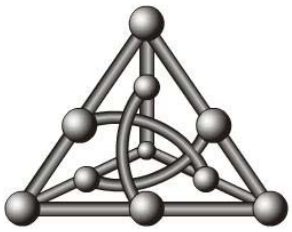
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

INSTANTFAQ: UMA FERRAMENTA WEB PARA GESTÃO DE FAQ CONSTRUÍDA SEGUNDO A ABORDAGEM ADCOMP

ÉRIKA ISHY

Dissertação de Mestrado em Ciência da Computação

Área: Engenharia de Software



DEPARTAMENTO DE COMPUTAÇÃO E ESTATÍSTICA

ÉRIKA ISHY

INSTANTFAQ: UMA FERRAMENTA WEB PARA GESTÃO DE FAQ CONSTRUÍDA SEGUNDO A ABORDAGEM ADCOMP

Dissertação apresentada como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação do Curso de Pós-Graduação em Ciência da Computação, Departamento de Computação e Estatística, Universidade Federal do Mato Grosso do Sul.

Orientador: Prof. Dr. Marcelo A. S. Turine

Campo Grande

2004

A Deus, aos meus amados pais e irmãos.

AGRADECIMENTOS

Ao longo deste período de estudos e pesquisas tenho muito a agradecer. Primeiramente, a Deus que não me deixou fraquejar e desistir, pois Ele não dá um fardo maior do que aquele que podemos carregar.

Igualmente, agradeço aos meus pais que me deram todo o apoio para que eu pudesse estudar, bem como às pessoas que convivem comigo diariamente.

Agradeço ao meu orientador pelo apoio e encorajamento na realização deste trabalho, uma vez que o tema era novo e, portanto, desafiador.

A CAPES pelo apoio financeiro a este trabalho de mestrado que incluiu viagens de estudo para São Carlos/SP onde aprimorei meus conhecimentos sobre componentes com o Prof. Dr. Antônio Francisco do Prado e seu aluno de mestrado Daniel Lucrédio, ambos do Departamento de Computação (DC) da Universidade Federal de São Carlos (UFSCar), a quem agradeço muito.

E a todos os amigos, colegas e professores que direta ou indiretamente contribuíram para a realização e divulgação deste trabalho, em especial ao Prof. Dr. Henrique Mongelli, coordenador do curso de Mestrado em Ciência da Computação do Departamento de Computação e Estatística (DCT) da Universidade Federal de Mato Grosso do Sul (UFMS), pela confiança depositada em mim.

Meninas boazinhas vão para o céu, as más vão à luta.

Ute Ehrhardt

SUMÁRIO

LISTA DE FIGURAS **8**
LISTA DE TABELAS **10**
LISTA DE QUADROS **10**
LISTA DE ABREVIATURAS E SIGLAS **11**
RESUMO **12**
ABSTRACT **13**
1. INTRODUÇÃO **14**
1.1. CONSIDERAÇÕES INICIAIS 14
1.2. MOTIVAÇÕES E OBJETIVOS 15
1.3. ORGANIZAÇÃO DO TEXTO 17
2. ENGENHARIA DE SOFTWARE BASEADA EM COMPONENTES **19**
2.1. CONSIDERAÇÕES INICIAIS 19
2.2. DEFINIÇÕES 19
2.2.1. Componentes 19
2.2.2. Modelo de Componentes 21
2.3. PROCESSO DA ESBC 22
2.3.1. Engenharia de Domínio 23
2.3.2. Desenvolvimento Baseado em Componentes 24
2.4. MÉTODOS DE DESENVOLVIMENTO 24
2.4.1. Catalysis 25
2.4.2. RUP: *Rational Unified Process* 25
2.4.3. *UML Components* 26
2.5. MODELOS DE COMPONENTES 27
2.5.1. CCM: *CORBA Component Model* 27
2.5.2. COM+: *Common Object Model* 28
2.5.3. EJB: *Enterprise JavaBeans* 29
2.6. CONSIDERAÇÕES FINAIS 31
3. ADComp: ABORDAGEM DE DESENVOLVIMENTO DE SOFTWARE BASEADA EM COMPONENTES **32**
3.1. CONSIDERAÇÕES INICIAIS 32
3.2. VISÃO GERAL 32
3.3. ETAPA 1: DEFINIÇÃO DE REQUISITOS 34
3.3.1. Modelo Conceitual de Negócio 36
3.3.2. Modelo de Casos de Uso 36
3.4. ETAPA 2: DEFINIÇÃO DE COMPONENTES 38
3.4.1. Identificação do Componente 39
3.4.2. Interação do Componente 45
3.4.3. Especificação do Componente 47
3.5. ETAPA 3: IMPLEMENTAÇÃO DE COMPONENTES 52
3.6. CONSIDERAÇÕES FINAIS 57
4. FERRAMENTA INSTANTFAQ **59**
4.1. CONSIDERAÇÕES INICIAIS 59

4.2. ARQUITETURA 59
4.3. DESENVOLVIMENTO DOS COMPONENTES 61
4.3.1. Etapa1: Definição de Requisitos 61
4.3.2. Etapa 2: Definição de Componentes 64
4.3.3. Etapa 3: Implementação de Componentes 73
4.4. FUNCIONALIDADES DA INSTANTFAQ 77
4.5. CONSIDERAÇÕES FINAIS 81
5. REUTILIZAÇÃO E INTEGRAÇÃO DA INSTANTFAQ 82
5.1. CONSIDERAÇÕES INICIAIS 82
5.2. ABORDAGENS DE INTEGRAÇÃO 82
5.2.1. Acesso Indireto 82
5.2.2. Acesso Direto 83
5.3. PROCESSO DE INTEGRAÇÃO 83
5.3.1. Regras de Integração 83
5.3.2. Integração no SIGS 87
5.3.3. Integração no TelEduc 88
5.4. AVALIAÇÃO 91
5.5. CONSIDERAÇÕES FINAIS 92
6. CONCLUSÕES 93
6.1. CONSIDERAÇÕES INICIAIS 93
6.2. CONTRIBUIÇÕES 95
6.3. TRABALHOS FUTUROS 95
REFERÊNCIAS 97

LISTA DE FIGURAS

FIGURA 2.1: RELAÇÃO ENTRE OBJETOS E COMPONENTES (BROWN, 2000, p. 78). 20
FIGURA 2.2: MODELO DE PROCESSO DA ESBC (PRESSMAN, 2001, p. 725). 23
FIGURA 2.3: SERVIDOR ENTERPRISE JAVABEANS. 29
FIGURA 2.4: TIPOS DE COMPONENTES EJB. 30
FIGURA 3.1: PROCESSO SIMPLIFICADO DA ADCOMP. 34
FIGURA 3.2: ARTEFATOS DA ADCOMP. 35
FIGURA 3.3: MCN INICIAL (CHESSMAN; DANIELS, 2001, p. 70). 37
FIGURA 3.4: MODELO DE CASOS DE USO (CHESSMAN; DANIELS, 2001, p. 81). 38
FIGURA 3.5: APLICAÇÃO EM CAMADAS (CHESSMAN; DANIELS, 2001, p. 11). 39
FIGURA 3.6: MAPEAMENTO DE CASOS DE USO PARA INTERFACES DE SISTEMA (CHESSMAN; DANIELS, 2001, p. 86-87). 41
FIGURA 3.7: REDEFINIÇÃO DO MCN INICIAL (CHESSMAN; DANIELS, 2001, p. 86-87). 42
FIGURA 3.8: MODELO DE TIPO DE NEGÓCIO (CHESSMAN; DANIELS, 2001, p. 91). 42
FIGURA 3.9: MODELO DE RESPONSABILIDADE DAS INTERFACES NO MODELO DE TIPO. 43
FIGURA 3.10: ESPECIFICAÇÕES DO COMPONENTE DE SISTEMA. 44
FIGURA 3.11: ESPECIFICAÇÕES DOS COMPONENTES DE NEGÓCIO. 44
FIGURA 3.12: ARQUITETURA INICIAL DA ESPECIFICAÇÃO DO COMPONENTE. 45
FIGURA 3.13: MODELO DE INTERAÇÃO DA INTERFACE DE SISTEMA IFAZERRESERVA. 47
FIGURA 3.14: OPERAÇÕES DA INTERFACE DE SISTEMA IFAZERRESERVA. 47
FIGURA 3.15: OPERAÇÕES DAS INTERFACES DE NEGÓCIO. 47
FIGURA 3.16: MODELO DE ESPECIFICAÇÃO DA INTERFACE DE NEGÓCIO. 49
FIGURA 3.17: MODELO DE ESPECIFICAÇÃO DA INTERFACE DE SISTEMA. 50
FIGURA 3.18: MODELO DE ESPECIFICAÇÃO DO COMPONENTE DE NEGÓCIO. 51
FIGURA 3.19: MODELO DE ESPECIFICAÇÃO DO COMPONENTE DE SISTEMA. 51
FIGURA 3.20: MODELO DE TIPO REVISADO. 53
FIGURA 3.21: MODELO DE TIPO TRANSFORMADO EM MODELO DE ENTITY BEANS. 54
FIGURA 3.22: OPERAÇÕES DAS INTERFACES HOME E GERADAS PELA TRANSFORMAÇÃO. 54
FIGURA 3.23: MODELO DE SESSION BEANS. 55
FIGURA 3.24: INTERFACES UTILIZADAS PELO SESSION BEAN. 55
FIGURA 3.25: IMPLEMENTAÇÃO DAS OPERAÇÕES. 56
FIGURA 3.26: MODELO DE COMPONENTES. 57
FIGURA 3.27: DEPLOY WIZARD. 57
FIGURA 4.1: ARQUITETURA DA FERRAMENTA INSTANTFAQ. 61
FIGURA 4.2: MODELO CONCEITUAL DE NEGÓCIO. 62
FIGURA 4.3: CASOS DE USO DO ATOR USUÁRIO-ONLINE. 63
FIGURA 4.4: CASOS DE USO APLICAÇÃO, ASSUNTO E ORIGEM DO ATOR GESTOR. 63
FIGURA 4.5: CASOS DE USO GESTOR E USUÁRIO DO ATOR GESTOR. 64
FIGURA 4.6: CASOS DE USO MENSAGEMFAQ E MENSAGEMFC DO ATOR GESTOR. 64
FIGURA 4.7: MODELO DE INFORMAÇÃO DA INTERFACE GENÉRICA IMANTERFAQ. 66
FIGURA 4.8: MODELO DE TIPO DE NEGÓCIO. 67
FIGURA 4.9: MODELO DE RESPONSABILIDADE. 68
FIGURA 4.10: INTERFACES DE NEGÓCIO. 68
FIGURA 4.11: MODELO DE COLABORAÇÃO DA OPERAÇÃO LISTARMSGFAQPUBLIC. 69
FIGURA 4.12: OPERAÇÕES DAS INTERFACES DE NEGÓCIO. 69
FIGURA 4.13: MODELO DE ESPECIFICAÇÃO DA INTERFACE IMENSAGEMFAQ. 70
FIGURA 4.14: MODELO DE COMPONENTES DE SISTEMA. 72

FIGURA 4.15: MODELO DE COMPONENTES DE NEGÓCIO. 72
FIGURA 4.16: MODELO DA ESPECIFICAÇÃO DO FALE CONOSCO USUARIO. 72
FIGURA 4.17: MODELO DA ARQUITETURA DOS COMPONENTES. 73
FIGURA 4.18: DIAGRAMA DE CLASSES PARA OS COMPONENTES DE NEGÓCIO. 74
FIGURA 4.19: MODELO DO ENTITY BEAN MENSAGEMFAQ COM SUAS INTERFACES. 74
FIGURA 4.20: MODELO DE COMPONENTES DE SISTEMA E INTERFACES. 75
FIGURA 4.21: IMPLEMENTAÇÃO DAS OPERAÇÕES DOS COMPONENTES. 76
FIGURA 4.22: MODELO DE COMPONENTES. 76
FIGURA 4.23: DEPLOYTOOL. 77
FIGURA 4.24: FUNCIONALIDADE FAQ DO USUÁRIO-ONLINE. 77
FIGURA 4.25: FUNCIONALIDADE FALE CONOSCO DO USUÁRIO-ONLINE. 78
FIGURA 4.26: NOVA MENSAGEM DE FALE CONOSCO DO USUÁRIO-ONLINE. 78
FIGURA 4.27: MENU DE ADMINISTRAÇÃO PARA O GESTOR. 78
FIGURA 4.28: CADASTRO DE APLICAÇÃO. 79
FIGURA 4.29: FAQ DO GESTOR. 79
FIGURA 4.30: NOVA MENSAGEM DE FAQ. 80
FIGURA 4.31: FALE CONOSCO DO GESTOR. 80
FIGURA 4.32: RESPONDER MENSAGEM DE FALE CONOSCO. 80
FIGURA 4.33: ENVIAR MENSAGEM DE FALE CONOSCO PARA A FAQ. 81
FIGURA 5.1: HOMEPAGE DO SIGS. 87
FIGURA 5.2: PÁGINA DE FERRAMENTAS DO SIGS E SEU FALE CONOSCO. 88
FIGURA 5.3: FAQ E ADMINISTRAÇÃO DA INSTANTFAQ INTEGRADOS NO SIGS. 88
FIGURA 5.4: TELEDUC PARA O CURSO PROJETO DE WEBSITES. 89
FIGURA 5.5: TELEDUC NA VISÃO DO ALUNO. 90
FIGURA 5.6: TELEDUC NA VISÃO DO FORMADOR. 90
FIGURA 5.7: DIRETO FAQ NA FERRAMENTA CORREIO. 91
FIGURA 5.8: DIRETO FAQ NA FERRAMENTA FÓRUM DE DISCUSSÃO. 91

LISTA DE TABELAS

TABELA 2.1: ELEMENTOS BÁSICOS DE UM MODELO DE COMPONENTE (WEINREICH; SAMETINGER, 2001, p. 38). 22
TABELA 3.1: PRIMITIVAS DO MODELO CONCEITUAL DE NEGÓCIO. 36
TABELA 3.2: PRIMITIVAS DO MODELO DE CASOS DE USO. 37
TABELA 3.3: CASOS DE USO E ATORES DO MODELO DE CASOS DE USO. 38
TABELA 3.4: PRIMITIVAS DOS MODELOS DA IDENTIFICAÇÃO DO COMPONENTE. 40
TABELA 3.5: PRIMITIVAS DO MODELO DE COLABORAÇÃO. 45
TABELA 3.6: ASSINATURAS DAS OPERAÇÕES DA INTERFACE IFAZERRESERVA. 46
TABELA 4.1: INTERFACES DE SISTEMA. 66

LISTA DE QUADROS

QUADRO 3.1: ESPECIFICAÇÃO EM OCL DE UMA OPERAÇÃO. 49
QUADRO 3.2: INVARIANTE ESCRITA EM OCL. 49
QUADRO 3.3: RESTRIÇÕES ENTRE INTERFACES OFERECIDAS. 51
QUADRO 3.4: RESTRIÇÕES ENTRE INTERFACES OFERECIDAS E UTILIZADAS. 52
QUADRO 4.1: PRÉ-CONDIÇÃO E PÓS-CONDIÇÃO DE UMA OPERAÇÃO. 70
QUADRO 4.2: RESTRIÇÕES ENTRE AS INTERFACES OFERECIDAS E AS UTILIZADAS. 71
QUADRO 5.1: REGRA DE INTEGRAÇÃO FAQ. 84
QUADRO 5.2: REGRA DE INTEGRAÇÃO FALE CONOSCO (AUTENTICAÇÃO INSTANTFAQ). 85
QUADRO 5.3: REGRA DE INTEGRAÇÃO FALE CONOSCO (AUTENTICAÇÃO WEBAPP). 85
QUADRO 5.4: REGRA DE INTEGRAÇÃO ADMINISTRAÇÃO (AUTENTICAÇÃO INSTANTFAQ). 85
QUADRO 5.5: REGRA DE INTEGRAÇÃO ADMINISTRAÇÃO (AUTENTICAÇÃO WEBAPP). 86
QUADRO 5.6: REGRA DE INTEGRAÇÃO DIRETO FAQ. 86

LISTA DE ABREVIATURAS E SIGLAS

ADComp	Abordagem de Desenvolvimento de software baseada em Componentes
API	*Application Protocol Interface*
ASP	*microsoft Active Server Pages*
BMP	*Bean-Managed Persistence*
CAPES	Coordenação de Aperfeiçoamento de PEssoal de nível Superior
CASE	*Computer-Aided Software Engineering*
CCM	*CORBA Component Model*
CEP	Código de Endereçamento Postal
CLSID	*CLaSse IDentifier*
CMC	Comunicação Mediada por Computador
CMP	*Container-Managed Persitence*
COM	*microsoft Component Object Model*
COM+	*microsoft Component Object Model+*
CORBA	*Common Object Request Broker Architecture*
DBC	Desenvolvimento Baseado em Componentes
DC	Departamento de Computação
DCT	Departamento de Computação e esTatística
DLL	*Dynamically Linked Library*
DNS	*Domain Name System*
EAD	Educação Aberta e a Distância
ED	Engenharia de Domínio
EJB	*Enterprise JavaBeans*
ESBC	Engenharia de Software Baseada em Componentes
FAQ	*Frequently Asked Questions*
GUI	*Graphical User Interface*
IEE	Instituto de Estudos Especiais
IDL	*Interface Description Languages*
IID	*Interface IDentifier*
J2EE	*Java 2 Enterprise Edition*
JDK	*Java Development Kit*
JDBC	*Java DataBase Conectivity*
JSP	*JavaServer Pages*
LEDES	Laboratório de Engenharia DE Software
MCN	Modelo Conceitual de Negócio
MCU	Modelo de Casos de Uso
MI	Modelo de Informação
NIED	Núcleo de Informática Aplicada à EDucação
OCL	*Object Constraint Language*
OMG	*Object Management Group*
PHP	*HyPertext Preprocessor*
PUC	Pontifícia Universidade Católica
RUP	*Rational Unified Process*
SADT	*Structured Analysis and Design Tecnhique*
SIGS	Sistema de Informação de Gestão Social
SIW	Sistemas de Informação Web
TIC	Tecnologias de Informação e de Comunicação
UFMS	Universidade Federal de Mato grosso do Sul
UFSCar	Universidade Federal de São Carlos
UML	*Unified Modeling Language*
UNICAMP	UNiversidade estadual de CAMPinas
XML	*eXtensible Markup Language*
WebApp	*Web-based APPlication*
WebCT	*Web Course Tools*

RESUMO

A influência positiva da Web tem motivado uma demanda de aplicações que combinam navegação e interatividade. Tais aplicações, cada vez mais complexas, necessitam de métodos e tecnologias que visam um desenvolvimento com qualidade a baixo custo. Nesta perspectiva, técnicas de reuso combinadas com métodos e tecnologias da Engenharia de Software Baseada em Componentes permitem que aplicações baseadas na Web (WebApps) sejam criadas rapidamente a partir de partes pré-construídas e testadas. Apesar das recentes pesquisas na área de Desenvolvimento Baseado em Componentes muitos métodos possuem uma didática de difícil compreensão e/ou seus processos são complexos, dificultando sua disseminação e sua utilização na prática. Desta forma, neste trabalho é proposta uma Abordagem de Desenvolvimento de Software Baseada em Componentes denominada ADComp, resultante da pesquisa de vários métodos atuais (*Catalysis*, *Rational Unified Process* e *UML Components*), objetivando definir um processo intuitivo, simples, direto e prático. A ADComp é dividida em três etapas: (1) definição de requisitos, (2) definição de componentes e (3) implementação de componentes; sendo a segunda etapa dividida nas seguintes fases: (i) identificação do componente, (ii) interação do componente e (iii) especificação do componente. Uma de suas características é modelar os componentes de forma a integrar a especificação dos componentes com sua implementação numa linguagem executável: Java como linguagem de programação e EJB (*Enterprise JavaBeans*) como modelo de componentes. Além disto, todas estas etapas e fases são desenvolvidas e gerenciadas pela ferramenta CASE MVCASE do Departamento de Computação da UFSCar que oferece suporte à UML (*Unified Modeling Language*), o modelo de componentes EJB e o padrão CORBA (*Common Object Request Broker Architecture*) para aplicações distribuídas. Para validar a abordagem é elaborado um estudo de caso de uma ferramenta Web de interatividade baseada em componentes para gestão de FAQ (*Frequently Asked Questions*) e de Fale Conosco denominada InstantFAQ. Uma ferramenta multicamada que pode ser integrada e reutilizada em diversas WebApps como mecanismo de comunicação e de coordenação para administradores e usuários. Como meio de validar suas funcionalidades e sua reutilização, a InstantFAQ é integrada em duas WebApps: (1) ambiente de Educação Aberta e a Distância TelEduc do Núcleo de Informática Aplicada à Educação da UNICAMP e (2) Sistema de Informação de Gestão Social (SIGS) para avaliação e monitoramento de programas de transferência de renda do Instituto de Estudos Especiais da PUC/SP em parceria com o Departamento de Computação e Estatística da UFMS.

Palavras-Chave: Engenharia de Software Baseada em Componentes, Desenvolvimento Baseado em Componentes, Reusabilidade, ADComp, InstantFAQ.

ABSTRACT

The positive influence of the Web has motivated a demand of applications that combine navigation and interativity. These applications are each time more complex and need methods and technologies that aim at a development with quality and low cost. In this perspective, techniques of reuse combined with methods and technologies of the Component-Based Software Engineering allow applications based on the Web (WebApps) to be quickly created from pre-constructed and tested parts. Despite the recent research in the area of Component-Based Development, many methods possess didactics of difficult understanding and/or its processes are complex, making it difficult its dissemination and its use. In this work, we propose a Component-Based Software Development approach called ADComp. It is a result of the research on some current methods (Catalysis, Rational Unified Process and UML Components). Our goal is to define an intuitive, simple, direct and practical process. The ADComp is divided in three stages: (1) definition of requirements, (2) definition of components and (3) implementation of components. The second stage subdivided in the following phases: (i) identification of the component, (ii) interaction with the component and (iii) specification of the component. One of its characteristics is to shape the components in order to integrate its specification and implementation in an executable language: Java as programming language and EJB (Enterprise JavaBeans) as model of components. Moreover, all these stages and phases are developed and managed for the CASE tool MVCASE from the Departamento de Computação of the UFSCar that offers support to UML (Unified Modeling Language), the model of components EJB and standard CORBA (Common Object Request Broker Architecture) for distributed applications. A case study of an interactive Web tool is elaborated to validate the approach. It is based on components for FAQ (*Frequently Asked Questions*) and phone calls management called InstantFAQ, a multilayer tool that can be integrated and be reused in several WebApps as mechanism of communication and coordination for administrators and users. To validate its functionalities and reutilization, the InstantFAQ is integrated in two WebApps: (1) The Open Education and Distance environment “TelEduc” from the Núcleo de Informática Aplicada à Educação of the UNICAMP and (2) Social Management Information System (SIGS) for evaluation and management of income transference programs from the Instituto de Estudos Especiais of the PUC/SP in partnership with the Departamento de Computação e Estatística of the UFMS.

Key-Words: Component-Based Software Engineering, Component-Based Development, Reusability, ADComp, InstantFAQ.

1 Introdução

1.1. CONSIDERAÇÕES INICIAIS

O avanço da Internet tem motivado o crescimento de uma nova geração de aplicações baseadas na Web (WebApps) que combinam navegação e interatividade num grande espaço de documentos heterogêneos. A Web tem se mostrado como um dos mais efetivos e atrativos meios de divulgação, de negociação e de disponibilização de bens e de serviços (GLOVER et al., 2002; BIEBER; GALNARES; LU, 1998; BALASUBRAMANIAN; BASHIAN; PORCHER, 1997).

O projeto de WebApps ainda não é um processo totalmente definido, e existem muitos aspectos que devem ser considerados no desenvolvimento de uma aplicação com qualidade; porém, é de consenso na literatura (BRAGA, 2003; GIMENEZ; TRAVASSOS, 2002; BOSCH et al., 1999; DE PAEZ, 1999) que, a partir do reuso de modelos de análise, de projeto e de partes da implementação de software bem especificadas, desenvolvidas e testadas pode-se construir novos produtos em menor tempo e com maior confiabilidade. Tais características motivam a integração de técnicas e de ferramentas de reutilização de software no processo de desenvolvimento de WebApps.

A reutilização de software é uma das áreas da Engenharia de Software que propõe um conjunto sistemático de processos, de técnicas e de ferramentas para obter produtos com alta qualidade e que sejam economicamente viáveis. Caracteriza-se pela utilização de artefatos de software numa situação diferente daquela para qual foram originalmente construídos.

A inclusão de Tecnologias de Informação e de Comunicação (TIC) combinadas à Web propicia o desenvolvimento de WebApps em diferentes áreas do conhecimento. Por exemplo, diversos ambientes de ensino e de aprendizagem via Web para Educação Aberta e a Distância (EAD) foram propostos nos últimos anos, tais como WebCT (*Web Course Tools* — http://www.webct.com) (GOLDBERG; SALARI; SWOBODA, 1996), TelEduc (um ambiente para a criação, participação e administração de cursos na Web desenvolvido pelo Núcleo Informática Aplicada à Educação (NIED) da Universidade Estadual de Campinas (UNICAMP) — http://teleduc.nied.unicamp.br) (ROCHA, 2002b) e AulaNet (um ambiente de aprendizado cooperativo baseado na Web desenvolvido no Laboratório de Engenharia de Software do Departamento de Informática da Pontifícia Universidade

Católica do Rio de Janeiro — http://ritv.les.inf.puc-rio.br/groupware/) (LUCENA et. al., 1998).

O objetivo destes ambientes é oferecer suporte, tanto presencial como a distância, às novas formas educacionais, combinando estratégias de ensino e de aprendizagem a fim de buscar metodologias inovadoras. Estes ambientes são compostos pela junção de várias ferramentas tecnológicas de Comunicação Mediadas por Computador (CMC), tais como fóruns e listas de discussão, correio eletrônico, gerenciador de FAQ (*Frequently Asked Questions*) e sistemas de conferência por computador, aliados a outros recursos da Web.

Tais ferramentas são softwares que podem ser reutilizados em diversos outros ambientes de EAD aumentando a capacidade de suporte a cursos via Web. Portanto, tais ferramentas devem ser projetadas a fim de facilitar seu reuso em outros ambientes e também sua integração com outras ferramentas disponíveis no mercado. Este é o contexto do presente trabalho.

Diante deste panorama e das tendências tecnológicas, várias pesquisas (GILL; GROVER, 2003; LUCRÉDIO et. al. 2003; VILJAMAA, 2003; WERNER et. al., 2003) estão sendo realizadas objetivando propor métodos, técnicas e tecnologias voltadas para um reuso mais eficiente e eficaz no processo de desenvolvimento de software. Surge assim, uma oportunidade para propor e explorar técnicas e modelos de processo para especificar, projetar e implementar componentes para serem reutilizados na criação de ferramentas interativas e reutilizáveis em ambientes Web, formando assim uma biblioteca de ferramentas componentizadas. Tais ferramentas podem ser utilizadas em diversos contextos Web, principalmente, em ambientes de EAD e de Sistemas de Informação Web (SIW).

1.2. MOTIVAÇÕES E OBJETIVOS

Muitas técnicas e formas de reuso têm sido propostas nos últimos anos (CRNKOVIC et. al., 2002; HEINEMAN; COUNCIL, 2001; PRESSMAN, 2001), tais como engenharia de domínio (SOMMERVILLE, 2001), *frameworks* (FAYAD; SCHMIDT, 1997; JOHNSON, 1997; 1992), padrões (GAMMA et al.,1995; 1993), desenvolvimento baseado em componentes (CRNKOVIC et al., 2002; D'SOUZA; WILLS, 1999) e geradores de aplicação (FRANCA; STAA, 2001; SMARAGDAKIS; BATORY, 1998).

A Engenharia de Software Baseada em Componentes (ESBC) tem como objetivo propor técnicas e modelos para projetar e construir softwares complexos a partir da reutilização de componentes de software pré-construídos e testados (PRESSMAN, 2001; WANG, 2000). Nesta abordagem, os sistemas de software são vistos como um conjunto de pequenos blocos denominados componentes,

desenvolvidos sob um determinado padrão a fim prover maior reusabilidade e facilitar sua manutenção. Devem prover interfaces bem definidas por meio das quais os componentes são integrados uns aos outros, formando um sistema de software baseado em componentes (EMMERICH, 2002).

Segundo D'Souza e Wills (1999) há duas abordagens de reutilização na ESBC: (1) desenvolvimento do produto, onde softwares são desenvolvidos a partir da seleção e montagem dos artefatos reutilizáveis necessários a implementação do novo software; e (2) desenvolvimento dos artefatos que são projetados e criados para que possam ser reutilizados em diferentes contextos.

Para que a reutilização seja efetiva, a ESBC deve oferecer métodos, técnicas e ferramentas que suportem a identificação, a especificação, o projeto e a implementação de componentes numa linguagem executável. A principal idéia é que novos softwares sejam construídos rapidamente com um alto grau de confiança a partir da integração de componentes pré-existentes e com poucos componentes novos; além de tratar as mudanças de requisitos por meio da substituição de um pequeno número de componentes (CUNNINGHAM et. al., 2003).

Os métodos de Desenvolvimento Baseado em Componentes (DBC) existentes (JACOBSON, 2001; D'SOUZA; WILLS, 1999) não são simples nem diretos e os processos são complexos, dificultando sua utilização na prática. Tais características motivam a necessidade de definir uma abordagem mista para o desenvolvimento de componentes que pode ser generalizada para construir ferramentas computacionais de interatividade para Web.

Neste contexto, o objetivo do presente trabalho é propor uma abordagem para o desenvolvimento de componentes denominada ADComp (Abordagem de Desenvolvimento de Software Baseada em Componentes) que utiliza vários conceitos de métodos atuais de DBC, tecnologias para componentes e uma ferramenta CASE para apoiar as tarefas de especificar, projetar, implementar e testar os componentes.

Na ADComp é formalizado um processo para desenvolvimento de componentes que visa ser mais simples e prático, a partir da recompilação de métodos independentes de tecnologia, como o *UML Components* (CHEESMAN; DANIELS, 2001), *Catalysis* (D'SOUZA; WILLS, 1999) e *Rational Unified Process* (RUP) (JACOBSON, 2001). Oferece suporte às etapas de identificação e de definição de componentes, utiliza uma ferramenta CASE capaz de dar suporte às etapas do processo e à implementação dos componentes especificados, etapa esta que é dependente de tecnologia, onde, neste trabalho, é utilizado o modelo de componentes *Enterprise JavaBeans* (EJB) (SUN, 2003; HAEFEL, 2001).

Para validar e testar a ADComp objetiva-se desenvolver uma ferramenta Web de interatividade baseada em componentes denominada InstantFAQ — responsável

pela gerência de FAQ e de Fale Conosco em WebApps. Uma ferramenta de comunicação e de coordenação que pode ser integrada e reutilizada em diversas WebApps para auxiliar o processo de gestão e de acompanhamento de perguntas mais freqüentes de um determinado assunto (FAQ), além de auxiliar na comunicação entre usuários da WebApp e sua administração, implantando um serviço de relacionamento simples e direto (Fale Conosco).

A ferramenta CASE a ser utilizada como mecanismo no processo é a MVCASE (MVCASE, 2004; ALMEIDA et. al., 2002) elaborada pelo Departamento de Computação (DC) da Universidade Federal de São Carlos (UFSCar), para oferecer suporte as etapas de identificação, de especificação e de implementação de componentes utilizando a linguagem UML (*Unified Modeling Language*) e o modelo de componentes EJB.

Para validar e testar a ferramenta InstantFAQ e as suas formas de reuso propostas são utilizadas duas WebApps: (1) o ambiente de EAD TelEduc desenvolvido pelo NIED/UNICAMP (ROCHA, 2002a) com o objetivo de oferecer cursos pela Web, onde o professor não precisa ser um especialista em computação ou em tecnologia Web para disponibilizar e criar os materiais didáticos, e (2) o Sistema de Informação de Gestão Social (SIGS) http://www.sigs.org.br) (TURINE et al. 2004; ACOSTA; TURINE, 2003), uma parceria entre o Instituto de Estudos Especiais (IEE) da Pontifícia Universidade Católica de São Paulo (PUC/SP) e o Departamento de Computação e Estatística (DCT) da Universidade Federal Mato Grosso do Sul (UFMS), utilizado para auxiliar o processo de avaliação e de monitoramento de programas de transferência de renda.

1.3. ORGANIZAÇÃO DO TEXTO

Este texto está organizado em seis capítulos. Neste capítulo são apresentados o problema a ser investigado, as motivações e os objetivos do presente trabalho No capítulo 2 está contida uma revisão da literatura que versa sobre reutilização e ESBC, os principais métodos (*Catalysis*, *RUP* e *UML Components*), modelos e tecnologias de componentes (CCM — *CORBA Component Model*, COM+ — *Microsoft Component Object Model*+ e EJB — *Enterprise JavaBeans*) para DBC.

Para especificar e implementar a ferramenta InstantFAQ é proposta uma abordagem de desenvolvimento de componentes denominada ADComp a partir dos principais métodos descritos no capítulo 2. No capítulo 3 são descritas as etapas da ADComp e os artefatos produzidos.

As funcionalidades e a arquitetura da ferramenta InstantFAQ são apresentadas no capítulo 4. Para especificar e implementar a InstantFAQ é utilizada a abordagem proposta no capítulo 3. No capítulo 4 são apresentados os artefatos e

os componentes gerados em cada etapa do processo de desenvolvimento dos componentes, e como estes são integrados para projetar a ferramenta InstantFAQ.

Para validar e mostrar a utilização da ferramenta InstantFAQ em contextos reais, no capítulo 5 são apresentados o processo e os mecanismos de integração realizados em duas WebApps disponíveis no mercado. As conclusões e as contribuições deste trabalho e as perspectivas para a sua continuidade são apresentadas no capítulo 6.

2 Engenharia de Software Baseada em Componentes

2.1. CONSIDERAÇÕES INICIAIS

A demanda por softwares complexos, de alta qualidade e a um baixo custo tem impulsionado pesquisas por métodos e tecnologias que suportem a aplicação efetiva de técnicas de reutilização no seu processo de desenvolvimento. A ESBC é uma das áreas da engenharia de software que visa desenvolver softwares a partir de componentes reutilizáveis, bem como construí-los por meio de técnicas e métodos que visem o reuso de artefatos em domínios específicos.

A reutilização objetiva diminuir o tempo e os custos de desenvolvimento, além de melhorar a qualidade do produto por meio de artefatos previamente construídos e testados. Segundo D'Souza e Wills (1999) há duas abordagens de reutilização: (1) desenvolvimento do produto: softwares são desenvolvidos a partir da seleção e da montagem dos artefatos reutilizáveis necessários a implementação do novo software; (2) desenvolvimento de artefatos: componentes reutilizáveis são projetados e criados para que possam ser reutilizados em diferentes contextos utilizando métodos de desenvolvimento e tecnologias específicas. Neste trabalho objetiva-se desenvolver o produto denominado InstantFAQ a partir da especificação e do projeto de componentes reutilizáveis, utilizando assim as duas abordagens citadas acima.

Este capítulo está organizado da seguinte forma: na seção 2.2 são apresentadas definições sobre componentes. Na seção 2.3 são descritas as duas atividades básicas do modelo de processo da ESBC: a Engenharia de Domínio e o DBC. Na seção 2.4 são apresentados os métodos de desenvolvimento de componentes *Catalysis*, RUP e *UML Components*. Na seção 2.5 são descritas as três tecnologias de componentes utilizadas atualmente CCM, COM+ e EJB. Na seção 2.6 são apresentadas as considerações finais do capítulo.

2.2. DEFINIÇÕES

2.2.1. Componentes

Segundo Pressman (2001), a ESBC é um modelo de processo que enfatiza o projeto e a construção de softwares utilizando artefatos reutilizáveis denominados componentes. É considerada uma evolução do paradigma de orientação a objetos (WEINREICH; SAMETINGER, 2001; SZYPERSKI,1998). Objetos são descritos como entidades que encapsulam um estado e um comportamento, e possuem um

identificador único. O comportamento e a estrutura dos objetos são definidos por meio das classes, que servem para diversos propósitos: implementam um tipo de dado abstrato, provêem uma descrição abstrata do comportamento de seus objetos e provêem a implementação dos objetos.

Igualmente às classes, os componentes definem o comportamento e a criação de objetos. Objetos criados por meio de componentes são denominados instâncias de componentes. Como ilustrado na Figura 2.1, componentes e objetos tornam disponíveis suas funcionalidades por meio de interfaces. Diferentemente das classes, um componente pode ser implementado internamente por uma única classe, por múltiplas classes, ou até mesmo por procedimentos tradicionais numa linguagem de programação não orientada a objetos. A distinção mais importante é que os componentes estão em conformidade com um padrão definido por um modelo de componente, que define um conjunto de regras que devem ser seguidas pelo componente para tornar as implementações de objeto acessíveis a outras implementações de outros objetos. Segundo Brown (2000), o componente pode ser visto como uma maneira conveniente de empacotar os diversos objetos, com suas implementações e suas interfaces, mantendo-as disponíveis para serem “montadas” num software mais complexo.

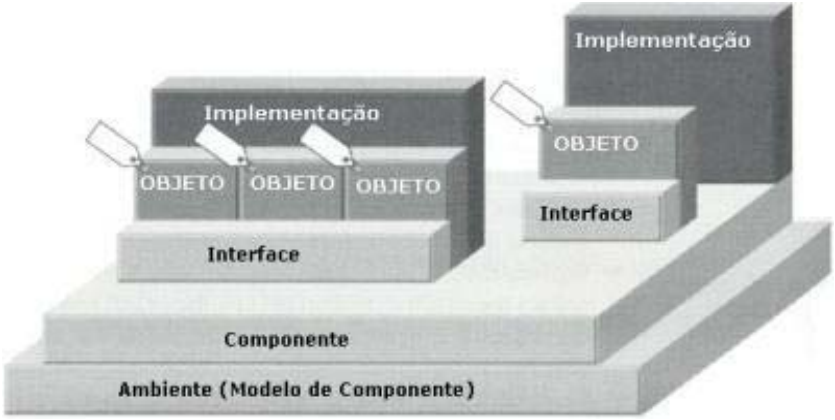


FIGURA 2.1: RELAÇÃO ENTRE OBJETOS E COMPONENTES (BROWN, 2000, p. 78).

Na literatura (BACHMANN, 2000; WANG, 2000; YACOUB; AMMAR; MILI, 1999a; YACOUB; AMMAR; MILI, 1999b; SZYPERSKI, 1998; SAMETINGER, 1997; LEEB, 1996), um componente de software pode ser compreendido pelas seguintes características:

1. Descrever ou realizar uma função específica;
2. Estar em conformidade e prover um conjunto de interfaces bem definidas;
3. Prover um grau de maturidade do componente quantificando sua reutilização em outros projetos;
4. Ser autocontido em termos da funcionalidade que provê;
5. Ser reutilizável, permitindo que novos softwares sejam desenvolvidos rapidamente a partir de componentes existentes e testados, reduzindo não somente o tempo de desenvolvimento, como também melhorando a qualidade do produto; e

6. Ser flexível e extensível, permitindo que novas funcionalidades possam ser incrementadas e integradas facilmente, habilitando soluções adaptáveis e customizáveis, além de reduzir significantemente os custos, e ao mesmo tempo, alcançando um maior nível de satisfação do usuário.

Segundo Emmerich (2002), um componente pode ser entendido como uma entidade autocontida (*blackbox*) que exporta e importa funcionalidades para seu ambiente por meio de interfaces bem definidas, que definem um conjunto de assinaturas de operações com semântica bem especificada. O papel da interface é ocultar a implementação do componente (COUNCIL; HEINEMAN, 2001; BACHMANN, 2000; SZYPERSKI, 1998).

De acordo com Williams (2001), os componentes de software podem ser classificados nas três categorias abaixo:

1. Componentes GUI (*Graphical User Interface*): são os tipos mais comuns, compreendem botões, calendários, calculadoras, menus, isto é, todos os tipos de componentes com os quais o usuário pode interagir visualmente;
2. Componentes de Serviço: compreendem serviços comuns necessários por aplicações, como acesso a bancos de dados, controle de transações, serviços de mensagens e de integração de softwares; e
3. Componentes do Domínio ou de Negócio: representam abstrações de um domínio do problema. As interações necessárias entre os componentes são freqüentemente restritas pelas regras de negócio que são específicas de um determinado domínio.

2.2.2. Modelo de Componentes

Segundo Councill e Heineman (2001), um componente de software é um elemento de software que obedece a um modelo de componente, que define padrões específicos de interação[1] e de composição[2]. Um modelo de componente opera em dois níveis: (1) define como construir um componente individual e (2) define a comunicação e interação entre os componentes num software baseado em componentes.

Como os componentes podem ser desenvolvidos independentemente e podem ser temas para composição de uma terceira parte (SZYPERSKI, 1998), são requeridos modelos de componentes que definam padrões de comunicação, de troca de dados e de interoperabilidade, por exemplo, entre componentes de diferentes fabricantes. Outros elementos básicos de um modelo de componente são padrões para interfaces, nomeação, metadados, customização, composição,

[1] Uma ação entre dois ou mais elementos de software (COUNCILL; HEINEMAN, 2001).

[2] A combinação entre dois ou mais componentes de software produzindo um novo comportamento para o componente num nível diferente de abstração. As características do novo comportamento são determinadas pelos componentes que estão sendo combinados e pela forma em que são combinados (COUNCILL; HEINEMAN, 2001).

evolução e *deployment*[3], como ilustrado na Tabela 2.1.

Padrões para	Descrição
Interfaces	Especificação de comportamento e propriedades dos componentes; definição das IDLs (*Interface Description Languages*).
Nomeação	Nomes globais únicos para interfaces e componentes.
Metadados	Informação sobre componentes, interfaces e seus relacionamentos; APIs (*Application Protocol Interface*) para prover tais informações.
Interoperabilidade	Comunicação e troca de dados entre componentes de diferentes vendedores, implementados em diferentes linguagens e plataformas.
Customização	Interfaces para customização de componentes. Ferramentas de customização amigáveis que utilizarão estas interfaces.
Composição	Interfaces e regras para combinar componentes na criação de estruturas mais complexas e para substituição e adição de componentes em estruturas existentes.
Suporte à Evolução	Regras e serviços para atualização de componentes ou interfaces por versões mais novas.
Empacotamento e *Deployment*	Implementação do empacotamento e recursos necessários para instalar e configurar um componente.

TABELA 2.1: ELEMENTOS BÁSICOS DE UM MODELO DE COMPONENTE (WEINREICH; SAMETINGER, 2001, p. 38).

Há vários modelos de componente atualmente disponíveis na literatura, mas os principais são: (1) CCM do *Object Management Group* (OMG, 1992); (2) COM+ da Microsoft (EWALD, 2001); e (3) EJB da *SUN Microsystems* (SUN, 2003; HAEFEL, 2001; THOMAS, 1998). Tais modelos são descritos na seção 2.5.

2.3. PROCESSO DA ESBC

Segundo Krueger (1992), como todo processo de Engenharia de Software, a ESBC deve prover uma disciplina sistemática no desenvolvimento de software, devendo ser cuidadosamente planejada e controlada para ser efetiva, reduzindo assim a distância cognitiva entre a modelagem conceitual do software e sua implementação final. Assim, é necessária uma abordagem que enfatize a reutilização de modelos nas etapas iniciais do desenvolvimento para garantir a reutilização de componentes na implementação do software.

Segundo Pressman (2001), um modelo de processo para a ESBC possui duas atividades principais: a Engenharia de Domínio (ED) e o Desenvolvimento Baseado em Componentes (DBC). Conforme ilustrado na Figura 2.2, o modelo enfatiza trilhas paralelas, onde a ED define a execução de atividades concorrentemente ao DBC. A ED executa o trabalho requerido para estabelecer um conjunto de componentes que

[3] Neste trabalho é utilizado o termo *deployment* como sinônimo de implantação.

possam ser reutilizados pelo engenheiro de software durante o DBC.

2.3.1. Engenharia de Domínio

A ED é o processo de identificar e organizar o conhecimento sobre um domínio de problema. Tem como objetivos identificar, construir, catalogar e disseminar um conjunto de componentes que tenham aplicabilidade em softwares existentes e novos (PRIETO-D'IAZ; ARANGO, 1991).

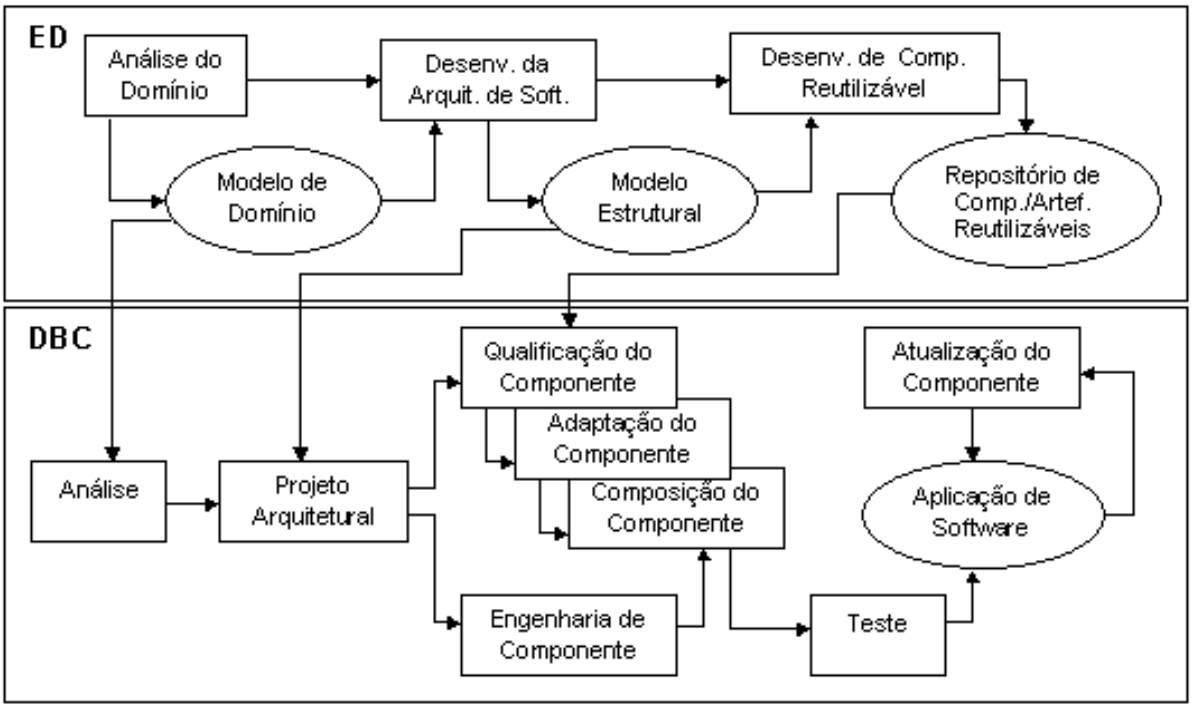


FIGURA 2.2: MODELO DE PROCESSO DA ESBC (PRESSMAN, 2001, p. 725).

As três principais atividades da ED, ilustradas na Figura 2.2, são (WERNER; BRAGA, 2000):

1. Análise do Domínio: nesta atividade as informações de um determinado domínio utilizadas no desenvolvimento do produto são definidas, capturadas e organizadas com o propósito de serem reutilizadas na criação de novos softwares e formalizadas por meio de um modelo de domínio;
2. Projeto do Domínio ou Desenvolvimento da Arquitetura de Software: nesta atividade são detalhados os conceitos e as características relevantes no domínio, construindo a arquitetura do software que atenda aos requisitos identificados no modelo de domínio. As oportunidades de reutilização, identificadas na análise do domínio, são refinadas para especificar as restrições do projeto; e
3. Implementação do Domínio ou Desenvolvimento de Componente Reutilizável: nesta atividade os componentes que compõem a arquitetura de software são implementados, enfatizando a definição de suas interfaces e armazenando-os num repositório de componentes e artefatos reutilizáveis.

2.3.2. Desenvolvimento Baseado em Componentes

A aquisição, a adaptação e a integração de componentes de software reutilizáveis para desenvolver rapidamente softwares complexos com pouco esforço de engenharia e a um baixo custo é o foco desta atividade.

Na Figura 2.2 são ilustradas as etapas da atividade de DBC, que ocorrem paralelamente à da ED, onde métodos de análise e de projeto arquitetural são utilizados para refinar um estilo arquitetural apropriado para o modelo de análise criado na ED. A arquitetura do software definida deve ser preenchida com componentes disponíveis em bibliotecas de reuso ou criados para atender às necessidades requeridas. Desta forma, o fluxo de tarefas de DBC apresenta duas abordagens: (1) reutilizar componente, quando este estiver disponível para uma integração, devendo ser qualificado e adaptado; e (2) criar um novo componente. Nas duas abordagens, o componente resultante deve ser testado e integrado num modelo arquitetural baseado em componentes.

Segundo Pressman (2001), quando se propõe reutilizar componentes, sua existência não garante uma integração arquitetural fácil e eficiente, pois alguns destes componentes reutilizáveis são desenvolvidos sem a utilização de técnicas (de forma “caseira”) ou extraídos de aplicações existentes, e até mesmo adquiridos de terceiros. Desta forma, sugere-se aplicar a seguinte seqüência de atividades:

1. Qualificar: utilizada para garantir que componentes candidatos executem a função requerida, ajustando-se corretamente ao estilo arquitetural e exibindo as características de qualidade requeridas pelo software;
2. Adaptar: utilizada quando componentes qualificados apresentam conflitos às regras do projeto arquitetural devendo ser adaptados para corresponder às necessidades requeridas no projeto ou, conforme o esforço requerido descartados e substituídos por outros mais apropriados; e
3. Compor: utilizada para integrar os componentes qualificados, adaptados e construídos numa arquitetura estabelecida.

2.4. MÉTODOS DE DESENVOLVIMENTO

Como as pesquisas na ESBC são recentes há carência de métodos, técnicas e ferramentas que suportem tanto o desenvolvimento de componentes quanto a sua reutilização em softwares de um determinado domínio. Para que o DBC seja efetivo, é necessário prover uma disciplina sistemática, cuidadosamente planejada e controlada, de forma que a reutilização ocorra em todas as etapas do desenvolvimento, não envolvendo apenas código (WERNER; BRAGA, 2000).

Na literatura, muitos métodos específicos para DBC são extensões de métodos existentes, em particular, métodos voltados para o desenvolvimento orientado a objetos. Dentre estes métodos destacam-se o *Catalysis* (D’SOUZA; WILLS, 1999), o

RUP (JACOBSON, 2001) e o *UML Components* (CHEESMAN; DANIELS, 2001), que são descritos a seguir.

2.4.1. Catalysis

O *Catalysis* é um processo de desenvolvimento de software orientado a objetos baseado em componentes. Segundo D'Souza e Wills (1999), pode ser considerado um dos melhores métodos para o desenvolvimento de software baseado na tecnologia de componentes, em que todos os modelos são descritos detalhadamente, enfatizando a especificação da arquitetura da aplicação.

O *Catalysis* é baseado em três conceitos de modelagem — colaboração, tipo e refinamento:

1. Colaboração: permite modelar as interações e os comportamentos entre grupo de objetos;
2. Tipo: define o comportamento externo e visível de um objeto, abstraindo os detalhes de sua representação interna; e
3. Refinamento: permite relacionar descrições de um mesmo objeto em diferentes níveis de detalhes.

O *Catalysis* é baseado em três níveis de modelagem: o domínio do problema, a especificação do software ou componente e o projeto interno do componente ou software. O domínio do problema ou de negócio abrange todos os conceitos relevantes no qual softwares são desenvolvidos. Estabelece as terminologias do domínio do problema, os processos do negócio, os papéis e as colaborações. Na especificação de componentes são descritos o comportamento externo requerido de um componente, seu escopo e suas responsabilidades, além de definir suas interfaces e especificar as operações necessárias para o componente. O projeto interno de um componente descreve sua construção a partir de pequenas partes que interagem e provêem os comportamentos requeridos, isto é, implementa as especificações dos componentes (arquitetura e colaborações).

2.4.2. RUP: *Rational Unified Process*

O RUP é um processo de desenvolvimento de software da *Rational Software Corporation* composto por um conjunto de atividades necessárias para transformar requisitos de usuário num conjunto consistente de artefatos que representam o software (JACOBSON, 2001). O método é iterativo, orientado a objetos, controlado e suportado por ferramentas, podendo ser aplicado em todo tipo de projeto de desenvolvimento de software. Além disto, RUP é um *framework* de processo genérico, capaz de ser especializado para diferentes áreas de aplicação, tipos de organização, níveis de competência e tamanhos de projeto.

O processo do RUP é divido em quatro fases: (1) concepção, (2) elaboração, (3) construção e (4) transição, representando o ciclo de vida de um software.

Durante a fase de concepção são definidos os casos de negócio para o software e os limites do escopo do projeto. Na fase de elaboração são realizadas a análise do domínio do problema, a definição arquitetural e um plano de projeto, além de eliminar elementos de risco para o projeto. Na fase de construção são desenvolvidos e integrados todos os componentes e características da aplicação dentro de um produto, resultando numa primeira versão do software. Na fase de transição, o software desenvolvido é testado e entregue ao usuário final, sendo que ajustes e alterações são realizados para corrigir algum problema encontrado pelo usuário ou desenvolver alguma outra característica necessária.

Cada uma destas fases é dividida em iterações, que correspondem a um ciclo de desenvolvimento completo e geralmente resulta numa versão executável do software. O RUP utiliza na modelagem UML (OMG, 2003) e introduz novos conceitos no processo, tais como (RUP, 2003):

- Trabalhadores: são perfis que correspondem à competências para realização de atividades;
- Atividades: são tarefas que podem ser entregues a trabalhadores individuais;
- Artefatos: são entradas e saídas das atividades; e
- W*orkflows*: são agrupamentos de atividades relacionadas, agrupamentos genéricos ou especializados por fases.

Estes conceitos capturam abstrações, como “quem” deve fazer “o quê”, “como” e “quando” (BOERTIN; STEEN; JONKERS, 2001). O principal conceito do RUP é a definição de *workflows*, pois a mera enumeração de trabalhadores, de atividades e de artefatos não constitui completamente o processo. Os *workflows* descrevem uma seqüência significativa de atividades que produzem um resultado e mostram as interações entre os trabalhadores.

2.4.3. *UML Components*

O método *UML Components* utiliza a UML para modelar todas as fases do processo de DBC. Este processo baseia-se em quatro *workflows*: requisitos, especificação, provisão e montagem. Todos os *workflows* geram um número de modelos de artefatos que servem de entrada para os demais *workflows*, bem como, também podem ser reutilizados em outros projetos de softwares de mesmo domínio (CHEESMAN; DANIELS, 2001).

O *workflow* requisitos é responsável pelo levantamento dos requisitos do domínio do problema, definindo “o quê” os componentes devem atender para solucionar o problema. A partir da definição dos requisitos, o *workflow* especificação tem como objetivo especificar os componentes. Tal atividade é o foco deste método e é dividida em três partes: (1) identificação do componente, (2) interação do componente, e (3) especificação do componente. A primeira fase produz uma

arquitetura inicial do componente a partir dos requisitos. A segunda fase recupera e descreve as operações necessárias e aloca responsabilidades. Na terceira fase são definidas as especificações precisas das operações, das interfaces e dos componentes. As fases são iterativas e o desenvolvimento dos artefatos é incremental com adições e modificações em todas as fases.

Os *workflows* provisão e montagem têm como objetivos, respectivamente, implementar os componentes especificados num determinado modelo de componentes ou procurar componentes prontos e adequados às especificações do projeto; e desenvolver softwares a partir da integração de componentes e softwares existentes, projetando uma interface de usuário baseada nos modelos de casos de uso.

O *UML Components* enfatiza o *workflow* especificação, pois, segundo Cheesman e Daniels (2001), o principal desafio de um método baseado em componentes é tratar as futuras mudanças que podem ser realizadas no componente, e tal tarefa depende da correta especificação dos componentes.

2.5. MODELOS DE COMPONENTES

Devido ao impacto potencial do reuso e da ESBC no processo de desenvolvimento de software, vários modelos e tecnologias que definem padrões de comunicação para dar suporte ao DBC foram propostos. As três tecnologias mais consolidadas são: (1) CCM: extensão do modelo de objetos CORBA (FRAMINGHAM, 1998; OMG, 1992); (2) COM+: evolução do padrão COM desenvolvido pela Microsoft para a construção de softwares distribuídos e baseados em componentes (EWALD, 2001); e (3) EJB: extensão do padrão de componentes *JavaBeans* desenvolvido pela *Sun Microsystems* que é utilizada neste trabalho.

2.5.1. CCM: *CORBA Component Model*

A tecnologia CORBA (*Common Object Request Broker Architecture*), proposta pelo OMG (*Object Management Group*) (OMG, 1992), define uma especificação de *middleware* para o projeto de aplicações distribuídas, não restringindo os recursos de localização de objetos, linguagem de programação, protocolos de comunicação ou hardware.

Para prover um maior nível de desenvolvimento para componentes reutilizáveis, o OMG especificou um conjunto de serviços de objeto CORBA que define interfaces padrão para acesso a serviços de distribuição. Apesar disto, o modelo CORBA tradicional definido em CORBA 2.4 (OMG, 2000) apresenta certas limitações, tais como, falta de um padrão para implantar as implementações de objetos, extensão limitada das funcionalidades de objetos e nenhum padrão para gerenciamento do ciclo de vida do objeto.

Para suprir tais limitações, o OMG adotou o CCM para estender e incluir o modelo de objetos CORBA. CCM define características e serviços que permitem aos desenvolvedores implementar, gerenciar, configurar e implantar componentes que integram serviços CORBA, tais como transação, segurança, estado persistente e notificação de eventos. Além disto, o padrão CCM permite maior reuso de software e provê maior flexibilidade para configuração de aplicações CORBA (WANG; SCHIMIDT; O'RYAN, 2001).

Componentes CCM são blocos de construção básicos dentro de um software CCM. Os desenvolvedores utilizam CCM para definir interfaces IDL (*Interface Definition Language*) que são suportadas pelas implementações do componente. Os componentes são implementados por ferramentas que suportam provedores CCM. As implementações de componente resultantes podem ser empacotadas num arquivo de montagem ou em DLL (*Dynamically Linked Library*). O CCM pode ser classificado com um novo modelo de programação apropriado para impulsionar tecnologias e serviços existentes na próxima geração de aplicações distribuídas e de alta escalabilidade.

2.5.2. COM+: *Common Object Model*

O modelo de componente COM (*Component Object Model*) foi desenvolvido pela Microsoft. COM+ é um *framework* que provê serviços avançados em tempo de execução (*run-time*) para as instâncias das classes COM tornando mais fácil construir sistemas distribuídos.

Segundo Rosenberry, Kenney e Fisher (1992), um componente é uma coleção de classes, sendo que cada classe é unicamente identificada por meio de um identificador de classe (CLSID). Cada classe COM pode implementar várias interfaces COM, que oferece um conjunto de operações e são unicamente identificadas por meio do identificador de interface (IID). O modelo COM diferencia os conceitos de componente e objeto, enquanto um componente é uma coleção de classes, um objeto COM é uma instância de uma classe COM (EWALD, 2001).

O *framework* COM+ é baseado em dois conceitos: contexto e casualidade. Um contexto é um objeto dentro de um processo no sistema operacional que provê um conjunto específico de serviços *run-time*. Cada objeto criado no processo COM+ pertence exatamente a um contexto que é configurado para encontrar as necessidades dos objetos. Este mecanismo genérico é utilizado por COM+ para implementar serviços transacionais e de segurança.

Alguns serviços precisam saber se múltiplas chamadas de métodos ao longo dos limites dos contextos estão casualmente relacionadas, isto é, se uma chamada de método implica uma outra chamada de método, criando as operações em cascata. Dentro de um único processo utilizando múltiplas *threads*, o relacionamento casual entre chamadas de métodos pode ser determinado por um identificador único

de *thread*. Entretanto, chamadas de métodos COM+ podem ser disparadas por diferentes *threads*, às vezes em diferentes processos. COM+ detecta os relacionamentos entre as chamadas de métodos utilizando uma abstração similar a uma *thread*, mas não limita a um processo particular. Esta abstração é denominada casualidade — uma corrente distribuída de chamadas de métodos COM+ que transporta qualquer quantidade de contextos em qualquer número de processos, possivelmente envolvendo múltiplas máquinas.

2.5.3. EJB: *Enterprise JavaBeans*

O modelo de componentes *JavaBeans* faz parte da plataforma *Sun Microsystems* denominada *Enterprise Java*. Oferece suporte ao processo de desenvolvimento de componentes reutilizáveis e tem como característica ser um modelo portável, pois utiliza a linguagem Java (GOSLING et. al., 1996) que é independente de plataforma. EJB é uma extensão de *JavaBeans* — um ambiente Java robusto para o desenvolvimento de softwares em camadas com rigorosos requisitos de escalabilidade, de distribuição e de disponibilidade (THOMAS, 1998).

Conforme ilustrado na Figura 2.3, o modelo EJB oferece um ambiente de execução de softwares baseado em componentes denominado *EJB server*, que gerencia e coordena a alocação de recursos. Este ambiente possui o *EJB container* e o *Web container*. O *EJB container* é responsável por encapsular os componentes EJB denominados *enterprise beans* (*bean EJB*) — componentes de software focados na lógica do problema ou do negócio do software, por isto, chamados componentes de negócio. O *Web container* é responsável por gerenciar arquivos Web, como páginas JSP (*JavaServer Pages*) e HTML, servlets Java, figuras e outros itens utilizados no cliente.

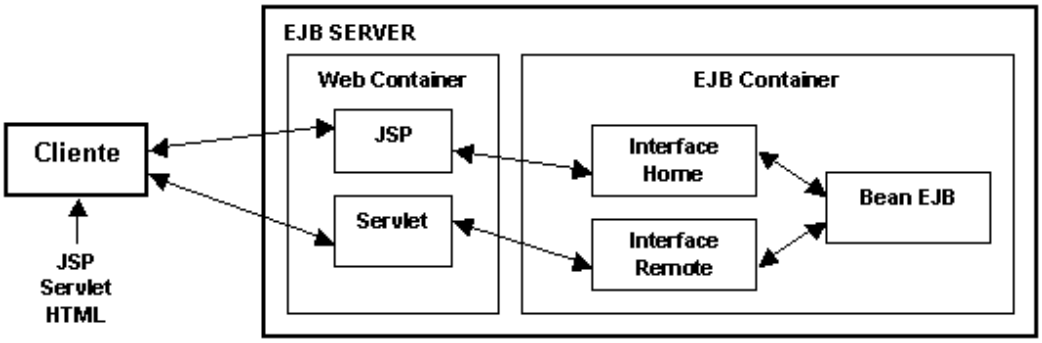


FIGURA 2.3: SERVIDOR ENTERPRISE JAVABEANS.

O *container* oferece um contexto de aplicação para um ou mais componentes, provendo serviços de gerenciamento e de controle para a execução dos componentes, tais como gerenciamento de transações, segurança, comunicação remota, gerenciamento do ciclo de vida e de conexões a banco de dados. Como os *beans* são encapsulados e gerenciados pelo *container*, não é possível acessá-los diretamente; a única forma é criar instâncias de um *bean* por meio de duas interfaces denominadas *home* e *remote*.

A interface *home* é responsável por prover serviços de criação e de remoção de instâncias do componente, cujas implementações são realizadas na classe do *bean*. Também é responsável por serviços de consulta e de manipulação das instâncias do *bean*. Estes últimos são chamados métodos de negócio da interface *home*. Os métodos de consulta, denominados *finders*, são responsáveis pela localização de uma ou mais instâncias. Toda interface *home* deve definir ao menos o método `findByPrimaryKey()`, que retorna uma instância cujo parâmetro é a chave primária.

A interface remota define a visão do cliente para uma instância do *bean*, concentrando métodos de negócio aplicáveis a instâncias individuais, sendo que cada método definido na interface remota deve possuir uma implementação na classe do *bean*.

Como ilustrado na Figura 2.4, a tecnologia EJB oferece suporte a objetos transientes e persistentes, isto é, os componentes são classificados em *beans* transientes, (denominados *session bean*) e em *beans* persistentes (denominados *entity bean*).

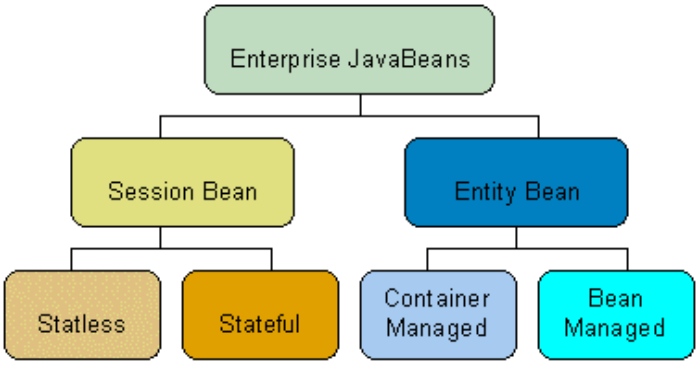


FIGURA 2.4: TIPOS DE COMPONENTES EJB.

Um *session bean* representa uma transação passageira com o cliente. Se o servidor ou o cliente “quebra/cai”, o *session bean* e seus dados são perdidos. Ao contrário, *entity beans* são persistentes e representam o dado no banco de dados. Se o servidor ou o cliente “quebra/cai”, os serviços de suporte garantem que o dado do *entity bean* seja salvo.

Session beans podem ser *stateless* ou *stateful*. Um *stateless session bean* basicamente se resume ao intervalo de execução de um serviço invocado pelo cliente. Por esta razão, *stateless session beans* são facilmente manipuláveis pelo *EJB container*. Um *stateful session bean* mantém um estado ao longo de toda a transação com um cliente. Seu tempo de vida é tipicamente o tempo de vida de seu cliente (SUN, 2002b).

Um *entity bean* é uma representação persistente em que os dados são mantidos num banco de dados e uma chave primária identifica cada instância do mesmo; podem ser criados pela inserção de dados diretamente no banco de dados ou pela criação de um objeto.

Entity beans podem gerenciar sua própria persistência — B*ean-Managed Persistence* (BMP) — ou podem delegar os serviços de persistência para o *container* — *Container-Managed Persistence* (CMP). Se o *bean* delega a persistência para o *container*, este executa automaticamente todas as operações de recuperação e de armazenamento de dados no lado do *bean*. Entretanto, se o *bean* gerencia sua própria persistência, o desenvolvedor de *beans* deve implementar as operações de persistência (JDBC ou chamadas SQL) diretamente nos métodos da classe do *bean*.

A opção de modelagem com CMP simplifica significativamente o processo de desenvolvimento de *entity beans*, pois o desenvolvedor não necessita codificar o acesso à camada de persistência. Esta é a razão pela qual este modelo é incentivado, em detrimento de BMP, aconselhado apenas no caso em que o desenvolvedor necessita de maior controle sobre a persistência.

2.6. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os conceitos e as características da ESBC utilizadas neste trabalho, cujas duas atividades principais são (1) ED e (2) DBC. A atividade de DBC apresenta duas abordagens de desenvolvimento, uma para desenvolvimento de software com componentes; e outra para o desenvolvimento de componentes que poderão ser reutilizados na criação de novos softwares.

Para dar suporte à ESBC, métodos e modelos de componentes permitem que componentes sejam especificados e implementados. Os métodos de DBC abordados foram *Catalysis*, RUP e *UML Components* que compõem a base para a abordagem proposta neste trabalho denominada ADComp. Os principais modelos de componentes da literatura CCM, COM+ e EJB foram descritos neste capítulo, enfatizando o último, pois é o modelo adotado no desenvolvimento da ferramenta InstantFAQ — especificada e validada segundo a abordagem ADComp.

A partir da revisão destes conceitos, no próximo capítulo é apresentada a proposta de uma abordagem de desenvolvimento de software baseada em componentes que integra diversos mecanismos — método *UML Components*, modelo de componentes EJB e ferramenta CASE MVCASE — a fim de auxiliar o engenheiro de software tanto na especificação e na implementação de componentes quanto na construção de novos softwares que reutilizam tais componentes.

ADComp: Abordagem de Desenvolvimento de Software Baseada em Componentes 3

3.1. CONSIDERAÇÕES INICIAIS

A Abordagem de Desenvolvimento de Software Baseada em Componentes denominada ADComp proposta neste trabalho é baseada em três etapas, na primeira são descritos os aspectos conceituais do domínio do problema; na segunda são definidas as especificações e a arquitetura dos componentes, e na última etapa é realizada a implementação dos componentes especificados de acordo com a tecnologia de componentes adotada. Para facilitar o entendimento da abordagem, é utilizado um estudo de caso simplificado do software de reserva de quartos de hotel, proposto por Cheesman e Daniels (2001).

Este capítulo está organizado da seguinte forma: na seção 3.2 é apresentada uma visão geral da abordagem proposta com suas respectivas etapas: definição de requisitos, definição de componentes e implementação de componentes. Na seção 3.3 são descritos a etapa definição de requisitos e os dois artefatos produzidos: modelo de casos de uso e modelo conceitual de negócio. Na seção 3.4 é apresentada a etapa definição de componentes, que é dividida em três fases (identificação, interação e especificação do componente). A etapa implementação de componentes é apresentada na seção 3.5. As considerações finais são apresentadas na seção 3.6.

3.2. VISÃO GERAL

A proposta da ADComp é integrar técnicas de modelagem e de especificação de componentes com técnicas de implementação, seguindo os padrões de um modelo de componentes. O objetivo é propor uma abordagem integrada e mista para especificar, desenvolver e documentar componentes a partir dos quais novos softwares num determinado domínio podem ser construídos, visando o princípio do reuso.

A abordagem incentiva a geração e a implementação de componentes numa linguagem executável sobre uma determinada tecnologia, resultando em dois produtos — os componentes e um software baseado em componentes. A geração do novo software dar-se-á com o auxílio da ferramenta CASE MVCASE (MVCASE, 2004; ALMEIDA et. al., 2002), que tem um papel fundamental em todas as etapas do processo, pois é o meio para especificar os artefatos e os modelos de cada uma das etapas da abordagem, incluindo a construção, a geração de código e o *deployment* do novo software.

As técnicas de modelagem e de especificação de componentes utilizadas na ADComp são baseadas nos métodos *UML Components* (CHEESMAN; DANIELS, 2001), *Catalysis* (D'SOUZA; WILLS, 1999) e RUP (JACOBSON, 2001), sendo que o primeiro método é mais utilizado, pois apresenta uma linguagem mais simples e direcionada ao desenvolvedor que deseja partir da técnica para a prática, além de ser referenciados pelos métodos *Catalysis* e RUP.

Quanto às tecnologias de implementação, a ADComp compreende o modelo de componentes EJB baseado na linguagem Java, que é utilizado pela ferramenta MVCASE na construção de componentes. Resumidamente, a ADComp deve oferecer suporte às etapas de identificação e de definição de componentes, bem como a implementação dos mesmos utilizando a linguagem Java (GOSLING et. al., 1996) e o modelo de componentes EJB (HAEFEL, 2003; SUN, 2003).

Utilizando a notação formal da técnica SADT (*Structured Analysis and Design Tecnhique*) (ROSS, 1985), conforme ilustrado na Figura 3.1, o processo simplificado de desenvolvimento de componentes da ADComp é constituído por três etapas denominadas (1) definição de requisitos, (2) definição de componentes e (3) implementação de componentes. Todos os modelos e artefatos propostos pela ADComp são baseados na notação da UML (OMG, 2003), e são elaborados por meio da ferramenta MVCASE.

A ADComp é iterativa e incremental, sendo que em cada uma das etapas do processo, artefatos e componentes produzidos na última iteração são melhorados e estendidos. Na Figura 3.2 são apresentados os artefatos gerados em cada uma das três etapas. Como ilustrado nas Figuras 3.1 e 3.2, genericamente, em cada etapa são produzidos diversos modelos e artefatos denominados especificações (do domínio do problema ou do componente/arquitetura), que são refinadas e transformadas em cada etapa no processo. Isto é feito para simplificar o processo de definição de componentes e ao mesmo tempo para guiar sua implementação de acordo com a tecnologia adotada.

Nas Figuras 3.1 e 3.2 observa-se que em todas as etapas, o engenheiro utiliza a MVCASE para auxiliar o desenvolvimento das especificações e dos artefatos, e na implementação dos componentes. Como dados de controle, nas duas primeiras etapas são utilizadas, principalmente, as técnicas do *UML Components*. Na primeira etapa, os dados de entrada utilizados são os requisitos do negócio que ajudam a definir "o quê" os componentes devem atender para solucionar o problema. Como saída, são gerados dois artefatos (modelo conceitual de negócio e modelo de casos de uso) que são utilizados como entrada para a segunda etapa.

Na segunda etapa são definidas as especificações e a arquitetura dos componentes de acordo com os requisitos e os artefatos da etapa anterior. Além das técnicas do *UML Components*, outros dados de entrada utilizados são informações sobre recursos de softwares existentes (sistemas legados, pacotes e bancos de dados) e restrições técnicas, como uso de uma arquitetura particular ou ferramenta.

As três fases (Figura 3.2) desta etapa são (1) identificação do componente, (2) interação do componente e (3) especificação do componente, cujo resultado é um conjunto de modelos e artefatos correspondentes às especificações das interfaces que os componentes suportam ou dependem, e à arquitetura dos componentes, que define as interações entre os mesmos.

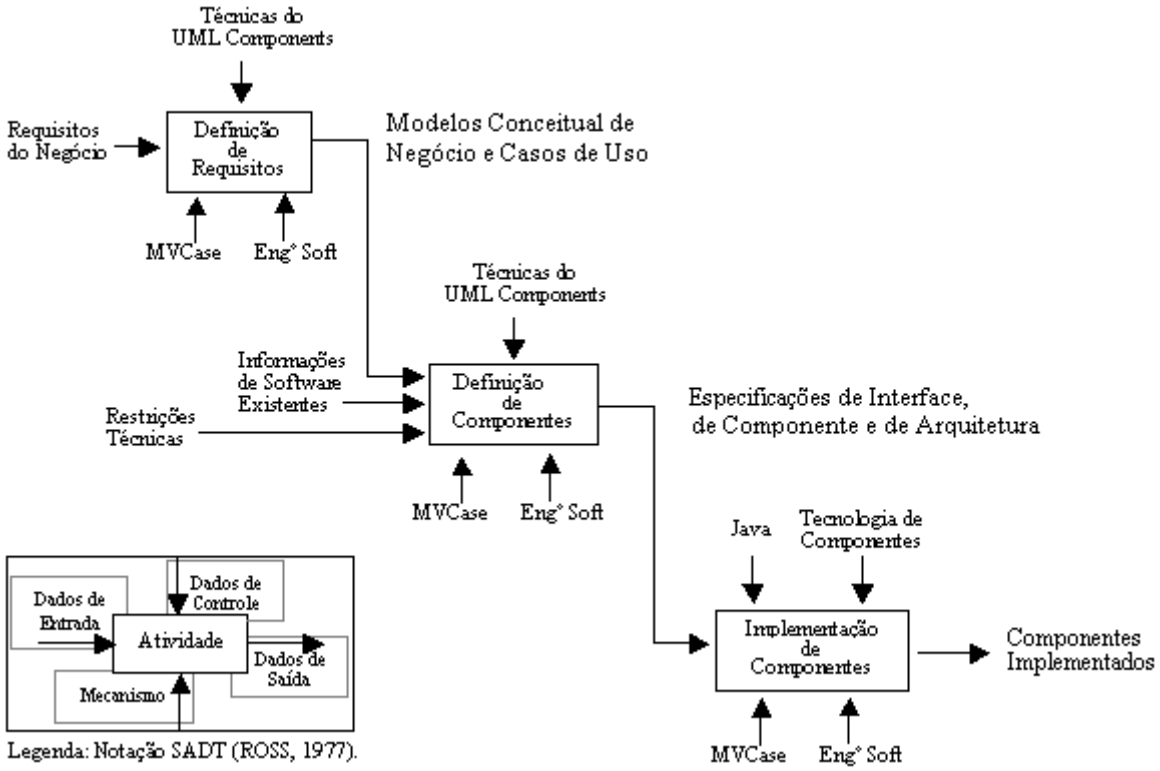


FIGURA 3.1: PROCESSO SIMPLIFICADO DA ADCOMP.

A terceira etapa é direcionada para a implementação de componentes utilizando uma tecnologia que oferece suporte à linguagem Java, como CORBA ou EJB. Nesta etapa, o engenheiro utilizando a MVCASE implementa os componentes especificados na etapa anterior, gerando código na linguagem Java seguindo a tecnologia de componentes adotada e modelada pelo engenheiro. Os principais artefatos gerados são os componentes implementados, que formam um repositório de componentes reutilizados no desenvolvimento de novos softwares.

3.3. ETAPA 1: DEFINIÇÃO DE REQUISITOS

Nesta primeira etapa os requisitos de negócio são especificados definindo “o quê” os componentes devem atender para solucionar o problema. Técnicas de elicitação e de especificação de requisitos devem ser utilizadas de acordo com a preferência do engenheiro de software. Tais requisitos são utilizados na elaboração do Modelo Conceitual de Negócio (MCN) e do Modelo de Casos de Uso (MCU).

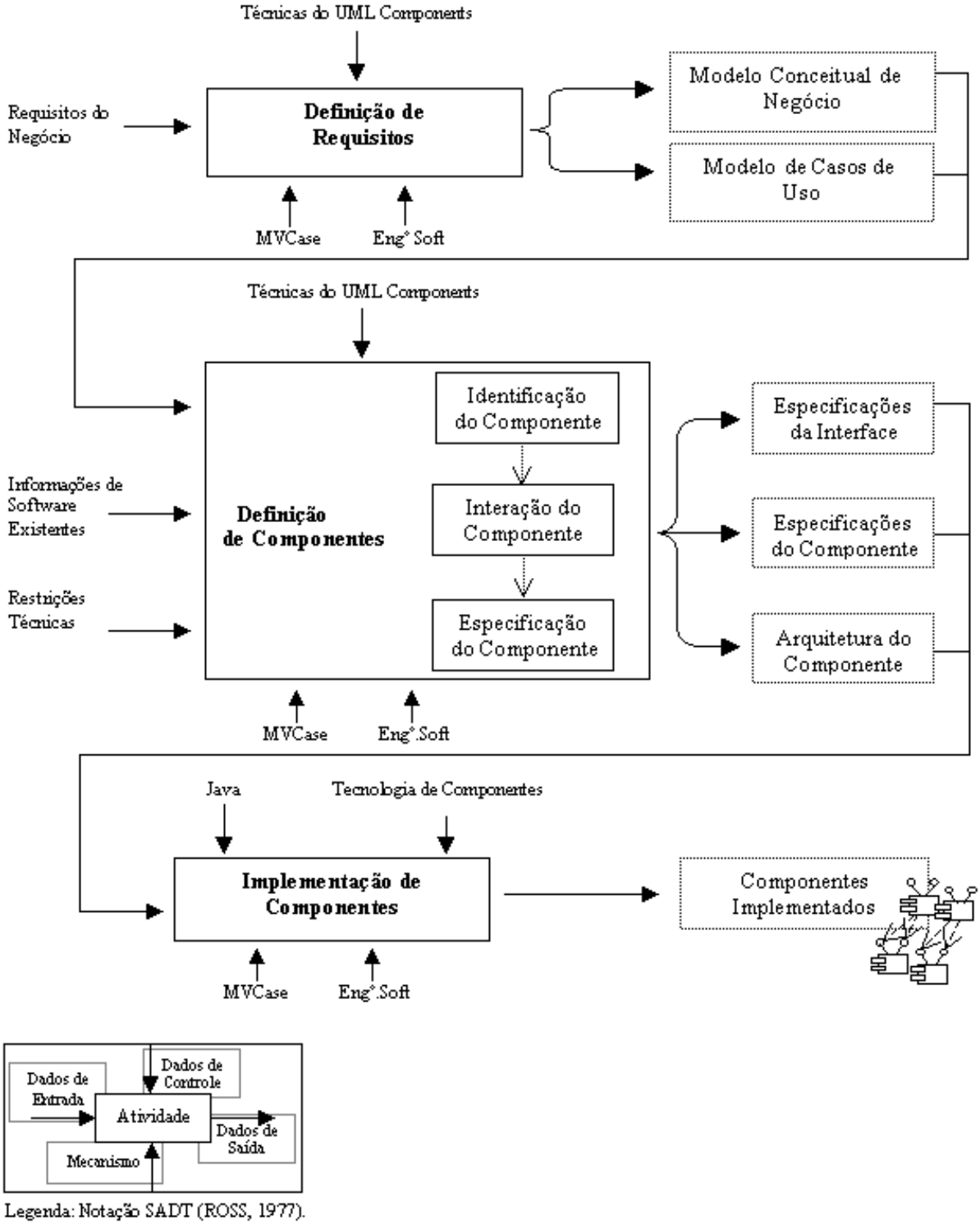


FIGURA 3.2: ARTEFATOS DA ADCOMP.

Para ilustrar o uso da ADComp é apresentado o estudo de caso de reserva de quartos de hotel (software de reservas), que tem como objetivo principal permitir a reserva de quartos em qualquer hotel de uma rede de hotéis. O software deve oferecer quartos de hotéis alternativos, caso o hotel desejado esteja lotado. Reservas podem ser feitas por telefone, pessoalmente ou pela Internet. Cada hotel tem um administrador responsável pelo controle das reservas, mas qualquer usuário autorizado pode fazer uma reserva. O tempo de efetivação da reserva, via telefone ou pessoalmente, é de três minutos; para melhorar o tempo do processo, detalhes

de clientes anteriores são armazenados. Concluída a reserva, o cliente receberá a confirmação da mesma (pessoalmente, por e-mail ou caixa postal), sendo gerada também uma conta da reserva para o cliente.

A seguir, são especificados os modelos MCN e MCU gerados pela ADComp na etapa definição de requisitos.

3.3.1. Modelo Conceitual de Negócio

O MCN é um modelo estático do domínio do problema, utilizado para identificar conceitos, métodos e relacionamentos. É representado por meio do diagrama de classes da UML, cuja ênfase é capturar o conhecimento do domínio. Para especificar o MCN, na ADComp são utilizadas as seguintes primitivas da UML: classe, atributo, método e relacionamento, como ilustradas na Tabela 3.1.

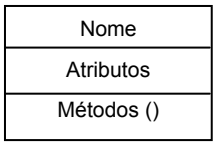
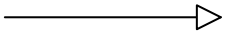
	Classe
Atributo: tipo	Atributo Simples
Método (param: tipo)	Método
	Relacionamento: Associação
	Relacionamento: Generalização

TABELA 3.1: PRIMITIVAS DO MODELO CONCEITUAL DE NEGÓCIO.

O MCN do software de reservas, ilustrado na Figura 3.3, é formado por dez (10) classes conceituais e construído na ferramenta MVCASE: Cadeia de Hotéis, Hotel, Secretário, Cliente, Endereço, Reserva, Quarto, Tipo de Quarto, Conta e Pagamento. Neste exemplo, não são representados atributos e métodos, e os relacionamentos do modelo representam as ligações entre as classes do domínio do problema.

3.3.2. Modelo de Casos de Uso

O MCU provê uma modelagem semiformal da interação software-usuário, descrevendo as interações entre um usuário (ou outro ator externo) e o software, ajudando na definição da fronteira do mesmo. Este é representado por meio do diagrama de casos de uso da UML, no qual são utilizadas as seguintes primitivas (Tabela 3.2): caso de uso, ator, relacionamentos: associação, generalização, *extend* e *include*.

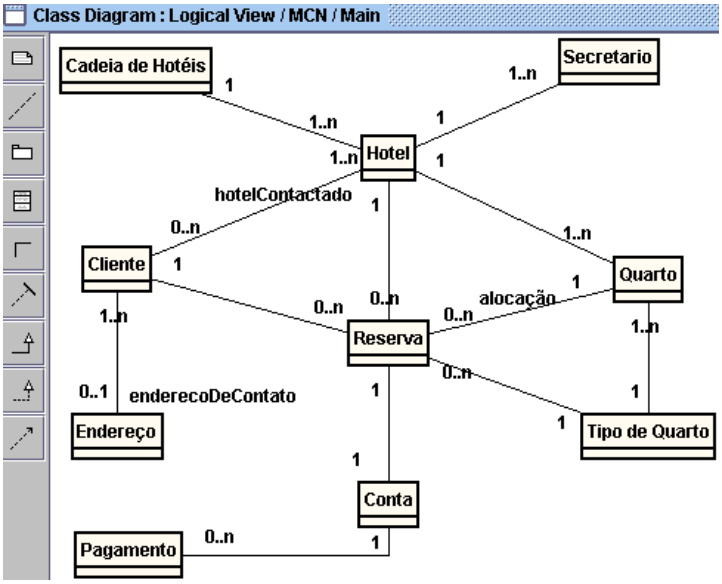


FIGURA 3.3: MCN INICIAL (CHESSMAN; DANIELS, 2001, p. 70).

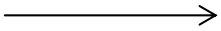
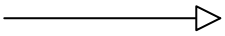
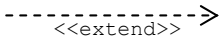
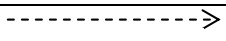
Nome	Caso de Uso
Nome	Ator
	Relacionamento: Associação
	Relacionamento: Generalização
 <<extend>>	Relacionamento: Dependência *Extend*
 <<include>>	Relacionamento: Dependência *Include*

TABELA 3.2: PRIMITIVAS DO MODELO DE CASOS DE USO.

O caso de uso é um tipo de classificador que representa uma funcionalidade provida pelo software, como o caso de uso `Fazer Reserva`, ilustrado na Figura 3.4. Atores são os participantes nos casos de uso, interagem com o software e tipicamente representam um papel. O relacionamento associação representa a participação do ator no caso de uso, podendo ou não apresentar cardinalidade. O relacionamento generalização permite que casos de uso possam ser generalizados em outros casos de uso. O relacionamento de dependência *include* de um caso de uso E para um caso de uso F indica que uma instância do caso de uso E também conterá o comportamento especificado por F. O relacionamento de dependência *extend* de um caso de uso A para um caso de uso B indica que uma instância de B pode ser aumentada pelo comportamento de A.

Assim, pode-se observar na Figura 3.4 que o MCU do software de reserva contém sete casos de uso e quatro atores com seus respectivos relacionamentos.

Na Tabela 3.3 são descritos, resumidamente, os casos de uso e os atores.

Casos de Uso	Atores
• Fazer Reserva; • Cancelar Reserva; • Alterar Reserva; • Retirar Reserva: o sistema de conta recebe o pagamento da reserva; e o hóspede pega a reserva; • *Process No-Shows*: procedimentos armazenados, gatilhos ou arquivos em lote (*batch)* executados automaticamente, por *clock* ou manualmente por um usuário; • Cadastrar Cliente: concentra todas as atividades de incluir, modificar e excluir dados de clientes; • Cadastrar Secretario: concentra todas as atividades de incluir, modificar e excluir dados de secretários; • Cadastrar Quarto: concentra todas as atividades de incluir, modificar e excluir dados de quartos; • Identificar Reserva.	• Gerente: realiza as principais tarefas de reservar um quarto; • Hóspede: faz/retira sua própria reserva; • Sistema de Conta: sistema ligado ao software de reservas; • Administrador: controla todas as reservas do hotel.

TABELA 3.3: CASOS DE USO E ATORES DO MODELO DE CASOS DE USO.

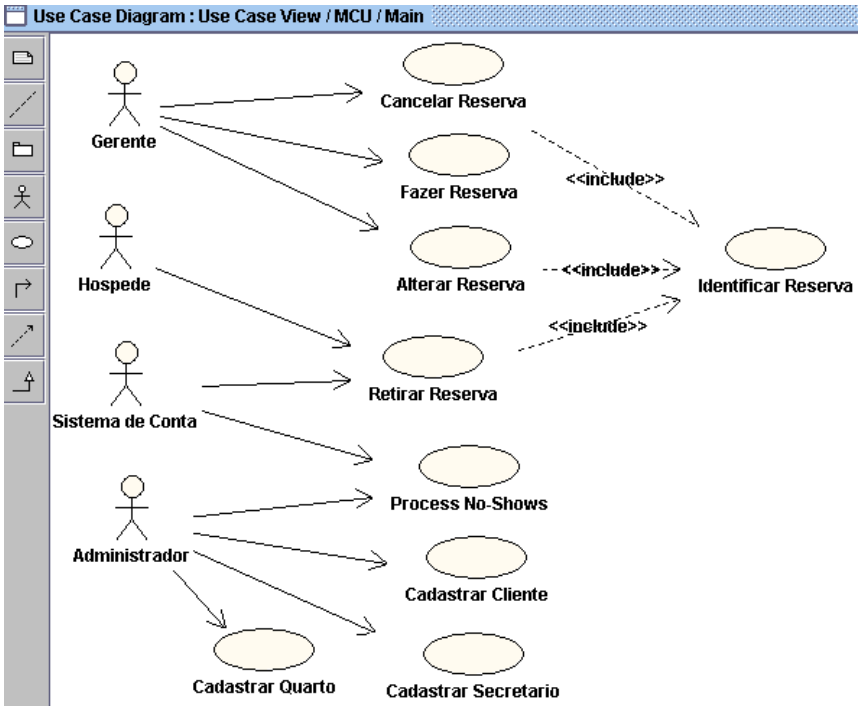


FIGURA 3.4: MODELO DE CASOS DE USO (CHESSMAN; DANIELS, 2001, p. 81).

3.4. ETAPA 2: DEFINIÇÃO DE COMPONENTES

O objetivo desta etapa é identificar e especificar os componentes a partir dos artefatos produzidos na primeira etapa. Esta etapa apresenta três fases: (1) identificação do componente, (2) interação do componente e (3) especificação do componente. A primeira fase produz uma arquitetura inicial dos componentes a partir

dos requisitos, identificando as interfaces e os componentes. A segunda fase descobre as operações necessárias e aloca responsabilidades. E a terceira fase cria especificações precisas das operações, interfaces e componentes.

Em cada fase é gerado um conjunto de modelos e de especificações que são agrupados em três grandes artefatos:

1. Especificações da interface: conjunto de especificações individuais de interfaces que descreve “o quê” o componente que oferece a interface deve fazer e o que um cliente da interface pode esperar, isto é, quais os métodos e operações que a interface disponibilizará para o cliente;
2. Especificações do componente: conjunto de especificações individuais de componentes que representa quais interfaces são oferecidas e utilizadas pelos mesmos. Nas especificações de componente são definidas as pré-condições e as pós-condições das operações, além de quaisquer restrições que o componente requer entre as interfaces e a implementação de suas operações; e
3. Arquitetura do componente: descreve como as especificações de componente e as de interfaces se relacionam no domínio do problema, incluindo as dependências das especificações individuais da interface dentro das dependências do componente.

3.4.1. Identificação do Componente

Baseando-se no conceito de software em camadas, nesta fase os componentes são organizados em componentes de sistema (serviços de sistema) e de negócio (serviços de negócio) (Figura 3.5). A meta é identificar um conjunto inicial de interfaces de negócio para os componentes de negócio e um conjunto inicial de interfaces de sistema para os componentes de sistema, e integrá-los formando uma arquitetura inicial de componentes. As interfaces e os componentes de sistema e de negócio são representados utilizando as seguintes primitivas do diagrama de classes da UML: classes, atributos, estereótipos, interface e relacionamentos, como ilustrado na Tabela 3.4.

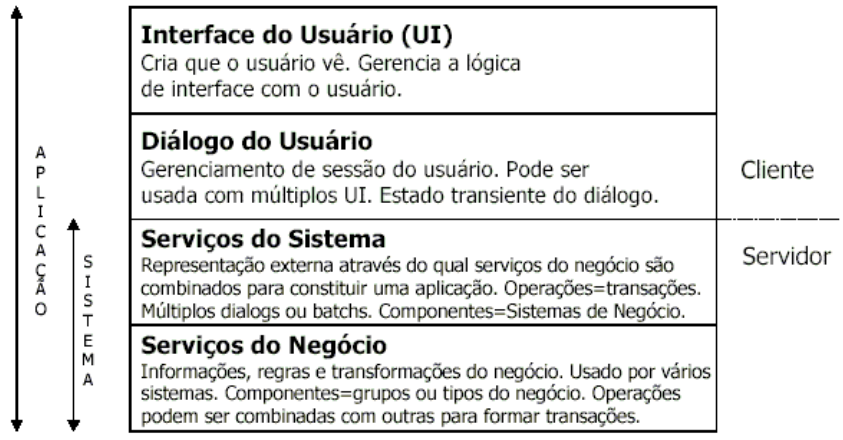


FIGURA 3.5: APLICAÇÃO EM CAMADAS (CHESSMAN; DANIELS, 2001, p. 11).

A classe tipo de negócio é utilizada no modelo de tipo de negócio que auxilia o engenheiro na identificação das interfaces de negócio. A classe interface é utilizada para representar as interfaces de sistema e a classe especificação de componente é utilizada para representar os componentes de sistema e de negócio com suas interfaces que são representadas pela notação “lolipop”. Por definição, um atributo parametrizado depende dos valores de seus parâmetros, por exemplo, o número de contato de uma pessoa pode depender do dia da semana.

<<type>> Nome Atributos	Classe: Tipo de Negócio
<<interface type>> INome Métodos()	Classe: Interface
<<comp spec>> Nome	Classe: Especificação de Componente
Atributo: tipo	Atributo Simples
Atributo (atributo: tipo)	Atributo Parametrizado
Método (param: tipo)	Método
INome	Interface (“*lollipop*” — UML)
	Relacionamento: Associação
	Relacionamento: Agregação
	Relacionamento: Composição
	Relacionamento: Dependência

TABELA 3.4: PRIMITIVAS DOS MODELOS DA IDENTIFICAÇÃO DO COMPONENTE.

Primeiramente, as interfaces de sistema são identificadas a partir do MCU da etapa de definição de requisitos (Figura 3.4), onde cada caso de uso representa uma interface de sistema, e cada fluxo/passo do caso de uso uma operação na interface. Na Figura 3.6 é ilustrado o mapeamento do caso de uso `Fazer Reserva` e seus passos para a interface de sistema `IFazerReserva` e suas operações. Analisando os passos do caso de uso `Fazer Reserva`, observa-se que o software deve identificar os quartos necessários, prover os preços e requisitar uma reserva. Desta forma as três operações para a interface são: (1) consultar detalhes do hotel — `getDetalhesHotel()`, (2) consultar informações sobre quartos — `getInfoQuarto()` e (3) fazer a reserva — `fazerReserva()`.

Para ilustrar os demais modelos e especificações de componentes são utilizados principalmente duas interfaces de sistema `IFazerReserva` e `IPegarReserva`, conforme ilustradas na Figura 3.6, respectivamente. Para o caso de uso `Pegar Reserva`, o primeiro passo é identificar e retornar dados sobre a reserva, que corresponde à operação `getReserva()` na interface

`IPegarReserva`, cujos detalhes da reserva são confirmadas ao hóspede. Ao iniciar a estadia no hotel, o software deve alocar um quarto e notificar o sistema de conta que uma estadia começou correspondendo à operação `iniciarEstadia()` na interface.

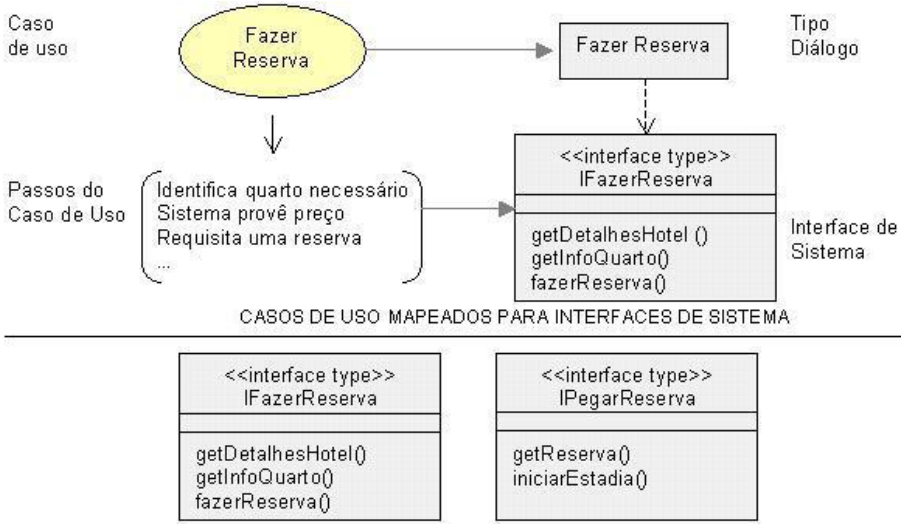


FIGURA 3.6: MAPEAMENTO DE CASOS DE USO PARA INTERFACES DE SISTEMA (CHESSMAN; DANIELS, 2001, p. 86-87).

Identificadas as interfaces de sistema e suas operações iniciais, a próxima tarefa é identificar as interfaces de negócio que são abstrações da informação que devem ser gerenciadas pelo software (CHEESMAN; DANIELS, 2001). O processo de identificação das interfaces consiste em:

1. Refinar o MCN para especificar e construir o modelo de tipo do negócio — ou modelo de tipo;
2. Refinar o modelo de tipo, adicionando uma interface de negócio para cada tipo (ou classe de tipo de negócio) por meio de um relacionamento de composição, indicando as responsabilidades da interface de negócio com o tipo associado.

Para identificar as interfaces de negócio, as técnicas do *UML Components* foram adaptadas. A alteração ocorre na definição da interface para cada tipo e não para os tipos principais (*core types*), conforme definido em (CHESSMAN; DANIELS, 2001). Esta adaptação é importante, pois oferece uma maior separação para as operações das interfaces de negócio, evitando o acúmulo e a concentração de muitas operações numa única interface.

O modelo de tipo elaborado é um modelo de especificação que objetiva modelar informações relevantes do escopo do negócio por meio do estereótipo `<<type>>`. Para exemplificar a identificação das interfaces de negócio, o MCN da Figura 3.3 é refinado para o modelo de tipo da Figura 3.8, onde classes são removidas ou adicionadas conforme a versão do novo escopo, acrescentando atributos e tipos aos atributos. Como ilustrado na Figura 3.7, os tipos `Cadeia de Hotéis`, `Pagamento`, `Conta`, `Secretário` e `Endereço` e o relacionamento

`Hotel-Cliente` foram excluídos para atender aos novos requisitos que incluem somente suporte a uma única cadeia de hotéis e não gerenciar contas e pagamentos; além da eliminação dos tipos `Secretário` e `Endereço` para simplificar o exemplo. O relacionamento incluído (Figura 3.8) foi o `Hotel-Tipo de Quarto` para refletir os tipos de quarto oferecidos pelo Hotel onde cada tipo de quarto possui um preço.

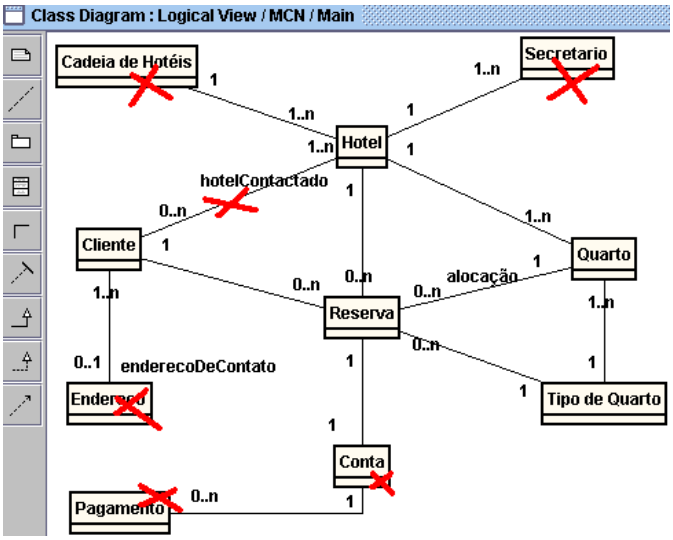


FIGURA 3.7: REDEFINIÇÃO DO MCN INICIAL (CHESSMAN; DANIELS, 2001, p. 86-87).

A partir do modelo de tipo, interfaces de negócio são criadas para cada tipo e relacionadas a ele por meio de um relacionamento de composição. A proposta é apontar que informações serão gerenciadas por quais interfaces, estabelecendo a composição dos tipos para a interface apropriada. Cada tipo deve ser propriedade de exatamente uma única interface. Desta forma, o refinamento do modelo de tipo é representado pelo modelo de responsabilidade das interfaces, como ilustrado na Figura 3.9. Por exemplo, a interface `ICliente` representa as informações do tipo `Cliente`. Cada interface gerencia um conjunto de instâncias permitindo que instâncias individuais do tipo possam ser separadas em objetos de componente.

É importante ressaltar que este é um modelo de tipo está no nível de especificação e não descreve o caminho no qual seu estado é implementado ou persistido.

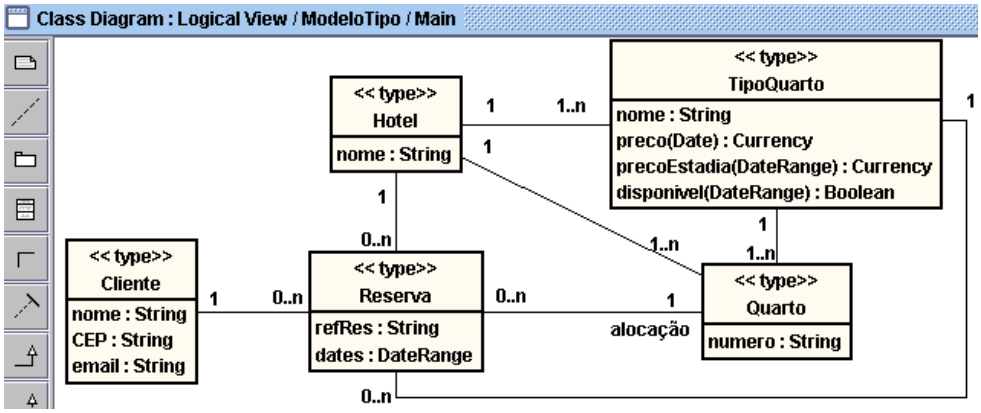


FIGURA 3.8: MODELO DE TIPO DE NEGÓCIO (CHESSMAN; DANIELS, 2001, p. 91).

O próximo passo desta fase é criar um conjunto inicial de especificações de componente de sistema e de negócio — que serão refinados na fase de interação do componente — e definir como estes se relacionam.

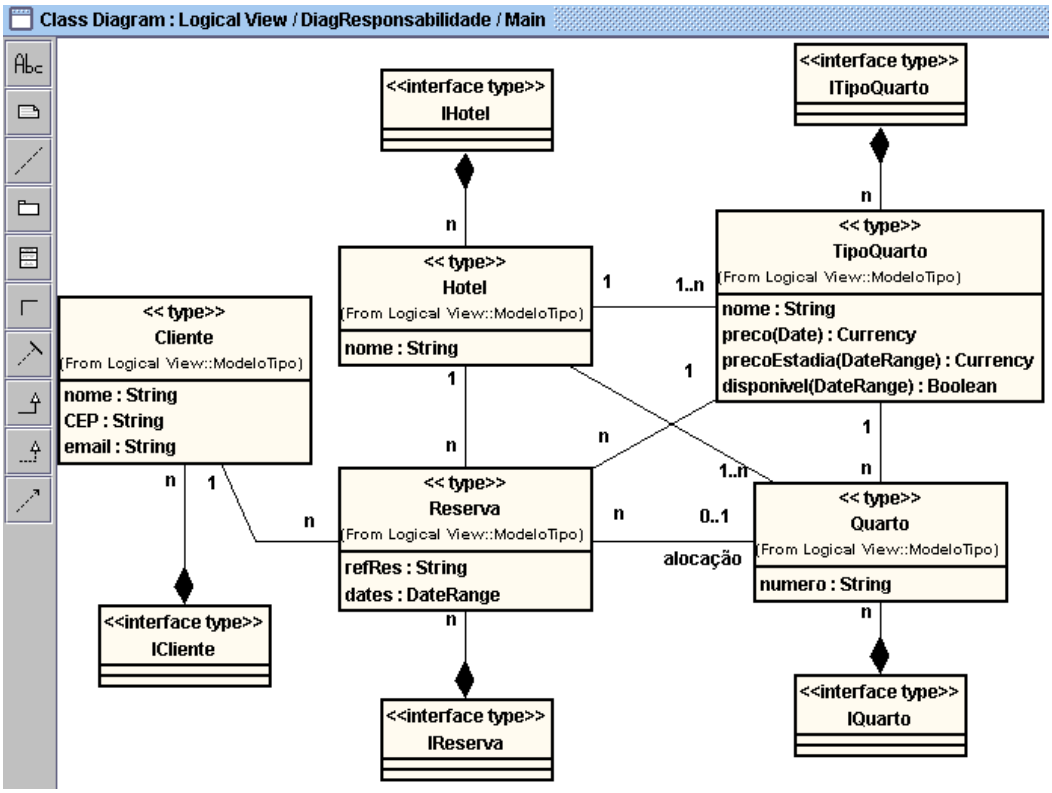


FIGURA 3.9: MODELO DE RESPONSABILIDADE DAS INTERFACES NO MODELO DE TIPO.

Os componentes de sistema realizam as operações das interfaces de sistema, podendo oferecer uma ou mais interfaces. Componentes com mais de uma interface podem ocorrer quando os conceitos representados por diferentes interfaces apresentam o mesmo ciclo de vida (criadas e excluídas no mesmo tempo); as interações entre as interfaces são complexas, freqüentes ou envolvem uma grande quantidade de dados, e há a necessidade de se manter a granularidade do componente num tamanho razoável para o gerenciamento do projeto e para propostas organizacionais.

Por exemplo, para o domínio do problema em questão, são definidos dois componentes de sistema ilustrados na Figura 3.10: (1) `SistemaReserva`, composto pelas interfaces `IFazerReserva` e `IPegarReserva`; e (2) `SistemaFaturamento`, composto pela interface `IFaturamento` para fazer a comunicação com um software específico de gerenciamento de contas. Na Figura 3.10, observa-se que as interfaces são apresentadas na notação “lolipop” e estas são relacionadas ao componente pelo relacionamento de associação.

Definidos os componentes de sistema e suas interfaces, deve-se definir quais interfaces o componente de sistema utiliza, mesmo não sabendo se estas realmente existirão nesta fase, pois é útil na definição inicial da arquitetura do componente. Na Figura 3.10 são ilustradas por meio do relacionamento de dependência as interfaces

utilizadas pelo componente `SistemaReserva` como: `IFaturamento`, `ICliente`, `IHotel`, `IReserva`, `IQuarto` e `ITipoQuarto`.

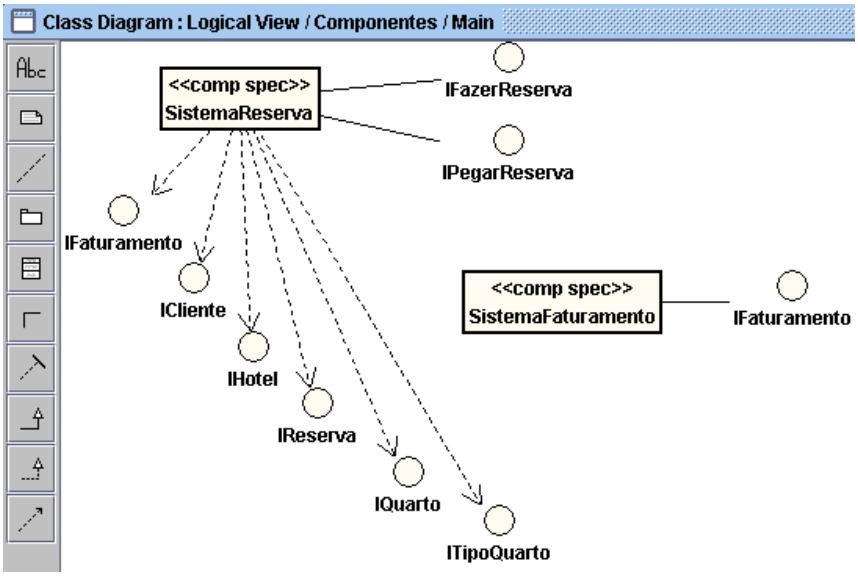


FIGURA 3.10: ESPECIFICAÇÕES DO COMPONENTE DE SISTEMA.

Os componentes de negócio realizam as operações das interfaces de negócio a serem identificadas em fases posteriores, e estão restritos a oferecer apenas uma única interface, diferentemente dos componentes de sistema que podem oferecer mais de uma interface. Na Figura 3.11, para cada interface de negócio, são ilustrados os respectivos componentes de negócio identificados no domínio do problema software de reserva. Observa-se que as interfaces de negócio são representadas pela notação “lolipop” relacionadas ao componente por meio do relacionamento de associação.

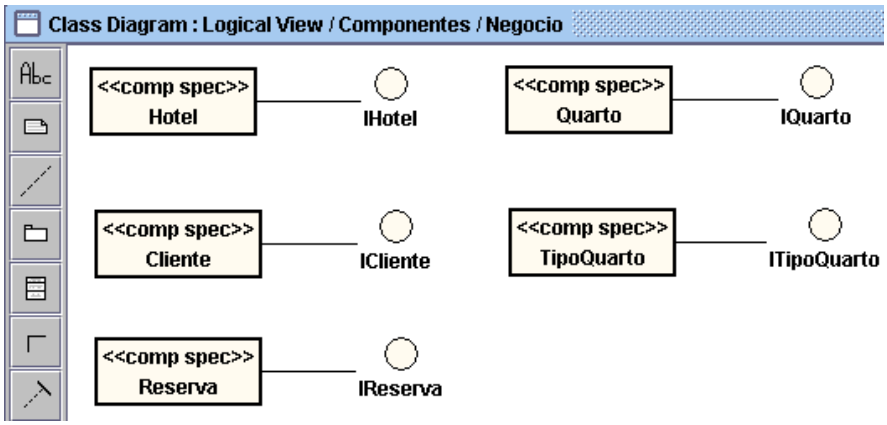


FIGURA 3.11: ESPECIFICAÇÕES DOS COMPONENTES DE NEGÓCIO.

Definidos os conjuntos iniciais de especificações dos componentes, incluindo interfaces suportadas (oferecidas) e dependentes (utilizadas), a arquitetura inicial da especificação do componente é elaborada incluindo os componentes e suas interfaces num único modelo. Na Figura 3.12 é ilustrado o componente de sistema `SistemaReserva` que oferece as interfaces de sistema `IFazerReserva` e `IPegarReserva`, e utiliza cinco interfaces de negócio com seus respectivos componentes de negócio `Hotel`, `Cliente`, `Reserva`, `Quarto`, `TipoQuarto`; e

uma interface de sistema com seu respectivo componente de sistema, `SistemaFaturamento`.

A ênfase nesta fase está na descoberta de quais informações necessitam ser gerenciadas, que interfaces são necessárias para gerenciar tais informações, quais componentes são necessários para prover as funcionalidades, e como os componentes se combinam.

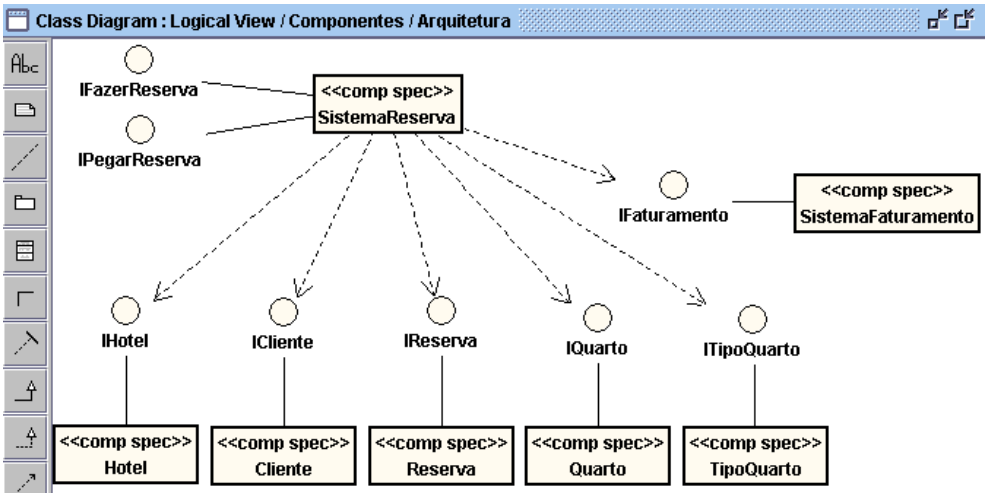


FIGURA 3.12: ARQUITETURA INICIAL DA ESPECIFICAÇÃO DO COMPONENTE.

3.4.2. Interação do Componente

A fase de interação do componente examina como cada uma das operações de sistema é executada. É nesta fase que os detalhes da estrutura do sistema emergem, mostrando claramente as dependências entre os componentes e como estes trabalham para entregar a funcionalidade requerida.

O objetivo desta fase é descobrir como e quais operações nas interfaces de sistema são “implementadas” nas várias interfaces de negócio, além de descobrir outras operações de negócio necessárias para realizar as operações de sistema.

A modelagem da interação é uma técnica de modelagem comportamental (CHEESMAN; DANIELS, 2001) que especifica restrições nos fluxos de execução. Os modelos de interação são representados pelo diagrama de colaboração da UML, com as seguintes primitivas: objeto, link e link de mensagem, como ilustrado na Tabela 3.5.

/ NomePapel : NomeClassificador	Objeto
———	Link
———▶	Link de Mensagem

TABELA 3.5: PRIMITIVAS DO MODELO DE COLABORAÇÃO.

A associação entre os objetos do modelo de colaboração representa a realização de uma operação. Estes objetos são nomeados pelo NomePapel que

(define a interface do objeto) e o NomeClassificador (nome da especificação do componente). Os links são utilizados para fazer a interação entre os objetos; e os links de mensagem representam um estímulo ou mensagem numa certa direção (Figura 3.13).

Para cada operação da interface de sistema é desenvolvido um modelo de colaboração, que exibe interações que representam possíveis execuções do fluxo, por exemplo, fazer uma reserva para um cliente cadastrado e fazer uma reserva para um cliente sem cadastro. As assinaturas das operações devem ser definidas utilizando a OCL (*Object Constraint Language*) (OCL, 2001) — uma linguagem declarativa que permite construir expressões lógicas, sem ambigüidades. Como por exemplo, para a interface de sistema `IFazerReserva()` há três operações, cujas assinaturas são exibidas na Tabela 3.6:

(1) `getDetalhesHotel()`: retorna uma lista de hotéis que iniciam por uma determinada palavra ou conjunto de letras (*string*) informada pelo usuário;

(2) `getInfoQuarto()`: retorna informações sobre o quarto, como sua disponibilidade de vaga e seu preço de acordo com o hotel escolhido, o período da estadia e o tipo de quarto escolhidos pelo usuário; e

(3) `fazerReserva()`: cria uma reserva a partir dos dados de detalhes da reserva requerida e as informações sobre o cliente. Em seguida, emite o número de referência da reserva e notifica o cliente por e-mail ou caixa postal.

`IFazerReserva::getDetalhesHotel(in match:String):DetalhesHotel[]`

`IFazerReserva::getInfoQuarto(`
`  in res:DetalhesReserva,`
`  out disponibilidade:Boolean,`
`  out preco:Currency)`

`IFazerReserva::fazerReserva(`
`  in res:DetalhesReserva,`
`  in cli:DetalhesCliente,`
`  out refRes:String):Integer`

TABELA 3.6: ASSINATURAS DAS OPERAÇÕES DA INTERFACE IFAZERRESERVA.

A partir da definição das assinaturas, para cada operação da interface de sistema é elaborado um modelo de colaboração, como ilustrado na Figura 3.13. Por exemplo, a operação `fazerReserva()` interage com as interfaces de negócio `IReserva` e `ICliente`. Além da operação para fazer reserva, duas novas operações são necessárias para completar a operação de reserva (`getCliente()`, `notificarCliente()`) que são inseridas na interface `ICliente`.

Definidas as operações de negócios, as especificações das interfaces de negócio devem ser atualizadas, refletindo as operações identificadas por meio dos modelos de colaboração, como por exemplo, nas Figuras 3.14 e 3.15 em que são

ilustradas as operações da interface de sistema `IFazerReserva` e as operações das interfaces de negócio descobertas, respectivamente.

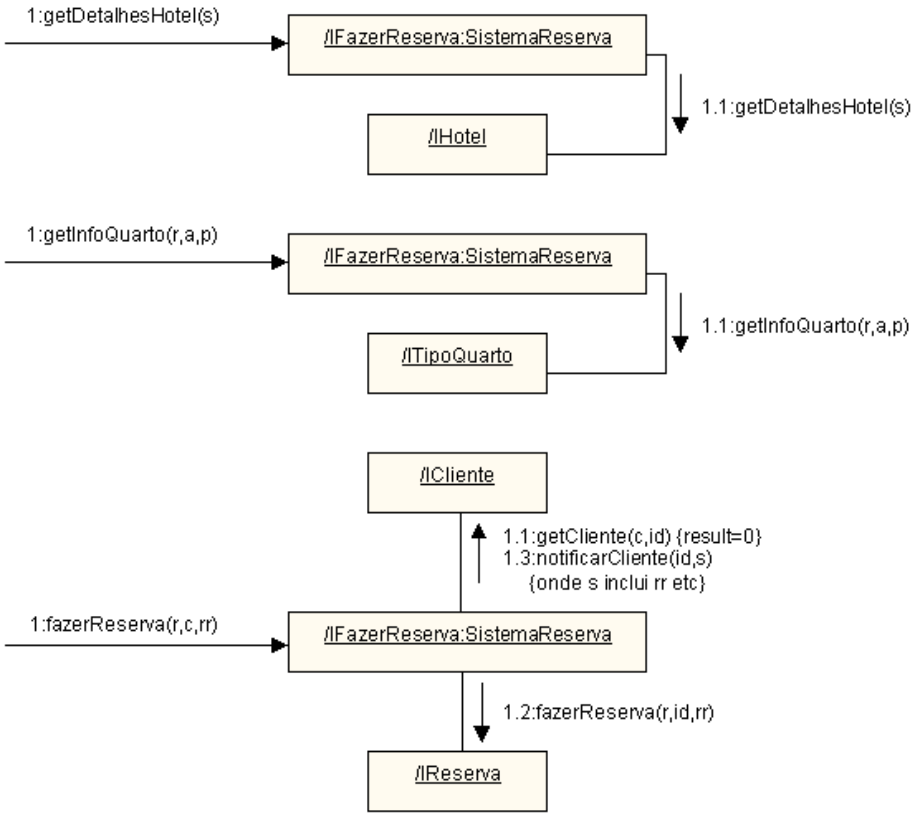


FIGURA 3.13: MODELO DE INTERAÇÃO DA INTERFACE DE SISTEMA IFAZERRESERVA.

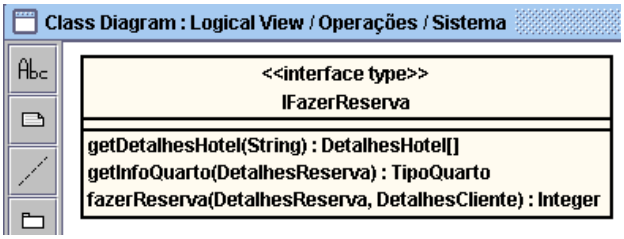


FIGURA 3.14: OPERAÇÕES DA INTERFACE DE SISTEMA IFAZERRESERVA.

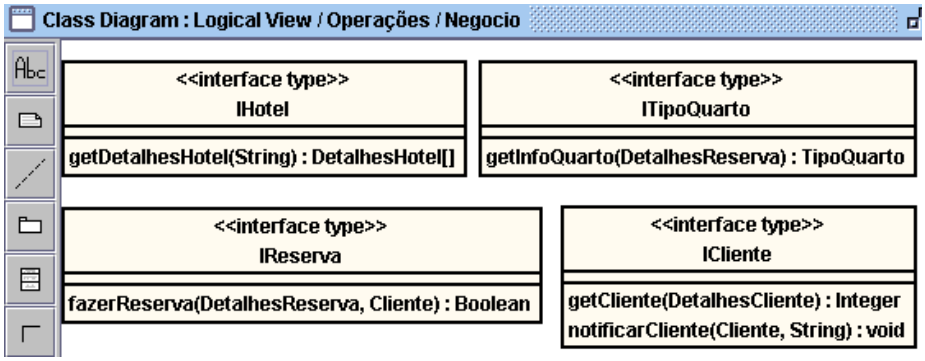


FIGURA 3.15: OPERAÇÕES DAS INTERFACES DE NEGÓCIO.

3.4.3. Especificação do Componente

Nesta fase são definidos os contratos de uso e de realização para os componentes. Contrato é um acordo formal entre duas ou mais partes; descreve

detalhes do acordo sem ambigüidades, envolvendo a definição de responsabilidades e obrigações de cada uma das partes envolvidas (CHEESMAN; DANIELS, 2001). Similarmente, os contratos de uso e de realização definem responsabilidades e obrigações.

O contrato de uso é especificado na forma de uma interface e descreve o relacionamento entre uma interface do objeto de componente e um cliente, definindo "o quê" o componente que oferece a interface deve fazer e "o quê" o cliente da interface pode esperar. O contrato de uso inclui os seguintes itens:

1. Operações: uma lista de operações providas pela interface, incluindo suas assinaturas e definições, em que cada operação é definida da seguinte maneira:
    a. Pré-condição: define condições que garante que a pós-condição é verdadeira;
    b. Pós-condição: descreve quais os efeitos da operação, dada a ocorrência correta da pré-condição.
2. Modelo de Informação (MI): uma definição abstrata de qualquer informação necessária para ajudar na especificação das operações da interface, como atributos, relacionamentos e tipos de atributos. Além de outras informações que as operações da interface necessitam, como estruturas de dados utilizadas e quais os tipos de dados que devem manter. Por exemplo, qualquer objeto de componente que oferece a interface `ICliente` deve manter informações sobre os clientes, como: nome, e-mail e caixa postal.

O contrato de realização relaciona uma especificação de componente e sua implementação. A especificação do componente define como a implementação deve interagir com outros componentes e informar se interações particulares devem ser incluídas com outros componentes como parte da implementação das operações.

A seguir, são explicitadas tais especificações no estudo de caso do software de reserva.

Inicialmente, é elaborado um MI para cada interface de negócio. Na Figura 3.16 a interface de negócio `ICliente` está associada ao tipo de informação `Cliente`, isto é, qualquer objeto de componente que oferece esta interface deve manter informações sobre os clientes. E para cada cliente pode haver outras informações necessárias, como indicado pela estrutura `DetalhesCliente`.

Além do MI, o contrato de uso inclui a descrição das operações da interface por meio das pré-condições e das pós-condições. Na UML essas condições são especificadas formalmente por meio da OCL.

No Quadro 3.1 é ilustrado um exemplo para a operação `getDetalhesCliente()`, uma nova operação da interface `ICliente` que deve retornar informações de um determinado cliente, dado seu código de identificação.

Na pré-condição o parâmetro de entrada `cli` corresponde ao identificador de um único cliente, e na pós-condição, o resultado esperado são informações sobre o cliente, como: nome, CEP e email retornados na estrutura `DetalhesCliente`.

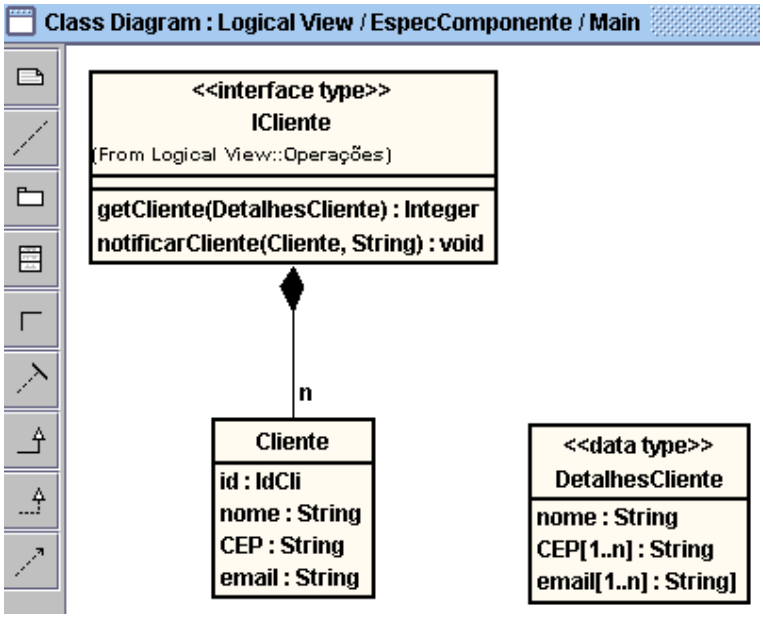


FIGURA 3.16: MODELO DE ESPECIFICAÇÃO DA INTERFACE DE NEGÓCIO.

```
context ICliente::getDetalhesCliente(in cli:IdCli):DetalhesCliente
pre:
  -- cli é um id de cliente válido
  cliente->exists(c | c.id = cli)
pos:
 -- os detalhes retornados correspondem aos detalhes do cliente cujo id é cli
 -- buscar o cliente
 Let oCliente = cliente->select(c | c.id = cli) in
   -- especifica o resultado
   result.nome = oCliente.nome and
   result.CEP = oCliente.CEP and
   result.email = oCliente.email
```

QUADRO 3.1: ESPECIFICAÇÃO EM OCL DE UMA OPERAÇÃO.

Além das pré-condições e pós-condições, regras de negócio podem ser definidas por meio de invariantes — restrições ligadas a um tipo e devem ser verdadeiras para todas as instâncias do tipo (CHEESMAN; DANIELS, 2001). Um exemplo de invariante pode ser um indicador de multiplicidade no relacionamento entre dois tipos, por exemplo, uma reserva está associada a um único cliente (Figura 3.8); ou podem ser escritas por meio da OCL, como ilustrado no Quadro 3.2, indicando que todo cliente cadastrado deve possuir um e-mail para contato.

```
context cli:Cliente inv:
  -- todo cliente deve possuir um e-mail
  cli.email <> ' '
```

QUADRO 3.2: INVARIANTE ESCRITA EM OCL.

As interfaces de sistema também apresentam MI, como as interfaces de negócio, mesmo que as operações da interface de sistema apenas repassem a

chamada para a interface de negócio apropriada. A existência de um MI da interface não implica que a interface deva armazenar a informação de forma persistente. De fato, interfaces de sistema nunca possuem armazenamento persistente; suas implementações obtêm a informação necessária por meio dos componentes de negócio. Como qualquer interface, o MI da interface de sistema deve conter informação suficiente para permitir que as operações sejam especificadas.

A definição do MI da interface de sistema inicia-se a partir do modelo de tipo. Entretanto, diferentemente das interfaces de negócio, onde o modelo de responsabilidade da interface indica quais os tipos que são ou não necessários, as informações para as interfaces de sistema podem não ser óbvias até que as operações tenham sido especificadas.

Por exemplo, como ilustrado na Figura 3.17, especificadas as operações foi concluído que o MI para a interface `IFazerReserva` não necessita de nenhum atributo do tipo `Quarto`. Isto não significa que para outros modelos de informação das demais interfaces de sistema esta informação não seja necessária, uma vez que cada interface possui seu próprio MI e os tipos podem não ser os mesmos em cada modelo.

O contrato de realização consiste de informações adicionais de especificações que o desenvolvedor e o construtor precisam conhecer, especialmente as dependências de um componente sobre outras interfaces. Toda especificação de componente deve informar quais interfaces são utilizadas por suas realizações, isto é, quais interfaces o componente depende; quais interfaces são oferecidas pelo componente, como exibido por meio do modelo da arquitetura inicial do componente (Figura 3.12).

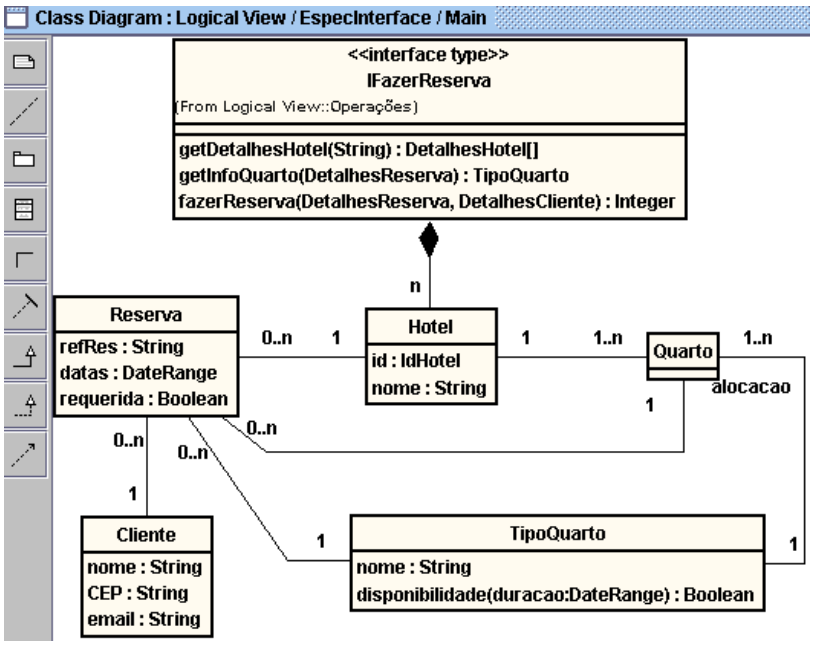


FIGURA 3.17: MODELO DE ESPECIFICAÇÃO DA INTERFACE DE SISTEMA.

Como descrito na fase de identificação do componente (Figura 3.10 e 3.11), no modelo de especificação para o componente de negócio `Hotel`, ilustrado na Figura

3.18, é informado que este componente deve oferecer a interface `IHotel` e que o mesmo não é obrigado a utilizar qualquer outra interface. Na Figura 3.19, no modelo de especificação do componente de sistema `SistemaReserva` é descrito que este componente deve oferecer duas interfaces de sistema (`IFazerReserva` e `IPegarReserva`) e utilizar seis outras interfaces (`IFaturamento`, `IHotel`, `ICliente`, `IReserva`, `IQuarto` e `ITipoQuarto`).

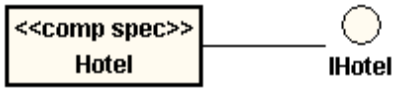


FIGURA 3.18: MODELO DE ESPECIFICAÇÃO DO COMPONENTE DE NEGÓCIO.

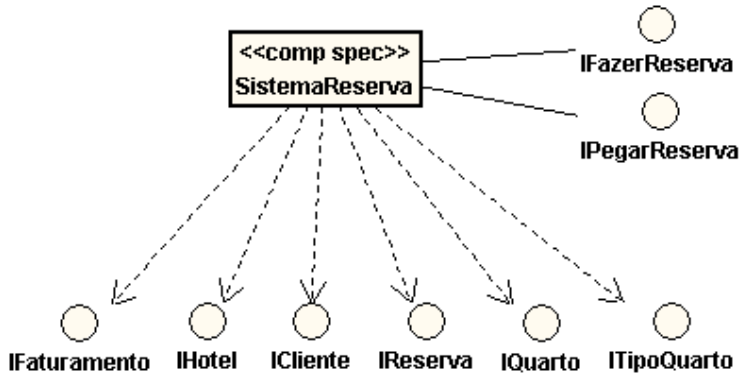


FIGURA 3.19: MODELO DE ESPECIFICAÇÃO DO COMPONENTE DE SISTEMA.

Completando o contrato de realização, duas restrições textuais escritas em OCL são definidas. Na primeira são descritas como as interfaces oferecidas relacionam-se umas com as outras; e, na segunda, como estão relacionadas às interfaces oferecidas e utilizadas.

A primeira restrição evita que um tipo de informação, encontrado em duas interfaces oferecidas pelo mesmo componente apresente conceitos diferentes. Por exemplo, reservas criadas por um objeto do componente `SistemaReserva` por meio da interface `IFazerReserva` devem ser as mesmas reservas localizadas e utilizadas pela interface `IPegarReserva`, como ilustrado no Quadro 3.3.

```
context SistemaReserva
  -- restrições entre as interfaces oferecidas
  IFazerReserva:hotel = IPegarReserva:hotel
  IFazerReserva:reserva = IPegarReserva:reserva
  IFazerReserva:cliente = IPegarReserva:cliente
```

QUADRO 3.3: RESTRIÇÕES ENTRE INTERFACES OFERECIDAS.

A segunda restrição informa aos desenvolvedores a necessidade de garantir que as informações utilizadas para satisfazer a pós-condição das operações oferecidas pelas interfaces de sistema devem corresponder às informações descritas nas interfaces de negócio. Por exemplo, a informação sobre cliente utilizada na interface de sistema para fazer uma reserva deve ser a mesma informação de cliente existente no componente de negócio sobre clientes. Por exemplo, no Quadro 3.4 a interface de sistema `IFazerReserva` utiliza a informação cliente que deve ser

a mesma referida pela interface de negócio `ICliente`.

```
context SistemaReserva
  -- restrições entre as interfaces oferecidas e interfaces utilizadas
  IFazerReserva:hotel = IHotel:hotel
  IFazerReserva:reserva = IReserva:reserva
  IFazerReserva:cliente = ICliente:cliente
```

QUADRO 3.4: RESTRIÇÕES ENTRE INTERFACES OFERECIDAS E UTILIZADAS.

3.5. ETAPA 3: IMPLEMENTAÇÃO DE COMPONENTES

Nesta etapa, as especificações e a arquitetura de componentes são mapeadas e implementadas de acordo com a tecnologia de componentes adotada. De acordo com a abordagem ADComp, esta etapa é dependente de tecnologias — linguagem de programação Java e modelo de componentes. Estas tecnologias são aplicadas com o auxílio da ferramenta MVCASE que oferece ao engenheiro e ao desenvolvedor as facilidades de implementação gerando código Java e fazendo o *deploy* dos componentes por meio de um *wizard* para o modelo de componentes escolhido.

A MVCASE (2004) é uma ferramenta CASE que oferece suporte ao DBC distribuídos; integra o método *Catalysis* e o padrão CORBA para especificação e distribuição de componentes; permite a especificação de requisitos de sistemas em alto nível de abstração utilizando a linguagem de modelagem UML (ALMEIDA et. al., 2002); e a projetos baseados nas tecnologias EJB, JDK (*Java Development Kit*) e Delphi.

Neste trabalho, o modelo de componentes adotado é o EJB, por ser um modelo suportado pela MVCASE e, principalmente, por ser independente de plataforma.

De acordo com a tecnologia EJB, as regras de mapeamento das especificações e da arquitetura dos componentes ocorrem da seguinte forma:

1. Componentes de negócio são mapeados para *entity beans* CMP;
2. Componentes de sistema são mapeados para *session beans*:
    a. Componentes de sistema que não precisam manter um estado conversacional com o cliente são mapeados para *session beans* do tipo *stateless*.
    b. Componentes de sistema que precisam manter um estado conversacional com um único cliente são mapeados para *session beans* do tipo *stateful*.

Na MVCASE, o engenheiro deve realizar, em ordem, as seguintes tarefas para implementar os componentes e gerar um novo software baseado em componentes:

(1) Construir os componentes de negócio;
(2) Construir os componentes de sistema;
(3) Implementar as operações dos componentes;

(4) Definir o modelo de componentes; e
(5) Fazer o *deploy* dos componentes.

Tais tarefas são descritas resumidamente a seguir.

Tarefa 1: Construir os componentes de negócio. A partir do modelo de tipo (Figura 3.8) gerado na fase de identificação do componente, o engenheiro inicia a construção dos componentes de negócio, pois é a partir do modelo de tipo que as interfaces de negócio são identificadas, e posteriormente, os componentes de negócio são construídos. Desta maneira, o engenheiro deve:

a) Alterar o estereótipo `<<type>>` para o estereótipo `<<LocalEntity>>` pré-definido na ferramenta, indicando que esta classe será transformada num componente do tipo *entity bean* local, como ilustrado na Figura 3.20;
b) Conferir os tipos dos atributos, observando que as chaves devem ser de tipos que encapsulam tipos primitivos em Java, como *Integer* e não *int*, para chaves do tipo inteiro, por exemplo;
c) Alterar todos os relacionamentos marcando-os como "n*avigable*" em ambos os lados do relacionamento, o que permite a "navegação" de um lado ao outro dos componentes (Figura 3.20);

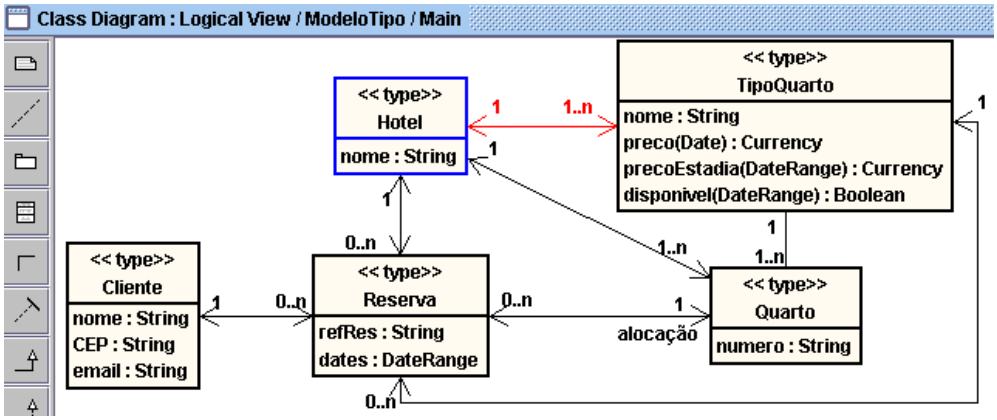


FIGURA 3.20: MODELO DE TIPO REVISADO.

d) Transformar as classes em objetos EJB *entity* por meio da funcionalidade *Transform Classes to EJB (By Stereotype)* da MVCASE. Como ilustrado nas Figura 3.21 e 3.22, o resultado é a criação de *entity beans* com suas duas interfaces *home* e *remote* e suas operações de negócio: `set()` e `get()`, `add()` e `remove()` na interface *remote*; e as operações do ciclo de vida: `create()` e de busca: `findByPrimaryKey()` localizadas na interface *home*. Os métodos do ciclo de vida podem ser redefinidos conforme os requisitos de criação e remoção de objetos de componentes. Para uma melhor visualização, organize os *entity beans* separando cada um deles em outro diagrama de classes, como ilustrado na Figura 3.22;
e) Adicionar em cada *entity bean*, por exemplo, `HotelBean`, as assinaturas das operações de negócio. Em relação às operações que envolvem busca, deve-se adicionar suas assinaturas nas interfaces *home*, conforme o modelo EJB.

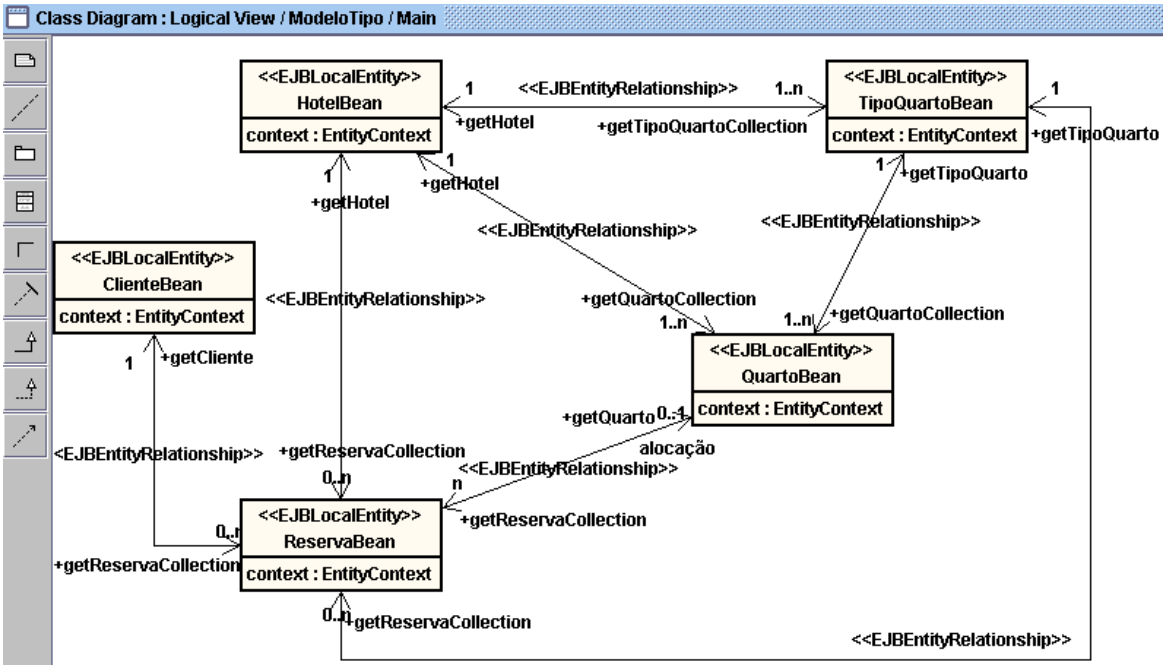


FIGURA 3.21: MODELO DE TIPO TRANSFORMADO EM MODELO DE ENTITY BEANS.

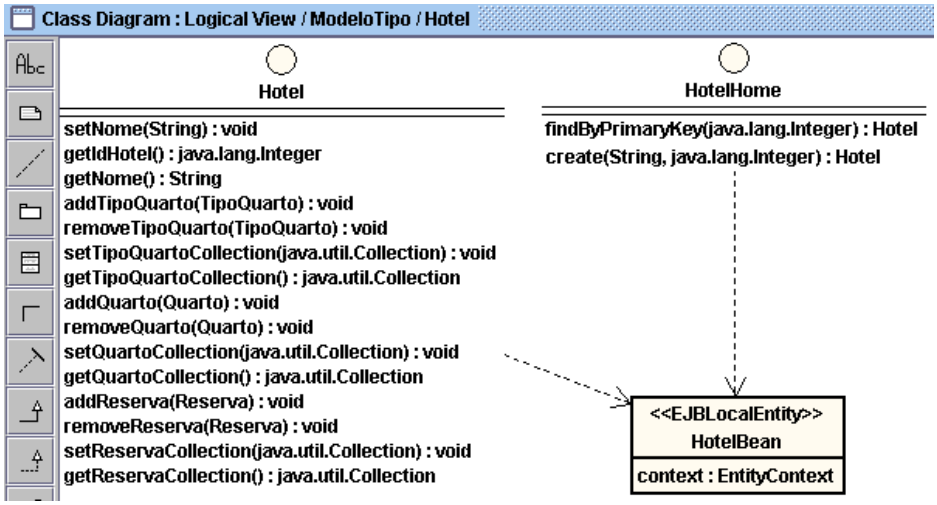


FIGURA 3.22: OPERAÇÕES DAS INTERFACES HOME E GERADAS PELA TRANSFORMAÇÃO.

Tarefa 2: Construir os componentes de sistema. A partir das especificações de interfaces e de componentes de sistema, o engenheiro deve:

a) Criar um novo diagrama de classes e adicionar uma classe para cada componente de sistema com o estereótipo `<<Session>>` pré-definido na ferramenta, indicando que as classes serão transformadas em componentes do tipo *session bean*;
b) Adicionar em cada classe *session* as assinaturas das operações de todas as interfaces oferecidas pelo componente de sistema (como definido na etapa de definição de componente), cujo estereótipo da operação é igual a `<<BusinessRule>>` pré-definido na ferramenta, indicando que são operações da regra de negócio;
c) Transformar as classes em objetos EJB s*ession* utilizando a funcionalidade *Transform Classes to EJB (By Stereotype)* da MVCASE. O resultado é a criação de *session beans* com suas duas interfaces *home* e *remote*, onde

na interface *home* encontra-se a operação do ciclo de vida `create()`; e na interface *remote* as operações da regra de negócio (Figura 3.23).

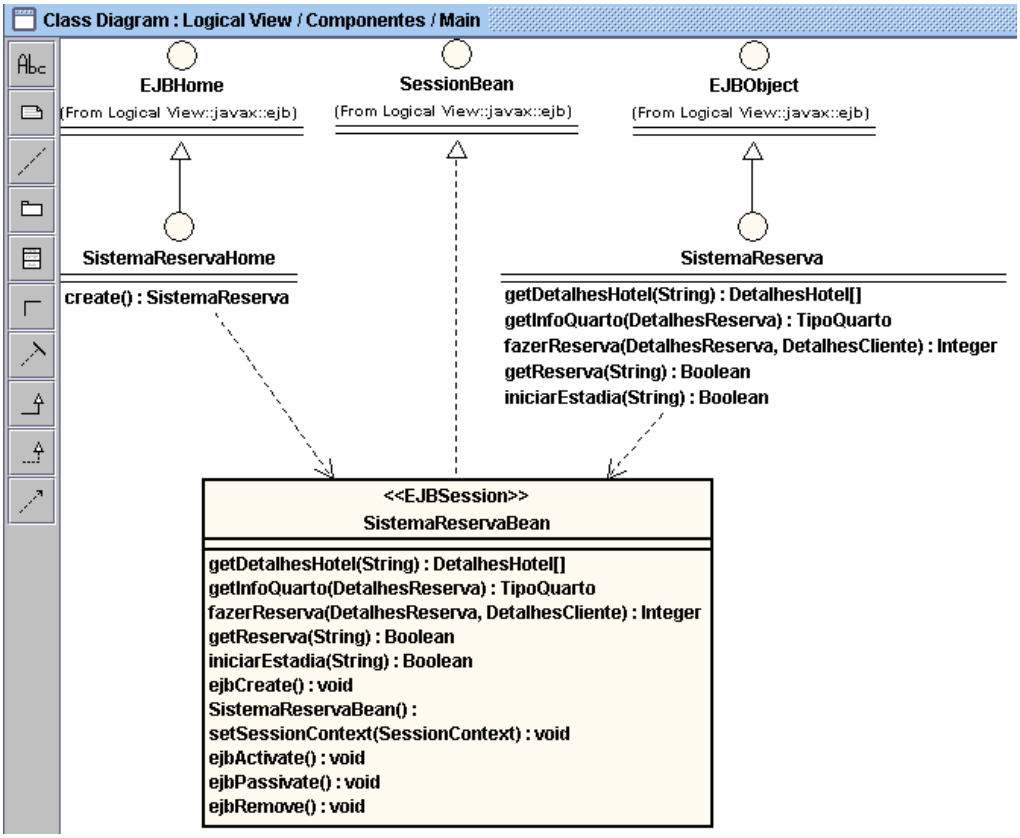


FIGURA 3.23: MODELO DE SESSION BEANS.

d) Criar um novo diagrama de classes para relacionar os componentes de sistema com as interfaces de negócio utilizadas. Isto é realizado para permitir que as operações das interfaces de sistema possam recuperar informações necessárias dos componentes de negócio. Assim, como ilustrado na Figura 3.24, deve-se distribuir os *session beans* (componentes de sistema) e as interfaces dos *entity beans* (interfaces de negócio) e criar uma referência estática por meio de uma associação do *session* para cada uma das interfaces de negócio (*home* e/ou *remote* do *entity*).

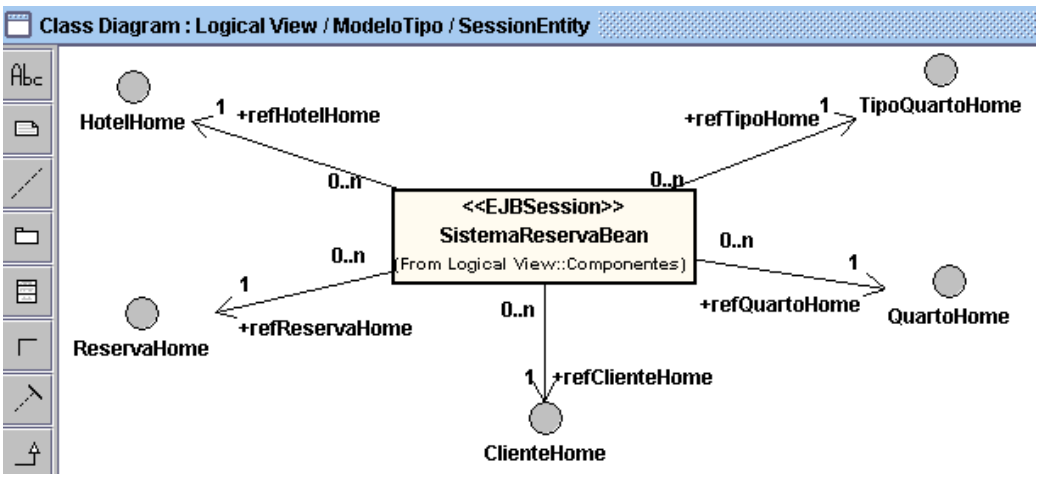


FIGURA 3.24: INTERFACES UTILIZADAS PELO SESSION BEAN.

Tarefa 3: Implementar as operações dos componentes. A partir das especificações em OCL das operações dos componentes, o engenheiro implementa as operações em Java, como ilustrado na Figura 3.25. É importante ressaltar que as operações de busca dos componentes de negócio são implementadas diretamente na própria interface *home* do componente. As operações dos componentes de sistema correspondem às operações definidas na etapa de especificação de componentes. Por exemplo, a operação `getInfoQuarto()` do componente de sistema `SistemaReserva` utiliza a operação `findDetalhes()` do componente de negócio `TipoQuarto` para buscar as informações sobre o quarto.

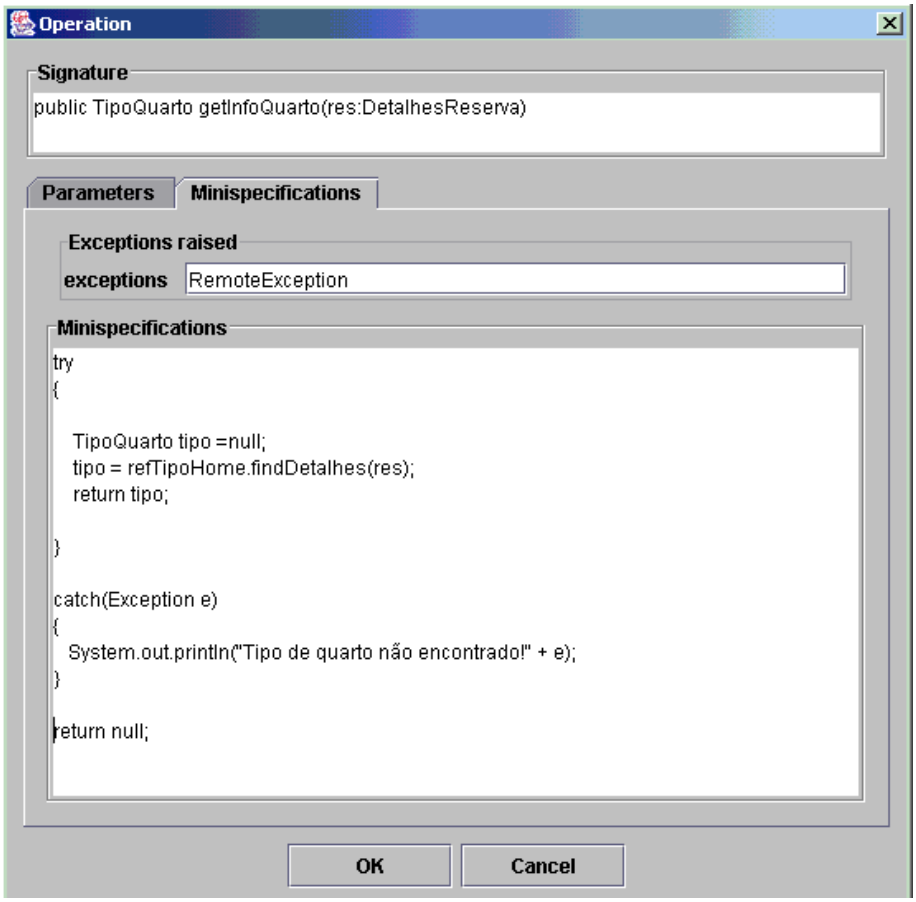


FIGURA 3.25: IMPLEMENTAÇÃO DAS OPERAÇÕES.

Tarefa 4: Definir o modelo de componentes. Similar ao modelo da arquitetura de componentes gerado na etapa 2 (Figura 3.12), este é um modelo de implementação utilizado para fazer o *deploy* dos componentes. Neste modelo são ilustrados os componentes na notação UML e as dependências existentes entre os componentes de sistema e os de negócio. Para criar o modelo de componentes, o engenheiro deve criar um novo diagrama de classes em *Component View*, distribuir os componentes deste pacote no diagrama e relacioná-los por meio de um relacionamento de dependência entre os componentes de sistemas e as interfaces de negócio utilizadas, como ilustrado na Figura 3.26.

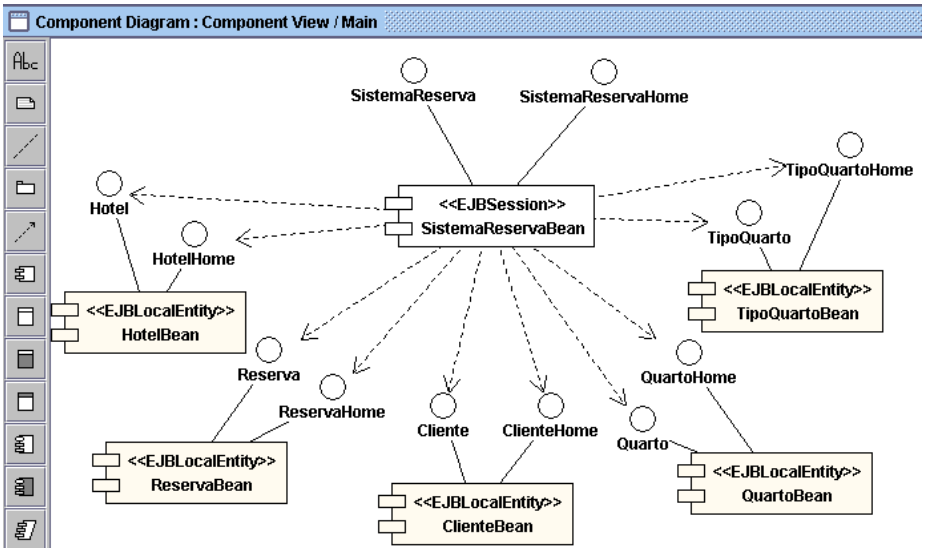


FIGURA 3.26: MODELO DE COMPONENTES.

Tarefa 5: Fazer o deploy dos componentes. Nesta tarefa, o engenheiro informa os componentes para o *deploy* e o JDBC do banco de dados em que as tabelas da aplicação serão criadas. O *EJB wizard* da MVCASE (Figura 3.27) gera e compila códigos Java criando os componentes *entity* e *session beans*; cria uma aplicação EJB e adiciona estes componentes no *EJB container* e depois realiza o *deploy* desta aplicação para o servidor, que neste trabalho é utilizado o J2EE (*Java 2 Enterprise Edition*) da *Sun Microsystems* (SUN, 2002a; 2002b), deixando-a pronta para ser utilizada pelo cliente.

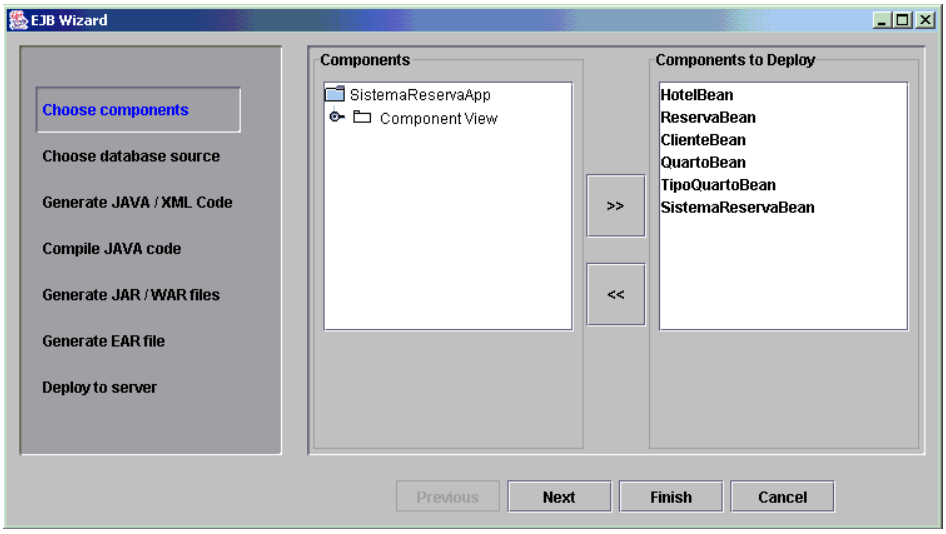


FIGURA 3.27: DEPLOY WIZARD.

3.6. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a abordagem mista ADComp que objetiva especificar e implementar componentes de uma forma direta, visando o desenvolvimento de um software reutilizável. A abordagem é dividida em três etapas, sendo que nas duas primeiras são realizadas a definição de requisitos e a definição de componentes utilizando, principalmente, as técnicas do *UML*

Components que integra técnicas do *Catalysis* e do RUP. As especificações e modelos gerados nestas etapas são elaboradas pelo engenheiro de software com o auxílio da ferramenta MVCASE que é utilizada em todas as etapas da ADComp. Na última etapa, as especificações de componentes são mapeadas para o modelo de componentes EJB, cujos componentes são implementados numa linguagem executável, gerando os componentes EJB da aplicação.

No próximo capítulo a abordagem ADComp é utilizada para especificar e implementar um produto de software baseado na reutilização de componentes denominada InstantFAQ — uma ferramenta baseada em componentes no domínio de gestão de FAQ e de Fale Conosco para WebApps. Os artefatos, os modelos e as especificações geradas na abordagem ADComp são ilustradas, bem como, as funcionalidades e a arquitetura da ferramenta.

Ferramenta InstantFAQ

4.1. CONSIDERAÇÕES INICIAIS

InstantFAQ é uma ferramenta Web de interatividade para gestão de FAQ e de Fale Conosco. Sua principal característica é ser baseada em componentes especificados e implementados segundo a abordagem de desenvolvimento ADComp. A ferramenta é baseada na arquitetura cliente/servidor, sendo que na camada servidora estão localizados os componentes responsáveis pelo controle e pela execução das requisições feitas pela camada cliente.

Este capítulo está organizado da seguinte forma: na seção 4.2 é apresentada a arquitetura geral da InstantFAQ. Na seção 4.3, são ilustrados os artefatos e as especificações geradas na abordagem ADComp no domínio do problema, e o mapeamento destas especificações para a implementação de componentes de acordo com o modelo EJB. Na seção 4.4, são descritos como os clientes JSP são adicionados à aplicação gerada pela ADComp, formando a InstantFAQ, e quais as funcionalidades oferecidas pela ferramenta. As considerações finais são apresentadas na seção 4.5.

4.2. ARQUITETURA

Diversas WebApps possuem ferramentas ou rotinas para a gestão de FAQ e de Fale Conosco, cujo objetivo é gerenciar e controlar as mensagens das perguntas mais freqüentes (FAQ) e da interação do administrador da WebApp com o usuário-online (Fale Conosco). Cada WebApp cadastrada na InstantFAQ deve ter suas próprias mensagens de FAQ, assuntos, origens de FAQ, mensagens de Fale Conosco, gestores e usuários, pois a InstantFAQ permite controlar FAQ e de Fale Conosco de diversas WebApps.

O controle de FAQ deve permitir ao administrador ou ao gestor da WebApp criar, editar, excluir e disponibilizar as mensagens para consulta dos usuários da aplicação. Este controle deve ser restrito somente aos gestores por meio de autenticação; e a apresentação das mensagens aos usuários deve ser apenas como consulta, não permitindo a edição das mesmas.

O controle de Fale Conosco deve permite aos usuários criar, consultar, excluir e enviar mensagens de dúvida, sugestão ou crítica, por exemplo, para o gestor da WebApp, que por sua vez deve respondê-las. A criação destas mensagens deve ser restrita somente aos usuários com permissões, pois todas as mensagens criadas

são armazenadas para seus respectivos usuários. Similarmente, as mensagens enviadas são respondidas somente pelos gestores da WebApp que são autenticados.

Para o domínio do problema em questão, os dois papéis desempenhados são pelo usuário-online e gestor. O gestor é o responsável por gerenciar o uso da InstantFAQ na WebApp, cadastrando informações relevantes para a gestão das mensagens de FAQ e de Fale Conosco. O usuário-online é o cliente da WebApp, a pessoa que "navega" pela aplicação, podendo apenas consultar as mensagens de FAQ, não tendo qualquer outro tipo de privilégio sobre as mesmas.

Toda mensagem de FAQ deve ser composta por uma pergunta e uma resposta, de um assunto e de uma origem de FAQ, por exemplo, se a mensagem originou de um e-mail, um fax, do próprio Fale Conosco da InstantFAQ. Em relação ao Fale Conosco, uma mensagem deve conter um texto e um assunto.

Para desenvolver a ferramenta InstantFAQ foi utilizada a abordagem de componentes reutilizáveis ADComp, que enfatiza o processo desde a definição de requisitos, a especificação e a implementação de componentes até a implantação da ferramenta. De acordo com o modelo de componentes EJB e com a etapa de definição de componentes da ADComp na qual são assumidos softwares em camadas, a arquitetura da InstantFAQ é composta pelas seguintes camadas: cliente, servidor e dados, conforme ilustrada na Figura 4.1. A camada cliente é subdividida em dois ambientes: (1) ambiente do usuário que permite aos usuários-online consultar as perguntas mais freqüentes (Consulta de FAQ) e enviar mensagens de Fale Conosco para os gestores (Editor de Fale Conosco); e (2) ambiente do gestor que permite criar e gerenciar a FAQ além de responder as mensagens de Fale Conosco enviadas pelos usuários e gerenciar outras informações para controle das mensagens (editores de assunto, origem, aplicação, gestor, usuário).

A camada servidora é responsável pelo gerenciamento e realização das operações requisitadas pela camada cliente. É nesta camada que se encontra o cliente JSP (páginas Web), localizadas no *Web Container*, responsável pela interface com usuários e gestores; e os beans (*session* e *entity*) responsáveis pela lógica do negócio e localizados no *EJB Container*. A camada de dados é responsável pelo armazenamento das mensagens de FAQ e de Fale Conosco da InstantFAQ, além de outras informações utilizadas para o controle e o gerenciamento da ferramenta.

As tecnologias envolvidas na implementação e implantação da InstantFAQ são:

- Java como linguagem de programação (GOSLING et. al., 1996);
- *JavaServer Pages* 2.0 (SUN, 2003; 2002a), HTML e *JavaScript* como tecnologias de apresentação;
- *Enterprise JavaBeans* 2.0 (SUN, 2002a; 2002b; 2002c; HAEFEL, 2001; SUN, 2000) como modelo de componente;

- *Cloudscape* 1.7 (SUN, 2004) como banco de dados relacional; e
- *Java 2 Enterprise Edition* (SUN, 2002a; 2002b; 2000) para Microsoft Windows como servidor de aplicações baseadas em componentes EJB.

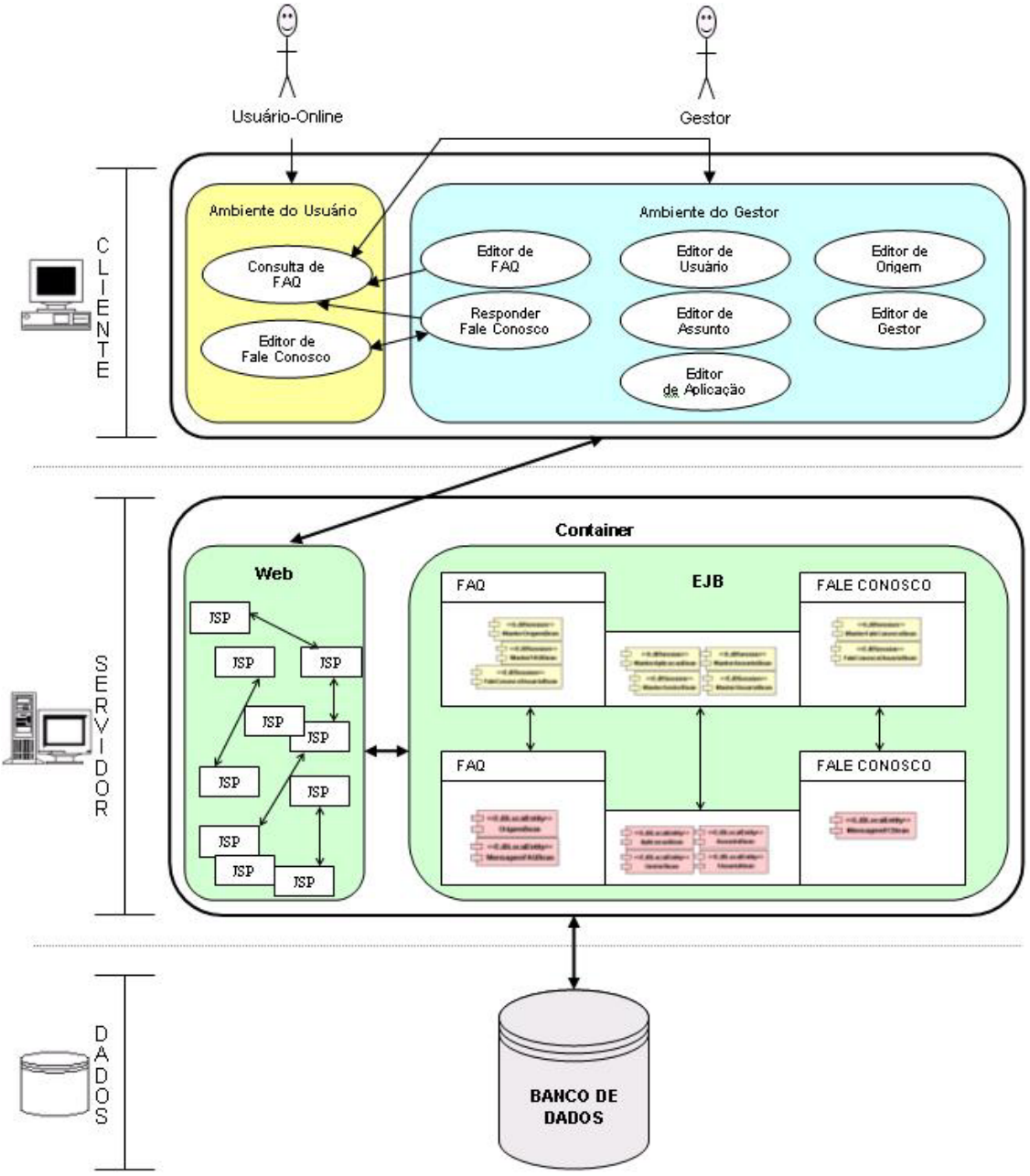


FIGURA 4.1: ARQUITETURA DA FERRAMENTA INSTANTFAQ.

4.3. DESENVOLVIMENTO DOS COMPONENTES

4.3.1. Etapa1: Definição de Requisitos

A partir da visão geral do domínio da InstantFAQ, nesta etapa são produzidos dois artefatos, o MCN e o MCU, que são utilizados como entrada para a etapa de especificação. Os principais conceitos da InstantFAQ são ilustrados no MCU da

Figura 4.2 e descritos a seguir:

1. `Aplicacao`: é o cadastro das WebApps que utilizam a InstantFAQ;
2. `Gestor`: é o cadastro dos gestores por WebApp. Os gestores são responsáveis pelas tarefas de gerenciamento de FAQ e de Fale Conosco, bem como cadastro de usuários, assuntos, origens de FAQ. Outra atribuição dos gestores é criar mensagens de FAQ e responder às mensagens de Fale Conosco enviadas pelos usuários da WebApp;
3. `Usuario`: é o cadastro de usuários por WebApp, os quais podem escrever mensagens de Fale Conosco e consultar as mensagens de FAQ que foram publicadas pelos gestores;
4. `MensagemFAQ`: é o cadastro das mensagens de FAQ por WebApp;
5. `MensagemFC`: é o cadastro das mensagens de Fale Conosco por WebApp enviadas pelos seus respectivos usuários;
6. `Assunto`: é o cadastro dos assuntos por WebApp a ser utilizado no cadastro das mensagens de FAQ e de Fale Conosco; e
7. `Origem`: é o cadastro das origens de FAQ por WebApp, isto é, qual o meio de comunicação que originou uma FAQ (correio, fórum, fax ou carta).

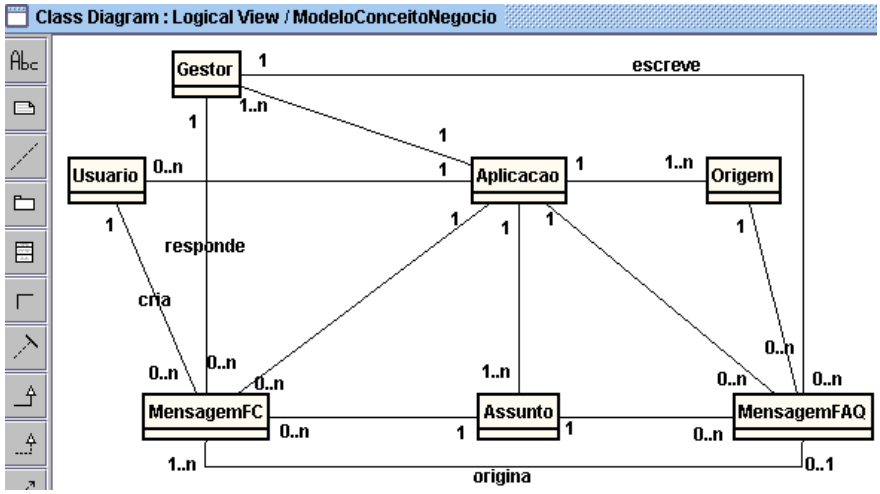


FIGURA 4.2: MODELO CONCEITUAL DE NEGÓCIO.

De acordo com os requisitos do problema, o modelo de casos de uso apresenta dois atores (1) Gestor e (2) Usuário-Online. Para o ator usuário-online foram identificados oito casos de uso, ilustrados na Figura 4.3, os quais correspondem às tarefas consultar as mensagens de FAQ, enviar e gerenciar mensagens de Fale Conosco mediante uma autenticação de usuário-online, pois cada usuário-online possui uma área exclusiva para enviar mensagens e consultar e atualizar suas próprias informações.

InstantFAQ permite dois tipos de autenticação tanto para o gestor quanto para o usuário-online (1) Com Cadastro Prévio e (2) Sem Cadastro Prévio. No primeiro, a WebApp que integra a InstantFAQ possui um cadastro de pessoas e o acesso é restrito a tais pessoas por meio de uma autenticação. Desta forma, a pessoa ao

utilizar a InstantFAQ não precisa passar pela autenticação da InstantFAQ. No segundo tipo de autenticação, como a WebApp não possui usuários cadastrados, a pessoa deve ser autenticada pela InstantFAQ por meio de um autocadastro efetuado pelos próprios usuários, isto é, se o usuário-online não possui *login* e senha, ele deve se cadastrar para depois utilizar as funcionalidades do Fale Conosco.

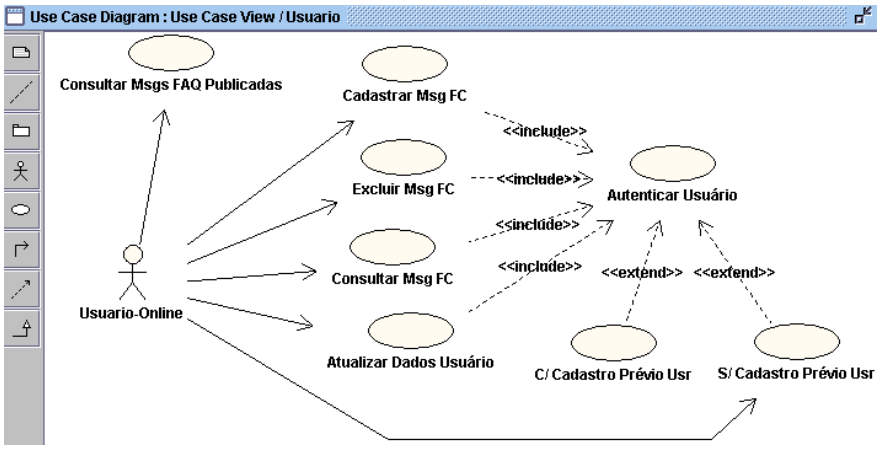


FIGURA 4.3: CASOS DE USO DO ATOR USUÁRIO-ONLINE.

Para o ator gestor, os casos de uso foram agrupados pelos cadastros aplicação, assunto, origem, mensagem FAQ, mensagem Fale Conosco, usuário e gestor, sendo que na Figura 4.4 são ilustrados os casos de uso referentes à Aplicação, Assunto e Origem; na Figura 4.5 os relativos à Gestor e Usuário; e na Figura 4.6 referentes à MensagemFAQ e MensagemFC.

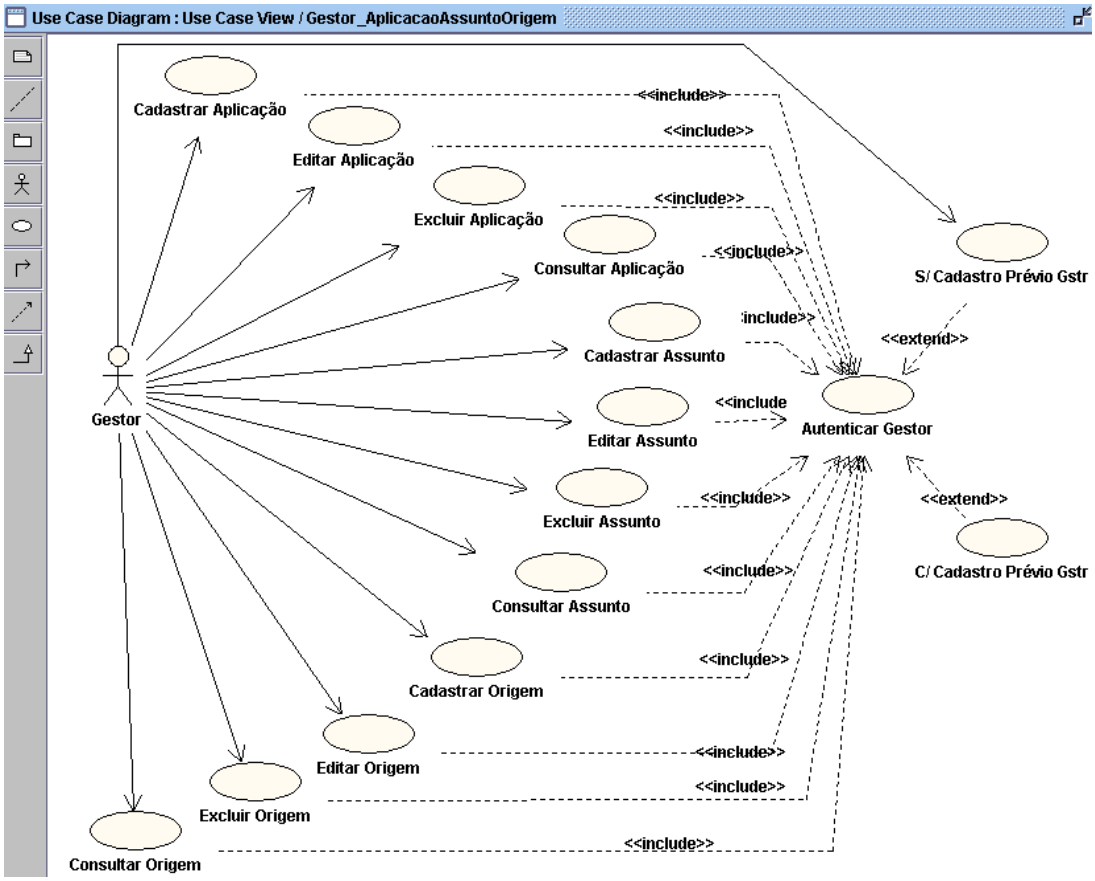


FIGURA 4.4: CASOS DE USO APLICAÇÃO, ASSUNTO E ORIGEM DO ATOR GESTOR.

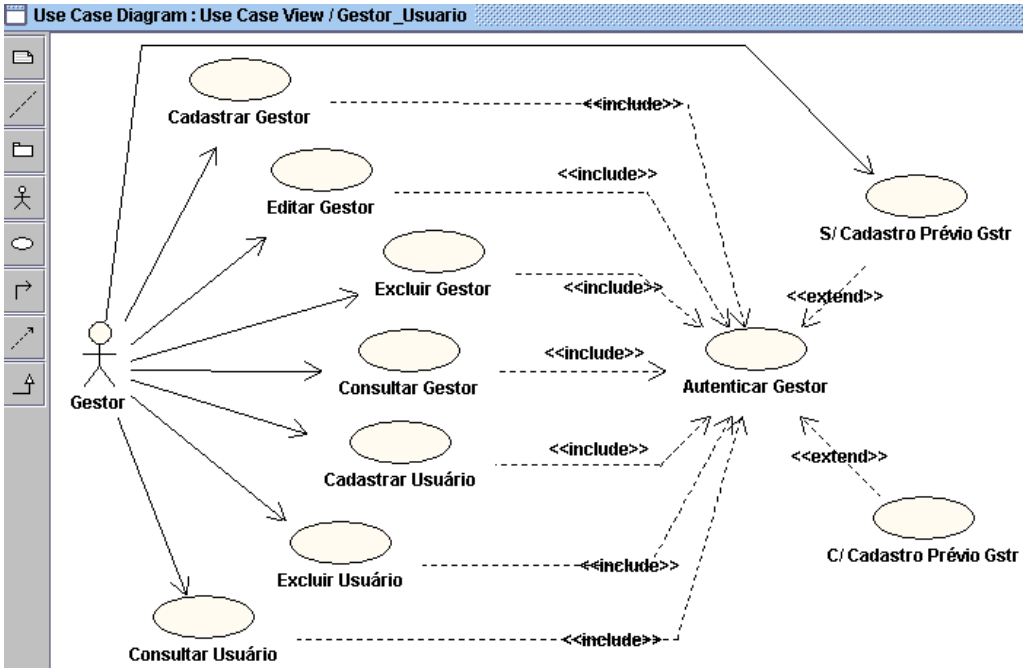


FIGURA 4.5: CASOS DE USO GESTOR E USUÁRIO DO ATOR GESTOR.

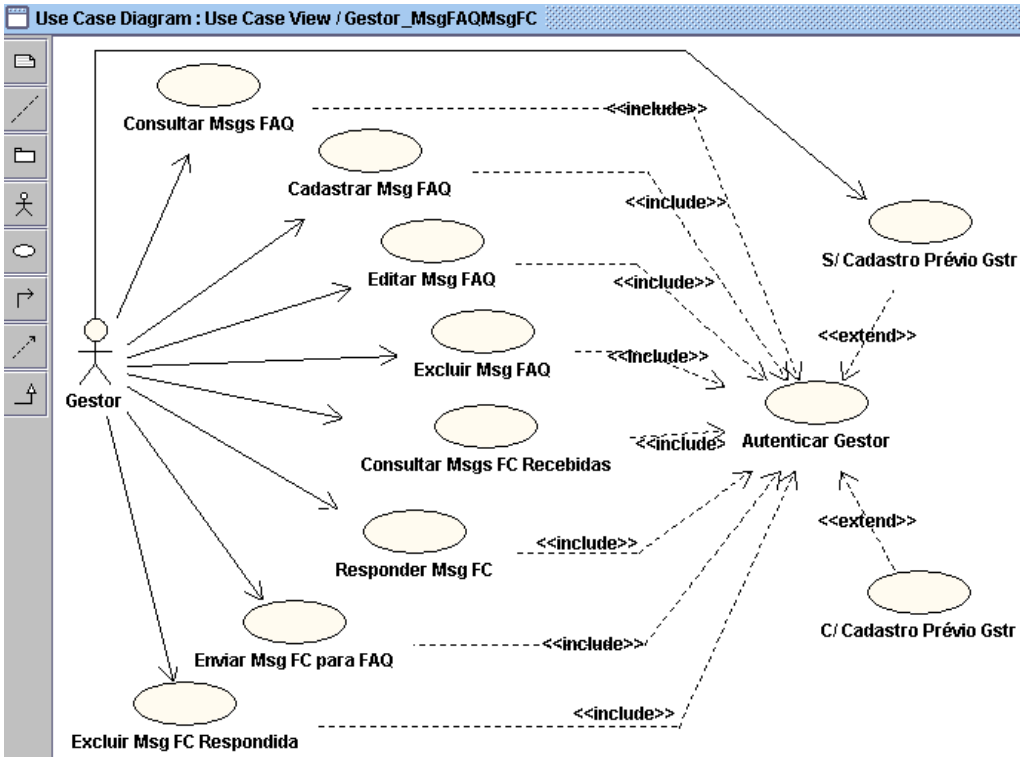


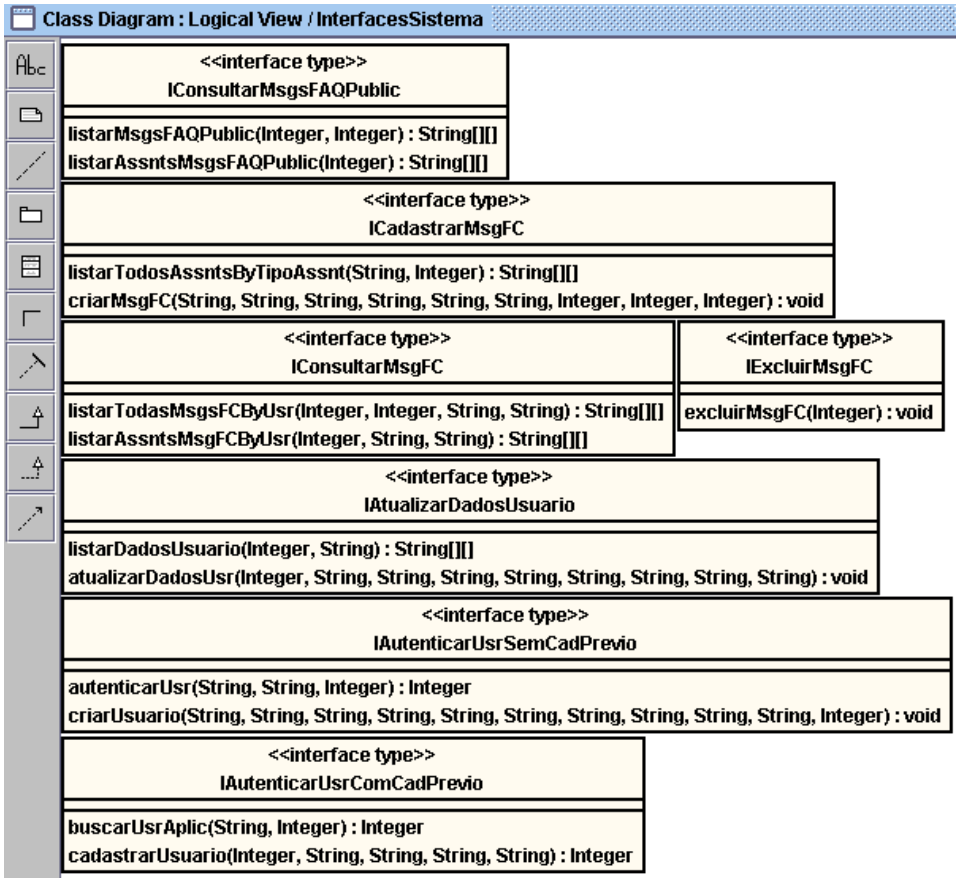
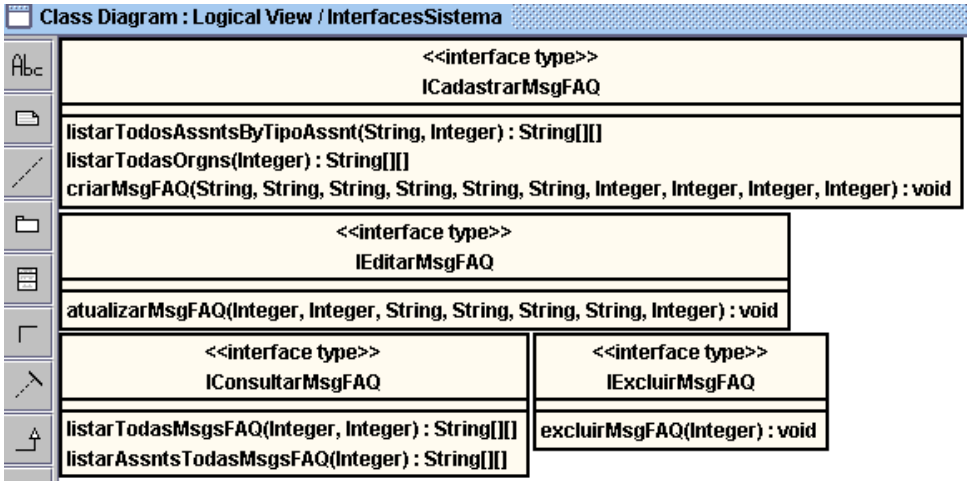
FIGURA 4.6: CASOS DE USO MENSAGEMFAQ E MENSAGEMFC DO ATOR GESTOR.

4.3.2. Etapa 2: Definição de Componentes

Na segunda etapa são especificados três grandes artefatos, conforme o processo proposto (Figura 3.2): (1) especificações da interface, (2) especificações de componente e (3) arquitetura dos componentes. Como a ADComp incentiva o desenvolvimento iterativo e evolutivo, nesta seção é apresentada a última versão das especificações e da arquitetura de componentes.

(1) ESPECIFICAÇÕES DA INTERFACE

Inicialmente, as interfaces de sistema são identificadas conforme os casos de uso do MCU da etapa anterior totalizando trinta e três (33) interfaces (Tabela 4.1), sendo que sete (7) estão relacionadas ao ator usuário-online e vinte e seis (26) ao ator gestor. As operações das interfaces são identificadas a partir dos passos dos casos de uso. Como ilustrado abaixo, algumas interfaces relacionadas ao gestor foram representadas somente por seus nomes, como `ICadastrarAplic`, mas todas as interfaces possuem operações relacionadas.

Ator	Interface de Sistema
Usuário-Online	
Gestor	

Gestor	ICadastrarAplic IEditarAplic IExcluirAplic IConsultarAplic	ICadastrarOrgm IEditarOrgm IExcluirOrgm IConsultarOrgm	ICadastrarAssnt IEditarAssnt IExcluirAssnt IConsultarAssnt
	ICadastrarGstr IEditarGstr IExcluirGstr IConsultarGstr	ICadastrarUsr IExcluirUsr IConsultarUsr	IConsultarMsgFC IResponderMsgFC IExcluirMsgFCRespondida

TABELA 4.1: INTERFACES DE SISTEMA.

Para facilitar a construção dos MI das interfaces de sistema ilustradas na Tabela 4.1 são definidas interfaces genéricas que agem como supertipo de outras interfaces, mantendo elementos comuns do MI. O agrupamento das interfaces ocorre por cadastros, como realizado nos casos de uso, pois apresentam conceitos relacionados. Por exemplo, para as interfaces `ICadastrarMsgFAQ`, `IEditarMsgFAQ` e `IConsultarMsgFAQ` é criada a interface genérica `IManterFAQ` que mantém tipos de informações comuns àquelas interfaces, como ilustrado na Figura 4.7. A interface `IExcluirMsgFAQ` não foi incluída, pois ela necessita apenas de duas informações, uma é o número da aplicação (`numAplic`) onde a mensagem de FAQ se encontra, e a outra é o número da mensagem de FAQ a ser excluída (`numMsgFAQ`).

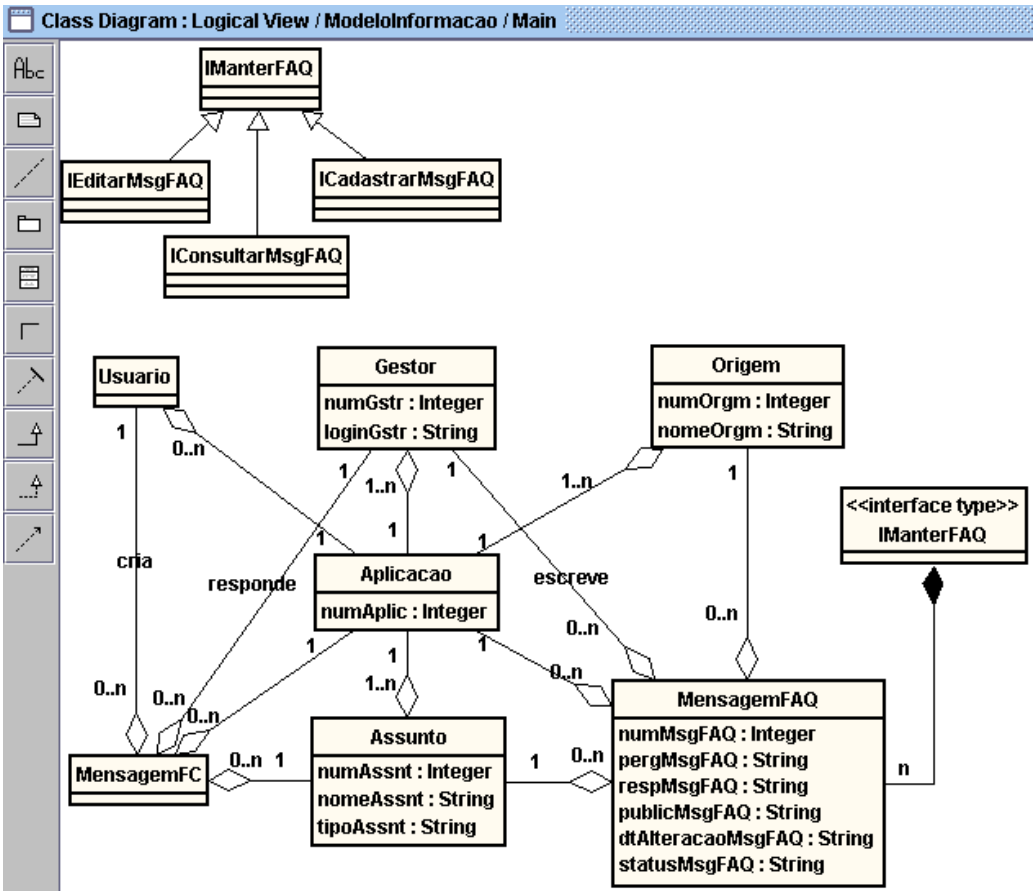


FIGURA 4.7: MODELO DE INFORMAÇÃO DA INTERFACE GENÉRICA IMANTERFAQ.

Na identificação das interfaces de negócio, foi utilizado o MCN da etapa anterior (Figura 4.2) para gerar um modelo de tipo contendo as informações representadas em cada tipo (Figura 4.8). Por meio do modelo de tipo são definidas interfaces de negócio, uma para cada tipo, representado no modelo de responsabilidade da interface (Figura 4.9). As interfaces de negócio definidas são: `IAplicacao`, `IAssunto`, `IOrigem`, `IGestor`, `IUsuario`, `IMensagemFAQ` e `IMensagemFC`, como ilustradas na Figura 4.10.

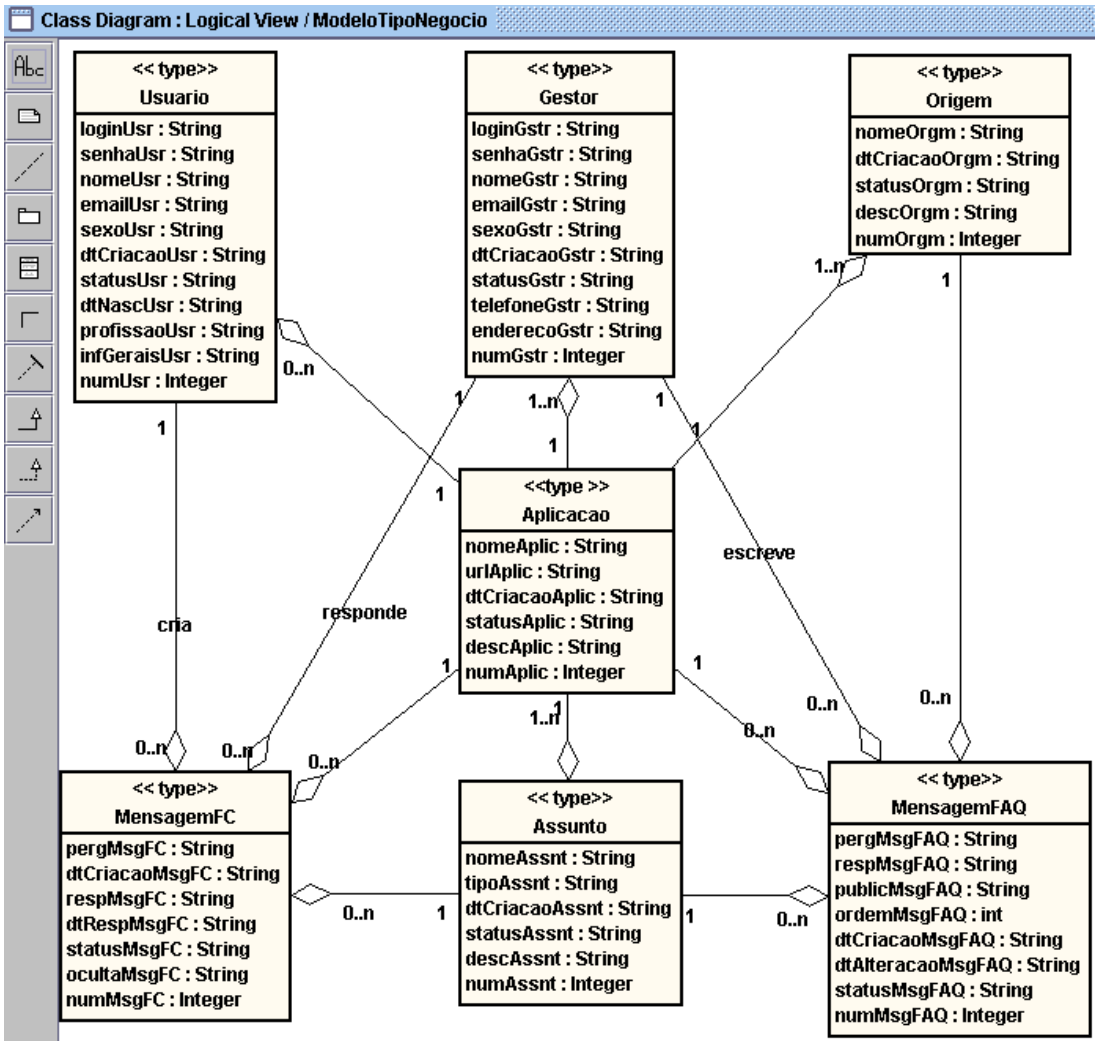


FIGURA 4.8: MODELO DE TIPO DE NEGÓCIO.

As operações das interfaces de negócio são identificadas por meio de modelos de interação. As interações informam em quais interfaces de negócio as operações da interface de sistema são implementadas, por exemplo, como ilustrado na Figura 4.11, a operação `listarMsgsFAQPublic` da interface de sistema `IConsultarMsgsFAQPublic` necessita de duas operações na interface de negócio `IMensagemFAQ`, uma se a *flag* do número do assunto for igual a zero (`numAssnt == 0`) devendo retornar todas as mensagens de FAQ classificadas pelos seus assuntos (`getMsgsPublicByAplic`); e outra se a *flag* for maior que zero (`numAssnt > 0`) devendo retornar somente mensagens de FAQ de um determinado assunto (`getMsgsPublicByAplicAssnt`).

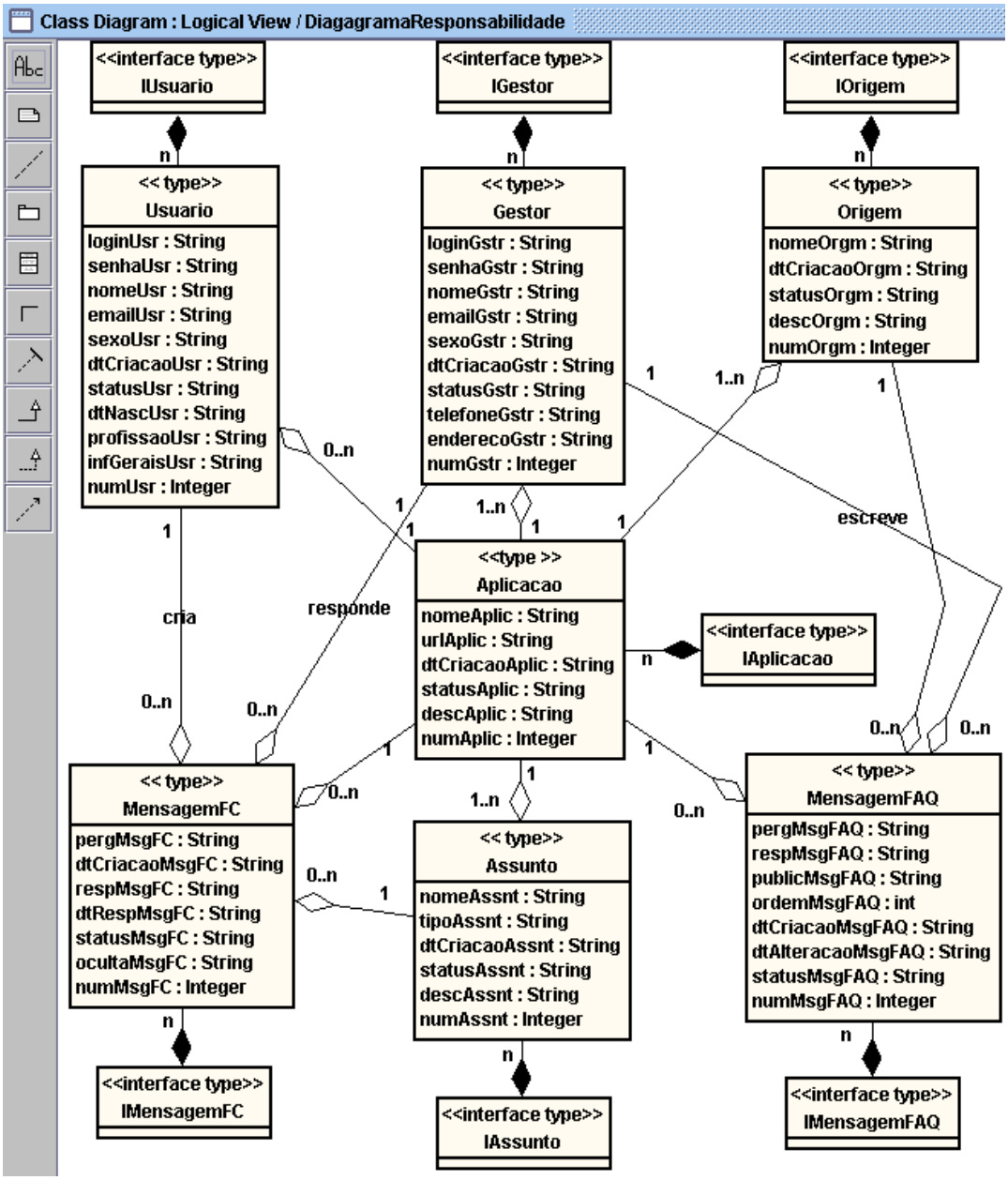


FIGURA 4.9: MODELO DE RESPONSABILIDADE.

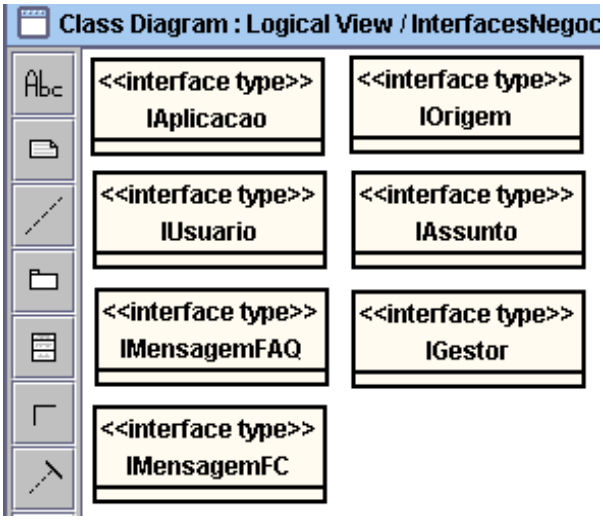


FIGURA 4.10: INTERFACES DE NEGÓCIO.

Assim, cada operação da interface de sistema passa por este processo, cujo resultado é a identificação das operações nas interfaces de negócio, como as operações das interfaces `IAplicacao` e `IMensagemFAQ` ilustradas na Figura 4.12.

Da mesma forma que as interfaces de sistema, as interfaces de negócio apresentam um MI individual. Como ilustrado na Figura 4.13, o MI da interface `IMensagemFAQ` está relacionada ao tipo `MensagemFAQ`.

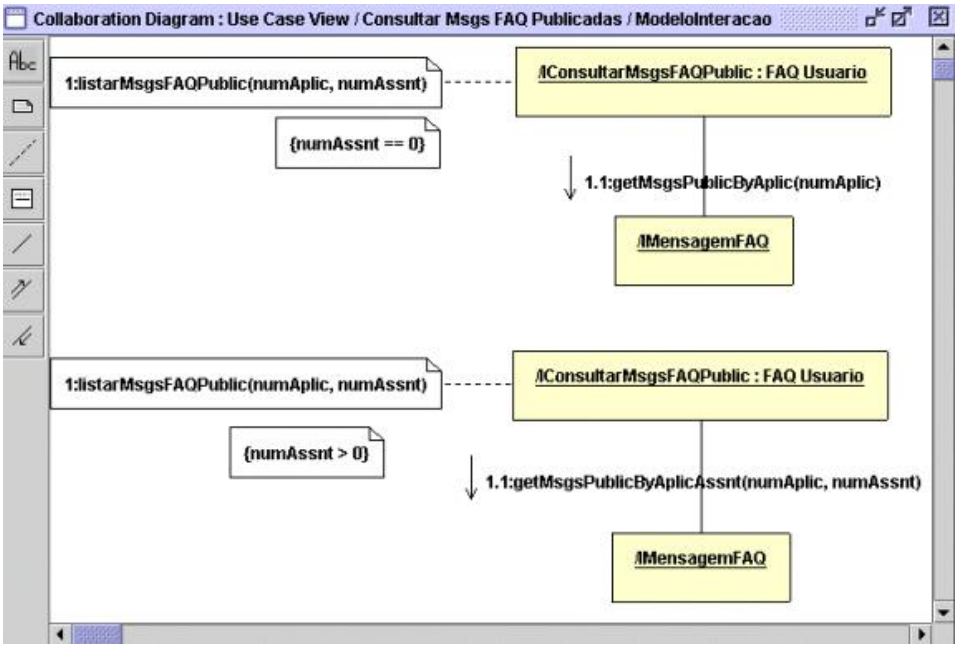


FIGURA 4.11: MODELO DE COLABORAÇÃO DA OPERAÇÃO LISTARMSGFAQPUBLIC.

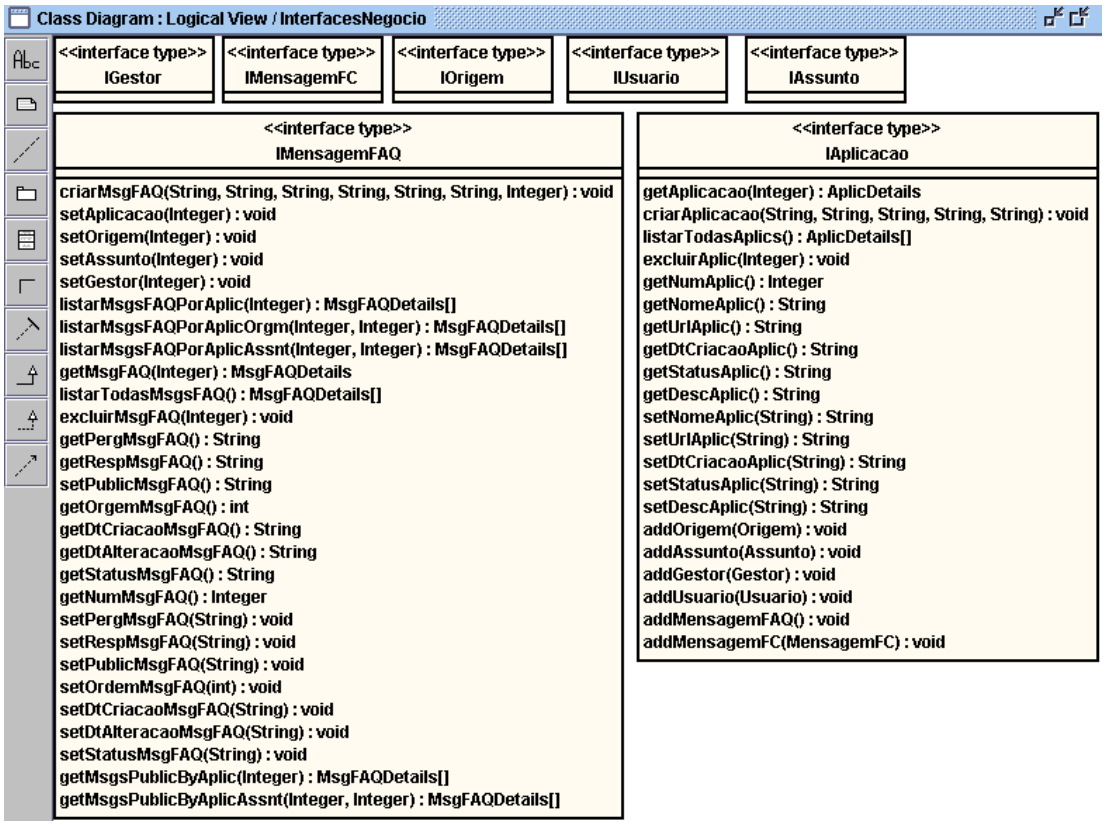


FIGURA 4.12: OPERAÇÕES DAS INTERFACES DE NEGÓCIO.

Além disto, cada operação na interface de negócio apresenta sua pré-condição e pós-condição escrita em OCL. Por exemplo, como ilustrado no Quadro 4.1, na pré-condição da operação `getMsgsPublicByAplic(aplic)`, o parâmetro de entrada `aplic` correspondente ao identificador de uma WebApp, deve existir em `Aplicação`. Na pós-condição, caso a pré-condição seja verdadeira, o resultado esperado são informações sobre mensagens de FAQ retornadas na estrutura `MsgFAQDetails`.

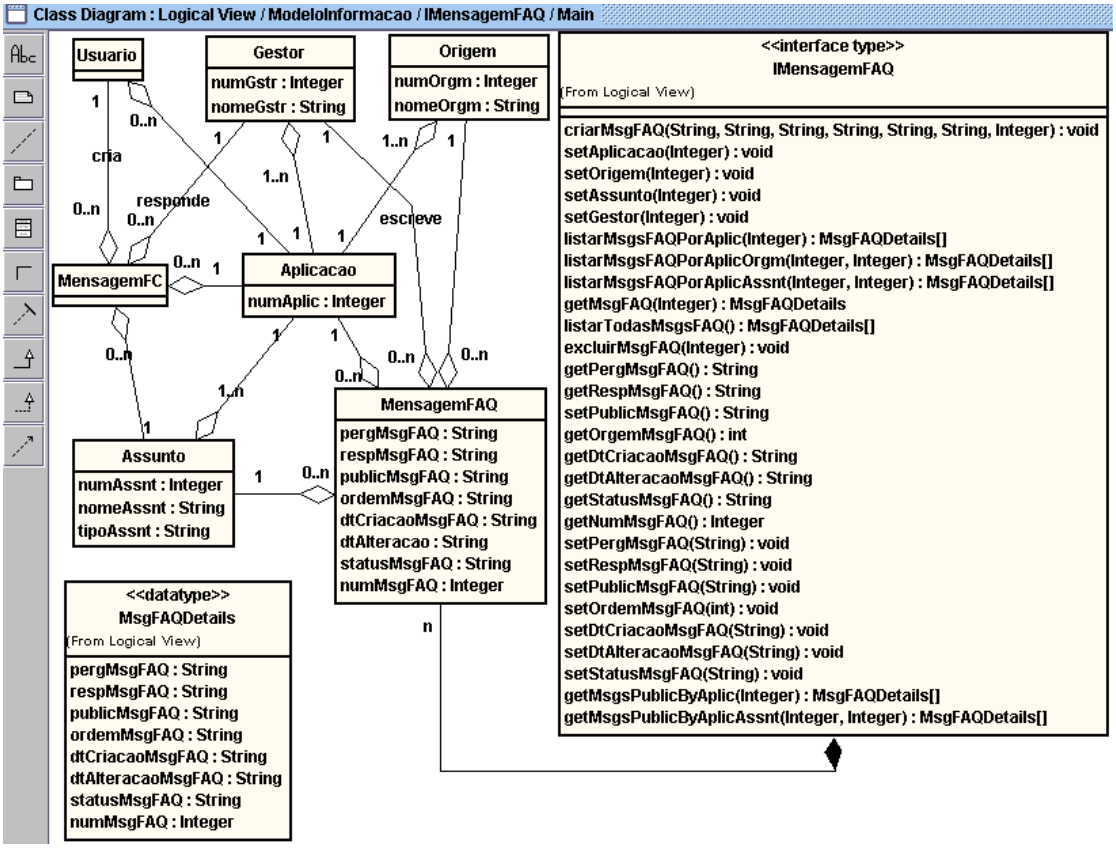


FIGURA 4.13: MODELO DE ESPECIFICAÇÃO DA INTERFACE IMENSAGEMFAQ.

```
context IMensagemFAQ::getMsgsPublicByAplic(
  in aplic: Integer): MsgFAQDetails[]
pre:
  -- aplic é um id de aplicação válida
    aplicação->exists(a | a.numAplic = aplic)
pos:
  -- as mensagens retornadas correspondem às mensagem de FAQ publicadas,
  -- isto é, com o atributo publicMsgFAQ = 'S'
  -- buscar as mensagens
    Let MsgFAQ = aplicação->select(m | m.aplicacao.numAplic = numAplic
    and m.publicMsgFAQ = 'S') in
  -- especifica o resultado
    result.pergMsgFAQ = MsgFAQ.pergMsgFAQ
    result.resMsgFAQ = MsgFAQ.resMsgFAQ
    result.publicMsgFAQ = MsgFAQ.publicMsgFAQ
    result.ordemMsgFAQ = MsgFAQ.ordemMsgFAQ
    result.dtCriacaoMsgFAQ = MsgFAQ.dtCriacaoMsgFAQ
    result.dtAlteracaoMsgFAQ = MsgFAQ.dtAlteracaoMsgFAQ
    result.statusMsgFAQ = MsgFAQ.statusMsgFAQ
    result.numMsgFAQ = MsgFAQ.numMsgFAQ
```

QUADRO 4.1: PRÉ-CONDIÇÃO E PÓS-CONDIÇÃO DE UMA OPERAÇÃO.

(2) ESPECIFICAÇÕES E ARQUITETURA DOS COMPONENTES

Uma especificação de componente é a construção de partes de uma arquitetura de componente, isto é, a especificação indica para cada componente o conjunto de interfaces que ele deve oferecer e utilizar. Além disto, a especificação define restrições requeridas pelo componente entre suas interfaces e qualquer restrições na implementação de suas operações. A arquitetura de componentes reúne estas especificações individuais numa única arquitetura, exibindo as dependências entre todos os componentes.

Na definição de componentes foram identificados nove (9) componentes de sistema (Figura 4.14) e sete (7) componentes de negócio (Figura 4.15). Os componentes de sistema podem oferecer uma ou mais interfaces de sistema, o que não ocorre com os componentes de negócio que apresentam somente uma única interface de negócio. Por exemplo, na Figura 4.14, o componente de sistema `Fale Conosco Usuario` empacota seis interfaces de sistema, pois estas estão fortemente interligadas e por isso devem estar no mesmo componente; e o componente de negócio `MensagemFAQ` empacota uma única interface.

Além das interfaces oferecidas pelos componentes, as especificações devem exibir as interfaces utilizadas e posteriormente definir as restrições entre suas interfaces oferecidas e utilizadas. Como ilustrado na Figura 4.16 o componente de sistema `Fale Conosco Usuario` utiliza quatro (4) interfaces de negócio (`IAplicacao`, `IMensagemFC`, `IUsuario` e `IAssunto`). E as restrições entre suas interfaces são ilustradas no Quadro 4.2.

```
context Fale Conosco Usuario
-- restrições entre interfaces oferecidas
ICadastrarMsgFC::aplicacao = IConsultarMsgFC::aplicacao
ICadastrarMsgFC::assunto = IConsultarMsgFC::assunto
ICadastrarMsgFC::mensagemFC = IConsultarMsgFC::mensagemFC
ICadastrarMsgFC::usuario = IConsultarMsgFC::usuario

ICadastrarMsgFC::aplicacao = IExcluirMsgFC::aplicacao
ICadastrarMsgFC::assunto = IExcluirMsgFC::assunto
ICadastrarMsgFC::mensagemFC = IExcluirMsgFC::mensagemFC
ICadastrarMsgFC::usuario = IExcluirMsgFC::usuario

IConsultarMsgFC::aplicacao = IExcluirMsgFC::aplicacao
IConsultarMsgFC::assunto = IExcluirMsgFC::assunto
IConsultarMsgFC::mensagemFC = IExcluirMsgFC::mensagemFC
IConsultarMsgFC::usuario = IExcluirMsgFC::usuario

--restrições entre interfaces oferecidas e interfaces utilizadas
ICadastrarMsgFC::aplicacao = IAplicacao::aplicacao
ICadastrarMsgFC::assunto = IAssunto::assunto
ICadastrarMsgFC::mensagemFC = IMensagemFC::mensagemFC
ICadastrarMsgFC::usuario = IUsuario::usuario

IConsultarMsgFC::aplicacao = IAplicacao::aplicacao
IConsultarMsgFC::assunto = IAssunto::assunto
IConsultarMsgFC::mensagemFC = IMensagemFC::mensagemFC
IConsultarMsgFC::usuario = IUsuario::usuario

IExcluirMsgFC::aplicacao = IAplicacao::aplicacao
IExcluirMsgFC::mensagemFC = IMensagemFC::mensagemFC
IExcluirMsgFC::usuario = IUsuario::usuario
```

QUADRO 4.2: RESTRIÇÕES ENTRE AS INTERFACES OFERECIDAS E AS UTILIZADAS.

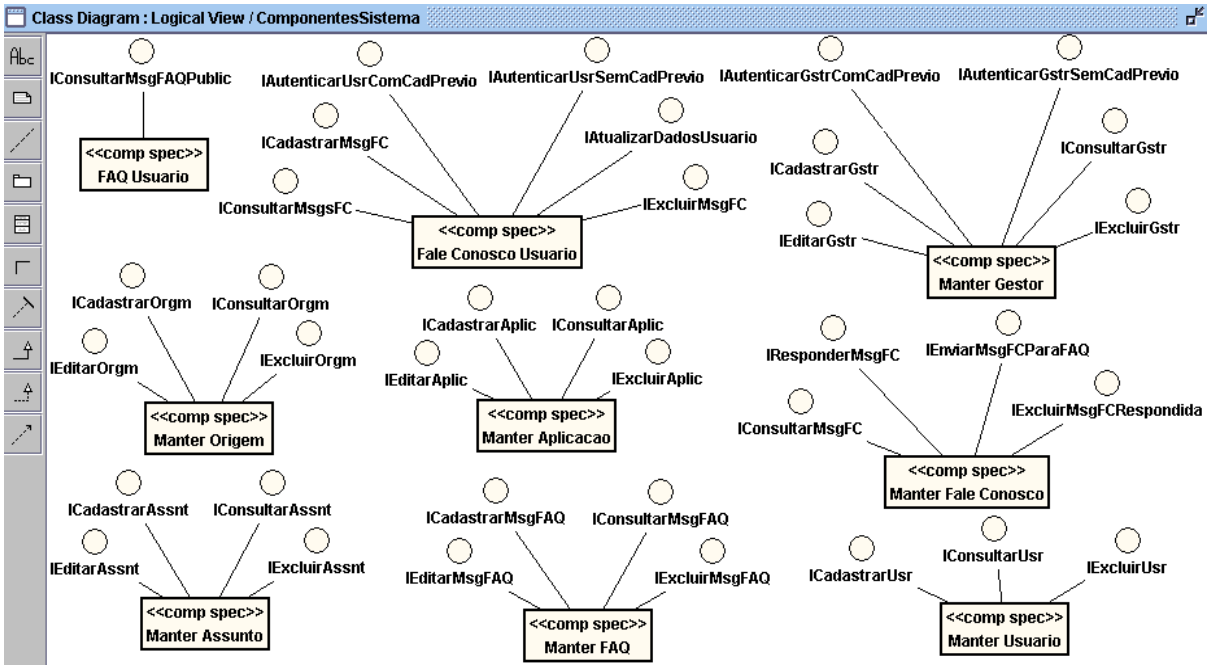


FIGURA 4.14: MODELO DE COMPONENTES DE SISTEMA.

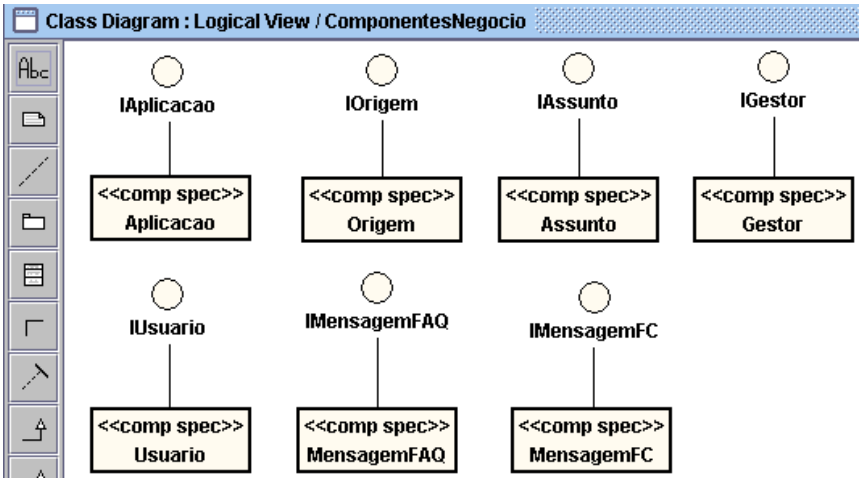


FIGURA 4.15: MODELO DE COMPONENTES DE NEGÓCIO.

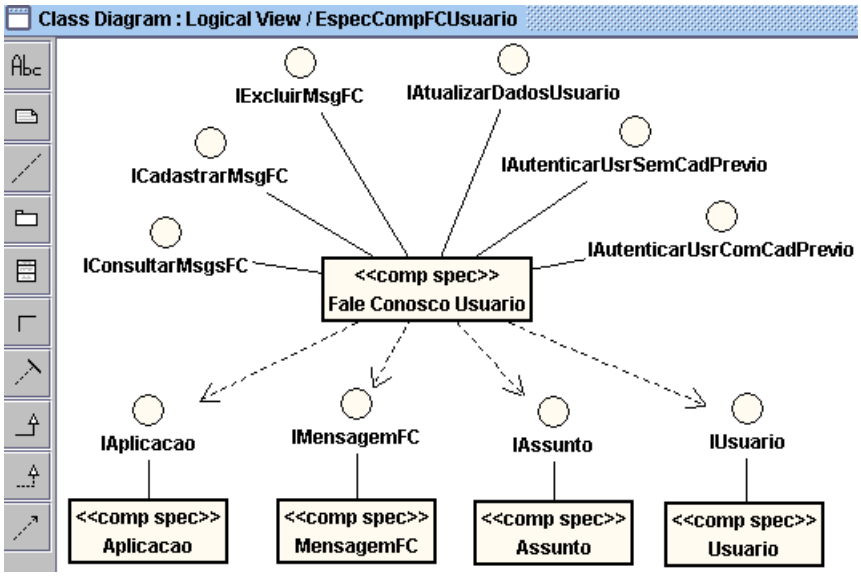


FIGURA 4.16: MODELO DA ESPECIFICAÇÃO DO FALE CONOSCO USUARIO.

Definidas as especificações de todos os componentes, o modelo da arquitetura de componente é reunida e ilustrada as dependências entre todos os componentes e suas interfaces, como observado na Figura 4.17. Para facilitar a visualização geral da arquitetura, os componentes foram representados sem suas interfaces oferecidas.

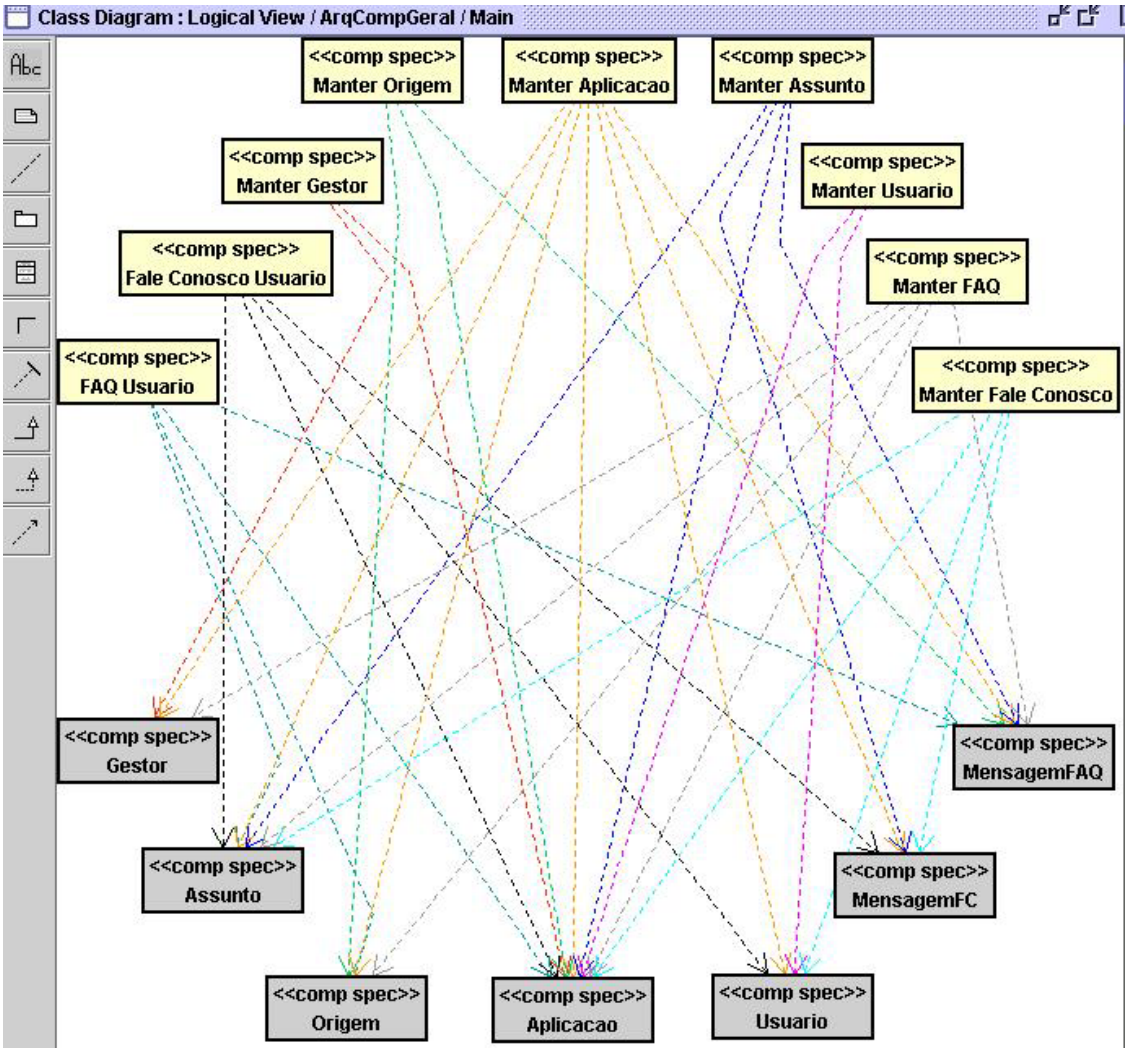


FIGURA 4.17: MODELO DA ARQUITETURA DOS COMPONENTES.

4.3.3. Etapa 3: Implementação de Componentes

Os componentes especificados são implementados utilizando o modelo EJB, cujos componentes de negócio são mapeados para *entity beans* e os componentes de sistema para *session beans*. Nesta etapa o modelo de tipo de negócio auxilia na criação do diagrama de classes com atributos tipificados, como ilustrado na Figura 4.18. O estereótipo destas classes são alterados de *<<type>>* para *<<LocalEntity>>* para que as classes sejam transformadas em componentes de negócio (*entity beans*).

Realizada a transformação, cada *entity* com suas interfaces *home* e *remote* são separados num diagrama de classes para melhor visualização de suas operações,

como na Figura 4.19 para o *bean* `MensagemFAQBean` e suas interfaces *remote* `MensagemFAQ` e *home* `MensagemFAQHome`.

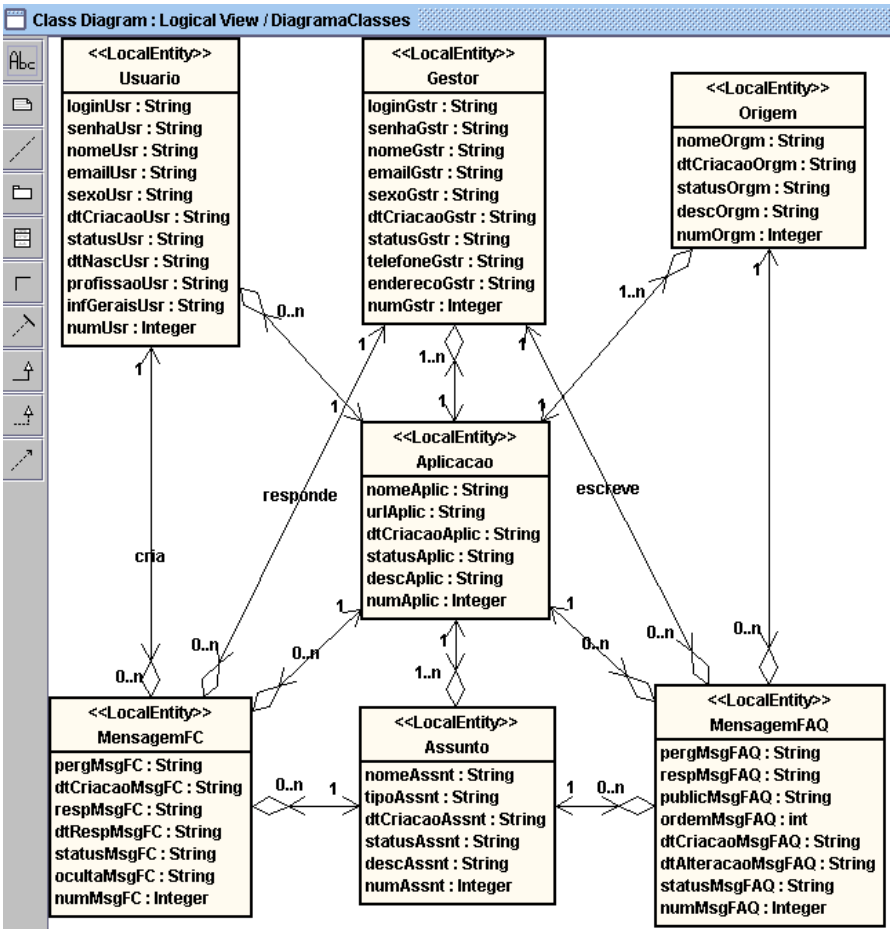


FIGURA 4.18: DIAGRAMA DE CLASSES PARA OS COMPONENTES DE NEGÓCIO.

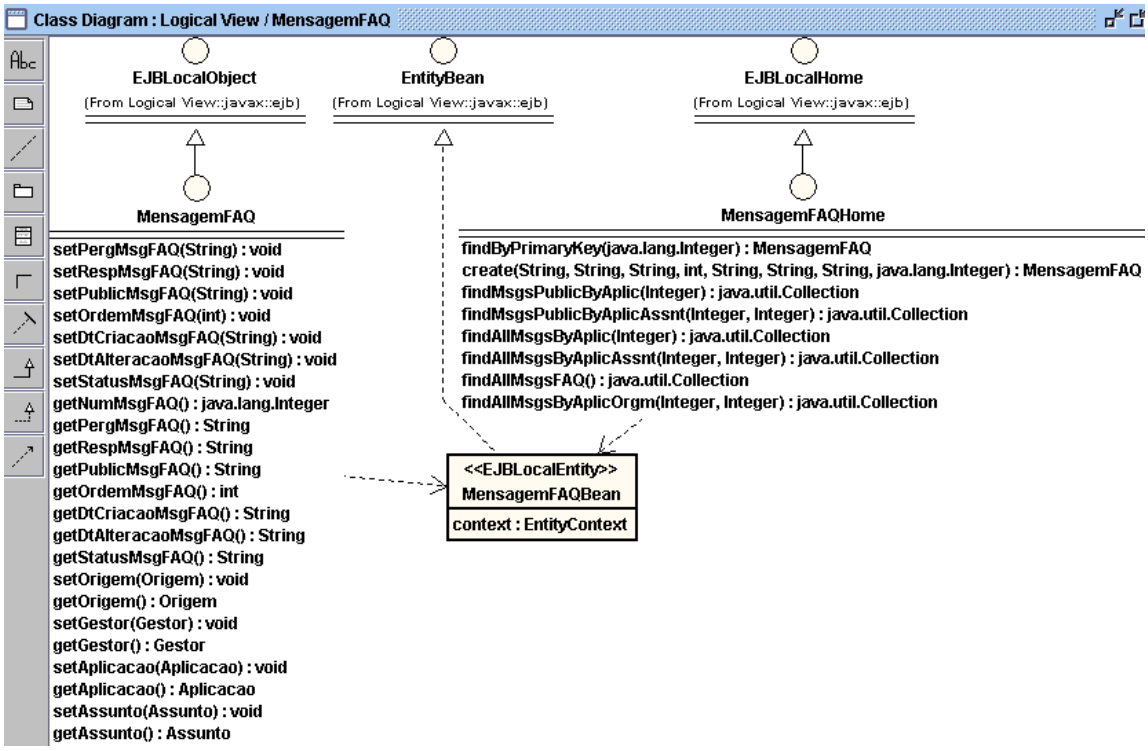


FIGURA 4.19: MODELO DO ENTITY BEAN MENSAGEMFAQ COM SUAS INTERFACES.

Os componentes de sistema são transformados em *sessions bean* do tipo *stateful*. Na Figura 4.20 são ilustrados os componentes de sistema relacionados às suas respectivas interfaces de negócio utilizadas (notação “lolipop”).

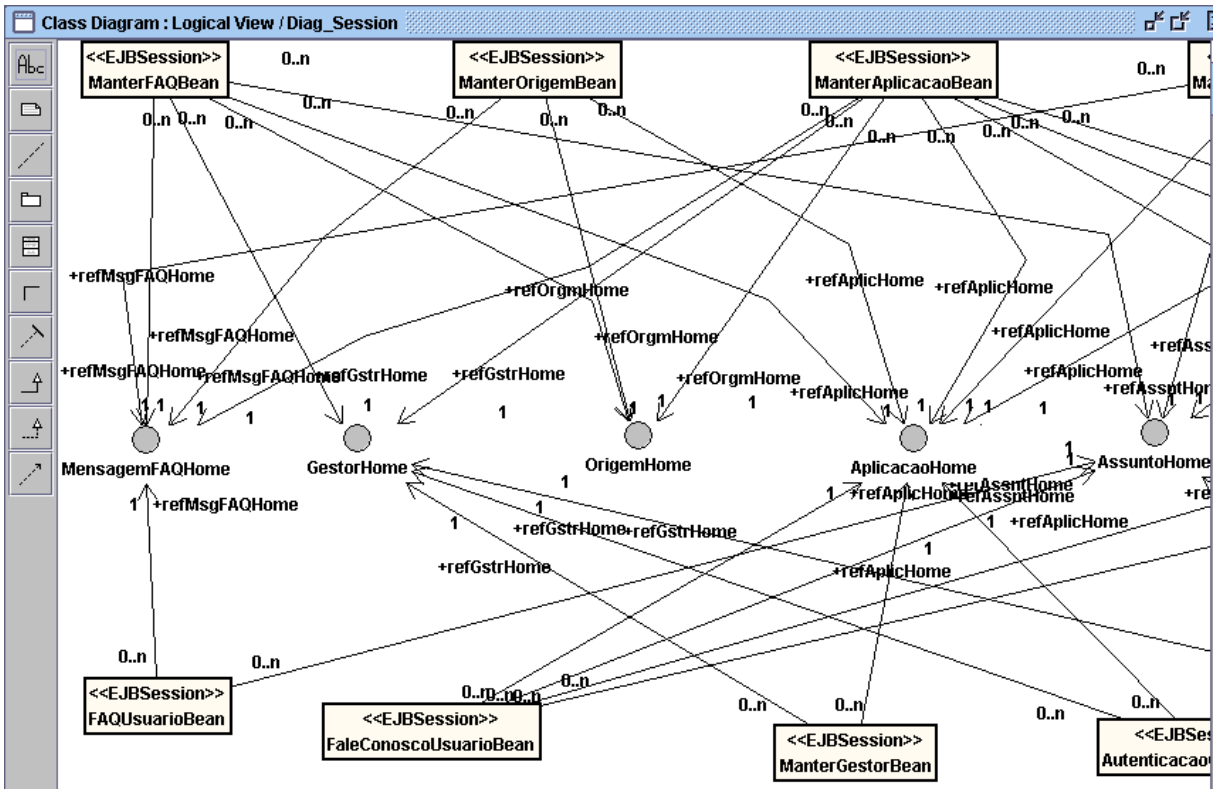


FIGURA 4.20: MODELO DE COMPONENTES DE SISTEMA E INTERFACES.

Definidos os *sessions bean*, como ilustrado na Figura 4.21, a implementação das operações dos componentes de sistema é realizada nos seus respectivos *session bean* (não nas interfaces *home* ou *remote*), como `FAQUsuarioBean`, `ManterFAQBean` e `ManterAplicacaoBean`. Somente as assinaturas das operações são escritas nas interfaces. Da mesma forma, as operações dos componentes de negócio seguem a mesma regra, isto é, a implementação é feita no *entity bean* — `MensagemFAQBean`, `AplicacaoBean` e `AssuntoBean` — e nas interfaces há somente as assinaturas das operações. As operações de busca — `finds()` — são implementadas na própria interface *home*.

Finalizada a implementação, a arquitetura ou modelo de componentes é desenvolvido na visão *Component View*, apresentando as interdependências entre os mesmos (Figura 4.22).

A última tarefa é fazer o *deploy* dos componentes utilizando a opção *Deploy Wizard* da MVCase, que gera uma aplicação EJB contendo o *EJB container* todos os componentes implementados, bem como suas interfaces, além de implantar a aplicação e criar as tabelas necessárias no banco de dados escolhido. Finalizado o *deploy*, é necessário adicionar o cliente JSP no *Web container* dentro da aplicação EJB, resultando no desenvolvimento da ferramenta InstantFAQ, cujas funcionalidades são descritas a seguir.

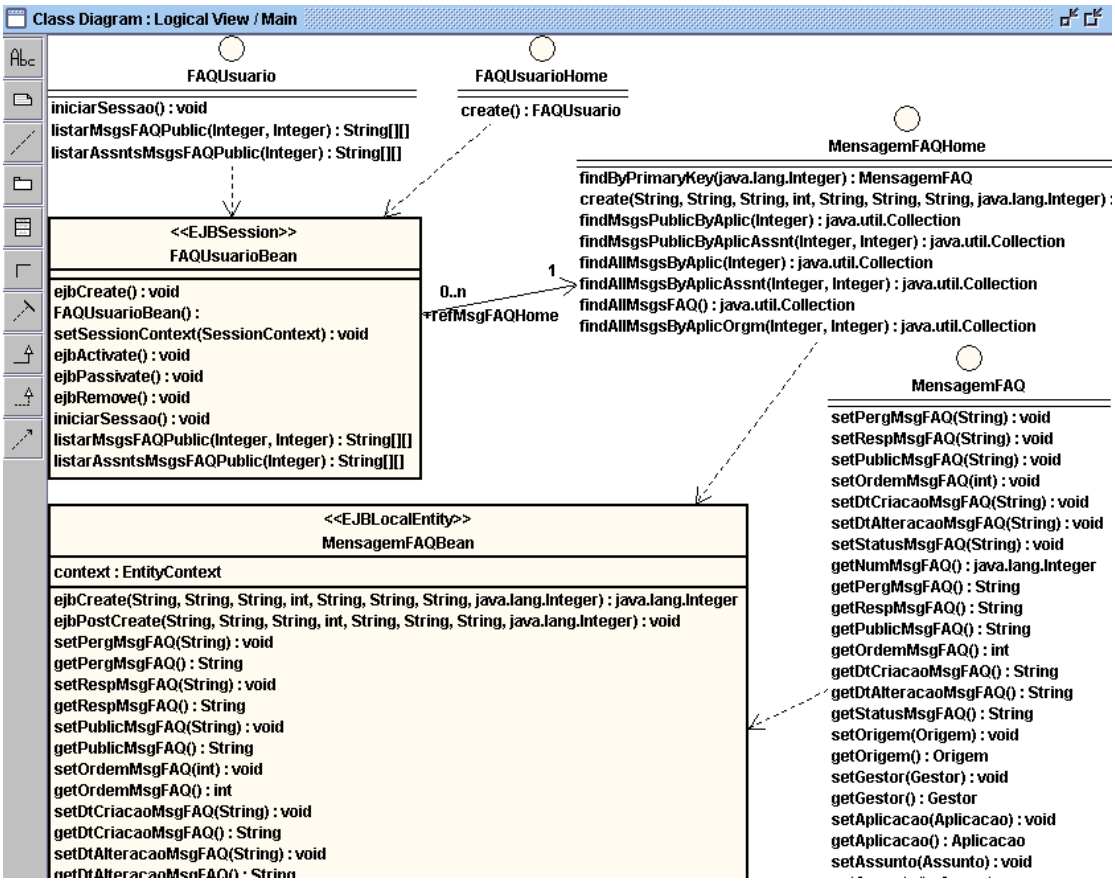


FIGURA 4.21: IMPLEMENTAÇÃO DAS OPERAÇÕES DOS COMPONENTES.

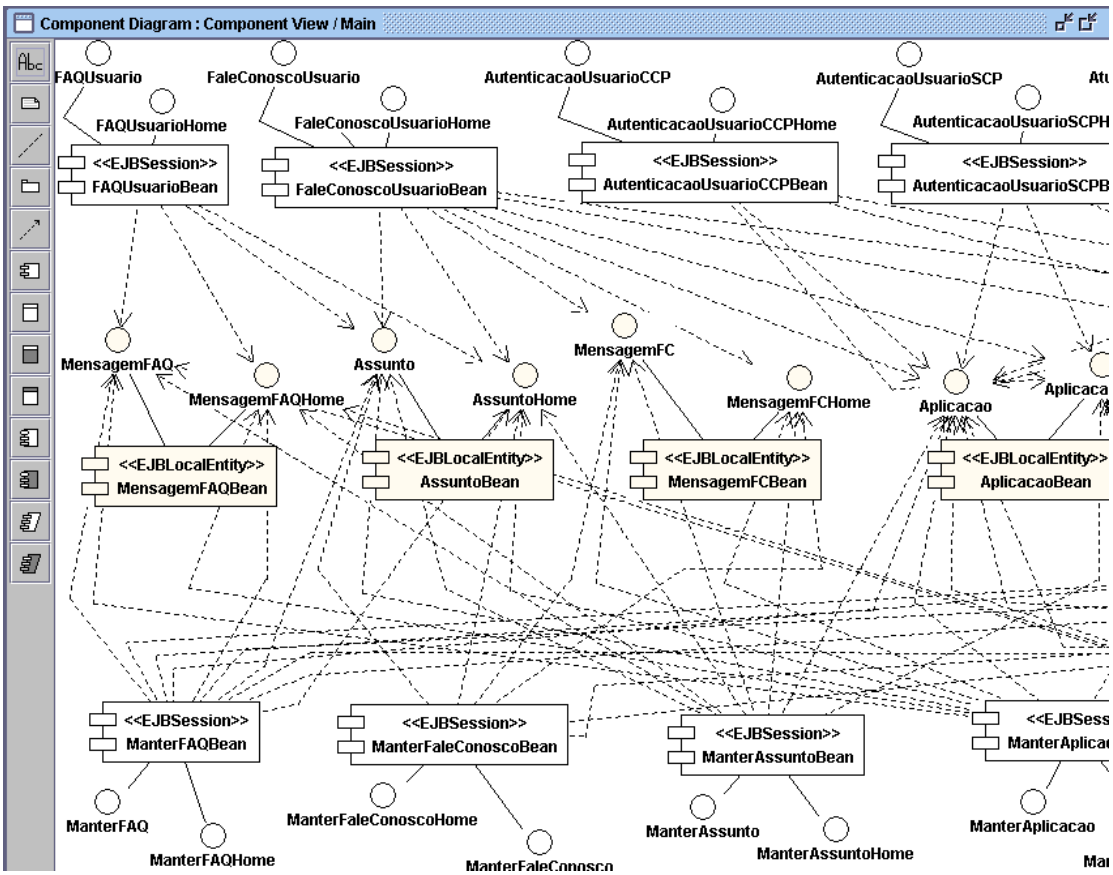


FIGURA 4.22: MODELO DE COMPONENTES.

4.4. FUNCIONALIDADES DA INSTANTFAQ

Por meio da ferramenta *Deploytool* (Figura 4.23) disponibilizada pelo servidor J2EE, a aplicação gerada pela MVCASE é modificada para incluir o cliente JSP no *Web container*. Novamente faz-se o *deploy* da aplicação, resultando na implantação da InstantFAQ. Na versão atual da MVCASE (MVCASE, 2004) o engenheiro realiza esta tarefa sem a necessidade de manipular a ferramenta *Deploytool* do J2EE e além disto, pode ser utilizado o servidor EJB JBoss ao invés do J2EE.

Para os usuários da ferramenta são oferecidas duas funcionalidades (1) FAQ e (2) Fale Conosco. A funcionalidade FAQ do usuário-online, ilustrado na Figura 4.24, permite aos usuários consultar as mensagens de FAQ, classificadas por assunto, e publicadas pelos gestores da WebApp. A funcionalidade Fale Conosco do usuário-online, ilustrado na Figura 4.25, oferece uma forma de comunicação dos usuários com os gestores da WebApp, onde os usuários enviam suas mensagens de crítica, sugestão ou sobre algum outro assunto definido para a WebApp (Figura 4.26), cujas mensagens são mantidas como histórico para cada usuário-online.

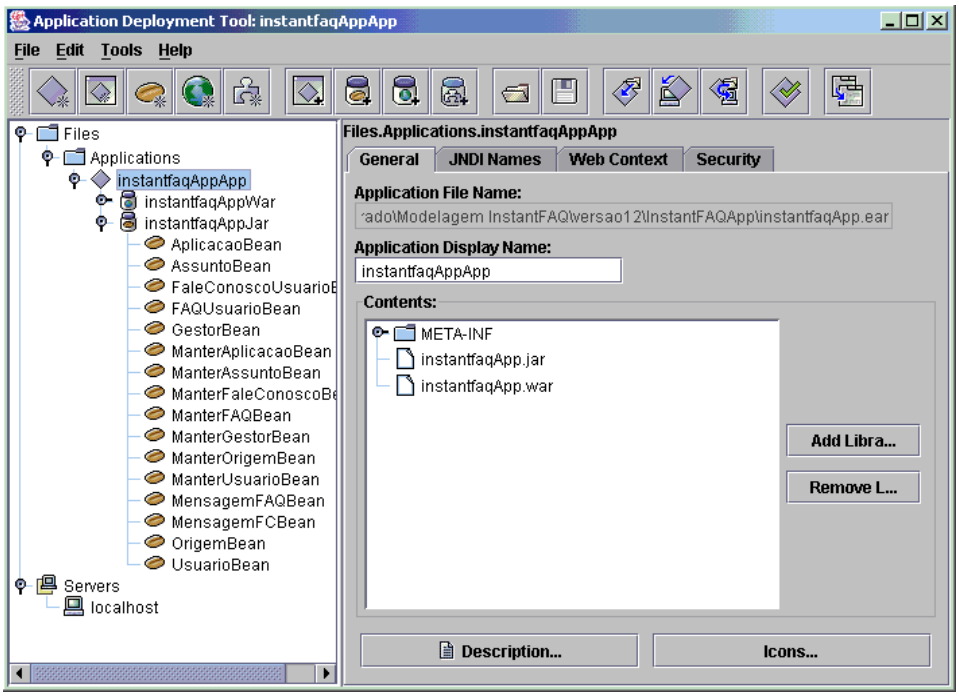


FIGURA 4.23: DEPLOYTOOL.

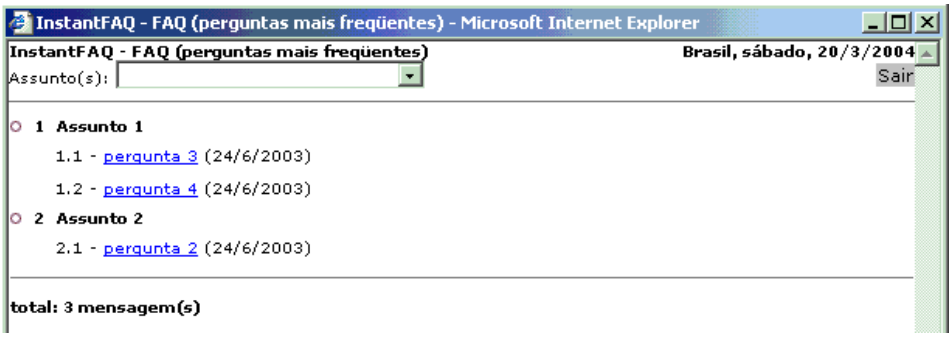


FIGURA 4.24: FUNCIONALIDADE FAQ DO USUÁRIO-ONLINE.

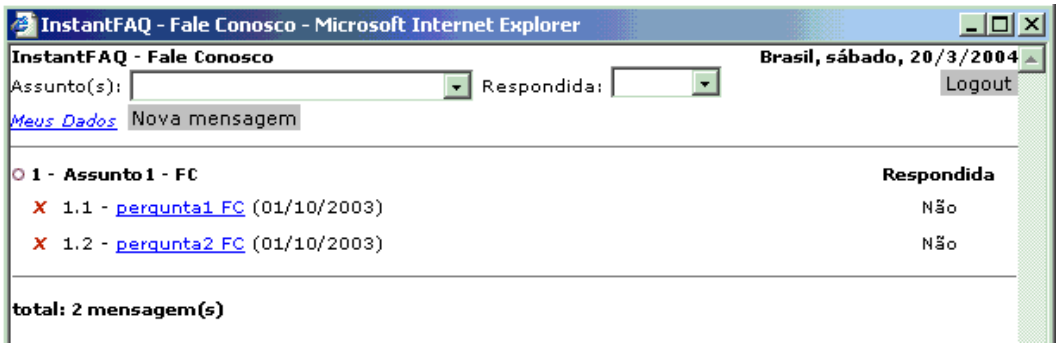


FIGURA 4.25: FUNCIONALIDADE FALE CONOSCO DO USUÁRIO-ONLINE.

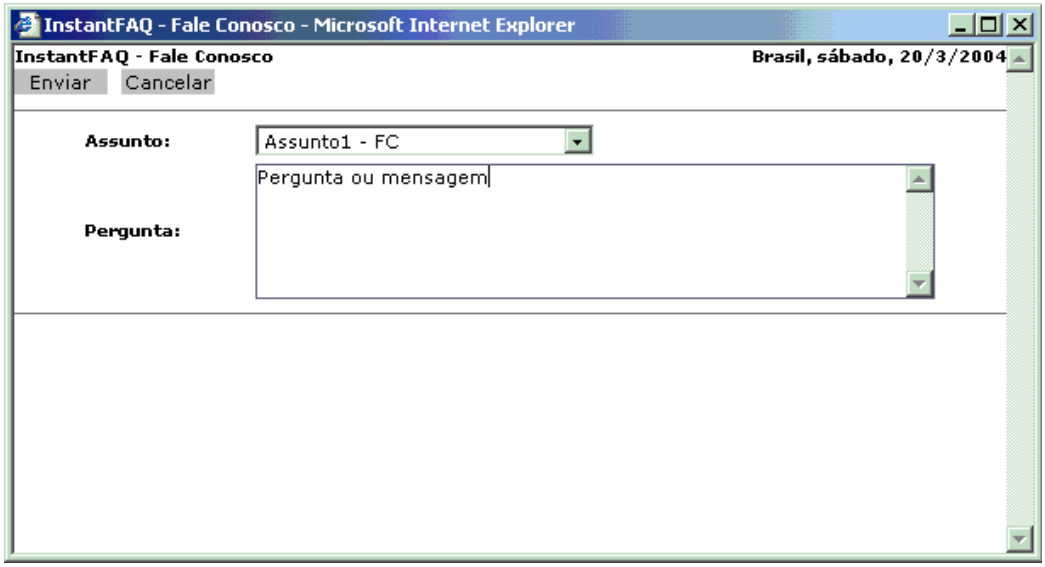


FIGURA 4.26: NOVA MENSAGEM DE FALE CONOSCO DO USUÁRIO-ONLINE.

Para os gestores da ferramenta há uma única área na InstantFAQ denominada Administração da InstantFAQ (Figura 4.27) que permite o acesso a todos os cadastros e funções necessárias para gerenciar as mensagens FAQ e Fale Conosco. Por exemplo, dentre as principais opções destaca-se: Cadastro de Aplicações, FAQ e Fale Conosco descritos a seguir:

FIGURA 4.27: MENU DE ADMINISTRAÇÃO PARA O GESTOR.

1) Cadastro de Aplicações: permite cadastrar a WebApp que utilizará a InstantFAQ por meio do botão `Nova aplicação` (Figura 4.28). A WebApp cadastrada tem um código único que é utilizado na integração com a ferramenta, é por este código que a InstantFAQ separa as mensagens de FAQ e de Fale Conosco de cada uma das WebApp que a utiliza;

2) FAQ: permite criar, editar e excluir as mensagens necessárias, como em `Nova mensagem` (Figura 4.29), onde o gestor cria uma mensagem de FAQ, informando o assunto, a origem da mensagem (correio, fórum de discussão, etc), a pergunta e a resposta (Figura 4.30); e

3) Fale Conosco: permite obter uma lista com todas as mensagens de Fale Conosco enviadas pelos usuários. Estas mensagens são classificadas por assunto, em que o gestor ainda pode classificar em mensagens respondidas, não respondidas ou ambas (Figura 4.31). Para responder uma mensagem, o gestor seleciona a mensagem e informa a resposta para a mesma (Figura 4.32). Neste momento, o gestor pode escolher se deseja responder a mensagem para o usuário-online e enviá-la para a FAQ, ou simplesmente responder para o usuário-online. Quando o gestor enviar a mensagem para FAQ, surgem novas opções (Figura 4.33) que permitem alterar o assunto da mensagem para um assunto mais adequado a FAQ, bem como a pergunta e a resposta, caso necessário.

FIGURA 4.28: CADASTRO DE APLICAÇÃO.

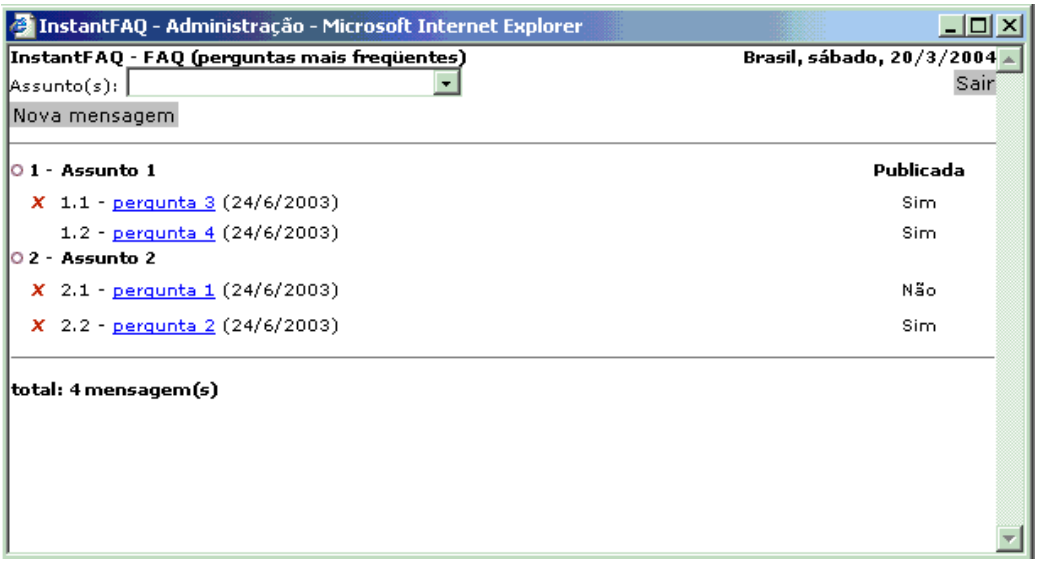


FIGURA 4.29: FAQ DO GESTOR.

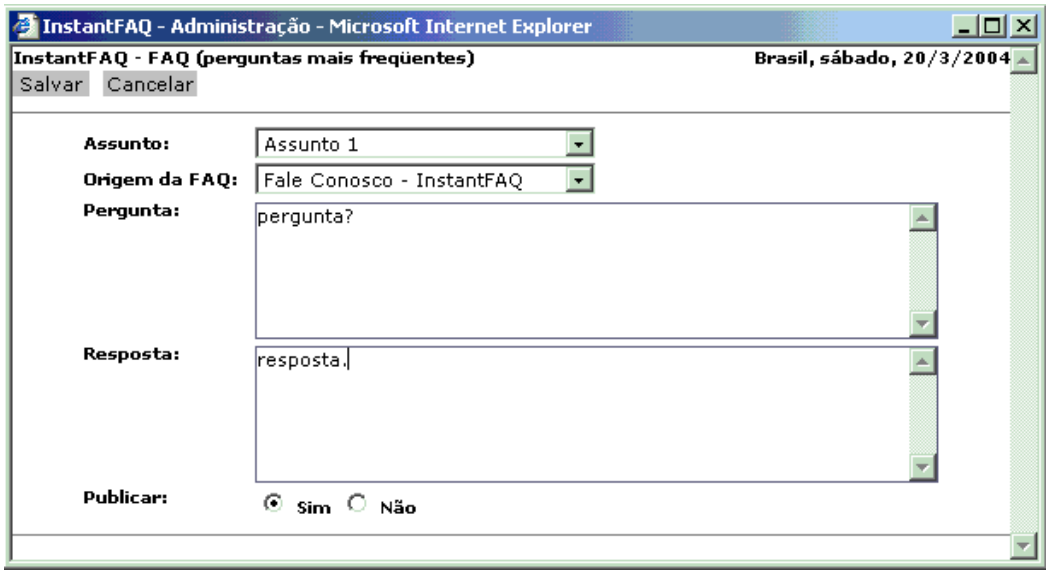


FIGURA 4.30: NOVA MENSAGEM DE FAQ.

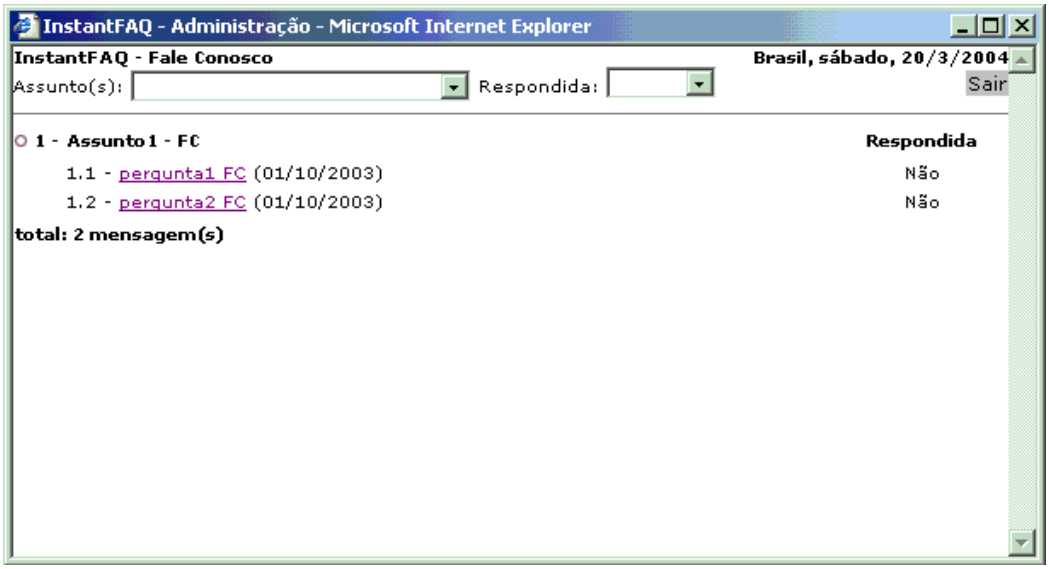


FIGURA 4.31: FALE CONOSCO DO GESTOR.

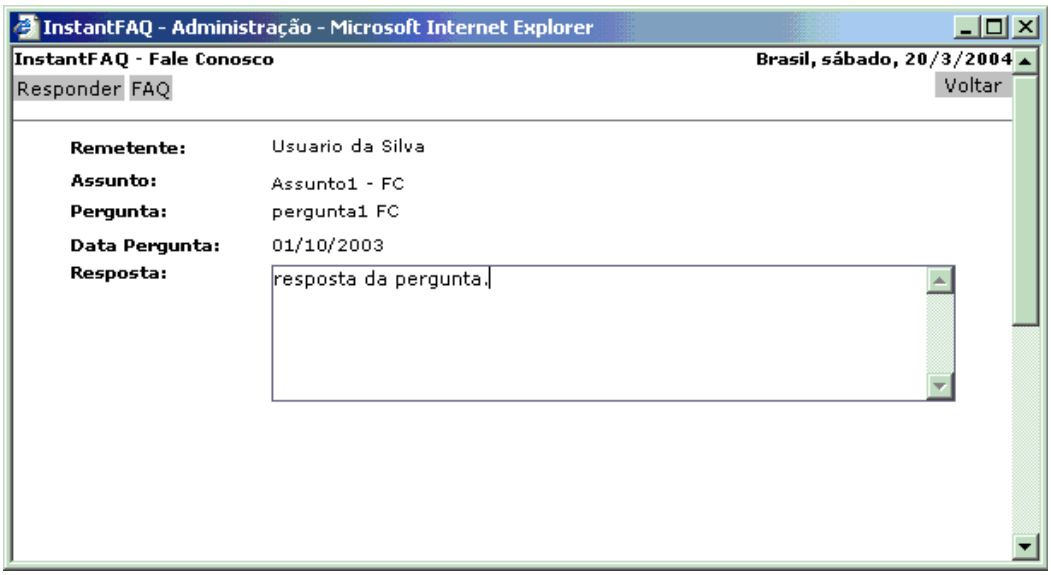


FIGURA 4.32: RESPONDER MENSAGEM DE FALE CONOSCO.

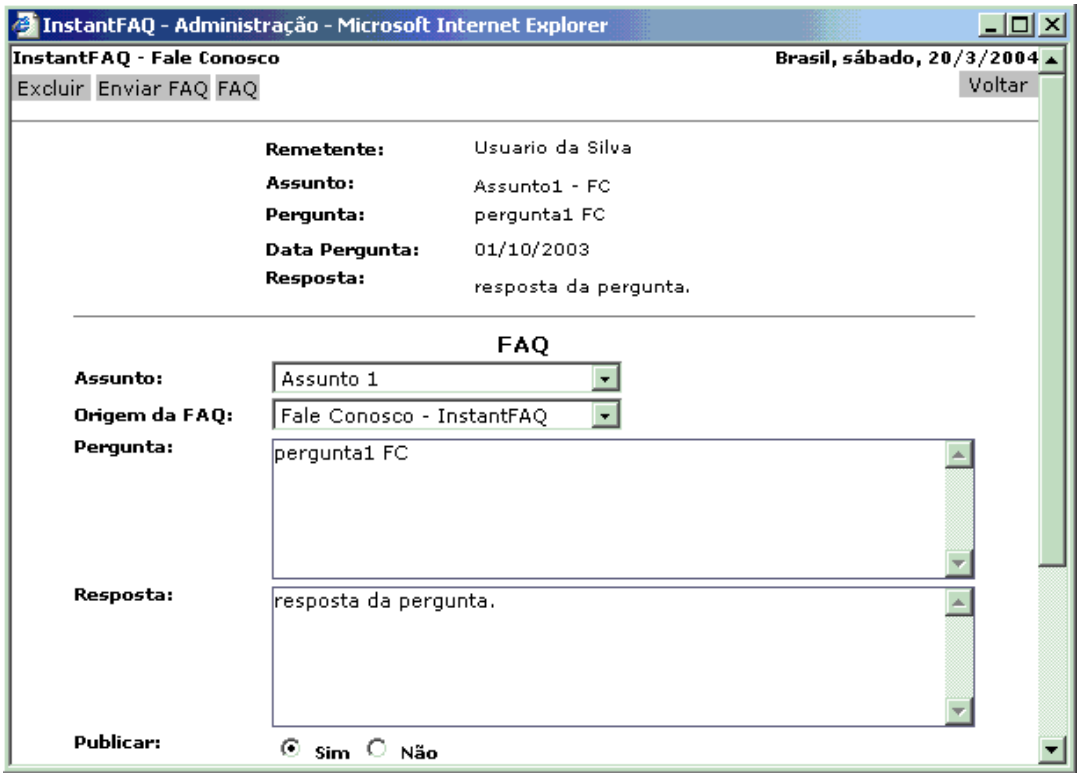


FIGURA 4.33: ENVIAR MENSAGEM DE FALE CONOSCO PARA A FAQ.

4.5. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a arquitetura em três camadas e as funcionalidades da ferramenta InstantFAQ. Dentre as funcionalidades apresentadas destaca-se o envio de mensagens de Fale Conosco diretamente para a FAQ, isto é, o gestor, ao responder a mensagem pode reescrever a pergunta e a resposta e cadastrar esta mensagem como uma mensagem de FAQ, diretamente do cadastro de Fale Conosco. Além disso, foi utilizada e validada a abordagem ADComp no desenvolvimento da ferramenta InstantFAQ, mostrando os diversos modelos e artefatos gerados nas três etapas da abordagem .

No próximo capítulo é apresenta a integração da InstantFAQ em duas WebApps — SIGS e TelEduc, descrevendo a forma e o tipo de integração, os resultados e as dificuldades na integração.

Reutilização e Integração da InstantFAQ 5

5.1. CONSIDERAÇÕES INICIAIS

Neste capítulo são apresentadas duas abordagens de integração para a InstantFAQ e seus componentes em WebApps disponíveis comercialmente. Para exemplificar, a InstantFAQ é integrada e reutilizada no ambiente de educação a distância TelEduc (ROCHA, 2002a; 2002b) e no SIGS (ACOSTA; TURINE, 2003).

O capítulo está organizado da seguinte forma: na seção 5.2 são apresentadas as abordagens de reutilização da InstantFAQ e seus componentes. Na seção 5.3 são descritas as regras de integração e o processo realizado para integrar a InstantFAQ nas WebApps TelEduc e SIGS. Na seção 5.4 é realizada uma avaliação da abordagem de integração utilizada e na seção 5.5 são apresentadas as considerações finais.

5.2. ABORDAGENS DE INTEGRAÇÃO

As duas abordagens de integração e de reutilização da ferramenta InstantFAQ e de seus componentes em WebApps são apresentadas a seguir. Resumidamente, a primeira, denominada Acesso Indireto, propõe utilizar a ferramenta com sua própria interface em JSP (cor, formatação de texto, layout e disposição das funcionalidades). A outra abordagem, denominada Acesso Direto, propõe reutilizar os componentes EJB que compõem a ferramenta, isto é, reaproveitar somente os métodos e as funcionalidades dos componentes para serem acoplados a outras tecnologias de apresentação, como PHP (*HyPertext Preprocessor*), ASP (m*icrosoft Active Server Pages*), XML (*eXtensible Markup Language*), entre outras possibilitando uma interface personalizada, sem modificar as características da interface da WebApp.

5.2.1. Acesso Indireto

Os componentes do domínio de gestão de FAQ e de Fale Conosco especificados e implementados foram reutilizados no desenvolvimento da ferramenta InstantFAQ, que possui o cliente em JSP com uma interface própria.

A integração da InstantFAQ a uma WebApp corresponde à utilização da ferramenta pronta e empacotada com suas características definidas, impossibilitando a personalização de sua interface. Isto limita a padronização de interfaces, mas

facilita a utilização das funcionalidades providas pela ferramenta, não necessitando desenvolver novas páginas e formulários na WebApp hospedeira para apresentar os resultados dos métodos dos componentes. Esta abordagem é utilizada neste trabalho para exemplificar a reutilização da ferramenta InstantFAQ em WebApps.

5.2.2. Acesso Direto

Esta abordagem implica na reutilização direta dos métodos disponibilizados pelos componentes (interoperabilidade) sem a necessidade de se utilizar o cliente JSP. Desta forma, a WebApp que reutiliza os métodos dos componentes tem possibilidade de personalizar as páginas, conforme a interface utilizada pela mesma.

Entretanto, o desenvolvedor da WebApp terá a responsabilidade de construir os formulários de entrada e de saída de dados, conforme os parâmetros dos métodos utilizados; além de conhecer a tecnologia de componentes e principalmente como invocar os métodos dos componentes EJB na linguagem de programação utilizada na WebApp. O desenvolver precisará criar pontes ou *bridges* e utilizar outros recursos e serviços para ter acesso aos componentes de negócio e seus métodos. A utilização desta abordagem não é contemplada neste trabalho, não pela sua dificuldade, mas por considerar estudos para uma proposta de trabalhos futuros.

5.3. PROCESSO DE INTEGRAÇÃO

5.3.1. Regras de Integração

Conforme a primeira abordagem de integração, o processo é baseado em chamada de páginas JSP e passagem de parâmetros, que são denominadas, neste trabalho, como regras de integração da InstantFAQ.

Cada regra de integração corresponde a uma funcionalidade que o usuário-online ou gestor tem acesso, como:

- Funcionalidade FAQ: utilizado pelos gestores e usuário-online; permite a consulta das perguntas mais freqüentes, não necessitando de autenticação, uma vez que a FAQ é apenas para consulta;
- Funcionalidade Fale Conosco: utilizado somente por usuários; permite enviar mensagens aos gestores e manter um histórico de todas as mensagens enviadas pelo usuário-online. Neste caso, o usuário-online é autenticado para que somente sejam apresentadas as mensagens referentes a ele;
- Funcionalidade Administração: utilizado somente por gestores, permite manter e gerenciar principalmente a FAQ (criar, editar, excluir mensagens) e o Fale Conosco (responder mensagens), além de outros cadastros auxiliares, requerendo desta forma, uma autenticação do gestor.

Além destas funcionalidades há um item extra da funcionalidade FAQ, o Direto FAQ, que permite ao gestor enviar um texto — uma possível mensagem para a FAQ — diretamente para a funcionalidade FAQ, por exemplo, a partir de uma mensagem do correio, do bate-papo ou do fórum de discussão do TelEduc. Deve-se ressaltar que o Direto FAQ pode ser utilizado somente quando a WebApp integrar as funcionalidades FAQ e Administração da InstantFAQ.

A autenticação desenvolvida para a InstantFAQ apresenta-se de duas maneiras:

1. Autenticação pela InstantFAQ: utilizada especificamente para WebApps que não possuem usuários e, portanto, não fazem autenticação. O gestor da InstantFAQ deve cadastrar os gestores da WebApp para que estes tenham acesso à funcionalidade de Administração. Quanto aos usuários que utilizarem o Fale Conosco, na primeira vez devem se cadastrar por meio de uma opção na página de autenticação do mesmo para, posteriormente, terem acesso ao Fale Conosco; e
2. Autenticação pela WebApp: utilizada especificamente para aplicações que exigem a autenticação de seus usuários. Tal estratégia evita que gestores e usuários passem por uma outra autenticação ao utilizarem a InstantFAQ. Isto é possível, pois o administrador da WebApp segue a regra de integração própria para este caso que cria, automaticamente, na primeira vez que o gestor/usuário-online utilizar a InstantFAQ, um cadastro para o gestor/usuário-online no banco de dados da ferramenta, garantindo que somente as informações respectivas ao gestor/usuário-online corrente sejam exibidas.

As regras de integração para as funcionalidades da ferramenta seguem o seguinte padrão: http://<DNS>:8000/AppInstantFAQRoot/<REGRA>, onde <DNS> refere-se ao *Domain Name System* onde a InstantFAQ está sendo executada, seguida pela porta do servidor J2EE (:8000); AppInsntantFAQRoot é o *alias* (apelido) da ferramenta no servidor e é fixo; e <REGRA> é a chamada a uma determinada página JSP com seus devidos parâmetros, conforme explicitados a seguir.

<table>
<tr><th colspan="2">Regra 1: Funcionalidade FAQ</th></tr>
<tr><td colspan="2">Página JSP: iFaqUsr_listarFaq.jsp</td></tr>
<tr><td colspan="2">Parâmetros:<table><tr><td>1) campo_numAplic</td><td>2) campo_numAssnt</td></tr><tr><td>Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.</td><td>Tipo inteiro (>= 0) correspondente ao número do assunto pelo qual as mensagens de FAQ são classificadas. Valor default é zero (0), indicando que todas as mensagens de FAQ são exibidas e agrupadas pelo respectivo assunto.</td></tr></table></td></tr>
<tr><td colspan="2">Exemplo: .../iFaqUsr_listarFaq.jsp?campo_numAplic=1&campo_numAssnt=0;</td></tr>
</table>

QUADRO 5.1: REGRA DE INTEGRAÇÃO FAQ.

As regras dois (2) e três (3) referem-se às funcionalidades da InstantFAQ que requerem autenticação de gestores/usuários e desdobram-se em 2.1 (Quadro 5.2), 2.2 (Quadro 5.3); 3.1 (Quadro 5.4) e 3.2 (Quadro 5.5) para tratar as duas maneiras de autenticação abordadas anteriormente. A regra quatro (4), (Quadro 5.6), pertencente à funcionalidade FAQ e embute a autenticação do gestor; em que somente o gestor pode enviar diretamente uma mensagem para a FAQ.

Regra 2.1: Funcionalidade Fale Conosco (Autenticação pela InstantFAQ)
Página JSP: iUsr_logInOut.jsp
Parâmetros: 1) campo_numAplic Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.
Exemplo: .../iUsr_logInOut.jsp?campo_numAplic=1;

QUADRO 5.2: REGRA DE INTEGRAÇÃO FALE CONOSCO (AUTENTICAÇÃO INSTANTFAQ).

Regra 2.2: Funcionalidade Fale Conosco (Autenticação pela WebApp)	
Página JSP: iFCUsr_semAutent.jsp	
Parâmetros:	
1) campo_numAplic	2) campo_loginUsr
Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.	Tipo texto correspondente ao *login* do usuário-online cadastrado na InstantFAQ.
3) campo_nomeUsr	
Tipo texto correspondente ao nome do usuário-online cadastrado na InstantFAQ.	
Exemplo: .../iFCUsr_semAutent.jsp?campo_numAplic=1&campo_loginUsr=rgolveia&campo_nomeUsr=Reginaldo Golveia;	

QUADRO 5.3: REGRA DE INTEGRAÇÃO FALE CONOSCO (AUTENTICAÇÃO WEBAPP).

Regra 3.1: Funcionalidade Administração (Autenticação pela InstantFAQ)
Página JSP: iGstr_logInOut.jsp
Parâmetros: 1) campo_numAplic Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.
Exemplo: .../iGstr_logInOut.jsp?campo_numAplic=1;

QUADRO 5.4: REGRA DE INTEGRAÇÃO ADMINISTRAÇÃO (AUTENTICAÇÃO INSTANTFAQ).

Regra 3.2: Funcionalidade Administração (Autenticação pela WebApp)	
Página JSP: iFaqGstr_semAutent.jsp	
Parâmetros:	
1) campo_numAplic	2) campo_loginGstr
Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.	Tipo texto correspondente ao *login* do gestor cadastrado na InstantFAQ.
3) campo_nomeGstr	
Tipo texto correspondente ao nome do gestor cadastrado na InstantFAQ.	
Exemplo:.../iFaqGstr_semAutent.jsp?campo_numAplic=1&campo_loginGstr=gferraz&campo_nomeGstr=Gilda Ferraz;	

QUADRO 5.5: REGRA DE INTEGRAÇÃO ADMINISTRAÇÃO (AUTENTICAÇÃO WEBAPP).

Regra 4: Direto FAQ	
Página JSP: iFaqGstr_diretoFaq.jsp	
Parâmetros:	
1) campo_numAplic	2) campo_loginGstr
Tipo inteiro (>= 1) correspondente ao número da WebApp cadastrada na Administração da InstantFAQ.	Tipo texto correspondente ao *login* do gestor cadastrado na InstantFAQ.
3) campo_nomeGstr	4) campo_pergunta
Tipo texto correspondente ao nome do gestor cadastrado na InstantFAQ.	Tipo texto contendo a possível pergunta de FAQ.
5) campo_numOrgm	
Tipo inteiro (>= 1) correspondente ao número da origem de FAQ, por exemplo, de um correio, mural, bate-papo cadastrados na Administração da InstantFAQ.	
Exemplo:.../iFaqGstr_diretoFaq.jsp?campo_numAplic=1&campo_loginGstr=gferraz&campo_nomeGstr=Gilda Ferraz&campo_pergunta=Qual o objetivo do texto 2?&campo_numOrgm=4;	

QUADRO 5.6: REGRA DE INTEGRAÇÃO DIRETO FAQ.

Deve-se observar que a funcionalidade Administração da InstantFAQ é obrigatória e deve ser utilizado em conjunto com as funcionalidades FAQ ou Fale Conosco, podendo estas duas ser utilizadas juntas ou separadas. Assim, uma WebApp pode integrar as funcionalidades Administração da InstantFAQ e FAQ; ou Administração da InstantFAQ e Fale Conosco; ou Administração da InstantFAQ, FAQ e Fale Conosco.

5.3.2. Integração no SIGS

O SIGS (http://www.sigs.org.br) (ACOSTA; TURINE, 2003) é uma WebApp desenvolvida pelo IEE/PUC-SP e pelo DCT/UFMS que objetiva auxiliar o processo de gestão, avaliação e monitoramento de programas sociais de transferência de renda. Atualmente, várias instituições são parceiras do projeto, tais como Prefeitura Municipal de São Paulo e Governo do Estado de Mato Grosso do Sul. O SIGS está sendo desenvolvido em ASP (*microsoft Active Server Pages*) e banco de dados Microsoft SQL Server 7.0.

Na Figura 5.1 é ilustrada a *homepage* do SIGS. A página com todas as ferramentas integradas ao SIGS que possibilitam uma gestão eficiente de programas sociais é apresentada na Figura 5.2.

Como se pode notar na página de ferramentas do SIGS, existe um Fale Conosco implementado (Figura 5.2), e desta forma somente as funcionalidades FAQ e Administração da InstantFAQ são integrados. As regras de integração utilizadas são: Regra 1 para a funcionalidade FAQ e Regra 3.2 para a funcionalidade Administração, pois o SIGS apresenta uma autenticação de usuário-online própria (Figura 5.1). As regras são ilustradas pelos links [`FAQ – InstantFAQ`] e [`Administração InstantFAQ`] na Figura 5.3.

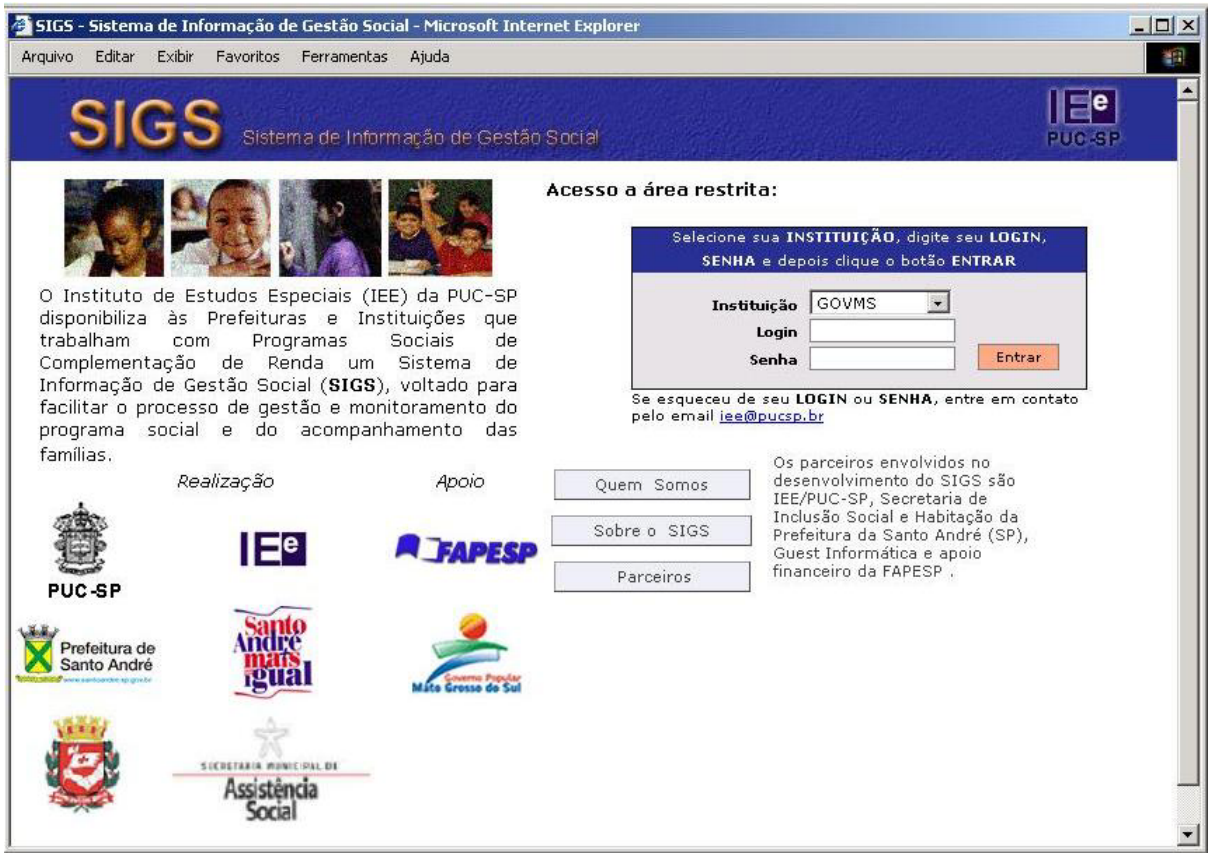


FIGURA 5.1: HOMEPAGE DO SIGS.

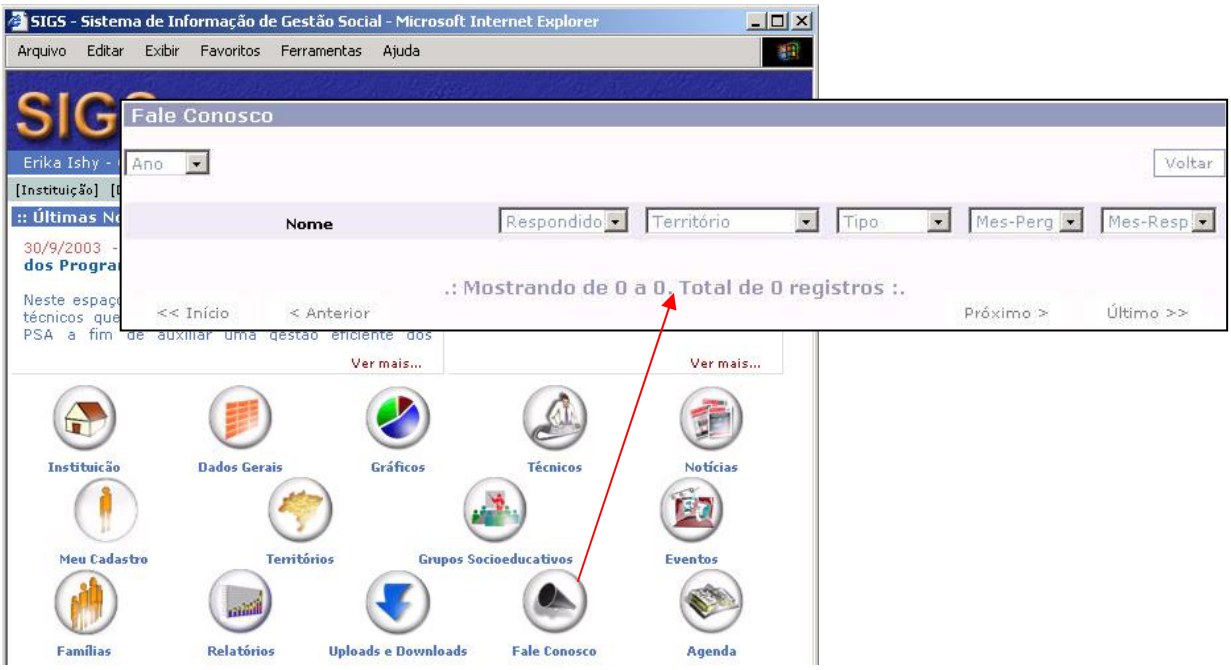


FIGURA 5.2: PÁGINA DE FERRAMENTAS DO SIGS E SEU FALE CONOSCO.

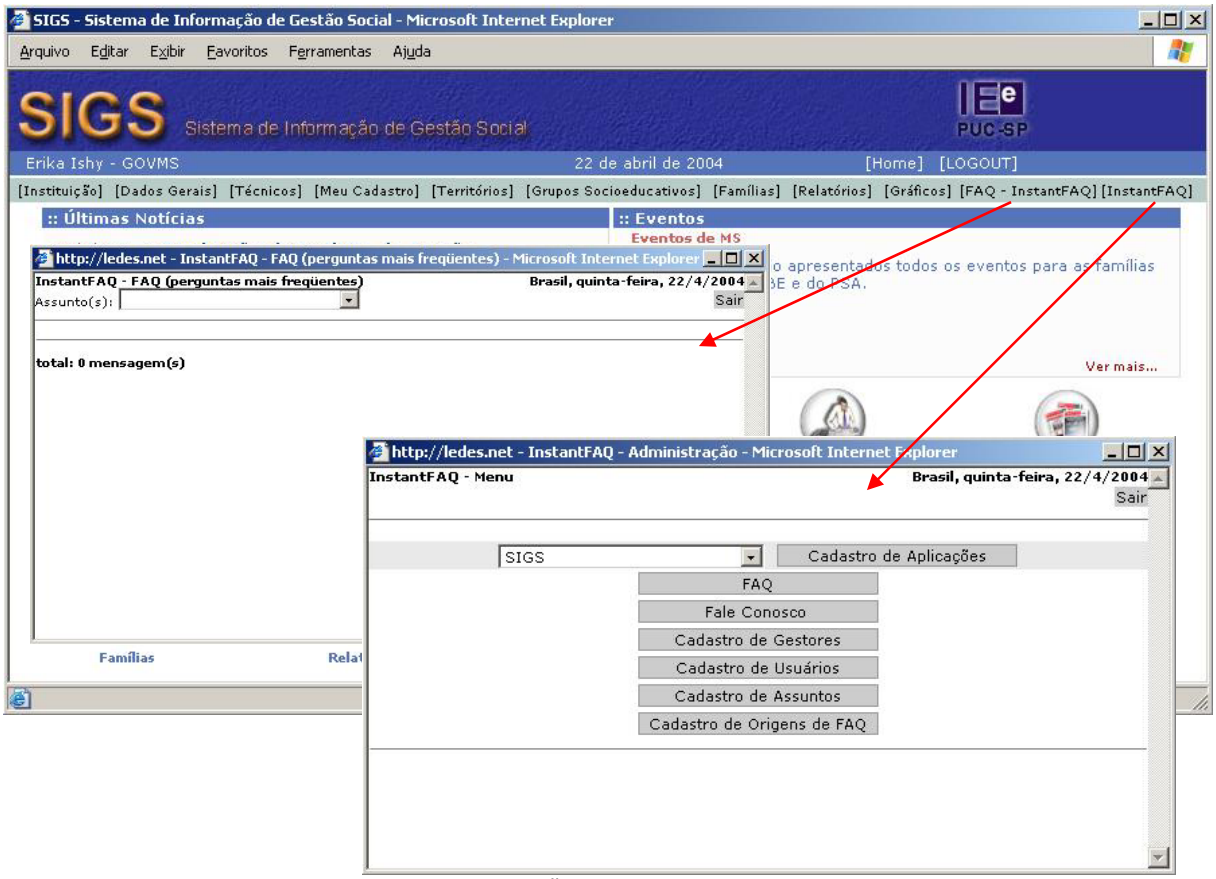


FIGURA 5.3: FAQ E ADMINISTRAÇÃO DA INSTANTFAQ INTEGRADOS NO SIGS.

5.3.3. Integração no TelEduc

O TelEduc (ROCHA, 2002a; 2002b) (http://ead.dct.ufms.br/~teleduc) é um ambiente de educação a distância para criação e gerenciamento de cursos online do NIED/Unicamp (Figura 5.4). O ambiente possui dois tipos de usuários: (1) Formador

— responsável pela criação e gerenciamento do curso, em que cada curso é composto por um certo número de ferramentas, tais como Agenda, Atividades, Material de Apoio, entre outras; e (2) Aluno — que participa do curso realizando as atividades e tarefas propostas. Desta forma, o ambiente separa as ferramentas do curso de acordo com o tipo de usuário-online corrente — Aluno ou Formador.

Seguindo esta estratégia, as regras de integração da InstantFAQ no TelEduc utilizadas são descritas a seguir.

A funcionalidade FAQ (regra 1) não necessita de autenticação, podendo ser utilizada por todos, e está disponível nas visões de Aluno e Formador representada pelo link `FAQ – InstantFAQ` (Figura 5.5 e Figura 5.6). Quanto à funcionalidade Fale Conosco (regra 2.2) representada pelo link `Fale Conosco – InstantFAQ` é exibido somente para os Alunos (Figura 5.5). A funcionalidade Administração da InstantFAQ (regra 3.2), representado pelo link `Administração InstantFAQ`, é exibido somente para os Formadores (Figura 5.6). Para melhor exemplificar, o Direto FAQ (regra 4) é utilizado somente pelos formadores nas ferramentas Correio (Figura 5.7) e Fórum de Discussão (Figura 5.8) do curso Projeto de Websites oferecido pelo Prof. Dr. Marcelo Augusto Santos Turine para o curso de Especialização em Análise de Sistemas (DCT/UFMS).

Igualmente ao SIGS, o TelEduc apresenta uma autenticação de usuários própria, devendo ser utilizadas as regras de integração da InstantFAQ que tratam da autenticação pela WebApp, no caso as regras 2.2 e 3.2.

FIGURA 5.4: TELEDUC PARA O CURSO PROJETO DE WEBSITES.

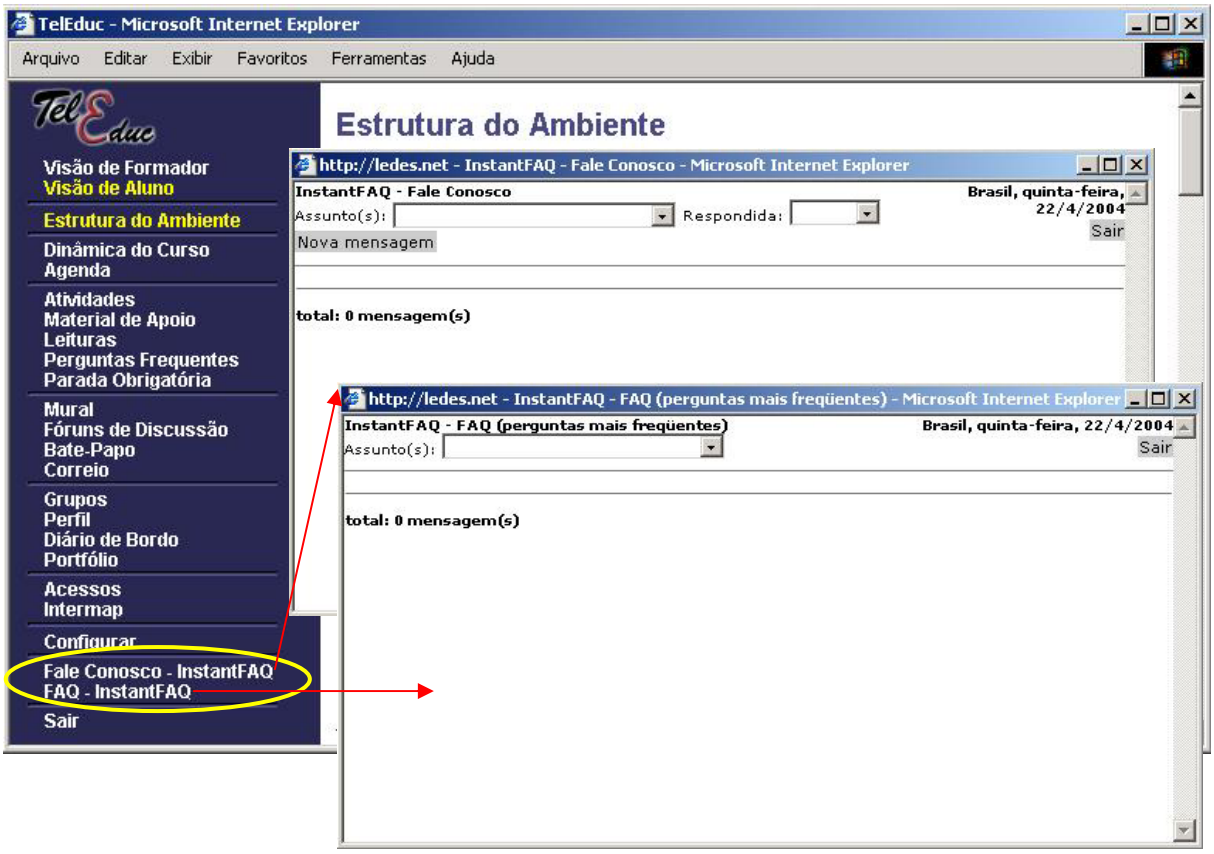


FIGURA 5.5: TELEDUC NA VISÃO DO ALUNO.

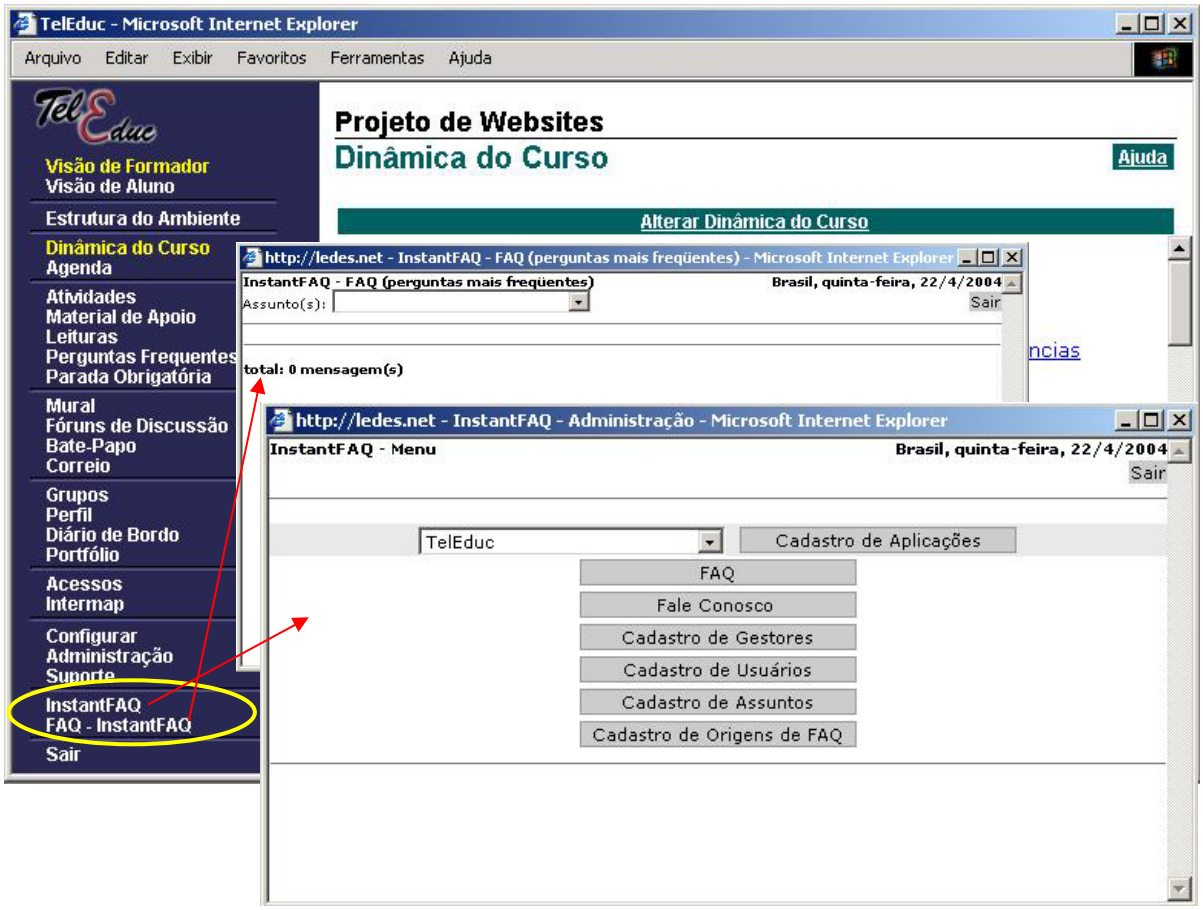


FIGURA 5.6: TELEDUC NA VISÃO DO FORMADOR.

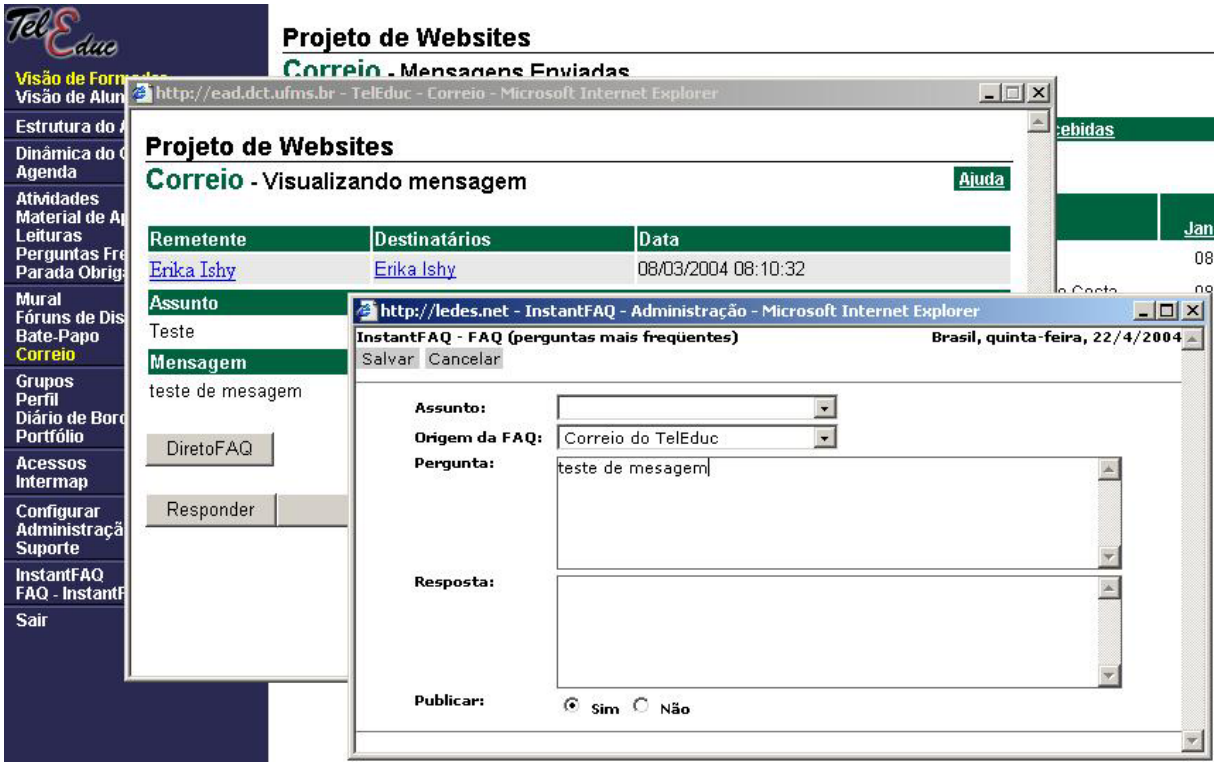


FIGURA 5.7: DIRETO FAQ NA FERRAMENTA CORREIO.

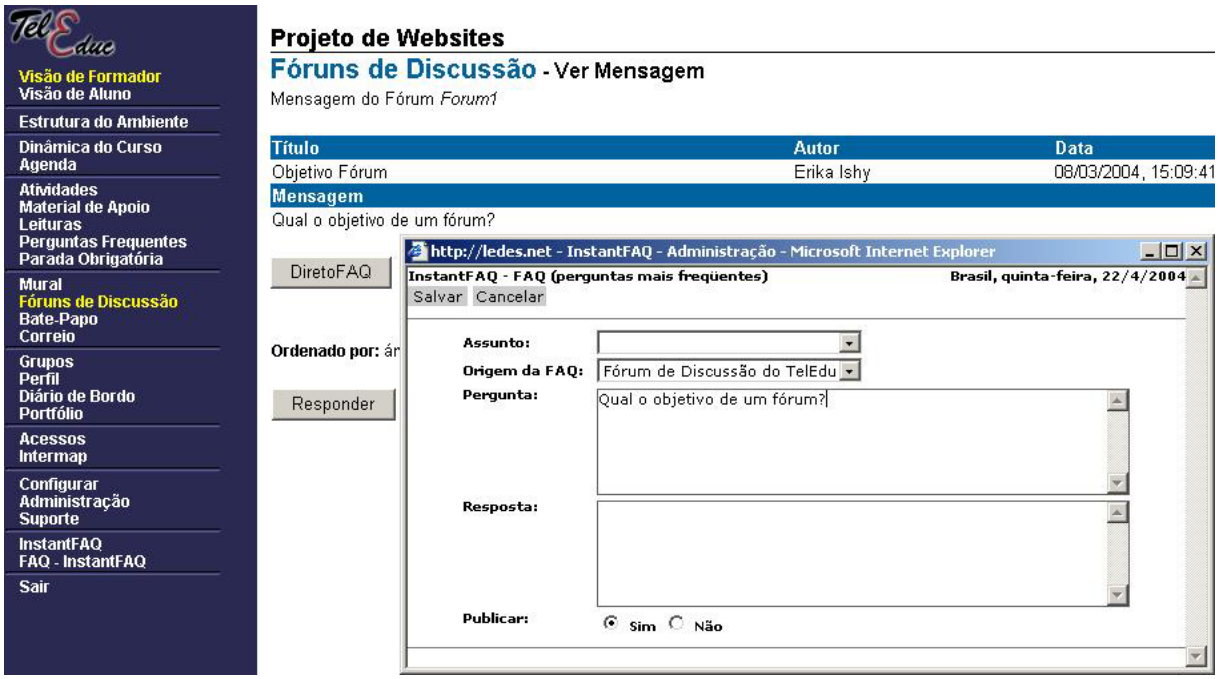


FIGURA 5.8: DIRETO FAQ NA FERRAMENTA FÓRUM DE DISCUSSÃO.

5.4. AVALIAÇÃO

O processo de integração, seguindo a primeira abordagem de reutilização, realizada no SIGS e no TelEduc mostrou-se eficaz. A utilização das regras de integração simplificou o processo, fazendo com que o administrador da WebApp reutilize todas as funcionalidades oferecidas pela InstantFAQ.

No entanto, esta forma de integração não permite a personalização da interface, pré-definida pelo cliente JSP, conforme os padrões utilizados pelas WebApps. Tal falta de padronização foi observada quando da avaliação da WebApp pelos usuários, que questionavam “porque as páginas têm layout diferentes?”.

Quanto ao desempenho das WebApps, após a integração da InstantFAQ, estas não são comprometidas, uma vez que a InstantFAQ é executada em outro servidor. A resposta às funcionalidades da ferramenta depende exclusivamente do servidor e do tempo de processamento dos métodos solicitados.

Como a InstantFAQ é uma aplicação EJB, necessita além do Java, o servidor J2EE para implantar a ferramenta com no mínimo 128 MB de memória no sistema operacional Microsoft Windows 2000. Esta é a principal requisição para se utilizar e integrar a ferramenta em qualquer WebApp. Por sua vez, o servidor que armazena a WebApp não precisa instalar pacotes específicos Java ou outro software para fazer a integração com a InstantFAQ.

A inclusão das regras de integração nas WebApps não é complicada, o importante é saber que variáveis armazenam, por exemplo, o login, a senha e o nome do gestor/usuário-online para que possam ser repassadas como parâmetros para as regras de integração; e se necessário, criar procedimentos e incluir códigos que tornem este repasse mais fácil para as variáveis das regras. O entendimento da lógica da WebApp também é muito importante, uma vez que as regras de integração são incluídas no código-fonte da WebApp.

5.5. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a reutilização da ferramenta InstantFAQ em duas WebApps (SIGS e TelEduc), indicando duas formas de abordagem para a integração, sendo que apenas a integração ao nível da ferramenta (acesso indireto) foi realizada, não sendo contemplado neste trabalho a integração ao nível de componentes (acesso direto). Além disso, neste capítulo foi descrito como integrar as funcionalidades da ferramenta, representadas pelas regras de integração referentes às funcionalidades FAQ, Fale Conosco e Administração da InstantFAQ. Uma avaliação realizada nas WebApps foi realizada, concluindo que a mesma foi satisfatória na validação da ferramenta baseada em componentes. No próximo capitulo são apresentadas as conclusões deste trabalho, apresentando suas contribuições e perspectivas para trabalhos futuros.

Conclusões 6

6.1. CONSIDERAÇÕES INICIAIS

A partir da pesquisa dos vários métodos existentes e objetivando construir um processo de desenvolvimento de componentes intuitivo, direto e baseado na UML nasceu a abordagem mista ADComp. Assim, neste trabalho foi elaborada uma abordagem para o desenvolvimento de componentes denominada ADComp, que é baseada, principalmente, no *UML Components* o qual integra conceitos do *Catalysis* e do RUP. A adaptação do *UML Components* à abordagem ocorre da definição das interfaces e dos componentes de negócio, visando uma maior separação e individualização das interfaces e componentes de acordo com seus conceitos.

A ADComp é dividida em três etapas: definição de requisitos, definição de componentes e implementação de componentes, sendo que a etapa de definição de componentes é dividida em identificação do componente, interação do componente e especificação do componente. Um das características da ADComp é que durante a modelagem dos componentes busca-se integrar a definição de componentes com sua implementação numa linguagem executável, conforme a linguagem de programação e o modelo de componentes adotados, como Java e EJB, respectivamente.

Dentre os modelos de componentes pesquisados foi adotado para a etapa de implementação de componentes o EJB 2.0, pois possui maior portabilidade, uma vez que é escrita para a linguagem Java que é independente de plataforma, além de ser um ambiente robusto que oferece suporte a aplicações com requisitos rigorosos de escalabilidade, de distribuição e de disponibilidade. Para armazenar os dados foi utilizado o banco de dados relacional *Clousdscape* por ser nativo do J2EE, servidor de aplicações EJB utilizado neste trabalho.

Para validar a ADComp e ilustrar na prática o desenvolvimento de componentes, foi realizado um estudo de caso da ferramenta Web de interatividade baseada em componentes para gestão de FAQ e de Fale Conosco denominada InstantFAQ. Com a utilização da ADComp foi possível identificar, especificar e implementar os componentes para gestão de FAQ, que foram integrados e formaram a ferramenta InstantFAQ. Estes componentes podem ser reutilizados para formar outras ferramentas de interatividade que por sua vez podem ser integradas em diversas WebApps.

Estes componentes podem ser reutilizados indiretamente por meio da ferramenta InstantFAQ, onde os métodos dos componentes são acessados via *layout* pré-definido na ferramenta. Ou estes podem ser reutilizados diretamente em WebApps, mantendo o padrão de *layout* destas aplicações, onde o desenvolvedor deverá criar pontes para acessar os métodos dos componentes.

Os componentes do domínio gestão de FAQ são reutilizáveis completamente em WebApps que necessitam de interatividade entre seus usuários-online e gestores de WebApps tanto educacionais ou não, pois apresentam funcionalidades suficientes para atender às necessidades deste domínio em qualquer WebApp.

A InstantFAQ é um software baseado em componentes e possui uma arquitetura cliente/servidor. O cliente foi desenvolvido em JSP, HTML e *JavaScript*; o servidor utilizado é o J2EE para Microsoft Windows 2000, um servidor de aplicações da SUN para EJB; e o banco de dados é o banco relacional *Cloudscape* que acompanha o servidor J2EE. Apesar de utilizar o servidor para Microsoft Windows, os componentes podem ser recompilados para o Linux, garantindo sua portabilidade. A ferramenta com os componentes e as páginas JSP tem tamanho (160 Kbytes), e a velocidade de processamento e carregamento das páginas é relativamente rápida, dependendo da velocidade da rede e do servidor onde a InstantFAQ está implantada.

Apesar da InstantFAQ utilizar as tecnologias acima citadas, para o modelo EJB, outras tecnologias podem ser utilizadas, como *Servlets*, SGBDs diversos (Oracle, SQL Server), além de outros servidores de aplicação como o JBoss.

A integração da InstantFAQ nas WebApps SIGS e TelEduc como ferramenta de interatividade foi satisfatoriamente realizada não havendo grandes alterações nos códigos-fonte das WebApps, além de manter a mesma performance das duas aplicações.

A principais dificuldades no decorrer deste trabalho estão relacionadas ao aprendizado dos métodos *Catalysis* e *UML Components* e das tecnologias envolvidas para a construção da InstantFAQ, principalmente, os conceitos de EJB e sua programação; a configuração e manipulação do servidor J2EE e do banco de dados *Cloudscape*; e a utilização da MVCASE, que foi de fundamental e gentil a ajuda do Prof. Dr. Antônio Francisco do Prado e seu aluno de mestrado Daniel Lucrédio, que possibilitou um intercâmbio entre o DCT/UFMS e o DC/UFSCar permitindo um aprendizado rápido e eficiente da MVCASE.

Neste capítulo são apresentadas as conclusões gerais do presente trabalho, mostrando as dificuldades encontradas, principais contribuições (seção 6.2) e sugestões para trabalhos futuros (seção 6.3).

6.2. CONTRIBUIÇÕES

As quatro principais contribuições deste trabalho são:

1. O desenvolvimento de uma abordagem de DBC (ADComp) que abrange etapas de identificação e de definição de componentes até a implementação dos componentes especificados numa linguagem executável e num modelo de componentes específico. Em todas as etapas é utilizada a ferramenta CASE MVCASE que auxilia o engenheiro e o desenvolvedor de software na construção e na especificação de componentes;
2. A especificação e a implementação dos componentes de interatividade no domínio de gestão de FAQ e de Fale Conosco na Web utilizando a abordagem proposta;
3. A especificação e a implementação da ferramenta InstantFAQ que reutiliza os componentes especificados em (2); e
4. Integração da InstantFAQ nas WebApps TelEduc e SIGS, definindo as diferentes regras de integração conforme a solução de integração utilizada.

Outras contribuições deste trabalho podem ser destacadas, tais como:

- Descrição das propostas de soluções de integração da ferramenta InstantFAQ a WebApps existentes por meio de regras de integração;
- Revisão bibliográfica dos principais métodos e modelos de componentes. A partir destes métodos foi possível analisar o *UML Components* e propor adaptações de sua utilização na abordagem proposta. Dentre os principais modelos de componentes pesquisados foi possível estudar o EJB além das tecnologias associadas, como JSP e J2EE;
- Teste e validação da InstantFAQ, pois os vários modelos da ADComp foram especificados e acoplados à ferramenta; e
- Fortalecimento da linha de pesquisa em desenvolvimento de componentes e *frameworks* no Laboratório de Engenharia DE Software (LEDES) do DCT/UFMS.

6.3. TRABALHOS FUTUROS

A partir dos resultados deste trabalho pode-se vislumbrar diversas pesquisas que podem ser exploradas em trabalhos futuros. Inicialmente, na etapa de definição de requisitos da ADComp pode-se incluir técnicas de coleta de requisitos para auxiliar e fomentar o trabalho do engenheiro de software. E na terceira etapa, implementação de componentes, pode ser estendida para incluir o padrão CORBA com a linguagem Java, ambos suportados pela MVCASE. Em relação à ferramenta CASE, é interessante investigar como a ferramenta CASE *Rational Rose* pode ser utilizada e integrada a ADComp, mostrando suas vantagens e desvantagens em relação à ferramenta MVCASE.

Somente o estudo de caso de gestão de FAQ e de Fale Conosco para a ADComp não é suficiente para validar a abordagem. Desta forma, outros estudos de caso devem ser desenvolvidos para uma melhor avaliação, podendo incluir um estudo experimental envolvendo alunos de graduação e de especialização no desenvolvimento de projetos baseados em componentes.

Para a ferramenta InstantFAQ, outro trabalho futuro é aperfeiçoar seu layout e suas funcionalidades. Desdobrar a InstantFAQ em duas ferramentas, uma específica para a gestão de FAQ e outra para a gestão de Fale Conosco — explicitando suas características individuais e inter-relacionamentos.

O projeto de uma biblioteca de ferramentas componentizadas no domínio Web, utilizando a ADComp para a construção de novas e diversas ferramentas interativas que possam ser integradas em WebApps é uma outra proposta de trabalho que está se iniciando no grupo de pesquisa do DCT/UFMS.

Desenvolver a integração de acesso direto em outras WebApps, isto é, em nível de componentes, conforme apresentado no capítulo 5 que não foi realizada neste trabalho. Este nível de integração permite o reuso dos métodos dos componentes no padrão da interface das WebApps.

Por último, com o crescimento da Web em diferentes áreas da sociedade, em especial da Educação por meio da EAD, vislumbra-se a construção de ferramentas Web de interatividade, a partir da biblioteca de componentes sugerida anteriormente, para serem integradas e utilizadas em ambientes de EAD. Ferramentas que auxiliam alunos e professores a ensinar e a aprender, fomentando os ambientes de EAD e até mesmo possibilitando a construção de novos ambientes a partir da integração destas ferramentas de interatividade.

Referências

ACOSTA, A. R.; TURINE, M. A. S. Sistema de informação de gestão social: monitoramento e avaliação de programas de complementação de renda. In: ACOSTA, A. R.; VITALE, M. A. F. *Família*: *Laços, redes e políticas públicas*. IEE-PUC/SP, 2003. p.150-162.

ALMEIDA, E. S.; LUCRÉDIO, D.; BIANCHINI, C. P.; PRADO, A. F.; TREVELIN, L. C. *Ferramenta MVCASE — uma ferramenta integradora de tecnologias para o desenvolvimento de componentes distribuídos*. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 16, 2002, Gramado-RS. *Anais do XVI SBES.* Porto Alegre: Instituto de Informática da UFRGS, 2002. p. 432-437.

BACHMANN, F.; BASS, L.; BUHMAN, C.; COMELLA-DORDA, S.; LONG, F.; ROBERT, J.; SEACORD, R.; WALLNAU, K. *Volume II: technical concepts of component-based software engineering*, 2 ed. Software Engineering Institute, 2000. Relatório técnico.

BALASUBRAMANIAN, V.; BASHIAN, A.; PORCHER, D. A large-scale hypermedia application using document management and Web technologies. In: INTERNATIONAL ACM HYPERTEXT CONFERENCE, 8, 1997, Southampton-UK. *Proceedings of Hypertext'97*. New York: ACM Press, 1997. p.134-145.

BIEBER, M.; GALNARES, R.; LU, Q. Web Engineering and flexible hypermedia. In: WORKSHOP IN ADAPTIVE HYPERTEXT AND HYPERMEDIA, 2, 1998, Pittsburgh-USA. *Proceedings of the 2nd Workshop in Adaptive Hypertext and Hypermedia*. Disponível em: <http://wwwis.win.tue.nl/ah98/Proceedings.html> Acesso em: 20 ago. 2003.

BOERTIN, N.; STEEN, M.; JONKERS, H. *Evaluation of component-based development methods*. In: INTERNATIONAL WORKSHOP ON EVALUATION OF MODELING METHODS IN SYSTEMS ANALYSIS AND DESIGN, 6, 2001.

BOSCH, J.; MOLIN, P.; MATTSSON, M.; BENGTSSON, P. O. Object oriented framework — problems & experiences In: FAYAD, M.; JOHNSON, R.; SCHMIDT, D. *Building application frameworks*: *object-oriented foundations of framework design*. JOHN WILLEY AND SONS, 1999. p. 55-82.

BRAGA, R. T. V. *Um processo para construção e instanciação de frameworks baseados em uma linguagem de padrões para um domínio específico*. São Carlos-SP, 2003. 232 f. Tese (Doutorado em Ciência da Computação) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

BROWN, A. W. *Large-scale, component-based development*. USA: Prentice Hall, 2000.

CHEESMAN, J.; DANIELS, J. *UML components — a simple process for specifying component-based*. Addison-Wesley, 2001.

COUNCILL, B.; HEINEMAN, G. T. Definition of a software component and its elements. In: HEINEMAN, G. T.; COUNCILL, W. T. *Component-based software engineering*. ADDISON-WESLEY, 2001. p. 05-19.

CRNKOVIC, I.; HNICH, B.; JONSSON, T.; KIZILTAN, Z. Specification, implementation, and deployment of components. *Communications of the ACM*, v. 45, n. 10, 2002.

CUNNINGHAM, H. C.; LIU, Y.; TADEPALLI, P.; FU M. Component software: a new software engineering course. *The Journal of Computing in Small Colleges*, v. 18, n. 6, p. 10-21, 2003.

D'SOUZA, D.; WILLS, A. *Objects, components and frameworks with UML — the catalysis approach*. Addison-Wesley, 1999.

DE PAEZ, R. A. Un acercamiento a la reutilización en ingeniería de software. *AFIT Magazine*, Universidade de EAFIT, Medellin-Colombia, 1999, n. 114, p. 45-63.

EMMERICH, W. State-of-the-art presentations: distribuited component technologies and their software engineering implications. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 24, 2002, Orlando-Florida. *Proceedins of the 24th International Conference on Software Engineering*. New York: ACM Press, 2002. p. 537-546.

EWALD, T. Overview of com+. In: HEINEMAN, G. T.; COUNCILL, W. T. *Component-based software engineering*. ADDISON-WESLEY, 2001. p. 573-588.

FAYAD, M. E.; SCHMIDT, D. C. Object-oriented application framework. *Communications of the ACM*, v. 40, n. 10, p. 32-38, 1997.

FRAMINGHAM, M. A. *The common object request broker architecture and specification, rev. 2.2*. Object Management Group, 1998. Relatório técnico.

FRANCA, L. P. A.; STAA, A. V. Geradores de artefatos: implementação e instanciação de frameworks. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 15, 2001, Rio de Janeiro-RJ. *Anais do XV SBES.* Rio de Janeiro: COPPE/UFRJ, UERJ, 2001. p. 302-315.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design patterns — abstraction and reuse of object-oriented design. In: EUROPEAN CONFERENCE, 7, 1993, Kaiserslautern-Germany. *Proceedings ECOOP'93 — Object-Oriented Programming*. Kaiserslautern-Germany: Spring, 1993. p. 406-431.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design patterns — elements of reusable object-oriented software*. Addison-Wesley, 1995.

GILL, N. S.; GROVER, P. S. Component-based measurement: few useful guidelines. *ACM SIGSOFT Software Engineering Notes*, v. 28, n. 6, 2003. Disponível em: <http://www.acm.org> The ACM Digital Library Acesso em: 11. fev. 2004.

GIMENEZ, I. M. DE S.; TRAVASSOS, G. H. *O enfoque de linha de produto para desenvolvimento de software*. In: CURSO JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 21, 2002, Florianópolis-SC.

GLOVER, E.; TSIOUTSIOULIKLIS, K.; LAWRENCE, S.; PENNOCK, D.; FLAKE G. Using Web structure for classifying and describing Web pages. In: INTERNATIONAL CONFERENCE ON THE WORLD WIDE WEB, 11, 2002, Honolulu-Hawaii. *Proceedings of the Eleventh International Conference on World Wide Web.* New York: ACM Press, 2002. p. 562-569.

GOLDBERG, M.; SALARI, S.; SWOBODA, P. *World wide web — course tool*: an environment for building www-based courses. In: PROCEEDINGS OF THE FIFTH INTERNATIONAL WORLD WIDE WEB CONFERENCE, 5., 1996, Paris, France. Disponível em:

<http://www5conf.inria.fr/fich_html/papers/P29/Overview.html> Acesso em: 10 dez. 2002.

GOSLING, J.; JOY, B.; STEELE, JR.; GUY, L. *The java language specification*. Addison-Wesley, 1996.

HAEFEL, R. M. *Enterprise javabeans™*. 3. O'Reilly, 2001.

HEINEMAN, G. T.; COUNCIL, W. T. *Component-basead software engineering*: *putting the pieces together*. Addison-Wesley, 2001.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *The unified software development process*. 4. ed. USA: Addison-Wesley, 2001.

JOHNSON, R. E. Documenting frameworks using patterns. In: CONFERENCE ON OBJECT ORIENTED PROGRAMMING SYSTEMS LANGUAGES AND APPLICATIONS OOPSLA, 7, 1992, Vancouver, British Columbia, Canada. *Conference Proceedings on Object-Oriented Programming Systems, Languages, and Applications*. New York-USA: ACM Press, 1992. p. 63-79.

JOHNSON, R. E. Components, frameworks, patterns. In: SYMPOSIUM ON SOFTWARE REUSABILITY, 1997, Boston-Massachusetts-USA. *Proceedings of the 1997 Symposium on Software Reusability.* New York-USA: ACM Press, 1997. p. 10-17.

KRUEGER, C. Software reuse. *ACM Computing Surveys*, v. 24, n. 2, p. 131-183, 1992.

LEEB, A. *A flexible object architecture for component software*. Massachusetts, 1996, 45 f. Dissertação (Mestrado em Ciência da Computação) — Departamento de Engenharia Elétrica e Ciência da Computação, Massachusetts Instituto de Tecnologia. Disponível em: <http://larch.lcs.mit.edu:8001/~aleeb/thesis.ps> Acesso em: 29 ago. 2002.

LUCENA, C. J. P. ET. AL. *AulaNet — an environment for the development and maintenance of courses on the web*. In: PROCEEDINGS OF ICEE'98 — INTERNATIONAL CONFERENCE ON ENGINEERING EDUCATION, 1998, Rio de Janeiro, RJ. Disponível em: <http://ritv.les.inf.puc-rio.br/groupware/> Acesso em: 10 dez. 2002.

LUCRÉDIO, D.; PRADO, A. F.; BIAJIZ, M.; ALMEIDA, E. S.; YAMAMOTO, C. H. Abordagens para recuperação eficiente de componentes utilizando indexação métrica. In: 3º WORKSHOP DE DESENVOLVIMENTO BASEADO EM COMPONENTES, 3., 2003, São Carlos-SP. *Anais*. São Carlos: Antônio Francisco do Prado-UFSCar, 2003. 1 CD-ROM.

MVCASE. *Ferramenta MVCASE*. Disponível em: <www.recope.dc.ufcar.br/mvcase> Acesso em: 05 jan. 2004.

OCL. Object constraint language specification. In: *OMG Unified modeling language specification version 1.4*. Object Management Group, 2001. p. 6-1/6-45.

OMG. *Object management architecture guide 2.0*, 2. ed. Object Management Group, 1992.

OMG. *The Common Object Request Broker: Architecture and Specification*, version 2.4, Object Management Group, 2000.

OMG. *Unified modeling language specification version 1.5*. Object Management Group, 2003. Disponível em: <http://www.omg.org/docs/formal/03-03-01.pdf> Acesso em: 24 fev. 2004.

PRIETO-D'IAZ, R.; ARANGO, G. *Domain analysis and software system modeling*. Califórnia: IEEE Computer Society Press Tutorial, 1991.

PRESSMAN, R. S. Component-based software engineering. In: PRESSMAN, R. S. *Software engineering: a practitioner's approach*. 5. ed., New York, USA: McGraw Hill, 2001. p. 721-746.

ROCHA, H. V. O ambiente TelEduc para educação a distância baseada na web: princípios, funcionalidades e perspectivas de desenvolvimento. In: MORAES, M. C. *Educação a distância: fundamentos e práticas*. Campinas, SP: UNICAMP/NIED, 2002a. p. 197-212.

ROCHA, H. V. *Projeto TelEduc*: pesquisa e desenvolvimento de tecnologia para educação a distância. Disponível em: <http://www.dcc.unicamp.br/~heloisa/> Acesso em: 02 dez. 2002b.

ROSENBERRY, W., KENNEY, D. e FISHER, G. *Understanding DCE*. O'Reilly, 1992.

ROSS, T. *Application and Extension of SADT.* IEEE Transaction of Software Engineering, abr. 1985.

RUP. *Rational unified process: best practices for software development teams*. Rational Software Corporation, 2003. Disponível em: <http://www-106.ibm.com/developerworks/rational/library/253.html> Acesso em: 30 out. 2003.

SAMETINGER, J. *Software engineering with reusable components*. Springer, 1997.

SMARAGDAKIS, Y.; BATORY, D. *Application generators.* Department of Computer Sciences. The University of Texas at Austin, 1998. Disponível em: <http://www.cc.gatech.edu/~yannis/generators.pdf> Acesso em: 06 jan. 2004.

SOMMERVILLE, I. *Software Engineering*. Addison-Wesley, 2001.

SUN MICROSYSTEMS. *Java™ 2 enterprise edition developer's guide*, version 1.2.1, 2000. Disponível em: <http://java.sun.com/j2ee/sdk_1.2.1/techdocs/guides/ejb/html/DevGuideTOC.html> Acessado em: 4 fev. 2004.

SUN MICROSYSTEMS. *The j2ee™ tutorial*, 2002a. Disponível em: <http://java.sun.com/j2ee/1.3/download.html#tutorial> Acesso em: 4 fev. 2003.

SUN MICROSYSTEMS. *Designing enterprise applications with the j2ee™ platform, second edition*, 2002b. Disponível em: <http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/index.html> Acesso em: 5 mar. 2003.

SUN MICROSYSTEMS. *Enterprise javabeans™ specification*, version 2.1, 2003. Disponível em: <http://www.java.sun.com/products/ejb/docs.html> Acesso em: 5 fev. 2004.

SUN MICROSYSTEMS. *JavaServer pages™ specification*, version 2.0, 2003. Disponível em: <http://java.sun.com/products/jsp/reference/api/index.html> Acessado em: 5 abr. 2004.

SUN MICROSYSTEMS. *Cloudscape server*. Disponível em: <http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Tools4.html> Acessado em: 5 abr. 2004.

SZYPERSKI, C. *Component software: beyond object-oriented programming*. Addison-Wesley, 1998.

THOMAS, A. *Enterprise javabeans™ technology: server component model for the Java™ plataform*. Patricia Seybold Group, dez. 1998. Relatório técnico. Disponível em: <http://www.ejbean.com/documents/downloads/white_paper.pdf> Acesso em: 16 nov. 2002.

TURINE, M. A. S.; ARAKAKI, A. A.; MONTEIRO, J. D. A.; BLANES, D. SIGSMOBILE: uma ferramenta para otimizar pesquisas de campo utilizando a tecnologia móvel pda em um sistema de informação de gestão social. In: CONGRESSO NACIONAL DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO — SUCESU'2004, 2004, Florianópolis-SC. A ser publicado.

VILJAMAA, J. Reverse engineering framework reuse interfaces. In: EUROPEAN SOFTWARE ENGINEERING CONFERENCE, 9., 2003; E ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON FOUNDATIONS OF SOFTWARE ENGINEERING, 10., 2003, Helsinki-Finland. *Proceedings of the 9th European Software Engineering Conference held jointly with 10th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. New York-USA: ACM Press, 2003. p. 217-226. Disponível em: <http://www.acm.org> (The ACM Digital Library) Acesso em: 11 fev. 2004.

YACOUB, S.; AMMAR, H.; MILI, A. Characterizing a software component. In: INTERNATIONAL WORKSHOP ON COMPONENT-BASED SOFTWARE ENGINEERING, 21., 1999a, Los Angeles, USA. *Anais eletrônicos…* Disponível em: <http://www.sei.cmu.edu/cbs/icse99/papers/34/34.htm> Acesso em: 16 nov. 2002.

YACOUB, S.; AMMAR, H.; MILI, A. A model for classifying component. In: INTERNATIONAL WORKSHOP ON COMPONENT-BASED SOFTWARE ENGINEERING, 21., 1999b, Los Angeles, USA. *Anais eletrônicos…* Disponível em: <http://www.sei.cmu.edu/cbs/icse99/papers/31/31.htm> Acesso em: 16 nov. 2002.

WANG, J. A. Towards component-basead software engineering. *The Journal of Computing in Small Colleges*, v. 16, n. 1, 2000.

WANG, N.; SCHMIDT, D. C.; O'RYAN, C. Overview of the corba component model. In: HEINEMAN, G. T.; COUNCILL, W. T. *Component-based software engineering*. ADDISON-WESLEY, 2001. p. 557-571.

WEINREICH, R.; SAMETINGER, J. Component models and component services: concepts and principles. In: HEINEMAN, G. T.; COUNCILL, W. T. *Component-based software engineering*. ADDISON-WESLEY, 2001. p. 33-48.

WERNER, C. M. L; BRAGA, R. M. M. *Desenvolvimento baseado em componente*. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 14., 2000, João Pessoa, PB. Disponível em: <http://www.cos.ufrj.br/~odyssey/publicacoes/SBES2000_TutorialDBC_Apresentacao.pdf> Acesso em: 21 out. 2002.

WERNER, C. M. L.; MURTA, L. G. P.; OLIVEIRA, H.; DANTAS, C. R. Um estudo sobre gerência de configuração de software aplicada ao desenvolvimento baseado em componentes. In: 3º WORKSHOP DE DESENVOLVIMENTO BASEADO EM COMPONENTES, 3., 2003, São Carlos-SP. *Anais*. São Carlos: Antônio Francisco do Prado-UFSCar, 2003. 1 CD-ROM.

WILLIAMS, J. The business case for components. In: HEINEMAN, G. T.; COUNCILL, W. T. *Component-based software engineering*. ADDISON-WESLEY, 2001. p. 79-97.