PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RADA: UMA ABORDAGEM PARA A DOCUMENTAÇÃO DE ARQUITETURAS DE REFERÊNCIA ATRAVÉS DE LINGUAGENS DE DESCRIÇÃO ARQUITETURAIS

Eduardo da Silva Brandes

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação, pelo Programa de Pós-Graduação em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientadora: Profa. Dra. Ana Paula Terra Bacelo

PORTO ALEGRE
2010

Dados Internacionais de Catalogação na Publicação (CIP)

B817R Brandes, Eduardo da Silva
RADA: uma abordagem para a documentação de arquiteturas de referência através de linguagens de descrição arquiteturais / Eduardo da Silva Brandes. – Porto Alegre, 2010.
211 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Profª. Drª. Ana Paula Terra Bacelo.

1. Informática. 2. Engenharia de Software. 3. Arquitetura de Computador. I. Bacelo, Ana Paula Terra. II. Título.

CDD 005.1

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada **"RADA: Uma Abordagem para a Documentação de Arquiteturas de Referência Através de Linguagens de Descrição Arquiteturais"**, apresentada por Eduardo da Silva Brandes, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 09/03/10 pela Comissão Examinadora:

Ana Paula T. Bacelo

Profa. Dra. Ana Paula Terra Bacelo – Orientadora — PPGCC/PUCRS

Marcelo Blois Ribeiro

Prof. Dr. Marcelo Blois Ribeiro – PPGCC/PUCRS

Cláudia Lima Werner

Profa. Dra. Claudia Maria Lima Werner – UFRJ

Homologada em 06/03/2014, conforme Ata No. 002 pela Comissão Coordenadora.

p/

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

À família,

Aos meus pais, Adilson e Sônia, e meu irmão Alexandre pelos exemplos de honestidade, dedicação e dignidade prestados ao longo da vida. São minhas referências até hoje.

À minha orientadora Ana Paula Bacelo,

Mais do que pela lição acadêmica valorosa, pelos exemplos de postura profissional, ética, dedicação e comprometimento. Também por acreditar em mim, pelas palavras de motivação nos momentos difíceis e mais ainda pelas críticas que me permitiram crescer e chegar até aqui.

Aos Amigos,

Alceu e Sandra, que mesmo a distância estão sempre torcendo por mim.

Ao Meu Colega Vinícius,

Mais do que um grande talento, é um grande colega.

Aos membros da banca,

Pela participação na avaliação deste trabalho. Em especial ao Prof. Dr. Marcelo Blois, o qual vem me acompanhando desde a graduação, sempre com valorosas e entusiasmadas críticas.

À PUC e aos Professores da Graduação e PPGCC

Pela acolhida desde o início das atividades da graduação até o Mestrado. Em especial aos professores Dr. Alfio e Dr. Celso.

À Dell

Pelo apoio financeiro no desenvolvimento deste trabalho.

À Pure Bros e ao Sr Fábio Magalhães,

Pela compreensão e pelo caráter demonstrado.

RADA: UMA ABORDAGEM PARA A DOCUMENTAÇÃO DE ARQUITETURAS DE REFERÊNCIA ATRAVÉS DE LINGUAGENS DE DESCRIÇÃO ARQUITETURAIS

RESUMO

Práticas de reúso em um contexto de desenvolvimento de software, assim como em outras atividades, contribuem significativamente para a melhoria da qualidade dos artefatos gerados. Nesse sentido, pesquisas em Engenharia de Domínio (ED) vêm propondo métodos e abordagens com o intuito de apoiarem o reúso de software. A fase de projeto dos métodos de ED visa à criação de artefatos com o objetivo de construir uma Arquitetura de Referência (AR) que constituem modelos de organização estrutural que representam os conceitos mais importantes entre as arquiteturas de software em um domínio. Porém, mesmo que a maioria dos métodos preveja o apoio a construção de AR, na prática o suporte oferecido é insuficiente ou até mesmo inexistente. Nesse contexto, o objetivo dessa pesquisa é propor uma abordagem sistematizada para a documentação de Arquiteturas de Referência para Domínios de Aplicações, integrada a um processo de Engenharia de Domínio com foco em Reuso.

Palavras-chave: Arquitetura de Referência, Arquitetura de Software, Engenharia de Domínio, Reúso, Linguagens de Descrição Arquiteturais.

RADA: AN APPROACH FOR DOCUMENTATION OF REFERENCE ARCHITECTURES USING ARCHITECTURAL DESCRIPTION LANGUAGES

ABSTRACT

Reuse practices in the software development context, as well as other activities, contribute significantly for improving the quality of generated artifacts. Domain Engineering (DE) has been proposing methods and approaches that aim at supporting software reuse. In the design phase of DE methods artifacts may be created for the Reference Architecture (RA) creation. A RA constitutes structural models that represent the main concepts among the software architecture of a given domain. Although the majority of methods intend to support of RA creation, in fact this support is not enough. The goal of this research is to propose a systematic approach for reference architecture documentation, integrated to a DE process.

Keywords: Reference Architecture, Software Architecture, Domain Engineering, Reuse, Architectural Description Languages.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ABLE - *Architecture-Based Languages and Environments*

ACME - *Architectural Description Language Carnegie Mellon*

AD - Análise de Domínio

ADA - Arquiteturas para Domínio de Aplicação

AE - *Architectural Element*

ADL - *Architectural Description Language*

ADLARS - *Architectural Description Language for Real-time Systems*

ANOVA - A*nalysis of Variance*

AOS - Arquitetura Orientada a Serviço

AR - Arquitetura de Referência

AS - Arquitetura de Software

CASE - *Computer-aided software engineering*

CBD-Arch-DE - *Component-Based Development - Architecture - Domain Engineering*

CMMI - *Capability Maturity Model Integration*

CMU - *Carnegie Mellon University*

DARE - *Domain Analysis and Reuse Environment*

DBC - Desenvolvimento Baseado em Componentes

DSSL - *Domain Software Specific Languages*

EA - Engenharia de Aplicação

EAADL - *Extended Architecture Analysis Description Language*

ED - Engenharia de Domínio

ES - Engenharia de Software

FODA - *Feature-Oriented Domain Analysis*

FORM - *Feature-Oriented Reuse Method*

GQM - *Goal Question Metric*

HTML - *HyperText Markup Language*

IDE - *Integrated Development Environment*

LP – Linha de Produto

LPS - Linha de Produto de Software

LDA - Linguagem de Descrição Arquitetural

MPS.Br - Melhoria de Processos de Software Brasileiro

OO - Orientação a Objetos

PDA - *Personal digital assistant*

P2P - *Peer-To-Peer*

PPGCC - Programa de Pós-Graduação em Ciência da Computação

PUCRS - Pontifícia Universidade Católica do Rio Grande do Sul

SEI - *Software Engineering Institute*

RADA - *Reference Architecture for Domain Applications*

UI - *User Interface*

UML - *Unified Modeling Language*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

> “Sentimos que, mesmo depois de serem respondidas todas as questões científicas possíveis, os problemas permanecem completamente intactos” [Wit09].

Ao longo dos anos a indústria de maneira geral vem se preocupando com a otimização e melhoria dos processos produtivos envolvidos na fabricação de seus produtos. O objetivo desse aprimoramento é tornar as empresas mais competitivas, uma vez que desenvolve nelas a capacidade de entregar ao mercado produtos melhores e mais baratos. Da mesma forma, as organizações que desenvolvem software buscam a melhoria contínua de seus processos de desenvolvimento para, dentre outras coisas, se manterem capazes de enfrentar um mercado cada vez mais globalizado e competitivo [Bir09].

Dentro desse contexto, empresas dedicadas ao desenvolvimento de software vêm adotando, por exemplo, modelos de qualidade software tais como o CMMI (*Capability Maturity Model Integration*) [Sof09b], MPS.BR (Melhoria de Processos de Software Brasileiro) [Sof09a], como forma de aprimoramento dos artefatos gerados como produto de trabalho.

Na contramão deste cenário, muitas empresas ainda utilizam processos de desenvolvimento *ad hoc*, onde cada produto é desenvolvido de uma maneira diferente, não existindo registros referentes à análise, projetos, casos de teste, componentes[1] gerados ou tampouco arquiteturas utilizadas. Nesse tipo de prática, a geração de artefatos depende muitas vezes, mais da experiência da equipe atuante em um determinado momento dentro da organização, do que propriamente de aplicações anteriores construídas [Bat95] [Haa95] [Ekl05]. Dessa forma, o histórico de aplicações desenvolvidas não é levado em consideração no momento da criação de novos

[1] Um componente é um artefato autocontido, claramente identificável e com interfaces bem definidas com documentação apropriada, e um grau de reutilização definido [Sam97].

produtos de software, não permitindo que aspectos comuns a vários produtos sejam identificados e, por conseguinte, reaproveitados [Gar97].

Tanto na produção de software quanto em outras áreas, a capacidade de utilizar um artefato para construir outro traz um ganho considerável, pois reduz o tempo necessário à sua elaboração, bem como previne que erros sejam cometidos duas ou mais vezes. Todas estas vantagens podem ser alcançadas, por exemplo, através da incorporação de práticas de reúso aos processos de desenvolvimento de software utilizados no dia a dia das empresas. Processos de desenvolvimento que adotam práticas de reúso levam a uma diminuição no tempo de desenvolvimento, o que ajuda a reduzir o *time to market* [Bat95] [Tra02]. Além disso, contribuem significativamente para o aumento da qualidade dos artefatos produzidos [Dav97].

O reúso de software tem como fundamento o reaproveitamento de componentes, partes de software, módulos de código, documentação, dados de teste, requisitos de projeto e *arquiteturas de software* [Dav97], dentre outros. Através de técnicas de reúso, e.g., Desenvolvimento Baseado em Componentes (DBC) [Dso99], *Frameworks*, Padrões, Arquiteturas Orientadas a Serviço (AOS), Engenharia de Domínio (ED), equipes de desenvolvimento de software têm condições de construir sistemas de maneira mais eficiente, em menos tempo e com maior qualidade [Bat95] [Siy99] [Coh03]. O princípio básico da reutilização é evitar o retrabalho, utilizando artefatos previamente desenvolvidos e testados para a criação de novos produtos [Bas03].

Nesse sentido, um dos principais problemas enfrentados por métodos de apoio ao reúso, e.g., *Features* [Kan90], DBC ou Linha de Produto de Software (LPS) [Sho03], está relacionado às dificuldades encontradas na criação de componentes reutilizáveis [Pri91]. Uma possível solução para esse problema consiste em estabelecer processos sistematizados para a criação de componentes em um domínio específico. Processos que fornecem esta capacidade, i.e efetuar a captura sistemática de componentes, são providos por métodos de Engenharia de Domínio [Pri91]. A Engenharia de Domínio estabelece uma sistemática para a geração de componentes, tendo como produto, por exemplo, a definição de uma linguagem do domínio, formada por uma coleção de regras que relacionam objetos e funções [Gim05]. Também são produzidos artefatos

que podem ser representados por casos de uso, modelos de domínio, arquiteturas de componentes, dentre outros [Bac06]. Um modelo de domínio é um elemento chave para relacionar os conceitos de mais alto nível aos demais artefatos de análise e também para os artefatos de projeto, além de ser considerado o melhor caminho para efetuar o recorte do domínio para a construção de uma aplicação [Bac04].

À medida que os estudos nessa área evoluíram, foram sendo propostos vários métodos de ED, sendo alguns deles apresentados de forma mais detalhada ao longo desse trabalho. Dentre os métodos de ED propostos podemos citar, em ordem cronológica: FODA [Kan90], Odyssey-DE [Bra00], Kobra [Atk02], FORM [Kan02], CBD-DARE [Fra05] e Arch-DE [Bac06].

Embora a literatura mostre que ao longo do tempo tenham sido propostos uma variedade de métodos de ED com o objetivo de apoiar ao reúso, percebe-se que os mesmos privilegiaram o reaproveitamento de alguns artefatos em detrimento de outros, tais como aqueles ligados a *criação* e *documentação* de Arquiteturas de Referência (AR). Arquiteturas de Referência, detalhadas no Capítulo 2, são modelos de organização estrutural que representam os conceitos mais importantes entre as arquiteturas de software em um domínio [Men02]. Tanto a representação de Arquiteturas de Software como de ARs, pode ser feita através do uso de Linguagens de Descrição Arquiteturais (ADL). As ADLs, conforme detalhamento apresentado no Capítulo 2 são linguagens utilizadas, originalmente, para descrever Arquiteturas de Software [Kou95] [Gar97] [Men00] [Har04], mas ao longo do tempo, pelo potencial observado em sua aplicação a Arquiteturas de Software, foram propostas algumas ADLs voltadas à representação de ARs, tais como: Koala [Asi04], ADLARS [Bas05] e EAADL [Oh07].

As abordagens de ED que demonstram alguma preocupação com a criação e documentação de Arquiteturas de Referência, não deixam claro como deve ser feita a especificação de tais arquiteturas [Bac04]. Além disso, tais abordagens não esclarecem a maneira como os componentes gerados através da execução de processos de ED devem ser dispostos para a criação e *documentação* de uma AR para um domínio [Asi03] [Bac04]. Também é observado que mesmo quando existe alguma

documentação, ela não é utilizada de maneira a apoiar a instanciação de aplicações dentro do domínio. Neste contexto, essa dissertação apresenta uma proposta para a documentação de Arquiteturas de Referência através de uma Linguagem de Descrição Arquitetural, de forma integrada a um processo ED, para apoiar o Arquiteto na documentação da arquitetura do domínio, bem como no momento da instanciação de uma aplicação específica para o domínio.

1.1 Questão de Pesquisa

Conforme exposto na seção anterior, este trabalho está inserido em um contexto de reúso de software, e dentre os vários artefatos passíveis de reutilização, tem foco na documentação de ARs e sua utilização durante a instanciação de uma aplicação de um domínio.

Arquiteturas de Software (AS), quando tratadas num contexto de reúso, são conhecidas como Arquiteturas de Referência. Uma AR, discutida em mais detalhes em uma seção específica, pode ser vista como um modelo de representação [Men02] [Bas03]. Dessa forma, a sua utilização se dá por meio de documentação específica elaborada através de uma notação *gráfica*, ou *textual* sob a forma de uma linguagem [Har04]. Com base nisso, formulamos a seguinte questão: “Uma ADL com suporte a especificação de propriedades inerentes a um domínio auxilia na melhor documentação de uma AR do domínio e na sua utilização durante a instanciação de uma aplicação?”.

1.2 Motivação

Ao longo do desenvolvimento dessa pesquisa foi realizada uma revisão sistemática e uma *survey*, conforme pôde ser observado no Anexo D. Estes dois trabalhos tiveram como objetivo o estudo de práticas relacionadas ao reúso e a *criação* e *documentação* de Arquiteturas de Referência. A revisão sistemática procurou capturar aspectos mais teóricos através da análise de artigos relacionados a ADLs aplicadas a *representação de ARs* (e.g Koala [Asi03], ADLARS [Bas05] e EAADL

[Oh07]) assim como as propostas para a *criação de ARs* (e.g., AR para Sistemas *Peer-To-Peer* [Haa06], Navegadores de Internet [God05], Controle de Aeronaves [Bat95] e Automotivos [Ekl05]). A *survey* buscou identificar aspectos mais práticos, através das respostas obtidas por meio de sua aplicação a profissionais da indústria.

Conforme pôde ser percebido através da análise dos resultados da revisão sistemática, foram observados alguns pontos em aberto tanto nos artigos relacionados as ADLs, quanto nas propostas para a criação de ARs. Dentre os problemas identificados, podemos destacar:

i. A fraca semântica dos diagramas utilizados para a representação das ARs geradas em um domínio, assim como a ausência de um mecanismo para a validação dos diagramas elaborados;
ii. A notação gráfica utilizada na construção dos diversos diagramas gerados durante a criação de uma AR, não segue nenhuma padronização, o que dificulta o seu entendimento;
iii. A falta de um ambiente com suporte a manipulação da ADL utilizada na documentação e/ou formalização das ARs;
iv. A inexistência de mecanismos dedicados a validação destas Linguagens de Descrição Arquiteturais;
v. A ausência de uma abordagem sistematizada para a geração de componentes utilizados na montagem das Arquiteturas de Referência;
vi. A falta de cobertura de documentação de requisitos de qualidade;
vii. A inexistência de rastro entre os artefatos, i.e., elementos arquiteturais criados durante a construção das Arquiteturas de Referência e demais artefatos do domínio, dos quais derivaram estes elementos arquiteturais, e.g., casos de uso, *features* e componentes. Um elemento arquitetural, no contexto desta pesquisa, é representado por um agrupamento de componentes realizado com base em critérios pré-definidos. Estes critérios de agrupamento são apresentados de forma mais detalhada no Capítulo 3.

Para facilitar a compreensão, estes itens foram organizados em 2 grupos: i. baixa qualidade dos elementos arquiteturais utilizados na construção das ARs, e ii. limitações semânticas das ADLs e falta de padronização da notação empregada por estas abordagens.

1.2.1 Baixa Qualidade dos Elementos Arquiteturais Utilizados na Construção de ARs

De acordo com o que pôde ser observado, a baixa qualidade dos elementos arquiteturais decorre da falta de definição de um processo sistematizado para a identificação desses componentes.

A qualidade dos componentes exerce influência direta na AR, pois os elementos arquiteturais que compõem uma AR são formados, basicamente, por componentes. Foi observado que em algumas abordagens, inclusive para domínios mais estabelecidos tais como Navegadores de Internet [God05] e sistemas para troca de informações *Peer-To-Peer* [Haa06], a identificação de componentes é demasiadamente dependente da experiência do arquiteto.

Outras propostas, tais como LegaToDSSA [Vas07], promovem a identificação de componentes através de engenharia reversa em sistemas legados. Esse tipo de abordagem, embora resolva o problema do excesso de dependência em relação a experiência do arquiteto no momento da criação de componentes, funciona apenas para sistemas os quais já foram criadas várias arquiteturas. Tal problema se deve pela necessidade da abordagem de operar sob arquiteturas pré-existentes.

Todos esses tipos de abordagens [Bat95] [Ekl05] [God05] [Haa06] [Vas07] para a criação de ARs apresentam dificuldades na fase de geração de componentes. Esses problemas, em geral, dizem respeito à geração de uma quantidade maior de componentes do que a realmente necessária para representar as características de um domínio, ou a criação de componentes redundantes, dentre outros.

1.2.2 Limitações Semânticas das ADLs e Falta de Padronização da Notação

Conforme pôde ser observado nas propostas para a documentação de ARs, normalmente, são fornecidas junto a cada uma delas duas formas de representação para uma arquitetura. Uma forma provida é através do uso de uma notação gráfica. Por exemplo, a Figura 1 mostra a notação gráfica de um AR a qual é formada por um conjunto de elementos que são utilizados para criação de um determinado diagrama. Em geral, o conjunto de elementos disponíveis para desenhar AR são pobres, o que limita semanticamente as representações elaboradas.

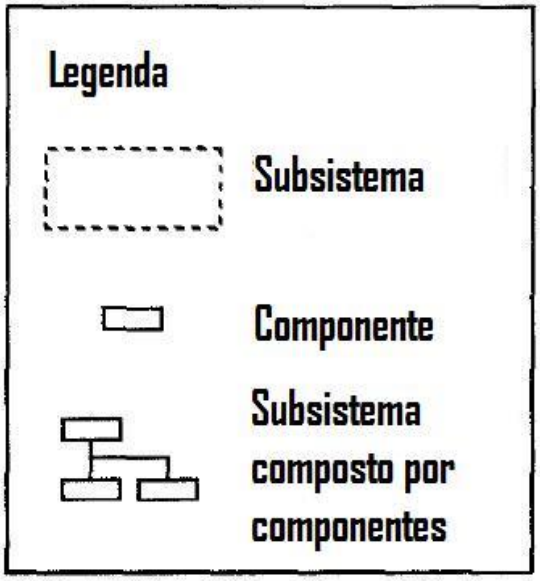


Figura 1 – Exemplo de notação gráfica para diagramas de Arquitetura de Referência (domínio Avionics) [Bat95]

Outra forma de representação provida é através da utilização de uma linguagem de descrição, não necessariamente formal, utilizada para representar a estrutura da arquitetura, e.g., componentes, portas, conectores, propriedades arquiteturais, etc.

Conforme observado nas diversas abordagens analisadas em relação aos diagramas utilizados, foi percebido que eles não apresentaram padronização em relação aos elementos pertencentes a notação empregada. Esse problema também se

repete em relação a sintaxe das linguagens de descrição propostas. É o caso da Linguagem de Descrição Arquitetural EAADL (*Extended Architecture Analysis Description Language*) [Oh07], ilustrada na Figura 2. Para esta ADL não foram apresentados maiores detalhes tais como grupo de construtores, compilador e estudos de caso, tampouco foi provido um ambiente para sua manipulação e validação sintática. Além disso, com o grupo de construtores disponibilizado por ela, não é possível representar, por exemplo, o relacionamento entre componentes, *features* e casos de uso. Esta limitação, observada também em ADLARS [Bas05] (outra ADL para representação de Arquiteturas de Referência junto a domínios modelados em termos de *features*), faz com que o registro da rastreabilidade entre esse tipo de artefato fique limitado.

```
process implementation A.configurable
  subcomponents
    Thread1: thread C;
  connections
    DataConnection1: data port input -> Thread1.input;
    DataConnection2: data port Thread1.output -> output;
  properties
    SPL::Variability => (VP, mandatory, external);
    SPL::Alternative => ("A.impl1", "A.impl2", "1..1");
    SPL::RequireRelation => "C.impl1";
end A.configurable;
```

Figura 2 – Exemplo de Linguagem de Descrição Arquitetural EAADL [Oh07]

Também foi observado que as ADLs propostas para representação de ARs, tais como KOALA [Asi04], ADLARS [Bas05] e EAADL [Oh07], não apresentaram suporte para representação de Estilos Arquiteturais. Um Estilo Arquitetural pode ser considerado o ponto de partida em um projeto de software [Men02], e dentre as muitas vantagens em sua utilização, está o fato dele prover uma organização padronizada para arquitetura, contribuindo significativamente para melhoria da manutenibilidade do software. Conforme verificado no grupo de palavras chave apresentado para EAADL,

não foram encontrados elementos que permitissem a representação de um Estilo Arquitetural, o que configura uma limitação importante para esta linguagem.

Além das limitações sobre criação e documentação de ARs, discutidos ao longo desta seção, as propostas apresentadas não provêem uma sistemática para a modelagem e gerenciamento dos elementos arquiteturais criados. Além disso, também não mencionam como pode ser feita a instanciação de uma Arquitetura para o Domínio, a partir da utilização da AR e documentação geradas, o que representa um dos maiores objetivos quando se criam Arquiteturas de Referência.

Esse conjunto de problemas, aliado ao fato das abordagens dedicadas a criação e documentação de ARs não estarem integradas a um processo de ED, nos motiva a buscar uma alternativa para a criação e representação de elementos relacionados a uma Arquitetura de Referência, para um domínio de aplicações.

1.3 Objetivo

Essa pesquisa tem como objetivo principal propor uma abordagem, denominada RADA (*Reference Architecture for Domain Applications*), para a *documentação* de Arquiteturas de Referência e Instanciação de Aplicações, através da utilização de uma Linguagem de Descrição Arquitetural. Além do objetivo principal, foram identificados alguns objetivos a serem atingidos pelo trabalho, conforme seguem:

i. Integrar a construção e documentação de uma Arquitetura de Referência a um processo de Engenharia de Domínio, disponibilizando junto a este processo, subsídios para apoiar a construção de uma Arquitetura de Referência, sob a forma de um *template* de arquitetura para o domínio;

ii. Dar apoio ao processo de instanciação de uma aplicação de um domínio através da documentação gerada para a respectiva Arquitetura de Referência do Domínio.

1.4 Estrutura da Dissertação

Essa dissertação está organizada de maneira a fornecer uma contextualização sobre o trabalho de pesquisa que foi desenvolvido. Para tanto, o Capítulo 1 introduz temas ligados ao reúso, da mesma forma como procura apresentar a base sobre a qual o trabalho veio sendo conduzido, descrevendo as dificuldades, conceitos e temas subjacentes ligados ao reúso.

Ao longo dos capítulos 2 e 3, são explorados temas ligados a base teórica, tais como Arquiteturas de Software, Arquiteturas de Referência, Linguagens de Descrição Arquiteturais e Engenharia de Domínio. Em relação à Arquitetura de Referência e Linguagens de Descrição Arquiteturais, são apresentadas as principais propostas referentes a esses dois temas. No que diz respeito à Engenharia de Domínio, são apresentados alguns conceitos relacionados a esta área de conhecimento, assim como descritos alguns dos métodos mais importantes. Um deles é o FODA [Kan90], tido como precursor de todos os outros, também é descrito o FORM [Kan02], semelhante ao FODA, mas com algumas melhorias e o CBD-Arch-DE [Bac06], que foi o método utilizado como base para nossa abordagem. Dentro do Capítulo 3 também é analisado junto aos métodos de ED, um método para a criação de arquiteturas baseado em funcionalidades [Bos00]. O Capítulo 4 apresenta em detalhes a abordagem RADA, mostrando todas as suas etapas e atividades envolvidas. Também no Capítulo 4 é apresentada mais especificamente a contribuição deste trabalho, sob o ponto de vista das extensões propostas (estrutura para a representação de uma AR e criação de suporte a descrição de AR através de uma ADL base). O Capítulo 5 mostra a implementação da abordagem RADA, realizada junto a um ambiente de reúso com suporte prévio ao método de ED utilizado. Esta implementação nos permitiu a realização de experimentos prévios e um estudo de caso para a avaliação da abordagem. O Capítulo 6 apresenta um cenário de utilização da abordagem demonstrando o uso na prática da RADA. O Capítulo 7 apresenta algumas considerações finais e trabalhos futuros para esta pesquisa. Juntamente a estas

considerações, são colocadas algumas contribuições alcançadas, assim como algumas limitações observadas.

2 ARQUITETURAS DE SOFTWARE, ARQUITETURAS DE REFERÊNCIA E LINGUAGENS DE DESCRIÇÃO ARQUITETURAIS

> "A arquitetura de um software define o que é um sistema em termos de componentes computacionais e os relacionamentos entre esses componentes" [Sha96].

Esse capítulo apresenta alguns conceitos relacionados a Arquitetura de Software (AS), Arquiteturas de Referência (AR) e Linguagens de Descrição Arquiteturais (ADL). Ao longo das próximas seções serão descritas algumas propostas para a criação de ARs e para a sua documentação por meio de ADLs.

2.1 Introdução

A palavra Arquitetura passou nos últimos anos a ser largamente utilizada no campo da tecnologia. Hoje se fala em arquiteturas para hardware, microprocessadores, redes de computadores, e sistemas de software. Já o termo software, sob o ponto de vista da construção de sistemas, não está restrito apenas a programas de computadores associados a uma aplicação, mas também envolve toda a *documentação* necessária para a instalação, uso, desenvolvimento e manutenção dos programas [Men02].

O desenvolvimento de software é um processo desafiador, e depende, dentre outros aspectos, de uma boa análise para que seja possível capturar requisitos e transformá-los em software, e.g., códigos fonte, casos de teste, projetos, manuais, etc. Dentro desse contexto, também é desafiador propor uma Arquitetura de Software (AS), pois os sistemas atuais estão se tornando cada vez maiores e mais complexos [Men02]. A tarefa de criação de uma arquitetura para um sistema é um momento onde muitas decisões precisam ser tomadas. Estas definições podem afetar todo o ciclo de desenvolvimento do sistema [Sho03] determinando, dentre outras coisas, a sua

manutenibilidade, escalabilidade e potencial de adaptação frente a cobertura de novos requisitos.

Arquiteturas de Software, *quando relacionadas ao reúso, são conhecidas como* Arquiteturas de Referência (AR), i.e., abstrações de arquiteturas a partir das quais podem ser instanciadas arquiteturas físicas em determinado domínio [Aco06]. Arquiteturas de Referência, também tratadas neste trabalho de forma intercambiável como Arquiteturas para Domínios de Aplicações (ADA), servem como uma base sólida para a derivação de arquiteturas físicas dentro de um domínio específico. Isto é possível, pois ADAs representam os conceitos mais relevantes entre as arquiteturas das aplicações nesse domínio [Men02].

Conforme observado na literatura [Cle00] [Men02] [Var02], a documentação de arquiteturas de software pode ser feita de várias formas, tais como:

i. Linguagens de Descrição Arquiteturais (ADLs) [Men02]
ii. Visões: Funcional e Lógica, Visão de Código, Visão de Desenvolvimento e Estrutural, Visão de Concorrência, Processo, *Thread*, Visão Física e Evolutiva, e Visão de Ação de Usuário e *Feedback* [Cle00], e
iii. Através de Padrões de Arquitetura [Var02].

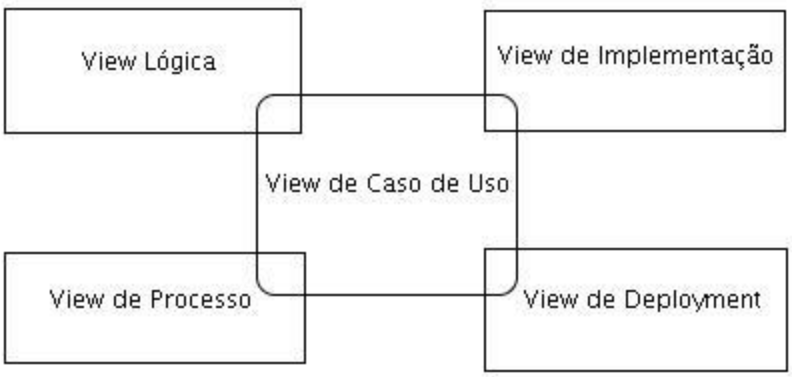


Figura 3 – Documentação de arquiteturas de software por Views

Na literatura, arquiteturas de software são documentadas de maneira mais recorrente utilizando a representação em termos de visões [Cle00] [Var02], conforme pôde ser observado na Figura 3. Visões são instâncias de pontos de vista, onde cada

ponto de vista representa a arquitetura sob a perspectiva de um conjunto de *stakeholders* [Var02]. Outra forma de documentar AS, é através da utilização de Linguagens de Descrição Arquiteturais (ADL), que são linguagens utilizadas para descrever arquiteturas de software [Kou95] [Gar97] [Men00] [Har04].

Ao longo das próximas seções são apresentados de maneira mais específica, alguns conceitos relacionados a Arquitetura de Software, Arquitetura de Referência e Linguagens de Descrição Arquiteturais. Estes conceitos são importantes para a compreensão do trabalho aqui proposto, uma vez que a pesquisa busca representar Arquiteturas de Referência através de uma Linguagem de Descrição Arquitetural, devidamente adaptada para essa finalidade.

2.2 Arquitetura de Software

Os ciclos de vida de desenvolvimento de sistemas de software, e.g Cascata [Roy70] e Espiral [Boe99], compreendem fases que vão desde a concepção do sistema, quando é feita a elicitação de requisitos, até a sua análise e implementação [Men02]. Após a realização da fase de elicitação de requisitos, onde as necessidades de *stakeholders* são levantadas e mapeadas para requisitos de software, se inicia a definição da uma *estrutura para software* através da identificação de seus componentes, mecanismos de interação e propriedades. O desenvolvimento de software no nível arquitetural compreende questões estruturais, dentre as quais se destacam:

i. Seleção de alternativas de projeto;
ii. Escalabilidade e desempenho;
iii. Organização e estrutura geral de controle;
iv. Protocolos de comunicação;
v. Sincronização;
vi. Atribuição de funcionalidade a componentes de projeto.

A Arquitetura de Software viabiliza, dessa forma, a comunicação entre todas as partes interessadas na construção de um sistema, facilitando a tomada de decisão sobre mudanças que podem causar impacto ao longo do trabalho de toda a Engenharia de Software (ES) [Gar97] [Sho03]. Além disso, ela constitui um modelo inteligível de como o sistema está estruturado e como os componentes trabalham simultaneamente para atingir determinados requisitos [Har04].

Tomando como base discussões realizadas no SEI/CMU (*Software Engineering Institute* da Carnegie Mellon University), a Arquitetura de Software foi definida por David Garlan e Dewayne Perry, como sendo a estrutura dos componentes de um sistema, seus inter-relacionamentos, princípios e diretrizes guiando o projeto e evolução ao longo do tempo [Gar97].

Especificamente no desenvolvimento de sistemas informatizados, uma Arquitetura de Software denota a organização existente entre um grupo de artefatos de software, bem como as relações existentes entre eles, de forma a viabilizar e apoiar a execução de alguma funcionalidade [Bas03]. Dessa forma, o projeto de arquiteturas representa um papel importante na construção de sistemas de software complexos, constituindo um fator determinante para o seu sucesso. Segundo Garlan [Gar97], a escolha de uma arquitetura apropriada pode levar a um produto que satisfaz requisitos e que é facilmente adaptável frente a mudanças ou a novos requisitos [Gar97].

Em outra definição, o autor [Pre05] coloca que uma Arquitetura de Software representa a estrutura de dados e componentes de um programa, necessários para a construção de um sistema baseado em computador. Essa definição afirma que uma arquitetura deve considerar o *Estilo Arquitetural* do sistema, avaliando as relações existentes entre todos os componentes do sistema. Os estilos de arquitetura expressam esquemas de organização estrutural de sistemas, fornecendo um conjunto de componentes do sistema, suas responsabilidades e a forma de interação entre eles, estabelecendo um padrão de utilização [Bus96]. Nesse sentido, a Arquitetura de Software é o estudo da organização global dos sistemas de software bem como do relacionamento entre subsistemas e componentes.

2.3 Arquitetura de Referência

Se a construção de uma arquitetura para uma única aplicação consiste em uma tarefa complicada, pois envolve a tomada de um conjunto de decisões que afetam todo o ciclo de desenvolvimento do sistema [Gar97] [Sho03], a construção de uma arquitetura para uma família de aplicações é uma atividade mais complexa ainda [Bac04]. Arquiteturas de Referência (AR), embora sejam utilizadas para representar Arquiteturas de Software (AS), não são AS. Uma AR consiste em um modelo onde são representados elementos de software que cooperativamente implementam as funcionalidades e fluxos de dados genéricos, observados em *várias arquiteturas* criadas dentro de um *domínio* específico [Dom07]. Uma Arquitetura de Referência necessita dar apoio aos requisitos impostos por uma família de aplicações [Bac04] [Poh07], e deve apresentar a capacidade de capturar, absorver e expressar as similaridades e variabilidades existentes entre os diferentes produtos instanciados junto a esta família [Bas05].

Em um artigo denominado *Uma Arquitetura de Referência para Web Browsers*, Alan Grosskurth e Michael W. Godfrey da Universidade Waterloo, no Canadá, definem AR como sendo a captura de subsistemas básicos comuns a vários sistemas de um domínio [God05]. Os autores ainda colocam uma vantagem importante, em relação a utilização de Arquiteturas de Referência. Segundo eles, ter uma Arquitetura de Referência disponível ajuda tanto no projeto quanto na manutenção, pois ela auxilia na compreensão dos sistemas e na análise de *trade offs*[2] entre diferentes opções de projeto, além de servir como um *template* para *novos sistemas* ou reengenharia em sistemas existentes.

Conforme visto até este ponto, a utilização de uma Arquitetura de Referência para derivar uma arquitetura física para uma nova aplicação pode alavancar e melhorar a qualidade do projeto destas novas arquiteturas de software. Esse aumento na

[2] Trade-off ou tradeoff é uma expressão que define uma situação em que pode haver conflito de escolha. Ele se caracteriza em uma ação econômica que visa à resolução de problema mas pode acarretar outro, obrigando uma escolha. Ocorre quando se abre mão de algum bem ou serviço distinto para se obter outro bem ou serviço distinto. [Wik09]

qualidade ocorre pelo fato da AR facilitar a captura de elementos arquiteturais [Bas03], o que proporciona um bom direcionamento das decisões sobre a Arquitetura a ser construída para uma aplicação.

2.4 Linguagens de Descrição Arquiteturais

Conforme mencionado na introdução deste capítulo, a documentação de Arquiteturas de Software pode ser feita através da utilização de várias abordagens, dentre elas as Linguagens de Descrição Arquiteturais, utilizadas para descrever arquiteturas de software [Kou95] [Gar97] [Men00] [Har04].

Em termos estruturais, ADLs precisam prover a capacidade de explicitar componentes, conectores e possíveis configurações entre eles [Gar97]. Segundo Harkki [Har04], o papel mínimo de uma ADL é ajudar *stakeholders* a compreender estruturas de sistemas [Har04]. Para tanto, elas devem ser simples, não necessariamente formais, e apresentar, quando possível, uma notação gráfica amigável. Também, segundo este autor, uma ADL deve apoiar a avaliação e validação de uma instância de arquitetura para uma aplicação, antes mesmo da sua construção, o que contribui para a realização de provas de conceito.

Segundo pesquisadores do SEI/CMU, uma ADL, conforme exemplo observado na Figura 4, usualmente deve fornecer um *framework* conceitual e uma sintaxe concreta para a caracterização de arquiteturas de software [Gar97]. De acordo com o autor, uma ADL tipicamente deveria dispor de ferramentas para *parsing*, *unparsing*, exibição, compilação, e análise ou simulação de descrições arquiteturais feitas na linguagem provida.

Ao longo do tempo, muitas Linguagens de Descrição Arquiteturais foram propostas, e.g., ADAGE (*Avionics Domain Application Generative Environmet*) [Cog93], Darwin [Mag95], Rapide [Luc96], Aesop, Meta-H, C2, SADL, ACME (*Architectural Description Language - Carnegie Mellon University*) [Gar97], Wright [All97], ADLARS [Bas05], EAADL (*Extended Architecture Analysis Description Language*) [Oh07].

Embora todas estas linguagens apresentem preocupação com a representação de elementos inerentes a uma arquitetura, cada uma propõe um grupo de capacidades distintas, por exemplo:

- Aesop: apóia o uso de estilos arquiteturais para a representação de arquiteturas de software, o que permite expressar esquemas de organização estrutural de sistemas.
- ADAGE: provê a capacidade de descrição de *frameworks* arquiteturais, os quais podem ser associados a mecanismos geradores de código, usados na criação de sistemas dedicados a navegação de aeronaves.
- C2: apóia a descrição de sistemas com interfaces de usuário (IU), através da utilização de estilos de mensagens.

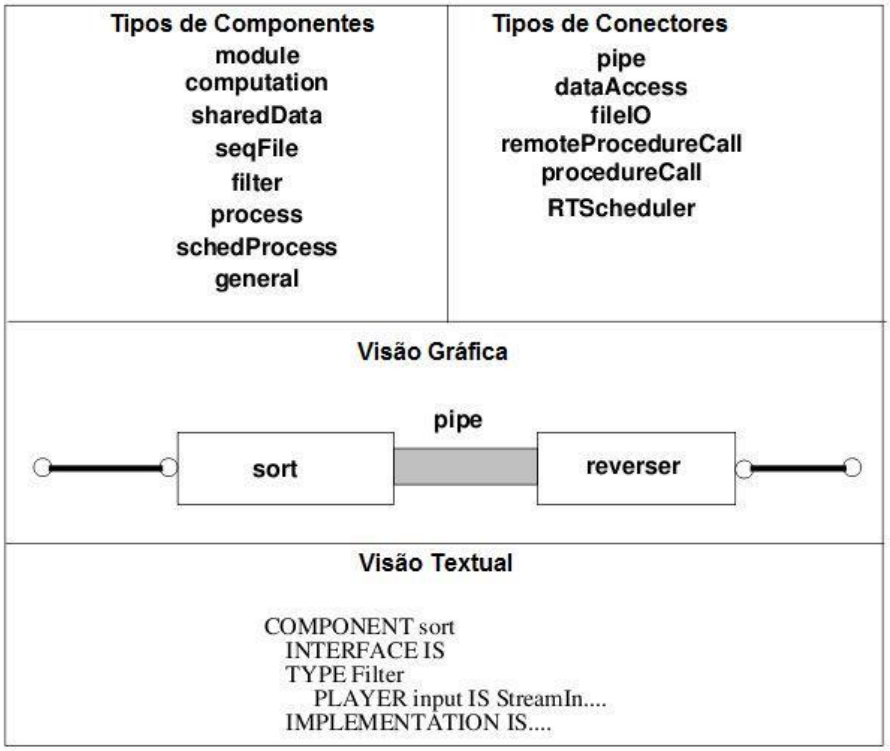


Figura 4 – Exemplo de Linguagem de Descrição Arquitetural [Har04]

Toda essa diversidade de linguagens dedicadas a representação de arquiteturas de software apresenta aspectos positivos e negativos. Por um lado, encontramos várias

linguagens de descrição explorando diferentes tipos de problemas inerentes ao projeto arquitetural, o que viabiliza o surgimento de diferentes caminhos e opções de solução para estes problemas [Gar97]. Por outro lado, cada uma dessas linguagens opera de forma isolada, o que dificulta a combinação das funcionalidades providas por elas [Gar97] [Har04].

Em um artigo apresentando a ADL ACME, pesquisadores do SEI/CMU destacaram a importância da formalização no projeto de arquiteturas. Segundo eles, grande parte do processo de projeto arquitetural é feito de maneira informal ou *ad hoc* [Gar97]. Como resultado, geralmente as arquiteturas criadas são parcialmente compreendidas pelos desenvolvedores [Har04]. A utilização de uma ADL melhora a compreensão sobre a implementação da arquitetura, pois fornece uma visão sobre protocolos de comunicação e o processo de desenvolvimento da arquitetura [Har04].

As Linguagens de Descrição Arquiteturais fornecem um caminho para representação de Arquiteturas de Software, não estando relacionadas a formalização de Arquiteturas de Referência. Sob o ponto de vista da criação de arquiteturas para domínios de aplicações junto a abordagens baseadas em *Features*, ou mesmo Linha de Produto de Software (SPL), a aplicação de uma ADL pode trazer os benefícios obtidos pelo seu uso na representação de arquiteturas de software.

Ainda, pelo fato de muitas linguagens serem formais, o processo de geração de um modelo passível de transformação independente de plataforma, por exemplo, através de *Model Driven Architecture* (MDA) [Gro09a], se tornaria mais viável. Da mesma forma, embora não seja objetivo dessa pesquisa, também se torna mais factível a *geração automatizada* de arquiteturas físicas a partir da ADL criada para o domínio.

2.5 Linguagens de Descrição Arquiteturais e Arquiteturas de Referência

Atualmente, considerando algumas técnicas de reúso de software tais como Desenvolvimento Baseado em Componentes (DBC) [Dso99], Engenharia de Domínio (Features [Kan90], Odyssey-DE [Wer09], CBD-Arch-DE [Bac06]) e Linha de Produto de Software (LPS) [Sho03], verificamos que elas fornecem pouco suporte a criação de

Arquiteturas de Referência (AR) (Oh07). Analisando estas abordagens sob o ponto de vista da disponibilização de uma documentação para ARs, principalmente no que diz respeito à representação de variabilidades, opcionalidades, dependências e parametrização junto a AR, estas técnicas apresentaram um suporte ainda menor [Bas05] [Oh07].

Para contribuir com a solução deste problema viabilizando a modelagem da variabilidade em AR, algumas ADLs foram propostas. Dentre as linguagens de descrição propostas com foco em representação de ARs podemos citar: Koala [Asi04], ADLARS [Bas05] e EAADL [Oh07]. Estas Linguagens tentaram através de mecanismos de modelagem, muitas vezes sob a forma de uma notação textual e/ou gráfica, prover recursos de modelagem para a representação de AR. Embora algumas destas ADLs tenham apresentado boas soluções para alguns dos problemas enfrentados durante a representação desse tipo de artefato, e.g., expressar dependência entre elementos, obrigatoriedade, opcionalidade, nem todos os problemas foram resolvidos.

Segundo Harkki [Har04] e Bass [Bas05], não existem ADLs capazes de dar apoio, por exemplo, ao relacionamento entre modelos de *Features* e a Arquitetura para um Domínio específico de Aplicações, tampouco para realizar a representação ou descrição de uma AR [Har04] [Bas05]. Isso deixa uma lacuna que, significativamente, aumenta a complexidade da análise de requisitos, por parte do Arquiteto, para construir e representar uma AR para um domínio [Bas05]. A conseqüência disso é que o Arquiteto não consegue assegurar que as características inerentes a um domínio, possam estar contempladas a cada nova arquitetura desenhada.

Durante o desenvolvimento desta pesquisa foram avaliadas algumas ADLs adaptadas para a representação de Arquiteturas de Referência, são elas: Koala [Asi04], ADLARS [Bas05] e EAADL [Oh07]. A partir desta análise, conforme já mencionado na seção 1.2, observamos que as linguagens deixam alguns pontos em aberto. Dentre eles podemos destacar: i) a falta de uma sistemática para seleção dos componentes utilizados na construção da AR, ii) a falta de padronização e suporte providos às linguagens e a notação gráfica propostas por estas ADLs e, iii) a falta de um ambiente de apoio à modelagem e construção destas arquiteturas. Por esta razão,

descartamos a utilização de quaisquer destas ADLs adaptadas como base para a nossa pesquisa.

Uma das ADLs incluídas no estudo foi a ACME [Gar07], conforme ilustrado na Figura 5. Esta linguagem foi construída com o objetivo de representar Arquiteturas de Software (AS), não estando em momento algum comprometida com a documentação de Arquiteturas de Referência (AR).

Embora ela não tenha sido elaborada para este fim, ACME apresenta mecanismos de extensão baseados em *property values* e elementos para a tipagem, o que permite a criação de novos elementos. Com isso amplia-se a capacidade de representação semântica da ACME, sem a necessidade de alterações no conjunto de construtores originais definidos pelo vocabulário da linguagem. A ACME é uma linguagem formal, e possui a capacidade de representação de arquiteturas de software. Esta representação é feita através da criação de estilos arquiteturais, os quais descrevem elementos que podem estar presentes em arquiteturas geradas com base nesses estilos. Conforme podemos observar na Figura 5, diferentemente da ADL apresentada como exemplo na Seção 1.2, a EAADL [Oh07], a qual apresentava como limitação o fato de não possuir elementos para a representação de Estilos Arquiteturais, ACME permite nativamente esta representação. Isso pode ser feito através da palavra chave *import*, que indica a inclusão de um determinado estilo pré definido, tal como *pipes*, *n-tier*, dentre outros.

ACME, assim como outras linguagens, possui um determinado conjunto de palavras chave. Dentre as principais palavras reservadas especificadas para o seu vocabulário, podemos destacar FAMILY, ELEMENT, COMPONENT, CONNECTOR, GROUP, PORT, ROLE, PROPERTY e TYPE. Através delas, a linguagem permite especificar novas classes de elementos, e.g., FAMILY NovaFamilia, COMPONENT NovoComponente.

Dessa forma, cada novo elemento criado, herda as propriedades do elemento da linguagem, a partir do qual ele foi criado. Um NovoComponente, por exemplo, herda a capacidade de representação de portas (PORT), papéis (ROLE), propriedades (PROPERTY), etc. Além desses elementos, a linguagem ainda disponibiliza instruções

de desvio condicional, de repetição e operadores baseados em lógica proposicional de segunda ordem.

```
TYPE-OR-FAMILY-DECLARATION;*

System SYSTEM-NAME [ : [ FAMILY-NAME, ]* FAMILY-NAME ] =
 new FAMILY-NAME extended with {
   COMPONENT-DECLARATION;*
   CONNECTOR-DECLARATION;*
   PROPERTY-DECLARATION;*
   REPRESENTATION-DECLARATION;*
   ATTACHMENT-LIST;
};

Property PROPERTY-NAME [ : TYPE-NAME ] = PROPERTY-VALUE;

CATEGORY-NAME : TYPE-REFS = {
    CHILDREN;
    PROPERTIES;
    REPRESENTATIONS;
};

ELEMENT-NAME : TYPE-REFS = {
    CHILDREN;
    PROPERTIES;
    REPRESENTATIONS;
};

Component NAME [ : [TYPE-NAME,]* TYPE-NAME ] = {
   PORT-DECLARATION;*
   PROPERTY-DECLARATION;*
   REPRESENTATION-DECLARATION;*
}
```

Figura 5 – Linguagem de Descrição Arquitetural ACME

Embora apresente recursos interessantes, ACME, assim como outras linguagens de descrição, se encontra isolada e operando de forma independente de qualquer processo de desenvolvimento de software, tampouco de Linha de Produto ou Engenharia de Domínio. Da mesma forma, essa linguagem não apresenta suporte à documentação de elementos ligados a variabilidade, opcionalidade ou rastros entre

artefatos, mesmo porque, apresenta foco na representação de Arquiteturas de Software, que por definição não tem estas características.

2.6 Considerações

Conforme pôde ser observado ao longo deste capítulo, o foco da utilização de Linguagens de Descrição Arquiteturais está na representação de arquiteturas de software. Também pode ser visto que ao longo do tempo surgiram algumas propostas de aplicação de ADLs adaptadas para a documentação de Arquiteturas de Referência. Embora tenham sido modificadas para tal, estas ADLs não conseguiram descrever a contento as propriedades inerentes as Arquiteturas de Referência para domínios baseados na modelagem em termos de *features*. Sendo assim, se apresentam ainda em um estado imaturo.

Dentro deste contexto, surgiu a ADL ACME, criada para a descrição de arquiteturas de Software. Esta ADL apresenta mecanismos de extensão que permitem que ela seja modificada para representar os mais diversos tipos de aplicações.

3 ENGENHARIA DE DOMÍNIO

> “O objetivo da Engenharia de Domínio é possibilitar que características comuns e variáveis possam ser identificadas e modeladas com base num processo previamente definido” [Bac06].

Esse capítulo apresenta o tema Engenharia de Domínio (ED). Conforme será visto, dentre as fases propostas por métodos de ED, tipicamente é definida uma etapa conhecida como fase de Projeto de Domínio. Nessa etapa é analisado o apoio dos métodos de ED às atividades de criação e documentação de Arquiteturas de Referência, assim como a instanciação de aplicações para um domínio específico. Dentre os métodos de ED existentes na literatura, é dado maior destaque aos métodos FODA [Kan90], Bosh [Bos00], FORM [Kan02] e CBD-Arch-DE [Bac06]. Os métodos de Kang, conforme comentado na seção 3.2, foram aqui destacados por terem sido utilizados como base para outros métodos de ED, tais como o CBD-Arch-DE. Ao final do capítulo é apresentada uma análise comparativa entre os métodos, seguindo alguns critérios previamente definidos e apresentados ao longo do capítulo, para caracterizar as lacunas deixadas pelas abordagens durante a execução da etapa analisada.

3.1 Introdução

Conforme discutido no Capítulo 1, as organizações buscam constantemente o aprimoramento dos seus processos de desenvolvimento. Uma forma considerável de otimização de processos, mas que apresenta muitos obstáculos em sua realização está relacionada à adoção de práticas de reúso de software. Dentre os obstáculos encontrados, podemos citar a dificuldade na criação de componentes que possam ser reutilizados em outras aplicações [Pri91]. Uma forma de mitigar este problema, segundo Prieto [Pri91], consiste em estabelecer processos sistematizados para a criação de componentes em um domínio específico, por meio de uma atividade denominada Análise de Domínio (AD). Para Gimenes [Gim05], a AD é uma forma de

identificar objetos, operações e relacionamentos entre tudo o que os especialistas julgam importante para o domínio [Gim05]. O principal produto destas atividades é a definição de uma linguagem de domínio, ou seja, o estabelecimento de um vocabulário único, formado por uma coleção de regras que relacionam objetos e funções [Gim05].

A ED é uma área de conhecimento que procura apoiar o reúso através de métodos dedicados a modelagem de domínios [Bac06]. O objetivo da ED é possibilitar que características comuns e variáveis possam ser identificadas e modeladas com base em um processo previamente definido [Bac06]. Para que isso possa ser viabilizado, alguns métodos de ED foram propostos ao longo dos anos, dentre eles podemos citar o FODA [Kan90], DARE [Kan00], Odyssey-DE [Bra00], FORM [Kan05] e CBD-Arch-DE [Bac06]. Estes métodos fornecem subsídios necessários para que a Engenharia de Domínio possa ser realizada. A execução de um processo de ED resulta em artefatos como casos de uso, modelos de domínio, arquiteturas de componentes, dentre outros [Bac06].

Estes artefatos são utilizados como insumo nas etapas posteriores de desenvolvimento de software, como a Engenharia de Aplicação (EA). A Engenharia de Aplicação atua junto a Engenharia de Domínio, e se ocupa em apoiar a construção de aplicações com base no reúso de artefatos fornecidos pela ED [Bac06]. Conforme apresenta a Figura 6, os artefatos produzidos pela ED junto a uma família de aplicações são consumidos pela EA através da instanciação de uma aplicação específica.

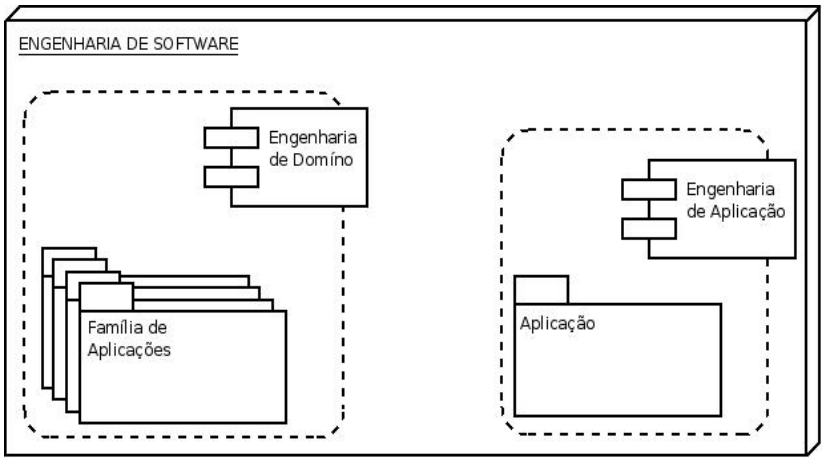


Figura 6 – Contexto de Engenharia de Aplicação

A ED busca viabilizar a reutilização pela identificação e modelagem de variabilidades e opcionalidades entre diversas aplicações de um domínio [Siy99]. As *variabilidades* refletem os elementos parametrizáveis em um domínio, de uma maneira abstrata, e devem ser configuráveis através de suas *variantes*, i.e., suas alternativas de configuração [Gim05]. A variabilidade é parte integrante dos artefatos em desenvolvimento, tal como prevê o Modelo de Features [Kan90]. As *opcionalidades* representam elementos que podem ou não pertencer a uma aplicação derivada de um domínio [Bac06].

Considerando esse fato, a ED aumenta seu potencial quando é empregada em organizações onde os produtos de software possuem características semelhantes, pois os processos tradicionais de análise e projeto não provêem métodos eficientes para capturar, e tirar vantagem destas similaridades. Nesse contexto, para viabilizar a reutilização, o Engenheiro de Domínio busca o conhecimento detalhado sobre as diversas aplicações existentes em um domínio para, a partir desta análise, definir rigorosamente uma Família de Produtos de Software (SPL) junto a esse domínio [Siy99]. Nesse sentido, podemos concluir que a ED procura, de maneira sistemática, identificar, construir, catalogar e disseminar artefatos de software que podem ser utilizados no futuro em outros softwares, num domínio particular de aplicações [Pre05].

3.2 Processos de Engenharia de Domínio

Em geral, as abordagens propostas definem uma série de atividades relacionadas à análise, projeto e implementação que geram artefatos reutilizáveis pela Engenharia de Aplicação. O esquema apresentado na Figura 7, mostra uma visão esquemática sobre a modelagem de um domínio.

Através do esquema apresentado, podemos observar que *n* aplicações instanciadas em um domínio específico, compartilham características comuns previamente identificadas e modeladas, através da execução de atividades previstas pela ED.

Um dos métodos de ED mais conhecidos é o FODA [Kan90]. O FODA (*Feature-Oriented Domain Analysis*) é o precursor dentre os métodos de ED, e foi proposto no SEI (*Software Engineering Institute*), onde foram iniciadas as primeiras pesquisas em reutilização de software e, em particular, em ED. Dentre os processos mais recentes, Odyssey-DE [Bra00], FORM (*Feature-Oriented Reuse Method*) [Kan02] e CBD-Arch-DE (*Component-Based Development - Architecture - Domain Engineering*) [Bac06], apresentam uma proposta de engenharia de domínio baseada em OO (Orientação a Objetos) e DBC (Desenvolvimento Baseado em Componentes).

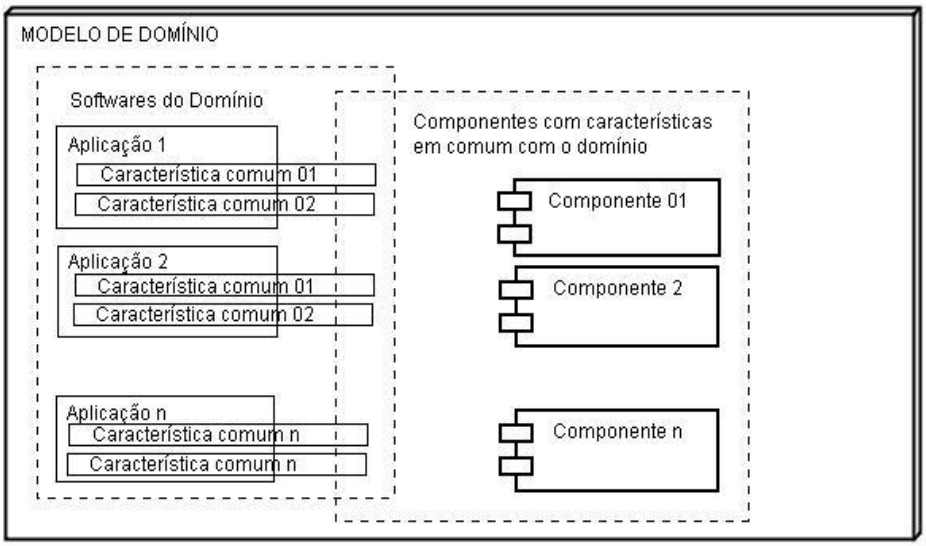


Figura 7 – Instanciação de Aplicação

Para o contexto dessa pesquisa, vamos analisar mais pontualmente a etapa de projeto de alguns destes métodos de ED, com o objetivo de avaliar como as abordagens existentes dão apoio à construção e documentação de uma Arquitetura de Referência para o domínio. Para tanto, foram identificados alguns critérios que conduzirão a análise dessas abordagens. Esses critérios foram baseados nos resultados obtidos pela aplicação de uma *survey*, conforme comentado na Seção 1.2. Nesse *survey*, procuramos identificar indícios de reúso na empresas, assim como as formas de reutilização de artefatos como arquiteturas de software (foco deste trabalho).

As questões propostas na *survey*, conforme Anexo D, foram elaboradas com base em um levantamento sobre Arquiteturas de Referência.

A avaliação da fase de projeto dos métodos de ED irá obedecer aos seguintes critérios:

I. Apoio a construção de uma Arquitetura de Referência:
   a) Suporte a Documentação: caracterizar que tipo de documentação é proposta pela abordagem de ED, como forma de registrar informações sobre os artefatos do domínio, rastros entre eles, as arquiteturas criadas para o domínio, e como esse registro é feito, e.g., documentação realizada através de descrição textual ou diagramas, seguindo ou não alguma padronização. Um exemplo de descrição é uma ADL.
   b) Suporte a Formalização: verificar nas abordagens avaliadas a existência de mecanismos dedicados a formalização das arquiteturas ou modelos criados para o domínio. Novamente, as ADLs possuem mecanismos de formalização, inerentes as linguagens, que poderiam também ser utilizados para formalizar arquiteturas de referência.
   c) Suporte Ferramental: avaliar, quando presentes, os ambientes que dão apoio ao processo de criação, formalização e documentação para ARs nas abordagens.
   d) Integração com um Processo de ED: verificar, qual o nível de integração demonstrado pelas abordagens, com relação às demais atividades de um processo de ED.

II. Apoio a Instanciação de Aplicações: a maioria dos métodos de ED utiliza modelos de *features* para apoiar a instanciação de aplicações. No entanto, *features* são abstrações de alto nível e que não necessariamente estão representando a Arquitetura de Referência do domínio construída durante a engenharia. Por outro lado, observa-se que uma Arquitetura de Referência, possivelmente documentada por meio de uma ADL, pode ser o artefato do domínio mais adequado para apoiar a instanciação de aplicações, pois a

arquitetura de uma aplicação deve ser consistente com a Arquitetura de Referência da qual foi gerada. Sendo assim, é importante observar se os métodos de ED e LP (Linha de Produto) utilizam a semântica de uma Arquitetura de Referência durante o processo de instanciação de uma nova aplicação.

3.2.1 FODA

O método FODA (*Feature-Oriented Domain Analysis*) [Kan90] é um dos mais conhecidos métodos de Engenharia de Domínio (ED). Este método procura apoiar o reúso em um nível funcional e arquitetural. A análise de domínio no FODA, conforme apresenta a Figura 8, prevê atividades relacionadas a análise de contexto, modelagem de domínio e modelagem da arquitetura.

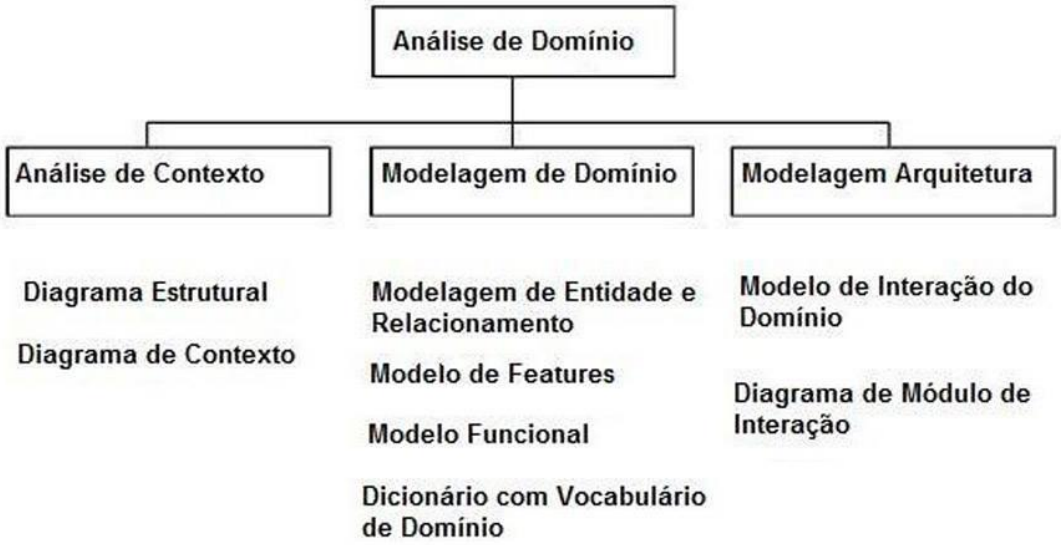


Figura 8 – Produtos e fases da Análise de Domínio [Kan90]

Embora exista uma etapa de modelagem da arquitetura no FODA, não fica evidente a criação de uma Arquitetura de Referência do domínio, tampouco algum processo de formalização ou documentação. Devido a esta limitação, o método

também não prevê ferramental de apoio à construção de AR e tampouco suporte a instanciação de aplicações por meio de AR.

3.2.2 FORM

Originado a partir do FODA [Kan90], o FORM (*Feature-Oriented Reuse Method*) [Kan05] é um método que apresenta ênfase na construção de Linha de Produto de Software (LPS) baseada em componentes. Da mesma forma que o seu precursor, apresenta como ponto forte às atividades ligadas a análise de domínio, e mesmo que ainda não existam casos relatando sua utilização prática, a literatura descreve alguma melhora do apoio ao ciclo completo de Engenharia de Domínio, comparado ao método FODA [Wer05] [Bac06]. Este método atende parcialmente a etapa de análise de domínio, uma vez que propõe um conjunto de classes derivadas das *features* do domínio. Embora os autores indiquem a necessidade de criação de uma arquitetura do domínio, na prática, não existe um apoio efetivo a sua construção.

3.2.3 Processo de ED com Foco em Projeto Arquitetural: CBD-Arch-DE

O CBD-Arch-DE (Component-Based Development – Architecture – Domain Engineering) [Bac06] é um processo de ED com foco no projeto arquitetural baseado em componentes. Componentes, de acordo com o conceito apresentado na introdução deste trabalho, são elementos autocontidos que apresentam uma interface e um grau de reutilização definidos [Sam97]. Conforme pode ser observado na Figura 9, ele propõe quatro grandes fases, como necessárias para a sua execução: planejamento, análise, projeto e implementação. Dentre as fases propostas por [Bac06], a fase de Projeto é a que está diretamente ligada ao contexto desta dissertação, pois é nesta etapa que a Arquitetura para o domínio deve ser elaborada.

A fase de Projeto inclui atividades para a criação e agrupamento de componentes, além da criação de Arquiteturas de Referência para o domínio. Nessa fase, baseada

em características específicas do domínio, a atividade relacionada à criação de componentes, produz diversos tipos de componentes, juntamente com os artefatos da análise de domínio dos quais eles se originaram. A criação de componentes, segundo orienta o processo CBD-Arch-DE, prevê:

i. *Categorização dos Componentes do Domínio,*
ii. *Criação de Componentes de Negócio,*
iii. *Criação de Componentes de Processo,*
iv. *Criação de Componentes Utilitários,*
v. *Criação de Componentes de Infra-estrutura.*

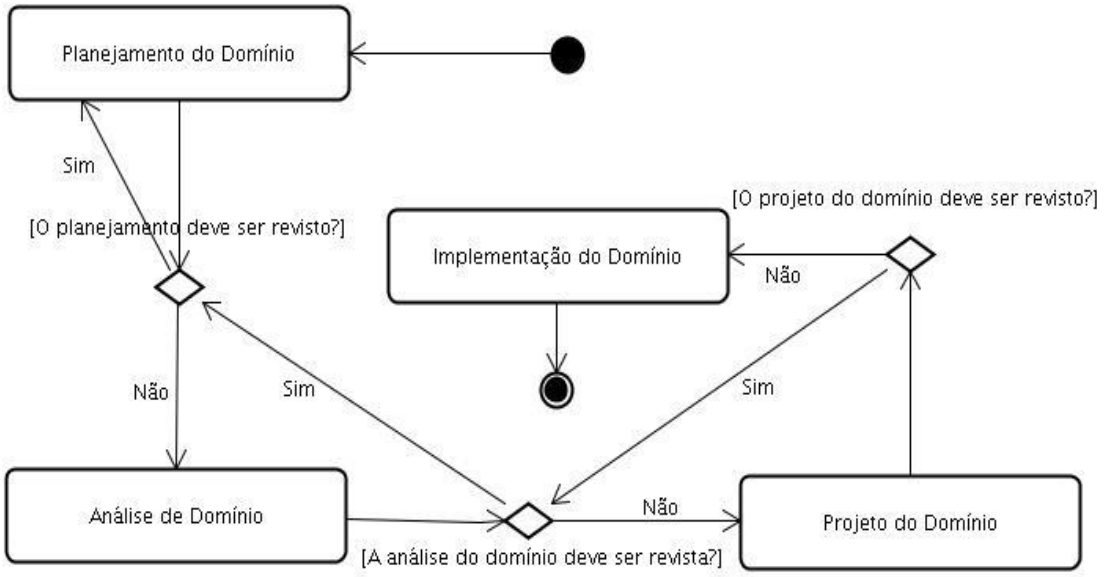


Figura 9 – Fases do Processo CBD-Arch-DE [Bac06]

Os componentes de negócio são equivalentes a classes compostas somente por atributos. Componentes de processo encapsulam regras de negócio, enquanto componentes utilitários encapsulam serviços genéricos. Componentes de infra-estrutura armazenam informações referentes à tecnologia do domínio. Conforme definido no processo, os componentes podem ser criados a partir de estilos arquiteturais, que utilizam os tipos de negócio e casos de uso para a criação de componentes do domínio e componentes de processo [Bac06].

Uma vez que os componentes e interfaces foram criados, é realizada a atividade de agrupamento de componentes, obedecendo a critérios definidos pela autora [Bac06]. Esses critérios consideram fortemente o acoplamento e coesão dos componentes, avaliado, dentre outros aspectos, a quantidade de mensagens trocadas entre componentes. Além disso, essa atividade procura sistematizar o grande volume de componentes gerados na fase anterior.

O processo CBD-Arch-DE sugere que os componentes sejam agrupados em um único elemento arquitetural, formado por um ou mais componentes, o qual pode representar um conjunto de requisitos do domínio que interagem entre si com maior incidência, quando comparados aos outros.

Na atividade de montagem da arquitetura, o projetista pode considerar o uso de estilos arquiteturais com o intuito de melhorar a organização da arquitetura em virtude dos requisitos funcionais e não-funcionais [Bac06]. Após a montagem da arquitetura baseada em componentes do domínio, espera-se como resultado, uma melhor organização dos componentes do domínio assim como uma compreensão da arquitetura do domínio baseado nos requisitos iniciais, e um aumento no potencial de reutilização desta arquitetura sob o ponto de vista da Engenharia de Aplicação.

Uma vez transcorridas todas as fases do processo CBD-Arch-DE, espera-se que o engenheiro tenha efetuado a modelagem do domínio de forma consistente em relação aos requisitos juntamente com suas variabilidades, opcionalidades e relacionamentos, bem como gerado possíveis Arquiteturas de Referência, além da codificação consistente em relação a estas arquiteturas e a tecnologia escolhida.

Embora o processo preveja uma atividade de criação de uma arquitetura para o domínio, na literatura disponível não são definidas atividades para atingir esse objetivo. Além disso, o processo não fornece mecanismos para quaisquer tipos de documentação relacionada à arquitetura do domínio, ou seja, a sua Arquitetura de Referência.

3.2.4 Construção de uma Arquitetura Baseada em Funcionalidades [Bos00]

O método apresentado nessa seção, não é considerado uma abordagem de Engenharia de Domínio, mas foi incorporado ao estudo por se tratar de uma abordagem para a criação de arquiteturas para domínios de aplicações.

Um projeto arquitetural é um processo de conversão de um grupo de requisitos para uma Arquitetura de Software (AS). O método proposto por [Bos00], tem como foco uma avaliação explícita de projeto orientado a qualidade de requisitos, e propõe a criação de uma Arquitetura de Referência baseada em requisitos funcionais. A construção desta arquitetura é feita em quatro passos distintos, conforme segue:

i. Definição do Contexto do Sistema;
ii. Identificação de Archetypes;
iii. Decomposição do Sistema em Componentes;
iv. Descrição das Instâncias do Sistema Utilizando Archetypes;

Dentre os quatro passos propostos por [Bos00] para construção de uma Arquitetura de Referência, apresentamos de forma mais detalhada apenas a segunda e terceira etapas, pois elas estão relacionadas diretamente ao foco desta dissertação. Esses passos compreendem a identificação de elementos e montagem da arquitetura. A seguir apresentaremos os passos para a criação de uma arquitetura baseada em funcionalidades:

3.2.4.1 Identificação de Archetypes e Decomposição do Sistema em Componentes

A modelagem da arquitetura, conforme definido por Bosh [Bos00], tem início com a identificação de abstrações do *core*, referenciadas aqui como *Archetypes*, sobre as quais o sistema será estruturado. Os *Archetypes* representam um pequeno grupo de entidades altamente abstratas que, quando combinadas, têm a capacidade de

descrever a maior parte dos comportamentos do sistema. Estas entidades podem ser instanciadas sob uma grande variedade de formas com o objetivo de popular o sistema, e a sua identificação está diretamente relacionada à *experiência* e criatividade *do arquiteto*, pois é necessário também o reconhecimento de padrões recorrentes em diversos sistemas.

Após a seleção de um grupo de elementos candidatos a *Archetypes*, é necessária uma análise para a seleção dos mais apropriados, em relação a criação de uma arquitetura para o domínio. Semelhante a atividade de agrupamento de componentes realizada em alguns métodos de ED, os *Archetypes* são agrupados de maneira a compor elementos arquiteturais que serão utilizados na elaboração da estrutura da Arquitetura (decomposição em componentes e as relações entre esses componentes).

A realização da decomposição do sistema em componentes não precisa ser, obrigatoriamente em um único nível, ela pode incorporar dois ou mais níveis recursivos de decomposição das partes críticas do sistema, de forma a tornar esse processo mais gradual.

Uma vez identificados os componentes em que a Arquitetura pode ser decomposta, é necessário estabelecer as relações entre esses componentes. Um cuidado deve ser tomado na hora de estabelecer as relações entre esses componentes, no sentido de preservar e manter as relações entre eles, situando-os na mesma camada de abstração, pois os componentes devem ser inter-relacionados entre si, de acordo com o nível de abstração que eles representam, e no caso de componentes que representem domínios ou entidades de domínio, vínculos para as ligações entre esse tipo de componente devem ser estabelecidos no modelo do domínio. É importante após essa fase, guardar o menor número possível de relações entre componentes, pois o contrário disso, níveis altos de conectividade entre componentes geralmente indicam que a decomposição da arquitetura tende a não possuir alta coesão e baixo acoplamento, princípios que podem ser indicativos de um potencial problema.

O processo descrito nesta seção apresenta preocupação com a geração e agrupamento de componentes, assim como observado nas outras abordagens de

Engenharia de Domínio analisadas. Além disso, observamos também uma grande preocupação do autor com o processo de montagem da arquitetura para o domínio, por exemplo, quando menciona a utilização de *Archetypes* passíveis de serem instanciados em diferentes aplicações. Esse detalhamento relativo a montagem da Arquitetura de Referência pode ser justificado pelo fato desta abordagem ser específica para estas atividades. Isso justificaria também que fosse provida uma boa documentação para as ARs criadas, o que na prática não acorre, fazendo assim com que esta seja uma grande limitação desta abordagem, uma vez que ARs são reutilizadas principalmente através de documentação (diagramas, descrição textual ou uma ADL).

3.3 Considerações

Conforme pôde ser observado ao longo deste capítulo no que diz respeito a construção de ARs pelos processos de ED analisados, percebemos que eles apresentaram uma preocupação considerável com a criação desse tipo de artefato. Contudo, embora deixem indicado em seus processos que existe a necessidade de criação de uma AR para o domínio, poucos, efetivamente, dão algum apoio através da definição de atividades específicas que permitam, pelo menos, a realização do agrupamento de componentes (passo importante para a criação de uma AR). Dessa forma, se estes processos não cobrem totalmente a criação de ARs, a documentação desse tipo de artefato apresenta menos apoio ainda.

Dentre as abordagens descritas neste capítulo, o processo CBD-Arch-DE foi o que apresentou maior cuidado com a elaboração de uma AR, fornecendo uma sistemática para a captura e agrupamento de componentes, obedecendo a critérios baseados em heurísticas pré-estabelecidas. No entanto, notamos que este processo, embora forneça um *background* interessante, não deixa claro como esses componentes devem ser organizados e dispostos na forma de camadas ou fazendo parte de um elemento arquitetural, de forma a compor uma AR. Em relação a instanciação de aplicações para um domínio específico, este processo prevê a instanciação baseada em *Features*. Modelos de *Features* está mais ligado aos

requisitos e *design* de alto nível, sendo necessárias outras etapas de refinamento deste modelo, até a criação de artefatos de mais baixo nível, mais apropriados a criação de arquiteturas.

De acordo com o que foi definido na Seção 3.2, onde foi descrito um protocolo utilizado como guia para realização da análise sobre os métodos de ED, vamos apresentar a seguir um comparativo sobre as capacidades dos métodos estudados, sob o ponto de vista do protocolo apresentado. A Tabela 1 apresenta os resultados apurados para cada método.

Tabela 1 – Critérios de Avaliação dos Métodos de ED para construção de AR

Atividade	**Critério Analisado**	**FODA**	**FORM**	**CBD-Arch-DE**	**Arq. Baseada em Funcionalidades**
Construção da AR	*Documentação*	Não	Não	Não	Não
	Formalização	Não	Não	Não	Não
	Ambiente de Apoio	Não	Não	Parcial	Não
	Integração com um Processo	Não	Não	Parcial	Não
Instanciação de Aplicações	*Criação de uma Arquitetura para uma Aplicação*	Features	Features	Features	Tipos Arquiteturais

Conforme pôde ser observado na Tabela 1, os métodos analisados não apresentaram apoio à atividade de elaboração de AR, tanto no que diz respeito a suporte a documentação, formalização, ambiente para a manipulação e integração com um processo de ED. Em relação ao apoio a instanciação de aplicações, os métodos fornecem algum suporte através de modelos de *Features*, que, conforme já comentado, não se constituem no artefato mais recomendado para esta finalidade, por tratarem da representação de conceitos do domínio, em um alto grau de abstração.

A *consequência* da falta de apoio na descrição de Arquiteturas de Referência geradas para um domínio, mesmo a partir de componentes gerados de forma eficiente por estes métodos, é a dificuldade no reúso dessa arquitetura pelo Engenheiro de Domínio no momento da instanciação de uma arquitetura física para uma aplicação.

Em função desta lacuna existente entre a descrição e o uso da Arquitetura de Referência foi desenvolvido este trabalho, com intuito de prover apoio ao Engenheiro de Domínio, para documentar de fato uma AR, facilitando a reutilização do domínio durante a Engenharia de Aplicação.

4 RADA: UMA ABORDAGEM PARA DOCUMENTAÇÃO DE ARQUITETURA DE REFERÊNCIA

Nesse capítulo é apresentada a abordagem proposta para a criação e documentação de Arquiteturas de Referência através da utilização de uma Linguagem de Descrição Arquitetural. Essa abordagem foi denominada RADA (*Reference Architecture for Domain Applications*), e ao longo das próximas seções são descritas de forma detalhada as etapas e atividades previstas pela abordagem.

4.1 Introdução

Conforme apresentado no Capítulo 2, as Arquiteturas de Software (AS) quando analisadas em um contexto de reúso, são conhecidas como Arquiteturas de Referência (AR) ou Arquiteturas para Domínios de Aplicações (ADA) [Aco06]. Estes artefatos são utilizados como base para a instanciação de arquiteturas para aplicações específicas de domínio [Aco06]. Ainda no capítulo referido, foram discutidas algumas estratégias para a documentação de AS [Cle00] [Men02] [Var02]. Nesse contexto, foram apresentadas as Linguagens de Descrição Arquiteturais (ADL) [Men02], dentre outras formas de documentação de ASs. Todavia, as ADLs, mesmo não tendo sido inicialmente concebidas para a documentação de ADA, apresentam um potencial a ser considerado, quando observada a sua aplicação para ASs.

Nesse sentido, de acordo com o que pode ser observado na literatura [Kou95] [Gar97] [Men00] [Har04], foram propostas algumas linguagens para a descrição de ADA, mas que, embora tenham representado um avanço nessa área, deixaram ainda muitos pontos em aberto, principalmente quando consideramos a representação de ADA para domínios modelados em termos de *Features* [Har04] [Bas05]. No Capítulo 3, onde realizamos uma análise sobre alguns métodos de Engenharia de Domínio (ED), observamos algumas lacunas deixadas por estes métodos, principalmente no que diz

respeito ao apoio a documentação de ADA e a instanciação de aplicações específicas de domínio.

Baseados neste cenário, esse capítulo apresenta uma abordagem para a descrição de ARs, denominada RADA (*Reference Architecture for Domain Applications*), utilizando uma ADL específica e fornecendo apoio ao Engenheiro de Domínio na atividade de documentação de uma AR, a qual é integrada as demais atividades de um processo de ED. A abordagem RADA também apóia a atividade de instanciação de uma aplicação para um domínio específico, junto a Engenharia de Aplicações.

Ao longo desse capítulo, apresentamos a abordagem de forma detalhada. Conforme será visto ao longo das próximas seções, a abordagem foi dividida em três etapas. Para cada etapa foram previstas atividades a serem realizadas de acordo com o objetivo de cada uma delas. A primeira fase dessa abordagem é baseada na etapa de Projeto do processo de Engenharia de Domínio (ED) CBD-Arch-DE [Bac06], o qual foi apresentado no Capítulo 2. A abordagem aqui apresentada realiza uma *extensão* deste processo, no sentido de complementar a sua fase de *Projeto*, fornecendo suporte a atividade de descrição de Arquiteturas de Referência, a qual não se encontra devidamente apoiada.

Conforme pode ser observado na Figura 10, a *fase de Projeto do Domínio* deste processo de ED apresenta suporte apenas às atividades de criação e agrupamento de componentes, deixando sem apoio a documentação das ARs.

De acordo com o esquema mostrado nessa figura, e detalhado ao longo desse capítulo, a abordagem propõe o uso de um *Template de AR*, estruturado em camadas, para organizar hierarquicamente os agrupamentos gerados pelo processo de ED. Da mesma forma, também é proposta a utilização de uma Linguagem de Descrição Arquitetural, com algumas adaptações, para realizar a documentação desta AR.

Como parte dos objetivos a serem alcançados por nossa abordagem, além da descrição de componentes, elementos arquiteturais e demais propriedades inerentes a uma Arquitetura de Referência, está à manutenção das informações sobre rastreabilidade. A rastreabilidade aqui tratada diz respeito a se manter o vínculo entre

os componentes levados para a Arquitetura de Referência e os casos de uso, *features* e tipos de negócio a partir dos quais eles foram criados.

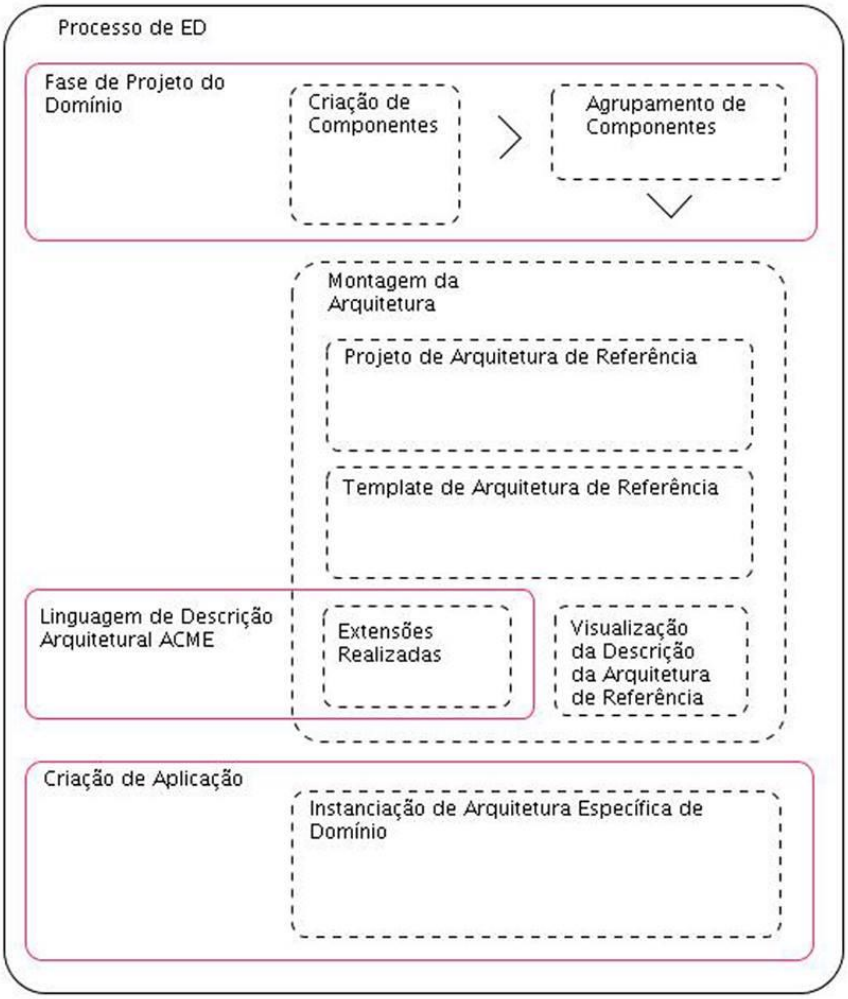


Figura 10 - Contextualização da abordagem proposta

Ao longo das próximas seções é detalhada a estrutura da abordagem RADA, as adaptações realizadas na ADL utilizada e no processo CBD-Arch-DE, e um detalhamento das etapas e atividades previstas dentro de cada etapa. Finalmente, são apresentadas algumas considerações sobre o capítulo.

4.2 Organização da Abordagem RADA

A abordagem proposta é baseada na execução de uma sequência de atividades, as quais foram distribuídas em três etapas. Essas etapas, conforme pode ser observado na Figura 11, foram planejadas para serem executadas dentro de um processo de Engenharia Domínio.

O objetivo das atividades propostas em cada uma das etapas é de apoiar o Engenheiro de Domínio no momento da documentação de uma Arquitetura de Referência, sendo que esta Arquitetura deve se basear em componentes gerados por um processo de ED, nesse caso, o CBD-Arch-DE [Bac06].

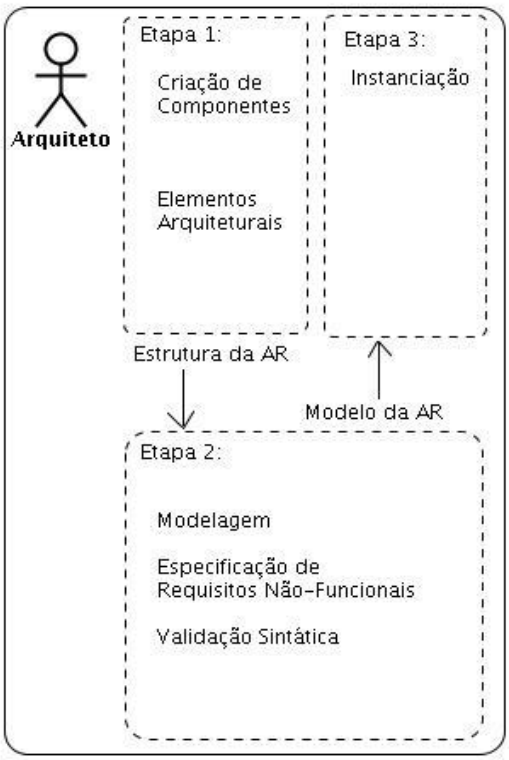


Figura 11 – Esquema Geral da Abordagem RADA

Basicamente, através das atividades da etapa 1 é possível gerar a estrutura da Arquitetura de Referência, a qual será documentada, na etapa 2, por meio da ADL ACME [Gar97]. Esta ADL estendida será utilizada na etapa 3, pelo arquiteto de

software, durante a instanciação de uma nova aplicação. Detalhes sobre as etapas e suas respectivas atividades são apresentados nas próximas seções.

4.3 Linguagem de Descrição Arquitetural ACME: Adaptações realizadas

A extensão realizada na ADL ACME se baseou em um mecanismo provido por esta linguagem justamente para esta finalidade, ou seja, introduzir na linguagem elementos externos de maneira a ampliar a sua capacidade de representação semântica. Este mecanismo permite realizar alterações na linguagem ampliando sua capacidade de representação, sem, no entanto, promover alterações em sua estrutura básica. Isso faz com que ela se torne bastante flexível, permitindo até mesmo a inclusão de outra linguagem dentro de ACME [Gar97]. Essa forma de extensão é menos invasiva, pois não altera o grupo de palavras-chave original da linguagem. Uma das grandes vantagens de se modificar ACME dessa forma, é que não perdemos o suporte a validação sintática e a notação gráfica fornecidas. No entanto, esta estratégia somente é válida para modificações feitas através de seu mecanismo de *property values*, o qual será apresentado em detalhes ao longo dessa seção.

Como ACME foi proposta para descrever Arquiteturas de Software (AS), e o objetivo do nosso trabalho é descrever Arquiteturas de Referência (AR), foi necessário modificar o comportamento padrão da linguagem, para permitir a representação de propriedades inerentes a AR. Para atingir esse objetivo, criamos um grupo de propriedades, tirando proveito do mecanismo de *property values* disponibilizado pela linguagem. Este novo grupo de propriedades criadas foi inserido no contexto da linguagem, conforme pode ser observado na Tabela 2, e têm como função:

i. Associar componentes a elementos arquiteturais;
ii. Registrar a necessidade de suporte a requisitos não-funcionais para uma dada arquitetura;
iii. Associar elementos arquiteturais a camadas da arquitetura;
iv. Representar informações inerentes a ARs como variabilidade e opcionalidade; e

v. Registrar informações necessárias a manutenção da rastreabilidade entre os artefatos do domínio (e.g., casos de uso, *features* e tipos de negócio no processo CBD-Arch-DE) e a AR.

Tabela 2 – Elementos criados pela extensão realizada

```
Property aTier : string = "CamadaNegocio";

Property aElementType : string = "ElementoArquiteturalSeguranca";

Property aVariability : string = "Invariante/Variante";

Property aMandatory : boolean = true;

Property aUseCases : string = "{'Caso_de_Uso_01', 'Caso_de_Uso_02'}";

Property aFeatures : string = "{'Features_01', 'Feature_02'}";

Property aBusinessTypes : string = "{'Tipo_de_Negocio_01',
'Tipo_de_Negocio_02'}";

Property aFailToler: string = "atende/nao-atende";

Property aSecurity: string = "atende/nao-atende";

Property aPerformance: string = "atende/nao-atende";

Property aAvailability: string = "atende/nao-atende";

Property aScalability: string = "atende/nao-atende";
```

As propriedades *aFailToler*, *aSecurity*, *aPerformance*, *aAvailability* e *aScalability* estão relacionadas ao registro dos requisitos não-funcionais Tolerância a Falhas, Segurança, Performance, Disponibilidade e Escalabilidade, respectivamente. Este conjunto de propriedades é bastante flexível, e pode ser estendido a qualquer momento. Esta extensão é feita através da inclusão de novas propriedades destinadas a representação de outros requisitos não-funcionais, de acordo com as necessidades de projeto. O objetivo da inclusão destas propriedades é deixar indicado, quais requisitos não-funcionais são contemplados por uma determinada AR, sem entrar no detalhe do apoio necessário para a sua cobertura. Por exemplo, a propriedade para o requisito não-funcional segurança (*aSecurity*) não especifica qual tipo de algoritmo é

utilizado para criptografia de dados, cabendo então na engenharia de aplicação, efetuar as devidas escolhas.

As propriedades *aTier* e *aElementType*, são utilizadas para a representação da estrutura hierárquica da Arquitetura de Referência. Através do uso de *aElementType*, por exemplo, é possível descrever as associações entre elementos e camadas. Da mesma forma, o uso de *aTier*, permite a associação de Elementos Arquiteturais as suas respectivas camadas. As propriedades *aVariability* e *aMandatory*, são utilizadas para a representação da variabilidade e opcionalidade. No caso da variabilidade, essa propriedade vai documentar se um determinado componente é uma variação, ponto de variação ou invariante. Já para a opcionalidade, a propriedade vai registrar se o componente é opcional ou mandatório.

As propriedades *aUseCases*, *aFeatures* e *aBusinessTypes* estão relacionadas a manutenção da rastreabilidade entre os Elementos Arquiteturais que compõem a arquitetura, e os artefatos dos quais eles se originam. Tal rastro é importante, pois dá visão ao Arquiteto sobre o impacto causado na arquitetura, face ao atendimento de novos requisitos ou alterações em requisitos existentes. As propriedades acima anteriormente mencionadas registram as seguintes informações relacionadas ao componente: *aUseCases*, registra uma lista de casos de uso, *aFeatures*, registra uma lista de *features* e *aBusinessTypes*, uma lista de tipos de negócio.

Analisando sob o ponto de vista de suporte de ambiente, estas propriedades podem ser manipuladas pelo Arquiteto ou Engenheiro de Domínio no ambiente ABLE [Dav09], o qual será detalhado no próximo capítulo. Esse ambiente fornece, dentre outras funcionalidades, um compilador para validação sintática da descrição gerada para uma Arquitetura de Referência, assim como editores gráficos para a manipulação de propriedades e demais elementos pertencentes ao grupo de constructos de ACME.

A utilização do ambiente ABLE, assim como da linguagem ACME se constitui em uma vantagem, uma vez que não precisamos nos preocupar com questões relacionadas a validação sintática da Linguagem de Descrição Arquitetural gerada para uma AR. Contudo, isto só foi possível devido à estratégia de extensão realizada, que de maneira pouco invasiva, não alterou o grupo original de construtores da linguagem,

preservando assim o suporte do ambiente. Dessa forma foi possível tirar proveito das funcionalidades de ABLE, para a realização da validação, dentre outras coisas, de descrições geradas para ARs.

4.4 Primeira Etapa da RADA

Esta etapa visa identificar os componentes utilizados na construção da AR. Nessa etapa, conforme pode ser observado na Figura 12, os componentes gerados durante o processo de ED são agrupados em elementos arquiteturais, e estes elementos por sua vez, são distribuídos entre as diversas camadas da arquitetura. A seguir serão detalhadas as atividades previstas dentro desta primeira etapa:

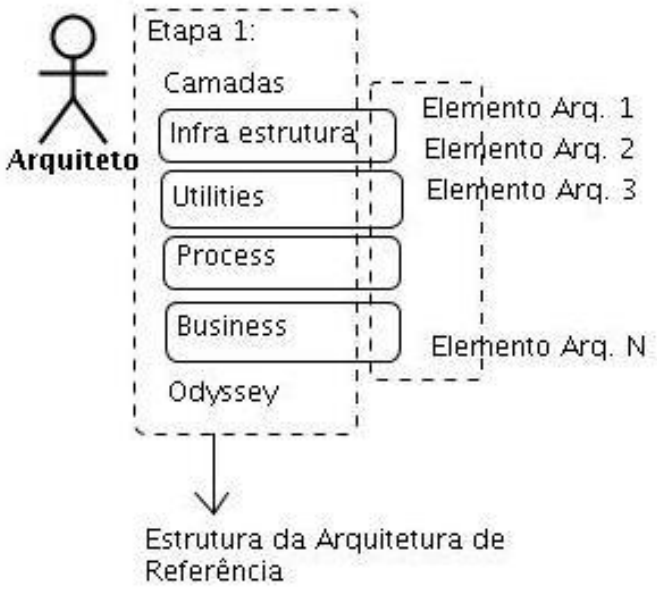


Figura 12 – Etapa 1 da Abordagem RADA

4.4.1 Identificação de componentes

Essa atividade tem como objetivo identificar os componentes utilizados na criação da AR bem como manter os rastros destes componentes com seus respectivos casos

de uso, tipos de negócio e *features*. Estes rastros, documentados com o apoio das propriedades *aUseCases*, *aFeatures* e *aBusinessTypes*, apresentadas na Seção 4.3, permitem uma análise futura sobre o impacto sofrido pela Arquitetura frente as mudanças causadas pela cobertura de novos requisitos, ou alterações em requisitos existentes.

Esta etapa da abordagem, e mais especificamente esta atividade, é baseada no processo CBD-Arch-DE [Bac06], que de acordo com o previsto em sua etapa de projeto, gera componentes em quatro agrupamentos lógicos, i.e., componentes de Processo, Negócio, Infra-estrutura e Utilitários, detalhados na seção 3.2.3. Estas informações sobre a categorização de componentes, junto com o apoio das heurísticas fornecidas pelo processo de ED, contribuem para à tomada de decisões sobre o agrupamento de componentes em elementos arquiteturais e distribuição de elementos arquiteturais em camadas, a qual é realizada nesta abordagem na próxima atividade.

4.4.2 Agrupamento de componentes em um Elemento Arquitetural

A atividade de agrupamento de componentes em um elemento arquitetural tem como objetivo promover a organização dos componentes em elementos com uma granularidade maior, e em um nível de abstração mais alto. Nesse ponto, a decisão sobre quais componentes serão agrupados em um determinado elemento, é feita com base em heurísticas [Bac06] pelo processo CBD-Arch-DE. Estas heurísticas consideram detalhes como a quantidade de mensagens trocadas entre componentes, número de interfaces comuns, dentre outros critérios, para propor um determinado agrupamento. Baseado nos agrupamentos gerados, nossa abordagem sugere que todos os grupos criados sejam documentados pela AR, e instanciados ou não pelo Engenheiro de uma determinada aplicação, considerando os requisitos a ser reutilizados.

4.4.3 Distribuição dos Elementos Arquiteturais em Camadas

Conforme afirma Bacelo [Bac06], em geral os métodos de DBC [Dso99] propostos para arquiteturas de referência baseadas em componentes, sugerem o uso do estilo arquitetural em camadas. Neste caso, os componentes, que representam diferentes abstrações do domínio (e.g., negócio, processo, infra-estrutura, utilitário), são distribuídos pelas camadas da arquitetura, sugerindo uma organização dos componentes de acordo com o serviço que eles oferecem.

Sendo assim, nesta abordagem, uma vez identificados os componentes que representam os conceitos mais importantes para o domínio, estes serão devidamente categorizados e agrupados em elementos arquiteturais que serão distribuídos nas suas respectivas camadas da AR.

A distribuição dos elementos entre camadas exige intervenção e conhecimento do Arquiteto sobre o domínio. Nessa atividade o Arquiteto distribuirá os elementos arquiteturais criados na atividade anterior através das 4 camadas da AR: Infra-estrutura, Negócio, Processo e Utilitário. Nesse ponto, como forma de apoiar o Arquiteto a realizar a distribuição dos componentes, é facultado a ele que observe o tipo de componente inserido em cada elemento arquitetural, para inferir em qual camada pode ser inserido, em consonância com os resultados sugeridos pelos critérios de agrupamento [Bac06].

4.5 Segunda Etapa da abordagem RADA

A segunda etapa, conforme pode ser observado na Figura 13, está relacionada a modelagem e validação da descrição gerada para a AR.

Nessa segunda etapa, como forma de prover suporte básico a edição da linguagem de descrição gerada para uma AR, a RADA conta com o apoio do ambiente ABLE. Uma funcionalidade importante fornecida pelo ABLE é um compilador utilizado para validação sintática da linguagem gerada para uma determinada arquitetura.

Nesse ambiente é realizada a modelagem da AR através da interligação de diversos elementos arquiteturais intra e extra camadas da AR.

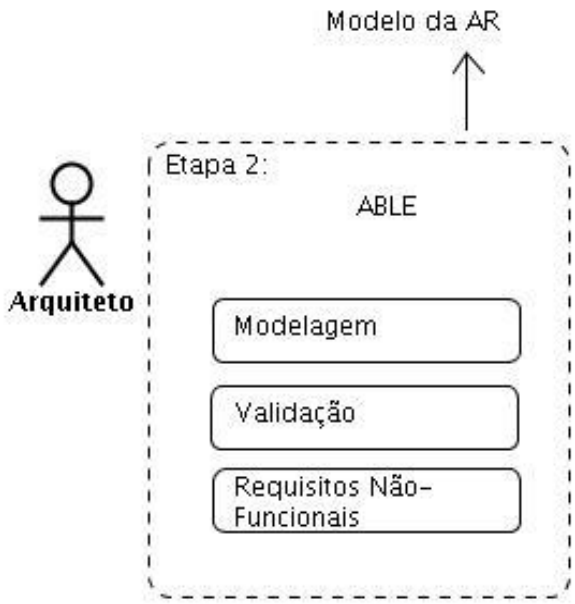


Figura 13 – Etapa 2 da Abordagem RADA

Neste mesmo ambiente também são registrados os requisitos não-funcionais e propriedades inerentes a variabilidade e opcionalidades da AR em construção. A seguir são apresentadas as atividades previstas para a segunda etapa da RADA.

4.5.1 Modelagem da Arquitetura através da ADL estendida

O primeiro passo a ser executado nessa atividade é a montagem da arquitetura com base no *Template* de Arquitetura de Referência proposto. Esse *Template*, baseado no Estilo Arquitetural Camadas, é formado por 4 camadas: *Infrastructure*, *Utilities*, *Process* e *Business*. Nesse momento, o Arquiteto, após realizar o agrupamento dos componentes em Elementos Arquiteturais na Etapa 1, distribui os elementos arquiteturais (contendo componentes agrupados), entre as 4 camadas da arquitetura. Após a distribuição dos elementos arquiteturais nas suas respectivas camadas, é feita a codificação da AR utilizando a ACME estendida. Esta codificação

inclui a documentação das camadas da arquitetura, seus respectivos elementos arquiteturais e componentes. Além disso, durante essa codificação, cada elemento arquitetural traz consigo informações sobre cada um de seus componentes, bem como sobre suas respectivas variabilidades, opcionalidades e rastros.

A etapa de modelagem da AR requer atenção especial do Arquiteto ou Engenheiro de Domínio, pois se trata de um processo de criação o qual não é totalmente automatizado. Sendo assim, cabe ao arquiteto realizar os ajustes necessários como forma de realizar um refinamento da AR em desenvolvimento. Estes ajustes podem variar desde a inclusão ou exclusão de relacionamentos entre elementos arquiteturais, até a alteração ou inclusão de componentes ou de propriedades, e edição de propriedades ligadas à variabilidade e opcionalidade. É importante destacar que embora se sugira o uso do ambiente ABLE para a edição da linguagem, esse trabalho pode ser feito em qualquer editor de arquivos do tipo texto. Contudo, o ambiente ABLE possui uma interface gráfica que auxilia significativamente a execução da modelagem da AR, além de fornecer uma mecanismo de validação sintática da AR gerada.

4.5.2 Indicação de Requisitos Não-funcionais

O objetivo dessa atividade é fornecer suporte à descrição em alto nível, junto a AR, de requisitos não-funcionais esperados pela arquitetura. Essa atividade não visa dar suporte automatizado ou semi-automatizado a requisitos não-funcionais, mas sim deixar um registro para que durante a engenharia de aplicação, tais requisitos não-funcionais venham a ser previstos por uma nova aplicação do domínio. A indicação destes requisitos não-funcionais na AR desta abordagem é documentada através de um grupo de propriedades da linguagem estendida, o qual foi descrito na Seção 4.3.

4.5.3 Validação Sintática da Arquitetura de Referência

Essa atividade tem como objetivo realizar uma validação sintática da linguagem gerada para a representação de uma AR. Essa verificação é feita ao mesmo tempo em que a modelagem é realizada. Qualquer inconsistência (e.g. uso de um construtor inapropriado) identificada na linguagem produzida é indicada através de mensagens de erro emitidas pelo ambiente de apoio. Estas inconsistências estão relacionadas a erros de sintaxe. Assim sendo, como esta atividade é bastante dependente da funcionalidade fornecida pelo ambiente ABLE, maiores detalhes serão apresentados na Seção 5.4, do capítulo de implementação da abordagem.

4.6 Terceira Etapa

A terceira etapa desta abordagem, conforme representado na Figura 14, está relacionada a instanciação de aplicações, apoiando-se na documentação criada nos passos anteriores.

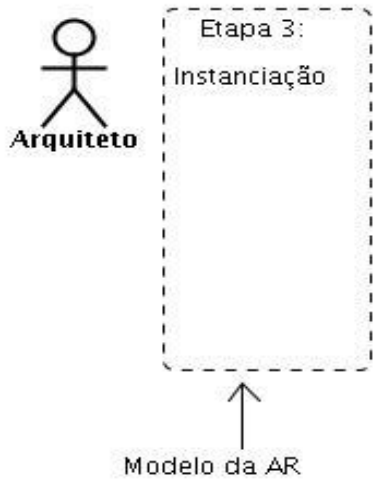


Figura 14 – Etapa 3 da Abordagem RADA

Para tanto, inicialmente é realizada uma simplificação da descrição da AR (criada na Etapa 2), partindo do formato codificado obedecendo a sintaxe da ADL utilizada

como base, para um formato baseado em uma visão hierárquica dos elementos da AR (mais acessível).

Posteriormente, essa etapa prevê a execução de outra atividade, a qual está relacionada a utilização deste documento de apoio, juntamente com a AR criada com base no *Template* de AR (organizado em camadas), para realizar a instanciação de uma aplicação específica.

4.6.1 Transformação da Descrição da AR para uma Visão em Camadas

Após a execução da segunda etapa de nossa abordagem, temos configurado e devidamente registradas todas as informações necessárias para a representação de uma AR. Estas informações ficam armazenadas em um arquivo, o qual tem uma estrutura semelhante a apresentada no Apêndice M e ilustrada resumidamente na Figura 15.

O formalismo desta AR gerado pela ACME estendida é interessante sob o ponto de vista de linguagem de codificação, pois favorece a realização da validação sintática. No entanto, sob o ponto de vista de utilização desta linguagem como documento de apoio a compreensão da organização da arquitetura, ela não é totalmente prática. Conforme Apêndice M e Figura 15, se percebe que os elementos são ligados, basicamente, através de propriedades (atributos e valores), que relacionam objetos (componentes) a níveis da arquitetura (elementos arquiteturais ou camadas). Dessa forma, essa atividade realiza uma transformação da descrição gerada no formato da ADL, para um formato simplificado. A partir deste novo formato é possível obter uma documentação da AR mais acessível pelos interessados do domínio (pressuposto básico das ADLs [Gar97]), dos quais não se espera o mesmo domínio sobre a sintaxe da ADL, quando comparados ao Arquiteto do Domínio.

Assim, esta transformação realizada pelo Arquiteto, consiste apenas na reorganização dos elementos da ADL, de forma a aninhar dentro de uma camada todos os seus elementos arquiteturais, e também para cada elemento arquitetural aninhar todos os seus componentes.

```
//-------------------------------------------------------------------------------------------
//----
//---- Architectural Description Language to Reference Architecture Documentation
//----
//---- Dominio: Telefonia Celular
//---- Data: Wed Jan 20 01:02:35 BRST 2010
//-------------------------------------------------------------------------------------------

import $AS_GLOBAL_PATH/families/ThreeTieredFam.acme;
Family OdysseyFamily extends ThreeTieredFam with {

    // Architectural Properties
    Property aFailToler: string = "True";
    Property aSecurity: string = "True";
    Property aPerformance: string = "False";
    Property aAvailability: string = "True";
    Property aScalability: string = "True";

    // Infrastructure-Tier:  Reference Architecture Layer
    // Used to group architectural elements
    Group Type Infrastructure-Tier = {

    }

    // ElemArquit-Monitoramento:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type ElemArquit-Monitoramento = {
        Property aTier : string = "Infrastructure-Tier";
    }

    // Criptografia:  Reference Architecture Component
    // Used to group architectural properties from reference architecture
    Component Type Criptografia = {
        Property aElementType : string = "ElemArquit-Monitoramento";
        Property aVariability : string = "Invariante";
        Property aMandatory : boolean = true;
        Property aUseCases : string = "{'Autenticar'}";
        Property aElementType : string = "ElemArquit-Seguranca";
        Property aFeatures : string = "{'FeatureMonitoramento', 'FeatureControle'}";
        Property aBusinessTypes : string = "{'AlgoritmoCript1', 'AlogoritmoCript2'}";
    }
}
```

Figura 15 – Parte de uma Descrição de uma AR usando a ACME estendida

A Figura 16 apresenta um exemplo de visualização em camadas dos elementos da linguagem.

Conforme pode ser notado, esta forma de apresentação dos elementos é mais acessível, e facilita a disseminação do conhecimento sobre a AR do domínio entre os envolvidos no projeto.

```
Property aFailToler: string = "True";
Property aSecurity: string = "True";
Property aPerformance: string = "False";
Property aAvailability: string = "True";
Property aScalability: string = "True";

Group Type Infrastructure {

    Element Type ElemArquit-Monitoramento {

        Component Type Criptografia {
            Property aVariability : string = "Invariante";
            Property aMandatory : boolean = true;
            Property aUseCases : string = "{'Autenticar'}";
            Property aFeatures : string = "{'FeatureMonitoramento', 'FeatureControle'}";
            Property aBusinessTypes : string = "{'AlgoritmoCript1', 'AlogoritmoCript2'}";

        }
        Component Type MonitorServico {
            Property aVariability : string = "Invariante";
            Property aMandatory : boolean = true;
            Property aUseCases : string = "NA";

        }
    }
}
```

Figura 16 – Visão em Camadas da ADL para Instanciação

Isto é viabilizado porque a sua interpretação não requer um aprendizado da sintaxe da linguagem comparado com a mesma descrição no formato da Linguagem de Descrição Arquitetural.

4.6.2 Instanciação de uma Aplicação

A atividade relacionada a instanciação de aplicações, tem como objetivo apoiar a Engenharia de Aplicação (EA) no processo de instanciação de aplicações para um domínio específico.

Nessa atividade, o Arquiteto conta com o apoio de uma documentação (descrevendo de forma detalhada a AR do domínio – Figura 16), e uma AR (organizada segundo um Estilo Arquitetural em Camadas). A partir destes dois artefatos, produzidos pelas duas etapas anteriores, pode ser criada uma nova Aplicação para o domínio.

Diferentemente da maioria dos processo de ED, incluindo o próprio CBD-Arch-DE [Bac06], que usam o modelo de *Features* para instanciar aplicações, esta abordagem utiliza toda a documentação da AR gerada, e a própria arquitetura construída a partir do estilo em camadas, para apoiar o processo de instanciação. Esta proposta se fundamenta na riqueza semântica e confiabilidade (baseada na validação sintática) da AR obtida com o uso da ADL estendida, a qual não pode e não deve ser descartada durante um processo de instanciação de uma aplicação que depende totalmente, de uma semântica bem definida do domínio para ter um reúso de fato. Finalmente, pela experiência que se tem dos processos de ED, os modelos de *Features* geralmente não são versionados, e esta falta de atualização pode não mais refletir o que de fato é o domínio durante uma instanciação.

Para tanto, o Arquiteto seleciona, dentre os componentes disponíveis na AR, aqueles componentes que julgar necessários a realização dos requisitos impostos por esta nova aplicação. Como forma de determinar quais requisitos não-funcionais a AR disponibiliza, por exemplo, o Arquiteto pode consultar as propriedades relacionadas a indicação de requisitos não-funcionais na descrição textual. Além disso, através da documentação o arquiteto da aplicação pode avaliar as relações existentes entre

componentes, casos de uso e *features*, como subsídio para realizar efetivamente a instanciação de componentes.

4.7 Considerações

Ao longo deste capítulo, apresentamos a forma como foi estruturada a abordagem RADA que dá apoio a documentação de Arquiteturas de Referência através de uma Linguagem de Descrição Arquitetural e do seu uso para a instanciação de aplicações.

Inicialmente foi realizado um detalhamento da extensão proposta para a ADL utilizada como base para a descrição das arquiteturas. Esse detalhamento apresentou o grupo de propriedades criadas, e a utilidade de cada uma dentro do processo de documentação.

Além da apresentação das alterações feitas na linguagem, foi feita uma descrição detalhada das etapas e das atividades envolvidas na abordagem proposta, a qual visa estender a fase de projeto do processo CBD-Arch-DE, principalmente na atividade de criação de Arquitetura de Referência.

Ao longo do próximo capítulo, será apresentado o protótipo elaborado para apoio a abordagem RADA. A partir da construção desse protótipo, mesmo com algumas limitações, foi possível a realização de experimentos para a avaliação da RADA, conforme será apresentado no Capítulo 6.

5 IMPLEMENTAÇÃO DA ABORDAGEM RADA

Esse capítulo mostra a implementação realizada junto a um ambiente de reúso, das principais funcionalidades propostas pela abordagem RADA, apresentada no Capítulo 4. A implementação será descrita através de diagramas específicos para este tipo de documentação e com auxílio de telas explicando, além da implementação propriamente dita, a execução das atividades da abordagem com a ferramenta elaborada.

5.1 Introdução

Nesse capítulo vamos apresentar a implementação de um protótipo, desenvolvido com o objetivo de apoiar a execução da abordagem RADA (*Reference Architecture for Domain Applications*).

Conforme descrito no Capítulo 3, nossa proposta foi baseada no processo de Engenharia de Domínio (ED) CDB-Arch-DE (*Component-Based Development - Architecture - Domain Engineering*) [Bac06]. Este processo foi implementado em um ambiente de reúso denominado Odyssey [Wer09], o qual apresenta funcionalidades destinadas à execução de atividades definidas pelo CBD-Arch-DE. Estas atividades estão relacionadas a identificação e agrupamento de componentes. Por este motivo, assim como pela facilidade de acesso a documentação e ao código fonte desse ambiente, o Odyssey foi escolhido como ferramenta base para implementação das atividades propostas pela RADA.

Além do Odyssey, utilizado como ferramenta para a manipulação das descrições de Arquiteturas de Referência (AR) geradas, e.g., visualização, edição e validação sintática, foi também utilizado o ambiente ABLE (*Architecture-Based Languages and Environments*) [Dav09]. O ABLE, conforme detalhado ao longo desse capítulo, dá apoio à manipulação da sintaxe da Linguagem de Descrição Arquitetural (ADL) ACME (*Architectural Description Language Carnegie Mellon*) [Gar97].

A implementação realizada nessa dissertação, apresentada daqui em diante, foi realizada no sentido de ampliar o grupo de funcionalidades das ferramentas envolvidas na solução proposta. No caso do Odyssey, foram adicionados novos elementos semânticos para a representação de uma AR. No que diz respeito a ACME, a extensão realizada se deu através do mecanismo de *property value*, disponibilizado pela própria linguagem.

A seguir são apresentados os diagramas UML [Uml10] utilizados para modelar os requisitos definidos em função da abordagem, assim como um detalhamento da implementação feita no Odyssey. Ao final é apresentado um exemplo de utilização da abordagem RADA através do Odyssey, em conjunto com ABLE. Este exemplo abrange a criação de uma documentação de AR básica, seguida de uma instanciação de uma aplicação para um domínio específico.

5.2 Funcionalidades Implementadas para a Abordagem RADA

A abordagem RADA prevê a execução de uma sequência de atividades. Algumas destas atividades, baseadas no processo CBD-Arch-DE, já se encontram implementadas no Odyssey. Dentre as funcionalidades já implementadas no Odyssey, podemos destacar a geração de Casos de Uso e Tipos de Negócio, a partir de um Modelo de Features [Kan90], assim como a geração automatizada de componentes a partir desses dois artefatos. Também, sob a forma um *plugin* desenvolvido para o Odyssey, é disponibilizada uma ferramenta para o agrupamento de componentes em elementos arquiteturais, obedecendo a critérios baseados em acoplamento e coesão [Bac06].

As funcionalidades implementadas no Odyssey para a abordagem RADA partem deste contexto, e são apresentadas na Figura 17, através de um diagrama de caso de uso.

Como forma de caracterizar o fluxo de atividades necessárias à execução da abordagem, foi elaborado também um diagrama de atividades, apresentado na Figura

18. Nesse diagrama é representada a interação entre o Engenheiro de Domínio e as ferramentas Odyssey e ABLE.

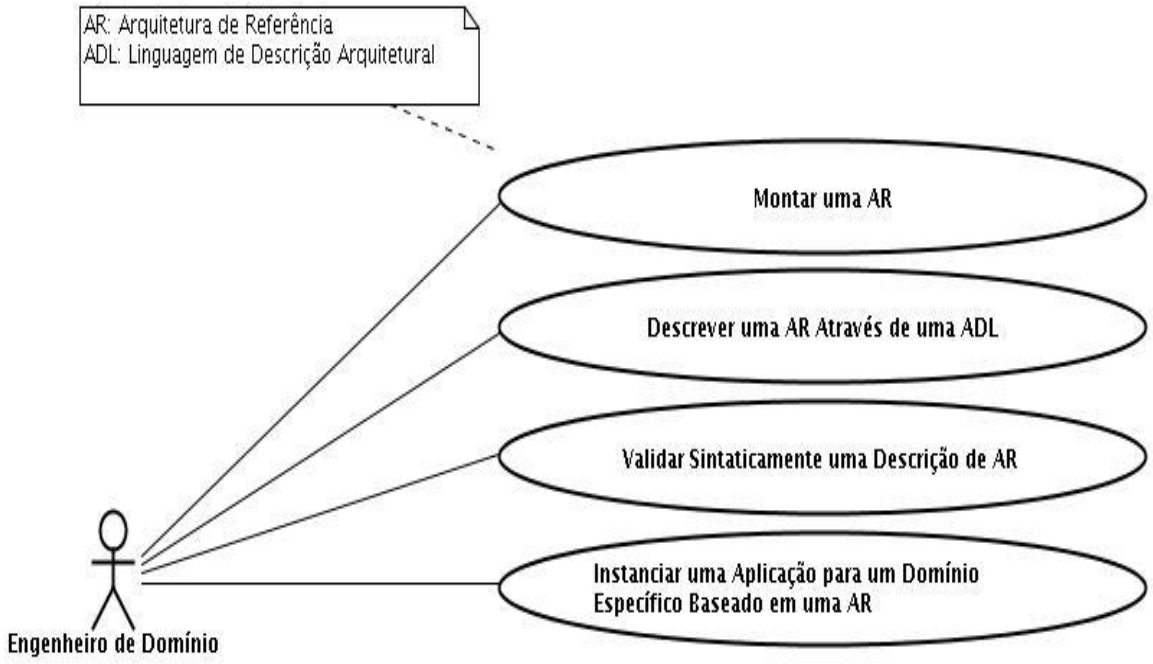


Figura 17 – Diagrama de Caso de Uso

Primeiramente, vamos definir um escopo inicial com o objetivo de explicar apenas as atividades relacionadas ao contexto da abordagem. Dessa forma, como ponto de partida para a execução das atividades vamos considerar a existência de um Modelo de *Features* já criado para o domínio. Da mesma forma, vamos considerar também que, a partir deste modelo, foram derivados artefatos representados por casos de uso e tipos de negócio, e que um grupo de componentes foi gerado com base nestes artefatos. Partindo destes componentes, o Engenheiro de Domínio realiza as atividades representadas no diagrama de atividades apresentado na Figura 18. Estas atividades compreendem desde o agrupamento de componentes, montagem e descrição da AR, até a instanciação de uma aplicação específica para o domínio.

A seguir são detalhadas as implementações inerentes a esta abordagem nos dois ambientes utilizados.

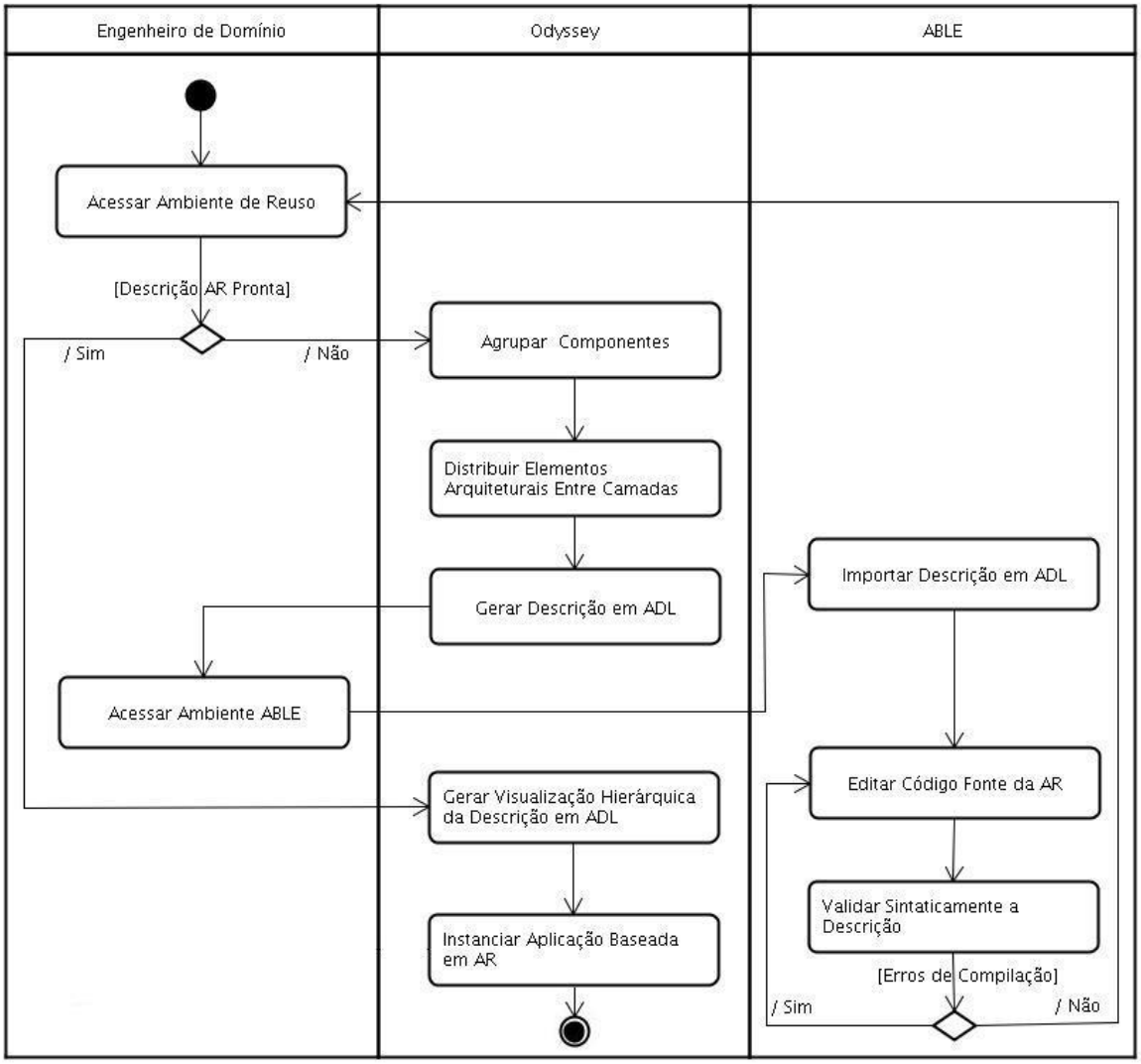


Figura 18 – Diagrama de Atividades

5.3 Odyssey – Engenharia de Domínio

5.3.1 Introdução

O Odyssey se constitui em um ambiente de reutilização estável, e que possui uma notação gráfica para a representação de variabilidade, baseada em Modelo de

Features [Kan90]. Um exemplo dos elementos que compõem a notação do Modelo de Features, assim como o seu significado semântico, pode ser observado no Anexo A. Este ambiente de reúso foi utilizado nessa dissertação, pois, além das vantagens descritas e de possuirmos livre acesso ao código fonte e documentação, ele já apresenta de forma integrada, suporte ao processo CBD-Arch-DE. A Figura 19 apresenta a IDE (*Integrated Development Environment*) *padrão,* disponibilizada pelo ambiente, *sem* as alterações realizadas pela abordagem.

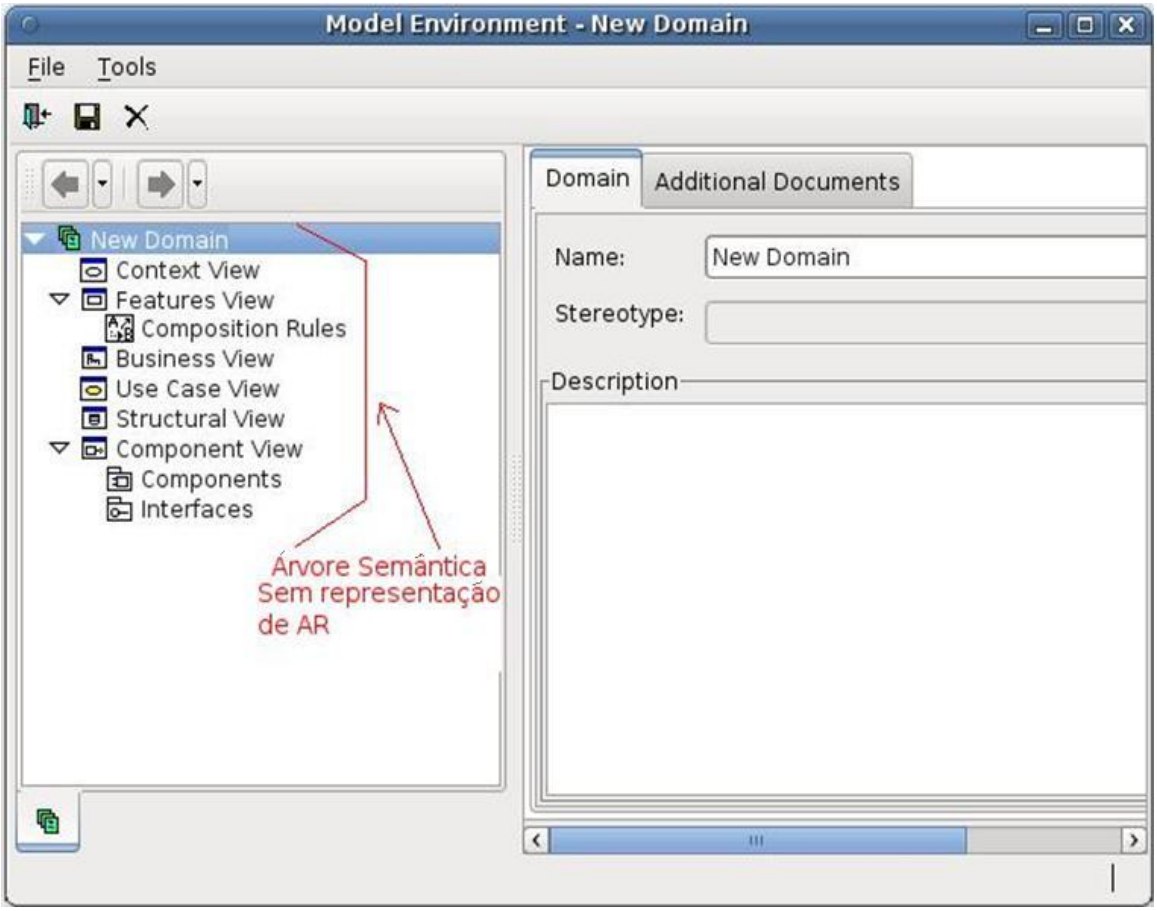


Figura 19 – Visão original do Odyssey sem alterações

Neste ambiente, são fornecidas ferramentas para a construção de domínios baseando-se em Modelos de Features. Além do processo CBD-Arch-DE, Odyssey têm implementado suporte a outros processos com foco no reúso de software, que podem

ter seus processos instanciados dentro do ambiente através de seus *Wizards*[3]. Outro ponto forte é o apoio a criação de diagramas no padrão UML, dentre os diagramas suportados podemos citar: diagrama de caso de uso, diagrama de classes e diagrama de componentes.

5.3.2 Extensões e Implementações Adicionais

As novas funcionalidades adicionadas ao Odyssey estão relacionadas aos seguintes pontos:

i. Representação do estilo arquitetural Camadas [Sha96];
ii. Representação de elementos arquiteturais;
iii. Representação de requisitos não-funcionais; e
iv. Criação de propriedades para a representação da variabilidade, e propriedades para o registro da rastreabilidade entre artefatos.

As extensões e as novas classes criadas no Odyssey para a implementação do suporte a criação de ARs, são apresentadas no diagrama de classes da Figura 20. Conforme observado nesse diagrama, as classes preenchidas com a cor cinza representam classes pré-existentes no ambiente de reúso, e foram destacadas por estarem mais fortemente relacionadas às principais entidades criadas no Odyssey para a nossa abordagem. Já as classes Camada, ElementoArquitetural e Componentes foram criadas através de herança, a partir da classe ModeloAbstrato. Isso foi necessário, basicamente, para aproveitar o suporte fornecido pela infra-estrutura do Odyssey, através de uma hierarquia de interfaces realizadas pelo ModeloAbstrato. Dessa forma, as funcionalidades herdadas de ModeloAbstrato permitiram a renderização dos elementos representados por estas entidades no sistema (Arquitetura de Referência, Camada, Elemento Arquitetural, Componente) na árvore semântica de

[3] Assistente ou Wizard é um padrão de projeto de software amplamente utilizado em interface gráfica do usuário para prover um meio simples de realizar tarefas complexas em sistemas computacionais, através de um esquema passo-a-passo. Em alguns softwares de código aberto é conhecido como Druida [Wik10d].

elementos. Essa estratégia de implementação também proporcionou que nossas entidades, além de serem desenhadas na árvore, polimorficamente, pudessem também trocar mensagens com outros elementos semânticos existentes. Essa troca de mensagens, por sua vez, viabiliza a extração de informações sobre associações existentes entre artefatos para a documentação de rastros, além do registro de propriedades inerentes a variabilidade, fornecidas pelo Modelo de Features.

O grupo de entidades apresentado no diagrama da estrutura da aplicação da Figura 20 representa apenas um pequeno subconjunto de todas as entidades criadas por nossa abordagem dentro do Odyssey, o qual era composto, até a finalização da implementação realizada para esta dissertação, por cerca de 800 classes.

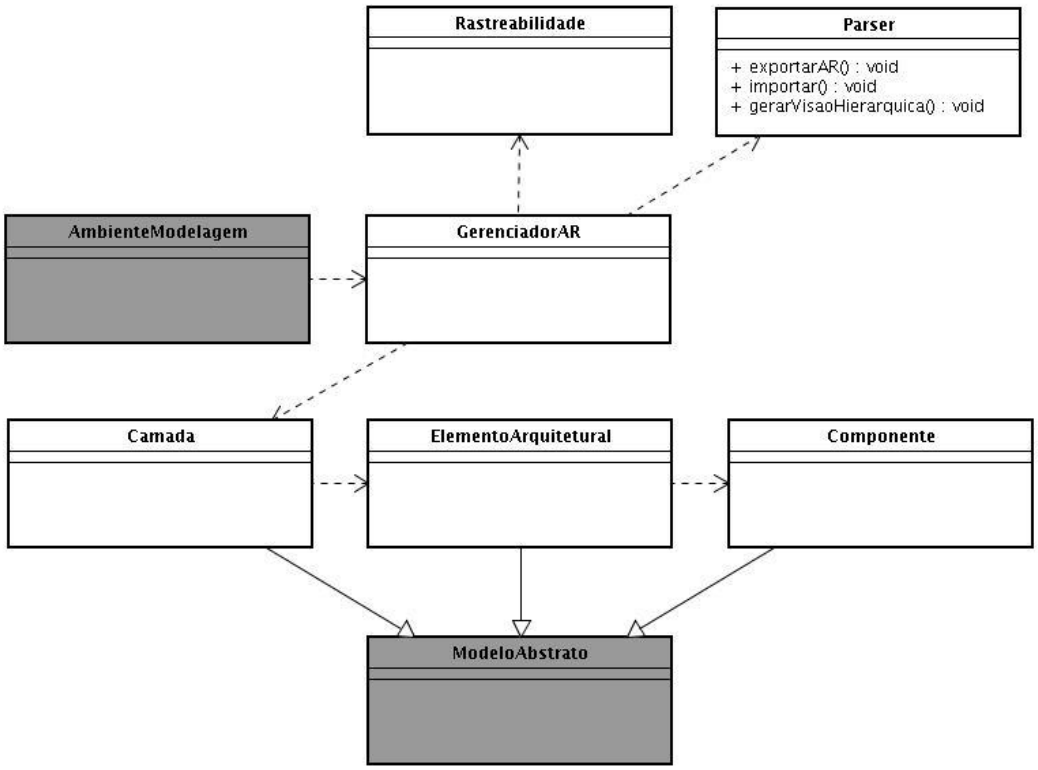


Figura 20 – Diagrama da estrutura da aplicação

Foi necessária, além do subconjunto aqui descrito, a implementação de dezenas de outras entidades, principalmente relacionadas ao sistema de tratamento de eventos do Odyssey, responsável por funcionalidades como inserir e remover itens da árvore semântica, atualização em cascata da árvore de elementos, dentre outros.

Dentre os elementos apresentados no diagrama, a classe *GerenciadorAR* representa um papel importante em nossa implementação. Ela é responsável por manter e controlar as instâncias das descrições de ARs construídas dentro do ambiente de reúso. Esta classe, implementada como um *Singleton* dentro da aplicação mantém as informações sobre uma descrição de AR, em estruturas de dados organizadas sob a forma de uma hierarquia de objetos constituída pelas classes Camada, ElementoArquitetural e Componente. Esse *Singleton* utiliza serviços de duas outras classes, *Rastreabilidade* e *Parser*, para prover funcionalidades necessárias a manipulação de uma instância de ADL. Essa manipulação está relacionada a funcionalidades de exportação (salvar a descrição no formato ACME a partir do Template de AR), importação (carregar a linguagem editada no ABLE, de volta para o Odyssey) e transformação da linguagem em formato ACME, para o formato em camadas, conforme detalhado na Seção 4.6.1 do Capítulo 4.

A Figura 21 apresenta uma visão geral do ambiente de projetos do Odyssey, agora com as alterações realizadas em função da abordagem RADA.

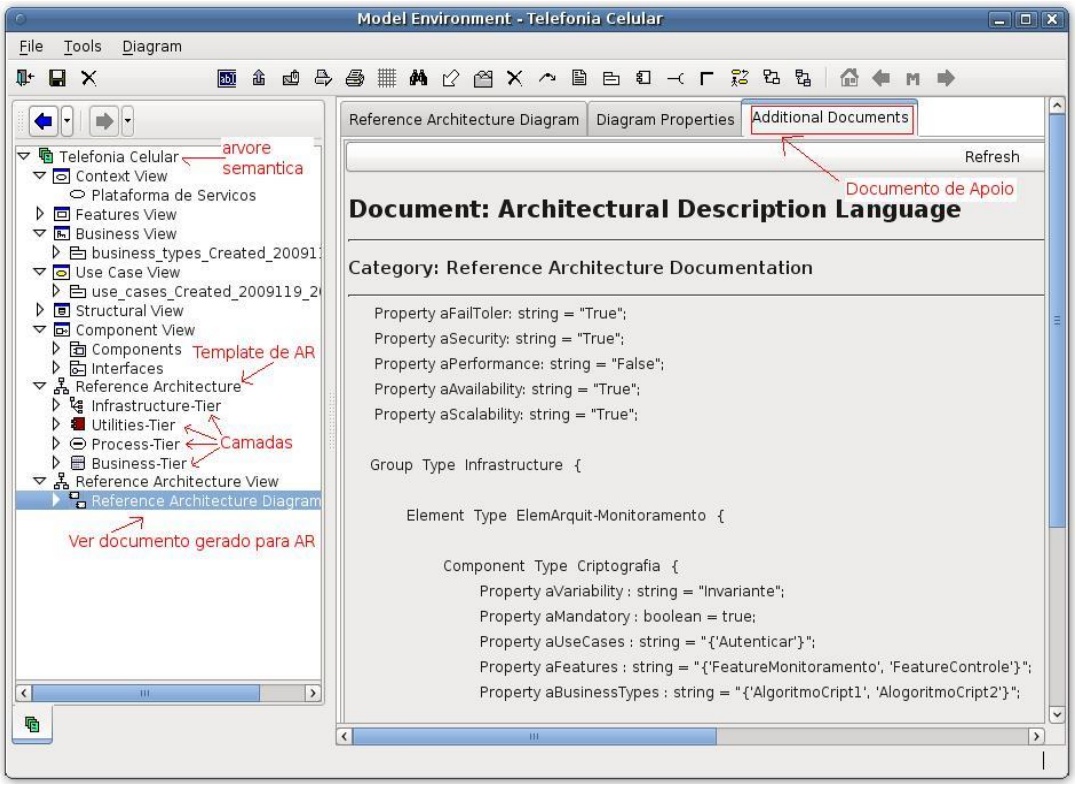


Figura 21 – Visão geral do Odyssey com alterações feitas em função da RADA

A implementação realizada dentro do ambiente Odyssey foi consideravelmente complexa. Tal complexidade se deve principalmente a integração das novas funcionalidades ao ambiente existente. A próxima seção mostra, em detalhes, as alterações realizadas no ambiente de reúso.

5.3.3 Alterações na Estrutura de Elementos Padrão

Inicialmente, ao ter acesso ao ambiente de reúso, é exibida a Árvore Semântica de elementos do domínio, conforme apresentado na Figura 22.

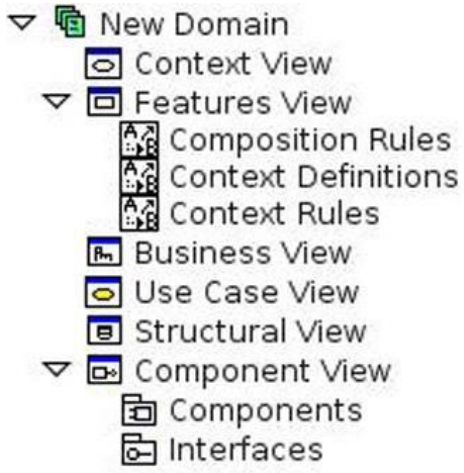


Figura 22 - Árvore Semântica padrão com artefatos de domínio no Odyssey

Esta estrutura é composta por elementos que representam os seguintes conceitos: Contextos de Domínio, *Features*, Elementos de Negócio, Casos de Uso, Elementos Estruturais e Componentes, dentre outros.

A esta estrutura foram adicionados novos elementos que, conforme apresentado na Figura 23, são utilizados para a representação semântica de uma Arquitetura de Referência. Este novo grupo de elementos é denominado *Template de Arquitetura de Referência*, e, de acordo com o proposto por nossa abordagem, traz por padrão alguns tipos pré definidos: *Infrastructure Tier*, *Utilities Tier*, *Process Tier* e *Business Tier*.

Figura 23 - Camadas do *template* de Arquitetura de Referência

Os novos nodos criados foram dispostos de modo a compor uma hierarquia constituída de camadas, elementos arquiteturais e componentes. Este *Template* segue a estrutura do estilo arquitetural camadas. As 4 camadas (Infra-estrutura, Negócio, Processo e Utilitário) foram criadas de maneira a comportar naturalmente os componentes gerados pelo processo CBD-Arch-DE, pois este gera componentes passíveis de agrupamento nesses 4 tipos. A estas camadas podem ser adicionados elementos arquiteturais, os quais possuem componentes agrupados, como mostra a Figura 24.

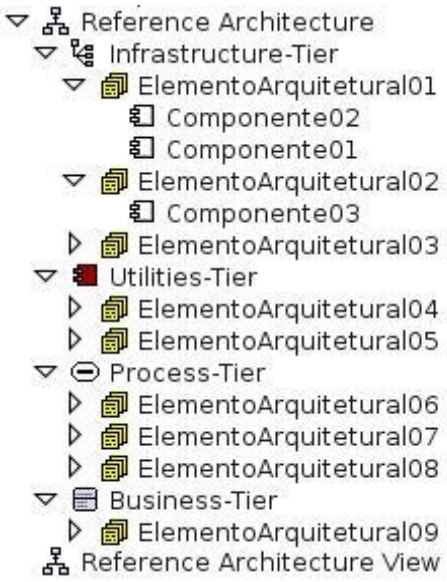


Figura 24 - Árvore Semântica com elementos para a representação de Arquiteturas de Referência

Para cada elemento arquitetural, também podem ser adicionados *n* componentes, cada um relacionado a um determinado artefato do domínio. Como o objetivo dentro deste capítulo é apresentar as novas implementações realizadas em função da abordagem proposta nesta dissertação, não entraremos em detalhes sobre a representação e derivação de outros artefatos no ambiente de reúso.

5.3.4 Gerando Descrição a Partir do Template de Arquitetura de Referência

Finalizada a etapa de montagem da AR, é possível então exportar esta estrutura para um arquivo no formato da ADL, ou seja, gerar a descrição da AR. Para viabilizar a representação de uma AR através de código em linguagem ACME, foram realizadas algumas adaptações no sentido de adicionarmos novos elementos a linguagem existente. Estas adaptações, apresentadas em detalhe na Seção 4.3 do Capítulo 4, foram implementadas através do uso de um recurso provido por ACME, que permite a criação de novos tipos através da especificação de uma propriedade tipada (e.g., Property propertyName = "Property Value";). Estas propriedades, não são tratadas internamente por ACME sob o aspecto semântico, elas são interpretadas e tratadas pelo Odyssey, e armazenam, dentre outras coisas, informações sobre a rastreabilidade entre artefatos, especificação de obrigatoriedades e opcionalidades.

A decisão sobre a agregação destas informações sob a forma de propriedades se deu pelo fato da ADL não dar apoio nativo a esse tipo de representação. Dessa forma, os elementos intrínsecos da linguagem (construtores) foram utilizados unicamente para a representação de elementos como camadas, componentes, elementos arquiteturais e requisitos não-funcionais. A Figura 25 apresenta um exemplo de descrição de AR gerada com base em um *Template* de AR. Nela podemos observar alguns elementos básicos relacionados a AR, tais como: um elemento arquitetural (linha 19) ligado a uma camada (linha 20), e um componente (linha 25), ligado a um elemento arquitetural (linha 26).

A geração do código relacionado a uma AR é feita através da iteração por todos os componentes, para cada elemento arquitetural em todas as camadas. Dessa forma,

visitando os componentes um por um, o algoritmo vai escrevendo no formato da linguagem de descrição, as informações necessárias para a construção da documentação da AR. Basicamente, para cada componente visitado, o algoritmo busca informações sobre dois artefatos associados ao componente (tipos de negócio e casos de uso). Esses artefatos, por sua vez, possuem informações agregadas sobre quais *features* eles realizam. Após determinar quais *features* foram realizadas pelo componente, estas features são identificadas no Modelo de Features do Domínio, para que sejam obtidas informações sobre a variabilidade e opcionalidade, além do rastro entre estes artefatos.

```
1
2  Family OdysseyFamily {
3
4      // Architectural Properties
5      Property aFailToler: string = "True";
6      Property aSecurity: string = "True";
7      Property aPerformance: string = "False";
8      Property aAvailability: string = "True";
9      Property aScalability: string = "True";
10
11     // Infrastructure-Tier:  Reference Architecture Layer
12     // Used to group architectural elements
13     Group Type Infrastructure-Tier = {
14
15     }
16
17     // ElemArquit-Monitoramento:  Architectural Element of Reference Architecture
18     // Used to join components
19     Element Type ElemArquit-Monitoramento = {
20         Property aTier : string = "Infrastructure-Tier";
21     }
22
23     // Criptografia:  Reference Architecture Component
24     // Used to group architectural properties from reference architecture
25     Component Type Criptografia = {
26         Property aElementType : string = "ElemArquit-Monitoramento";
27         Property aVariability : string = "Invariante";
28         Property aMandatory : boolean = true;
29         Property aUseCases : string = "{'Autenticar'}";
30         Property aElementType : string = "ElemArquit-Seguranca";
31         Property aFeatures : string = "{'FeatureMonitoramento', 'FeatureControle'}";
32         Property aBusinessTypes : string = "{'AlgoritmoCript1', 'AlogoritmoCript2'}";
33     }
34 }
```

Figura 25 – Descrição de AR gerada no Odyssey

Uma vez obtidas as informações necessárias para a documentação, a estratégia de representação adotada optou pelo uso de elementos intrínsecos da linguagem,

como recurso para representar entidades como: componentes, camadas e elementos arquiteturais. Dessa forma, estes elementos são reconhecidos pelo ambiente da ACME (ABLE), o qual disponibiliza uma notação gráfica para a sua visualização e edição. Já elementos ligados a rastreabilidade, de acordo com a estratégia de representação adotada, foram expressos através de elementos externos sob a forma de propriedades, tais como as observadas nas linhas 29, 31 e 32 da Figura 25. Assim sendo, estes elementos possuem representatividade semântica apenas quando visualizados dentro do Odyssey.

5.4 ABLE: Modelando uma Arquitetura de Referência

O ABLE (*Architecture-Based Languages and Environments*) [Dav09] é um ambiente dedicado a modelagem de arquiteturas de software, e apresenta suporte a manipulação da linguagem ACME [Gar97]. Esse ambiente foi integrado de maneira a operar junto ao Odyssey, e o objetivo de sua utilização é viabilizar o trabalho de modelagem e validação de uma AR, através de uma ADL estendida por nossa abordagem. A Figura 26 apresenta o ambiente gráfico provido por ABLE para a manipulação da linguagem de descrição ACME.

A descrição gerada no ambiente Odyssey, apresentado na seção anterior (Figura 25), pode ser carregada no ambiente ABLE. Nesse ponto, de acordo com o previsto pela abordagem RADA, o Arquiteto ou Engenheiro de Domínio, pode realizar a complementação da modelagem da AR. Isso é feito através da agregação de informações sobre requisitos não-funcionais, mudanças em elementos arquiteturais ou componentes, e alteração ou agregação de propriedades referentes a variabilidade de componentes.

A modelagem pode ser feita através da interface gráfica provida pelo ABLE, por meio da configuração de propriedades, ou diretamente no código fonte da arquitetura. Nesse momento também é realizada a validação sintática da linguagem, que é disparada através de um *menu* de contexto disponibilizado pelo ambiente.

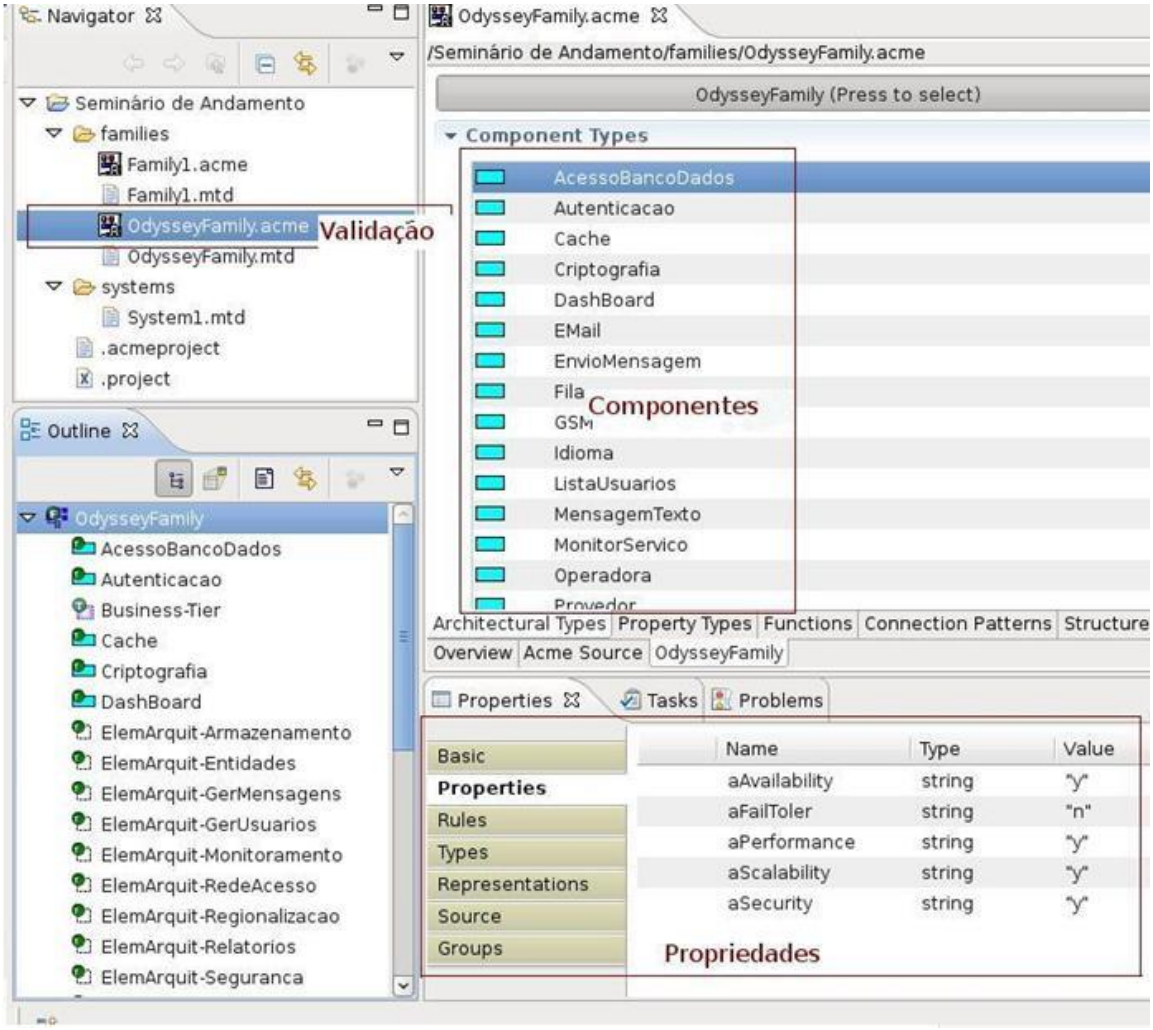


Figura 26 – Visualização das funcionalidades providas por ABLE

A Figura 27 apresenta a mensagem exibida ao Arquiteto, como resultado da validação de uma descrição de AR, sintaticamente válida.

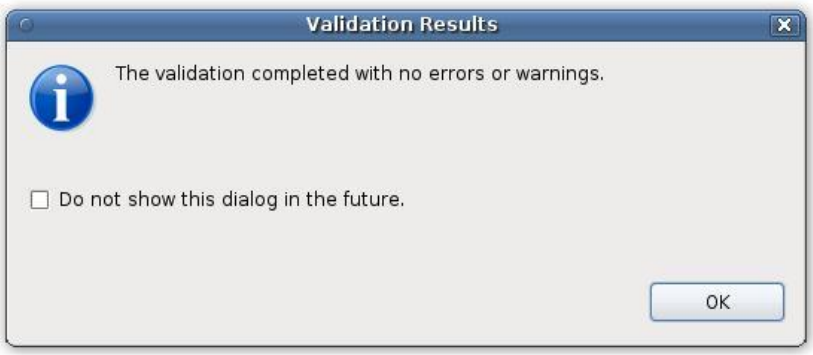


Figura 27 – Validação Sintática feita com sucesso

A Figura 28 apresenta o resultado da validação sintática de uma descrição realizada no formato da ADL, a qual, nesse exemplo, apresentou erros de sintaxe.

Quando isso ocorre, o ambiente identifica na janela de edição de código, qual a linha com problema, e indica no painel de mensagens, conforme exibido na Figura 28, qual a causa mais provável do erro. No caso do erro utilizado nesse exemplo, o mesmo componente foi declarado duas vezes na arquitetura.

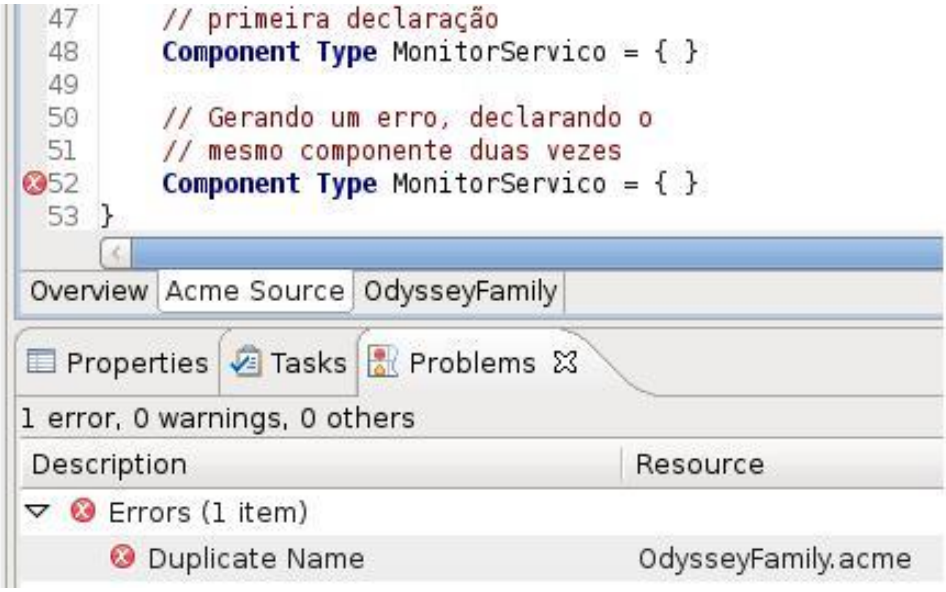


Figura 28 – Validação Sintática alertando sobre erros

Uma vez encerradas as atividades de modelagem da AR, o arquiteto pode retornar ao ambiente Odyssey, onde é feita a importação da AR modelada. Esta representação de AR para o domínio serve como uma base sólida para a Engenharia de Aplicação (EA), no momento da criação de aplicações para o domínio. Principalmente porque a AR aponta algumas possibilidades de disposição de componentes para a criação de uma arquitetura física.

Na próxima seção veremos como esta documentação é utilizada na prática, para apoiar a instanciação de uma aplicação específica para um domínio.

5.5 Odyssey – Instanciação de uma Aplicação

Quando a etapa referente a modelagem da AR no ABLE é encerrada, o arquiteto pode retornar ao ambiente de reúso Odyssey e realizar a importação da estrutura modelada para visualizar a AR. A partir desse momento pode-se dar início ao processo de instanciação de uma aplicação específica.

Conforme discutido e argumentado no Capítulo 4, este processo de instanciação é orientado pela AR criada na ED, diferentemente do processo até então disponível no Odyssey, utilizado pela maioria dos métodos de ED, os quais instanciam aplicações através de modelos de *features*.

5.5.1 Obtendo a Linguagem de Apoio

O acesso a esta estrutura importada é feito através do item da árvore semântica do domínio denominado *Reference Architecture View*. Ao ser selecionado ele carrega, conforme apresentado na Figura 29, uma estrutura contendo vários elementos representando a AR para o domínio.

Nessa estrutura estão agregados *n* elementos arquiteturais, cada qual contendo diversos componentes. Nesse ponto da execução da abordagem, é possível obter uma visualização completa sobre os componentes que foram agrupados em um determinado elemento, assim como também é possível consultar informações referentes aos rastros entre estes componentes e demais artefatos do domínio (artefatos dos quais eles foram originados). Estas informações são muito importantes, pois contribuem significativamente na tomada de decisões por parte do arquiteto, frente a mudanças em requisitos existentes, ou agregação de novas funcionalidades.

Isto porque melhora a visibilidade em relação aos impactos causados por estas mudanças na organização da arquitetura. A tradução proposta por nossa abordagem na visão padrão oferecida por ABLE (no formato ACME) facilita a leitura e interpretação das informações, mesmo para aqueles que não têm um domínio maior sobre a sintaxe da linguagem.

```
Property aFailToler: string = "True";
Property aSecurity: string = "True";
Property aPerformance: string = "False";
Property aAvailability: string = "True";
Property aScalability: string = "True";

Group Type Infrastructure {

    Element Type ElemArquit-Monitoramento {

        Component Type Criptografia {
            Property aVariability : string = "Invariante";
            Property aMandatory : boolean = true;
            Property aUseCases : string = "{'Autenticar'}";
            Property aFeatures : string = "{'FeatureMonitoramento', 'FeatureControle'}";
            Property aBusinessTypes : string = "{'AlgoritmoCript1', 'AlogoritmoCript2'}";

        }
        Component Type MonitorServico {
            Property aVariability : string = "Invariante";
            Property aMandatory : boolean = true;
            Property aUseCases : string = "NA";

        }
    }
}
```

Figura 29 – Visualização da ADL utilizada no apoio a instanciação de Aplicações

Isto é possível, pois nessa visualização, a linguagem, agora já validada, é remontada de forma a aninhar sintaticamente os componentes aos elementos

arquiteturais dos quais eles fazer parte, sem a necessidade do uso de propriedades para a ligação entre os elementos.

5.5.1.1 Selecionando os Componentes para a Instanciação da Aplicação

Uma vez que documentação da AR foi obtida, conforme descrito na Seção 5.4.3.1, chega o momento da instanciação de uma nova aplicação dentro do domínio. Para realizar esta tarefa utilizando o Odyssey, o Engenheiro de Domínio deve sair do painel de edição de domínio, conforme visto na Figura 21 da Seção 5.3.2, e acessar através de um *menu* de contexto, a área do Odyssey dedicada à instanciação de aplicações. Nesta área da aplicação, ao iniciar o *Wizard* de criação de uma nova aplicação baseada em um contexto de domínio, conforme foi implementado, o Arquiteto deve selecionar a opção *Create New Application Based on Reference Architecture*, conforme apresentado na Figura 30.

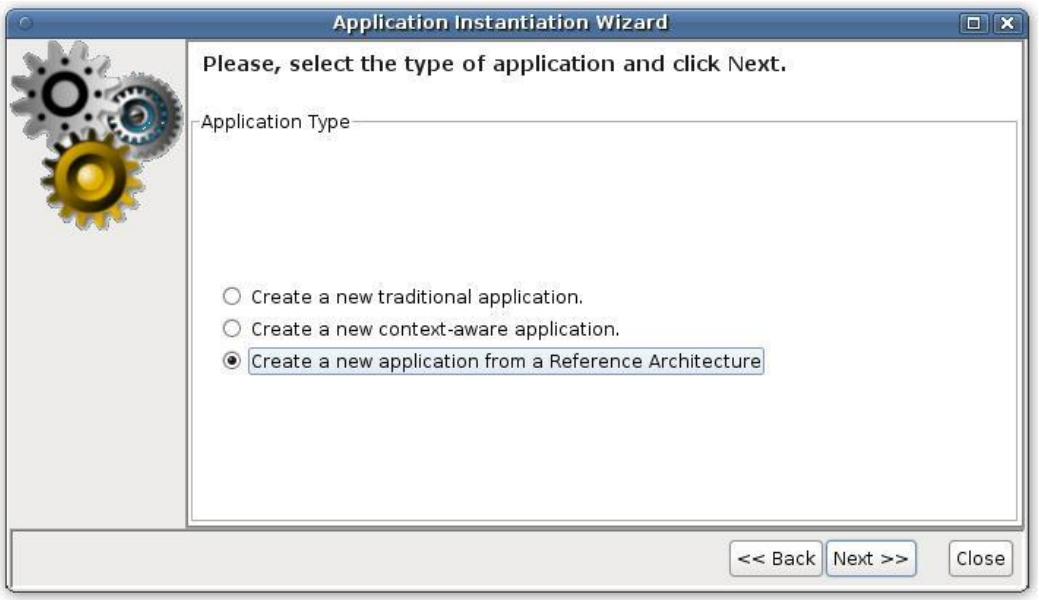


Figura 30 – Wizard: ilustrando passos para a instanciação de aplicações

A partir desse ponto, seguindo o fluxo padrão proposto pelo *Wizard*, o especialista será guiado até a tela mostrada na Figura 31.

Nessa interface, ele realiza a seleção dos componentes distribuídos entre os diversos elementos arquiteturais, com o objetivo de construir uma nova aplicação específica. A escolha dos componentes é também apoiada pelo documento criado para AR, conforme exemplo apresentado na Figura 29.

Contudo, embora seja guiado pela abordagem, esse processo requer ainda a intervenção do Arquiteto na decisão de quais componentes devem ser utilizados. Imagina-se que esta atividade seja também influenciada pelos requisitos impostos para cada nova aplicação dentro domínio, ou por alterações em aplicações existentes.

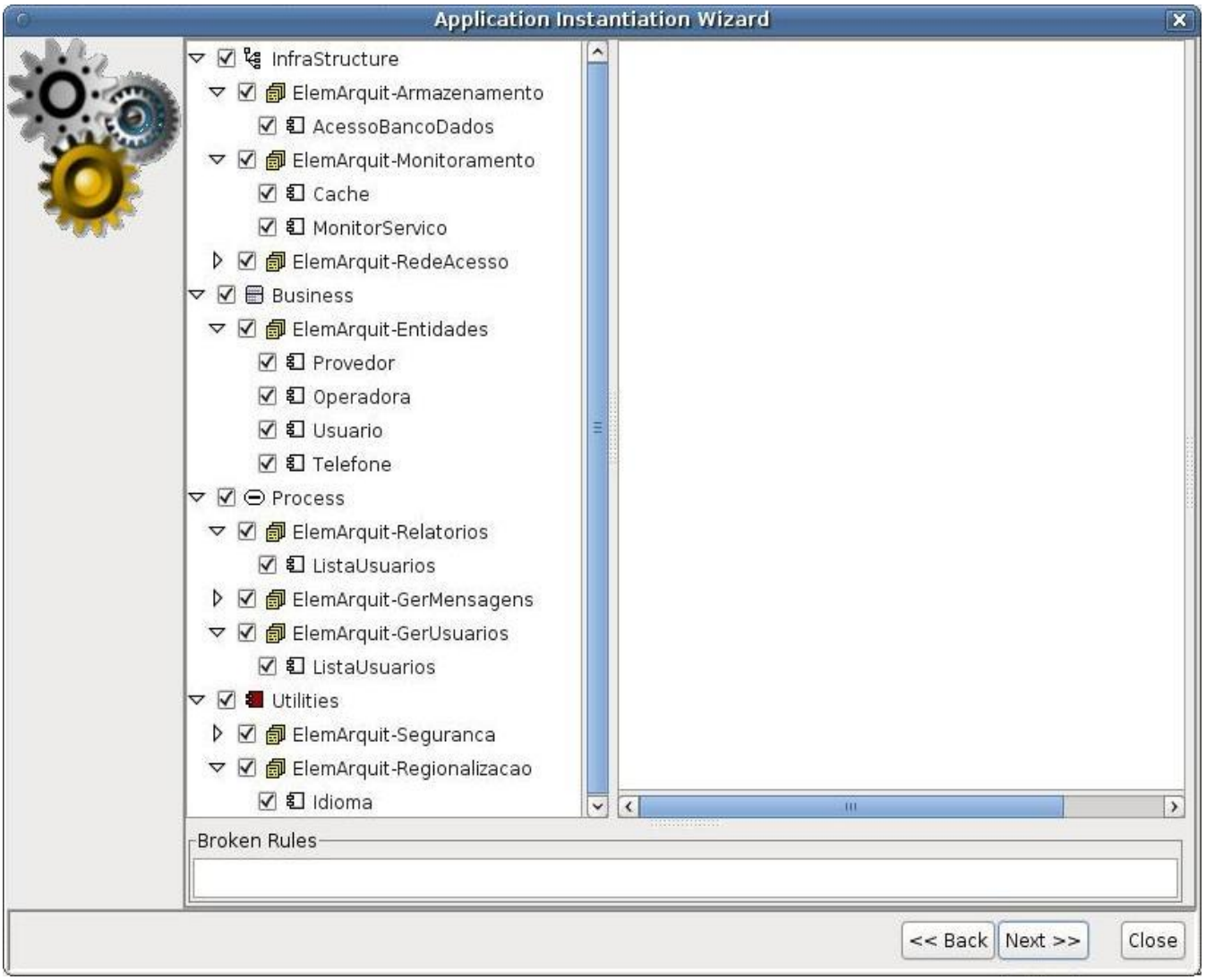


Figura 31 – Instanciação de Aplicações

Uma AR representa um modelo confiável sobre variações de configurações de agrupamentos entre componentes em um domínio, quando comparado a outros artefatos utilizados de base para a instanciação, como o Modelo de Features, conforme já comentado na Seção 4.6.2. A seguir, o Capítulo 6 apresenta um experimento que, dentre outros aspectos, pretende confirmar esta percepção a respeito do uso da AR para apoio a instanciação de aplicações.

5.6 Considerações Finais

Este capítulo apresentou a implementação realizada em função da abordagem proposta nesta dissertação.

Inicialmente foram descritas as funcionalidades implementadas e o fluxo de atividades previsto para a execução dos passos propostos. Essa descrição foi apoiada pela modelagem seguindo o padrão UML [Uml10], utilizando 3 de seus diagramas. Dentre os diagramas empregados, além do diagrama de casos de uso e atividades, foi também explorado o diagrama de classes. Esse último mostrou, como uma forma de documentação mais técnica utilizada, as responsabilidades das principais entidades envolvidas na implementação. Além disso, também foi apresentada a estratégia de implementação das funcionalidades criadas no Odyssey, e a descrição do algoritmo utilizado para a construção dos rastros.

Posteriormente a descrição da implementação, foi apresentado o ambiente Odyssey, o qual serviu de base para a maior parte das atividades realizadas. A descrição deste ambiente privilegiou a sua utilização através das funcionalidades implementadas para a execução das atividades propostas. Dentre as funcionalidades apresentadas foram mostradas a montagem de uma AR, sua documentação por ADL e instanciação de uma aplicação com base na documentação e AR geradas. Da mesma forma, foi apresentado o ambiente ABLE. Com relação a este ambiente, foi apresentada a sua interface, janela de edição, paleta de propriedades e mecanismo de compilação.

No que diz respeito ao Odyssey, foi necessário um estudo exaustivo sobre a arquitetura do ambiente para identificar os pontos onde efetivamente as implementações e extensões necessárias deveriam ser aplicadas. Assim como no Odyssey, foi também realizado um estudo exaustivo do ambiente ABLE para identificar as possibilidades de extensão da ADL no ambiente, feitas através do recurso de *properties values*. Portanto, podemos confirmar que esta pesquisa aborda uma visão de desenvolvimento com reúso, no que tange as atividades de implementação da RADA, e o desenvolvimento para reúso, no que diz respeito ao uso da abordagem por um Arquiteto.

Com base nas implementações desenvolvidas, foi possível conduzir uma atividade de experimentação, a qual será descrita no próximo capítulo.

6 AVALIAÇÃO DA PROPOSTA

Esse capítulo apresenta um cenário de uso da abordagem RADA com o objetivo de obter indícios de viabilidade da mesma. Conforme pode ser observado no Apêndice O, foi inicialmente realizado um experimento como forma de avaliação da abordagem proposta no Capítulo 5, mas, pelo fato dos resultados obtidos através da realização deste experimento não terem se mostrado suficientemente representativos, optou-se pela realização de outra avaliação sob a forma de um cenário de uso, apresentada em detalhes ao longo desse capítulo.

A forma de avaliação apresentada a seguir é baseada no processo proposto por Oates [Oat06]. Esse processo é indicado para abordagens nas quais a base da proposta é dependente do protótipo desenvolvido. A seguir são apresentados o protocolo de avaliação e sua instanciação junto a abordagem.

6.1 Introdução

Em muitos projetos de pesquisa, mais especificamente em computação, está envolvido o desenvolvimento de um produto baseado em computação. Como exemplos de produtos podemos citar um *web site*, uma animação, um sistema de apoio a alguma atividade, etc. Estes projetos exploram e mostram alternativas de tecnologia sob a forma digital. Todavia, para serem considerados uma pesquisa, esses projetos precisam demonstrar qualidades como análise, argumentos e critérios de avaliação para, dessa forma, representarem uma contribuição para o conhecimento em uma determinada área. Essa contribuição depende do papel desempenhado pela tecnologia dentro da pesquisa. Nesse contexto, a tecnologia pode, segundo Oates [Oat06], exercer um dos seguintes papéis:

I. Representar o foco principal da pesquisa,
II. Se tornar um facilitador para a execução de alguma atividade, ou

III. Apresentar um produto final tangível de um projeto onde o foco está no processo de desenvolvimento.

Para o caso da avaliação de nossa abordagem vamos considerar que o papel desempenhado pela tecnologia representa o foco principal da pesquisa, pois está associado a viabilizar a criação de uma documentação para uma Arquitetura de Referência (AR) e suporte a sua instanciação.

Dessa forma, para avaliar a abordagem RADA, vamos seguir o processo definido por Oates [Oat06], o qual envolve os seguintes passos: reconhecimento do problema, proposta, desenvolvimento, avaliação e conclusão.

Reconhecimento do Problema: é a racionalização e articulação do problema. Isto pode ser feito através de uma revisão da literatura (autores relacionados a área de pesquisa), análise de novas descobertas em outra área relacionada (a partir das necessidades de praticantes ou clientes), ou a partir de um campo de pesquisa específico ou desenvolvimento recente na área da tecnologia;

Proposta: está relacionada ao potencial criativo proveniente da curiosidade a respeito de uma possível solução que foi proposta para um determinado problema;

Desenvolvimento: está relacionado à implementação da solução idealizada. A forma como o desenvolvimento é feito, depende do tipo de tecnologia e de artefatos que estão sendo propostos para a solução. Por exemplo, um novo algoritmo necessita de uma prova formal; uma nova interface com funcionalidades específicas para atender necessidades especiais de seres humanos necessita o desenvolvimento de um sistema de software.

Avaliação: examina os artefatos desenvolvidos e procura avaliar sua eficácia e deficiência em relação ao efetivo apoio provido na execução das atividades a que se propôs.

Conclusão: é o passo onde os resultados do processo são consolidados e analisados. Nessa etapa, o conhecimento obtido, assim como os problemas identificados, é registrado para o seu uso em pesquisas futuras, servindo como base para novas soluções ou lições apreendidas.

A seguir é apresentada a execução das etapas propostas por [Oat06] dentro do contexto dessa pesquisa, como forma de avaliar a viabilidade da abordagem RADA.

6.2 Reconhecimento do Problema

Conforme descrito na introdução desse capítulo, o reconhecimento do problema pode ser feito de diversas maneiras. No caso desta pesquisa, foi realizada uma revisão da literatura, a qual foi apresentada no Capítulo 3. A partir desta revisão, que analisou os principais processos de Engenharia de Domínio e o suporte provido a manipulação de AR, foram identificados problemas relacionados à criação de uma documentação para uma Arquitetura de Referência e problemas também relacionados ao processo de instaciação de aplicações.

Estes problemas, conforme já abordado na seção 1.2 (referente à motivação), implicam, basicamente, em dificuldades e falta de apoio para a atividade de criação de documentação para uma AR e instanciação de aplicações.

6.3 Proposta

Com o objetivo de fornecer apoio à atividade de criação de uma documentação para AR e instanciação de aplicações, essa pesquisa propôs, sob o ponto de vista conceitual, a abordagem RADA, conforme detalhe no Capítulo 4.

Como forma de suportar essa abordagem, é proposta uma ferramenta, conforme apresentado no Capítulo 5, que automatiza parte do processo de documentação definido pela abordagem. Através do uso desta ferramenta pretende-se atender aos seguintes requisitos:

I. Apoiar ao engenheiro de domínio na atividade de criação de uma documentação para uma AR no formato ACME;
II. Viabilizar a validação sintática da descrição gerada;
III. Permitir a indicação de requisitos não funcionais;
IV. Servir como base para instanciação de aplicações:

A seguir, ao longo desta subseção, vamos apresentar um cenário de utilização da ferramenta dentro de um processo com apoio ao reúso, apoiando a atividade de criação de uma documentação.

6.3.1 Cenário de Utilização

O cenário de utilização descrito nessa subseção está inserido dentro do fluxo de execução de atividades do processo de Engenharia de Domínio (ED) CBD-Arch-DE. Dentro das atividades previstas por este processo ED, o cenário apresentado está contextualizado mais especificamente em sua fase de projeto, onde as Arquiteturas de Referência são criadas e documentadas. O domínio utilizado como base para a construção do cenário apresentado está relacionado a aplicações de *data warehouse*, e.g., aplicações de relatório, monitoramento, *dash boards*, extração e transformação de dados, etc. Para este domínio, através da execução de atividades de análise de domínio, foram identificados alguns componentes, dentre os quais podemos citar: componentes com funcionalidades para extração de transformação de dados (ETL), criptografia (*Encryption*), etc.

Para fins de definição de escopo e viabilidade da descrição do cenário desejado, assumimos que a fase análise de domínio e de projeto, já foram executadas, e que possuímos devidamente criados diversos agrupamentos de componentes. Esses agrupamentos, gerados com base em critérios de acoplamento e coesão, conforme discutido na seção 3.2.3 do Capítulo 3, são mantidos dentro do ambiente de reúso em uma estrutura em forma de árvore e organizados em camadas, conforme pode ser observado na Figura 32. A atividade de agrupamento de componentes é realizada com

o apoio de uma funcionalidade já existente no ambiente de reúso, desenvolvida em decorrência da implementação do suporte ao processo de ED. Essa funcionalidade, implementada sob a forma de um *plugin*, permite a geração automatizada de elementos arquiteturais, e sua inserção na estrutura representada pelo *template* de AR. A partir dessa estrutura, o Engenheiro do Domínio pode gerar a documentação para a AR.

Para realizar essa tarefa, de acordo com a implementação realizada, é necessário selecionar com o botão direito do mouse a opção *Reference Architecture*, da estrutura mostrada na Figura 32, e realizar uma exportação dos elementos da estrutura para um arquivo no formato da ADL ACME. O algoritmo responsável pela exportação realiza um *parsing* dessa estrutura, visitando, para cada camada existente no *template* de AR, todos os elementos arquiteturais pertencentes a cada uma das camadas.

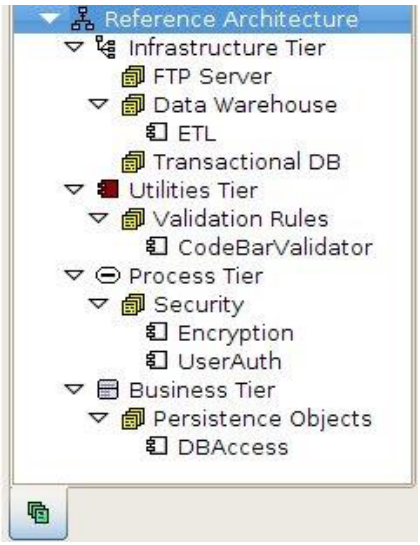


Figura 32 – Arquitetura de Referência no ambiente de reúso

Para cada elemento arquitetural, são visitados todos os componentes, e para cada componente, diversas informações são obtidas. Essas informações incluem, além do nome do componente e elemento arquitetural no qual está inserido, informações sobre rastreabilidade entre artefatos. No caso da AR exibida na Figura 32, por questões de simplificação, parte do código foi omitida, mas para este cenário,

basicamente, é utilizado o construtor *Group Type* para representar camadas, e.g., *Group Type InfraStructure*, *Group Type UtilitiesTier*, *Group Type ProcessTier* e *BusinessTier.* Já elementos arquiteturais são documentados com o construtor *Element Type*, e.g., *Element Type FTPServer, Element Type DataWareHouse, Element Type ValidationRules, Secutiry* e *Element Type PersistenceObjects*. Componentes são documentados com o uso do construtor *Component Type*, e.g., *Component Type ETL, Component Type CodeBarValidator, Component Type Encryption, Component Type UserAuth* e *Component Type DBAccess.*

A documentação referente a um determinado componente da AR, com todas as informações relacionadas à variabilidade, assim como a documentação de requisitos não funcionais, é feita através de um grupo de propriedades, conforme detalhado na seção 4.3 do Capítulo 4. A Figura 33 mostra a linguagem gerada como resultado da exportação do para ACME, da AR mostrada na figura anterior.

```
1
2  Family OdysseyFamily {
3
4      // Architectural Properties
5      Property aFailToler: string = "True";
6      Property aSecurity: string = "True";
7      Property aPerformance: string = "False";
8      Property aAvailability: string = "True";
9      Property aScalability: string = "True";
10
11     // Infrastructure-Tier:  Reference Architecture Layer
12     // Used to group architectural elements
13     Group Type Infrastructure-Tier = {
14
15     }
16
17     // ElemArquit-Monitoramento:  Architectural Element of Reference Architecture
18     // Used to join components
19     Element Type ElemArquit-Monitoramento = {
20         Property aTier : string = "Infrastructure-Tier";
21     }
22
23     // Criptografia:  Reference Architecture Component
24     // Used to group architectural properties from reference architecture
25     Component Type Criptografia = {
26         Property aElementType : string = "ElemArquit-Monitoramento";
27         Property aVariability : string = "Invariante";
28         Property aMandatory : boolean = true;
29         Property aUseCases : string = "{'Autenticar'}";
30         Property aElementType : string = "ElemArquit-Seguranca";
31         Property aFeatures : string = "{'FeatureMonitoramento', 'FeatureControle'}";
32         Property aBusinessTypes : string = "{'AlgoritmoCript1', 'AlogoritmoCript2'}";
33     }
34 }
```

Figura 33 – ADL Gerada no ambiente Odyssey

No cenário apresentado, o componente Criptografia tem o seu comportamento documentado através de algumas propriedades introduzidas na ADL ACME. A propriedade *aElementType* define a qual elemento arquitetural esse componente está associado. Propriedades como *aVariability* e *aMandatory* estão relacionadas a especificação da variabilidade desse componente. Finalmente, são utilizadas na documentação da AR descrita nesse cenário, propriedades especiais, representadas por: *aUseCases*, *aFeatures* e *aBusinessTypes*. Estas propriedades são preenchidas com listas de elementos. Isto porque, tomando como exemplo a propriedade *aUseCase*, um componente pode ter sido originado em função da realização de vários casos de uso, assim como pode estar associado a implementação de várias *features* (*aFeatures*) ou vários tipos de negócio (*aBusinessTypes*).

6.3.2 Validação Sintática

Após a geração da documentação, é possível realizar a validação sintática da descrição gerada e a indicação de requisitos não funcionais. A indicação de requisitos não funcionais é realizada no ambiente ABLE e utiliza, conforme a descrição mostrada na figura anterior, as propriedades *aFailToler*, *aSecurity*, *aPerformance*, *aAvailability* e *aScalability*. Através destas propriedades é possível indicar em alto nível, quais requisitos não funcionais são atendidos pela arquitetura.

No ABLE, conforme detalhado na seção 5.4 do Capítulo 5, além da indicação dos requisitos não funcionais e da edição dos elementos descritos através da linguagem, é realizada a validação sintática. A validação é feita através da utilização do compilador provido pelo ambiente, que pelo fato da descrição ser gerada no formato ACME, é capaz de realizar *parsing* em busca de erros relacionados à sintaxe da descrição. O processo de validação da descrição foi apresentado em detalhes na Seção 4.5.3 do Capítulo 4. Para fins de apresentação deste cenário, é apenas colocado que, através da automação realizada no processo, e principalmente por ter sido desenvolvido de forma integrada com um ambiente específico para a

manipulação de arquiteturas de software, essa tarefa encontra-se plenamente apoiada.

6.4 Avaliação

Conforme descrito na introdução deste capítulo, está previsto dentro do processo definido por Oates [Oat06], uma etapa relacionada à avaliação. Nesta etapa os artefatos desenvolvidos são analisados para que seja possível avaliar os pontos fortes e fracos da abordagem em relação ao efetivo suporte provido na execução das atividades a que se propôs. Dessa forma, tal verificação busca apurar se através do uso da ferramenta é possível atingir os requisitos apresentados na Seção 6.3, relacionados à proposta.

De acordo com o que foi apresentado neste cenário, se conclui que a abordagem cumpre o que se propôs. Conforme apresentado na Tabela 3, todos os requisitos foram contemplados.

Tabela 3 – Cobertura de requisitos pela ferramenta

REQUISITO	COBERTURA
1. *Apoiar ao engenheiro de domínio na atividade de criação de uma documentação para uma AR no formato ACME,*	Sim
2. *Viabilizar a validação sintática da descrição gerada e*	Sim
3. *Permitir a indicação de requisitos não funcionais.*	Sim

Em relação ao requisito número 1, a abordagem apresenta cobertura, pois apóia ao Engenheiro de Domínio através de um mecanismo de *parsing* que, quando acionado, gera automaticamente um artefato sob o formato de um documento ACME representando a AR.

No que diz respeito a viabilizar a validação sintática da descrição e indicação de requisitos não funcionais (requisitos 2 e 3), o apoio é dado pela integração feita entre o ambiente de reuso e o ambiente para a manipulação da linguagem. Essa integração

permite que a descrição gerada seja validada e requisitos não funcionais sejam especificados através de um grupo de propriedades adicionados a linguagem.

6.5 Conclusão

Este capítulo apresentou a avaliação realizada como forma de verificar a viabilidade da abordagem apresentada nesta dissertação. Esta avaliação, baseada no processo definido por Oates [Oat06], foi realizada através da descrição de um cenário de utilização da RADA. Neste cenário, partindo de uma AR já criada dentro da fase de projeto de um processo de ED, um Engenheiro de Domínio realiza a geração e validação da documentação para uma AR.

Conforme foi apresentado na subseção de Avaliação, a RADA atinge o objetivo ao qual se propôs através da cobertura de três requisitos. Como vantagens observam-se, por exemplo, o apoio a geração automatizada do documento de arquitetura de referência no formato ACME, a indicação de requisitos não funcionais e a validação da descrição gerada. Como ponto fraco está o fato do mecanismo de geração da descrição trabalhar de maneira exclusiva com a estrutura do *template* de AR organizado segundo o estilo arquitetural camadas. Tal limitação impede a documentação de AR organizadas sob outros estilos. Outro ponto observado é o fato de não haver uma validação semântica da linguagem, quando confrontada com os componentes carregados na AR.

Mesmo com as limitações apresentadas, passíveis de evolução em trabalhos futuros, a abordagem avaliada neste capítulo (e apresentada em detalhes no Capítulo 4), atende ao que se propôs, pois ao final de sua execução são gerados artefatos que constituem uma documentação específica para uma AR.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma proposta para a documentação de Arquiteturas de Referência para domínios de Aplicações através da utilização de uma Linguagem de Descrição Arquitetural estendida. A Abordagem RADA sugere a integração das atividades necessárias para a elaboração de uma documentação de Arquitetura de Referência, a um processo de Engenharia de Domínio, em um contexto de Reúso de software.

Conforme apresentado ao longo desta dissertação, foi realizada uma revisão da literatura com o objetivo de avaliar o apoio dos métodos de Engenharia de Domínio ao processo de construção de documentação de Arquiteturas de Referência, Foi observado através de uma análise, levando em consideração alguns critérios estabelecidos, que estes métodos falham ao não apoiarem por completo a construção de uma AR. Dessa forma, abordagem RADA, dá um pequeno passo, porém importante, no sentido de melhorar o apoio a etapa de projeto da Engenharia de Domínio, mais especificamente com relação a criação de documentação de Arquiteturas de Referência.

A verificação da viabilidade desta abordagem foi realizada através da elaboração de um protótipo, no qual foram implementadas as principais funcionalidades propostas pela abordagem. Através deste protótipo foi realizada uma avaliação orientada por um protocolo de experimentação proposto por Oates [Oat06]. Através deste protocolo foi possível evidenciar a utilidade da RADA, segundo os requisitos definidos para o cenário analisado.

Além da abordagem proposta, este trabalho agrega como contribuição um levantamento por meio de uma revisão da literatura, a respeito do apoio oferecido por métodos de Engenharia de Domínio no processo de criação e documentação de Arquiteturas de Referência e instanciação de aplicações. A criação de um documento sobre a AR do domínio, mais facilmente compreensível, facilita a disseminação do conhecimento entre os diversos participantes interessados no projeto, não ficando

restrito apenas aos Arquitetos com domínio da linguagem, o que representa um avanço em relação a uma das principais desvantagens encontradas na utilização de ADLs, relacionada à dificuldade de compreensão da linguagem, devido a curva de aprendizagem. Outras vantagens estão relacionadas a rastreabilidade dentro de um processo de Engenharia de Domínio, pois além de descrever os atributos inerentes a variabilidade, mantém também o rastro entre os componentes documentados na Arquitetura de Referência e as *features* criadas no domínio, o que ajuda o Arquiteto a dimensionar o impacto de alterações nas arquiteturas das aplicações instanciadas dentro do domínio.

Como trabalhos futuros desta pesquisa são sugeridos:

a) A necessidade de um tratamento mais detalhado as propriedades da AR relacionadas à representação de requisitos não-funcionais, no sentido de viabilizar a representação de medidas referentes a um determinado requisito não-funcional, ao invés de apenas indicar se ele está presente na arquitetura;
b) Estender o *template* de Arquitetura de Referência permitindo a estruturação de outros estilos arquiteturais é não só ao estilo arquitetural camadas como foi proposto nesta abordagem. Neste sentido, a abordagem RADA poderia ser integrada a qualquer processo de ED e não só a um processo baseado no conceito de DBC, como é o caso do CBD-Arch-DE;
c) Em relação a linguagem ACME, é sugerida a criação de mecanismos dedicados a validação semântica das descrições geradas, para identificar, sobretudo, a ocorrência de desvios ou violações arquiteturais;
d) Apresentar a abordagem proposta junto a equipe do projeto Odyssey, com o objetivo de agregar estas novas funcionalidades a distribuição principal do projeto;
e) Apoio a criação de mais de uma AR para um mesmo domínio, uma vez que o trabalho futuro discutido no item b tenha sido desenvolvido;

f) Uma vez que artefatos da ED tenha vido versionados, também manter o versionamento da AR que documenta estes artefatos de domínio;
g) Permitir o versionamento das ARs geradas, de acordo com as novas versões dos artefatos gerados durante posteriores iterações do processo de engenharia para um dado domínio.

Finalmente, podem ser feitos trabalhos no sentido de melhoria do apoio a Engenharia de Aplicações no momento da instanciação de arquiteturas, pois o avanço da pesquisa pode ajudar na geração automatizada de código específico da arquitetura, em uma tecnologia específica.

REFERÊNCIAS BIBLIOGRÁFICAS

[Aco06] Acosta, C.; Sastrón, B. "Schematic Architecture: Reference Architecture Frameworks Particular Models for the Shop Floor Environment". In: IEEE Industrial Electronics, IECON 2006 32nd Annual Conference, 2006, pp. 453-454.

[All97] Allen, R. J. "A Formal Approach to Software Architecture". Pittsburgh PA Carnegie Mellon University, May 1997, pp. 39-40 e 59-66.

[Ara06] Araújo, M. A.; Barros, M.; Travassos, G.; Murta, L. "Métodos Estatísticos aplicados em Engenharia de Software Experimental". http://gbd.dc.ufscar.br/download/files/meetings/meetings2009-2/luana.metodos%20estatisticos.eselaw.pdf, Novembro 2009.

[Asi03] Asikainen, T.; Soininen, T.; Männistö, T. "A Koala-Based Approach for Modelling and Deploying Configurable Software Product Families". In: Software Product-Family Engineering, 5th International Workshop, PFE, Siena, Italy, November 2003, pp. 231-249.

[Atk02] Atkinson, C.; Bayer, J.; Bunse, C.; et al. "Component-based product line engineering with UML". In: 7th International Conference, ICSR-7 Austin, TX, Abril 2002, pp. 343-344.

[Bac04] Bacelo, A. P. T.; Vasconcelos, A.; Werner, C. "Representação de Variabilidades em Componentes de Negócio no Contexto da Engenharia de Domínio". In: 5o Workshop de Desenvolvimento Baseado em Componentes Juiz de Fora, Minas Gerais, Brasil, Novembro, 2004, pp. 2-8.

[Bac06] Bacelo, A. "Uma Abordagem de Projeto Arquitetural Baseado em Componentes no Contexto de Engenharia de Domínio". Tese de Doutorado, COPPE/UFRJ, 2006, 207p.

[Bas94] Basili, V. R.; Caldiera, G.; Rombach, H. D. "The Goal Question Metric Approach". Wiley Library - Encyclopedia of Software Engineering , 15 de Janeiro, 1994, 10p.

[Bas03] Bass, Len.; Clements, Paul.; Kazman, Rick. "Software Architecture in Practice". Addison Wesley, vol. 2, Abril 2003, pp. 57-61.

[Bas05] Bashroush, R.; Brow.; Spence, I.; Ilpatrick, P. "ADLARS, An Architecture Description Language for Software Product Lines". In: Annual IEEE/NASA Software Engineering Workshop, 2005, Greenbelt, pp. 163-173.

[Bat95] Batory, D.; Coglianese, M.; Goodwin S. S. “Creating Reference Arquitectures: An Example from Avionics”. In: Proceedings of the Symposium on Software reusability ACM SIGSOFT, 1995, pp. 25-34.

[Bir09] Biró, M.; Messnarz, R. "SPI experiences and innovation for Global Software Development". Interscience Wiley, vol. 14, Junho 2009, pp. 243-245.

[Boe88] Boehm, B. "A Spiral Model of Software Development and Enhancement", IEEE Computer Society, vol. 21, Maio 1988, pp. 61-72.

[Bos00] Bosh, J. “Design and use of Software architectures” Nova Yorque: ACM Press Addison-Wesley, 2000, pp. 52-115.

[Bra00] Villela, R.M.M.B. “Busca e Recuperação de Componentes em Ambientes de Reutilização de Software”, Tese, COPPE, UFRJ, 2000, pp. 96-138.

[Bri96] Briand, L. C.; Emam, K. E.; Morasca, S. "On the Application of Measurement Theory in Software Engineering", Journal of Empirical Software Engineering, vol. 1, 1996, pp. 61-88.

[Bus96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P. “Pattern-oriented software architecture: A System of Patterns”. Chichester UK: John Wiley, 1996, vol. 1, pp. 394-403.

[Cha09] Change, V. “JUDE Design & Communication”, Capturado em: http://jude.change-vision.com, Novembro 2009.

[Cle00] Clements, P.; Bachmann, F.; Bass, L.; Carrieri, J.; Garlan, D. “Software Architecture Documentation in Practice”. Nova Jersey: Addison-Wesey, Edição 3, pp. 331-341.

[Cog93] Coglianese, L.; Szymanski, R. “DSSA ADAGE: An Environmet for Architecture based Avionics Development”. In: Proceedings of AGARD IBM Owego, 1993, pp. 12-14.

[Coh03] Cohen, S. “Predicting When Product Line Investment Pays”, In: Proceedings of the Second ICSE on Software Product Lines, 2003, pp. 385-409.

[Cza07] Czarnecki, K. "Software Reuse and Evolution with Generative Techniques" In: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, Atlanta, Georgia, USA, 2007, 575p.

[Dso99] D'souza, D. F.; Wills, A. C. “Objects, components, and frameworks with UML: the catalysis approach" Chichester: Addison-Wesley Longman Publishing Co., Inc, 1999, vol. 1, 816p.

[Dav97] David, C. R. "Success Factors for Software Reuse That are Applicable Across Domains and Businesses". In: Proceedings of the ACM Symposium on Applied Computing, Jose, California, USA, 1997, pp. 182-186.

[Dav09] Dave W. “The ACME Project”, Capturado em: http://www.cs.cmu.edu/~acme, Julho 2009.

[Dom07] Domingos, M. "Uma Arquitetura de Referência para Sistemas de Informação e Portais de Serviços de Governo Eletrônico”, Dissertação de Mestrado, Universidade Federal de Santa Catarina UFSC, 2007, 129p.

[Ecl09] Eclipse F. “The Eclipse Project”, Capturado em: http://eclipse.org, Julho 2009.

[Ekl05] Eklund, U.; Askerdal, Ö.; Granho, J.; Alminger, A.; Axelsson, J. "Experience of introducing reference architectures in the development of automotive electronic systems". ACM SIGSOFT Software Engineering Notes, vol. 30 Issue 4, Julho 2005, pp. 1-6.

[Fon96] Fonseca, J. S.; Martins, G. A. “Curso de Estatística”. São Paulo: Atlas, 1996, Edição 6, vol. 1, 89p.

[Fra05] Frakes, W. B.; KANG, K. “Software Reuse Research: Status and Future”. IEEE Transactions on Software Engineering, vol. 31, Julho 2005, pp. 529-536.

[Gar97] Garlan, D.; Monroe, R.; Wile, D. “Acme: An Architecture Description Interchange Language”. In: CASCON (Conference of the Center for Advanced Studies on Collaborative research). 1997, pp. 169-183.

[Gim05] Gimenes, I.; Huzita, E. “Desenvolvimento Baseado em Componentes”, Rio de Janeiro: Ciência Moderna, 2005, Edição 1, 2005, 304p.

[God05] Godfrey, W. M.; Grosskurth, Alan. "A Reference Architecture for Web Browsers". In: Proceedings of the 21st IEEE Computer Society International Conference, 2005, pp. 661-664.

[Gom04] Gomma, H. "Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures", Redwood: Addison Wesley, 2004, Edição 1, 736p.

[Gro09] Grupo de discussão Acrópolis (Filosofia). “Versão eletrônica do livro: O Discurso do Método”. Capturado em: http://br.egroups.com/group/acropolis, Junho 2008.

[Gro09a] Group, O. M. “OMG Model Driven Architecture”. Capturado em: http://www.omg.org/mda/, Dezembro 2009.

[Haa95] Haahr, M.; Sing, A. "Peer-To-Peer Reference Architecture Communication System Software and Middleware". In: Comsware First International Conferencepp. 1995, pp. 137-153.

[Har04] Harkki, J. “Architecture Description Languages Theory”. In: IFIP TC-2 Workshop on Architecture Description Languages (WADL), World Computer Congress, 2004, 207p.

[Ine10] INE. “Departamento de Informática e Estatística - UFSC”, Capturado em: http://www.inf.ufsc.br/~barbetta/disciplinas/CCO/Tabelas.pdf, Janeiro 2010.

[Ins10] Instituto, E. “Universidade Federal de Alagoas – Instituto de Matemática”, Capturado em: http://www.im.ufal.br, Dezembro 2009.

[Kan90] Kang, C.; Cohen, S.; Hess, J.; Novak, W.; Peterson, A. "Feature-Oriented Domain Analysis. Feasibility Study". Software Engineering Institute, Pittsburgh CMU/SEI-90-TR-21, 1990.

[Kan98] Kang, K. C.; Kim, S.; Lee, J.; Kim, K.; Shin, E.; Huh, M. “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures”. In: Annals of Software Engineering, vol. 5, 1998, pp. 143-168.

[Kan02] Kang, K. C.; LEE, J.; Donohoe, P. "Feature-Oriented Product Line Engineering" In: IEEE Software, vol. 9, issue 4, Julho/Agosto 2002, pp. 58-65.

[Kan05] Kang, K.; William B. Frakes. "Software Reuse Research: Status and Future". In: IEEE Transactions on Software Engineering, vol. 31 issue 7, Julho 2005, pp. 102-135.

[Kit01] Kitchenham, P. “Principles of Survey research”. In: ACM SIGSOFT, vol. 26 issue 6, Novembro 2001, pp. 16-18.

[Kou95] Kougut, P.; Clements, P. "Features of Architecture Description Languages". In: Software Technology Conference, Salt Lake City, Abril 1995.

[Luc96] Luckham D. C. "Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events". In: Proceedings of DIMACS Partial Order Methods Workshop IV, Princeton University, Julho 1996.

[Lik32] Likert, R. "A Technique for the Measurement of Attitudes", In: Archives of Psychology, vol. 22 issue 140, pp. 1-55.

[Mag95] Magee, J. "Specifying Distributed Software Architectures". In: Proceedings ESEC ’95, SpringerVerlag, vol. 989, 1995, pp. 137-153.

[Mai06] Maia, N. "Odyssey-MDA: Uma abordagem para transformação de modelos". Dissertação de Mestrado, COPPE UFRJ, Rio de Janeiro, 2006, pp. 27-82.

[Mat02] Matinlassi, M.; Niemelä, E.; Dobrica, L. “Quality-driven architecture design and quality analysis method, A revolutionary initiation approach to a product line architecture”. Helsinque: Technical Research Centre of Finland, 2002, Edição 1, pp. 84-88.

[Men02] Mendes, A. “Arquitetura de Software: desenvolvimento orientado a arquitetura”. Rio de Janeiro: Editora Campus, 2002, Edição 1, pp. 45-52.

[Nol07] Noll, R. P. “Rastreabilidade Ontológica Sobre o Processo Unificado”. Dissertação de Mestrado Porto Alegre Fac. De Informática, PUCRS, 2007, 78p.

[Oat06] Oates, B. “Researching Information Systems and Computing: Design

and Creation”. Londres: SAGE Publications Ltd, 2006, Edição 1, pp. 108-124.

[Oh07] Oh, Y.; Lee, D.; Kang, S.; Lee, J. "Extend Architecture Analysis Description Language for Software Product Line Approach in Embedded Systems". In: Proceedings of the 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign, Washington, DC, USA, 2007, pp. 87-88.

[Pfl98] Pfleeger, S. L. “Software Engineering: Theory and Practice”. Amsterdam: Prentice Hall, vol. 1, 1998, 659p.

[Poh07] Pohl, K.; Metzger, A. “Variability Management in Software Product Line Engineering”. In: ICSE Companion, Washington, DC, USA. 29th International Conference, 2007, pp. 186-187.

[Pre05] Pressman, S. R. “Software Engineering A PRACTITIONER'S APPROACH”. São Paulo: Makron Books do Brasil Ltda, 2005, Edição 5, 450p.

[Pri91] Prieto-Diaz, R.; Arango, G. "Domain Analysis Concepts and Research Directions & Domain Analysis and Software Systems Modeling”. IEEE Computer Society Press, 1991.

[Roy70] Royce, W. "Managing the Development of Large Software Systems". In: ICSE '87 Proceedings of the 9th international conference on Software Engineering, 1987, pp. 328-338.

[Sam97] Sametinger, J., “Software Engineering with Reusable Components”. Nova Iorque: Springer-Verlag,1997.

[Sha65] Shapiro, S. S.; Wilk, M. B. "Analysis of variance test for normality (complete samples)". Oxford Journals Biometrika, 1965, vol. 52, pp. 591–611.

[Sha96] Shaw, M.; Garlan, D. "Software Architecture – Perspectives on an Emerging Discipline". Pittsburgh: Prentice-Hall, 1996.

[Sho03] Sholom, C. "Predicting When Product Line Investment Pays". In: Workshop at 23rd International Conference on Software Engineering, 2003, pp. 5-15.

[Sil09] Silva, F. S. "Uso de Representação de Conhecimento para Documentação de Metodologias Ágeis". Dissertação de Mestrado,

PUCRS, 2009, 152p.

[Sjo02] Sjober, D.; Anda, B.; Arisholm, E.; Dyba, T.; Jorgensen, M.; Karahasanovic, A.; Koren, E.; Vokác, M. “Conducting Realistic Experiments in Software Engineering”. In: Proc. ISESE Washington, DC, USA, 2002, pp. 17-26.

[Sof09a] SOFTEX “Melhoria de Processos de Software Brasileiro”. Capturado em: http://www.softex.br/mpsbr, Novembro 2009.

[Sof09b] Software Engineering Institute - Carnegie Mellon "Capability Maturity Model Integration". Capturado em: http://www.sei.cmu.edu/cmmi, Novembro 2009.

[Sil01] Silva, L. F.; Paula, V. C. “Um Meta-Modelo para Especificação de Arquiteturas de Software em Camadas”. In SBES - Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 2001, pp. 132-144.

[Siy99] Siy, H.; Mockus, A. “Measuring Domain Engineering Effects on Software Change Cost”. In IEEE Computer Society, 1999, pp. 304-311.

[Tra02] Travassos, G. H.; Gurov, D.; Amaral, E. A. G. G. "Introdução à Engenharia de Software Experimental". Rio de Janeiro, Editora COPPE/UFRJ, Relatório Técnico ES-590/02-Abril. Programa de Engenharia de Sistemas e Computação, 2002.

[Uml10] UML. “Unified Modeling Language”. Capturado em: http://www.uml.org, Janeiro 2010.

[Var02] Varoto, A. C. “Visões em Arquitetura de Software”, Dissertação, USP, 2002, 108p.

[Vas07] Vasconcelos, A. P. V. “Uma Abordagem de Apoio à Criação de Arquiteturas de Referência de Domínio Baseada em Sistemas Legados”, Tese, COPPE/UFRJ, 2007, 284p.

[Wer05] Werner, C. M. L.; Braga, R. M. M. "A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes". Rio de Janeiro: Ciência Moderna 2005, 1ª Edição, pp. 55-90.

[Wer09] Werner, C.; et. al. "Projeto Odyssey - Laboratório de Engenharia de Software, COPPE Universidade Federal do Rio de Janeiro",

Capturado em: http://reuse.cos.ufrj.br, Julho 2009.

[Wik10] Wikipedia, E. L. “Psicometria Psicologia“. Capturado em: http://pt.wikipedia.org/wiki/Psicometria (Psicologia), Janeiro 2010.

[Wik10a] Wikipedia, E. L. “Experiência de Hawthorne”. Capturado em: http://pt.wikipedia.org/wiki/Experiência_de_Hawthorne, Janeiro 2010.

[Wik10b] Wikipedia, E. L. “Variância Estatística”. Capturado em: http://pt.wikipedia.org/wiki/Variância, Janeiro 2010.

[Wik10c] Wikipedia, E. L. “Padrões de Projeto”. Capturado em: http://pt.wikipedia.org/wiki/Singleton, Janeiro 2010.

[Wik10d] Wikipedia, E. L. “Assistente (software)”. Capturado em: http://pt.wikipedia.org/wiki/Singleton, Janeiro 2010.

[Wit09] Wittgenstein. L. “O Pensador” Capturado em: http://www.pensador.info/, Dezembro 2009.

[Woh00] Wohlin, C; Runeson, P.; Höst, M.; et al. "Experimentation in Software Engineering, an Introduction", Capturado em: http://books.google.com.br/books?id=nG2UShV0wAEC&printsec=frontcover&dq=Experimentation+in+Software+Engineering:+an+introduction#v=onepage&q=&f=false, Novembro 2009.

APÊNDICE A – MENSAGEM ENVIADA AOS PARTICIPANTES

Mensagem eletrônica enviada aos participantes do experimento

Prezado Participante,

Estamos realizando um experimento relacionado a minha dissertação de mestrado junto ao Programa de Pós-Graduação em Ciência da Computação da PUCRS, cujo título é Arquiteturas de Referência para Domínios de Aplicações: Uma Abordagem para a Documentação Através de Linguagens de Descrição Arquiteturais.

Através da execução deste experimento, pretendo avaliar a eficiência e a eficácia de uma abordagem para a descrição de Arquiteturas de Referência (AR) para domínios de aplicações, através da utilização de uma Linguagem de Descrição Arquitetural. Também será avaliada a utilidade e usabilidade desta abordagem, no momento da criação de uma aplicação a partir de uma Arquitetura de Referência de um domínio de aplicações.

A condução deste experimento está dividida em duas etapas: 1ª) descrição de uma Arquitetura de Referência e 2ª.) instanciação de uma Arquitetura de Software para um domínio.

A identidade dos indivíduos participantes do experimento será mantida em sigilo. O perfil necessário do participante deste estudo inclui experiência mínima em documentação, desenvolvimento e criação de arquiteturas de software.

A previsão para execução das atividades previstas no experimento é de aproximadamente 1 hora.

Sendo assim, qual seria a sua disponibilidade em termos de dia(s) e horário(s) para realizar o experimento?

Desde já, agradecemos a sua atenção e colaboração.

Eduardo Brandes

APÊNDICE B – FORMULÁRIO DE CONSENTIMENTO

DECLARAÇÃO

Eu declaro ter mais de 18 anos de idade e concordo em participar de um estudo conduzido por Eduardo Brandes, como parte das atividades para a obtenção do título de mestre em Ciência da Computação junto ao Programa de Pós-Graduação em Ciência Computação da PUCRS.

OBJETIVO

Este estudo tem como objetivo, avaliar a viabilidade da descrição de Arquiteturas de Referência para domínios de aplicações, através da utilização de uma Linguagem de Descrição Arquitetural estendida.

PROCEDIMENTO

Este experimento acontecerá em duas etapas: a primeira relacionada a geração da documentação de uma Arquitetura de Referência e outra relacionada a instanciação de uma Arquitetura de Software.

Para os indivíduos que irão realizar o experimento com o apoio da abordagem proposta nesta pesquisa, será apresentado um treinamento abordando os seguintes temas: conceitos básicos de reúso, modelagem domínio em termos de *features*, componentes, apresentação da ADL, ambiente ABLE, ambiente Odyssey (criação e modelagem de domínios e instanciação de arquiteturas a partir de Arquiteturas de Referência), estilo arquitetural camadas e *template* de Arquitetura de Referência proposto.

Para os indivíduos que irão executar o experimento sem o apoio da abordagem proposta nesta pesquisa, será apresentado um treinamento abordando os seguintes temas: conceitos básicos de reúso, modelagem domínio em termos de *features*, componentes, formas de documentação de Arquiteturas de Software, ambiente Odyssey (criação e modelagem de domínios e instanciação de arquiteturas baseadas em modelo de *features*).

O treinamento tem como objetivo realizar um nivelamento do conhecimento em torno dos conceitos subjacentes envolvidos nesta pesquisa. Além disso, apresentam as características e notação gráfica deste tipo de modelagem.

Durante o treinamento, será explicado também como proceder o preenchimento da tabela de métricas de execução. Tanto para coleta do tempo gasto na execução das atividades de criação, quanto as métricas relacionadas a precisão da descrição gerada. Ao final do treinamento, será entregue a todos os participantes um modelo de componentes do domínio, utilizado no experimento, e uma especificação contendo requisitos não-funcionais, a qual será utilizada pelos indivíduos para instanciar uma arquitetura na segunda etapa. Para os participantes que irão executar o experimento com o apoio da abordagem, será distribuído, além do modelo de componentes do domínio, e a especificação contendo requisitos não-funcionais, um glossário explicando os constructos e propriedades que formam a ADL utilizada.

Na segunda etapa os participantes irão realizar a instanciação de uma arquitetura para o domínio. Para a realização da segunda etapa, assim como na primeira, metade dos participantes vai seguir a abordagem proposta nesta pesquisa (instanciação baseada em Arquitetura de Referência), e outra metade fará a instanciação sem o apoio da abordagem (instanciação baseada em *features*).

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a forma de documentar uma Arquitetura de Referência, a qual o

experimento se propõe a conduzir.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software com foco em reúso.

EQUIPE:

PESQUISADOR RESPONSÁVEL

BEL. Eduardo Brandes - Programa de Pós-Graduação em Ciência da Computação – PUCRS

PROFESSOR RESPONSÁVEL

Profa. Dra. Ana Paula Terra Bacelo - Programa de Pós-Graduação em Ciência da Computação - PUCRS

Nome (em letra de forma):_____

Assinatura:_____ Data:_____

APÊNDICE C – FORMULÁRIO DE CARACTERIZAÇÃO DA EXPERIÊNCIA DOS PARTICIPANTES

CARACTERIZAÇÃO DO ARQUITETO

Nome: _____

Formação Acadêmica:

( ) Doutorado

( ) Mestrado

( ) Especialização

( ) Graduação

( ) Técnico

( ) Outra: _____

Formação Geral:

Qual é a sua experiência com a documentação de Arquiteturas de software com o foco em Engenharia de Domínio ou Linha de Produtos *na prática*? (marque aqueles itens que melhor se aplicam)

( ) nunca desenvolvi Arquiteturas de software baseado em reutilização.

( ) tenho desenvolvido Arquiteturas software com foco em artefatos reutilizáveis para uso próprio.

( ) tenho desenvolvido arquiteturas de software com foco em artefatos reutilizáveis como parte de uma equipe, relacionado a um curso.

( ) tenho desenvolvido arquiteturas de software com foco em artefatos reutilizáveis como parte de uma equipe, na indústria.

Considerando a escala abaixo, qual a sua experiência na construção de arquiteturas para Linha de Produto de Software?

1 ( ) nenhuma 2 ( ) muito pouca 3 ( ) pouca 4 ( ) muita 5 ( ) sou experiente

Como é feita a documentação de arquiteturas de software em sua empresa ou universidade?

_____

_____

Inclua o número de semestres ou número de anos de experiência relevante. (e.g., “Eu trabalhei por 10 anos como programador na indústria”)

_____

_____

Você já fez reúso baseado em arquitetura? Qual parte da arquitetura foi reutilizada?

_____

_____

Você já utilizou ou utiliza alguma técnica para apoio ao desenvolvimento com foco em reutilização?

( ) Não.

( ) Sim. Indique a(s) técnica(s)?

Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:

1 - nenhum

2 - estudei em aula ou em livro

3 - pratiquei em 1 projeto em sala de aula

4 - usei em 1 projeto na indústria

5 - usei em vários projetos na indústria

Questão	1	2	3	4	5
Experiência em projeto de arquiteturas de sistemas					
Experiência em projeto de arquiteturas utilizando padrões arquiteturais					
Domínio de Conceitos de reutilização de software					
Experiência com Unified Modeling Language (UML)					

Experiência em Contextos Diferentes:

Esta seção será utilizada para compreender o quão familiar você está com o domínio que será utilizado para as atividades durante o experimento.

Por favor, indique o grau de experiência nesta seção seguindo a escala de 3 pontos abaixo:

1 - Eu não tenho familiaridade com a área. Eu nunca fiz isto.

2 - Eu utilizo isto algumas vezes, mas não sou um especialista.

3 - Eu sou muito familiar com esta área. Eu me sentiria confortável fazendo isto.

Área: Telefonia Celular: _____

APÊNDICE D – TREINAMENTO APRESENTADO AOS PARTICIPANTES DO EXPERIMENTO

Programa de Pós-Graduação em Ciência da Computação PPGCC-PUCRS

Treinamento para Experimento em Engenharia de Software

Aluno: Eduardo Brandes
Orientadora: Dra. Ana Paula Terra Bacelo

Agenda

- Introdução
- Visão Geral
- Engenharia de Software Experimental
  - Objetivo Global
  - Objetivo do Estudo
- Ferramentas (com abordagem)
  - *Template* de Arquitetura de Referência
  - ADL ACME
  - ABLE
  - Odyssey
- Ferramentas (sem abordagem)
  - Odyssey
  - Modelagem de *Features*
  - Modelagem de Componentes
  - Jude

Introdução – Conceitos Subjacentes

- **Métodos de Apoio ao Reuso: desafios**

- **Engenharia de Domínio (ED)**
  - Identificação de conceitos em um domínio
  - Geração de componentes reutilizáveis
  - Criação de Arquiteturas de Referência
  - **Vantagem**
    - Estabelece uma Linguagem Específica para o Domínio (DSSL) Ex. SQL X Álgebra Relacional

- **Engenharia de Aplicação (EA)**
  - Instanciação de aplicações

Visão Geral - Estabelecendo Vocabulário Comum

- Arquiteturas de Software (AS)
  - Importância
  - Caminhos para Documentação
- Linguagens de Descrição Arquiteturais (ADL)
  - *Features* básicas desejadas
  - ADLs estudadas
- Arquiteturas de Referência (AR)
  - O que são?
  - Quais as diferenças em relação a AS

Engenharia de Software Experimental

Estudos de Caso
Comparar projetos similares

Pesquisa de Campo
Método em Retrospectiva

Experimento
Hipóteses
Variáveis
Ambiente Controlado

Engenharia de Software Experimental

- Objetivo Global

"Comparar, em um processo de desenvolvimento com foco em Reuso, o esforço e a precisão da documentação e instanciação de Arquiteturas de Software, a partir de uma Arquitetura de Referência."

Engenharia de Software Experimental

- Objetivo do Estudo

Comparar, em um processo de desenvolvimento de um domínio de aplicações, a instanciação de Arquiteturas de Software a partir de uma Arquitetura de Referência documentada por ADL, com a instanciação baseada em *features*,

Com o propósito de caracterizar o tempo gasto no uso da documentação por ADL, criada pela abordagem,

Com foco no esforço e precisão,

Sob o ponto de vista do Arquiteto de Software ou Engenheiro de Domínio,

No contexto da criação de uma Arquitetura de Software para um sistema desenvolvido por estudantes, no domínio de telefonia celular

Ferramentas (Com abordagem)

Ambiente de Reuso e Ambiente Para Manipulação de ADL

Odyssey

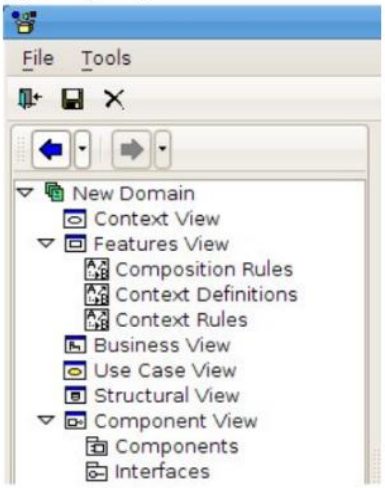
Framework onde modelos conceituais, arquiteturas de software, e modelos implementacionais são especificados para domínios de aplicação previamente selecionados

Plataforma JAVA.

ABLE

Ambiente para a modelagem de arquiteturas de Software desenvolvido por pesquisadores do SEI-CMU

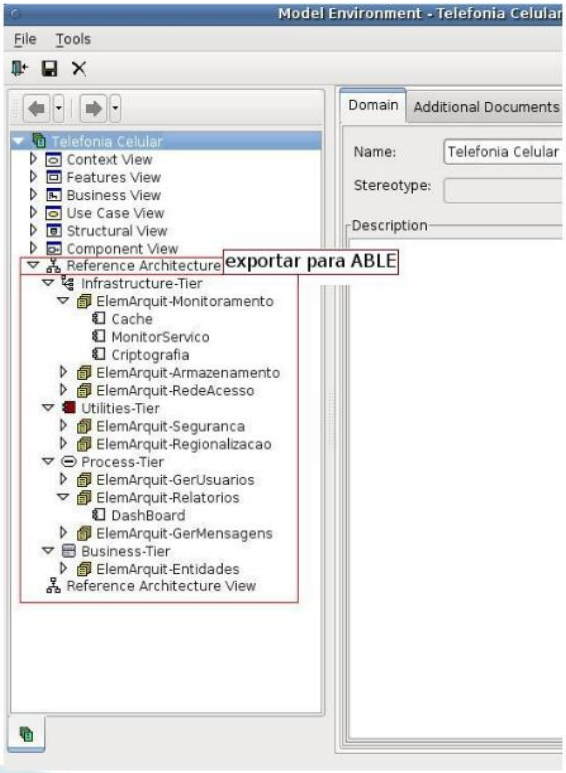
Construído com base no Eclipse



Ferramentas - Protótipo

Extensão Realizada

- Alterações na árvore de elementos semânticos
- Ampliação da capacidade de expressão para viabilizar a representação de AR
- Inclusão de elementos para representação do estilo arquitetural Camadas [3]
- Opção para exportação no formato ACME estendido.



Ferramentas – ADL - Código Gerado

XML Gerado no Ambiente de Reuso

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<AcmeModel>
  <family name="NewDomain">
    <userdata key="connection-patterns">
      <data>
        <portConnectionRule connector="Connector" name="">
          <association componentType="Component" portName="p" portType="Port" roleName="r" roleType="Role"/>
          <association componentType="Component" portName="p" portType="Port" roleName="r" roleType="Role"/>
        </portConnectionRule>
      </data>
    </userdata>
    <system name="prototype">
      <userdata key="connection-patterns">
        <data>
          <portConnectionRule connector="Connector" name="">
            <association componentType="Component" portName="p" portType="Port" roleName="r" roleType="Role"/>
            <association componentType="Component" portName="p" portType="Port" roleName="r" roleType="Role"/>
          </portConnectionRule>
        </data>
      </userdata>
    </system>
  </family>
</AcmeModel>
```

Ferramentas – Descrição AR

Código Gerado

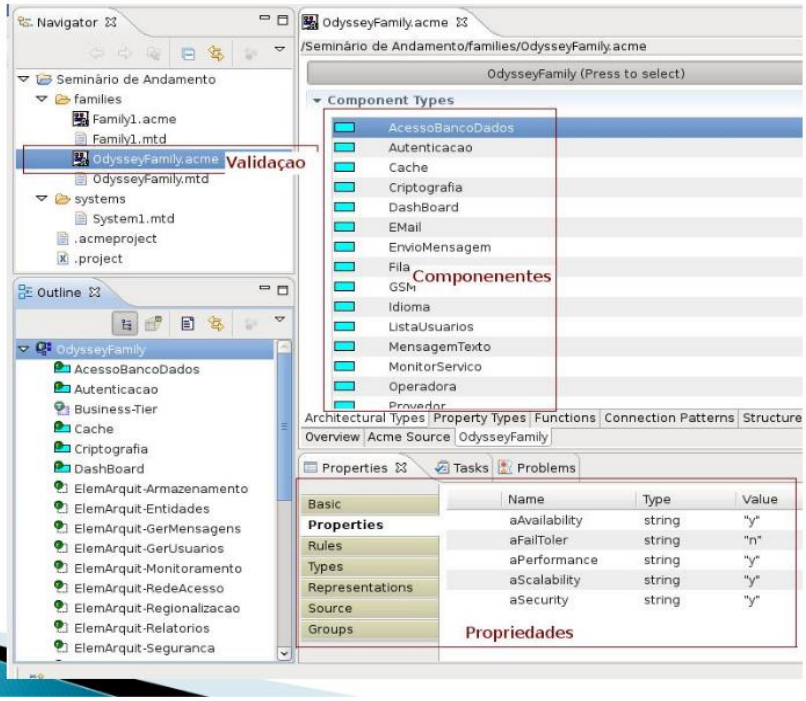
- **Definição de Componentes**
  - Encryption
  - DBAccess
  - CodeBarValidator
  - ETL
  - UserAuth
- **Definição de Elementos Arquiteturais**
  - Security
  - DataWarehouse
  - PersistenceObjects
  - FTPServer
  - TransactionalDB

```
Family NewDomain = {
    Element Type Security = {
        Property aTier : string = "process";
    }
    Element Type DataWarehouse = {
        Property aTier : string = "infrastructure";
    }
    Element Type PersistenceObjects = {
        Property aTier : string = "business";
    }
    Element Type FTPServer = {
        Property aTier : string = "infrastructure";
    }
    Element Type TransactionalDB = {
        Property aTier : string = "infrastructure";
    }
    Element Type ValidationRules = {
        Property aTier : string = "utilities";
    }
    Component Type Encryption = {
        Property aElementType : string = "Security";
    }
    Component Type DBAccess = {
        Property aElementType : string = "PersistenceObjects";
        Property aMandatory : boolean = true;
    }
    Component Type CodeBarValidator = {
        Property aElementType : string = "ValidationRules";
    }
    Component Type ETL = {
        Property aElementType : string = "DataWarehouse";
    }
    Component Type UserAuth = {
        Property aElementType : string = "Security";
    }
}
```

Ferramentas - ABLE

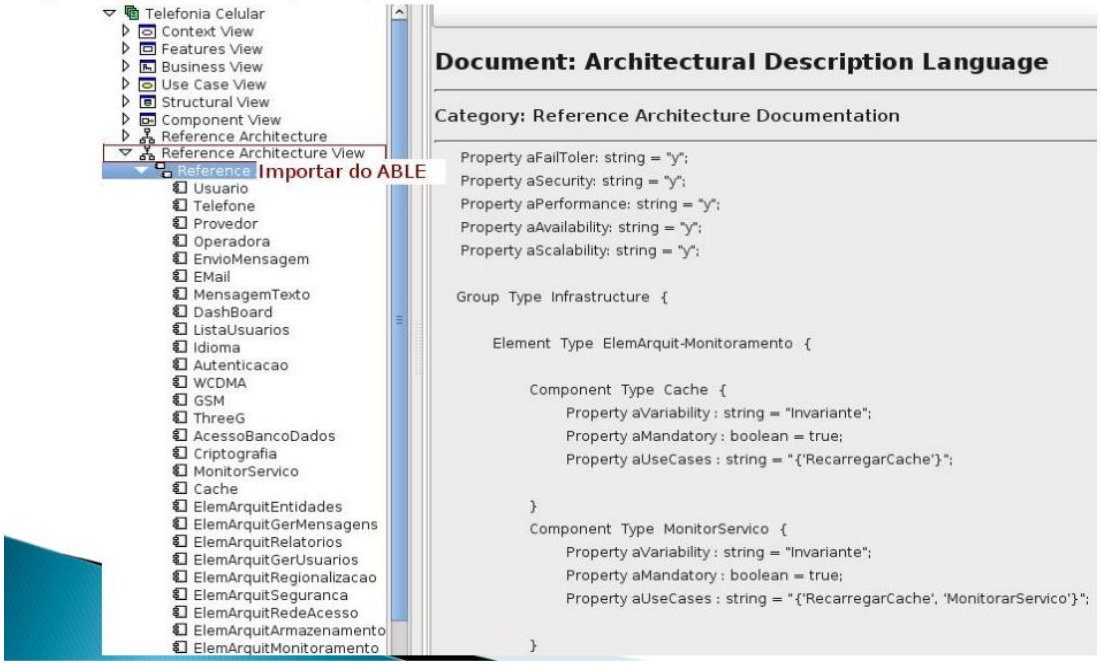
Ambiente para Manipulação de ADL

- Complementação
- Validação



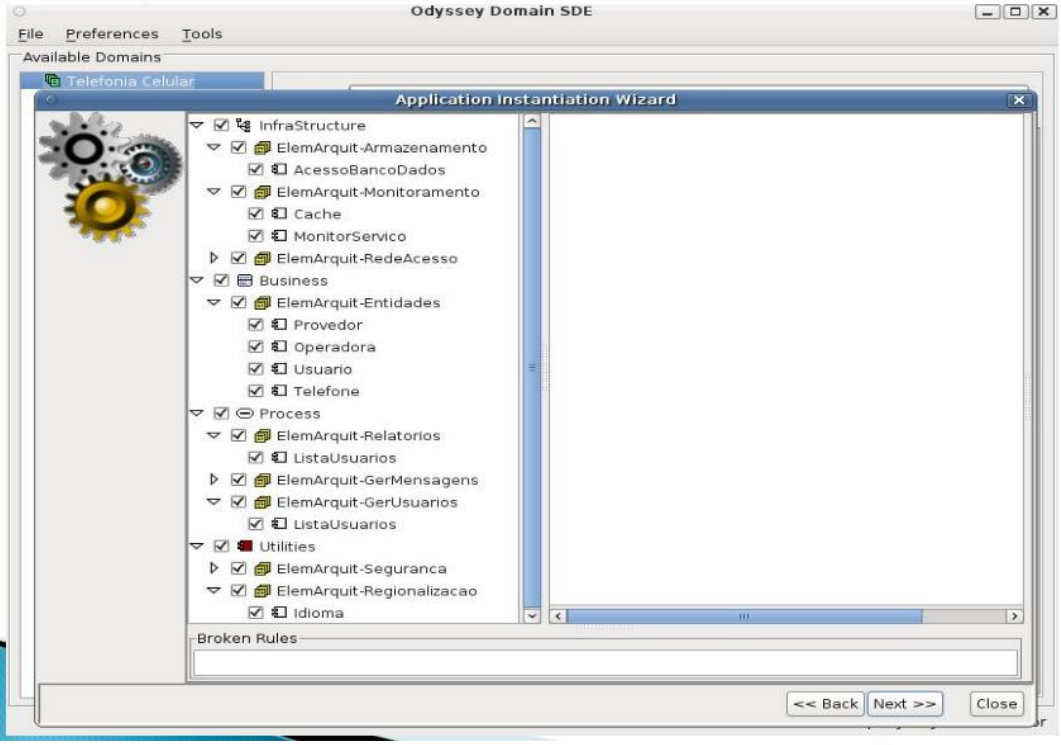
Ferramentas

Importação da Descrição a partir do ABLE



Ferramentas

Instanciação de Arquitetura Baseada em Arquitetura de Referência

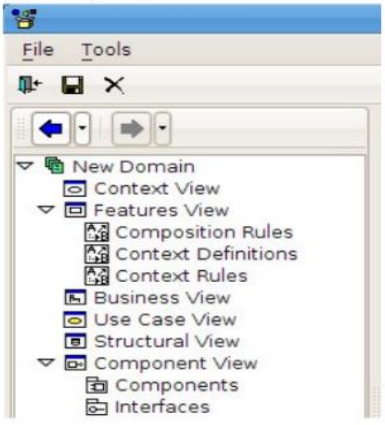


Ferramentas (sem abordagem)

Ambiente de Reuso e Ambiente Para Manipulação de ADL

Odyssey

- *Framework* onde modelos conceituais, arquiteturas de software, e modelos implementacionais são especificados para domínios de aplicação previamente selecionados
- Plataforma JAVA.



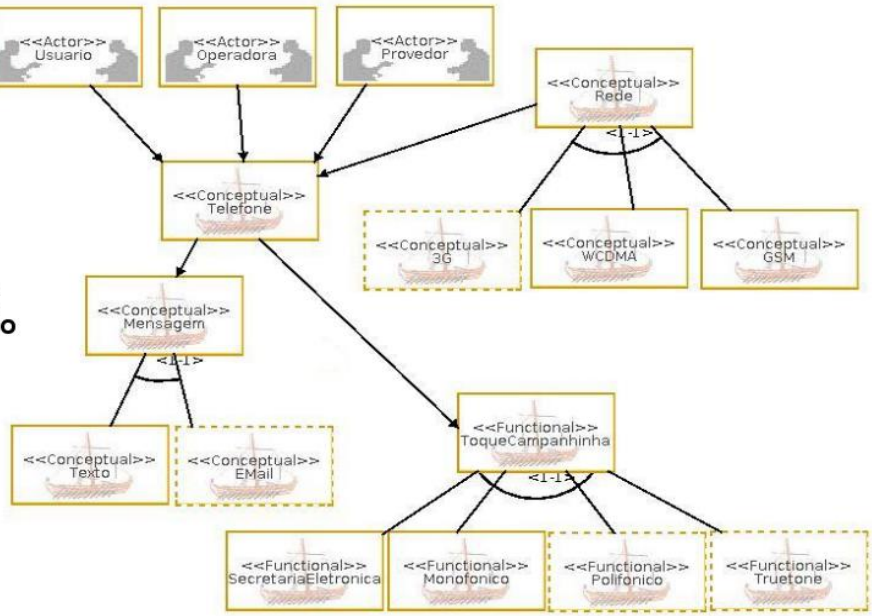
JUDE

- Ambiente para a modelagem UML
- Descrição textual

Modelagem de Features

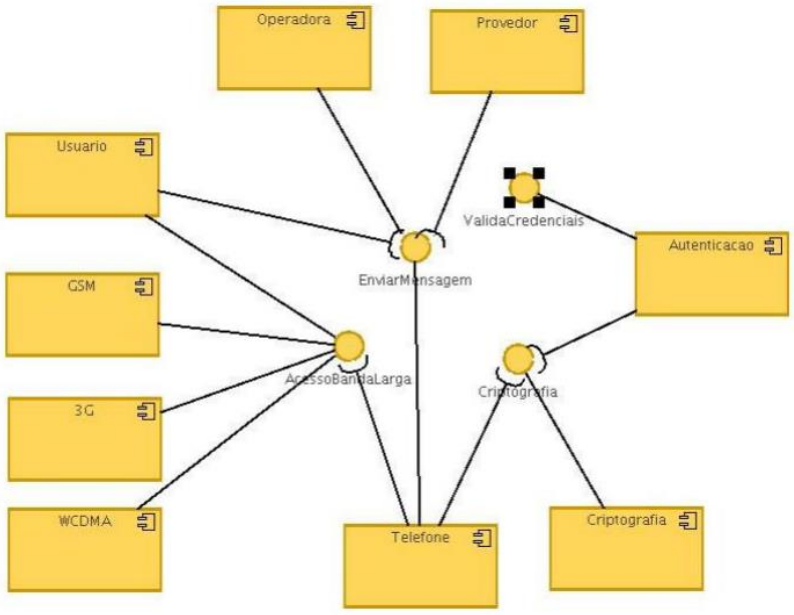
FEATURES

- **Conceituais**
- **Funcionais**
- **Atores**
- **Tecnologia de Implementação**
- **Contextuais**
- **Ambiente Operacional**



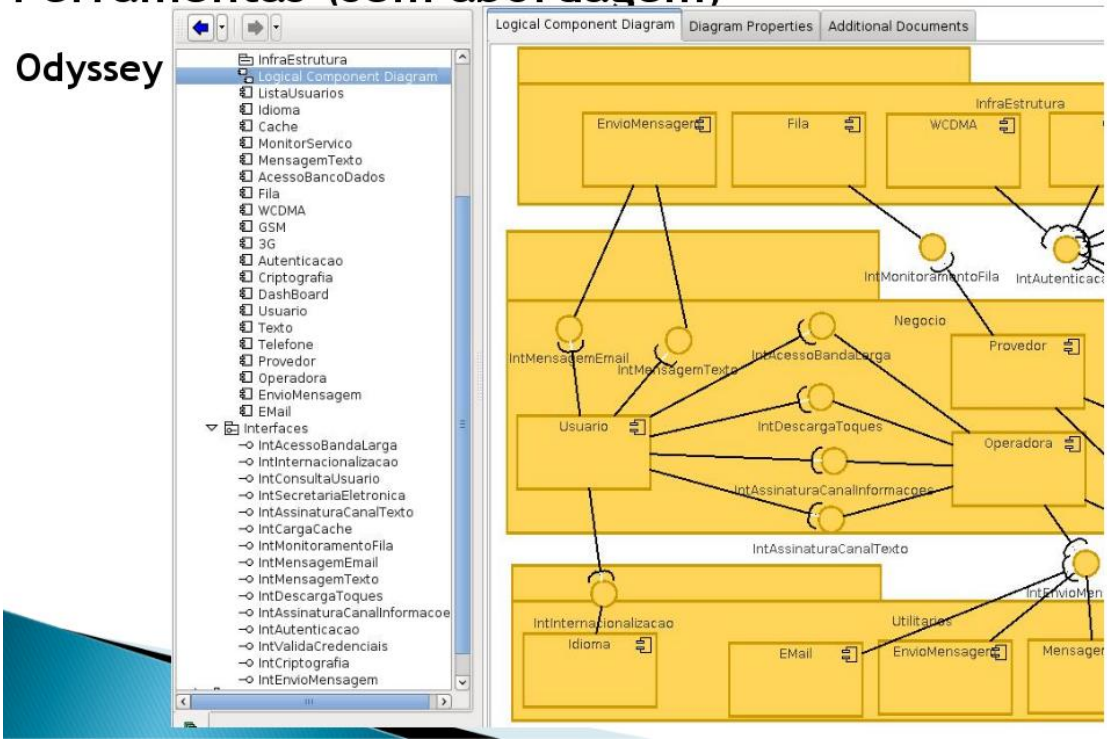
Modelagem de Componentes

- **Serviços**
- **Interfaces**
  - Providas
  - Requeridas



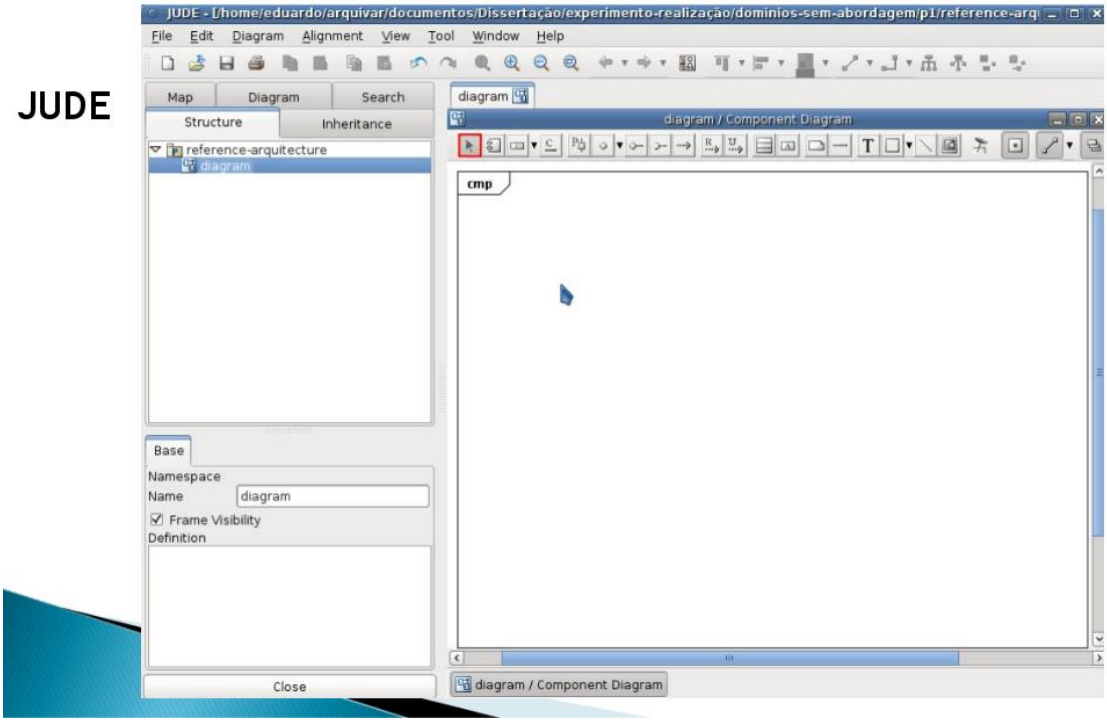
Ferramentas (sem abordagem)

Odyssey



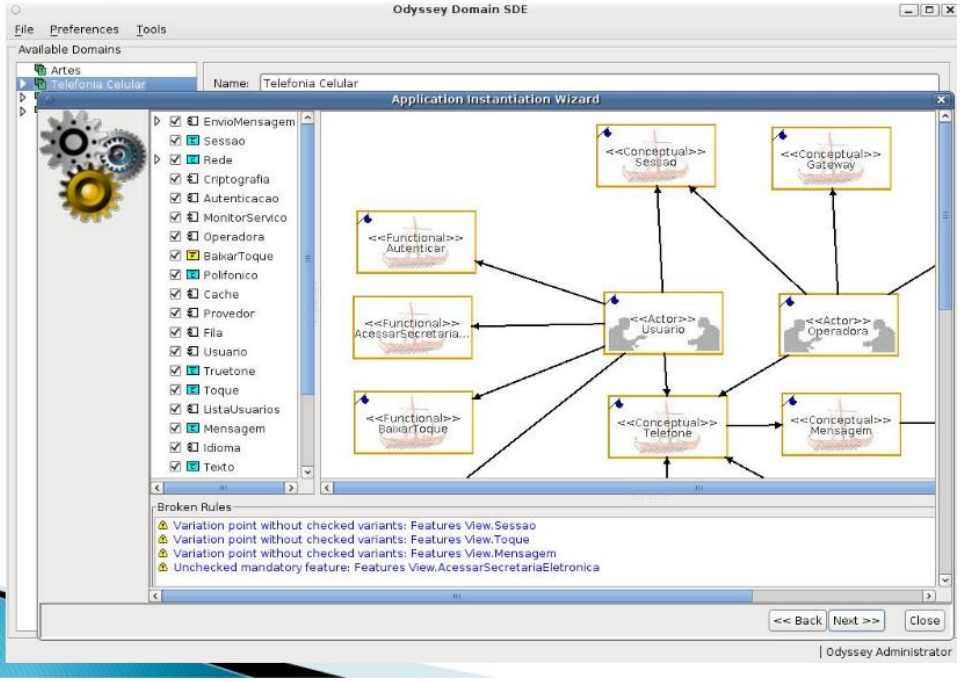
Ferramentas (sem abordagem)

JUDE



Ferramentas

Instanciação de Arquitetura Baseada features



Referências

[1] Kang, C., Cohen, S., Hess, J., Novak, W., Peterson, A., Feature-Oriented Domain Analysis. Feasibility Study: Software Engineering Institute, Pittsburgh CMU/SEI-90-TR-21, 1990.

[2] Bacelo, A., Becker, K., Werner, C., Um Processo de Engenharia de Domínio com foco no Projeto Arquitetural Baseado em Componentes: COPPE/UFRJ, 2004.

[3] Shaw Mary, Garlan David: Software Architecture – Perspectives on an Emerging Discipline, Prentice-Hall, 1996.

Questionamentos

APÊNDICE E – TUTORIAL PARA DESCRIÇÃO COM APOIO DA ABORDAGEM

Pontifícia Universidade Católica do Rio Grande do Sul	ENGENHARIA DE SOFTWARE EXPERIMENTAL Documentação de Arquiteturas de Referência (AR) através de ADL, e instanciação baseada em Arquiteturas de Referência Mestrando: Eduardo Brandes

Este documento apresenta os passos necessários para condução do experimento utilizando a abordagem para a documentação da Arquitetura de Referência.

Atividade 1: Descrição de Arquitetura de Referência por ADL

1. Registrar horário de início da atividade
2. Abrir na ferramenta Odyssey o domínio “experimento-com-arq-ref.domain”.
3. Visualizar AR, criar descrição em ADL e exportar para ABLE.
4. Abrir no ambiente ABLE o arquivo “OdysseyFamily.acme”.
5. Complementar e compilar a descrição.
6. Na ferramenta Odyssey, importar o arquivo “OdysseyFamily.acme”.
7. Registrar o horário de término da atividade.

Atividade 2: Instanciação baseada em Arquitetura de Referência

1. Na ferramenta Odyssey, salvar e fechar o domínio editado na atividade 1, e criar uma nova aplicação a partir deste domínio.
2. Abrir documento HTML com a representação da ADL, gerado no passo 6.
3. Selecionar opção para instanciação baseada em AR.
4. Selecionar componentes para criação de uma arquitetura compatível com a

especificação de requisitos fornecida.

5. Finalizar *wizard* de criação e abrir aplicação.
6. Preencher o questionário de avaliação

APÊNDICE F – TUTORIAL PARA DESCRIÇÃO SEM APOIO DA ABORDAGEM

Pontifícia Universidade Católica do Rio Grande do Sul	ENGENHARIA DE SOFTWARE EXPERIMENTAL Documentação de Arquiteturas de Referência baseada em componentes e instanciação baseada em *features* Mestrando: Eduardo Brandes

Este documento apresenta os passos necessários para condução do experimento sem a utilização da abordagem para a documentação da Arquitetura de Referência.

Atividade 1: Descrição de Arquitetura baseada em componentes

1. Registrar horário de início da atividade.
2. Na ferramenta Odyssey abrir arquivo “experimento-sem-arq-ref.domain”.
3. Analisar diagrama de componentes.
4. Criar documentação.
5. Registrar horário de término da atividade.

Atividade 2: Instanciação baseada em *features*

1. Na ferramenta Odyssey, fechar o domínio analisado na atividade 1, e criar uma nova aplicação a partir deste domínio.
2. Selecionar opção para instanciação baseada em contexto.
3. Selecionar *features* para a criação de uma arquitetura compatível com a especificação de requisitos fornecida.
4. Finalizar criação e abrir aplicação.

APÊNDICE G – REQUISISTOS PARA INSTANCIAÇÃO DE UMA APLICAÇÃO NO DOMÍNIO DA TELEFONIA CELULAR

ACME INTERATIVIDADES, Inc.

Versão: 1.0
Data: 25/11/2009

Histórico

Versão	Data	Autor/Revisor	Descrição
1.0	25/11/2009	A: Eduardo Brandes	Criação do documento
1.0	29/11/2009	A: Eduardo Brandes	Inclusão das seções escopo e atores
1.0	01/12/2009	A: Eduardo Brandes	Requisitos não-funcionais

Participantes do Projeto

Nome	Função no Projeto	E-mail
Eduardo Brandes	Gerente do Projeto	eduardo@ebrandes.com
Eduardo Brandes	Analista de Sistemas	eduardo@ebrandes.com
Px	Arquiteto de Software	px@mail.com
Ana Bacelo	Sponsor	ana.bacelo@pucrs.br

1. Definição do Escopo

O projeto tem por objetivo desenvolver um sistema que suporte interatividades para celular. Essas interatividades variam desde a venda de conteúdo para celular através de download de toques e papéis de parede, quanto a participação em gincanas, quiz, leilão reverso, assinaturas em canais de texto e serviços de informações diversas. Cada um desses serviços vai ser implementado sob a forma de uma aplicação, e incorporada dentro da plataforma através de parametrização. O projeto aqui proposto será divido em várias etapas, de forma que nessa primeira etapa, será elaborada a arquitetura do sistema, para a realização de provas de conceito.

2. Atores

A tabela abaixo mostra os atores que irão interagir com a plataforma de interatividades.

Ator	Função
Operadora de Telefonia	Utilizar uma ferramenta para consultar os serviços nos quais os usuários estão inscritos.
ACME Interatividades	Prover infra-estrutura tecnológica para suportar os serviços, assim como painel de métricas para monitoramento e controle dos fluxos de dados entre as aplicações.
Clientes	Consumir os serviços disponibilizados a através de seus aparelhos celulares.

3. Requisitos Não-Funcionais

Os requisitos que devem ser alcançados pela arquitetura são:

3.1. Segurança

Cada ator que interagir com a plataforma deve ser devidamente identificado no sistema. Os níveis de acesso devem prever acesso de alto, médio e baixo níveis, e.g., administrador, convidado, operador. Além disso, dever ser montada uma trilha de auditoria que permita listar as operações realizadas por cada usuário durante a sua experiência utilizando qualquer aplicação dentro da família.

É facultado ao usuário, a partir da autenticação em uma aplicação, ter acesso, sem a necessidade de uma nova autenticação, a todas as outras aplicações as quais as credenciais desse usuário têm direito de acesso.

As senhas devem ser armazenadas criptografadas, e as sessões devem expirar em um tempo de *time out* passível de configuração por aplicação, sem a necessidade de reinicializar as aplicações.

3.2. Tolerância a Falhas

Cada aplicação inserida dentro da plataforma deve ter a capacidade de recuperação em casos de pane, seja por falta de energia, *crash* de disco, ou qualquer outro tipo de desligamento não programado. Nestes casos, a aplicação deve prover a capacidade de iniciar em outro dispositivo de *hardware* (nó), exatamente do ponto onde foi interrompida antes da falha.

3.3. Alta Disponibilidade

O volume de mensagens processadas pela plataforma é de cerca de 500 mensagens por segundo. Para atender essa demanda e aumentar à vazão, quando necessário, as aplicações devem estar preparadas para fornecer escalabilidade horizontal, de forma que possam ser incorporadas novas máquinas, para executar outras instâncias de aplicações.

3.4. Ocultamento de Informações

Os diversos sistemas que serão adicionados a plataforma, possuem regras de negócio e requisitos funcionais bem distintos. Dessa forma, é desejável que detalhes da arquitetura passem despercebidos para os desenvolvedores, prevenindo assim a ocorrência de violações arquiteturais, ou seja, implementação de requisitos funcionais sobrepondo requisitos não-funcionais.

APÊNDICE H – FORMULÁRIO PARA PREENCHIMENTO DAS MÉTRICAS DE EXECUÇÃO

Métricas de Execução			
Atividade	**Participante**		
	***Início**	***Fim**	****Precisão**
Descrição da AR			
Instanciação	-----	-----	-----
***Código de Identificação do participante			

** Coluna preenchida pelo participante, com o tempo em minutos*

*** Coluna preenchida pelo pesquisador, com valores de 0,0 até 1;*
Este valor é obtido pela divisão do número total de elementos documentados (pelo participante), pelo total de elementos da Arquitetura de Referência padrão definida pelo arquiteto especialista no domínio. O valor total será fornecido pelo pesquisador durante a execução do experimento.

**** Preenchido pelo participante com código de identificação do participante, fornecido no e-mail de confirmação de aceitação de participação na pesquisa (ex. P1).*

APÊNDICE I – QUESTIONÁRIO PARA AVALIAÇÃO QUALITATIVA DA INSTANCIAÇÃO DE ARQUITETURAS DE SOFTWARE COM APOIO DA ABORDAGEM PROPOSTA

Pontifícia Universidade Católica do Rio Grande do Sul	ENGENHARIA DE SOFTWARE EXPERIMENTAL Instanciação de Arquiteturas de Software baseada em Arquitetura de Referência e ADL Mestrando: Eduardo Brandes

Questionário para Avaliação Qualitativa

(1) Muito Ruim	(2) Ruim	(3) Regular	(4) Boa	(5) Muito Boa

	Q1: Como você considera a *usabilidade* desta técnica para a instanciação de uma arquitetura?
	Q2: Como você considera a *utilidade* desta técnica instanciação de uma arquitetura?

(1) Muito Alto	(2) Alto	(3) Aceitável	(4) Baixo	(5) Muito Baixo

	Q3: Como você considera o *esforço para aprendizagem* da técnica?
	Q4: Como você considera o *esforço para* a instanciação de uma arquitetura utilizando esta técnica?

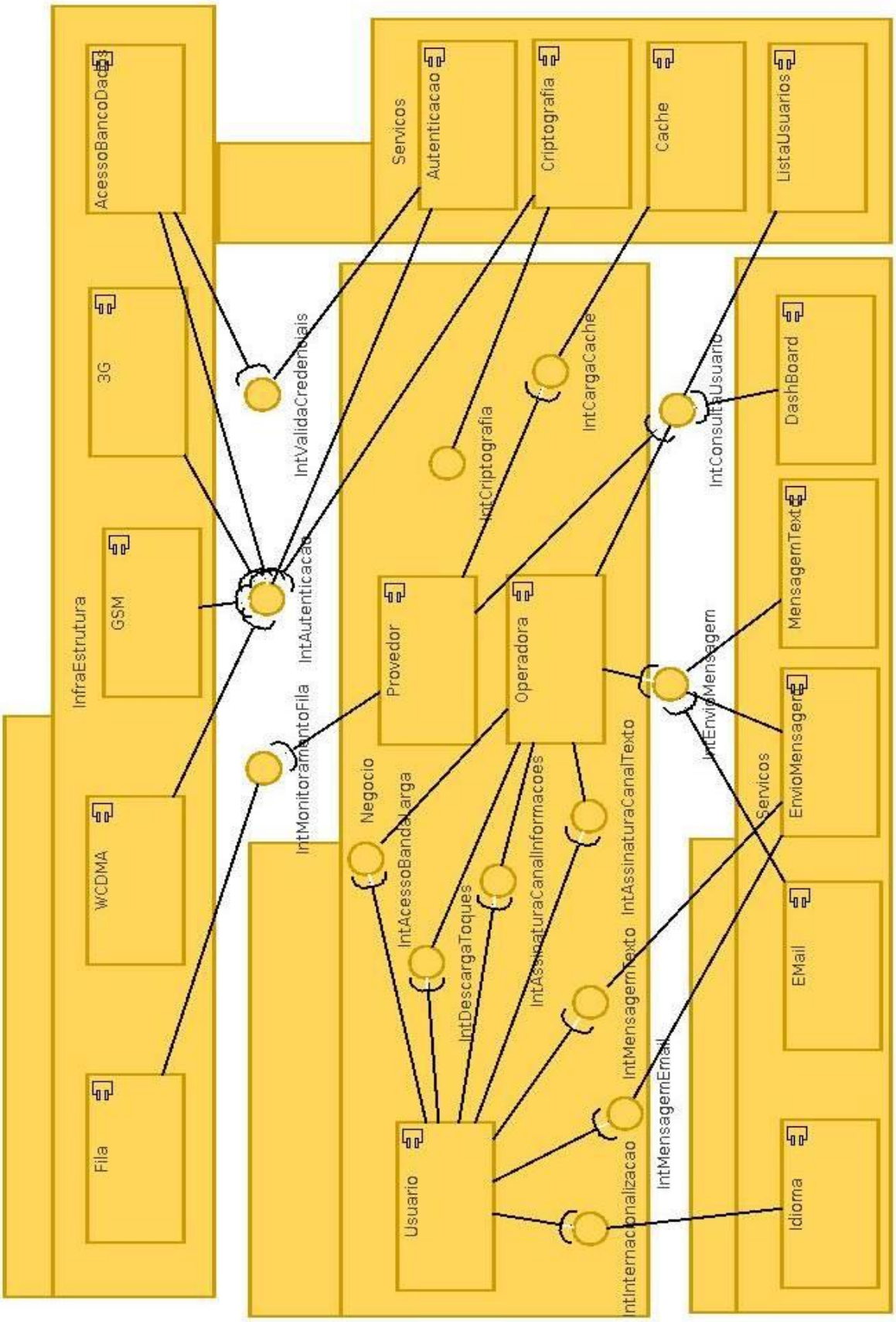
(1) Discordo Plenamente	(2) Discordo	(3) Não Concordo nem Discordo	(4) Concordo	(5) Concordo Plenamente

	Q5: Você *usaria na prática* novamente esta técnica para a instanciação de uma arquitetura?

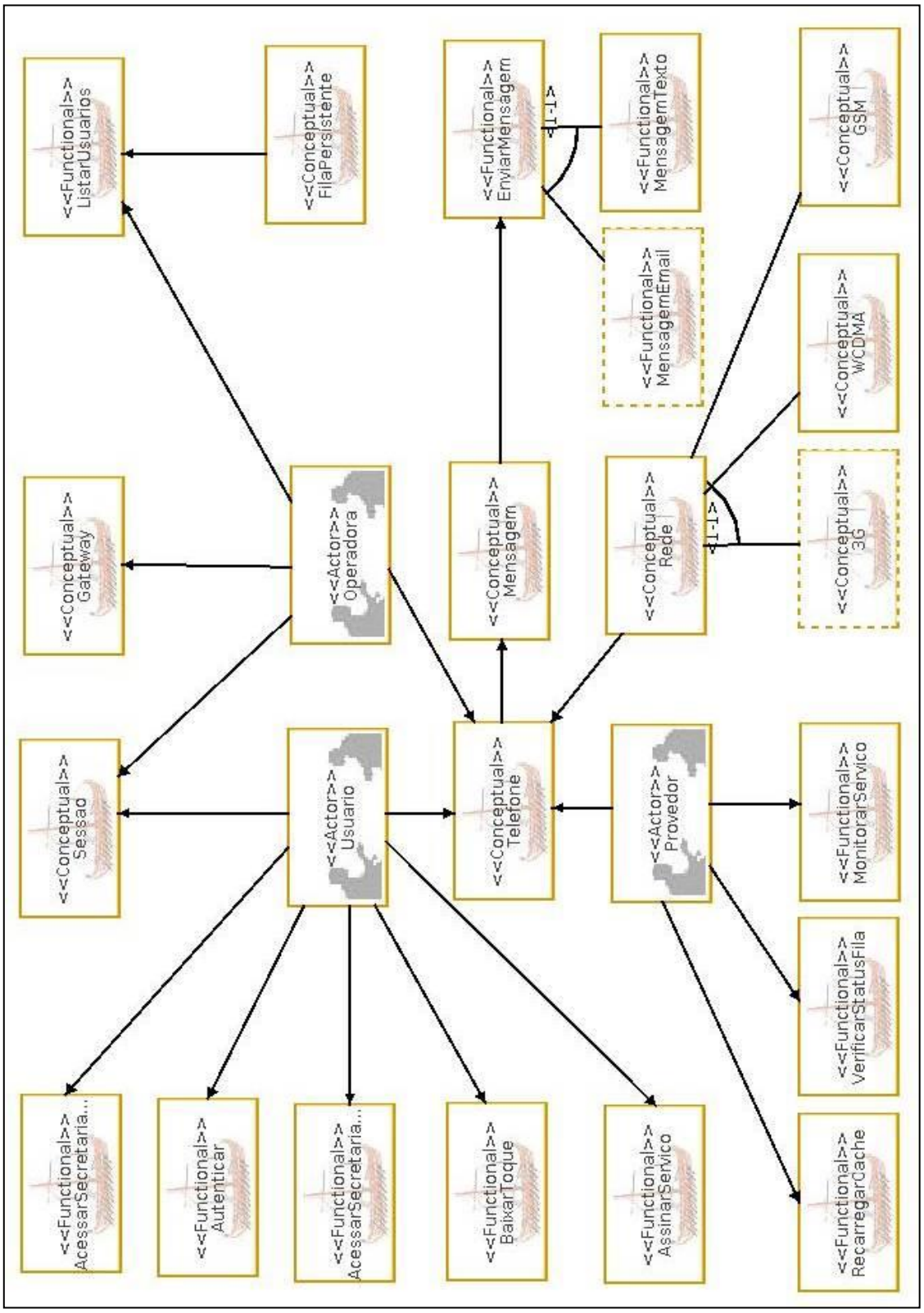
	Q6: A técnica de instanciação de uma arquitetura *atende o que se propôs*?

	Q7: Comente, concordando ou discordando, com a seguinte premissa: “a utilização desta abordagem, de alguma forma compensou o fato do arquiteto não possuir experiência na construção de arquiteturas para o domínio em questão, e esse conhecimento, *suprido pela abordagem*, não fará falta no futuro, pois pode ir sendo obtido gradativamente, inclusive através do próprio uso da abordagem”. _____ _____ _____ _____ _____ _____ _____
	Q8: Comente, concordando ou discordando, com a seguinte premissa: “a utilização desta abordagem, de alguma forma compensou o fato do arquiteto não possuir experiência na construção de arquiteturas para esse domínio, e esse conhecimento, suprido pela abordagem, fará falta no futuro, no momento da criação de uma nova arquitetura”. _____ _____ _____ _____ _____ _____ _____

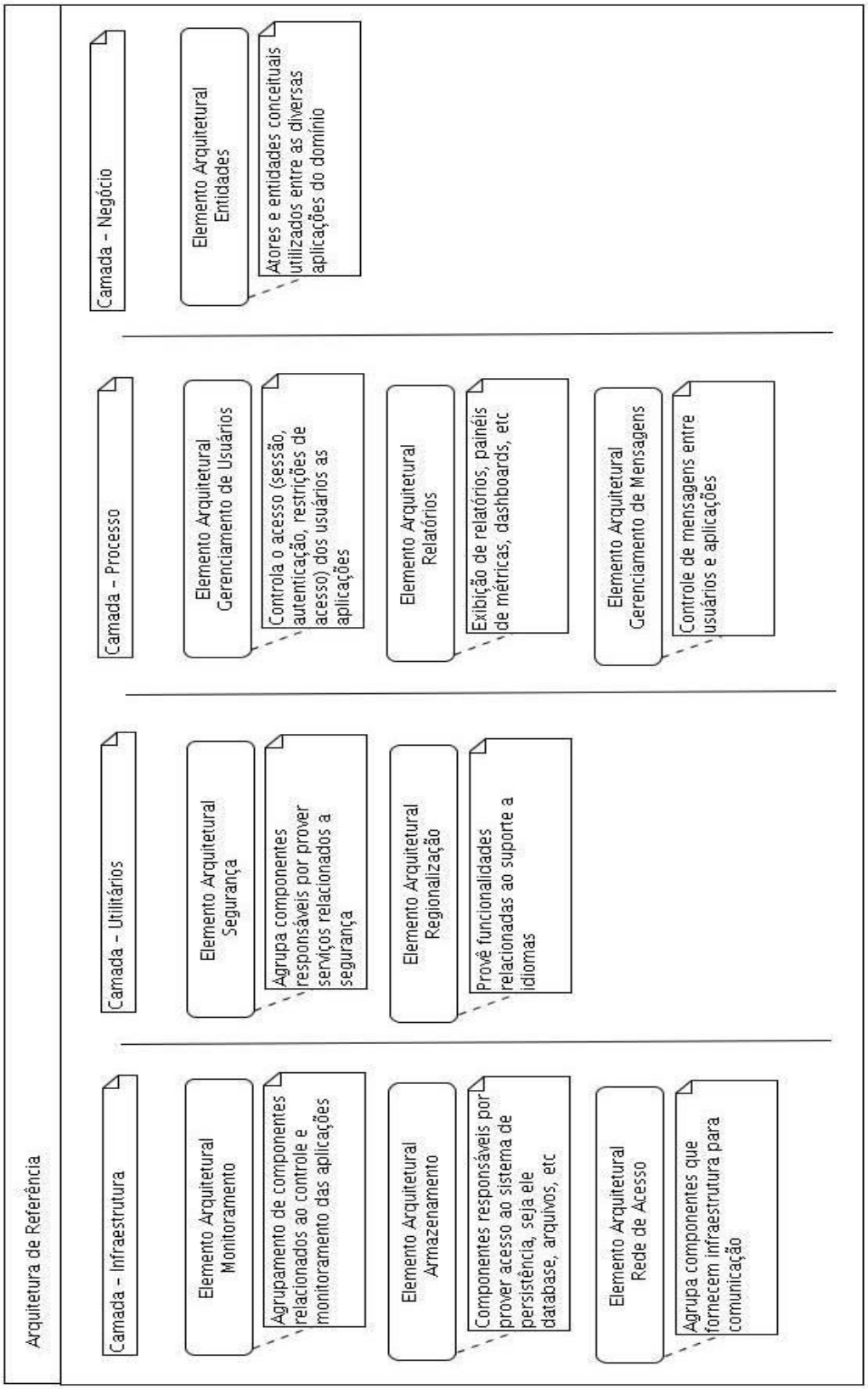
APÊNDICE J – MODELO DE COMPONENTES UTILIZADO NO EXPERIMENTO, PARA DOCUMENTAÇÃO SEM ABORDAGEM



APÊNDICE K – DIAGRAMA DE FEATURES UTILIZADO NO EXPERIMENTO PARA A INSTANCIAÇÃO DE ARQUITETURAS SEM APOIO DA ABORDAGEM



APÊNDICE L – DIAGRAMA DA ARQUITETURA DE REFERÊNCIA DO DOMÍNIO



APÊNDICE M – REPRESENTAÇÃO DE ARQUITETURAS DE REFERÊNCIA PADRÃO PARA O DOMÍNIO DE TELEFONIA CELULAR COM ADL

```
//-------------------------------------------------------------------------
//--
//-- Architectural Description Language to Reference Architecture documentation
//--
//-- Dominio: Telefonia Celular
//-- Data: Wed Nov 25 23:26:13 BRST 2009
//-------------------------------------------------------------------------

import $AS_GLOBAL_PATH/families/ThreeTieredFam.acme;
Family OdysseyFamily extends ThreeTieredFam with {

    // Architectural Properties
    Property aFailToler: string = "True";
    Property aSecurity: string = "True";
    Property aPerformance: string = "False";
    Property aAvailability: string = "True";
    Property aScalability: string = "False";

    // Infrastructure-Tier:  Reference Architecture Layer
    // Used to group architectural elements
    Group Type Infrastructure-Tier = {

    }

    //  Monitoramento :  Architectural Element of Reference Architecture
    // Used to join components
    Element Type Monitoramento = {
        Property aTier : string = "Infrastructure-Tier";
    }

    // Armazenamento:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type Armazenamento = {
         Property aTier : string = "Infrastructure-Tier";
    }

    // RedeAcesso:  Architectural Element of Reference Architecture
    // Used to join components
```

```
Element Type RedeAcesso = {
    Property aTier : string = "Infrastructure-Tier";
}

// 3G:  Reference Architecture Component
// Used to group architectural properties from reference architecture
Component Type 3G = {
    Property aElementType : string = "RedeAcesso";
    Property aVariability : string = "Invariante";
    Property aMandatory : boolean = true;
    Property aUseCases : string = "NA";
}

// GSM:  Reference Architecture Component
// Used to group architectural properties from reference architecture
Component Type GSM = {
    Property aElementType : string = "RedeAcesso";
    Property aVariability : string = "Invariante";
    Property aMandatory : boolean = true;
    Property aUseCases : string = "NA";
}

// WCDMA:  Reference Architecture Component
// Used to group architectural properties from reference architecture
Component Type WCDMA = {
    Property aElementType : string = "RedeAcesso";
    Property aVariability : string = "Invariante";
    Property aMandatory : boolean = true;
    Property aUseCases : string = "NA";
}

// Utilities-Tier:  Reference Architecture Layer
// Used to group architectural elements
Group Type Utilities-Tier = {

}

// Seguranca:  Architectural Element of Reference Architecture
// Used to join components
Element Type Seguranca = {
    Property aTier : string = "Utilities-Tier";
}

// Regionalizacao:  Architectural Element of Reference Architecture
// Used to join components
Element Type Regionalizacao = {
    Property aTier : string = "Utilities-Tier";
}
```

```
    // Process-Tier:  Reference Architecture Layer
    // Used to group architectural elements
    Group Type Process-Tier = {

    }

    // Relatorios:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type Relatorios = {
        Property aTier : string = "Process-Tier";
    }

    // GerenciamentoChamadas:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type GerenciamentoChamadas = {
        Property aTier : string = "Process-Tier";
    }

    // GerenciamentoMensagens:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type GerenciamentoMensagens = {
        Property aTier : string = "Process-Tier";
    }

    // Entidades:  Architectural Element of Reference Architecture
    // Used to join components
    Element Type Entidades = {
        Property aTier : string = "Business-Tier";
    }
}
```

APÊNDICE N – LISTA PADRÃO DE ELEMENTOS DOCUMENTÁVEIS

Lista de Elementos Padrão	
1	Fila
2	Usuário
3	Provedor de Serviços
4	Operadora de Telefonia Celular
5	Rede 3G
6	Rede GSM
7	Rede WCDMA
8	Acesso a Banco de Dados
9	Autenticação
10	Criptografia
11	Cache
12	Lista de Usuários
13	Idioma
14	E-Mail
15	Envio de Mensagem
16	Mensagem de Texto
17	Dash Board

APÊNDICE O – AVALIAÇÃO DA PROPOSTA

> “Inexiste no mundo coisa mais bem distribuída que o bom senso, visto que cada indivíduo acredita ser tão bem provido dele, que mesmo os mais difíceis de satisfazer em qualquer outro aspecto, não costumam desejar possuí-lo mais do que já possuem” [Gro09].

Esse capítulo apresenta um experimento realizado para avaliar a abordagem RADA considerando aspectos quantitativos de qualitativos. Ao final do capítulo, além das avaliações dos resultados com base na estatística, será também apresentada uma avaliação da experiência dos participantes.

7.1 Introdução

Experimentação, conforme definido por Travassos [Tra02], é o centro do processo científico. Segundo ele, a experimentação oferece um modo sistemático, disciplinado, computável e controlado para avaliação da atividade humana. Dessa forma, somente experimentos podem explorar os fatores críticos e dar luz ao fenômeno novo, para que as teorias possam ser formuladas e corrigidas [Tra02]. Já o autor [Pfl98], colocando de maneira mais formal, fala do experimento como sendo um tipo de estudo controlado, geralmente realizado em laboratórios, no qual os valores de variáveis independentes, representadas pelas entradas, são manipulados para se observar as mudanças nos valores de variáveis dependentes, representadas pelas saídas. Sendo este estudo seguido de uma posterior análise, interpretação, apresentação e empacotamento dos resultados.

Segundo Wholin [Woh00], existem quatro métodos relevantes para condução de experimentos na área da Engenharia de Software (ES): científico, engenharia, experimental e analítico. A realização de atividades de experimentação de um método, processo, técnica ou ferramenta, representa uma forma cientificamente aceita para obter indícios sobre a viabilidade e eficiência, dos benefícios esperados de alguma destas propostas [Bac04]. Em [Tra02], o autor destaca o aumento da quantidade de

trabalhos científicos envolvendo atividades de experimentação em ES e, como consequência, a oportunidade dos pesquisadores se concentrarem em abordagens promissoras, de modo a rejeitarem as ideias duvidosas.

Nesse sentido, segundo Wholin [Woh00], existem três principais estratégias de experimentação, que podem variar por influência das condições existentes para uma investigação científica, assim como pelo propósito da avaliação, i.e., se é uma técnica, método ou ferramenta. As estratégias são:

i. **Pesquisas de campo**: é uma investigação realizada em retrospectiva e conduzida com o objetivo de avaliar algumas técnicas ou ferramentas já utilizadas [Kit01]. Geralmente, são conduzidas entrevistas e aplicados questionários para a coleta de dados;
ii. **Estudos de caso**: são utilizados para monitorar projetos, atividades ou atribuições. Através destes estudos, é possível observar um atributo ou estabelecer relacionamentos entre diferentes atributos. Estudos de caso podem ser organizados para comparar resultados de projetos similares, um utilizando uma tecnologia atual e outro uma nova tecnologia. Também podem ser aproveitados para aplicar um método aleatoriamente atribuído a um projeto específico de uma empresa, e não atribuí-lo a outros da mesma organização como forma de comparação;
iii. **Experimentos**: é uma atividade geralmente efetuada em laboratório, com o propósito de descobrir algo desconhecido ou de testar uma hipótese. Experimentos são apropriados para confirmar teorias, avaliar a predição dos modelos ou validar as medidas. As principais características dos experimentos estão no controle total sobre o processo e variáveis, e na possibilidade de ser repetido.

Segundo Travassos [Tra02], em Engenharia de Software (ES), a estratégia mais apropriada para a validação é a experimentação, pois ela considera a proposição e avaliação do modelo com os estudos experimentais. Da mesma forma, por considerar o envolvimento do fator humano na descrição de Arquiteturas de Referência (AR), é

justificável a utilização de um experimento como abordagem para a realização da avaliação desta proposta [Sil09].

Nesta dissertação, conforme será detalhado mais adiante nesse capítulo pretendemos avaliar três pontos relacionados à abordagem proposta, são eles:

i. O tempo gasto na construção de uma documentação para uma Arquitetura de Referência;
ii. O quanto esta descrição gerada é precisa;
iii. A utilidade, usabilidade e aplicabilidade da proposta.

Pelo fato do experimento permitir uma avaliação essencialmente quantitativa [Woh00], o que abrange apenas dois (i e ii), dos três pontos que desejamos avaliar neste trabalho, optamos também por realizar uma pesquisa de campo (*survey*). A realização desta *survey* vai nos permitir obter uma avaliação qualitativa do terceiro ponto (iii), o qual não foi contemplado pela avaliação quantitativa.

Para apresentar os experimentos desenvolvidos nesta dissertação, o restante desse capítulo está organizado da seguinte forma: a Seção 7.2 apresenta o protocolo de experimentação utilizado, sua instanciação e execução, como forma de avaliar este trabalho; a Seção 7.3 apresenta os resultados obtidos a respeito do estudo quantitativo realizado sobre a documentação de Arquiteturas de Referência com e sem a abordagem proposta; a Seção 7.4 apresenta os resultados do estudo qualitativo referente a utilidade, usabilidade e aplicabilidade do uso de AR, como base para a instanciação de arquiteturas de software (AS); a Seção 7.5 mostra uma análise do perfil dos participantes do experimento, considerando as informações de competências dos participantes à respeito das atividades inerentes aos estudos qualitativos e quantitativos aos quais participaram; por fim a Seção 7.6 apresenta algumas considerações sobre os resultados obtidos pela aplicação do experimento e avaliação qualitativa realizadas.

7.2 Base Teórica: Engenharia de Software Experimental

A execução de um experimento exige preparo, principalmente para que seja possível fazer uma análise correta do objeto de estudo. Através da preparação e do controle sobre variáveis independentes, a realização do experimento irá nos levar a conclusões relevantes sobre as variáveis dependentes. Isto porque, um experimento entrega uma resposta específica para uma determinada configuração [Sil09] e [Bri96].

Na realização da avaliação da abordagem proposta nesta dissertação, é preciso controlar as atividades de indivíduos envolvidos em um processo de documentação de Arquiteturas de Referência *com a* utilização de uma Linguagem de Descrição Arquitetural (ADL), *e sem* o apoio desta linguagem, i.e., procedendo de forma *ad hoc*. Para tanto, vamos utilizar como apoio as abordagens de [Woh00] e [Tra02].

O processo de experimentação utilizado para avaliar esta proposta, envolve a realização dos seguintes passos:

i. **Definição:** representa o estabelecimento de hipóteses e objetivos claros, a partir do problema a ser solucionado; Nesse passo são delineados o escopo, os objetivos, os objetos e os grupos envolvidos na realização do experimento;

ii. **Planejamento:** representa a base do experimento, i.e., a forma como ele será conduzido. Um bom planejamento é importante para que seja viabilizado um controle efetivo sobre o experimento, de maneira que sejam evitadas distorções indesejáveis nos resultados. Esse passo envolve a formalização da hipótese nula e das hipóteses alternativas, estabelecimento das variáveis dependentes e independentes, escolha dos indivíduos integrantes da amostra utilizada, preparação conceitual da experimentação e a consideração sobre a validade do experimento; Uma amostra é um subconjunto de indivíduos da população alvo. Existem dois tipos de amostras, as probabilísticas, baseadas nas leis de probabilidades, e as amostras não probabilísticas, que tentam reproduzir o mais fielmente possível a população alvo. Entretanto, somente as

amostras probabilísticas podem, por definição, originar uma generalização estatística apoiada no cálculo de probabilidades, o que permite a utilização da potente ferramenta que é a inferência estatística [Ins10].

iii. **Execução:** envolve a preparação, execução das atividades pelos participantes e validação dos dados. Durante essa etapa, ocorre a interação humana, sendo necessária a preparação dos indivíduos sob o ponto de vista moral e metodológico, evitando assim resultados errôneos ou desinteressados;

iv. **Análise e interpretação:** nessa etapa os dados provenientes da execução são analisados e interpretados. Para tanto, é recomendado que se atente para a utilização de escalas, as quais podem indicar quais operações podem ser aplicadas aos valores das variáveis;

v. **Empacotamento:** esse passo está relacionado a descrição e organização dos resultados de maneira a viabilizar que o experimento possa ser reproduzido.

Nas próximas seções, são detalhados os passos executados durante a realização do experimento.

7.2.1 Definição

Compreende a etapa onde é determinada a fundamentação do experimento, apresentando o motivo pelo qual ele está sendo realizado. Nessa etapa são definidos os objetivos, o escopo, os objetos e os grupos envolvidos no experimento.

O autor [Bas94] propõe a abordagem *Goal Question Metric* (GQM), têm sua aplicação recomendada para ser utilizada no apoio a execução da fase de definição. Esta abordagem se baseia na definição de metas, em alto nível, que serão usadas para a geração de questões, sendo estas finalmente apresentadas por meio de métricas correspondentes as metas definidas [Nol07].

7.2.2 Planejamento

Segundo Wholin [Woh00], durante a fase de planejamento, é estabelecida a forma pela qual o experimento será conduzido. O experimento precisa ser planejado para que se estabeleça um controle efetivo sobre ele, prevenindo assim que ocorram distorções indesejáveis sobre resultados obtidos.

Esta fase é composta pelos seguintes passos:

i. **Seleção do contexto:** na seleção de contexto é feita a escolha do ambiente onde o experimento será executado;
ii. **Formulação de hipóteses:** estabelecimento de hipóteses para a realização de posterior análise estatística;
iii. **Seleção das variáveis:** consiste em definir as variáveis independentes, correspondendo às entradas, e variáveis dependentes, correspondendo as saídas do processo;
iv. **Seleção dos indivíduos:** consiste em definir um conjunto representativo e generalizado de indivíduos para a realização do experimento;
v. **Projeto:** propor a forma como o experimento será conduzido, com base nas hipóteses e variáveis selecionadas, incluindo seus objetos e participantes selecionados;
vi. **Instrumentação:** diz respeito a realização prática do experimento, fornecendo meios para a condução e monitoramento;
vii. **Avaliação da validade*:*** realização da verificação dos resultados sob o ponto de vista interno e externo, incluindo sua construção e conclusão.

7.2.2.1 Seleção do Contexto

A Seleção do Contexto determina as condições onde o experimento será realizado. Durante a realização dessa etapa, precisam ser consideradas quatro dimensões, conforme segue:

i. **Processo:** *off-line*, correspondendo a um projeto desenvolvido em paralelo, ou *on-line*, correspondendo a um projeto real;
ii. **Participantes:** estudantes ou profissionais integrantes do grupo de indivíduos convidados a participar do experimento;
iii. **Realidade:** problema de sala de aula (*toy example*) ou real;
iv. **Generalidade:** específico ou geral, ligado ao domínio da Engenharia de Software.

7.2.2.2 Formulação das Hipóteses

A definição de um experimento se dá através da formulação de uma hipótese fundamental chamada de *Hipótese Nula* (H_0), assim como pela definição de uma ou mais *Hipóteses Alternativas* (H_1, H_2, ..., H_n).

A Hipótese Nula está ligada a não derivação dos objetivos do experimento, é uma hipótese que é presumida verdadeira, até que provas estatísticas indiquem o contrário [Ins10]. As demais hipóteses correspondem as hipóteses em favor das quais a Hipótese Nula poderá ser rejeitada.

7.2.2.3 Seleção das Variáveis

A seleção das variáveis deve ser realizada criteriosamente, com base no conhecimento sobre o domínio da situação. Variáveis independentes, i.e., variáveis de entrada, devem influenciar as variáveis dependentes, i.e., variáveis de saída. Além disso, variáveis independentes podem ser referenciadas como sendo um fator, e para cada fator estabelecido, devem ser verificados os devidos tratamentos, que nesta pesquisa correspondem a uma abordagem para descrição de Arquiteturas de Referência através de uma Linguagem de Descrição Arquitetural, e a instanciação de Arquiteturas de Software baseadas em AR. A Tabela 4, mostra de forma tabular, as variáveis envolvidas no processo.

Tabela 4 – Tipos de Variáveis

Variáveis Independentes	**Variáveis Dependentes**
Variável de Entrada 1	*Variável de Saída 1*
Variável de Entrada 2	*Variável de Saída 2*
Variável de Entrada n	*Variável de Saída n*

7.2.2.4 Seleção dos Indivíduos

A amostra ligada a execução do experimento deve ser representativa o bastante para garantir a geração de resultados relevantes para o experimento. Quando a probabilidade de seleção dos indivíduos é conhecida, podemos utilizar uma amostragem probabilística, do contrário, utilizamos uma abordagem não probabilística.

O tamanho da amostra pode influenciar nos resultados e na força dos testes estatísticos. O tamanho pode ser escolhido com base na análise dos dados, considerando os seguintes aspectos: quanto maior for o tamanho da amostra, menor será a ocorrência de erros na generalização dos resultados, e quanto maior for à variabilidade na população, maior é o tamanho da amostra exigida.

7.2.2.5 Projeto

A realização de um experimento deve ser cuidadosamente planejada. Nesse sentido, o projeto de um experimento referencia e orienta a maneira pela qual as atividades serão planejadas e conduzidas. O resultado de um experimento é alcançado por intermédio da aplicação de métodos de análise estatística, sobre os dados coletados durante a sua realização. Estes métodos utilizados para a análise

de dados devem ser indicados por ocasião do projeto, mantendo sempre uma relação entre o projeto e a interpretação do experimento [Sil09].

Uma vez que um experimento corresponde a um conjunto de testes aplicados aos tratamentos de variáveis de entrada, esses testes devem ser planejados com a intenção de determinarmos o número de testes necessário para tornar visível o efeito dos tratamentos sobre as variáveis [Nol07]. Um projeto apropriado também possibilita, posteriormente, a replicação do experimento para outras configurações.

De acordo Wholin [Woh00], entre os princípios genéricos inerentes ao projeto de um experimento, estão:

i. **Aleatoriedade:** diz respeito ao estabelecimento de uma ordem obrigatória aplicada à alocação de objetos e indivíduos;
ii. **Obstrução:** com o objetivo de aumentar a precisão dos resultados, é sugerido que sejam bloqueados quaisquer fatores que, porventura, possam a vir exercer influência indesejável sobre o resultado da amostra. Para tanto, estes fatores devem ser conhecidos e passíveis de serem controlados. Assim sendo, eles podem ser removidos da comparação entre os tratamentos, para aumentar a precisão dos resultados;
iii. **Balanceamento:** estabelece que cada tratamento sobre as variáveis tenha a mesma quantidade de indivíduos.

Para a execução da análise de um experimento, são aplicadas, na maioria dos experimentos [Nol07], duas abordagens, são elas:

i. **Projeto completamente aleatório:** aplicação aleatória de cada instância do fator à apenas um dos tratamentos definidos, levando em consideração os dados obtidos. De acordo com essa abordagem, deve haver um balanceamento entre a distribuição dos tratamentos entre os fatores.
ii. **Projeto de comparação pareado:** estabelece que a todas as instâncias do fator são aplicadas tratamentos definidos, levando em

consideração os dados obtidos. De acordo com esta abordagem, a escolha do fator a ser executado sobre os tratamentos deve seguir os princípios de aleatoriedade e balanceamento.

A distribuição do fator sobre os tratamentos pode ser representada por meio de uma tabela, conhecida como tabela de contingência, que representa a distribuição do fator sobre os tratamentos de acordo com o tipo de projeto estabelecido, i.e., aleatório ou pareado. Esta tabela é uma forma de organizar informação sobre dados bivariados, ou seja, dados quantitativos e dados qualitativos.

7.2.2.6 Instrumentação

A instrumentação define os recursos para a condução do experimento e análise posterior. Para esta etapa são definidos alguns tipos de instrumentos, conforme descritos a seguir:

i. **Objetos:** modelos, documentos de especificação, código fonte, aplicações e diagramas, dentre outros;
ii. **Guias:** têm como objetivo apoiar os participantes durante a realização do experimento, e.g., tutoriais, *checklists* e manuais;
iii. **Métricas:** viabilizam a análise dos dados coletados, e são registradas em formulários ou questionários ao longo das entrevistas realizadas com os participantes.

7.2.2.7 Avaliação da Validade

A análise da validade dos resultados de um experimento, deve ser incluída na etapa de planejamento. Conforme observado na literatura, o planejamento da etapa

de avaliação deve levar em consideração quatro tipos de validação de resultados, conforme segue:

i. **Validade interna:** verifica se a relação observada entre o tratamento e resultado é casual, ou se é dependente de algum fator não previsto que possa causar influência, sobretudo, com relação aos participantes durante o experimento;
ii. **Validade externa:** verifica a possibilidade dos resultados obtidos ao longo do experimento serem generalizados para um ambiente externo, levando em consideração o fato dos participantes terem ou não, representatividade diante do público alvo pretendido;
iii. **Validade de construção:** verifica a correspondência entre a teoria colocada em função do experimento, e aquela apresentada aos indivíduos, assim como os efeitos da experimentação, posteriormente;
iv. **Validade da conclusão:** verifica a capacidade de obtenção de uma conclusão condizente aos tratamentos e resultados alcançados, levando em consideração os testes estatísticos adotados, os indivíduos, as medidas e dados dos tratamentos.

7.2.3 Execução

A execução do experimento corresponde a fase operacional, envolvendo diretamente os participantes através da execução das atividades propostas em função da realização do experimento. Isso irá possibilitar a obtenção das métricas estabelecidas. Nesse ponto, i.e., execução propriamente dita é importante destacar que, mesmo que o planejamento tenha sido bem feito, e que os dados tenham sido coletados e validados com os métodos mais apropriados, a validade dos resultados dependerá do comprometimento dos indivíduos com as atividades propostas, devendo estes executá-las com responsabilidade. Para a realização da etapa de execução do experimento, são sugeridos os seguintes passos:

i. **Preparação:** envolve a escolha dos participantes do experimento, provendo a eles acesso a treinamentos, glossários, manuais, guias, *checklists* e todas as informações necessárias para a execução efetiva das atividades propostas. A etapa de preparação envolve também a seleção dos instrumentos elaborados para o experimento, e.g., formulários, tutoriais, modelos, diagramas e aplicações;
ii. **Execução:** envolve a realização das atividades pelos participantes, seguindo os diferentes tratamentos propostos. A execução pode se dar em um período de tempo breve ou longo, e com ou sem a presença constante do responsável;
iii. **Validação dos resultados:** envolve a validação e análise dos dados coletados através da execução do experimento. A corretude dos dados está ligada a fatores como: a compreensão dos indivíduos em face às atividades a serem executadas, formulários a serem preenchidos, comprometimento com o qual desempenharam as atividades, e etc. Nesse ponto devemos observar e invalidar os resultados referentes aos erros detectados.

7.2.4 Análise e Interpretação

Após realização da etapa de execução, os dados gerados precisam ser analisados de maneira que possamos extrair conclusões válidas a partir deles.

Inicialmente, para observarmos os dados obtidos no experimento, é sugerido que atentemos para a medida de suas escalas. As escalas irão determinar as operações que poderão ser aplicadas aos valores das variáveis. Conforme verificado na literatura, foram definidas as seguintes escalas:

i. **Nominal:** relacionada a valores distintos os quais não possuem interpretação numérica e nem ordenação, e.g., nomes próprios, nomenclatura de variáveis do tipo texto em geral, e etc;

ii. **Ordinal:** relacionada a valores distintos sem interpretação numérica, mas passíveis de ordenação, e.g., avaliações do tipo: ótimo, bom, regular, ruim e péssimo;

iii. **Intervalar:** relacionada a valores distintos, passíveis de ordenação, e nos quais a distância entre os números, possui uma interpretação comum, e.g., escalas de temperatura, distância percorrida, tempo;

iv. **Razão:** corresponde a valores distintos, passíveis de ordenação, os quais a distância e razão podem ser interpretadas, e.g., peso, tamanho, tempo, precisão, etc.

Logo depois da verificação das medidas das escalas de cada variável, realizamos uma análise numérica dos dados, aplicando recursos da estatística descritiva.

Tomando como base a análise da distribuição geral do conjunto de dados, eventuais distorções devem ser eliminadas, e.g., dados anormais, valores extremos (*outliers*), de maneira que não interfiram na validade das conclusões. Depois disso, podemos obter as conclusões do experimento através do teste das hipóteses. Segundo Wholin [Who00], o teste das hipóteses pode ser:

v. **Paramétrico:** incide explicitamente sobre um parâmetro de uma ou mais populações e utiliza formas fechadas, derivadas de propriedades de distribuições de frequências conhecidas. É aplicada em variáveis razão e intervalares, determinando que sejam utilizados dados que possuam como características:

   a. **Normalidade:** quando os valores apresentam uma tendência a se concentrarem próximos a média, e, quanto mais distante da média, menor é a frequência das observações. Neste caso, a hipótese nula é considerada verdadeira até que haja evidência estatística de que os dados apontam para a sua rejeição [Ins10];

   b. **Homocedasticidade:** quando os valores apresentam uma variância constante.

vi. **Não Paramétrico:** quando os valores observados não atendem a critérios de normalidade e de homocedasticidade, desconsiderando-se a distribuição destes. Segundo Fonseca [Fon96], os testes não paramétricos são mais indicados para a avaliação de dados qualitativos, quando se trabalha com amostras pequenas (inferiores a 30). Os principais representantes da estatística não paramétrica são os testes de Qui Quadrado, de Wilcoxon, de Mann-Whitney, da Mediana, e de Kruskall-Wallis.

Em relação ao tipo de teste estabelecido, i;e; paramétrico ou não-paramétrico, em um experimento que considera um fator e dois tratamentos, o teste das hipóteses é realizado por meio das seguintes abordagens:

i. **Teste T (Student's *t*-tests):** é um teste paramétrico que utiliza duas amostras independentes, comparando duas médias a partir de uma hipótese nula quando os desvios padrão populacionais são desconhecidos, o que ocorre na grande maioria dos casos [Fon96]. O Teste T pressupõe a inexistência de diferenças relevantes entre os grupos considerados.

ii. **Teste de Mann-Whitney:** empregado na realização de testes não-paramétricos, prova se dois grupos independentes tem origem na mesma população.

7.2.5 Empacotamento

Uma vez encerrada a etapa de análise e interpretação de dados, é sugerida a etapa de empacotamento. Essa etapa se destina a elaboração da documentação dos aspectos relacionados ao experimento, proporcionando assim, dentre outras coisas, que o experimento possa ser replicado.

Quando o experimento é executado em contextos distintos, é possível adquirirmos novos conhecimentos sobre os conceitos estudados. Dessa forma, é importante documentar diversos aspectos ligados ao experimento, tais como os descritos a seguir:

i. **Comunidade:** caracteriza os indivíduos envolvidos na realização do experimento, e.g., pesquisadores, participantes e interessados em aproveitar os resultados;
ii. **Organização:** caracteriza as etapas referentes ao planejamento, projeto, execução e instrumentação do experimento;
iii. **Artefatos:** caracterizam as especificações, diagramas, modelos, tutoriais, formulários, entre outros artefatos definidos para a instrumentação do experimento;
iv. **Resultados:** apresenta um detalhamento dos dados obtidos e refinamentos aplicados, a análise realizada para testar as hipóteses e as conclusões encontradas.

A Seção 7.3 apresenta o empacotamento do experimento realizado para avaliação quantitativa da abordagem proposta para descrição de Arquiteturas de Referência através da utilização de uma Linguagem de Descrição Arquitetural.

7.3 Estudo Quantitativo

7.3.1 Definição

Para a realização deste estudo experimental, optamos por utilizar a abordagem GQM, como forma de definição para: estudo e objetivo global de medição.

7.3.1.1 Objetivo Global

Avaliar, em um processo de desenvolvimento com foco em reúso, o esforço e a precisão da construção da documentação de uma Arquitetura de Referência.

7.3.1.2 Objetivo do Estudo

Comparar, em um processo de desenvolvimento de um domínio de aplicações, a criação de uma documentação de AR por meio de uma ADL com a criação de uma documetação de AR num processo *ad-hoc*,

Com o propósito de caracterizar o tempo gasto na construção da documentação de uma AR,

Com foco no esforço e precisão,

Sob o ponto de vista do Arquiteto de Software ou Engenheiro de Domínio,

No contexto da criação de uma documentação de Arquitetura de Referência para um sistema desenvolvido por estudantes, no domínio de telefonia celular.

7.3.1.3 Objetivo da Medição

Em uma atividade de desenvolvimento de um domínio, caracterizar:

i. Qual o esforço, medido em minutos, necessário de cada participante para a criação da documentação de uma Arquitetura de Referência, com e sem o uso da ADL?
ii. Qual a precisão, definida por meio de corretude e completude, da documentação da AR gerada, com e sem o uso de ADL?

7.3.1.4 Questões

i. O esforço envolvido na criação de uma documentação para uma Arquitetura de Referência utilizando uma Linguagem de Descrição

Arquitetural é igual ao esforço envolvido na documentação sem a utilização da linguagem?

ii. A precisão dos elementos arquiteturais documentados na ADL é igual à precisão da documentação *ad hoc* criada para o domínio?

7.3.1.5 Métricas

A métrica associada à *Questão i* corresponde ao esforço medido pela relação do tempo gasto por cada participante para realizar a documentação da AR, com e sem a utilização da ADL.

A métrica relacionada à *Questão ii* corresponde a precisão na compreensão da descrição gerada, sob a forma de ADL, para uma AR. Neste caso, se entende por precisão a razão entre a quantidade de elementos arquiteturais documentados (com e sem a abordagem), pela quantidade de elementos arquiteturais esperados. O número total de elementos integrantes de uma AR ideal do domínio será definida por um especialista.

7.3.2 Planejamento

7.3.2.1 Seleção do Contexto

Para a realização desse experimento, foi utilizada a infra-estrutura do Programa de Pós-Graduação da FACIN. Segundo Wholin [Who00], as dimensões a serem consideradas na seleção do contexto são:

i. **Processo**: neste estudo, optamos pela utilização de uma abordagem *In-vitro*, onde os participantes executam o experimento num ambiente controlado. Além disso, como a execução ocorre num ambiente acadêmico, o experimento é considerado *off-line*;

ii. **Participantes**: este estudo será realizado com alunos de graduação e pós-graduação da FACIN, e profissionais da indústria;
iii. **Realidade**: vamos estudar um problema de sala de aula (*toy example*). Esse problema corresponde a um sistema modelado e desenvolvido por alunos da pós-graduação, durante uma disciplina do PPGCC, da PUCRS, no domínio de telefones celulares;
iv. **Generalidade**: o experimento é específico, e possui validade apenas no âmbito do estudo atual.

7.3.2.2 Formulação das Hipóteses

A formalização das hipóteses e suas respectivas medidas estão apresentadas a seguir:

1. **Hipóteses relacionadas à variável esforço:**
    a. **Medidas:** o esforço será avaliado pelo tempo gasto em minutos com a elaboração da descrição da Arquitetura de Referência, com e sem a abordagem avaliada. É representado pela diferença entre o tempo final e o tempo inicial gasto em cada abordagem, onde:
        i. $\Delta_{\mathbf{tar}}$: representa a variação de tempo gasto em minutos para criação da descrição da Arquitetura de Referência com o apoio da abordagem; e
        ii. $\Delta_{\mathbf{tcomp}}$: representa a variação de tempo gasto em minutos para descrição da Arquitetura de Referência sem o apoio da Abordagem.
    b. **Hipótese Nula, H_0:** $\Delta_{tar} = \Delta_{tcomp}$: o esforço para documentação da Arquitetura de Referência com a apoio da abordagem é igual ao esforço para documentação da Arquitetura de Referência sem a abordagem.

c. **Hipótese Alternativa, H_1:** $\Delta_{tar} > \Delta_{tcomp}$: o esforço para documentação da Arquitetura de Referência com a apoio da abordagem é maior do que o esforço para documentação da Arquitetura de Referência sem a abordagem.
d. **Hipótese Alternativa, H_2:** $\Delta_{tar} < \Delta_{tcompt}$: o esforço para documentação da Arquitetura de Referência com a apoio da abordagem é menor do que o esforço para documentação da Arquitetura de Referência sem a abordagem.

2. **Hipóteses relacionadas à variável precisão:**
    a. **Medidas:** a precisão corresponde a razão entre o conjunto de elementos arquiteturais documentados na Linguagem de Descrição Arquitetural (ADL) e *ad hoc*, pelo conjunto total esperado, sendo:
        i. Δ_{tar}: precisão associada à descrição da Arquitetura de Referência descrita sob a forma de ADL;
        ii. Δ_{tcomp}: precisão associada à descrição da Arquitetura de Referência sem ADL;
    b. **Hipótese Nula, H_0:** $\Delta_{tar} = \Delta_{tcomp}$: a precisão associada à descrição da Arquitetura de Referência por ADL é igual a precisão associada a descrição da Arquitetura de Referência sem ADL.
    c. **Hipótese Alternativa, H_1:** $\Delta_{tar} > \Delta_{tcomp}$: a precisão associada à descrição da Arquitetura de Referência por ADL é maior que a precisão associada à descrição da Arquitetura de Referência sem ADL.
    d. **Hipótese Alternativa, H_2:** $\Delta_{tar} < \Delta_{tcomp}$: a precisão associada à descrição da Arquitetura de Referência por ADL é menor que a precisão associada à descrição da Arquitetura de Referência sem ADL.

7.3.2.3 Seleção das variáveis

7.3.2.3.1 Variáveis Independentes

Foram assumidas como variáveis independentes, as seguintes variáveis de entrada:

i. Experiência dos participantes do experimento; e
ii. Técnica para a descrição de Arquiteturas de Referência.

7.3.2.3.2 Variáveis Dependentes

Foram assumidas como variáveis dependentes:

i. **Esforço** para a elaboração das descrições das arquiteturas;
ii. **Precisão** através do número de elementos arquiteturais documentados utilizando determinada abordagem com relação ao conjunto total de artefatos.

7.3.2.3.3 Seleção dos Indivíduos

A população deste experimento é formada por profissionais da indústria e alunos dos cursos de graduação e pós-graduação da Faculdade de Informática da PUCRS, constituindo um total de doze participantes distribuídos conforme segue:

i. Dois alunos de graduação;
ii. Três alunos de pós-graduação;
iii. Sete profissionais da indústria.

A seleção dos indivíduos será feita através de uma amostragem não probabilística, considerando uma abordagem por conveniência e por cota:

i. **Amostragem por conveniência:** serão convidados os indivíduos mais convenientes para o experimento; e
ii. **Amostragem por quota:** esta abordagem determina a seleção de indivíduos pertencentes a populações distintas, nesse caso, graduação, pós-graduação e profissionais da indústria. Pressupõe-se que a experiência na construção de arquiteturas de software entre estas duas populações seja diferente.

7.3.2.4 Projeto do experimento

Para a realização deste experimento, vamos caracterizar os seguintes princípios genéricos:

i. **Aleatoriedade:** será utilizada a aleatoriedade como forma de definir quais indivíduos irão executar cada abordagem para a documentação da AR;
ii. **Obstrução:** conforme observado na Seção 7.3.2.3.3, os participantes não possuem o mesmo nível de experiência. Tal cenário justifica a escolha pelos critérios de quota e conveniência para a seleção dos indivíduos como forma de minimizar o impacto causado por esse lacuna no experimento;
iii. **Balanceamento:** vamos utilizar este princípio na realização de nosso experimento para que a atividade de documentação de AR, com e sem a abordagem, seja efetuada pela mesma quantidade de participantes.

7.3.2.5 Distribuição da Amostra e Definição dos Testes de Hipóteses

Para cada hipótese, serão utilizadas as seguintes notações:

μar - Abordagem baseada em Linguagem de Descrição Arquitetural;

µcomp - Abordagem baseada em componentes.

Projeto: optamos pela utilização do projeto aleatório, pois nesse tipo de projeto cada indivíduo irá executar apenas uma abordagem de forma aleatória. A Tabela 5, conhecida como tabela de contingência[4], mostra a distribuição dos participantes nas duas abordagens, com uma ordem definida de maneira aleatória.

Tabela 5 – Distribuição dos Participantes

Participante	µar	µcomp
AR01	X	
CO02		X
CO03		X
CO04		X
AR05	X	
AR06	X	
CO07		X
CO08	X	
CO09		X
AR10	X	
AR11	X	
CO12		X

Teste das hipóteses: considerando o tipo de amostra, vamos utilizar os seguintes testes:

i. Teste paramétrico de significância: Teste T; e
ii. Teste não paramétrico de significância: Mann-Whitney.

Nível de significância é definido como a probabilidade de cometer o erro de tipo *x*, por exemplo, o que implicaria em rejeitar a hipótese nula (H_o), quando ela é verdadeira [Ins10]. Em Estatística, um resultado é significante se for improvável que tenha ocorrido por acaso, caso uma determinada hipótese nula seja verdadeira, mas não sendo improvável caso a hipótese base seja falsa [Fon96].

Para a seleção do teste a ser aplicado (paramétrico ou não paramétrico), vamos considerar:

i. Análise da normalidade: teste de Shapiro-Wilk [Sha65];
ii. Análise da variância dos dados: teste de Levene.

7.3.2.6 Instrumentação

Na condução deste experimento vamos utilizar:

Objetos: um modelo de *features* do domínio de telefonia celular (constituído de características funcionais, conceituais e atores), conforme apresentado no Apêndice K; um diagrama de componentes (apresentando relacionamentos e agrupamentos), conforme Apêndice J; e uma descrição da Arquitetura de Referência, conforme apresentado no Apêndice L. Além disso, vamos utilizar uma especificação de requisitos, conforme apresentado no Apêndice G, a qual servirá de base para os participantes realizarem a instanciação de uma Arquitetura de Software.

Através da utilização destes recursos, pretendemos facilitar a compreensão dos artefatos que compreendem o domínio, e.g., casos de uso, *features*, notação envolvida. Isto vai fazer com que não seja necessário alocar muito tempo dos participantes para a realização de

[4] Em estatística, as tabelas de contingência são usadas para registrar e analisar o relacionamento entre duas ou mais variáveis, normalmente de escala nominal [Ins10].

treinamentos. Nesse sentido, optamos pela utilização de *features* básicas, i.e., que representam características de negócio, atores e funcionalidades;

Guias: será disponibilizado um treinamento aos participantes, com o intuito de nivelar o conhecimento sobre a modelagem de domínios em termos de características e componentes. Também será apresentado, de forma sucinta, uma visão sobre a utilização das ferramentas Odyssey [Wer09], ABLE [Dav09] e Jude [Cha09], além da distribuição de um glossário com as palavras chave pertencentes a Linguagem de Descrição Arquitetural utilizada, para os participantes que irão utilizar a abordagem. Para evitar distorções, vamos apresentar algumas alternativas para a documentação de uma Arquitetura, para os participantes que não irão utilizar a abordagem, assim como disponibilizar uma ferramenta CASE;

Métricas: os dados serão registrados através de uma planilha, e preenchidos pelos participantes com base nas atividades realizadas nas ferramentas Odyssey, ABLE e JUDE.

7.3.2.7 Análise da Validade

7.3.2.7.1 Validade interna

Vamos considerar quatro critérios, conforme definido a seguir:

i. **Histórico:** a data de realização do experimento será estabelecida de forma a se adequar a disponibilidade dos indivíduos;
ii. **Maturação:** serão utilizadas técnicas para incentivar os indivíduos;
iii. **Seleção dos grupos:** vamos realizar um treinamento para nivelar o conhecimento dos participantes. A realização do experimento será individual;

iv. **Difusão:** durante a realização do experimento, será solicitado que os participantes não interajam entre si.

7.3.2.7.2 Validade externa

Para esta avaliação procuramos selecionar indivíduos com algum conhecimento prévio na modelagem de sistemas e construção de arquiteturas de software.

7.3.2.7.3 Validade de construção

Serão considerados os seguintes aspectos:

i. **Inadequada explicação pré-operacional:** consiste na explicação operacional do experimento, visando amadurecer a forma pela qual será realizada a descrição de uma Arquitetura de Referência;
ii. **Adivinhação de hipóteses:** pelo fato dos participantes serem humanos, é possível que pela sua interação com o experimento, surjam novas hipóteses. Dessa forma, é necessário que se mantenha o foco no estudo planejado;
iii. **Expectativas do condutor do experimento:** o responsável pela condução do experimento pode exercer influência sobre as variáveis envolvidas e sobre o material elaborado. Por isso, ao longo desse experimento, todo o material utilizado foi avaliado por outro responsável.

7.3.2.7.4 Validade da conclusão

Para a validação da conclusão, serão consideradas as seguintes perspectivas:

Manipulação dos dados: como o pesquisador será o responsável pela manipulação dos dados, podem ocorrer variações;

Confiabilidade das medidas: as medidas foram definidas pelo pesquisador de maneira objetiva;

Confiabilidade na implementação dos tratamentos: a criação de Arquiteturas de Referência e arquiteturas de software, dependem da experiência do Arquiteto. Isso representa um risco, considerando o fato de que cada indivíduo pode executar de maneira distinta os passos definidos pelo experimento. Provavelmente, participantes diferentes irão criar arquiteturas de software consideravelmente distintas para um mesmo domínio, utilizando as mesmas abordagens, mesmo tendo sido apresentado à ambos, o mesmo treinamento sobre um determinado domínio;

Configurações do ambiente do experimento: se refere à interferência causada pelo ambiente externo durante a execução do experimento, e o quanto isso pode influenciar nos resultados. Nesse sentido, o experimento é executado em uma sala, na qual não é permitida a utilização de aparelhos tal como telefones celulares, *pagers*, PDA's ou computadores dos participantes;

Heterogeneidade aleatória dos participantes: a seleção de participantes com diferentes perfis e grau de experiência nas atividades de criação e documentação de arquiteturas de software, se constitui em um risco a ser considerado, com relação a variação dos resultados obtidos pela execução do experimento. Dessa forma, procuramos balancear a distribuição dos indivíduos entre os dois grupos os quais eles foram divididos, de maneira que os grupos ficassem com um grau de

experiência mais homogêneo possível. A experiência é um atributo difícil de ser mensurado. O critério adotado para verificar as competências de cada indivíduo, para realizar a distribuição e constituição dos dois grupos, levou em consideração o número, em meses, de experiência em: desenvolvimento de sistemas, elaboração de arquiteturas de software, conhecimento de alguma técnica de reúso, e formação acadêmica. Estas informações foram obtidas através do Questionário de Caracterização da Amostra, conforme apresentado no Apêndice C.

7.3.3 Execução

7.3.3.1 Preparação

A preparação para a execução deste experimento atentou-se para:

Consenso com o experimento: segundo Wholin [Who00], os indivíduos ao participarem do experimento, devem estar cientes dos objetivos da pesquisa, do contrário, corremos o risco de que a participação dos indivíduos não saia de acordo com os objetivos. Dessa forma, deve ser fornecida aos participantes durante a fase de preparação, toda a base necessária sobre o experimento, assim como apresentados os objetivos e metas a serem alcançados. Esta base, contendo as diretrizes, procedimentos e objetivos, foi apresentada de forma clara aos indivíduos através do Formulário de Consentimento, conforme pode ser observado no Apêndice B;

Resultados sensitivos: para evitar que os resultados do estudo sejam influenciados por qualquer questão relacionada à sensibilidade do participante por estar sendo avaliado, optamos por não identificar os indivíduos ao longo da descrição da experimentação. Nesse sentido, para cada participante será atribuído um código. Os códigos utilizados para a

identificação são formados pelo prefixo AR concatenado com um número (ex. AR01, AR02, AR03), para os participantes que utilizam a abordagem, e pelo prefixo CO concatenado com um número, para os participantes que não utilizam a abordagem (ex. CO01, CO02, CO03).

Instrumentação: No que diz respeito ao instrumento de pesquisa, tanto variáveis quanto recursos, foram previamente definidos antes da execução do experimento. Nesse sentido, foi realizado um treinamento específico sobre modelagem de um domínio em termos de *features* e componentes, e um treinamento específico sobre o modelo de componentes proposto para o domínio com o auxílio de um especialista. O treinamento referente a estes artefatos citados, pode ser observado no Apêndice D. O especialista que contribuiu para a construção do modelo de domínio utilizado como padrão, não fez parte do grupo de indivíduos que participaram do experimento. Ao final do treinamento, foi entregue a *todos* os participantes o modelo de componentes do domínio utilizado no experimento (conforme apresentado no Apêndice J), e uma especificação contendo requisitos do domínio, a qual será utilizada pelos indivíduos para instanciar uma aplicação específica de domínio na etapa 2 do experimento, referente a avaliação qualitativa. Para os participantes que irão executar o experimento *com o apoio* da abordagem, foi entregue também um glossário explicando os construtores e propriedades que formam ADL utilizada. Já para os participantes que *não irão utilizar* a abordagem proposta, foi disponibilizada a ferramenta CASE JUDE. A coleta de dados será realizada pelos próprios participantes, e o registro será feito em um formulário para preenchimento dos resultados, conforme Apêndice H.

7.3.3.2 Execução

Este experimento foi elaborado de maneira a ser executado em um curto espaço de tempo, com a presença constante do pesquisador. Para tanto, o pesquisador ficou envolvido nas diversas atividades necessárias a sua realização.

Conforme apresentado no item Métricas, da Seção 7.3.2.6, a coleta de dados referentes aos resultados, fica sob responsabilidade do participante, estando o pesquisador disponível durante todo o processo para o esclarecimento de eventuais dúvidas que surgirem durante a execução do experimento.

7.3.4 Análise e Interpretação

Inicialmente, a primeira análise que vamos apresentar, está relacionada à classificação das escalas de variáveis estabelecidas no experimento. Através desta escala, conforme pode ser observado na Tabela 6, é possível saber quais operações podem ser feitas sobre as variáveis.

Tabela 6 – Escalas de variáveis

Variáveis	**Nome**	**Escala**
Dependentes	Tempo	Razão
	Precisão	
Independentes	Experiência dos participantes	Nominal
	Técnica para a descrição de arquiteturas	

7.3.4.1 Análise Tabular e Gráfica

O experimento realizado obteve como resultados para as variáveis *esforço* e *precisão*, os dados apresentados na Tabela 6.

Tabela 7 – Tabulação de valores brutos

Abordagem	**Participante**	**Esforço (minutos)**	**Precisão**
Com uso da abordagem	AR01	21	1,00
	AR05	22	1,00
	AR06	18	0,98
	AR08	25	1,00
	AR10	29	0,84
	AR11	19	1,00
Sem uso da abordagem	CO02	51	0,82
	CO03	46	0,76
	CO04	49	0,82
	CO07	45	0,88
	CO09	50	0,88
	CO12	48	0,82

Conforme pode ser observado na Figura 34, são apresentados sob a forma de um gráfico de código de barras, os resultados referentes ao *esforço* de cada participante para a criação de uma descrição de Arquitetura de Referência, com e sem a utilização da abordagem apresentada neste trabalho.

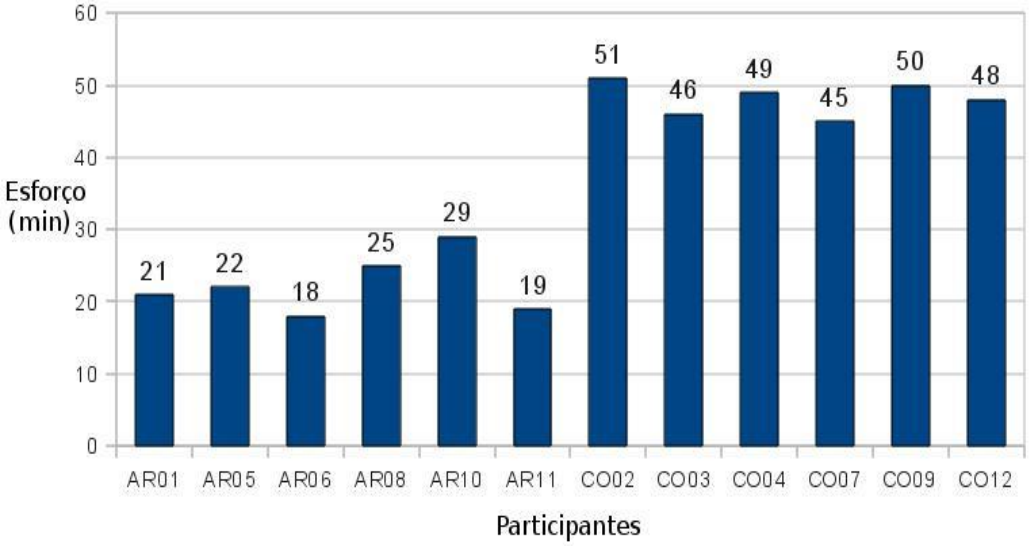


Figura 34 – Esforço para a criação de descrição de AR

A Figura 35 mostra o gráfico relacionado a variável *precisão*, que corresponde a corretude das descrições de AR geradas por cada participante, durante a condução do experimento.

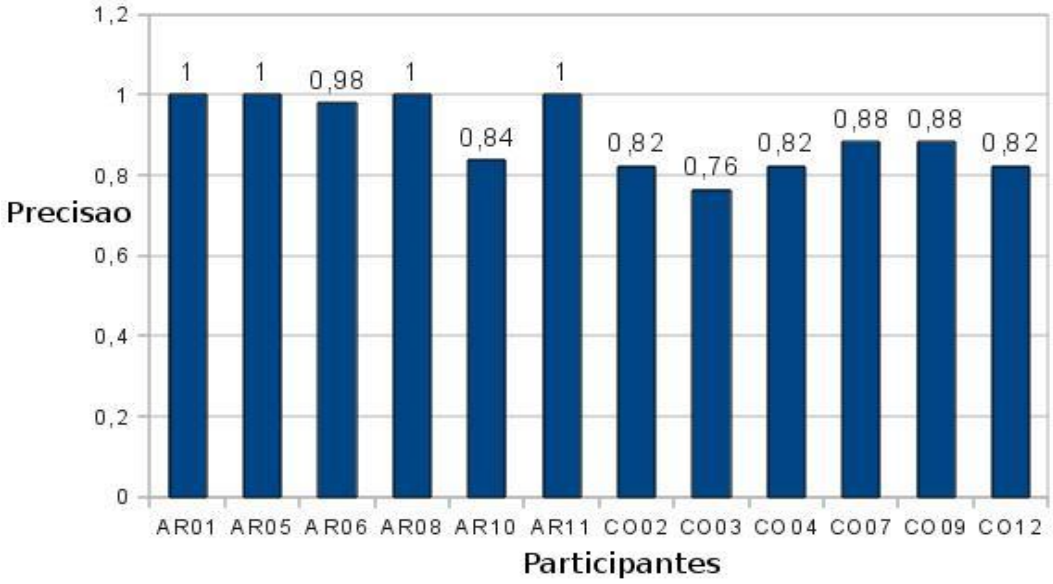


Figura 35 – Precisão para a descrição de AR criadas

7.3.4.2 Estatística Descritiva

Conforme descrito anteriormente na seção que apresentou a Formulação de Hipóteses, as variáveis dependentes se encontram caracterizadas na escala da razão, o que permite o cálculo da normalidade e homocedasticidade. O padrão de tipo de teste especificado, conforme também já descrito, é o Teste T para duas amostras independentes, se paramétrico, ou teste Mann-Whitney, se não paramétrico.

A avaliação será realizada em cima das variáveis *esforço* e *precisão*. Como forma de eliminar valores anormais que porventura possam vir a distorcer os resultados dos testes aplicados em nossa amostra, será utilizado o gráfico de dispersão *boxplot* para a identificação de possíveis anormais (*outliers*).

7.3.4.2.1 Análise sobre a Variável Esforço

Distribuição: inicialmente, para a análise da distribuição, vamos utilizar o gráfico de dispersão *boxplot*. Essa análise nos dá uma visibilidade sobre possíveis valores *outliers*, conforme pode ser visualizado na Figura 36.

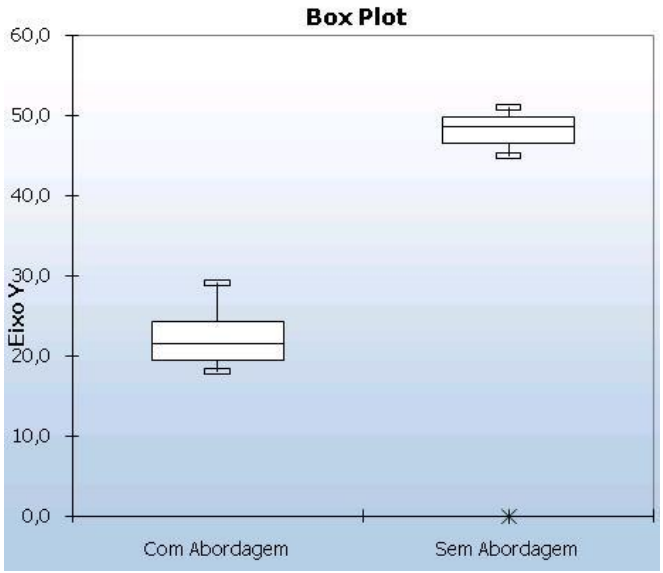


Figura 36 – Gráfico *boxplot* para a variável Esforço

Seleção da Amostra: conforme pode ser observado no gráfico gerado em função da variável esforço, não foram identificados valores anormais (*outliers*) na amostra referente ao tempo gasto pelos participantes. Dessa forma, todos os valores serão considerados. A Tabela 8 mostra a média e o desvio padrão calculados para a variável esforço.

Tabela 8 - Média e desvio padrão para a variável Esforço

Abordagem	Média	Desvio Padrão
Com Abordagem	22,330	4,082
Sem Abordagem	48,170	2,317

Análise da Normalidade: a etapa referente a análise da normalidade procura descobrir se os dados seguem uma distribuição normal. Para tanto, foram definidas duas hipóteses, uma *nula* e outra *alternativa*, conforme segue:

i. **H_0:** a distribuição é normal;
ii. **H_1:** a distribuição não é normal.

Levando em consideração que a variável esforço apresenta menos de 50 valores (< 50), é recomendável a utilização do teste de Shapiro-Wilk [Sha65]. A Tabela 9, mostra o resultado da execução do teste para a variável esforço.

Tabela 9 – Aplicação do teste de normalidade Shapiro-Wilk para a variável esforço

Variável	Abordagem	Estatística	Grau de Liberdade	Significância
Esforço	Com Abordagem	0.939	6	0,650
	Sem Abordagem	0.958	6	0,801

Tomando como base a Tabela 9, percebemos que a significância do resultado do teste Shapiro-Wilk executado é superior ao nível de significância estabelecido (0,05 ou 5%), para as duas amostras, com abordagem (=0,650) e sem abordagem (=0,801). Isso indica que não há indícios para rejeitar a hipótese nula, referente a distribuição da normalidade. Dessa forma, alcançamos o primeiro requisito para a utilização do teste paramétrico.

Análise da Homocedasticidade: a análise da homocedasticidade busca avaliar a variância entre os dois grupos. Para atingirmos esse objetivo, foram definidas duas hipóteses conforme segue:

i. **H_0:** as variâncias são iguais;
ii. **H_1:** as variâncias não são iguais.

O teste desta hipótese é executado com a significância obtida pela aplicação Teste de Levene. O Teste de Levene é utilizado para avaliar se *n* amostras têm a mesma variância[5]. A Tabela 10 mostra os resultados obtidos pela execução do Teste de Levene para a variável esforço.

Tabela 10 - Teste de Levene para a variável Esforço

Variável	Variâncias Iguais	Significância
Esforço	Assumindo	0,244
	Não Assumindo	0,000

Levando em consideração a Tabela 10, verificamos que o nível de significância para variâncias iguais (0,244) é superior ao nível de significância estabelecido (0,05 ou 5%). Baseados nisso, não conseguimos rejeitar a hipótese nula para variâncias. Assim, alcançamos o segundo requisito para a utilização de teste *paramétrico*.

De acordo com a definição feita no planejamento do projeto desse experimento, e conforme estabelecido na Seção 7.3.4.2, é indicado o Teste T para a avaliação das hipóteses para duas variáveis independentes. Com isso é possível

avaliar a sua utilização, pois se trata de um teste paramétrico onde os seus requisitos foram explicitamente atendidos.

Conforme hipóteses declaradas na fase de projeto do experimento, temos:

H_0: $\mu_{tar} = \mu_{tcomp}$

H_1: $\mu_{tar} > \mu_{tcomp}$

H_2: $\mu_{tcomp} > \mu_{tar}$

O critério para rejeição de **H_0** em favor de **H_1** é:

H_1: ($\mu_{tar} > \mu_{tcomp}$): rejeitamos **H_0** se $t_0 > t_{a,\ n + m - 2}$, onde:

- t_0: é o valor estabelecido para *t* através do Teste T.
- $t_{a,\ n + m - 2}$: é o valor obtido pela tabela de Distribuição de T, exibida no Anexo C, onde $n + m - 2$ representa o grau de liberdade. Nesse contexto, *n* é o número de indivíduos para uma abordagem (=6), e *m* o número de indivíduos para a outra abordagem (=6). O grau de liberdade da abordagem proposta neste trabalho é igual a 10 ($n + m - 2 = 10$).

Grau de liberdade é um conceito ligado ao número de dados disponíveis (livres) para a realização do cálculo da estatística [Ins10]. Por exemplo, ao estimarmos a média populacional com a média amostral, perdemos um grau de liberdade, assim a estatística *t*-student (Teste T) terá *n* - 1 graus de liberdade. No caso da Tabela de ANOVA (análise de variância), o grau de liberdade do grupo será igual ao número de grupos menos 1, o grau de liberdade total será igual a n-1, e os graus de liberdade do resíduo serão a diferença entre esses dois.

O Teste T para duas amostras independentes foi realizado neste contexto, e os seus resultados podem ser observados na Tabela 11.

---

[5] Na teoria da probabilidade e na estatística, a variância de uma variável aleatória é uma medida da sua dispersão estatística, indicando quão longe, em geral, os seus valores se encontram do valor esperado [Wik10b].

Tabela 11 – Teste T para duas amostras independentes para a variável Esforço, agrupada por descrição baseada em AR

Variável	Variâncias Iguais	T	Grau de Liberdade	Significância (bicaudal)
Precisão	Assumindo	-13,481	10	0,000
	Não Assumindo	-13,481	7,918	0,000

Baseados na Tabela 11, obtivemos o t_0 = -13,481 e, baseados na tabela de Distribuição, obtivemos o valor de $t_{a,\, n + m - 2}$ = 2,228. Como o t_0 é menor que $t_{a,\, n + m - 2}$, não é possível rejeitar a hipótese nula a um nível de significância de 5% em favor de H1: $\mu_{tar} > \mu_{tcomp}$.

O critério para rejeição de **H_0** em favor de **H_2** é:

H_2: ($\mu_{tcomp} > \mu_{tar}$): rejeitamos **H_0** se $t_0 > t_{a,\, n + m - 2}$

O Teste T para duas amostras independentes foi realizado neste contexto, e os seus resultados podem ser observados na Tabela 12.

Tabela 12 - Teste T para duas amostras independentes para a variável esforço, agrupada por descrição baseada em Componentes

Variável	Variâncias Iguais	T	Grau de Liberdade	Significância (bicaudal)
Precisão	Assumindo	13,481	10	0,000
	Não Assumindo	13,481	7,918	0,000

Tomando como base a Tabela 12, obtivemos para t_0 o valor (= 13,481), e para $t_{a,\, n + m - 2}$ o valor (= 2,228). Como o valor t_0 é maior que $t_{a,\, n + m - 2}$, podemos rejeitar a hipótese nula a um nível de significância de 5% em favor de **H_2:** ($\mu_{tcomp} > \mu_{tar}$).

Considerações: de acordo com a observação dos resultados das análises estatísticas realizadas em função da variável esforço para a descrição de Arquiteturas de Referência com e sem a abordagem proposta nesta pesquisa, concluímos que existe uma diferença estatisticamente significativa. Esta diferença aponta que o esforço necessário para a descrição de uma AR com a abordagem é menor do que o esforço necessário para descrição de uma AR sem a abordagem.

7.3.4.2.2 Análise sobre a Variável Precisão

Distribuição: para a análise da distribuição sobre a variável precisão, vamos utilizar, da mesma forma que para a variável esforço, o gráfico de dispersão *boxplot*. Essa análise vai nos dar visibilidade sobre possíveis valores *outliers*, conforme pode ser visualizado na Figura 37.

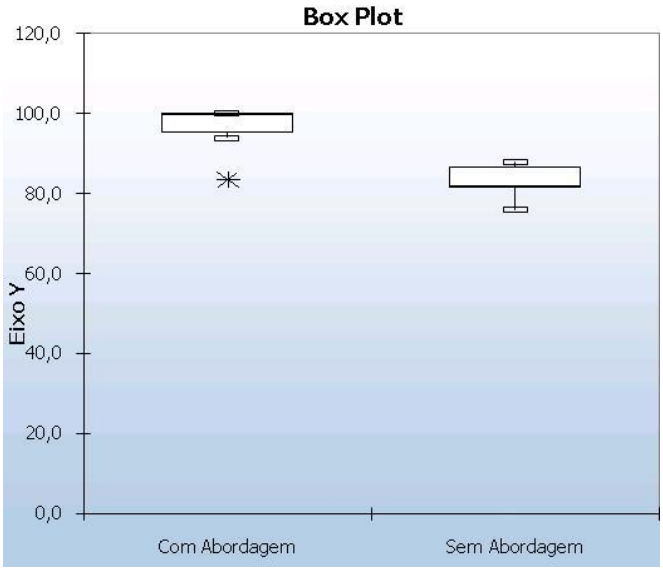


Figura 37 – Gráfico *boxplot* para a variável Precisão

Seleção da Amostra: conforme pode ser observado na Figura 37, a variável precisão apresentou um *outlier*, o participante AR10. Como critério para a eliminação

de valores anormais, optamos pela identificação numérica, a qual estabelece que valores que não alcançarem a média com dois desvios padrão devem ser excluídos. Dessa forma, obedecendo a este critério, o participante AR10 foi eliminado da amostra A Tabela 13 mostra a média e o desvio padrão calculados para a variável precisão.

Tabela 13 – Média e desvio padrão para a variável Precisão

Abordagem	Média	Desvio Padrão
Com Abordagem	97,00	6,40
Sem Abordagem	83,00	4,50

Análise da Normalidade: a etapa referente à análise da normalidade procura descobrir se os dados seguem uma distribuição normal. Para tanto, foram definidas duas hipóteses, uma *nula* e outra *alternativa*, conforme segue:

i. **H_0:** a distribuição é normal;
ii. **H_1:** a distribuição não é normal.

Levando em consideração que a variável esforço apresenta menos de 50 valores (< 50), é recomendável a utilização do teste de Shapiro-Wilk [Sha65]. A Tabela 14 mostra o resultado da execução do teste para a variável precisão.

Tabela 14 - Aplicação do teste de normalidade Shapiro-Wilk para a avaliação da variável Precisão

Variável	Abordagem	Estatística	Grau de Liberdade	Significância
Precisão	Com Abordagem	0.552	5	0,000

Resultado: conforme pode ser observado na Tabela 14, a significância dos dados do Teste de Shapiro-Wilk é inferior na abordagem baseada em Arquitetura de Referência, ao nível de significância definido (0,05 ou 5%). Dessa forma, há indícios

para rejeitar a hipótese nula, e, consequentemente, não se pode aplicar um teste paramétrico para avaliação das hipóteses. Nesse caso, conforme definido nos critérios de projeto, optamos pela aplicação do teste não paramétrico Mann-Whitney para duas amostras independentes. O objetivo da aplicação desse teste é apurar se as diferenças entre as médias são estatisticamente significativas.

Aplicação do Teste de Mann-Whitney: levando em consideração as hipóteses estabelecidas, foram consideradas as seguintes possibilidades:

i. **H_0:** não há diferença entre as médias ($\mu_{tcomp} = \mu_{tar}$);
ii. **H_1:** há diferença entre as médias ($\mu_{tcomp} \neq \mu_{tar}$).

O resultado da aplicação do teste Mann-Whitney sobre as amostras é apresentado na Tabela 15.

Tabela 15 – Teste não paramétrico de Mann-Whitney para a avaliação da variável Precisão

Variável	U de Mann-Whitney	W de Wilcoxon	Z	Sig. Assimpt. (bilateral)	Sig. Exata [2*(Sig. Unilateral)]
Precisão	0,000	21,000	-2,961	0,003	0,002a

De acordo com os resultados apresentados na Tabela 15, podemos observar que o grau de significância associado (Sig. Assimpt.) é de 0,003. Assim sendo, ele é menor que a significância assumida de 0,005, o que nos leva a rejeitar H_0, ou seja, há diferença entre as médias.

Em face deste resultado, consideramos que para a variável precisão, existe diferença entre as médias para a descrição de Arquiteturas de Referência realizadas com, e sem apoio da abordagem.

Resultado: o teste Mann-Whitney, aplicado sobre a amostra para verificar se as diferenças entre as médias são estatisticamente significativas, rejeitou a hipótese nula, mas não avaliou as hipóteses alternativas. Isto porque através desse teste não é possível realizar avaliações do tipo *maior que*, entre as amostras analisadas.

No entanto, as hipóteses alternativas podem ser verificadas por comparação, através da análise descritiva das médias da amostra, conforme apresentado na Tabela 16.

Considerando as médias apresentadas na Tabela 16, para a descrição com e sem abordagem, e os valores para cada uma delas, podemos observar que, na média, a descrição com apoio da abordagem apresentou valor mais alto, em comparação com o valor médio calculado para as descrições realizadas sem a abordagem.

Tabela 16 – Análise descritiva para a variável Precisão (com *outliers* corrigidos)

Abordagem	**Média**
Com RADA	0,996
Ad hoc	0,830

7.4 Avaliação Qualitativa

Conforme apresentado na seção sobre Engenharia de Software Experimental, um experimento, tal como o que foi realizado e apresentado até este ponto, serve apenas para a realização de uma avaliação *quantitativa*. Dessa forma, para viabilizar a realização uma avaliação *qualitativa* da abordagem proposta, abrangendo pontos como a utilidade, usabilidade e aplicabilidade, em um contexto de instanciação de arquiteturas físicas para um domínio, foi proposta a aplicação de um questionário, conforme apresentado no Apêndice I. Este questionário foi aplicado aos participantes ao final da realização do experimento, logo após a execução da atividade de instanciação de uma aplicação específica.

De acordo com o que foi definido a Seção 7.3.2.5, cada participante executou atividades relacionadas a apenas uma das abordagens, e o critério de resposta utilizado foi baseado na escala de Likert [Lik32]. A escala de Likert é uma escala bipolar psicométrica[6], comumente utilizada em questionários, onde os participantes respondem a um determinado questionamento, especificando seu nível de concordância com uma questão [Lik32]. Conforme definido por Likert [Lik32], esta escala pode ser utilizada para medir respostas negativas ou positivas. Nesse caso são utilizados quatro itens como opção de resposta (2 positivos e 2 negativos), para forçar o participante a, obrigatoriamente, concordar ou discordar com a questão. No caso desta avaliação qualitativa, foi utilizada uma escala com 5 valores, o que dá ao entrevistado a oportunidade de optar por uma opção central (nem concorda e nem discorda). As escalas de repostas utilizadas como opção de resposta para cada pergunta[7] no contexto deste experimento foram:

i. **Q1** e **Q2**: (1) *Muito Ruim*, (2) *Ruim*, (3) *Regular*, (4) *Boa* e (5) *Muito Boa*;
ii. **Q3** e **Q4**: (1) *Muito Alto*, (2) *Alto*, (3) *Aceitável*, (4) *Baixo* e (5) *Muito Baixo*;
iii. **Q5** e **Q6**: (1) *Discordo Plenamente*, (2) *Discordo*, (3) *Não Concordo nem Discordo,* (4) *Concordo* e (5) *Concordo Plenamente*.

A Tabela 17 mostra a tabulação dos resultados brutos, obtidos pela aplicação das questões relacionadas à avaliação qualitativa.

Em uma análise inicial, extraímos a média aritmética das questões para cada uma das abordagens avaliadas, conforme pode ser observado na Tabela 18.

Conforme pode ser observado na Tabela 18, a média aritmética para todas as questões relacionadas à execução das atividades propostas com o apoio da

[6] É uma área da Psicologia que faz a ponte entre as ciências exatas, principalmente a matemática aplicada a Estatística e a Psicologia [Wik10].

7 Q1 - Como você considera a usabilidade desta técnica para a instanciação de uma arquitetura?
Q2 - Como você considera a utilidade desta técnica instanciação de uma arquitetura?
Q3 - Como você considera o esforço para aprendizagem da técnica?
Q4 - Como você considera o esforço para a instanciação de uma arquitetura utilizando esta técnica?
Q5 - Você usaria na prática novamente esta técnica para a instanciação de uma arquitetura?

abordagem, foi superior a média das respostas obtidas pelos participantes que não tiveram apoio da abordagem. A questão que apresentou a maior média foi à questão Q6 (=4,50), que procurou verificar se as abordagens atendiam a finalidade proposta. Em relação ao resultado obtido para a Q6, embora se constitua na média mais alta (em favor do uso da abordagem proposta), a *diferença* para a média obtida para aqueles que não utilizaram a abordagem para instanciar uma arquitetura ficou baixa. Dessa forma, a maior parte dos participantes concorda que as duas abordagens atendem ao que se propõem. A questão que apresentou maior diferença média (=1,5) entre os participantes foi a questão Q1. Analisando percentualmente em relação ao escore total possível para cada questão (=5), 1,5 representa 30% deste total. Esta pergunta (Q1) procurou avaliar a usabilidade das abordagens, e com base na análise percentual dos resultados obtidos na Q1, observamos que a usabilidade da instanciação baseada em AR é consideravelmente maior que a instanciação baseada em features.

Tabela 17 – Tabulação de resultados brutos da Avaliação Qualitativa

Abordagem	**Participante**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
Com uso da abordagem	AR01	5	5	4	3	5	5
	AR05	4	3	3	4	4	4
	AR06	4	5	3	4	4	4
	AR08	4	4	3	4	4	4
	AR10	5	4	4	3	4	5
	AR11	4	3	3	3	5	5
Sem uso da abordagem	CO02	2	3	3	2	4	4
	CO03	3	4	3	4	4	5
	CO04	4	3	2	3	5	5
	CO07	3	3	2	4	4	4
	CO09	3	4	3	4	4	4
	CO12	2	3	2	3	4	4

Q6 - A técnica de instanciação de uma arquitetura atende ao que se propôs?

Tabela 18 – Média da satisfação das questões aplicadas na Avaliação Qualitativa

Abordagem	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**
Com uso da abordagem	4,3	4,0	3,3	3,5	4,33	4,50
Sem o uso da abordagem	2,8	3,3	2,5	3,3	4,16	4,33

Considerando a utilização da escala de Likert, com um grau de satisfação progressiva iniciando em 1 e variação até 5, verificamos que a avaliação qualitativa da proposta de instanciação baseada em Arquiteturas de Referência foi superior a instanciação baseada em *Features*.

7.4.1 Avaliação Qualitativa Complementar

Além das questões propostas e analisadas na avaliação qualitativa realizada, foram definidas outras duas questões de caráter descritivo. Estas questões, também respondidas pelos participantes logo após a execução das atividades de instanciação, com e sem o apoio da abordagem aqui proposta, tiveram como objetivo verificar que influências, o uso de uma abordagem como esta pode exercer sobre o aprendizado do Arquiteto ou Engenheiro de Domínio. Os enunciados das questões propostas, são apresentados na Tabela 18.

Tabela 19 - Questões complementares da avaliação qualitativa

Q7	*Comente, concordando ou discordando, com a seguinte premissa: "a utilização desta abordagem, de alguma forma compensou o fato do arquiteto não possuir experiência na construção de arquiteturas para o domínio em questão, e esse conhecimento, suprido pela abordagem, não fará falta no futuro, pois pode ir sendo obtido gradativamente, inclusive através do próprio uso da abordagem".*
Q8	*Comente, concordando ou discordando, com a seguinte premissa: "a utilização desta abordagem, de alguma forma compensou o fato do arquiteto*

	não possuir experiência na construção de arquiteturas para esse domínio, e esse conhecimento, suprido pela abordagem, fará falta no futuro, no momento da criação de uma nova arquitetura".

Para cada uma destas questões foi solicitado aos participantes que respondessem, concordando ou discordando, da premissa estabelecida. Uma consolidação dos resultados destas questões pode ser observada na Tabela 20.

Em geral podemos observar que, independentemente da utilização ou não de alguma abordagem, quase a totalidade dos participantes integrantes da amostra concordaram que o uso de uma abordagem compensa algum possível déficit de conhecimento do profissional em relação ao domínio. O único participante que não concordou com a Q7, e consequentemente concordou com a Q8, foi participante AR10. Analisando este caso mais especificamente, percebemos que este indivíduo foi classificado como *outlier* durante a análise estatística realizada para a variável

Tabela 20 – Tabulação de resultados da avaliação qualitativa

Abordagem	**Participante**	**Q7**	**Q8**
Com uso da abordagem	AR01	Concordo	Discordo
	AR05	Concordo	Discordo
	AR06	Concordo	Discordo
	AR08	Concordo	Discordo
	AR10	Discordo	Concordo
	AR11	Concordo	Discordo
Sem uso da Abordagem	CO02	Concordo	Discordo
	CO03	Concordo	Discordo
	CO04	Concordo	Discordo
	CO07	Concordo	Discordo
	CO09	Concordo	Discordo
	CO12	Concordo	Discordo

Precisão, ou seja, a descrição gerada por ele com o apoio da abordagem ficou abaixo da média, mais de duas vezes o desvio padrão calculado para a amostra.

Continuando a análise, verificamos que em quase sua totalidade, os participantes discordam que este conhecimento irá fazer no futuro (Q8). Isto porque, além do motivo apresentado na premissa da Q7, o uso repetido das abordagens, e a própria interação do arquiteto com os artefatos do domínio, gerados pelas abordagens, fariam com que naturalmente o profissional aprenda cada vez mais sobre o domínio.

7.5 Avaliação Experiência dos Participantes

Para a distribuição dos indivíduos nos dois grupos de participantes (um realizando as atividades com apoio da abordagem e outro sem), foi solicitado a eles, logo após o aceite ao convite de participação enviado por correio eletrônico, conforme pode ser observado no Apêndice A, que respondessem a um questionário de caracterização de experiência. Este questionário, conforme pode ser observado no Apêndice C foi aplicado com o objetivo de nivelar os dois grupos, de maneira a se constituir dois grupos com um nível de experiência mais homogêneo possível.

A Tabela 20 mostra o nível de experiência dos participantes que realizaram o experimento com apoio da abordagem proposta.

A Tabela 21 mostra o nível de experiência dos participantes selecionados para integrar o grupo que realiza o experimento sem o apoio da abordagem. Conforme pôde ser observado, analisando os dados tabulados nas duas tabelas anteriores, as médias gerais se mantiveram bastante próximas entre os dois grupos.

Importante colocar que, embora tivéssemos conseguido constituir dois grupos idênticos sob o ponto de vista de meses de experiência em todos os quesitos (desenvolvimento, projeto, reúso e domínio de celulares), assim como na formação acadêmica (mestrado, especialização e graduação), ainda assim não haveria garantia de homogeneidade, por se tratar a experiência de algo subjetivo.

Tabela 21 – Experiência do grupo utilizando a abordagem

PARTICIPANTE	EXPERIÊNCIA (em meses - grupo usando abordagem)				FORMAÇÃO[8]
	EM DESENVOLVIMENTO	PROJETO ARQUITETURAL[9]	REÚSO[10]	NO DOMÍNIO DE CELULARES	
AR01	5	2	0	1	G
AR05	32	2,5	1	1	M
AR06	10	2	1	5	G
AR08	10	1	0	1	E
AR10	3	1,5	0	1	G
AR11	12	2	0	5	G
TOTAL	**72**	**11**	**2**	**14**	-
MÉDIA	**12**	**1,83**	**0,33**	**2,33**	-

A aplicação deste questionário foi uma tentativa de evitar distorções nos resultados dos testes, através formação de grupos isentos de qualquer tipo de tendência, tanto para uma ou outra abordagem.

Tabela 22 - Experiência do grupo sem o uso da abordagem

PARTICIPANTE	EXPERIÊNCIA (em meses - grupo sem o uso da abordagem)				FORMAÇÃO
	EM DESENVOLVIMENTO	PROJETO ARQUITETURAL	REÚSO	NO DOMÍNIO DE CELULARES	

[8] Média aritmética entre a experiência em projetos de arquiteturas com e sem o uso de padrões

[9] Graduação = G, Mestrado = M e Especialização = E

[10] Sim = 1 e Não = 0

CO02	12	4,5	1	1	M
CO03	7	1	0	1	G
CO04	21	1	0	5	G
CO07	12	1,5	0	1	E
CO09	14	1	1	1	M
CO12	13	2,5	0	5	G
TOTAL	**79,00**	**11,50**	**2,00**	**14,00**	**-**
MÉDIA	**13,17**	**1,92**	**0,33**	**2,33**	**-**

7.6 Considerações

Neste capítulo foi apresentada uma avaliação quantitativa e qualitativa da proposta para a descrição de Arquiteturas de Referência, com o apoio de um estudo experimental. Também foi apresentada, ao final do capítulo, uma avaliação da experiência dos indivíduos que participaram do estudo.

O objetivo da realização deste estudo foi verificar elementos como aplicabilidade, usabilidade, utilidade, relevância, dentre outros, quando confrontada com a forma usual de documentação, observada na literatura. Durante a condução deste experimento, mesmo procurando evitar ao máximo a influência do pesquisador sobre as atividades realizadas, é possível que, mesmo sendo indesejável, essa influência (conforme prevista na literatura) possa interferir nos resultados obtidos.

Os resultados alcançados não são generalizáveis para o todo o processo de desenvolvimento de software, estando restritos ao contexto selecionado para o experimento. Este contexto abrange a descrição de uma Arquitetura de Referência, com e sem o apoio de uma linguagem de descrição, durante a fase de Projeto de um domínio. Ao longo da execução do experimento a descrição da Arquitetura de Referência com o apoio da abordagem se consistiu na codificação de elementos arquiteturais (inseridos no *template* de arquitetura organizado em camadas), usando a sintaxe específica da extensão da ADL proposta. Já a descrição realizada sem o apoio

da abordagem, foi realizada com base em um diagrama de componentes, conforme apêndice J, e foi apoiada pela ferramenta CASE Jude e descrição textual. Importante destacar que muitos dos participantes que realizaram a descrição *ad hoc*, preferiram trabalhar com prototipação em papel. Embora utilizando recursos diferentes, as descrições realizadas utilizaram os mesmos componentes relacionados ao domínio de telefonia celular.

Durante a realização do experimento os participantes coletaram uma métrica de execução das atividades, relacionada ao esforço para a documentação (tempo gasto em minutos). Outra métrica utilizada, a precisão, foi calculada posteriormente pelo pesquisador com base nos artefatos gerados. O cálculo feito para se chegar a precisão dividiu o número de elementos documentados por cada participante, pelo número total esperado de elementos documentáveis (=17) para uma Arquitetura de Referência deste domínio, conforme Apêndice N.

A análise estatística realizada com base nas informações referentes ao esforço, iniciou com a elaboração de um gráfico *boxplot*, que não identificou nenhum valor anormal (*outlier*). Após, foram realizados testes para verificar a normalidade e homocedasticidade, que apontaram para uma distribuição normal dos dados, pela diferença estatisticamente não significante observada entre suas variantes. Baseados nisso, foi executado o Test T (teste paramétrico), e foi verificada a validade da hipótese alternativa que define que o esforço para descrição de Arquiteturas de Referência com o apoio da abordagem é menor que esforço necessário sem a abordagem.

Após a avaliação da variável esforço, foi realizada a avaliação da variável precisão. A análise inicial feita sobre a variável precisão, iniciou também com a elaboração de um gráfico *boxplot*, o qual identificou um valor anormal (*outlier*), que foi removido da amostra. Diferentemente da variável esforço, os testes realizados sobre a variável precisão, não revelaram uma distribuição normal dos dados, o que impediu a realização de um teste paramétrico. Conforme estabelecido nos critérios de projeto para este caso, foi então realizado um teste não paramétrico (Mann-Whitney). Este teste revela se as diferenças entre as médias são significativas. A sua aplicação para os dados referentes a precisão, revelou que existe diferença significativa entre os

grupos. Dessa forma, a hipótese nula (H_0) foi rejeitada. Como esse teste não permite a realização de avaliações do tipo *maior que*, as hipóteses alternativas não foram avaliadas. Não sendo possível avaliar *estatisticamente* as variáveis alternativas, optamos por comparar as médias utilizando uma *análise descritiva*. O resultado desta avaliação mostrou que a precisão da descrição realizada com apoio da abordagem é maior do que a precisão da descrição realizada sem o apoio da abordagem, pois a sua média apresentou um valor mais alto.

Com o objetivo de avaliar qualitativamente a abordagem, mais especificamente a sua etapa de instanciação, foi também realizada uma avaliação qualitativa. Esta avaliação, realizada através de um questionário, procurou avaliar itens como usabilidade, utilidade, aplicabilidade. Analisando os resultados com base nas médias das notas atribuídas pelos participantes para questões aplicadas (Q1, Q2, Q3, Q4, Q5, Q6), observamos que as atividades realizadas com apoio da RADA obtiveram média mais alta que as atividades executadas sem o apoio da RADA.

Durante a realização do experimento, mais especificamente na criação da documentação, onde a variável envolvida foi o esforço, observamos que as descrições realizadas com a abordagem estavam sendo feitas de maneira ágil. Isto pode ter ocorrido pelo fato do arquiteto já partir de um modelo de organização hierárquica da Arquitetura de Referência pré-definido. Por outro lado, na descrição sem a abordagem, os participantes empreenderam um tempo considerável pensando em como organizar os elementos, para só depois iniciar a elaboração da documentação.

Esse comportamento observado pode ter ocorrido também por consequência de uma possível percepção dos participantes, por estarem utilizando ou não a abordagem proposta. Isto poderia fazer com que eles tendessem a pensar que com o uso da abordagem, as atividades são mais eficientes do que sem o apoio dela.

O experimento realizado neste capítulo teve como escopo a construção, dentre outras atividades, de uma documentação para arquiteturas de referência no domínio da telefonia celular. É sugerido, como forma de ampliar o conhecimento sobre metodologias para a documentação deste tipo de artefato, a realização de replicações deste experimento aplicadas a outros domínios. Esta generalização pode proporcionar

o surgimento de novos *insights* sobre a proposta aqui apresentada. Para tanto, sugerimos também, além do aumento no número de participantes, a aplicação de treinamento mais detalhado sobre os conceitos relacionados ao tema, uma vez que se trata de uma área muito específica, o que torna difícil a compreensão dos conceitos relacionados, mesmo para profissionais experientes e com formação acadêmica.

APÊNDICE P – A SURVEY ON SOFTWARE ARCHITECTURES FOR DOMAIN APPLICATIONS: THE INDUSTRY STATE.

Eduardo Brandes
Programa de Pós Graduação em Ciência da Computação – PUCRS
Av. Ipiranga, 6681 Prédio 32, ZIP: 90619-900 - Porto Alegre - RS - Brazil
eduardo.brandes@pucrs.br

Abstract

This paper shows the results obtained by a survey which was developed to evaluate the practices of software architectures creation for domain applications in organizations. Based on these results, a discussion is presented showing the current state of the industry concerning the construction of software architectures with the focus on reuse. It also presents how the existent methodologies are able to provide the industry necessities in this aspect. The research has been conducted with the elaboration of a survey for software architects. The questions were based on a previous study. It evaluated approaches for the construction of reference architectures from the point of view of industry experiences. The results of this study may improve the understanding on how to build software architectures for domain applications in the organizations.

1 INTRODUCTION

Considering the activities supported by software engineering, reuse is seen as one of the most promising ways of improvement in the software production [2]. Development processes that promote reuse bring a potential profit regarding time of development, contributing significantly to increase the quality of the products, and helping to reduce the time to market, making the organizations more competitive [1].

In the reuse context, there are many artifacts that can be reused, e.g., experience, source code, test cases, projects, architectures [3]. The approaches proposed in the last years have focused on the source code reuse, despite other artifacts produced during the application construction process. Hence, a few methods have been proposed to support the creation of reusable architectures. These reusable architectures are called Reference Architectures, and it impose a pattern able to represent all architectures in a domain [21]. It can be observed that the artifacts produced to the software architecture depend more on the professionals experience than any other consolidated technique.

Software Product Line (SPL) [5] is a reuse approach that emphasizes the creation of different application instances based on a given domain, i.e., software architectures for reuse. This approach provides a systematic creation of several similar products aiming lower costs, shorter time and better quality [9]. In SPL approach, the architect must identify the variabilities concerning the domain and those that will be evaluated during the creation of a specific application. The variabilities are essential in SPL and must be observed when building reference architectures for a given domain. The correct identification and use of the variabilities within the reference architecture contribute to a successful SPL [9].

This paper presents a survey that identifies issues considered by software architects when building architectures and some ideas about how we can address the variability in the existing architectures for SPL. The organization of the paper presents an initial background section where it is described how the survey questions were proposed. Then, we present how the questions were grouped through a quantitative analysis. Finally, there is a discussion on the qualitative point of view of the responses. We also present the lessons learned with this research.

2 RESEARCH METHOD

Empirical analysis over a method, process, technique or tool has been one of the most accepted ways to evaluate or validate a scientific study. Travassos et al., in [16], show that the number of empirical studies in Software Engineering has increased in the last years, allowing researchers of this area to select a great amount of empirically proven assertions to base their works.

The survey approach is one of the most common empirical method. Case studies and controlled experiments are other examples [17]. A survey can be defined as a method of gathering information from a sample of individuals. This sample represents just a fraction of the studied population. Consequently, it may be projected to a larger population [18].

In this work we have chosen the survey method due to the fact that it is adequate to our purpose of collecting information about practices and experiences of software architects. To guide our survey, we have used a structured process proposed by Pfleeger & Kitchenham [19]. This process suggests ten guidelines *described below:*

1. **Setting objectives:** our main goal is to extract information from professional software architects, of their practices and methods in this role, in order to discover the state of industry in software architecture construction. Further details in section 3;
2. **Planning and scheduling the survey:** the survey was planned after a literature review that is explained in Background section. The questions were created and published in a web site. Thirty-two invites were sent for professionals and the survey was online during one month.
3. **Ensuring that appropriated resources are available:** the resources needed were the participants and a web site to publish the survey. The invited participants work in a technological center that has many software companies allocated. To publish the question form we have used the Google Docs platform [20].
4. **Designing the survey:** the survey contains a set of questions regarding the professional profile, practices and methods from the software architects. Detailed information about the questions are presented in section 4;
5. **Preparing the research instrument:** The research instrument was constructed through a web form with 25 multiple-choice questions and descriptive questions. The questions are shown in section 4;
6. **Validating the instrument:** the questions proposed in this survey were approved previously through tests with two experts in software architecture;
7. **Selecting participants:** we have taken software architects working on different kinds and sizes of companies in a technological center at south of Brazil.
8. **Administrating and scoring the instrument:** it was invited 32 architects and after one month we received 21 answers. After this period we finished the survey and start the data analysis;
9. **Analyzing the data:** The data was analyzed quantitatively in order to establish a ranking of the most used practices. The data analysis is shown in section 5;
10. **Reporting the results:** the results are shown in section 5.

In the next sections, we present the literature review, the survey development and its results.

3 BACKGROUND

This survey is a result of a literature research which concerns the existing software architecture construction support, in particular reference architectures for domain applications.

To create this survey we adopted the following strategy: 1) it has considered practical issues, looking for information about procedures carried out by the industry during the construction of reference architectures (RA) and 2) a systematic review of literature in which was identified practical issues of each methodology to support the RA construction. [15].

In this sense, we analyzed different proposal to construct RA. We can observe that the criteria to guide the construction of RA are no really established. Besides, the analyzed works does not show the process used to assert that an architectural element represents reusable artifacts of the domain and the way that they are specified into the architecture.

During the literature review we also analyzed methodologies to guide the construction of reference architectures or domain architectures, such as FODA [14], FAST [5], FORM [6], Kobra [7], QADA [8] and COPA [4]. We can observe that all of them are based on features model to identify the domain variabilities, and, as a consequence, the creation of the RA of a given domain. These methods emphasize the functional

requirements of the domain to guide the construction of RA. Nevertheless, Bosh [15] proposes a methodology the guides the construction of RA also based on the quality of requirements. So, this last approach can be considered more suitable compare to the other ones.

Based on the research performed in this wrk, we can propose the questions that compoe the survey. The following section describes the character and the goal behind the proposed questions.

4 PROPOSED QUESTIONS

The main goal in the survey is to gather information on the practices of software architecture construction with focus on reuse. Through this survey the architects may judge the important issues to be present into the architecture, which support several similar applications of a given domain. These elements can be understood like models, standards and patterns regarding to the importance of the non-functional requirements, imposition of the customer requirements, time to market, and other issues discussed along this section.

To reach this goal, the questions were prepared to extract answers that could point to these elements qualitatively and quantitatively. So, besides the information on which the elements are presented into the architecture, we might have a quantitative data to analyze the artifacts that are common into different architectures.

Considering the issues discussed in the previous section, the questions were prepared to analyze: (i) the experience as software architect and the architectural artifacts of the applications built by the architect; (ii) the professional profile; and (iii) the organization profile.

i. **Experience**: we have checked how much the professional experience in practical and theoretical aspect is important. The questions were prepared to observe: 1) the time of experience of each interviewee as software architect; 2) the type of applications that were developed and, 3) if the interview works as architect of a software product family. The questions related to this item are the following:

1. *How many time did you work as a software architect?*
2. *Which kinds of applications did you work as software architect?*
3. *Have you ever worked on the development of a software product family architecture?*

ii. **Professional skill**: questions of this item concerned the interviewee professional profile. We try to identify if there is a framework or patterns used during the creation of the architecture and which important information about the project eventually guide the choice of such framework/pattern. Also, we have asked which elements are critical during the choice of the artifacts that will compose the application core. Besides, the survey asks the influence of functional and non-functional requirements in the patterns selection. Other questions regard the criteria used to group artifacts in an architectural element, and if some technique is used to group components into architectural elements.

Besides the items above, we try to find out how quality requirements (such as availability, coupling, extensibility, flexibility, maintainability, performance, separation of concepts, scalability, security and usability) interfere in the organization of the applications core architecture.

Due to the importance of the architect initiative to make architectural decisions and to use methods and practices, the questions of this item required a special attention. Although fifteen questions were proposed, only the most relevant questions are shown here due to space limitations:

1. *Do you usually adopt design patterns, architectural patterns or consolidated frameworks to construct software architecture?*
2. *Which information regarding the project will determine the choice of one specific pattern or framework?*
3. *Which information leads you to decide if artifacts are part of the application core or aren't? Which criteria are used to grouping artifacts into architectural element (i.e., dependencies, the number of required interfaces, the type of interchanged messages)? Explain.*
4. *In your experience, the functional requirements of an application impose some architectural patterns?*
5. *How the quality requirements (e.g., scalability, performance) influence the organization of the application core?*
6. *Are there non-functional requirements which are common in the applications that you have worked?*
7. *In your experience, the application construction to one single customer is different from the construction of the same application to different customers?*

iii. **Profile of the Organization**: the questions of this item were meant to identify the level of existent reuse

in the organization, as well as the architectural and design patterns that are most adopted. Also, if there is imposition of some standard for construction of new architectures, or if there are conflicts among architectural patterns and the organization ones. The proposed questions are in the following:

1. *Is there some standardization in your organization regarding the software architecture construction?*
2. *Is there some standardization in the software architecture that you developed?*
3. *Is there some standardization in your organization that you have developed?*
4. *Considering the question above, if the response was positive, which patterns your company makes use?*
5. *Are there any imposed patterns or framework for architecture construction in your company? If so, this imposition was valuable? Adaptations are needed frequently?*
6. *Considering the existence of some patterns or frameworks imposed by your company, which adaptations are usually needed?*
7. *How much of an architecture can be reused in a new architecture creation activity?*
8. *Are there some conflict among imposed patterns/frameworks and the logic organization of the components?*
9. *Considering the question above, if the response was positive, which are the most common conflicts?*

5 ANALYZED RESULTS

This section presents the results obtained through the proposed survey. The survey was blind, i.e., the answers are not associated to any professional or organization. Although they have not been identified, we assert that this survey includes professionals of national and international companies which are manufacturers of hardware and software in general. The next sections present the results of the first item questions.

5.2 Architects and Organizations Analysis

As presented in section 3, the questions were prepared and arranged into 3 categories. Here we present the results of the first category.

i. Profile of Professionals: the majority of the professionals (42,85%) work as software architect for for 1 to 3 years. In addition, 29% of participants work as architect in software product line projects.

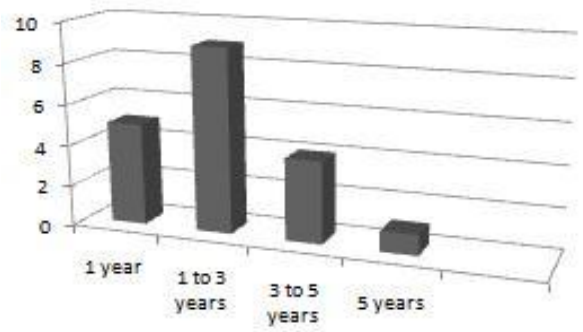


Figure 1. Know-how of the participants

The survey also asked about the type of application (commercial, industrial, etc) that are developed by the companies. In this sense, 62,50% of them worked in transactional applications, followed by distributed applications with 31,25 % of the total. This information is very relevant due to the commercial frameworks give a high support for transactional applications (CRUD's). So, this information can explain the great popularity of this type of framework in the software industry. Frameworks concern one way to promote software reuse with great results. Mostly because they also represents architecture reuse.

ii. Regarding the professional profiles, there is a set of questions that ask about the activities performed during the construction of new software architecture.
The first question refers to the activities performed during the domain analysis to identify core elements at the application. We can observe that 63,5% of the interviewee need to execute an especial activity to identify core elements of the application. Considering the descriptive question related to this subject, the participants report that the identification of core elements is an informal activity. It consists basically of a brainstorm among expert's architects and analysts. This assumption enforce to us the difficult also presented in the literature to really establish a systematic way to obtain core elements during the construction of a RA. It also was observed that the criteria used to group products/artifacts in an architectural element are related to the reduction of the number of dependences among elements to increase the cohesion of the application. This cohesion can be obtained through the grouping of components with different structural features, which would promote a low coupling between the different architectural

elements and improve the reuse of architectures. Since the elements were identified, the existent dependency among them has been the main aspect for the selection of elements that need to be part of the application.

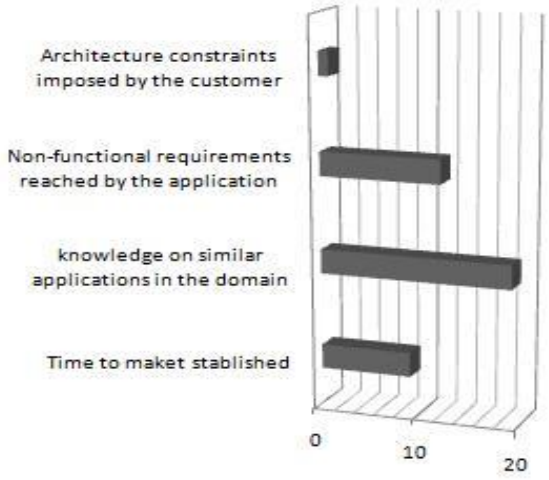


Figure 2. The influence of the Project Information

The influence of functional requirements to select a pattern or framework and other information about the project to make decision about the use of them were also analyzed. In this survey, 45,50% of the professionals said that functional requirements have few influence at the architecture creation activity. On the other hand, as shown in Figure 2, the non-functional requirements are very important when architects must make decisions about the architecture of the application. Besides, Figure 2 shows that the time to market is considered an important aspect, which has influence in the adoption of architectural patterns or frameworks to organize the new application architecture. Considering the RA construction, these limitations can be bigger, since the variabilities concerning the domain also influence the choice of a framework or pattern that will be used to construct an architectural element of the domain.

iii. The questions of this item were proposed to obtain the profile of the organizations analyzed. It was evaluated the reuse infra-structure or reuse initiatives adopted during the architecture construction for one application or for an application family.

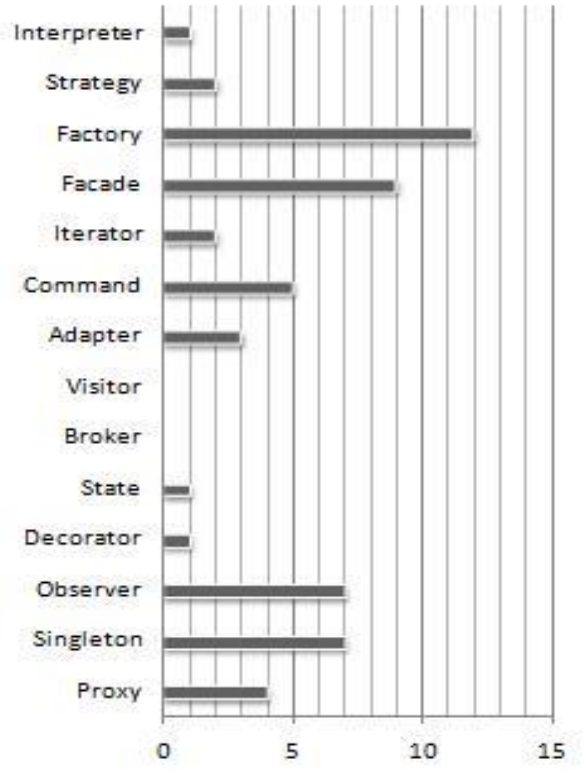


Figure 3. Design Patterns

Initially, we recognizes the infrastructure in which the architectures are developed. The questions aims at the identification of the level of existent standards on the organization. So, it was possible to know the most recurrent used frameworks and architectural and design patterns. Regarding the use of design patterns, as shown Figure 3, there was a homogeneous distribution among the structural, creation and behavioral patterns [22]. The patterns that had more applicants were the creation factory, and the behavioral observer. Regarding the used frameworks, the enterprises ones were divided practically in two equivalent groups. In one of them it can be arranged those that use proprietary frameworks built on basis of consolidated standards. In the second group, there are the enterprises that use the specification JEE combined within some commercial framework – e.g., Struts, Hibernate, Spring, Velocity - to compose their architectures. Regarding the architectural patterns adopted, as shown Figure 4, we note that the standard MVC is largely used, representing 52,85 % of the total.

In this sense, during the construction of a RA, the functional and non-functional requirements must be customized into the infra-structured required by the framework or enterprise. So, it must be think about a way to guide the software architect to support the domain variabilities into this imposed infra-structure.

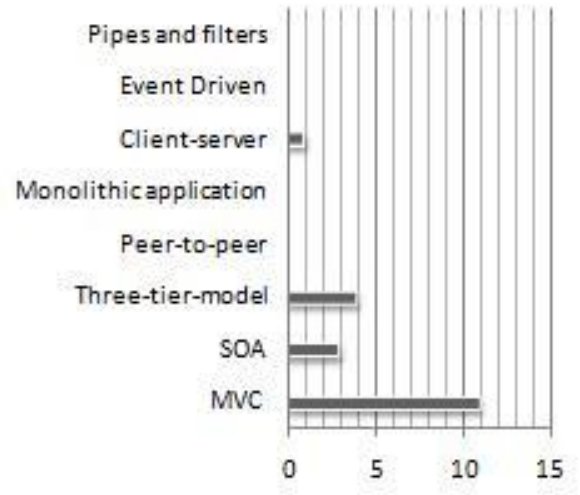


Figure 4. Architectural Patterns

Note that due to the fact that the analyzed sample is composed by organizations that mostly developed transactional applications, patterns as Peer-to-peer, Event Driven Architecture and Pipes and filters, were not reviewed.

5.3 Non-Functional Requirements Analysis

Regarding non-functional requirements, we have tried to identify those observed in more than one instance of architecture. This group of questions tries to find out answers to key problems analyzed in this paper, since non-functional requirements correspond to the main element that must be filled by an architecture. Even more when it comes to an architecture which is expected to instantiate several others for different kinds of applications in a domain. As it was mentioned, the non-functional requirements that presented the largest reuse were performance, scalability and security, followed by requirements such as flexibility and maintainability. Some important requirements with information hidden and separation of concerns were not mentioned. That is explained because most architectures are based fundamentally in a larger framework which has in its background implicitly these non-functional requirements contemplated.

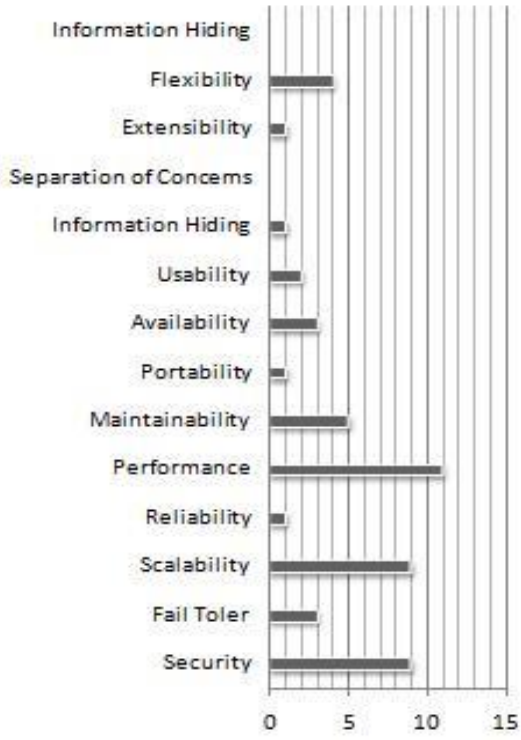


Figure 5. Non-Functional Requirements

It was also proposed additional descriptive question that attempted to identify in what way the requirements of quality have influence in the architecture core organization. According to the architects' group sample analyzed, the core architecture must be as flexible as possible to facilitate the development of features not planned, that is, emphasizing the flexibility and promoting scalability through the constant demand to maintain a low coupling among the modules. Others responded that non-functional requirements, as shown in Figure 5, determine which elements will be part of the core architecture, e.g., cache, queue control, load balancing, fault-tolerance, as well as being a factor that directly influences the choice of architectural patterns and technologies to be used in an architectural project.

6 LESSONS LEARNED

During the execution of the inquiry reported, we have found some difficulties. The problems varied from the selection of the tool used to capture the answers, up to the formation of the group of participants.

The form used to post the online questions was built on top of the freely available tools in the Internet for managing online documents. Besides the lack of

customization of the tool regarding creation of questions of multiple choice with quantitative values – e.g., percentage range values – it also did not present any features referring the consolidation of the provided results.
The lack of flexibility to create the type of necessary questions for the survey resulted in the partition of one question in two, sometimes even three, which ended up adding an unnecessary size to the survey. The lack of an interface for consolidation of the results was not flexibility a major problem because it hardly caused an overhead of work to calculate the results and to create the charts to each new computed answer, since the size of the sample was relatively small.
Another difficulty was to constitute a representative group of professional participants. Besides the fact that the software family of products' architect's profession is not a very common profession, the enterprises, mainly the multinational ones, present restrictions under the form of politics and agreements of confidentiality, preventing their professionals from participating of this type of interview. This had a quite deep impact in the inquiry, since it reduced perceptibly the available sample to be analyzed. Specially because reuse is represented more potentially in the organizations of large port.

7 CONCLUSIONS

This work has shown that enterprises are very dependent of professionals experience for the construction of architectures for their products. That came to agree one of the steps proposed by Bosh in his Features Based Architecture Design method [15], which predicts the use of experience and knowledge on domain features as the form of carrying out the core elements identification, as well as in the grouping and decomposition of the architecture in modules.
The work also pointed to a strong tendency of the organizations regarding adoption of proprietary standards and commercial frameworks for the composition of its application architectures. The consequence of this fact is that a smaller concern with the non-functional requirements can affect the quality of architecture negatively. This happens because the frameworks which are able to provide many of these requirements, still need to be customized. Hence, the architect's attention on some important non-functional requirements (such as information hiding and separation of concerns) has grown with the advance of the frameworks capabilities. Besides that, the frameworks have become greater at a point that it imposes many architectural patterns by itself.
Finally, the results have shown that, in general, the industry is not focusing on reuse, not even regarding software product lines. The lack of consolidated methods added to the advance of the commercial frameworks are the main reasons that make it difficult the creation of software architectures for domain specific applications.

8 FUTURE WORK

From the results obtained by this work, it will be possible to improve the understanding of the real necessities of those enterprises that have initiatives of reuse. Through a clearer understanding, the adaptation of existent approaches will be possible, and also the preparation of more compatible new approaches with what the market is currently demanding.

9 REFERENCES

[1] D.C. Rine, Success Factors for Software Reuse That are Applicable Across Domains and Businesses: Proceedings of the 1997 ACM symposium on Applied computing, 1997.

[2] S. Cohen, Predicting When Product Line Investment Pays, Software Engineering Institute (SEI), ICMU/SEI-2003-TN-017, July 2003.

[3] H. Gomaa, Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures: Addison Wesley, 2004.

[4] P. America, H. Obbink, J. Muller, and R. van Ommering, COPA: A Component-Oriented Platform Architecting Method for Families of Software Intensive Electronic Products, .Denver, Colorado: The First Conference on Software Product Line Engineering, 2000.

[5] D. Weiss, C. Lai, and R. Tau, Software product-line engineering: a family-based software development process. Addison-Wesley, Reading, MA, 1999.

[6] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, Annals of Software Engineering, vol. 5, 1998, pp. 143 - 168.

[7] C. Atkinson et al., Component-based product line engineering with UML. Addison-Wesley, London, New York, 2002.

[8] M. Matinlassi, E. Niemelä, and L. Dobrica, Quality-driven architecture design and quality analysis method, A revolutionary initiation approach to a product line architecture, VTT Technical Research Centre of Finland, Espoo, 2002.

[9] K. Pohl, A. Metzger, Variability Management in Software Product Line Engineering, ICSE 2007 Companion. 29th International Conference on, May 2007.

[10] U. Eklund, Ö. Askerdal; Granho, A. Alminger, A. Jakob. Experience of introducing reference architectures in the development of automotive electronic systems: ACM May, 2005.

[11] W. Godfrey, A. Grosskurth, A Reference Architecture for Web Browsers: Proceedings of the 21st IEEE International Conference, 2005.

[12] M. Haahr, A. Sing. Peer-To-Peer Reference Architecure: Communication System Software and Middleware, 2006. Comsware 2006. First International
Conference on Volume , Issue , 08-12 Jan. 2006 Page(s): 1 – 10.

[13] D. Batory, M. Coglianese, M. Goodwin, S. Steve. Creating Reference Arquitectures: An Example from Avionics: Proceedings of the 1995 Symposium
on Software reusability. Publisher: ACM, August 1995.

[14] K. C. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, Feature-Oriented Domain Analysis. Feasibility study Software Engineering Institute, Pittsburgh CMU/SEI-90-TR-21, 1990.

[15] J. Bosh, Design and use of Software architectures: ACM Press Addison-Wesley, May 29 2000.

[16] G.H. Travassos, D. Gurov, E.A.G.G. Amaral, 2002, Introdução à Engenharia de Software Experimental, Relatório Técnico ES-590/02-Abril. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

[17] C. Wohlin, P. Runeson, M. Höst, et al., 2000, Experimentation in Software
Engineering: an introduction, USA, Kluwer Academic Publishers.

[18] F. Scheuren, What is a Survey. 2004. Available at http://client.norc.org/whatisasurvey/downloads/pamphlet_current.pdf. Captured in 10/2008.

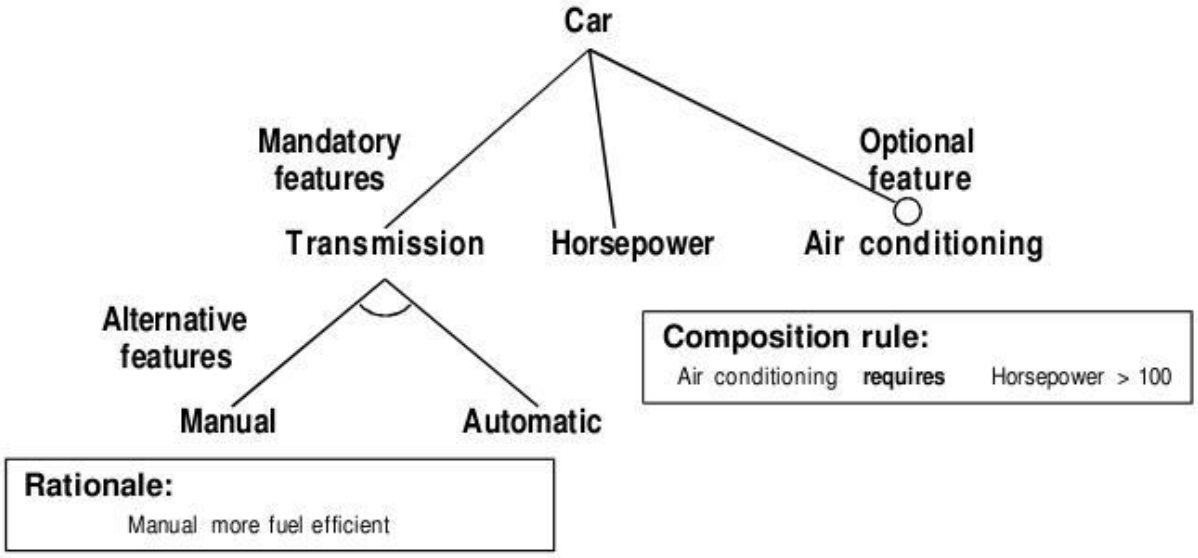
[19] Pfleeger and Kitchenham (2001). Principles of survey research. Parts 1-5. Software Engineering Notes.

[20] Google, Google Docs Beta. 2008. Available at http://docs.google.com. Captured in 10/2008.

[21] A. MENDES, Arquitetura de Software: desenvolvimento orientado a arquitetura, Rio de Janeiro, Editora Campus 2002.

[22] D. BRAUN, J. SILVILS, A. SHAPIRO, J. VERSTEEGH, Design Patterns: Modeled in UML. Object Oriented Analysis and Design Team
,Kennesaw State University, CSIS 4650 – Springer

ANEXO A – MODELO DE FEATURES: NOTAÇÃO



Fonte: Extraído de [Kan90]

ANEXO B – TREINAMENTO DA NOTAÇÃO ODYSSEY-FEX: LEITURA INTRODUTÓRIA

A Engenharia de Domínio (ED) é uma área da Engenharia de Software cujo principal objetivo é desenvolver artefatos de software para uma família de aplicações, de modo que estes possam ser reutilizados em aplicações deste domínio.

A ED incorpora uma etapa de Análise de Domínio onde o engenheiro do domínio deve obter os requisitos do domínio os quais podem representar características funcionais, conceituais, tecnológicas, que por sua ver podem ser ou não obrigatórias para todas as aplicações derivadas deste domínio, e ainda podem ter diferentes formas de implementação. Neste sentido, é importante que na análise de domínio se tenha uma forma sistemática de representar estas diferentes características, bem como suas opcionalidades e variabilidades e, não menos importante, propagar estas propriedades para os diferentes níveis de abstração do domínio. No estudo de observação que você irá participar é utilizada uma notação (Odyssey-FEX) para representar todas as características acima mencionadas. Nesta notação, as características podem ser classificadas quanto à sua categoria, variabilidade e opcionalidade.

Tabela 1 – Tipos de Características na notação Odyssey-FEX

Ícone	Tipo de Característica	
	**Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades e/ou os conceitos do modelo e correspondem a casos de uso e componentes estruturais concretos.	Características de Análise
	**Características de Entidade** – São os atores do modelo. Entidades do mundo real que atuam sobre o domínio. Podem, por exemplo, expor a necessidade de uma interface com o usuário ou de procedimentos de controle.	
	**Características de Ambiente Operacional** - Características que representam atributos de um ambiente que uma aplicação do domínio pode usar e operar. Ex: tipo de terminal, sistemas operacionais, bibliotecas etc.	Características Tecnológicas
	**Características de Tecnologia de Domínio** - Características que representam detalhes de implementação de mais baixo nível, específicos para o contexto de um domínio. Ex: métodos de navegação em um domínio de aviões.	
	**Características de Técnicas de Implementação** – Características que representam detalhes de implementação de mais baixo nível, contudo de cunho mais genérico que as relativas à camada de tecnologia de domínio. Ex: técnicas de sincronização.	

Quanto à sua categoria, as características podem ser classificadas conforme apresento na Tabela 1. Estas características são mutuamente excludentes, ou seja, não podem pertencer simultaneamente a mais de uma categoria. As características na notação Odyssey-FEX possuem a seguinte classificação quanto a sua variabilidade:

i. **Ponto de Variação**: são as características que refletem a parametrização no domínio de uma maneira abstrata. São configuráveis através das variantes.
ii. **Variantes:** são características que atuam como alternativas para se configurar um ponto de variação.
iii. **Invariantes:** são as características fixas, que representam elementos não configuráveis em um domínio.

A classificação das características quanto a opcionalidade indica justamente a obrigatoriedade ou não obrigatoriedade da presença de um determinado elemento nas aplicações ou produtos a serem desenvolvidos. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. Características que são opcionais em relação ao domínio, mas que por ventura venham a ser mandatórias em relação a outras características a serem selecionadas, deverão expressar essa informação através de Regras de Composição. Características opcionais são evidenciadas no modelo por um contorno pontilhado.

Fonte: Extraído de [Bac06]

ANEXO C – DISTRIBUIÇÃO T

gl	0,25	0,10	0,05	0,025	0,01	0,005	0,0025	0,001	0,0005
1	1,000	3,078	6,314	12,71	31,82	63,66	127,3	318,3	636,6
2	0,816	1,886	2,920	4,303	6,965	9,925	14,09	22,33	31,60
3	0,765	1,638	2,353	3,182	4,541	5,841	7,453	10,21	12,92
4	0,741	1,533	2,132	2,776	3,747	4,604	5,598	7,173	8,610
5	0,727	1,476	2,015	2,571	3,365	4,032	4,773	5,894	6,869
6	0,718	1,440	1,943	2,447	3,143	3,707	4,317	5,208	5,959
7	0,711	1,415	1,895	2,365	2,998	3,499	4,029	4,785	5,408
8	0,706	1,397	1,860	2,306	2,896	3,355	3,833	4,501	5,041
9	0,703	1,383	1,833	2,262	2,821	3,250	3,690	4,297	4,781
10	0,700	1,372	1,812	2,228	2,764	3,169	3,581	4,144	4,587
11	0,697	1,363	1,796	2,201	2,718	3,106	3,497	4,025	4,437
12	0,695	1,356	1,782	2,179	2,681	3,055	3,428	3,930	4,318
13	0,694	1,350	1,771	2,160	2,650	3,012	3,372	3,852	4,221
14	0,692	1,345	1,761	2,145	2,624	2,977	3,326	3,787	4,140
15	0,691	1,341	1,753	2,131	2,602	2,947	3,286	3,733	4,073
16	0,690	1,337	1,746	2,120	2,583	2,921	3,252	3,686	4,015
17	0,689	1,333	1,740	2,110	2,567	2,898	3,222	3,646	3,965
18	0,688	1,330	1,734	2,101	2,552	2,878	3,197	3,610	3,922
19	0,688	1,328	1,729	2,093	2,539	2,861	3,174	3,579	3,883
20	0,687	1,325	1,725	2,086	2,528	2,845	3,153	3,552	3,850
21	0,686	1,323	1,721	2,080	2,518	2,831	3,135	3,527	3,819
22	0,686	1,321	1,717	2,074	2,508	2,819	3,119	3,505	3,792
23	0,685	1,319	1,714	2,069	2,500	2,807	3,104	3,485	3,768
24	0,685	1,318	1,711	2,064	2,492	2,797	3,091	3,467	3,745
25	0,684	1,316	1,708	2,060	2,485	2,787	3,078	3,450	3,725
26	0,684	1,315	1,706	2,056	2,479	2,779	3,067	3,435	3,707
27	0,684	1,314	1,703	2,052	2,473	2,771	3,057	3,421	3,689
28	0,683	1,313	1,701	2,048	2,467	2,763	3,047	3,408	3,674
29	0,683	1,311	1,699	2,045	2,462	2,756	3,038	3,396	3,660
30	0,683	1,310	1,697	2,042	2,457	2,750	3,030	3,385	3,646
35	0,682	1,306	1,690	2,030	2,438	2,724	2,996	3,340	3,591
40	0,681	1,303	1,684	2,021	2,423	2,704	2,971	3,307	3,551
45	0,680	1,301	1,679	2,014	2,412	2,690	2,952	3,281	3,520
50	0,679	1,299	1,676	2,009	2,403	2,678	2,937	3,261	3,496
z	0,674	1,282	1,645	1,960	2,326	2,576	2,807	3,090	3,291

Fonte: Extraído de [INE10]