Com a implementação

das regras de negócio no meta-modelo proposto pelo padrão, pode-se alterá-las sem que haja a necessidade de recompilar a aplicação. Esta visão é mais apropriada do que a dada pelo especialista D, quando o considera inapropriado por ser baixa a modularidade dos componentes do padrão.

Como há apenas uma referência à aplicabilidade do padrão “PAC” neste cenário de projeto, consideramos ser muito difícil sua análise para a elaboração de resultados mais expressivos.

Mentor está afim com a avaliação dos especialista quando indica os padrões “Microkernel” e “Reflection” como os mais apropriados e o padrão “Blackboard” como um dos menos favoráveis. Mas as indicações dos padrões “Pipes & Filters” e “Broker” vão de encontro à analise dos resultados, sendo necessário os seus respectivos ajustes. Em relação ao padrão MVC, os especialistas realçam a maturidade do software que o utiliza. A avaliação desta característica foi importante para que reconsiderássemos sua avaliação, haja visto que houve uma avaliação imprecisa do padrão neste ponto.

5.3.3.5 Cenário 5: Comportamento em Relação aos Recursos e Adaptabilidade

A tabela 5.8 apresenta os resultados do quinto e último cenário de projeto. Aqui os especialistas apresentam o padrão “Microkernel” como o mais adequado ao cenário, seguido pelos padrões “MVC” e “Reflection”. A aplicação do padrão “Blackboard” no contexto deste cenário é tida como desfavorável. Os resultados apresentados para os padrões “Camadas” e “Pipes & Filters” requerem uma análise mais detalhada.

Especialistas / Padrões	A	B	C	D	E	M
Camadas		-		+	+	**3° (1,10)**
Pipes&Filters		-		+		**8° (1,32)**
Blackboard	-			-	-	**7° (1,26)**
Broker			+	-	-	**6° (1,18)**
MVC		+		+	+/-	**2° (1,09)**
PAC				+		**5° (1,14)**
Microkernel	+	+	+	+		**1° (1,07)**
Reflection	+			-	+	**3° (1,10)**

Tabela 5.8: Resultado do quinto cenário de projeto.

O padrão “Camadas” é considerado pelos especialistas A, B e E como uma opção inapropriada para a adaptabilidade da aplicação. Porém, os especialistas C e D colocam justamente o contrário, justificando com a possibilidade de utilização de uma camada básica de sistema com interfaces genéricas que poderia ser substituída de forma transparente para as outras camadas da aplicação. Neste sentido, concordamos com estes especialistas e consideramos que realmente podemos aplicar o padrão nesta situação.

Já o padrão “Pipes & Filters”, em geral, apresenta os filtros no mesmo nível de abstração e sua adaptabilidade à outras plataformas fica muito difícil, conforme afirma o especialista B. Neste sentido, temos a mesma opinião sobre a aplicabilidade deste padrão neste cenário. Porém, em relação ao comportamento em relação aos recursos, não tivemos nenhuma análise que permitisse a avaliação do padrão em relação a esta característica.

Mentor em grande parte está ajustada aos resultados obtidos para este cenário. Isto pode representar um estado atual da avaliação dos padrões que potencialmente indica resultados afinados com a experiência em desenvolvimento.

5.3.4 Conclusão

Este estudo permitiu comparar a avaliação dos padrões arquiteturais com o conhecimento dos especialistas. Porém, para que possamos ter melhores subsídios para apoiar a avaliação contínua dos padrões, e estes passem a representar de melhor forma o conhecimento sobre a aplicação dos padrões em cenários de projeto arquitetural, é necessário que haja um maior volume de participantes, para que resultados divergentes possam ser analisados com mais precisão.

5.4 O Ajuste na Avaliação dos Padrões Arquiteturais

A análise dos resultados obtidos com a realização do estudo nos mostrou a necessidade do ajuste da primeira avaliação dos padrões arquiteturais (vide item 4.3). Esta nova avaliação sintetiza o resultado do processo de realimentação que iniciou-se com a participação dos especialistas e termina com um novo estado da base de conhecimento. A tabela 5.9 apresenta os novos resultados obtidos. A legenda utilizada para representar o aparente grau de esforço de utilização dos padrões arquiteturais foi mantida e apresenta os valores Muito Bom (MB), Bom (B), Médio (M), Ruim (R) , Muito Ruim (MR) e desconhecido (D). Os valores que foram modificados são apresentados em negrito.

Para melhor vislumbrarmos o efeito destas mudanças, submetemos Mentor novamente aos 5 cenários de projeto, com o objetivo de produzir indicações arquiteturais que reflitam os resultados obtidos com a realização do estudo. Para complementar o quadro de indicações arquiteturais referentes a cada cenário, colocamos uma breve descrição dos resultados esperados nesta nova fase de avaliação dos padrões.

Padrões Arquiteturais / Características de Qualidade	Camadas	Pipes & Filters	Blackboard	Broker	MVC	PAC	Microkernel	Reflection
Interoperabilidade	B	D	D	B	D	M	M	D
Segurança de acesso	MB	**D**	M	B	B	M	B	**MR**
Maturidade	M	R	MR	M	**B**	M	B	D
Tolerância a falhas	M	R	**R**	R	M	M	R	**MR**
Recuperabilidade	M	MR	R	**MR**	D	R	R	D
Operacionalidade	B	D	**R**	D	MB	MB	D	D
Comportamento em relação ao tempo	M	B	MR	MR	M	R	M	R
Comportamento em relação aos recursos	R	R	MR	MR	M	R	M	R
Modificabilidade	B	M	MB	B	B	R	MB	MB
Testabilidade	MB	R	MR	R	B	MR	M	D
Adaptabilidade	MB	MR	D	B	B	D	B	B

Tabela 5.9: Resultado da reavaliação dos padrões arquiteturais utilizados.

A síntese dos padrões que sofreram alterações em suas avaliações para cada um dos cenários e as indicações arquiteturais dadas pela ferramenta Mentor, antes e depois da evolução da base de avaliações, podem ser vistas, respectivamente, nas tabelas 5.10 e 5.11.

CENÁRIO	PADRÕES ALTERADOS
1	Pipes & Filters, Broker e Reflection
2	*Nenhum*
3	Blackboard e Reflection
4	MVC
5	*Nenhum*

Tabela 5.10: Síntese da análise dos cenários.

PADRÕES	CENÁRIO 1	CENÁRIO 2	CENÁRIO 3	CENÁRIO 4	CENÁRIO 5
Camadas	1º (1,05)	2º (1,13)	5º (1,13)	4º (0,96)	3º (1,10)
Pipes & Filters	5º (1,15)	1º (0,87)	3º (1,08)	5º (1,03)	8º (1,30)
Blackboard	7º (1,22)	8º (1,23)	7º (1,18)	7º (1,09)	7º (1,23)
Broker	6º (1,16)	6º (1,16)	4º (1,12)	6º (1,07)	6º (1,16)
MVC	2º (1,10)	6º (1,16)	2º (1,01)	3º (0,94)	3º (1,10)
PAC	2º (1,10)	4º (1,14)	1º (0,83)	8º (1,20)	5º (1,14)
Microkernel	4º (1,12)	2º (1,13)	8º (1,19)	2º (0,84)	1º (1,07)
Reflection	8º (1,41)	5º (1,15)	6º (1,15)	1º (0,78)	1º (1,07)

Tabela 5.11: Indicações da ferramenta Mentor

5.5 Conclusão

A elaboração do processo de reavaliação dos padrões arquiteturais se fez necessária porque percebemos que a confiabilidade das indicações utilizadas pela abordagem de seleção arquitetural estava sujeita à limitada fonte de referências existente, o que poderia comprometer a qualidade dos resultados apresentados com o uso deste mecanismo de seleção de padrões.

Sua execução, através da aplicação do estudo das relações existentes entre as características arquiteturais utilizadas e os padrões arquiteturais OO, mostrou que um processo deste tipo pode ser aplicado repetidas vezes, possibilitando que a base de avaliações evolua o suficiente para apoiar corretamente a aplicação de padrões arquiteturais no desenvolvimento de software. No entanto, este processo deve ser realizado de maneira acumulativa, para que a base de avaliações não corra o risco de sempre refletir os resultados obtidos com o último grupo avaliado.

Porém, percebemos que este estudo está limitado a um pequeno número de cenários de projeto, bem como no número reduzido de características arquiteturais envolvidas em cada cenário, o que acaba restringindo o escopo dos resultados obtidos. No entanto, com o tempo, prevemos a possibilidade de novas situações serem propostas, ampliando assim o leque de opções no que se refere aos exemplos utilizados em cada nova aplicação do estudo.

Capítulo 6 - Conclusão

6.1 Visão geral

Segundo (KRUEGER, 1992), para que uma técnica de reutilização de software seja efetiva, esta deve reduzir a distância cognitiva entre os conceitos iniciais identificados e a implementação final do software. A arquitetura de referência de um domínio de aplicação pretende tornar mais efetivo o compartilhamento e a reutilização de idéias, métodos, componentes e produtos dentro de uma comunidade de desenvolvedores de aplicações similares. Esta abordagem representa a concretização de uma proposta de reutilização de software com ênfase arquitetural.

Este trabalho foi pautado por todos estes aspectos, sempre com o intuito de facilitar a reutilização de software em infra-estruturas de suporte às engenharias de Domínio e de Aplicação. Assim, acreditamos que, uma vez definidas etapas e atividades, estas poderão apoiar o desenvolvimento de software centrado em sua arquitetura.

6.2 Comparação com outras abordagens

Na tabela 6.1, é mostrada uma comparação entre as propostas contidas neste trabalho e algumas abordagens descritas na literatura. Foram escolhidas RSEB/OOSE (JACOBSON *et al.*, 1997), EDLC/KBRET (GOMAA e FARRUK, 1999), MRAM/TRAM (MANNION *et al.*, 1999) e ARPA/DSSA (HAYES, 1994) por representarem boa parte das abordagens de reutilização de software baseada em arquiteturas de referência. Os tópicos escolhidos para a comparação são o Nível de detalhamento dos processos de criação e instanciação de arquiteturas de referência, o Critério utilizado para a definição da organização dada à arquitetura de referência, a Representação das características arquiteturais de domínio, a Utilização de base de padrões arquiteturais, o Formalismo dado à representação arquitetural, o Grau de integração entre os modelos conceituais e arquiteturais e o Critério utilizado para minimizar os possíveis conflitos arquiteturais que podem ocorrer durante a instanciação da arquitetura de referência. Esta comparação foi feita com base em dados existentes na literatura.

Abordagens / Critérios	ODYSSEY	RSEB/OOSE	EDLC/KBRET	MRAM/TRAM	ARPA/DSSA
Nível de detalhamento	**Alto**	**Médio**	**Médio**	**Médio**	**Médio**
Organização da arq. referência	**Escolha apoiada**	**Em camadas**	**Influência do domínio**	**Em camadas**	**Em camadas**
Características arquiteturais	**Sim**	**Sim**	**Não**	**Não**	**Sim**
Base de padrões	**Sim**	**Não**	**Não**	**Não**	**Não**
Formalismo	**Médio**	**Médio**	**Alto**	**Médio**	**Médio**
Integração	**Alto**	**Alto**	**Baixo**	**Baixo**	**Baixo**
Instanciação	**Apoiada**	**Direta**	**Direta**	**Não definida**	**Direta**

Tabela 6.1: Comparação entre abordagens de reutilização apoiadas em arquiteturas de referência

Quanto ao nível de detalhamento dos processos de criação e instanciação de arquiteturas de referência, o consideramos alto, pois preocupamo-nos em descrever suas atividades de maneira a ser possível a sua utilização. As outras quatro abordagens têm um nível de detalhamento considerado médio, seja porque não detalham os critérios de definição da arquitetura de referência (EDLC), ou porque dão pouca ênfase ao processo de instanciação desta arquitetura (RSEB, MRAM e ARPA).

Quanto a organização dada à arquitetura de referência, o Odyssey adota a abordagem proposta por esse trabalho no que diz respeito ao apoio dado durante a definição da arquitetura de referência. A proposta do EDLC estabelece que a organização da arquitetura depende da natureza do domínio que está sendo analisado. As demais abordagens utilizam uma organização em camadas de suas arquiteturas, independentemente das características do domínio.

O Odyssey define explicitamente uma fase de especificação de características arquiteturais que são utilizadas tanto para indicar possíveis soluções para a criação da arquitetura de referência quanto para estabelecer diretrizes que apoiarão a instanciação desta mesma arquitetura. Representações explícitas das características arquiteturais de qualidade também são encontradas nas abordagens RSEB e ARPA, mas não são definidos quaisquer critérios que as utilizem durante os processos de criação e instanciação da arquitetura de referência. As demais abordagens não mencionam qualquer atividade que se atenha a representar ou utilizar tais características.

Somente o Odyssey faz uso de uma base de padrões arquiteturais. A abordagem EDLC sequer faz referência a possibilidade de utilizá-la e as demais abordagens teoricamente não precisam desta base porque, por *default*, suas arquiteturas sempre seguem o modelo arquitetural Camadas.

Somente a abordagem EDLC apresenta um maior formalismo na representação da arquitetura de referência. As demais abordagens utilizam uma representação arquitetural baseada em modelos OO.

A integração dos modelos que compõem o domínio foi uma preocupação constante na proposta do Odyssey, que prevê a interligação entre os componentes e faz uso dela durante a criação e instanciação da arquitetura de referência. A abordagem RSEB também prevê um conceito similar, possuindo alta coesão entre os modelos. As demais abordagens são frágeis neste ponto, principalmente ao não indicar de forma explícita as ligações existentes entre os conceitos de domínio e os seus correspondentes em projeto.

O Odyssey adota um critério de apoio a instanciação da arquitetura de referência, principalmente nas situações em que há divergência entre as características arquiteturais de domínio e as da aplicação. As abordagens RSEB, EDLC e ARPA estabelecem que a instanciação da arquitetura de referência é feita diretamente, sem considerar que este tipo de conflito possa existir em quaisquer das aplicações desenvolvidas para o domínio. A abordagem MRAM não apresenta qualquer descrição sobre o processo de instanciação.

6.3 Contribuições da abordagem

Nesta tese, identificamos as principais etapas e atividades envolvidas nos processos de criação e instanciação de arquiteturas de referência no contexto da infra-estrutura de reutilização Odyssey. Esta atividade teve como foco o apoio à transição entre os modelos conceituais e arquiteturais.

A proposta aqui definida seguiu os principais pontos considerados de consenso na literatura. Podemos citar como exemplos as possibilidades de representação arquitetural, a importância de um processo de desenvolvimento centrado na arquitetura.

A integração dos processos de ED e EA, a partir da arquitetura de referência, também foi enfatizada neste trabalho. Deste modo, foram sugeridas extensões nas etapas de modelagem conceitual e arquitetural em ambos os processos, para que os processos de criação e instanciação de arquiteturas de referência pudessem ser viabilizados. As principais propostas neste sentido são a abordagem de seleção de padrões arquiteturais, que possui características que visam apoiar a transição entre as fases de análise e de projeto de domínio, e o apoio a instanciação da arquitetura de

referência no contexto de uma aplicação, que estabelece uma estratégia para o seu uso, extensão e/ou modificação.

Também foi implementada, no contexto deste trabalho, uma ferramenta que apóia parte do processo de criação de arquiteturas de referência. Assim, tarefas repetitivas, como a especificação de características arquiteturais de referência, e que envolvem uma tomada de decisão, como a escolha de um padrão arquitetural, são apoiadas por esta ferramenta.

Procuramos estabelecer, ainda, um processo de avaliação da aplicação de padrões arquiteturais em contextos específicos de projeto de software, para que possa contribuir no levantamento do conhecimento que é utilizado para apoiar a utilização dos padrões durante os processos de criação e instanciação da arquitetura de referência.

6.4 Deficiências da proposta e tópicos para pesquisa futura

O processo proposto nesta tese não teve a oportunidade de ser utilizado em, pelo menos, um estudo de caso real. A amplitude e complexidade das atividades necessárias para a criação e instanciação de arquiteturas de referência inviabilizaram o desenvolvimento de todo o ferramental necessário para apóia-los no contexto da infra-estrutura do Odyssey, como por exemplo a ferramenta que deverá apoiar a instanciação dos padrões arquiteturais. Contudo, o detalhamento das atividades possibilita novos esforços de trabalho relacionados a esta área de pesquisa.

O conjunto de características arquiteturais utilizado está limitado às influências técnicas, sendo as demais descaracterizadas do contexto desta tese.

A abordagem de seleção arquitetural carece de maior apoio tanto ao tratamento de possíveis conflitos entre as características arquiteturais de referência, tratadas hoje como características independentes umas das outras, quanto em relação a abordagem de seleção de padrões arquiteturais a partir da avaliação de distâncias vetoriais, que atende limitadamente os seus objetivos. A utilização dos padrões arquiteturais carece de maior tratamento para a utilização de mais de um padrão durante a criação da arquitetura de referência.

Entre os trabalhos futuros sugeridos no intuito de dar continuidade a este trabalho, podemos citar a extensão do conjunto de padrões arquiteturais OO bem como o conjunto de ferramentas necessárias para suportar a aplicação dos padrões no contexto do projeto, o desenvolvimento de uma abordagem que permita a avaliação dos aspectos qualitativos relativos ao modelo arquitetural definido, a extensão do

conjunto de características arquiteturais utilizados e o tratamento da interdependência entre as características arquiteturais, o ajuste da abordagem de classificação dos padrões arquiteturais para o tratamento da evolução dos critérios de avaliação utilizados e o desenvolvimento de um estudo que permita a identificação de um conjunto maior de características arquiteturais presentes no contexto de projeto e de que forma a aplicação de determinado padrão contribui para que estas características sejam ou não privilegiadas.

Referências

ABOWD, G., ALLEN, R., GARLAN, D., 1993, "Using Style to Give Meaning to Software Architecture." In: *Proceedings of SIGSOFT'93: Foundations of Software Engineering*, pp.9-20, Los Angeles, EUA, Dezembro.

APPLETON, B., 1997, *Patterns and Software: Essential Concepts and Terminology*, http://www.enteract.com/~bradapp/docs/patterns-intro.html.

ASADA, T., SWONGER, R. F., BOUNDS, N. *et al.*, 1992, *The Quantified Design Space: A Tool for the Quantitative Analysis of Design*, SEI Technical Report CMU/SEI-92-TR-213, Carnegie Mellon University.

BASS, L., CLEMENTS, P., KAZMAN, R., 1998, *Software Architecture in Practice*, 1 ed., Addison-Wesley.

BARBACCI, M., CARRIERE, S., KAZMAN, R., *et al.*, 1998, *Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis*, SEI Technical Report CMU/SEI-97-TR-029, Carnegie Mellon University.

BARROS, M., 1995, *Recuperação de Componentes em Bibliotecas de Software: Uma Abordagem Conexionista*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BARROS, M., WERNER, C. M. L., TRAVASSOS, G. H., 1999, "Risk Analysis: a Key Success Factor for Complex System Development", In: *Proceedings of the 12th International Conference on Software & System Engineering and their Applications (ICSSEA'99)*, v.5, artigo 19-1, Paris, França.

BERGLAND, G. D.,1981, "A Guided Tour of Program Design Methodologies", *IEEE Computer*, v. 14, n. 10 (Outubro), pp.13-37.

BOEHM, B. W., BROWN, J. R., KASPAR, J. R., *et al.*, 1978, "Characteristics of Software Quality", 1 ed., North-Holland.

BOLDRINI, J. L., COSTA, S. R., FIGUEIREDO, V. L., *et al.*, 1980, *Álgebra Linear*, 3 ed., capítulo 8, Harper & Row do Brasil.

BOOCH, G., 1994, *Object-Oriented Analysis and Design with Applications,* 2 ed., capítulo 1, The Benjamin/Clummings Publishing Company.

BOOCH, G., 1996, *Object Solutions: Managing the Object-Oriented Project,* 1 ed., capítulo 1, Addison-Wesley.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 1998, *The Unified Modeling Language User Guide,* 1 ed., Addison-Wesley.

BOSCH, J., MOLIN, P., 1999, “Software Architecture Design: Evaluation and Transformation”, In: *Proceedings of the Engineering of Computer-Based Systems Conference (ECBS99)*, pp. 1-10, Nashville, EUA, Março.

BOSCH, J., 2000, *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*, 1 ed., Addison-Wesley.

BLUEPRINT, 2001, "Framework Studio", http://www.blueprinte-technologies.com.

BUSCHMANN, F., MEUNIER R., 1994, “A Systems of Patterns”, In: *Proceedings of the First International Conference on Pattern Languages of Programming (PloP94)*, pp. 325-343, Illinois, EUA, Janeiro.

BUSCHMANN, F., MEUNIER R., ROHNERT, H., *et al.*, 1996, *Pattern-Oriented Software Architecture, A System of Patterns*, 1 ed., John Wiley & Sons.

BUSCHMANN, F., 1999, “Building Software With Patterns”. http://www.daimi.au.dk/~apaipi/dpf/EuroPLop.pdf.

BRAGA, R. M. M., WERNER, C. M. L., 1999, “Odyssey-DE: Um Processo para Desenvolvimento de Componentes Reutilizáveis”, In: *X Conferência Internacional em Tecnologia de Software (X CITS)*, pp. 177-194, Curitiba, Brasil, Maio.

BRAGA, R. M. M., WERNER, C. M. L., MATTOSO, M. L. Q., 1999, “Odyssey: A Reuse Environment Based on Domain Models”, 2nd IEEE *Symposium on Application-Specific System and Software Engineering Technology (ASSET’99),* pp. 50-57, Richardson, EUA, Março.

BRAGA, R. M. M., 2000, “Busca e Recuperação de Componentes em Ambientes de Reutilização de Software”, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BROOKS, F. P., 1975, *The Mythical Man-Month – Essays on Software Engineering,* 1 ed., Addison-Wesley.

CHAN, S., LAMMERS, T., 1998, “Reusing a Distributed Object Domain Framework”, In: *Proceedings of the* 5th *International Conference on Software Reuse (ICSR-5),* pp. 110-122, Vitória, Canadá, Junho.

CHUNG, L. K., NIXON, B., YU, E., 1994, "Using Quality Requirements to Drive Software Development", *Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence*, pp. 16-17, Sorrento, Itália, Maio.

CHUNG, L. K., NIXON, B., 1995, “Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach”, In: *Proceedings of 17th International Conference on Software Engineering*, pp. 24-28, Washington, EUA, Abril.

CHUNG, L. K., NIXON, B., YU, E., *et al.,*1999, “Non-Functional Requirements in Software Engineering”, 1 ed., Kluwer Academic Publishers.

CLEMENTS, P., 1995, “Understanding Architectural Influences and Decisions in Large System Projects”. In: *Proceedings of the 1st International Workshop on Architectures for Software Systems*, pp.31-43, Seattle, EUA, Abril.

CLEMENTS, P., 1996, *Coming to Abstractions In Software Architecture,* SEI Technical Report - CMU/SEI - 96- TR-008, Carnegie Mellon University.

CORREA, A., WERNER, C., ZAVERUCHA, G., 2000, “Object Oriented Design Expertise Reuse: an Approach based on Heuristics, Design Patterns and Anti-Patterns”, In: *6th International Conference on Software Reuse (ICSR-6)*, pp. 336-352, Viena, Áustria, Junho.

CYSNEIROS, L. M., 2001, *Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual*. Tese de D.Sc., PUC-RJ, Rio de Janeiro, RJ, Brasil.

DATE, C. J., 1991, *Introdução a Sistemas de Banco de Dados*, 4 ed., capítulo 1, Editora Campus.

DEWAYNE , E., ALEXANDER L. W., 1992, “Foundations for the Study of Software Architecture”, *ACM SIGSOFT Software Engineering Notes*, v. 17 n. 4, (Outubro), pp. 40-52.

FOX, R., 1996 , “News Track”, *Communications of the ACM*, v. 40, n. 5, (Maio), pp. 9-10.

GACEK, C., 1995, “Exploiting Domain Architectures in Software Reuse”. In: *Proceedings of the ACM-SIGSOFT* Symposium on Software Reusability (*SSR’95*), pp. 229-232, Seattle, EUA, Abril.

GACEK, C., BOEHM, B., ADB-ALLAH, A., *et al.*, 1995 , “On the Definition of Software Architecture”. In: *Proceedings of the First International Workshop on Architectures for Software Systems – In Cooperation with the 17th International Conference on Software Engineering*, D. Garlan (ed.), pp.85-95, Seattle, EUA, Abril.

GACEK, C., 1998, *Detecting Architectural Mismatches During Systems Composition*. Tese de D.Sc., University of Southern California, Los Angeles, EUA.

GAMMA, E., HELM, R., JONHSON, R., *et al.*, 1995, *Design Patterns – Elements of Reusable Object-Oriented Software*, 1ed. Addison-Wesley.

GRISS, M., FAVARO, J., D'ALESSANDRO, M., 1998, "Integrating Feature Modeling with RSEB", 5th International Conference on Software Reuse (ICSR-5), ACM/IEEE, Victoria, Canadá, Junho, pp. 36-45.

GRISS, M., 1999, “Domain Engineering and Reuse”, IEEE *Computer,* v.32, n.5 (Maio), pp 23-33.

GOMAA, H., FARRUKH, G. A., 1999, “A Reusable Architecture for Federated Client/Server Systems”, In: *Proceedings of the Symposium on Software Reusability (SSR’99*), pp. 113-121, Los Angeles, EUA, Maio.

GUARINO, N., 1998, "Formal Ontology and Information Systems", In: *Proceedings of the 1st International Conference on Formal Ontology in Information Systems*, IOS Press, pp.3-15, Trento, Itália, Junho.

HAYES, R., 1994, *Architecture-Based Acquisition and Development of Software Guidelines and Recommendations from the ARPA Domain-Specific Software Architecture (DSSA) Program*. In: Technical Report, Tecknowledge Federal System, Outubro.

HP, 2000, HP Architecture, http://www.architecture.external.hp.com.

ISO9126, 1992, “*International Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use”.* International Organization for Standardization, Geneva.

JACOBSON, I., GRISS, M., JONSSON, P., 1997, S*oftware Reuse: Architecture, Process and Organization for Business Success*, 1 ed., Addison-Wesley.

JOHNSON, R., FOOTE, B. , 1988, “Designing Reusable Classes”, *Journal of Object-Oriented Programming,* v.1, n.2, (Fevereiro), pp.22-25.

JONES, T. C., 1986, *Programming Productivity.* 1ed., McGraw-Hill Book Company.

KAZMAN, R., BASS, L., ABOWD, G., *et al.*, 1994, "SAAM: A Method for Analyzing the Properties of Software Architectures". In: *Proceedings of the 16th International Conference on Software Engineering* (ICSE), pp. 81-90, Sorento, Itália, Maio.

KAZMAN, R., KLEIN, M., 1999, *Attribute-Based Architectural Styles*, SEI Technical Report CMU/SEI-99-TR-022, Carnegie Mellon University.

KONTIO, J., 1995, *OTSO: A Systematic Process for Reusable Software Component Selection*, Computer Science Technical Report CS-TR-3478, University of Maryland.

LANE, T. G., 1990, *A Design Space and Design Rules for User Interface Software Architecture*, SEI Technical Report CMU-SEI-90-TR022, Carnegie Mellon University.

MAK, V., 1992, “Connection: An Inter-component Communication Paradigma for Configurable Distributed Systems”. In: *Proceedings of the International Workshop on Configurable Distributed Systems,* pp. 25-30, Londres, Inglaterra, Março.

MALAN, R., BREDEMEYER, D., 2000, “The Role of the Architect”, Bredemeyer Consulting, http://www.eacommunity.com/articles/art14.asp.

MANNION, M., KAINDL, H., WHEADON, J., *et al.*, 1999, “Reusing Single System Requirements from Application Family Requirements”, In: *Proceedings of the 21st International Conference of Software Engineering (ICSE’99)*, pp. 453-461, Los Angeles, EUA, Maio.

MEDVIDOVIC, N., TAYLOR, R., WHITEHEAD, J., 1996, “Formal Modeling of Software Architectures at Multiple Levels of Abstraction”. In: *Proceedings of the California Software Symposium* 1996*,* pp. 28-40, Los Angeles, EUA, Abril.

MEDVIDOVIC, N., TAYLOR, R., OREIZY, P., 1997, "Reuse of Off-the-Shelf Components in C2-Style Architectures", In: *Proceedings of the 1997 Symposium on Software Reusability (SSR'97)*, pp. 190-198, Boston, EUA, Maio.

MEEKEL, J., HORTON, T., FRANCE, R., *et al.*, 1997, “From Domain Models to Architecture Frameworks”. In: *Proceedings of the 1997 Symposium on Software Reusability (SSR’97)*, pp. 75-80, Boston, EUA, Maio.

MELLOR, S. J., JOHNSON, R., 1997, “Why Explore Objects Methods, Patterns, and Architectures?”, *IEEE Software*, v. 14, n.1, (Janeiro), pp. 27-30,

METTALA, E., GRAHAM, M. H., 1992, “The Domain-Specific Software Architecture”. SEI Technical Report CMU/SEI-92-SR-009, Carnegie Mellon University.

MILLER, N., 2000, *A Engenharia de Aplicações no Contexto da Reutilização Baseada em Modelos de Domínio*, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MONETA, C., VERNAZZA, G., ZUNINO, R., 1990, *A Vectorial Definition of Conceptual Distance for Prototype Acquisition and Refinement*. Technical Report TUM-I9019, Technical University Munich.

MONROE, R., GARLAN, D., 1996, “Style-Based Reuse for Software Architectures”. In: *Proceedings of the Fourth International Conference on Software Reuse,* pp. 23-26, Orlando, EUA, Abril.

MURTA, L., 2000, *Uma Máquina de Processo de Desenvolvimento Baseado em Agentes Inteligentes,* Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil (em andamento).

NII, H., 1986 , “Blackboard Systems”, *AI Magazine,* v. 7, n. 3 (Março), pp. 38-53 e v. 7, n. 4 (Abril), pp. 82-107.

PARNAS, D. L., 1972, “On the Criteria To Be Used in Decomposing System Into Modules”, Communications of the ACM, v. 15, n. 12 (Dezembro), pp. 1053-1058.

PRESSMAN, R., 1995, “Software e Engenharia de Software”. In: *Engenharia de Software,* 3 ed., capítulo 1, Makron Books do Brasil.

PFLEEGER, S. L., 1998, “Design the System”, In: S*oftware Engineering: Theory and Practice.* 1 ed., capítulo 5, Prentice-Hall.

PRIETO-DÍAZ, R., 1987, “Domain Analysis for Reusability”. In: *Proceedings of COMPSAC 1987*, pp. 45-60 , Los Alamitos, EUA, IEEE Computer Society Press.

PRIETO-DÍAZ, R., FREEMAN P., 1987, “Classifying Software for Reusability”, IEEE Software, v. 4, n. 1 (Janeiro), pp. 6-17.

PRIETO-DÍAZ, R., 1991, “Implementing Faceted Classification for Software Reuse”, Communications of the ACM, v.34, n. 5, (Maio), pp. 88-97.

ROBBINS, J., REDMILES, D.,1998, “Software Architecture Critics in the Argo Design Environment”, *Knowledge-based Systems*, v. 11, n. 1 (Janeiro), pp. 47-60.

ROCHA, A. R., 1983, *Um Modelo para Avaliação da Qualidade de Especificações,* Tese de D. Sc., PUC-RJ, Rio de Janeiro, RJ, Brasil.

ROCHA, R., 2000, *Ruth Rocha conta a Odisséia*. 1 ed., Companhia das Letrinhas.

SHAW, M., 1989, “Large Scale Systems Require Higher Level Abstractions”, *IEEE Computer Society, Software Engineering Notes*, v. 14, n. 3 (Março), pp. 143-146.

SHAW, M., GARLAN, D., 1993, “An Introduction to Software Architecture”. In: V. Ambriola and G. Tortora (eds.), Advances in Software Engineering and Knowledge, Series on Software Engineering and Knowledge Engineering, Vol. 2 World Scientific Publishing Company, pp.1-39.

SHAW, M., GARLAN, D., 1994, *Characteristics of Higher-Level Languages for Software Architecture*. SEI Technical Report CMU-CS-94-210, Carnegie Mellon University.

SHAW, M., DELINE, R., KLEIN, D. V., *et al.*, 1995, “Abstractions for Software Architectures and Tools to Suport Them”, *IEEE Transactions on Software Engineering*, vol. 21, no. 4, Abril, pp. 314-335.

SHAW, M., CLEMENTS, P., 1996, “A Field Guide to Boxology: Preliminary Classification of Architecture Styles for Software Systems” manuscript.

SHAW, M., GARLAN, D., 1996, S*oftware Architecture: Perspectives on an Emerging Discipline*. 1 ed. New Jersey, Prentice-Hall.

SIMOS, M., , 1996, “Organization Domain Modeling (ODM): Domain Engineering as a Co-Methodology to Object-Oriented Techniques”, In: *Fusion Newsletter,* v. 4, Hewlett-Packard Laboratories, pp. 13-16.

SIMOS, M., ANTHONY, J., 1998, “Weaving the Model Web: A Multi-Modeling Approach to Concepts and Features in Domains Engineering”, In: *Proceedings of*

the 5th International Conference on Software Reuse (ICSR-5), ACM/IEEE, pp.94-102, Victoria, Canadá, Junho.

STAA, A., 2000, *Programação Modular.* 1 ed. Campus.

SZYPERSKI, C., 1998, *Component Software – Beyond Object-Oriented Programming*. 1 ed. Addison-Wesley.

TANENBAUM, A. S., 1995, *Sistemas Operacionais Modernos*. 1 ed. Prentice-Hall do Brasil Ltda.

TRACZ, W., 1987, "Software Reuse: Motivations and Inhibitors". In: *Proceedings of COMPCON'87,* pp. 358-363, Santa Clara, EUA, Fevereiro.

TRACZ, W., 1994, "DSSA (Domain-Specific Software Architecture) Pedagogical Example", *ACM Software Engineering Notes*, v. 20, n. 3 (Julho), pp. 49-62.

TRACZ, W., HAYES, R., 1994, "*DSSA Tool Requirements For Key Process Functions".* In: Technical Report, ADAGE-IBM-93-13B, Loral Federal Systems - Owego, Outubro, Version 1.1.

WENTZEL, K., 1994, "Software Reuse, Facts and Myths". In: *Proceedings of 16th Annual International Conference on Software Engineering,* pp. 267-273, Sorrento, Itália, Maio.

Anexo I: Formulário para a Avaliação da Utilização de Padrões Arquiteturais na Fase de Projeto de Aplicações

Caracterização do Especialista:

Nome (opcional): _____
e-mail (opcional): _____

Nível de Formação	
	Informática (bacharelado, ciência da computação, engenharia de sistemas/computação, etc.)
	Área tecnológica com extensão em informática
	Área humana com extensão em informática
Tempo de formação:_____ anos	

Área(s) de atuação(ões)			
Empresa		Universidade	
	Empresário		Professor
	Gerente de Informática		Pesquisador
	Gerente de Projeto		Aluno de Doutorado
	Analista de sistema		Aluno de Mestrado
	Programador		Aluno de Pós-Graduação
	Outro:		Aluno de Graduação
Tempo de atuação em empresa: _____ anos		Tempo de atuação em universidade: _____ anos	

Área(s) de trabalho com maior experiência			
	Gerência		Programação
	Especificação		Teste
	Projeto		Treinamento:
Outros:			

Natureza dos sistemas desenvolvidos			
	Sistemas de informações		Sistemas especialistas
	Sistemas operacionais		Sistemas em tempo real
Outros:			

Domínios de aplicação em que trabalhou			
	Administração		Gerenciador de Informações
	Agropecuário		Gerenciador de Redes
	Financeiro		Saúde
	Educacional		Software Gráfico
	Entretenimento		Teleinformática / Telecomunicações
	Ferramenta de Desenvolvimento de SW		Transportes
Outros:			

Já ouviu falar na ISO 9126 ?			
	Sim, conheço bem		Sim, porém não me lembro bem dos detalhes
	Não, nunca ouvi falar		

Já utilizou padrões arquiteturais e de projeto no desenvolvimento de software?	
	Sim, na maioria das aplicações
	Sim, porém em poucas aplicações
	Não, porém conheço bem os conceitos relacionados à aplicação de padrões
	Não sei do que se trata, mas tenho idéia sobre arquiteturas de software
	Nunca desenvolvi com o suporte dos padrões e não tenho idéia sobre arquiteturas de software

Qual a importância normalmente dada à especificação arquitetural das aplicações em que você participou do desenvolvimento?							
	Alto		Médio		Baixo		Nenhum

A Arquitetura do software influenciou algumas das características de qualidade das aplicações que você participou?							
	Completamente		Exerce influência secundária				Nenhuma
Se existiu alguma influencia, quais características você observou foram afetadas:							
	funcionalidade		usabilidade		manutenibilidade		complexidade
	confiabilidade		eficiência		portabilidade		risco

Já houve situação em que você encontrou a possibilidade de aplicar mais de uma solução arquitetural para resolver uma especificação de software?
Não
Sim, e optei pelo(s) seguinte(s) critério(s) de diferenciação:
( ) Utilizei a solução arquitetural que havia aplicado em outros projetos ( ) Utilizei uma solução arquitetural que é padrão onde eu trabalho ( ) Utilizei a solução arquitetural que melhor suportava as características críticas da especificação (performance, distribuição, etc.) ( ) Utilizei a solução arquitetural mais simples ( ) Outro:

Instruções para a avaliação dos padrões arquiteturais:

Assinale de acordo com a escala ordinal de (0) – (4) conforme descrito na tabela 1, o valor que melhor representa a aparente relação de esforço de desenvolvimento para que um projeto arquitetural representada por determinado padrão arquitetural OO disponibilize as características de qualidade utilizadas para a primeira fase da avaliação:

Valor	Equivalente a	Interpretação
0	Esforço desconhecido	Desconheço esta relação de esforço aplicado
1	Esforço baixo	Característica principal do padrão utilizado
2	Esforço médio	A utilização do padrão aplica-se a situações em que o esforço desprendido se justifica perante o benefício encontrado
3	Esforço grande	A utilização do padrão é injustificável na maioria das situações
4	Esforço muito grande	A utilização do padrão é altamente desfavorável

Tabela 1 – Graus de Presença do Critério – Escala Ordinal

Avaliação dos Padrões Arquiteturais OO

Padrão Arquitetural: Camadas

Descrição: padrão arquitetural que propõe a decomposição de aplicações em grupos de subtarefas, onde cada grupo se apresenta em um nível particular de abstração. Arquiteturas em camadas existem para fornecer níveis cumulativos de abstração sobre o topo de alguma base de funcionalidade. A principal característica estrutural deste padrão restringe a utilização dos serviços de uma camada do nível J apenas à camada J+1.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Pipes & Filters

Descrição: este padrão arquitetural suporta a divisão de uma função ou processo em várias etapas de processamento seqüenciais. Cada etapa recebe, processa e disponibiliza uma cadeia de dados e o fluxo de saída de uma etapa corresponde ao fluxo de entrada da etapa seguinte.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Blackboard

Descrição: Neste padrão, vários subsistemas organizam o conhecimento para a construção de uma possível solução aproximada ou parcial através do uso de uma memória compartilhada. Cada subsistema é especializado para solucionar uma fase particular da tarefa completa, e todos os subsistemas trabalham juntos para a aquisição da solução. Tais subsistemas não interagem diretamente entre si e nem há uma seqüência determinada para as suas ativações. Em vez disto, a direção que o sistema toma é principalmente determinada pelo estado corrente da aplicação. A principal variação deste padrão é o Repositório, uma generalização onde não há a especificação de um componente responsável pelo controle interno da aplicação. Sistemas que utilizam banco de dados tradicionais podem ser considerados como exemplos de repositórios.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Broker

Descrição: Pelo uso deste padrão, uma aplicação pode acessar serviços de outras aplicações simplesmente pelo envio de mensagens a objetos mediadores sem se preocupar com questões específicas relacionadas à comunicação entre processos (acoplamento, localização, transmissão de dados, etc.).

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Model – View – Controller

Descrição: Propõe a divisão da aplicação em três áreas principais: processamento, entrada e saída. O componente "modelo" encapsula dados e funcionalidade principais da aplicação. A "visão" mostra informações ao usuário obtidas de um modelo. Cada visão apresenta um componente "controlador" que recebe as entradas do usuário (na forma de eventos disparados) e os traduz em solicitações de serviços ao modelo e/ou visão. O modelo é independente de suas possíveis representações bem como do comportamento de entrada e pode apresentar múltiplas visões.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Presentation – Abstraction – Control

Descrição: Define uma estrutura para sistemas interativos na forma de uma hierarquia de agentes cooperativos. Cada agente é responsável por um aspecto específico da funcionalidade da aplicação e é composto por três componentes: apresentação, abstração e controle.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Microkernel

Descrição: Propõe a separação de um centro de funcionalidade mínimo das funcionalidades estendidas e partes específicas de clientes. O encapsulamento dos serviços fundamentais da aplicação deve ser realizado no componente Microkernel. As funcionalidades estendidas e específicas devem ser distribuídas entre os componentes restantes da arquitetura.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Padrão Arquitetural: Reflection

Descrição: Uma arquitetura que é dividida em duas principais partes: o nível meta, que fornece uma representação do software para o autoconhecimento da estrutura e comportamento do sistema através de componentes chamados *metaobjetos*, e o nível base, que define a lógica da aplicação, onde a implementação do sistema utiliza os metaobjetos. Uma interface é especificada para a manipulação dos metaobjetos através de um *protocolo de metaobjetos*.

Objetivo: Nível de **funcionalidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Interoperabilidade da aplicação	(0) (1) (2) (3) (4)
Segurança de acesso	(0) (1) (2) (3) (4)

Objetivo: Nível de **confiabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Maturidade	(0) (1) (2) (3) (4)
Tolerância a falhas	(0) (1) (2) (3) (4)
Recuperabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **usabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Operacionalidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **eficiência** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Comportamento em relação ao tempo	(0) (1) (2) (3) (4)
Comportamento em relação aos recursos	(0) (1) (2) (3) (4)

Objetivo: Nível de **manutenibilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Modificabilidade	(0) (1) (2) (3) (4)
Testabilidade	(0) (1) (2) (3) (4)

Objetivo: Nível de **portabilidade** alcançada através da arquitetura da aplicação

Critérios	Avaliação
Adaptabilidade	(0) (1) (2) (3) (4)

Anexo II: Exercício de avaliação da utilização dos padrões

Nome (opcional) _____
Código do Questionário: PG

Nível de Formação	
	Informática (bacharelado, ciência da computação, engenharia de sistemas/computação, etc.)
	Área tecnológica com extensão em informática
	Área humana com extensão em informática
Tempo de formação:_____ anos	

Área(s) de atuação(ões)			
Empresa		**Universidade**	
	Empresário		Professor
	Gerente de Informática		Pesquisador
	Gerente de Projeto		Aluno de Doutorado
	Analista de sistema		Aluno de Mestrado
	Programador		Aluno de Pós-Graduação
	Outro:		Aluno de Graduação
Tempo de atuação em empresa: _____ anos		Tempo de atuação em universidade: _____ anos	

Natureza dos sistemas desenvolvidos			
	Sistemas de informações		Sistemas especialistas
	Sistemas operacionais		Sistemas em tempo real
Outros:			

Domínios de aplicação em que trabalhou			
	Administração		Gerenciador de Informações
	Agropecuário		Gerenciador de Redes
	Financeiro		Saúde
	Educacional		Software Gráfico
	Entretenimento		Teleinformática / Telecomunicações
	Ferramenta de Desenvolvimento de SW		Transportes
Outros:			

Experiência em desenvolvimento de software
Por favor, classifique sua experiência nesta seção em relação à escala de 5-pontos:
1 = nenhuma; 2 = estudei em classe ou em livros; 3 = pratiquei em um projeto em classe;
4 = utilizei em um projeto na indústria; 5 = utilizei em mais de um projeto na indústria;

<u>Experiência em desenvolvimento</u>

- Experiência elicitando requisitos 1 2 3 4 5
- Experiência em projeto de sistemas 1 2 3 4 5
- Experiência na criação de projetos Orientado a Objetos (OO) 1 2 3 4 5
- Experiência em modificação de projetos para manutenção 1 2 3 4 5
- Experiência em codificação, baseado em projeto OO 1 2 3 4 5
- Experiência em manutenção de código 1 2 3 4 5
- Experiência em projeto utilizando algum padrão arquitetural 1 2 3 4 5

Exercício

A empresa de desenvolvimento "SIGA Soft" adota uma sistemática de desenvolvimento onde se busca alcançar características de qualidade[15] em seus produtos através de um projeto arquitetural bem realizado. Um dos seus principais clientes necessita de um software que o atenda em critérios bem definidos de qualidade. O propósito é descrever uma solução que demonstre ser adequada à suas necessidades.

Descrição do Sistema

O supermercado "Popular" identificou a necessidade de informatizar algumas de seus processos de negócio. O sistema principal será desenvolvido com o propósito de registrar as vendas de mercadorias e lidar com as possíveis formas de pagamento (dinheiro, cheque e cartão de crédito). O controle da abertura e fechamento dos caixas também deve ser tratado, bem como o processo de sangria (retirada de espécie monetária dos caixas para o cofre). Além disso, adotou-se uma política de descontos progressiva, onde o cliente que possui um histórico de compras alto ou de fidelidade ganha um desconto no ato do pagamento de acordo com uma política definida pela direção.

De acordo com a descrição do sistema, solicitamos que você responda as seguintes questões, para cada um dos cenário de projeto apresentados:

Cenário 1:

As maiores preocupações do projeto relacionam-se com a segurança e a integridade do sistema. Desta forma, a arquitetura deve privilegiar um acesso seguro as funções de abertura, fechamento e sangria dos caixas e em caso de falhas deve ser capaz de restabelecer e restaurar dados após a falha.
Características esperadas: Segurança de acesso e recuperabilidade.

Questões para o cenário 1:

1. Quais os padrões arquiteturais mais adequados de serem utilizados no contexto descrito acima (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:
_____
_____
_____

2. Quais os padrões arquiteturais menos adequados de serem utilizados no contexto descrito para o sistema (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

[15] A descrição das características de qualidade utilizadas neste exercício encontra-se no anexo III ao final deste material.

( ) Camadas ( ) Pipes & Filters ( ) Blackboard ( ) Broker
( ) MVC ( ) PAC ( ) Microkernel ( ) Reflection

Justificativa:
_____
_____
_____

3. Dos padrões arquiteturais não considerados nas duas questões acima, descreva as suas observações acerca dos motivos o levaram a excluí-los das listas especificadas anteriormente.

Resposta:
_____
_____
_____

Cenário 2:

Cenário 2: As maiores preocupações do projeto relacionam-se com a necessidade deste sistema em interagir com outros sistemas e com o seu desempenho. No primeiro caso, o sistema deverá interagir com os sistemas de tele-cheques e das administradoras de cartões para a confirmação da existência de crédito para as compras realizadas. E, finalmente, o acesso às informações sobre preços e descontos em compras deve ser restrito ao tempo máximo de 2 segundos.

Características esperadas: Interoperabilidade e comportamento em relação ao tempo.

Questões para o cenário 2:

4. Quais os padrões arquiteturais mais adequados de serem utilizados no contexto descrito acima (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas ( ) Pipes & Filters ( ) Blackboard ( ) Broker
( ) MVC ( ) PAC ( ) Microkernel ( ) Reflection

Justificativa:
_____
_____
_____

5. Quais os padrões arquiteturais menos adequados de serem utilizados no contexto descrito para o sistema (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas ( ) Pipes & Filters ( ) Blackboard ( ) Broker
( ) MVC ( ) PAC ( ) Microkernel ( ) Reflection

Justificativa:
_____
_____
_____

6. Dos padrões arquiteturais não considerados nas duas questões acima, descreva as suas observações acerca dos motivos o levaram a excluí-los das listas especificadas anteriormente.

Resposta:

_____
_____
_____
_____

Cenário 3:

Cenário 3: As maiores preocupações de projeto relacionam-se com o uso do sistema e do seu tratamento às falhas eventuais. Os funcionários responsáveis pela operação dos caixas em sua maioria não utilizam com freqüência computadores. Por isso, a operação e o controle do sistema devem ser intuitivos e de fácil aprendizado. Além disso, eventuais problemas que ocorram (cancelamento de uma venda, quebra de um posto de venda, etc.) não devem prejudicar o desempenho do sistema como um todo.
Características esperadas: Operacionalidade e tolerância à falhas.

Questões para o cenário 3:

7. Quais os padrões arquiteturais mais adequados de serem utilizados no contexto descrito acima (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:
_____
_____
_____

8. Quais os padrões arquiteturais menos adequados de serem utilizados no contexto descrito para o sistema (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:
_____
_____
_____

9. Dos padrões arquiteturais não considerados nas duas questões acima, descreva as suas observações acerca dos motivos o levaram a excluí-los das listas especificadas anteriormente.

Resposta:
_____
_____
_____

Cenário 4:

Cenário 4: As maiores preocupações de projeto relacionam-se com a baixa freqüência de falhas do sistema e da necessidade em mudar periodicamente o conjunto de regras de desconto aplicados. O sistema deverá ser usado exaustivamente, 24 horas por dia, 7 dia por semana e índices baixos de falhas devem ser mantidos. Deve ser também privilegiada a capacidade de modificar e testar o sistema nos casos de adição de novas políticas e/ou modificação das mesmas.
Características esperadas: Maturidade, modificabilidade e testabilidade.

Questões para o cenário 4:

10. Quais os padrões arquiteturais mais adequados de serem utilizados no contexto descrito acima (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:

_____
_____
_____

11. Quais os padrões arquiteturais menos adequados de serem utilizados no contexto descrito para o sistema (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:

_____
_____
_____

12. Dos padrões arquiteturais não considerados nas duas questões acima, descreva as suas observações acerca dos motivos o levaram a excluí-los das listas especificadas anteriormente.

Resposta:

_____
_____
_____

Cenário 5:

Cenário 5: As maiores preocupações de projeto relacionam-se com a capacidade de adaptação do sistema a um novo ambiente operacional e com o seu desempenho em relação ao uso dos recursos de hardware. Com a previsão de abertura de novas filiais e a possibilidade de negociação do sistema com outras redes parceiras, a opção da adaptação do sistema a novas plataformas operacionais mostra-se uma necessidade a ser atingida. Por fim, o sistema terá que utilizar de alguns recursos para a realização completa de sua função principal (leitores de código de barras, impressoras, etc). A comunicação com os mesmos deve ser otimizada, de forma que o processo de venda não fique de maneira alguma prejudicado por qualquer demora nesta comunicação. Características esperadas: Adaptabilidade e comportamento em relação aos recursos.

Questões para o cenário 5:

13. Quais os padrões arquiteturais mais adequados de serem utilizados no contexto descrito acima (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:

_____
_____
_____

14. Quais os padrões arquiteturais menos adequados de serem utilizados no contexto descrito para o sistema (indique pelo menos 2 padrões justificando o porquê de cada escolha) ?

( ) Camadas	( ) Pipes & Filters	( ) Blackboard	( ) Broker
( ) MVC	( ) PAC	( ) Microkernel	( ) Reflection

Justificativa:

_____
_____
_____

15. Dos padrões arquiteturais não considerados nas duas questões acima, descreva as suas observações acerca dos motivos o levaram a excluí-los das listas especificadas anteriormente.

Resposta:

_____
_____
_____

Avaliação final do exercício:

Para este exercício, quanto tempo foi gasto para analisar as alternativas arquiteturais e quais foram as suas fontes de decisão?

Resposta:

_____
_____
_____

Anexo III: Descrição das características e subcaracterísticas de qualidade utilizadas

(ISO/IEC 9126-1)

Características relacionadas a funcionalidade do software : referem-se à existência de um conjunto de funções que satisfaz necessidades explícitas ou implícitas e suas propriedades específicas, para a finalidade a que se destina o produto. São elas:

Interoperabilidade: Capacidade de interagir com outros sistemas;

Segurança de acesso: Capacidade de evitar acesso não autorizado a programas e dados.

Características relacionadas a confiabilidade do software: referem-se à capacidade do software manter o seu nível de desempenho, sob condições estabelecidas, por um determinado período de tempo. São elas:

Maturidade: Avalia a freqüência de falhas no software.

Tolerância a falhas: Avalia a capacidade de manter o nível de desempenho em casos de falhas.

Recuperabilidade: Avalia a capacidade do software em restabelecer e restaurar dados após a falha.

Características relacionadas a usabilidade do software: referem-se ao esforço necessário ao uso e à homologação individual de tal uso, por um conjunto de usuários estabelecidos ou subentendido. Representada por uma característica que é a:

Operacionalidade: Avalia o esforço do usuário para operar e controlar a operação de software.

Características relacionadas a eficiência do software: Refere-se ao relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizada, sob condições estabelecidas. São elas:

Comportamento em relação ao tempo: Avalia o tempo de resposta, o tempo de processamento e as taxas de "*troughput*" durante a execução do software.

Comportamento em relação aos recursos: Avalia a quantidade de recursos utilizada e a duração desta utilização durante a execução do software.

Características relacionadas a manutenibilidade do software: Refere-se ao esforço necessário para fazer modificações específicas no software. São elas:

Modificabilidade: Avalia o esforço necessário para a modificação e remoção de defeitos;

Testabilidade: Avalia o esforço necessário para validar as modificações realizadas.

Características relacionadas a portabilidade do software: Refere-se à habilidade do software ser transferido de um ambiente para outro. Representada por apenas característica que é a:

Adaptabilidade: Avalia a capacidade de adaptação do software em outros ambientes sem exercer ações e procedimentos adicionais e diferentes daqueles previstos originalmente para esta finalidade.