UNIVERSIDADE SALVADOR – UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO
MESTRADO PROFISSIONAL EM SISTEMAS E COMPUTAÇÃO

CLEVERSON SACRAMENTO DE OLIVEIRA

rasea

UMA SOLUÇÃO UNIFICADA PARA CONTROLE DE ACESSO CROSS-PLATAFORMA DE APLICAÇÕES COMERCIAIS

Salvador
2009

CLEVERSON SACRAMENTO DE OLIVEIRA

UMA SOLUÇÃO UNIFICADA PARA CONTROLE DE ACESSO CROSS-PLATAFORMA DE APLICAÇÕES COMERCIAIS

Dissertação apresentada ao Curso de Mestrado Profissional em Sistemas e Computação, Universidade Salvador – UNIFACS, como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. André Santanchè

Salvador
2009

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador – UNIFACS)

Oliveira, Cleverson Sacramento de
RASEA: uma solução unificada para controle de acesso para cross-plataforma de aplicações comerciais / Cleverson Sacramento de Oliveira. - 2009.
107 f.

Dissertação (Mestrado) – Universidade Salvador – UNIFACS. Curso de Mestrado em Sistemas e Computação, 2009.
Orientador: Prof. Dr. André Santaché.

1. Computadores – Controle de acesso. 2. Serviços da Web. I. Santaché, André, orient. II. Universidade Salvador – UNIFACS. III. Título.

CDD: 005.8

TERMO DE APROVAÇÃO

CLEVERSON SACRAMENTO DE OLIVEIRA

RASEA: UMA SOLUÇÃO UNIFICADA PARA CONTROLE DE ACESSO CROSS-PLATAFORMA DE APLICAÇÕES COMERCIAIS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador – UNIFACS, pela seguinte banca examinadora:

André Santanchè – Orientador _____
Doutor em Ciência da Computação pela Universidade Estadual de Campinas – UNICAMP
Universidade Salvador – UNIFACS

José Augusto Suruagy Monteiro _____
Doutor em Ciência da Computação pela University of California, Los Angeles – UCLA
Universidade Salvador – UNIFACS

Angelo Amâncio Duarte _____
Doutor em Informática pela Universidad Autonoma de Barcelona – UAB
Universidade Estadual de Feira de Santana – UEFS

Salvador, 13 de outubro de 2009

AGRADECIMENTOS

Este trabalho é o resultado do investimento de mais de 30 (trinta) meses em pesquisas, implementações, testes, análises e debates, envolvendo diversos profissionais responsáveis por arquitetura, projeto técnico, interface, modelagem, integração, segurança, gerência de projetos, qualidade de *software*, inovação, testes e suporte à infra-estrutura.

Agradeço especialmente ao Prof. Dr. André Santanchè que colaborou com diversas críticas construtivas e sugestões de melhoria, despertando questões imprescindíveis para a concisa evolução do projeto. Ofereço minha sincera gratidão por aceitar este novo desafio sempre com alta dose de motivação e entusiasmo.

Agradeço principalmente à equipe propulsora do projeto *open source* RASEA, composta por mim e meus grandes amigos Paulo Suzart, Robson Ataíde e Abraão Isvi, que dedicaram muitas madrugadas, feriados e finais de semana, além de abdicarem de muitas coisas não menos importantes, tornando possível a realização deste sonho. Somente vocês sabem o quão recompensador é contribuir para a evolução do nosso trabalho que já está sendo aplicado no dia-a-dia de muitas pessoas.

Agradeço à empresa Avansys Tecnologia que acreditou, proporcionou, investiu e que hoje colhe os frutos deste trabalho. Sem o seu apoio este trabalho não seria nada mais do que um mero devaneio.

Finalmente agradeço à minha companheira, parceira de todas as horas, colega de profissão e esposa Claudia Tedesco pelas críticas pontuais e incansáveis revisões deste trabalho. Agradeço ao meu símbolo de força e otimismo, que renova minhas energias com suas palavras: minha mãe. Serei eternamente grato a vocês duas pela enorme motivação, fazendo-me acreditar que tudo isso é possível.

Dedico este trabalho a todos que participaram direta ou indiretamente, em especial ao meu amigo e colega de trabalho Elvis Vaz. Tenho plena convicção que os objetivos não seriam alcançados sem a colaboração e dedicação de cada uma das pessoas envolvidas. Obrigado!

“Simplicidade é a sofisticação máxima.”

Leonardo da Vinci

RESUMO

A redundância do controle de acesso às aplicações comerciais faz parte da realidade das empresas. Cada sistema implantado nas organizações pode ter um controle de acesso próprio para garantir a confidencialidade da informação. Os efeitos negativos deste cenário são percebidos pelos usuários dos sistemas, que possuem diversas senhas, e pela própria empresa, que precisa garantir o controle por meio de processos e auditorias, onerando os custos operacionais. Com uma arquitetura de aplicações adequada, especializada na autenticação e autorização dos sistemas, é possível garantir a convergência do controle de acesso. Este trabalho tem como objetivo prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos. O projeto RASEA foi concebido como uma solução unificada para controle de acesso *cross*-plataforma de aplicações comerciais, implementando os conceitos abordados. Uma das principais contribuições, e característica marcante do projeto RASEA, é a simplicidade na concepção alcançada através das experiências adquiridas em mais de 30 (trinta) meses de pesquisas e implementações. Outra grande contribuição deste trabalho, e diferencial em relação às soluções de mercado, é o elemento 'agente' da arquitetura do projeto, que possibilita a extensibilidade e integração com diversas tecnologias e plataformas. Este trabalho não tem como objetivo substituir tecnologias existentes, e sim integrá-las. Para tanto, foi feito o levantamento e análise comparativa de ferramentas existentes no mercado. Para garantir o controle de acesso e a independência de plataforma, a solução proposta utiliza os conceitos RBAC e SOA. As idéias apresentadas neste trabalho foram validadas com simulações e testes estrategicamente alinhados ao objetivo do projeto. Deste modo o projeto RASEA foi capaz de promover uma economia de 31% (trinta e um por cento) nos custos do desenvolvimento de aplicações comerciais e a simplificação na administração do controle de acesso dos sistemas corporativos.

Palavras-chave: Controle de Acesso. Autorização. RBAC. *Web Services*. SOA.

ABSTRACT

Redundant access control to applications is an important issue for enterprises nowadays. Each deployed system inside an organization has its own access control to grant right access for information. The negative effects of this scenario is perceived by the end users, dealing with too many passwords, and by the organization that needs to ensure control processes and auditing, increasing the operational costs. Appropriate application architecture, specialized in authentication and authorization, allows convergence in access control. This work aims to providing a service-oriented solution to support unified access control to applications by users, based on existing concepts and technologies to achieve interoperability in heterogeneous software environments. The RASEA project was conceived as a solution to enforce unified cross-platform access control for enterprise applications, embodying the presented concepts. One of the main contributions, and noteworthy characteristic of RASEA, is its simplicity in project design achieved through more than 30 (thirty) months of researchs and implementations. Another important contribution of this work, and a key differentiation compared to market solutions, is the 'agent' element from the project architecture, allowing extensibility and integration of several technologies and platforms. This work is not intended to be a replacement of existing technologies, but integrates with them. Therefore, a comparison between RASEA and the existing solutions off-the-shelf is provided. To ensure access control and platform independence, the proposed solution uses the concepts of RBAC and SOA. The ideas shown in this work were validated with simulations and tests strategically aligned with the goals of this project. Thus RASEA was able to promote 31% (thirty one percent) savings on application development costs and simplifying the access control administration for corporate systems.

Keywords: Access Control. Authorization. RBAC. Web Services. SOA.

LISTA DE FIGURAS

Figura 1.1 – Divisão lógica da aplicação 13
Figura 1.2 – Descentralização no controle de acesso 14
Figura 1.3 – Controle de acesso unificado 14
Figura 1.4 – Logotipo do projeto RASEA 16
Figura 2.1 – Controle de permissões do *Microsoft Windows* 19
Figura 2.2 – Relacionamento RBAC 21
Figura 2.3 – Modelo de Referência do *Core* RBAC 22
Figura 2.4 – Especificação Funcional do *Core* RBAC 23
Figura 2.5 – Modelo de Referência do *Hierarchical* RBAC 24
Figura 2.6 – Modelo de Referência do *Constrained* RBAC 24
Figura 2.7 – Uso do XML-RPC 26
Figura 2.8 – Elementos básicos do *Web Service* 27
Figura 2.9 – Uso do JAX-WS 28
Figura 2.10 – Visão geral da avaliação das ferramentas 30
Figura 2.11 – Avaliação da ferramenta *Seam Security* 31
Figura 2.12 – Redundância no controle de acesso do *Seam Security* 32
Figura 2.13 – Arquitetura do projeto MACA 33
Figura 2.14 – Avaliação da ferramenta MACA 34
Figura 2.15 – Arquitetura do OES 34
Figura 2.16 – Avaliação da ferramenta OES 35
Figura 3.1 – Arquitetura dos Sistemas Internos 37
Figura 3.2 – Arquitetura do projeto Inscrição Online 39
Figura 3.3 – Economia na utilização do S4A 40
Figura 3.4 – Macro-cronograma dos projetos: AvanSEG, S4A e RASEA 41
Figura 3.5 – Problema de compartilhamento de papéis entre aplicações 43
Figura 3.6 – Comparativo entre os projetos: AvanSEG, S4A e RASEA 44
Figura 3.7 – Exemplo de apresentação das autorizações de forma matricial 45
Figura 4.1 – Principais elementos e perspectivas da arquitetura do projeto RASEA 46
Figura 4.2 – Avaliação do projeto RASEA 47
Figura 4.3 – Visão geral da arquitetura do projeto RASEA 49
Figura 4.4 – Diagrama de Entidade-relacionamento do RASEA 50
Figura 4.5 – Diagrama de Classes de Domínio do RASEA 51
Figura 4.6 – Comunicação utilizando padrões abertos 52
Figura 4.7 – Integração com o serviço de gerenciamento 55
Figura 4.8 – Integração com base de usuários legada 55
Figura 4.9 – Arquitetura do servidor RASEA 56
Figura 4.10 – Estratégia de armazenamento das credenciais de usuários 57
Figura 4.11 – Arquitetura em camadas do servidor RASEA 58
Figura 4.12 – Balanceamento de carga 59
Figura 4.13 – Tipos de agente 61
Figura 4.14 – Processo de autenticação e armazenamento dos privilégios de acesso 62
Figura 4.15 – Processo de verificação e autorização de acesso 62
Figura 4.16 – Agente do servidor RASEA consumindo os serviços de controle de acesso 63
Figura 5.1 – Comparativo entre as ferramentas existentes e o RASEA 67

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
AD	Active Directory
ALES	AquaLogic Enterprise Security
ANSI	American National Standards Institute
APF	Análise de Ponto de Função, ou FPA
API	Application Programming Interface
AS	Application Server
ASP	Active Server Pages
CRM	Customer Relationship Management
CORBA	Common Object Request Broker Architecture
DAC	Discretionary Access Control
DAO	Data Access Object
DER	Diagrama de Entidade-relacionamento
DSD	Dynamic Separation of Duty Relations
ERP	Enterprise Resource Planning
FPA	Function Point Analysis, ou APF
FSF	Free Software Foundation
GNU	GNU's Not Unix
GPL	GNU General Public License
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I18N	InternationalizatioN
IDE	Integrated Development Environment
IDL	Interface Description Language
IEC	International Electrotechnical Commission
IFPUG	International Function Point Users Group
IIS	Internet Information Services
INCITS	InterNational Committee for Information Technology Standards
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition, ou JEE
JEE	Java Enterprise Edition, ou J2EE
JAAS	Java Authentication and Authorization Service
JAX-RPC	Java APIs for XML based RPC
JAX-WS	Java API for XML-Based Web Services
JOSSO	Java Open Single Sign-On
L10N	LocalizatioN
LDAP	Lightweight Directory Access Protocol
LDAPS	Secure LDAP
LGPL	GNU Lesser General Public License
MAC	Mandatory Access Control
MACA	Middleware de Autenticação e Controle de Acesso
NBR	Norma ABNT
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OES	Oracle Entitlements Server

OO	Orientação a objetos
PA	Permission Assignment
PAP	Policy Administration Point
PDP	Policy Decision Point
PF	Ponto de Função
PHP	PHP: Hypertext Preprocessor
PIM	Platform-Independent Model
RASEA	cRoss-plAtform accesS control for Enterprise Applications
RBAC	Role-Based Access Control
RH	Role Hierarchy
RPC	Remote Procedure Call
S4A	Security for Applications
SAML	Security Assertion Markup Language
SSD	Static Separation of Duty Relations
SDK	Software Development Kit
SOA	Service-Oriented Architecture
SSO	Single Sign-On
SVN	Sistema de controle de versão: SubVersioN
TI	Tecnologia da Informação
UA	User Assignment
UDDI	Universal Description, Discovery, and Integration
UNIFACS	Universidade Salvador
UML	Unified Modeling Language
USP	Universidade de São Paulo
UTF	Unicode Transformation Format
W3C	The World Wide Web Consortium
WS	Web Service
XACML	eXtensible Access Control Markup Language

SUMÁRIO

1 INTRODUÇÃO 12
1.1 CONTEXTUALIZAÇÃO 12
1.2 MOTIVAÇÃO 13
1.3 OBJETIVO 15
1.4 CONTRIBUIÇÕES 15
1.5 ESTRUTURA DO DOCUMENTO 17
2 SERVIÇOS E CONTROLE DE ACESSO 18
2.1 AUTENTICAÇÃO E AUTORIZAÇÃO 18
2.2 RBAC 20
2.3 SOA E *WEB SERVICES* 25
2.4 FERRAMENTAS EXISTENTES 29
3 EVOLUÇÃO DO PROJETO 36
3.1 CENÁRIO A – SISTEMAS INTERNOS 36
3.2 CENÁRIO B – PROJETO INSCRIÇÃO ONLINE 38
3.3 CENÁRIO C – FÁBRICA DE *SOFTWARE* 39
3.4 CRONOLOGIA E EXPERIÊNCIAS ADQUIRIDAS 41
4 PROJETO RASEA 46
4.1 VISÃO GERAL 46
4.2 MODELO DE DADOS 50
4.3 INDEPENDÊNCIA DE PLATAFORMA E INTEGRAÇÃO 51
4.4 SERVIDOR RASEA 56
4.5 AGENTE RASEA 60
5 CONSIDERAÇÕES FINAIS 66
5.1 RESULTADOS OBTIDOS 66
5.2 TRABALHOS FUTUROS 69
REFERÊNCIAS 71
APÊNDICE A – Arquivo de configuração do servidor RASEA 76
APÊNDICE B – Arquivo de configuração do agente RASEA 80
APÊNDICE C – Módulo de serviços do Servidor RASEA 81
APÊNDICE D – Antigo módulo administrativo do servidor RASEA 98
APÊNDICE E – Atual módulo administrativo do servidor RASEA 101
APÊNDICE F – *Script* de migração do S4A para o RASEA 105
APÊNDICE G – Configuração do agente *Seam Security* em uma aplicação parceira 107

1 INTRODUÇÃO

A redundância do controle de acesso às aplicações comerciais fazem parte da realidade das organizações. Os efeitos negativos deste cenário são percebidos pelos usuários dos sistemas e pela própria empresa, que mantém o controle redundante, onerando os custos operacionais. Com uma arquitetura adequada, orientada ao controle de acesso às aplicações, é possível garantir a unicidade do serviço. Este capítulo delimita o escopo do objeto de estudo, apresenta a motivação, o objetivo do trabalho e suas contribuições.

1.1 CONTEXTUALIZAÇÃO

O termo 'aplicação comercial' é utilizado para representar um *software* comercial adquirido ou desenvolvido por uma organização, o qual possui recursos consumidos por usuários. Presman (2002, p. 10) define aplicações comerciais como sistemas que manipulam e modificam dados transacionais, consolidando informações que auxiliam na tomada de decisões gerenciais relevantes para o negócio da organização. São exemplos de aplicações comerciais: sistemas de folha de pagamento, sistemas de compras, sistemas de vendas, sistemas de tarifação, ERP, CRM, sistemas específicos para o negócio da empresa, dentre outros.

Por manipular informações empresariais, o acesso às aplicações comerciais geralmente é restrito a usuários autorizados, que possuem privilégios para executar determinadas operações nos sistemas. O controle de acesso baseia-se em regras incorporadas ao sistema durante o processo de desenvolvimento do *software*, utilizando padrões e boas práticas (BUEGE; LAYMAN; TAYLOR, 2003) para auxiliar na segurança da aplicação. De acordo com Kern (2004, p. 87), existem poucas publicações a respeito de segurança de aplicações, se comparadas à vasta literatura sobre segurança de redes.

Sob a ótica deste trabalho, uma aplicação comercial pode ser logicamente subdividida em dois blocos, conforme a Figura 1.1. Na primeira parte, 'A' representa as funcionalidades da aplicação, composta por telas, regras de negócio, relatórios e integrações com outros sistemas.

O elemento ‘B’ simboliza a base de dados que armazena as informações associadas às regras de negócio da aplicação. Na segunda parte, ‘C’ simboliza o controle de acesso, responsável por prevenir o acesso não autorizado ao sistema ou às suas partes, bem como estabelecer o conjunto de operações que cada usuário pode executar, de acordo com a NBR ISO/IEC 27001 (ABNT, 2006, p. 80). A base de dados (D) contém as informações suficientes e necessárias para garantir o controle de acesso à aplicação.

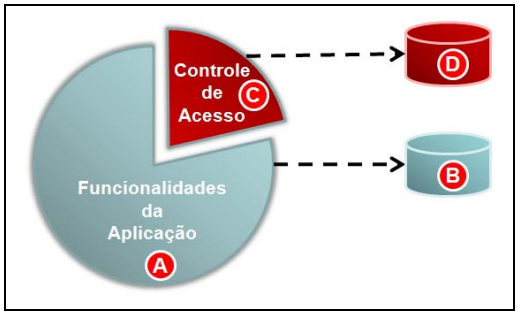

Figura 1.1 – Divisão lógica da aplicação

Não faz parte do escopo deste trabalho tratar sobre segurança de redes, este assunto é vastamente abordado em outros trabalhos. Também não é o foco aprofundar no tema gestão de segurança da informação, abordado pela NBR ISO/IEC 27001 (ABNT, 2006). O escopo deste trabalho delimita-se ao controle de acesso às aplicações comerciais empregando fundamentalmente os conceitos *Role-Based Access Control* (RBAC) e *Service-Oriented Architecture* (SOA).

1.2 MOTIVAÇÃO

Os autores Gaedke, Meinecke e Nussbaumer (2005, p. 1156) afirmam que diversas empresas costumam projetar diferentes módulos de controle de acesso para cada sistema implementado, gerando gastos desnecessários decorrentes do retrabalho e da dificuldade em gerenciar as políticas de segurança. Supondo que uma empresa fictícia, que possua 20 (vinte) aplicações comerciais no seu inventário, contrate 10 (dez) novos funcionários. Qual seria o custo estimado para registrá-los nos sistemas da empresa? Qual seria o custo para controlar de forma descentralizada o acesso às aplicações? No momento da demissão, qual o custo envolvido para revogar o acesso a cada sistema? Seja qual for o custo, notoriamente trata-se um gasto desnecessário.

A Figura 1.2 ilustra as aplicações ‘A’, ‘B’, ‘C’, ‘D’ e ‘E’ de uma organização em que cada uma possui o seu próprio controle de acesso. Nota-se que cada módulo de segurança mantém uma base de dados replicada desnecessariamente. Este cenário tem como vantagem a autonomia do controle de acesso das aplicações. Por outro lado, as desvantagens, do ponto de vista do controle de acesso, são: (i) redundância desnecessária do módulo de controle de acesso e das respectivas bases de dados; (ii) possível falta de padronização das bases de dados e; (iii) redundância desnecessária no gerenciamento.

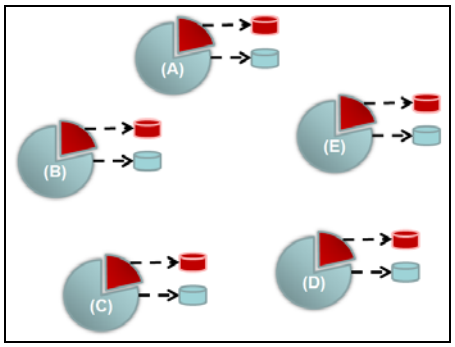

Figura 1.2 – Descentralização no controle de acesso

Uma possível solução para este problema é a unificação do controle de acesso. No modelo apresentado na Figura 1.3, as aplicações ‘A’, ‘B’, ‘C’, ‘D’ e ‘E’ integram com um serviço unificado (F), evitando a redundância desnecessária do módulo de controle de acesso. Cada aplicação possui um ponto de acesso único (ALUR; CRUPI; MALKS, 2003, p. 164), que, com base nas informações disponibilizadas pelo serviço (F), concede ou revoga o acesso ao sistema. A utilização de *Web Services* garante a interoperabilidade entre o serviço de controle de acesso e as aplicações de forma independente de plataforma.

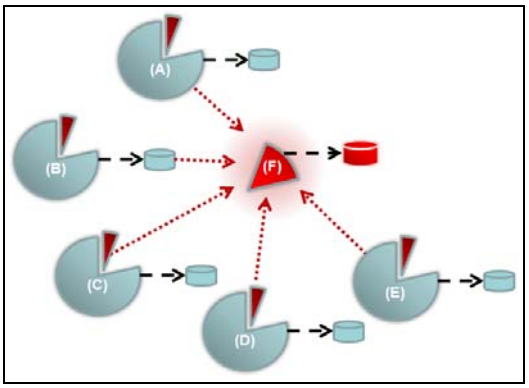

Figura 1.3 – Controle de acesso unificado

Dessa maneira, é possível manter uma base de dados unificada, responsável pelo armazenamento das informações referentes ao controle de acesso e às credenciais dos

usuários. A modelagem da base de dados deve seguir o RBAC: um padrão eficaz para garantir o controle de acesso às aplicações comerciais (SANDHU; FERRAIOLO; KUHN, 2000, p. 1). Com o intuito de facilitar a manipulação dos dados, o componente 'F' (Figura 1.3) deve disponibilizar uma interface gráfica, contendo telas para cadastro de recursos e gerenciamento de permissões.

1.3 OBJETIVO

Este trabalho tem como principal objetivo prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos.

Os objetivos específicos são os seguintes:

a) Implementar a solução proposta e disponibilizá-la à comunidade por meio de uma licença *open source*;

b) Garantir que a arquitetura proposta beneficie aplicações em diferentes plataformas.

1.4 CONTRIBUIÇÕES

Para auxiliar na resolução dos problemas decorrentes do controle de acesso descentralizado, concebeu-se uma abordagem de arquitetura baseada em SOA. A validação do modelo ocorreu em laboratório com a implementação do projeto-piloto AvanSEG. Posteriormente, o modelo foi implantado em ambiente de produção, ganhando o nome *Security for Applications* (S4A): uma versão estável e evoluída do projeto-piloto. Os projetos AvanSEG e S4A foram idealizados, gerenciados, projetados e codificados pelo autor deste trabalho e sua equipe nas instalações da Avansys Tecnologia (http://www.avansys.com.br). Com o reuso do S4A em regime de Fábrica de *Software*, notou-se uma redução de 31% (trinta e um porcento) nos custos do desenvolvimento de aplicações comerciais.

A partir da experiência adquirida nos projetos AvanSEG e S4A, que envolveram diversas interações com clientes e refinamentos sucessivos da solução, foram definidos os requisitos essenciais para a construção do projeto RASEA (http://www.rasea.org). Este nome é um acrônimo do inglês *c**R**oss-pl**A**tform acces**S** control for **E**nterprise **A**pplications*, que pode ser traduzido como: controle de acesso *cross*-plataforma para aplicações comerciais. Regido por licenças *open source* (GPL, 2007; LGPL, 2007), o RASEA está disponível publicamente no *SourceForge* (http://sourceforge.net/projects/rasea/). Até onde se sabe, não existem outras ferramentas *open source* que façam exatamente o que está sendo proposto neste trabalho.

Figura 1.4 – Logotipo do projeto RASEA

Um dos diferenciais do RASEA, fruto do processo contínuo de interação com o usuário e sucessivo refinamento, é ser um produto que reúne elementos essenciais para o controle de acesso, combinando uma arquitetura flexível e extensível com simplicidade na concepção. A simplicidade, além de facilitar o gerenciamento e desenvolvimento de aplicações, pode melhorar o controle da segurança, uma vez que possibilita uma fácil visualização do estado global do sistema.

Não faz parte do objetivo deste trabalho substituir tecnologias existentes, muito menos reinventar algo já consolidado. O cerne do que está sendo proposto é a integração de tecnologias e a utilização de boas práticas (GAMMA, 2000) para prover uma solução unificada para controle de acesso de aplicações comerciais. Durante a implementação deste trabalho, pôde-se constatar a flexibilidade do modelo ao agregar novos conceitos, tecnologias e paradigmas em prol do resultado final.

Este trabalho apresenta como grande contribuição o relato das experiências adquiridas durante o período de mais de 30 (trinta) meses de concepção, implementação e validação dos projetos AvanSEG, S4A e RASEA, envolvendo laboratórios e ambientes de produção. Durante a elaboração deste trabalho, diversas melhorias – sugeridas por 12 (doze) usuários responsáveis pela administração de acesso das organizações envolvidas – contribuíram para a evolução da solução. Como resultado da extensa experiência adquirida, este trabalho apresenta a arquitetura do projeto RASEA, que, em contraste com as soluções de mercado, almeja a simplicidade no gerenciamento das permissões e no controle de acesso.

O projeto RASEA contribui para a consolidação do padrão RBAC em soluções de controle de acesso unificado a múltiplas aplicações. Este trabalho explora a integração do RBAC com *Web Services*, possibilitando o controle de acesso de forma independente de plataforma. Adicionalmente, o RASEA é capaz de beneficiar: (i) empresas de desenvolvimento de *software* e profissionais autônomos, com a redução de custos no desenvolvimento de sistemas e; (ii) organizações cliente, com a unificação da base de usuários e do controle de acesso às aplicações, promovendo simplicidade no gerenciamento de permissões.

Outra grande contribuição deste trabalho, e diferencial em relação às soluções de mercado para controle de acesso, é o elemento 'agente' da arquitetura do projeto. O agente RASEA possibilita a extensibilidade da arquitetura, beneficiando a integração com diversas tecnologias e plataformas. Foi possível observar que, graças ao agente, é possível controlar o acesso não somente de aplicações comerciais, mas de sistemas em geral que necessitam restringir o acesso aos seus recursos.

1.5 ESTRUTURA DO DOCUMENTO

Este documento está dividido em cinco capítulos. Neste primeiro capítulo foram apresentadas as motivações, os objetivos e as contribuições deste trabalho. O segundo capítulo apresenta o referencial teórico indispensável para o entendimento do projeto, abordando temas como autenticação, autorização, controle de acesso, RBAC, SOA e *Web Services*, bem como a análise das ferramentas existentes. O terceiro capítulo trata das experiências adquiridas durante a concepção, implementação e validação dos projetos envolvidos neste trabalho. No quarto capítulo, são explanados os fundamentos do projeto RASEA. No quinto capítulo são apresentadas as considerações finais, analisando os resultados obtidos e propondo trabalhos futuros.

2 SERVIÇOS E CONTROLE DE ACESSO

O problema gerado pela descentralização no controle de acesso às aplicações comerciais motivou esta pesquisa no sentido de unificar o controle e garantir a independência de plataforma. Este capítulo trata dos padrões, modelos, conceitos, paradigmas, tecnologias e ferramentas relacionadas com a solução proposta, tais como: autenticação, autorização, controle de acesso, RBAC, SOA e *Web Services*.

2.1 AUTENTICAÇÃO E AUTORIZAÇÃO

É comum que pessoas não habituadas com o assunto confundam os termos 'autenticação' e 'autorização'. Resumidamente, a autenticação enfoca na identificação enquanto a autorização define o que é possível fazer. Para controlar o acesso a um recurso, é preciso que haja a identificação do indivíduo (autenticação) e a possibilidade de acessar o que se deseja (autorização).

De acordo com Silva (2004, p. 66), "autenticação é a capacidade de garantir que alguém, ou alguma coisa, é de fato quem diz ser, dentro de um contexto definido". Existem diversas técnicas e tecnologias que auxiliam neste processo, tais como: OpenID (http://www.openid.net), Certificado Digital e *Single Sign-On* (http://www.opensso.org) (SSO). O enfoque deste trabalho está no processo de autorização e controle de acesso. Para o processo de autenticação, entretanto, serão utilizadas soluções já estabelecidas por terceiros e, por este motivo, questões relativas à autenticação não serão detalhadas nesta revisão.

Garantir a confidencialidade da informação é uma necessidade latente nas aplicações comercias, por isso somente usuários autorizados devem ter acesso ao sistema. Assegurar a autorização significa que ninguém poderá executar ações sobre um recurso caso não tenha permissão (JAAS, 2001). Segundo Barker (2008, p. 144) uma permissão é composta por uma ação e um recurso, par '(A, R)', sendo que 'A' representa a ação e 'R' o recurso. Quando uma permissão é concedida a um usuário, ou grupo de usuários, tem-se uma autorização. Alguns autores, tais como Sandhu, Ferraiolo e Kuhn (2000), utilizam o termo 'operações' para referenciar as 'ações' e o termo 'objeto' para referenciar os 'recursos'.

A Figura 2.1 ilustra a funcionalidade do *Microsoft Windows* para concessão de permissões de grupos de usuários (papéis) de uma organização. O item 'A' representa um recurso: uma pasta ou um arquivo. Na solução apresentada neste trabalho, um recurso simboliza uma tela ou um relatório de uma aplicação comercial, por exemplo. O elemento 'B' representa os grupos que têm permissão ao recurso 'A'. As operações ou ações que podem ser executadas em A são ilustradas pelo item 'C'. A associação entre a operação 'C' e o recurso 'A', atribuída ao grupo de usuários 'B', representa uma autorização 'D'.

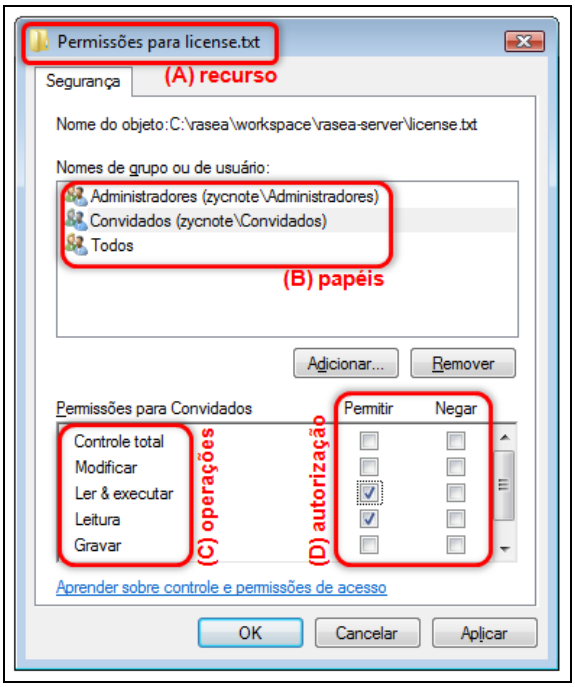

Figura 2.1 – Controle de permissões do *Microsoft Windows*

A definição de controle de acesso segundo a norma internacional ISO/IEC 10181-3 (ISO, 1996, p. 1, tradução nossa), que trata sobre *framework* para controle de acesso, é a seguinte: "o processo de determinar quais utilizações dos recursos [...] são permitidas, [...] prevenindo o acesso não autorizado, é chamado de controle de acesso". Com base neste conceito, este trabalho considera que o controle de acesso de uma aplicação comercial abrange desde o gerenciamento até o processo de verificação das permissões. Para garantir o tratamento e o controle efetivo das requisições de acesso ao sistema, este trabalho considera que os usuários estejam devidamente autenticados ao executarem operações sobre os recursos das aplicações (BARKER, 2008).

A regra para garantir o controle proposto neste trabalho segue a premissa de que tudo é proibido, a menos que explicitamente permitido, conforme a norma ISO/IEC 17799 (ABNT, 2005, p. 66), ao invés da regra mais fraca onde tudo é permitido, caso não esteja

expressamente proibido. Desta forma, o usuário somente terá acesso ao recurso caso o administrador de segurança autorize-o explicitamente, assegurando a confidencialidade necessária das informações mantidas pelas aplicações comerciais.

Autorizações podem ser concedidas ou negadas aos usuários. Esta característica é também conhecida como autorizações positivas ou negativas (BERTINO, 2001, p. 42). Suponha um usuário participante do grupo 'gerentes' (positivamente autorizado ao recurso 'X') que, simultaneamente, faz parte do grupo '*hackers*' (negativamente autorizado ao recurso 'X'). Caso este usuário tente acessar 'X', a proibição prevalecerá sobre a concessão e a requisição será recusada.

Os conceitos explanados até então foram: autenticação, autorização, controle de acesso, recursos, operações e permissões. A arquitetura do projeto RASEA, proposta neste trabalho, utiliza estas definições em conjunto com um modelo para controle de acesso de aplicações comerciais, elucidado a seguir.

2.2 RBAC

A sigla RBAC, do inglês *Role-Based Access Control*, significa controle de acesso baseado em papéis. Um papel representa um grupo de usuários de uma organização, que reúne pessoas do mesmo cargo, função, responsabilidade, atividade ou qualquer outro critério de categorização. Para o RBAC, segundo Sandhu, Ferraiolo e Kuhn (2000, p. 1), este conceito possibilita estabelecer permissões baseadas em papéis funcionais da organização, associando usuários a um ou mais papéis. A abordagem orientada a papéis é o diferencial do modelo RBAC em relação aos seus antecessores *Mandatory Access Control* (MAC) e *Discretionary Access Control* (DAC).

Por mais de 30 anos, o conceito de papéis é empregado em sistemas informatizados, contudo apenas em meados da década de 1990 atingiu a maturidade (FERRAIOLO, 2001, p. 225). O modelo RBAC é considerado mais adequado que o MAC e DAC para utilização em ambientes empresariais (BEZNOSOV; DENG, 1999, p. 19). O MAC garante o controle de acesso de acordo com rótulos de segurança aplicados a cada usuário, enquanto o DAC baseia-se na concessão de permissões por usuário (SANDHU, 1996, p. 4). Os modelos MAC e DAC

enfocam no usuário e não no seu papel funcional, dificultando o gerenciamento das políticas de segurança. De acordo com Beznosov e Deng (1999, p. 19), o maior propósito do RBAC é facilitar a administração do controle de acesso aos sistemas em ambientes corporativos.

Por oferecer redução de complexidade, contenção de custos e diminuição de erros na concessão de permissões, a demanda pelo RBAC aumentou, sendo utilizado em produtos como: bancos de dados, sistemas operacionais e gerenciamento de sistemas. Devido à diversidade de implementações do RBAC, o *National Institute of Standards and Technology* (NIST) consolidou as estratégias em um único modelo (SANDHU; FERRAIOLO; KUHN, 2000, p. 2), dando origem ao padrão ANSI INCITS 359-2004 (ANSI, 2004). A arquitetura do projeto RASEA tem como base o padrão ANSI INCITS 359-2004, doravante referenciado apenas como RBAC.

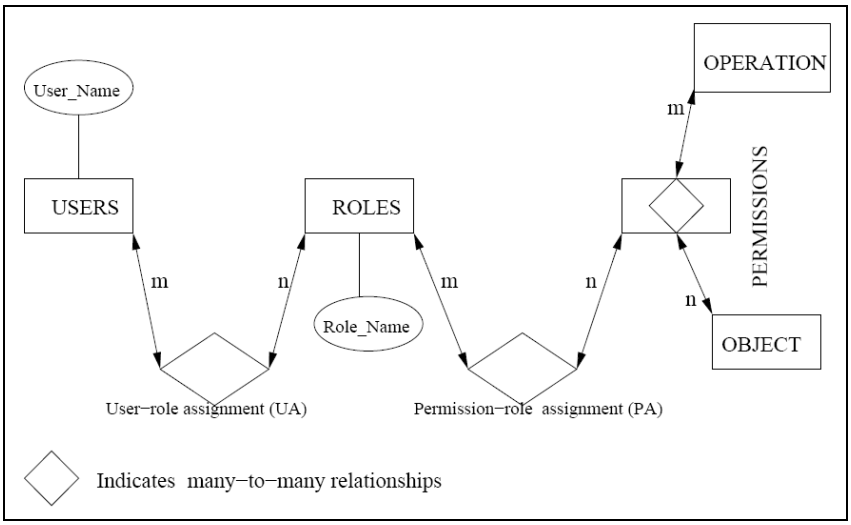

Figura 2.2 – Relacionamento RBAC
Fonte: Wang (2006, p. 2).
Nota: Adaptação do autor.

Na Figura 2.2, Wang (2006) apresenta um Diagrama de Entidade-relacionamento (DER) ilustrando a associação entre as entidades do modelo RBAC, onde o item *object* simboliza um recurso protegido do sistema (KERN, 2004, p. 91). A entidade *operation* representa as ações que podem ser executadas pelos usuários autenticados. Ações associadas a recursos resultam em *permissions*. Os usuários (*users*) desempenham papéis (*roles*), que por sua vez agrupam usuários. Esta associação é representada pelo relacionamento *user-role assignment*. A concessão de permissões aos *roles*, também conhecida como autorização, é definida pela relação *permission-role assignment*.

O padrão RBAC é composto por componentes abordados pelo Modelo de Referência e pela Especificação Funcional. O Modelo de Referência estabelece rigorosamente as entidades RBAC (Figura 2.2) e seus relacionamentos, tais como: *users*, *roles*, *permissions*, *operations, objects, user assignment* e *permission assignment*. É também responsabilidade do Modelo de Referência definir os termos e nomenclaturas utilizadas pela Especificação Funcional. A Especificação Funcional estabelece as funcionalidades do RBAC, por meio de interfaces bem definidas, responsáveis por criar, excluir e manter os elementos do Modelo de Referência e seus relacionamentos. Ou seja, a Especificação Funcional define as operações suportadas pelo RBAC (ANSI, 2004).

O Modelo de Referência e a Especificação Funcional são definidos em três segmentos, conhecidos como componentes RBAC (FERRAIOLO, 2001, p. 228), que são: (i) *Core* RBAC; (ii) *Hierarchical* RBAC; (iii) *Constrained* RBAC. Este trabalho se concentrará no componente *Core* RBAC, por tratar com mais aprofundamento os elementos fundamentais do controle de acesso. Será apresentado o *Hierarchical* RBAC, que possibilita a hierarquia de papéis, e o *Constrained* RBAC, que trata sobre o conflito de interesses no estabelecimento de papéis.

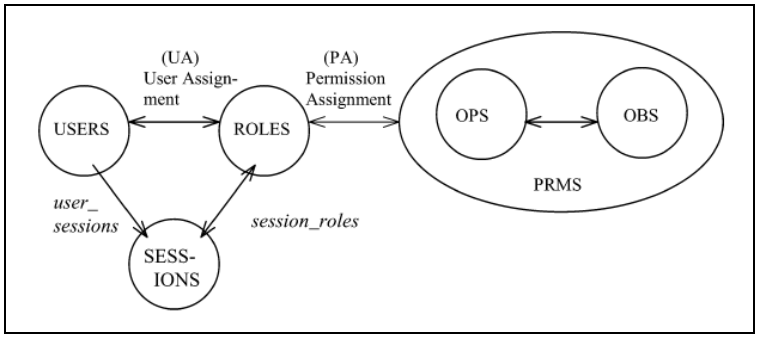

Figura 2.3 – Modelo de Referência do *Core* RBAC
Fonte: ANSI (2004).

O componente *Core* RBAC fundamenta-se no conceito básico do RBAC, que, conforme Ferraiolo (2001, p. 228), os usuários são membros de papéis (ou grupos), as permissões são concedidas aos papéis, e, conseqüentemente, os usuários adquirem permissões por serem integrantes dos papéis. O Modelo de Referência do *Core* RBAC – ilustrado na Figura 2.3 – estabelece que a relação entre as entidades 'usuários' e 'papéis' necessita ser do tipo muitos para muitos. Além disso, deve ser possível associar usuários aos papéis e vice-versa. Da mesma forma, a relação entre papéis e permissões deve ser do tipo muitos para muitos. Na Figura 2.3, a entidade *sessions* e suas relações (*user_sessions* e *session_roles*) representam

sessões de usuários autenticados fazendo uso do sistema sob o controle de acesso. As entidades 'OPS', 'OBS' e 'PRMS' representam os elementos *operations*, *objects* e *permissions*, respectivamente.

Core RBAC

		nome	descrição
comandos e funções	administrativos	*AddUser*	Cria um novo usuário.
		DeleteUser	Exclui um usuário existente.
		AddRole	Cria um novo papel.
		DeleteRole	Exclui um papel existente.
		AssignUser	Associa um usuário a um papel.
		DeassignUser	Desassocia um usuário a um papel.
		GrantPermission	Concede permissão para um papel.
		RevokePermission	Revoga permissão para um papel.
	apoio ao sistema	*CreateSession*	Cria uma nova sessão.
		DeleteSession	Deleta uma sessão existente.
		AddActiveRole	Adiciona um papel a uma sessão existente.
		DropActiveRole	Remove um papel de uma sessão existente.
		CheckAccess	Indica se o usuário da sessão possui autorização para executar uma determinada operação em um recurso do sistema.
	consulta	*AssignedUsers*	Lista os usuáros associados a um papel.
		AssignedRoles	Lista os papéis associados a um usuário.
		RolePermissions	Lista as permissões concedidas a uma papel.
		UserPermisssions	Lista as permissões concedidas a um usuário.
		SessionRoles	Lista os papéis associados a uma sessão.
		SessionPermissions	Lista todas as permissões associadas a uma sessão.
		RoleOperationsOnObject	Lista as operações que um papel pode executar em um determinado recurso.
		UserOperationsOnObject	Lista as operações que um usuário pode executar em um determinado recurso.

Figura 2.4 – Especificação Funcional do *Core* RBAC
Fonte: ANSI (2004, p. 11–17).

A Figura 2.4 representa as operações suportadas pelo componente *Core* RBAC agrupadas em três categorias, conforme o padrão ANSI INCITS 359-2004 (ANSI, 2004, p. 11), que são: (i) comandos administrativos, responsáveis pela criação e manutenção dos elementos RBAC e suas relações; (ii) funções de apoio ao sistema, as quais manipulam informações das sessões de usuários e executam o controle de acesso ao sistema e; (iii) funções de consulta, que recuperam informações referentes aos elementos RBAC e suas relações.

O *Hierarchical* RBAC acrescenta ao *Core* RBAC o conceito de hierarquia de papéis, onde um papel pode herdar as autorizações de outro papel (SANDHU; FERRAIOLO; KUHN, 2000, p. 3). Supondo a existência de um papel denominado 'Gerência', o qual possui determinados privilégios que devem ser aplicados para todas as gerências da organização, basta que a 'Gerência de Recursos Humanos' herde de 'Gerência' para que todas as permissões sejam também atribuídas à 'Gerência de Recursos Humanos'. O componente *Hierarchical* RBAC contribui com a relação *role hierarchy* (Figura 2.5) para o Modelo de Referência e com quatro

operações administrativas para a Especificação Funcional, responsáveis pela manipulação da relação *role hierarchy*, que são: (i) *AddInheritance*; (ii) *DeleteInheritance*; (iii) *AddAscendant* e; (iv) *AddDescendant* (FERRAIOLO, 2001, p. 244; ANSI, 2004, p. 17–18).

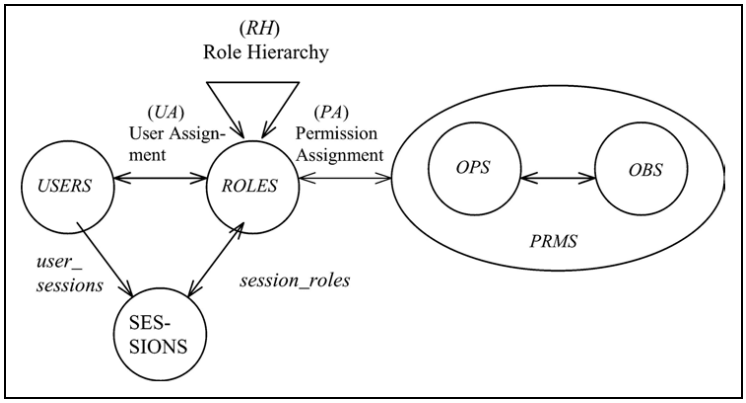

Figura 2.5 – Modelo de Referência do *Hierarchical* RBAC
Fonte: ANSI (2004, p. 6).

O componente *Constrained* RBAC acrescenta ao *Hierarchical* RBAC o tratamento de conflitos de interesses no estabelecimento de papéis aos usuários. Com o *Constrained* RBAC é possível estabelecer regras que impossibilitem que um usuário seja simultaneamente associado ao papel solicitante e aprovador de orçamentos, por exemplo. O *Constrained* RBAC é composto por dois elementos que contribuem com o Modelo de Referência (Figura 2.6), que são: (i) *Static Separation of Duty Relations* (SSD), aplicado às relações *user assignment* (UA) e *role hierarchy* (RH) e; (ii) *Dynamic Separation of Duty Relations* (DSD), aplicado à relação *session_roles*.

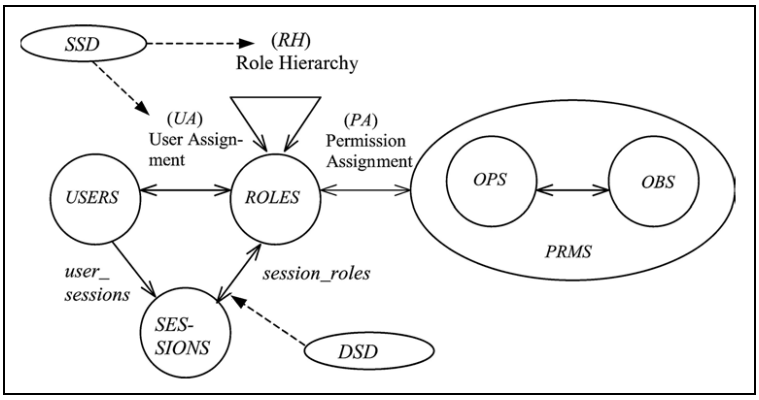

Figura 2.6 – Modelo de Referência do *Constrained* RBAC
Fonte: ANSI (2004, p. 9–10).
Nota: Adaptação do autor.

O SSD é aplicado diretamente nas operações administrativas *AssignUser* e *DeassignUser* da Especificação Funcional do *Core* RBAC bem como nas quatro operações administrativas do

Hierarchical RBAC. Em contrapartida, o DSD é aplicado nas funções de apoio ao sistema *AddActiveRole* e *DropActiveRole* da Especificação Funcional do *Core* RBAC. O SSD em conjunto com o DSD incorporam à Especificação Funcional do RBAC oito comandos administrativos e seis funções de consulta (FERRAIOLO, 2001; ANSI, 2004).

O termo '*separation*' é utilizado pelo SSD e DSD para representar a impossibilidade de associar regras conflitantes a um determinado usuário. O SSD é considerado estático por ser aplicado à relação entre usuários e papéis, de forma independente da sessão. O DSD tem o caráter dinâmico por tratar os conflitos de interesse na sessão do usuário. Com o DSD, é possível estabelecer regras que possibilitem que um usuário seja 'solicitante' e 'aprovador' simultaneamente. Porém, caso tenha atuado previamente como 'solicitante' em uma determinada sessão, o usuário não mais terá permissões de 'aprovador' na mesma sessão. Esta característica dinâmica é oriunda do DSD, em contraste ao caráter estático do SSD.

O padrão RBAC não menciona o controle de acesso a múltiplos sistemas. A arquitetura do projeto RASEA utiliza o padrão RBAC ANSI INCITS 359-2004 (ANSI, 2004), acrescentando-se a entidade *application* com o propósito de controlar o acesso às diversas aplicações comerciais simultaneamente.

2.3 SOA E *WEB SERVICES*

A sigla SOA significa Arquitetura Orientada a Serviços, do inglês *Service-Oriented Architecture*. Existem muitas definições para SOA, uma delas afirma que não se trata de uma tecnologia, mas de uma arquitetura, uma estratégia (DAVIES; KRISHNA; SCHOROW, 2007, p. xxi). Os autores Papazoglou e Heuvel (2007, p. 390) abordam SOA como uma filosofia de projeto, que possibilita a independência de qualquer tecnologia específica. Existem também pesquisas que tratam SOA como um paradigma que proporciona aumento da flexibilidade no desenvolvimento de aplicações (JOSUTTIS, 2007, p. 11).

Uma premissa básica da SOA é a disponibilização de funcionalidades do *software* como serviços reutilizáveis (PAPAZOGLOU; HEUVEL, 2007, p. 389). Sob este aspecto, surgem três elementos da arquitetura: (i) o provedor (*service provider*), que é o responsável pela administração e implementação do serviço; (ii) o consumidor (*service consumer* ou *service*

client), que faz uso dos serviços para um determinado fim e; (iii) o registro do serviço (*service registry*), que possibilita a publicação e disponibiliza a descrição dos serviços (YU, 2008, 541).

Pode-se considerar que um serviço é a exposição de determinadas funcionalidades de um sistema (JONES, 2006, p. 9) com o intuito de serem reutilizadas pelos consumidores. No contexto organizacional, serviços representam funcionalidades de negócio (do inglês *business functionality*) expressas em linguagem livre de tecnologia, podendo ser implementados em qualquer plataforma (JOSUTTIS, 2007, p. 2). Nota-se que a adoção de SOA no contexto organizacional está intimamente relacionada aos processos de negócio e alinhada às estratégias empresariais (JONES, 2006, p. 3).

É importante ressaltar que SOA não implica em *Web Services* (WS). Os *Web Services* são uma das possibilidades de materialização dos conceitos da SOA. Tomando como base as diversas definições para WS, confrontadas por Alonso (2004, p. 124–125), conclui-se que *Web Services* são: aplicações acessíveis através da *Web*, que possuem interfaces baseadas em padrões abertos, as quais são definidas, descritas e descobertas utilizando artefatos XML, suportando interações diretas com outras aplicações através de protocolos abertos. O projeto RASEA, produto deste trabalho, utiliza a tecnologia Java para implementar *Web Services*, objetivando a interoperabilidade com diversas tecnologias, linguagens de programação, plataformas e sistemas operacionais (DIRGAHAYU; QUARTEL; SINDEREN, 2008).

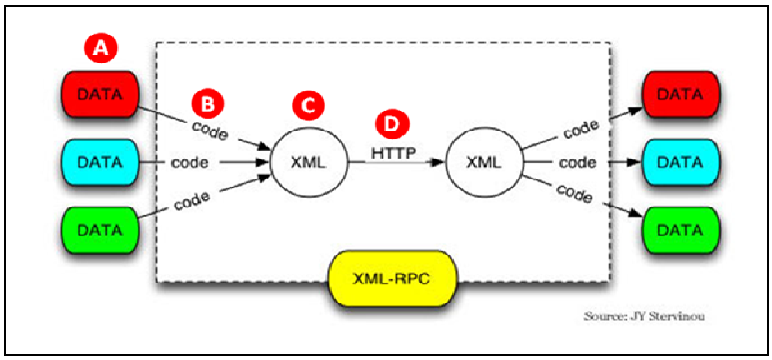

Figura 2.7 – Uso do XML-RPC
Fonte: XMLRPC (2009).
Nota: Adaptação do autor.

Os *Web Services* adotam o formato *Extensible Markup Language* (XML) para a comunicação entre aplicações, transmitidas através do protocolo *Hypertext Transfer Protocol* (HTTP), dando origem ao *Simple Object Access Protocol* (SOAP) (W3C, 2007). O SOAP foi criado em 1998, suprindo deficiências do seu antecessor XML-RPC (BOX, 2001), tais como: (i)

definição de tipos complexos, como: estruturas, listas e objetos; (ii) utilização do *Unicode Transformation Format* (UTF) para representação das mensagens, ao invés do conjunto limitado de caracteres suportado pelo protocolo XML-RPC.

O XML-RPC foi criado em resposta aos complexos protocolos proprietários utilizados em chamadas de procedimentos remotos (RPC, do inglês *Remote Procedure Call*). A Figura 2.7, publicada no site oficial do XML-RPC (http://www.xmlrpc.com), ilustra a simplicidade na comunicação utilizando o XML-RPC, onde o elemento 'A' representa os dados em uma tecnologia específica a serem transmitidos para o receptor. Os dados são submetidos ao processo de transformação (B), convertendo-os no formato XML (C) e transmitindo-os via HTTP (D). O receptor executa o procedimento contrário, convertendo a mensagem XML para o formato específico de sua tecnologia.

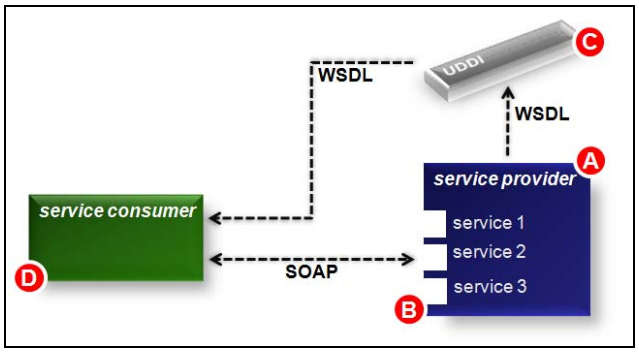

Figura 2.8 – Elementos básicos do *Web Service*

Conforme a Figura 2.8, um *Web Service* descreve os seus serviços (B) através da *Web Services Description Language* (WSDL) no formato XML, que é publicada e administrada pelo *service provider* (A). Sinteticamente, uma descrição WSDL envolve a declaração de uma interface de serviços em formato aberto. Tal declaração tem o mesmo papel das *Interface Description Languages* (IDL) usuais no contexto de sistemas distribuídos. A partir das informações contidas no WSDL é possível gerar uma aplicação cliente (D) em qualquer tecnologia que suporte o consumo de *Web Services*. Para possibilitar a descoberta de serviços, foi criado o padrão *Universal Description, Discovery, and Integration* (UDDI) (C), responsável por listar e descrever os serviços disponíveis (ALONSO, 2004, p. 125).

O SOAP é utilizado na troca de mensagens (requisição e resposta) entre provedores e consumidores de *Web Services*. De acordo com a especificação do protocolo SOAP (W3C, 2007), versão 1.2, a estrutura da mensagem contempla dois elementos: i) o cabeçalho e; ii) o corpo. A especificação estabelece o elemento 'envelope', composto pelo 'cabeçalho' e o

'corpo' da mensagem. Este trabalho utiliza o SOAP como protocolo para a troca de mensagens entre o provedor e os consumidores, tendo como principal vantagem a independência de plataforma.

Uma das grandes motivações para a escolha de *Web Services* é a utilização de padrões abertos (HULL; SU, 2005, p. 87), mantidos por órgãos internacionalmente conhecidos, tais como: o *World Wide Web Consortium* (W3C) (http://www.w3.org) e a *Organization for the Advancement of Structured Information Standards* (OASIS) (http://www.oasis-open.org). Os principais padrões, já apresentados, são: XML, SOAP, WSDL e UDDI.

Com o intuito de padronizar a implementação de *Web Services* na linguagem Java, criou-se a especificação JSR-101, intitulada *Java APIs for XML based RPC* (JAX-RPC). Com o surgimento de melhorias no SOAP e no WSDL, bem como evoluções na plataforma Java, o JAX-RPC tornou-se defasado. O JAX-WS, acrônimo de *Java API for XML-Based Web Services*, foi criado para suprir as deficiências do seu antecessor. Considerada uma evolução do JAX-RPC, a especificação do JAX-WS é também conhecida como JSR-224. (BUTEK; GALLARDO, 2006)

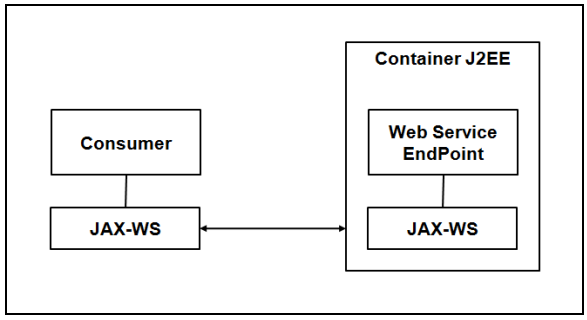

Figura 2.9 – Uso do JAX-WS
Fonte: Sampaio (2006, p. 8).
Nota: Adaptação do autor.

Na Figura 2.9, Sampaio (2006, p. 8) apresenta os elementos básicos envolvidos na implementação de *Web Services* em Java. O elemento *Consumer* representa um consumidor de um serviço disponível no *Container J2EE*. Basicamente, a função do contêiner é hospedar aplicações *Java Enterprise Edition* (J2EE ou JEE) como, por exemplo, um *Web Service*. A figura original ilustra o uso do JAX-RPC, porém a imagem foi adaptada para o JAX-WS. O elemento *Web Service EndPoint* representa um conjunto de funcionalidades providas pelo *Web Service* (SAMPAIO, 2006, p.39). Em outras palavras, um *EndPoint* é um agrupamento de serviços.

O projeto RASEA faz uso do JAX-WS em detrimento do JAX-RPC por alguns motivos: (i) aderência ao padrão SOAP 1.2; (ii) suporte ao HTTP *binding* do WSDL 1.1 e; (iii) suporte às novas funcionalidades da versão 5 do Java (JSE, 2004).

2.4 FERRAMENTAS EXISTENTES

Existem diversas soluções que tratam sobre segurança de aplicações. Muitas delas são bibliotecas ou *frameworks* que subsidiam tarefas de autenticação e autorização, enquanto outras oferecem serviços para controle de acesso. Este trabalho não tem como objetivo substituir tecnologias existentes e sim integrá-las, possibilitando o controle de acesso *cross*-plataforma às aplicações comerciais de forma unificada. Para tanto, foi feito o levantamento das ferramentas existentes, onde os seguintes aspectos foram analisados:

a) **Autenticação** – A ferramenta possui suporte à autenticação? Suporta integração com a base de autenticação da organização? Possibilita a integração com diferentes estratégias de autenticação, e.g., *Lightweight Directory Access Protocol* (LDAP), OpenID, bancos de dados?

b) **Autorização** – A solução auxilia ou executa o efetivo controle de acesso aos recursos do sistema, possibilitando o gerenciamento de permissões e autorizações das aplicações?

c) **RBAC** – Provê o gerenciamento do controle de acesso baseado em papéis? Está aderente ao padrão ANSI INCITS 359-2004: RBAC?

d) **Padrões abertos** – Faz uso de padrões abertos para comunicação, armazenamento e parametrização? Utiliza padrões abertos para integração com sistemas já existentes da organização?

e) **Código aberto / livre** – A licença da ferramenta está de acordo com as definições da *Free Software Foundation* (http://www.fsf.org) ou instituições similares? A ferramenta é gratuita? O código-fonte está disponível para a comunidade em repositórios abertos? A documentação é gratuita?

f) **Suporte** – A ferramenta possui documentação de referência, manual de instalação e utilização? Libera atualizações constantes para evolução e correções da ferramenta?

g) ***Web*** – A solução contempla aplicações na plataforma *Web*? O acesso ao painel de gerenciamento está disponível via *Web*? A interface de integração com outras aplicações é acessível pela *Web*?

h) **Módulo de gerenciamento** – Possui um módulo de gerenciamento com interface gráfica? Facilita o controle e o gerenciamento das permissões e autorizações às aplicações comerciais? Possui uma interface padronizada para integração com outras aplicações?

i) **Unificação** – Garante o controle unificado de acesso às aplicações comerciais? Existe um único ponto para gerenciamento do controle de acesso? A base de credenciais dos usuários é única para todas as aplicações?

j) **Independência de plataforma** – A solução possibilita a integração com aplicações comerciais de forma independente de plataforma? A ferramenta pode ser instalada em Sistemas Operacionais diversos?

	A Autenticação	B Autorização	C RBAC	D Padrões abertos	E Código aberto / livre	F Suporte	G Web	H Módulo de gerenciamento	I Unificação	J Independência de plataforma
1 Seam Security	✓	✓	!	✓	✓	✓	!		!	✗
2 MACA	✓	✓	!	✓	✓	!	!	✓	✓	✓
3 OES	✓	✓	!	✓	✗	✓	✓	✓	✓	✓

Legenda

✓ Atende completamente

! Atende parcialmente

✗ Não atende

Não se aplica / impossível avaliar

Figura 2.10 – Visão geral da avaliação das ferramentas

A Figura 2.10 apresenta o resultado da avaliação de cada uma das ferramentas analisadas. O critério de escolha das ferramentas foi o seguinte: ferramenta largamente utilizada no mercado ou publicada em repositório *open source*. Segue o descritivo de cada uma das soluções:

1) ***Seam Security*** – Ferramenta distribuída como parte integrante do *JBoss Seam Framework* (http://www.seamframework.org), mantido pela multinacional *Red Hat* (http://www.redhat.com), é responsável pelas tarefas de autenticação e autorização do *framework*. O *JBoss Seam Framework* foi criado para integrar as diversas tecnologias necessárias para o desenvolvimento de aplicações comerciais na plataforma Java, facilitando e agilizando a construção de aplicações complexas. O termo '*seam*', que pode ser traduzido como 'costura', faz uma alusão ao esforço necessário para integrar *frameworks* existentes, tarefa indispensável para o desenvolvimento de aplicações comerciais em Java. Por isso o *JBoss Seam* é conhecido como um *framework* de integração.

	A Autenticação	B Autorização	C RBAC	D Padrões abertos	E Código aberto / livre	F Suporte	G Web	H Módulo de gerenciamento	I Unificação	J Independência de plataforma
Seam Security	✓	✓	!	✓	✓	✓	!		!	✗

Legenda
- ✓ Atende completamente
- ! Atende parcialmente
- ✗ Não atende
- Não se aplica / impossível avaliar

Figura 2.11 – Avaliação da ferramenta *Seam Security*

O *JBoss Seam* possui um módulo de segurança, o *Seam Security*, que provê tarefas de autenticação e autorização em conformidade com padrões da plataforma Java (JAAS, 2001). A especificação do Seam Security (SEAM, 2009) está dividida em duas questões: (i) autenticação e; (ii) autorização. A autenticação baseia-se em repositórios de usuários (do inglês *Identity Store*), sendo possível, por meio de parametrizações, optar por uma das estratégias de armazenamento disponíveis, tais como: LDAP, OpenID e bancos de dados. As tarefas de autenticação utilizam processadores de

permissões (do inglês *Permission Resolver*), sendo possível escolher uma das diversas estratégias pré-implementadas por meio de parametrizações.

O ponto forte do *Seam Security* é a flexibilidade, pois, apesar de disponibilizar recursos pré-implementados, é possível criar componentes personalizados. Porém, de acordo com os critérios estabelecidos para a análise das ferramentas, o *Seam Security* apresenta como ponto fraco o não atendimento aos itens: RBAC (C), *Web* (G), Módulo de gerenciamento (H), Unificação (I) e Independência de plataforma (J), apresentados na Figura 2.11.

Apesar de prover um gerenciamento do controle de acesso baseado em papéis, o *Seam Security* não atende ao item C, pois não está em conformidade com o padrão ANSI INCITS 359-2004. O *Seam Security* pode ser utilizado por aplicações na plataforma *Web*, entretanto, por se tratar de uma biblioteca de funções, não possui um painel de gerenciamento disponível via *Web*, atendendo parcialmente ao item G. Pelo mesmo motivo, o item H (módulo de gerenciamento) não pôde ser avaliado.

Como o *Seam Security* deve ser implantado em cada aplicação, é impossível manter o gerenciamento unificado do controle de acesso para todas as aplicações, conforme ilustrado na Figura 2.12. Entretanto, é possível manter uma base única de credenciais dos usuários compartilhada por todas as aplicações, atendendo parcialmente ao item I (Figura 2.12). A utilização do *Seam Security* é restrita à plataforma Java, não atendendo ao critério J (Figura 2.11).

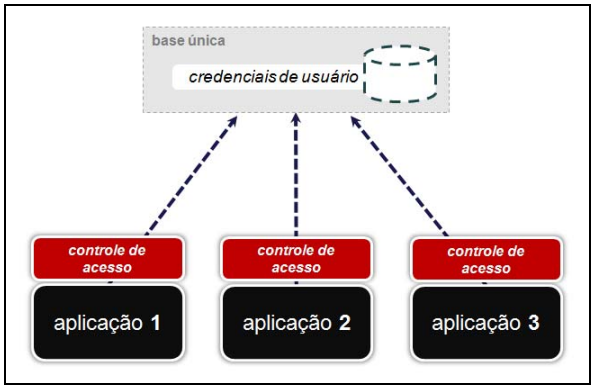

Figura 2.12 – Redundância no controle de acesso do *Seam Security*

2) **MACA** – Resultado da tese de doutorado de Motta (2004) na Universidade de São Paulo (USP), o MACA está armazenado no repositório de projetos de código aberto *SourceForge* (http://www.sf.net). O objetivo do projeto MACA é "prover os serviços

de autenticação de usuário e de autorização de acesso para aplicações legadas ou em desenvolvimento, independente de plataforma e de linguagem de programação" (http://maca.sourceforge.net). Acrônimo de *Middleware* de Autenticação e Controle de Acesso, o MACA foi concebido para atender demandas específicas de controle de acesso ao Sistema de Prontuário Eletrônico de Pacientes Clínicos.

Na arquitetura MACA (Figura 2.13), o elemento 'A' representa o provedor unificado de serviços de autorização e controle de acesso, os quais são acessados pelas aplicações 'B', fazendo uso do padrão aberto *Common Object Request Broker Architecture* (CORBA). O item 'C' ilustra a base única de usuários, responsável por armazenar as credenciais dos usuários da organização e prover serviços de autenticação. O elemento 'D' representa os sistemas de infra-estrutura, que fazem uso dos serviços de autenticação da base única. A arquitetura foi projetada para um ambiente de intranet, conforme demonstrado pelo item 'E'.

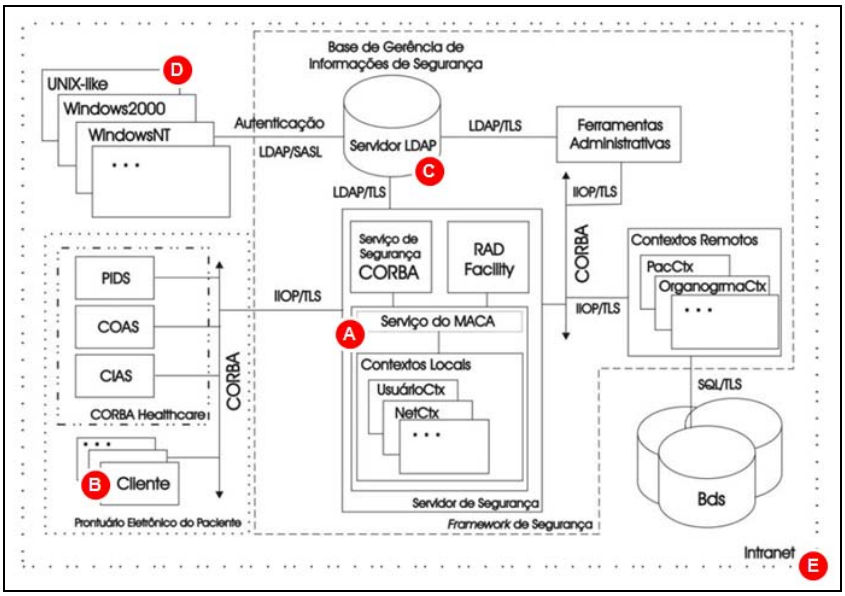

Figura 2.13 – Arquitetura do projeto MACA
Fonte: Motta (2004, p. 101).

De acordo com a Figura 2.14, o projeto MACA atende parcialmente ao critério C por não estar em conformidade com a nomenclatura estabelecida pela Especificação Funcional do RBAC no padrão ANSI INCITS 359-2004. O MACA disponibiliza documentações de referência no *site* do projeto (http://maca.sourceforge.net), porém a última atualização ocorreu em 2005, impactando negativamente no item F. O item G foi parcialmente atendido, pois um dos painéis de gerenciamento não suporta a plataforma *Web* (MOTTA, 2004, p. 132–133).

3) **OES** – O *Oracle Entitlements Server* (OES) é um produto da multinacional Oracle (http://www.oracle.com) de Tecnologia da Informação (TI). O OES é originado do produto *AquaLogic Enterprise Security* (ALES) da empresa BEA *Systems* (http://www.bea.com). Porém, com aquisição da BEA em 2008, o nome do produto mudou de ALES para OES. O principal objetivo do *Oracle Entitlements Server* é oferecer serviços de autorização para unificar e simplificar o gerenciamento das políticas de acesso aos sistemas das organizações (ORACLE, 2008, p. 3).

	A Autenticação	B Autorização	C RBAC	D Padrões abertos	E Código aberto / livre	F Suporte	G Web	H Módulo de gerenciamento	I Unificação	J Independência de plataforma
MACA	✓	✓	!	✓	✓	!	!	✓	✓	✓

Legenda
✓ Atende completamente
! Atende parcialmente

Figura 2.14 – Avaliação da ferramenta MACA

A Figura 2.15 ilustra a arquitetura do OES com seus principais elementos: (i) *Policy Administration Point* (PAP) e; (ii) *Policy Decision Point* (PDP). O PAP (B) é responsável pelo gerenciamento de papéis funcionais e políticas de acesso às aplicações através de telas administrativas. O administrador de segurança (A) utiliza o PAP (B) para gerenciar o OES. O PDP (C e F) toma como base as informações mantidas pelo PAP para controlar efetivamente o acesso dos usuários às aplicações.

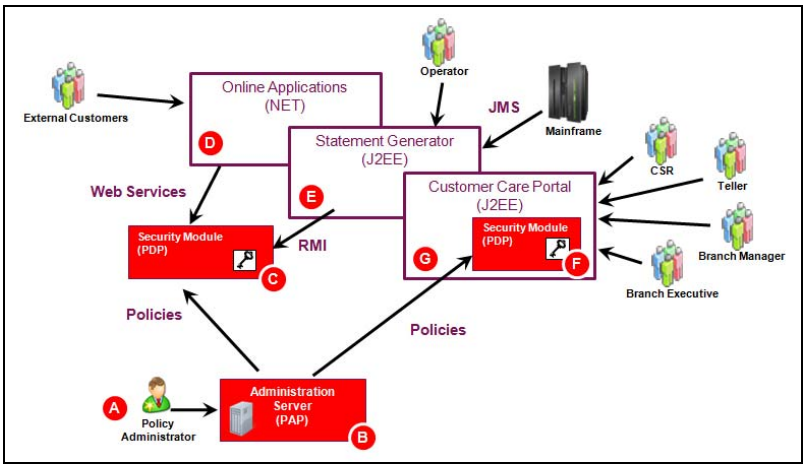

Figura 2.15 – Arquitetura do OES
Fonte: Oracle (2008, p. 11).

O OES oferece dois tipos de PDP (ORACLE, 2008, p. 8–9): (i) PDP embutido e; (ii) PDP centralizado. O PDP embutido, representado pelo item F (Figura 2.15), é implantado diretamente nos Servidores de Aplicação. Desta forma, todas as aplicações instaladas no servidor G (Figura 2.15) terão seu controle de acesso gerenciado pelo OES. O OES oferece também o PDP centralizado (C), acessado diretamente pelas aplicações (D e E) em diferentes tecnologias a partir de serviços expostos utilizando padrões abertos.

Com base nos critérios estabelecidos para a análise das ferramentas, apesar de prover um gerenciamento do controle de acesso baseado em papéis, o OES atende parcialmente ao critério C (Figura 2.16) por não estar em conformidade com a nomenclatura estabelecida pela Especificação Funcional do RBAC no padrão ANSI INCITS 359-2004. Por se tratar de um produto comercial e proprietário, o OES não atende ao item E (Figura 2.16). O preço do OES é de aproximadamente R$ 100.000,00 (cem mil reais), incluindo licença de uso para quantidade ilimitada de usuários com suporte e atualização de versões por um ano. O custo da ferramenta pode inviabilizar sua implantação em organizações que dispõem de recurso orçamentário insuficiente para este fim.

	A Autenticação	B Autorização	C RBAC	D Padrões abertos	E Código aberto / livre	F Suporte	G Web	H Módulo de gerenciamento	I Unificação	J Independência de plataforma
OES	✔	✔	!	✔	✘	✔	✔	✔	✔	✔

Legenda
- ✔ Atende completamente
- ! Atende parcialmente
- ✘ Não atende

Figura 2.16 – Avaliação da ferramenta OES

Este capítulo apresentou os conceitos fundamentais para a solução proposta neste trabalho, tratando sobre autorização, controle de acesso, serviços e o padrão ANSI INCITS 359-2004: RBAC. A próxima seção abordará experiências adquiridas durante a concepção, implementação e validação dos projetos nos cenários analisados, seguida pela seção de apresentação dos fundamentos e da arquitetura do projeto RASEA.

3 EVOLUÇÃO DO PROJETO

As idéias apresentadas neste trabalho foram validadas com aplicações práticas antes da concepção do projeto RASEA. Em princípio, criou-se o projeto-piloto AvanSEG, que atingiu maturidade de produção e passou a ser chamado S4A. O RASEA surgiu como uma alternativa de código aberto gratuita, consolidando os conceitos em laboratório e ambiente real. Este capítulo apresenta as principais contribuições deste trabalho, descrevendo as experiências adquiridas. Por questões de confidencialidade, foram associados nomes fictícios às aplicações parceiras e às identidades das organizações.

3.1 CENÁRIO A – SISTEMAS INTERNOS

Em meados de 2007, a Avansys Tecnologia investiu na unificação da base de usuários da organização. Até então, cada sistema possuía uma base própria com credenciais de usuários, dificultando o gerenciamento da política de segurança da empresa. Diversos estudos foram feitos para identificar as informações relevantes para a organização, descartando os dados redundantes e desnecessários. Após a criação da base unificada surgiu outro problema: o controle descentralizado de acesso às aplicações internas da empresa.

O projeto de unificação da base de usuários envolveu diretamente 20 (vinte) sistemas críticos, acessados por aproximadamente 450 (quatrocentos e cinqüenta) usuários da organização. Para evitar a indisponibilidade dos serviços, o projeto foi cautelosamente planejado e executado no cronograma de 18 (dezoito) meses. Paralelamente à execução do projeto, a equipe de inovação da empresa – coordenada pelo autor deste trabalho – concebeu uma solução para unificação do controle de acesso, tendo como premissa básica: integrar com as aplicações internas de forma independente de tecnologia. Surgiu, em meados de 2007, o projeto-piloto AvanSEG.

O AvanSEG valeu-se da flexibilidade dos *Web Services* para alcançar a solução adequada. A validação do modelo foi feita utilizando-se três aplicações internas em diferentes tecnologias: *Active Server Pages* (ASP), PHP *Hypertext Preprocessor* (PHP) e Java. As aplicações parceiras integravam diretamente com os serviços providos pelo ‘servidor’, exigindo que cada

aplicação implementasse independentemente a lógica de autorização; este inconveniente foi solucionado com o emprego dos agentes (s. 4.5) em produtos sucessores. Apesar de possuir uma interface gráfica pouco intuitiva, o propósito do projeto-piloto AvanSEG foi alcançado, sendo implantado provisoriamente para fins de avaliação. Durante os testes, surgiram algumas oportunidades de melhoria da solução, tais como: (i) implementar algoritmos mais eficientes e; (ii) possibilitar o escalonamento do servidor.

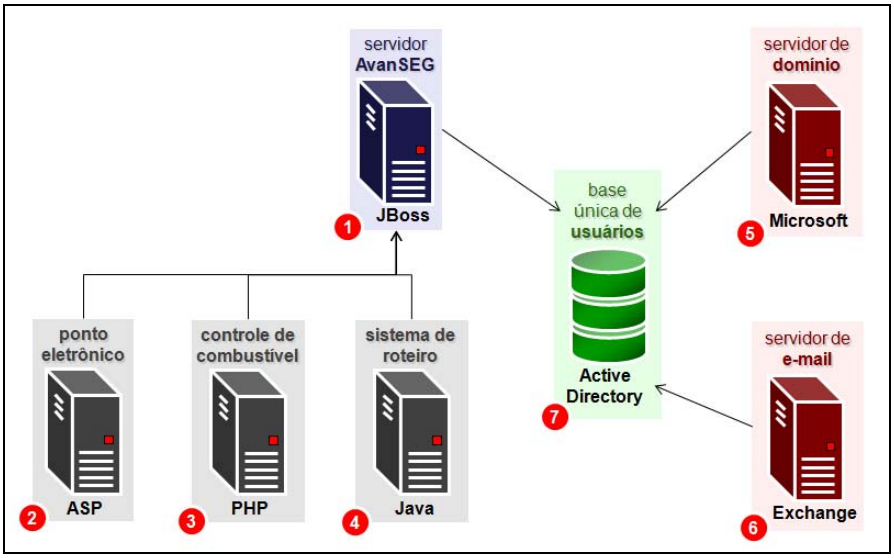

Figura 3.1 – Arquitetura dos Sistemas Internos

Com o intuito de dispor o cenário de avaliação do projeto-piloto, a Figura 3.1 apresenta os componentes da arquitetura e suas relações. O elemento ① representa o servidor, que provê os serviços de controle de acesso para as aplicações parceiras e integra com a base única de usuários ⑦, armazenada no servidor *Active Directory* (AD). O servidor AvanSEG foi construído na plataforma JEE, hospedado no servidor de aplicações JBoss AS. A aplicação parceira ②, projetada em ASP, hospedada no servidor *Internet Information Services* (IIS), acessa os serviços disponibilizados por ① para controle de acesso. O mesmo ocorre com as aplicações ③ e ④, construídas nas tecnologias PHP e Java respectivamente.

Ainda na Figura 3.1, nota-se a existência de dois serviços de infra-estrutura (⑤ e ⑥), não caracterizados como aplicações parceiras. O servidor ⑤ é responsável pelo acesso dos usuários à rede coorporativa, enquanto ⑥ é encarregado por gerenciar as mensagens eletrônicas da organização. Ambos estão integrados com a base única de usuários ⑦. Apesar de não fazer parte do escopo do projeto-piloto, os elementos ⑤ e ⑥ evidenciam a harmonia das ferramentas de infra-estrutura com a solução proposta, graças à utilização da tecnologia ⑦.

3.2 CENÁRIO B – PROJETO INSCRIÇÃO ONLINE

Após testes exaustivos e adequações no projeto-piloto AvanSEG, surgiu o *Security for Applications* (S4A) como uma solução para ambiente de produção. O projeto S4A foi integrado com diversas aplicações fabricadas pela Avansys para seus clientes, sob a coordenação do autor deste trabalho. Em princípio, pretendia-se tornar o projeto S4A *open source*, porém, devido às demandas internas, o plano foi suspenso. O S4A aplica os conceitos apresentados neste trabalho e serviu como alicerce para criação do projeto RASEA.

Em 2008, a Avansys desenvolveu a aplicação comercial Inscrição Online para um de seus clientes. O sistema Inscrição Online foi projetado para suportar 720 (setecentos e vinte) usuários com acesso simultâneo em horários de pico pré-determinados, geograficamente distribuídos por mais de 600 (seiscentas) unidades conectadas à Internet. Os acessos à aplicação foram controlados pelo S4A. Os requisitos não-funcionais do sistema focaram em dois aspectos: performance e disponibilidade.

O objetivo do sistema Inscrição Online é integrar o processo de matrícula dos alunos, conectando postos de inscrição distribuídos em diversos municípios com informações *online*. O sistema possui funcionalidades administrativas e gerenciais, que viabilizam o processo de matrícula. Durante o período de 60 (sessenta) dias, aproximadamente 1.000.000 (um milhão) de alunos foram cadastrados no sistema, gerando picos de acesso durante o horário comercial. Apesar de existir planos de contingência, a indisponibilidade ou lentidão do sistema poderia acarretar em re-planejamento do período de matrícula, gerando impactos financeiros diretos para a organização.

Apesar de utilizar apenas dois servidores, o sistema foi projetado para *cluster* com alta escalabilidade, possibilitando a substituição de servidores sem interrupção do serviço. Construído na linguagem Java, a aplicação Inscrição Online foi implantada em servidores JBoss AS, com balanceamento de carga no servidor Apache. A organização cliente possui uma base de usuários *Active Directory*, acessada pelo S4A para garantir a integração com a base única de usuários da organização.

A Figura 3.2 destaca a interação entre o sistema Inscrição Online e o S4A. O elemento ① representa os usuários da aplicação, que acessam o sistema através da intranet ou Internet. O

item ② representa o servidor JBoss AS, que hospeda o projeto Inscrição Online. O controle de acesso à aplicação parceira ② é executado pelo agente S4A implantado na aplicação Inscrição Online, que acessa o servidor (④ ou ⑤) através do balanceador de carga Apache ③. A base unificada de usuários da organização é ilustrada pelo elemento ⑥. Durante o processo de matrícula, foram utilizados dois servidores com balanceamento de carga. Nenhuma configuração especial foi feita no S4A para suportar *cluster*, e sim no JBoss AS, conforme a documentação oficial da ferramenta (STANSBERRY; ZAMARRENO, 2009). Apesar de utilizar apenas dois servidores, a solução suporta mais nós no *cluster*.

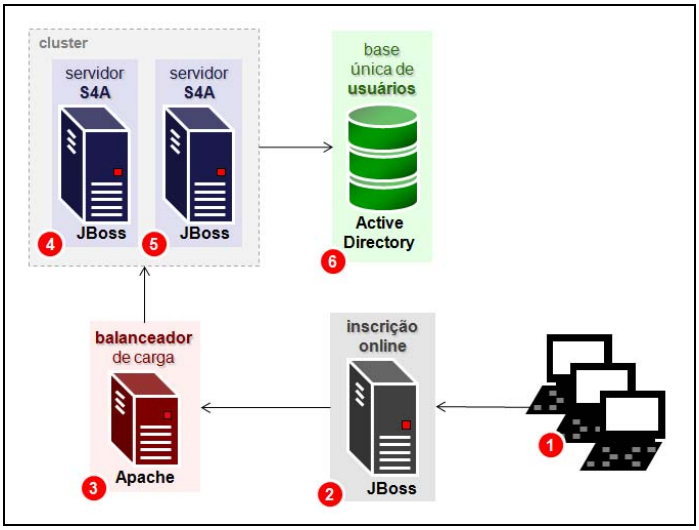

Figura 3.2 – Arquitetura do projeto Inscrição Online

3.3 CENÁRIO C – FÁBRICA DE *SOFTWARE*

O portfólio da Avansys Tecnologia é composto por uma variedade de serviços, dentre eles o desenvolvimento de sistemas sob regime de Fábrica de *Software*. O estabelecimento dos custos e prazos dos projetos é feito com base na estimativa do tamanho funcional do sistema, utilizando a técnica de Análise de Pontos de Função (APF) (VAZQUEZ; SIMÕES; ALBERT, 2003) do inglês *Function Point Analysis* (FPA). O método APF, mantido pelo *International Function Point Users Group* (IFPUG) (http://www.ifpug.org), é largamente utilizado no mercado como métrica de *software*.

Para cada nova aplicação, fazia-se necessária a construção de um novo módulo para controle de acesso que, com base na estimativa da organização, representa 190 PF (cento e noventa

pontos de função). Supondo um sistema de 1.000 PF (mil pontos de função), o controle de acesso representaria 19% (dezenove por cento) do custo de desenvolvimento. Empregando o S4A como componente reutilizável (DAN; JOHNSON; CARRATO, 2008, p. 25), a organização economiza 190 PF (cento e noventa pontos de função) a cada aplicação desenvolvida, com investimento mínimo na integração do *software*. Por questões de confidencialidade, não serão apresentados o custo do PF nem a produtividade da organização.

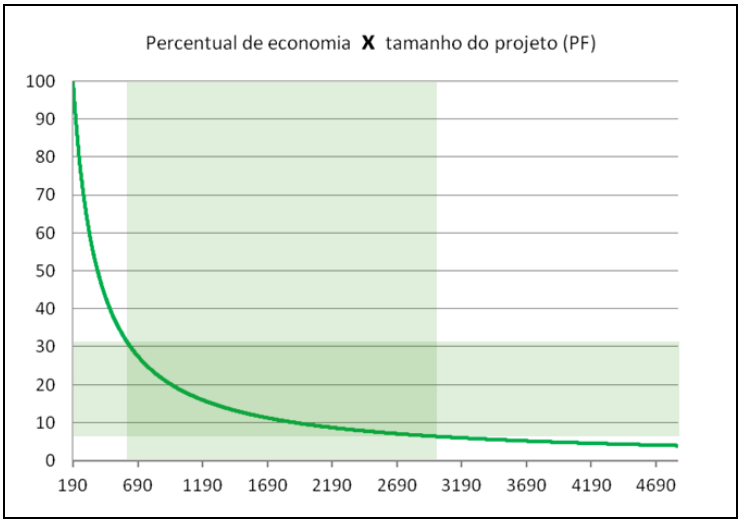

Figura 3.3 – Economia na utilização do S4A

A Figura 3.3 apresenta a economia obtida na Fábrica de *Software* com a utilização do S4A. O eixo horizontal representa o tamanho funcional da aplicação, variando entre 190 PF (cento e noventa pontos de função) e 5.000 PF (cinco mil pontos de função). O eixo vertical indica o percentual de economia, considerando que a aplicação possua funcionalidades para controle de acesso. Percebe-se que, quanto menor o tamanho do sistema maior será a economia obtida. Esta característica ocorre devido ao custo constante das funcionalidades de controle de acesso em relação às demais funcionalidades do projeto. De acordo com o histórico da Avansys, os sistemas construídos pela Fábrica de *Software* geralmente variam entre 600 PF (seiscentos pontos de função) e 3.000 PF (três mil pontos de função), gerando uma economia entre 6% (seis por cento) e 31% (trinta e um por cento) nos custos do projeto.

Com base em decisões estratégicas, a economia obtida pode servir para aumentar a competitividade da organização, maximizar a lucratividade ou melhorar os indicadores de satisfação do cliente e da equipe. Em um determinado projeto, o cliente solicitou um controle de acesso simples para a aplicação, com apenas 65 PF (sessenta e cinco pontos de função). Para atender este requisito, a Avansys precisaria abrir mão do S4A. Em comum acordo, a organização disponibilizou o seu componente sem custo adicional para o cliente. Como

resultado, houve o aumento da satisfação do cliente, que lucrou com 125 PF (cento e vinte e cinco pontos de função), e uma economia de 65 PF (sessenta e cinco pontos de função) no desenvolvimento do sistema.

Todos os benefícios com a utilização do S4A em regime de Fábrica de *Software* podem ser obtidos com o projeto RASEA. O S4A está sendo gradativamente substituído pelo RASEA nos projetos da Fábrica de *Software* da Avansys Tecnologia, e, de acordo com o planejamento, será completamente substituído até o final de 2009. Além disso, o projeto RASEA é utilizado pela organização para controlar o acesso às suas aplicações internas, que suportam as atividades de *backend* da empresa. O projeto RASEA foi exaustivamente testado e validado nos três cenários descritos neste capítulo.

3.4 CRONOLOGIA E EXPERIÊNCIAS ADQUIRIDAS

Nesta subseção serão apresentadas cronologicamente as experiências adquiridas durante a evolução deste trabalho, servindo como contribuição para o desenvolvimento de módulos para controle de acesso de aplicações comerciais. Durante o período de mais de 30 (trinta) meses de pesquisas e implementações, envolvendo organizações e empresas distintas, foi possível experimentar na prática a junção dos conceitos RBAC e *Web Services*. Participaram ativamente do processo, 12 (doze) usuários responsáveis pela administração de acesso nas organizações envolvidas.

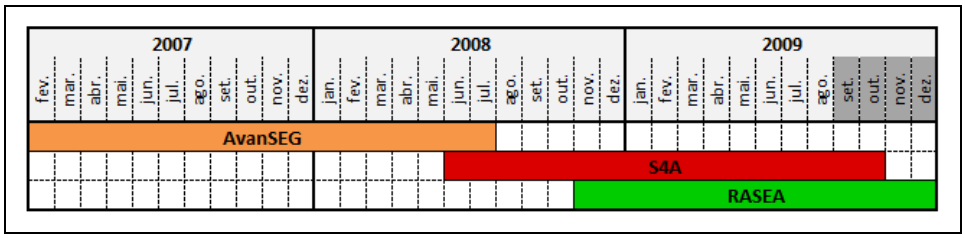

Figura 3.4 – Macro-cronograma dos projetos: AvanSEG, S4A e RASEA

O projeto-piloto AvanSEG surgiu em fevereiro de 2007 (Figura 3.4) para possibilitar o controle de acesso unificado dos sistemas internos da organização. Nesta época, por questões estratégicas, a Fábrica de *Software* da Avansys havia sido desativada, possibilitando que o setor de inovação concentrasse todo esforço em atividades estruturantes para a sede da

empresa. As atividades foram coordenadas pelo autor deste trabalho, o qual atuou também como projetista e codificador.

O AvanSEG foi integralmente desenvolvido pela equipe de inovação, sendo homologado com sistemas em diferentes tecnologias (ASP, PHP e Java), utilizando *Web Services* para garantir a integração de forma independente da plataforma. A execução do projeto foi bem sucedida, cumprindo o cronograma de 18 (dezoito) meses, finalizando em julho de 2008.

Em junho de 2008, a Fábrica de *Software* da Avansys foi reativada. Desde então, o autor deste trabalho passou a ser responsável não só pela equipe de inovação, mas também pela gerência da Fábrica de *Software*. Com o aumento da demanda por aplicações comerciais para a Fábrica de *Software*, surgiu a necessidade de garantir o controle de acesso dos sistemas fabricados, motivando a concepção do projeto S4A a partir do processo de refatoração do código-fonte do AvanSEG. As oportunidades de melhorias, identificadas no período de testes do AvanSEG, foram contempladas no S4A.

Por questões estratégicas, a Avansys Tecnologia decidiu que o novo sistema para controle de acesso, o S4A, deveria ser publicado em um repositório internacional de projetos *open source*. Para atender este requisito, a refatoração do S4A gerou impactos radicais no código-fonte, envolvendo alterações de idioma (do português para o inglês) no nome de: variáveis, arquivos, classes, pacotes, tabelas e banco de dados. Devido à grande quantidade de atividades prioritárias demandadas pelos clientes para a Fábrica de *Software*, a publicação do projeto foi suspensa.

Após a implantação do S4A em ambiente de produção, durante os 5 (cinco) primeiros meses, surgiram diversas sugestões de melhorias apontadas pelos usuários, as principais foram: (i) simplificação da tela de concessão de permissões; (ii) não compartilhamento dos papéis e das operações entre as aplicações e; (iii) possibilitar a parametrização do servidor através de arquivo de configuração. Além de sugerir melhorias, os usuários fizeram questão de ressaltar o ponto forte do S4A: a simplicidade no controle e no gerenciamento das permissões de acesso.

A Figura 3.5 ilustra o problema de compartilhamento dos papéis entre as aplicações. No cenário ①, o Sistema de Matrícula compartilha os papéis com o Sistema de Almoxarifado, que são: professor, gerente, aluno, faxineiro, diretor e despachante. Desta maneira o papel ‘professor’ será disponibilizado no gerenciamento das permissões do Sistema de

Almoxarifado, quando deveria estar disponível apenas no contexto do Sistema de Matrícula. No cenário ②, sugerido pelos usuários, os papéis estão associados às aplicações, evitando que os papéis de uma aplicação sejam disponibilizados para outra aplicação. A implementação desta funcionalidade no S4A foi avaliada tecnicamente como inviável, devido ao grande impacto na estrutura do projeto.

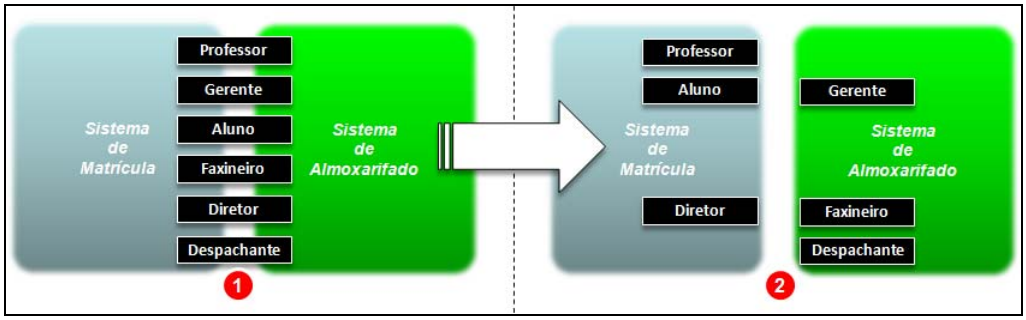

Figura 3.5 – Problema de compartilhamento de papéis entre aplicações

Em setembro de 2008, o autor deste trabalho iniciou uma pesquisa aprofundada sobre o padrão RBAC. Como o S4A encontrava-se implantado em ambiente de produção, decidiu-se criar um projeto livre de interferências externas para suprir as principais carências detectadas nos projetos anteriores (Figura 3.6), que são: (i) disponibilização do projeto em repositório *open source*; (ii) aderência ao padrão internacional RBAC e; (iii) parametrização do servidor através de arquivo de configuração. Todos os serviços foram re-projetados para atender à Especificação Funcional e ao Modelo de Referência do RBAC (APÊNDICE C – Módulo de serviços do Servidor RASEA). Este novo projeto, denominado RASEA, possibilita o isolamento de papéis entre aplicações, atendendo às solicitações dos usuários.

Para possibilitar o suporte multi-idioma do módulo administrativo, o servidor RASEA foi projetado para adaptar-se a outros idiomas sem a necessidade de refatoração do código-fonte. Os conceitos utilizados foram o *Internationalization* e o *Localization* (SMITH-FERRIER, 2006), largamente conhecidos pelas abreviaturas I18N e L10N, respectivamente. O módulo administrativo do RASEA conta atualmente com o arquivo de tradução para o português (pt_BR), mas pode ser traduzido para qualquer idioma suportado pela especificação JSR-000314 (2009).

Uma forte característica do projeto RASEA é a simplicidade, facilitando o gerenciamento e o controle de acesso às aplicações comerciais de forma intuitiva. O projeto foi fundado e publicado no *SourceForge* (http://www.sf.net) em novembro de 2008 pelo autor deste trabalho. O RASEA é administrado pelo autor deste trabalho, que atua como principal

executante das seguintes atividades: (i) arquitetura, (ii) projeto técnico; (iii) codificação. Desde o primeiro semestre de 2009, a Avansys Tecnologia investe em atividades para evolução da ferramenta, gerenciadas pelo autor deste trabalho, que atua como responsável e líder da equipe. A equipe do projeto RASEA é composta atualmente por 8 (oito) colaboradores, que executam as tarefas delegadas pelo autor deste trabalho, inclusive atividades referentes ao projeto de interface.

	Acesso à base única de usuários	Controle de acesso unificado	Independência de plataforma	Escalonamento do servidor	Código-fonte escrito na língua ing	Suporte à internacionalização	Código aberto / livre	Aderência ao padrão RBAC	Arquivo de configuração
AvanSEG	✓	✓	✓	✗	✗	✗	✗	✗	✗
S4A	✓	✓	✓	✓	✓	✗	✗	✗	✗
rasea	✓	✓	✓	✓	✓	✓	✓	✓	✗

Legenda
✓ Atende
✗ Não atende

Figura 3.6 – Comparativo entre os projetos: AvanSEG, S4A e RASEA

Em janeiro de 2009, o RASEA foi implantado na Avansys Tecnologia simultaneamente em 4 (quatro) projetos da Fábrica de *Software* e 2 (dois) projetos de uso interno da empresa. Em maio deste mesmo ano, o RASEA foi instituído como sistema padrão para controle de acesso às aplicações da Fábrica de *Software* e aos sistemas internos da organização. Com base no planejamento da empresa, o S4A será completamente substituído em toda linha de produção da Fábrica de *Software* até novembro de 2009, envolvendo aproximadamente 5 (cinco) projetos em fase de desenvolvimento e manutenção.

Em maio de 2009, ocorreu a migração do S4A para o RASEA em 3 (três) projetos da Fábrica de *Software* da Avansys. Por manipular dados confidenciais e críticos para o negócio da organização, este processo foi minuciosamente planejado, executado e validado em ambiente de homologação. Como a arquitetura interna do RASEA, que está em conformidade com o Modelo Funcional do RBAC, contrasta com a estrutura interna do S4A, que não é aderente ao padrão RBAC, foi necessário mapear criteriosamente as informações, evitando a perda ou

inconsistência dos dados no processo de migração (APÊNDICE F – *Script* de migração do S4A para o RASEA).

Um dos grandes desafios na construção do projeto RASEA foi possibilitar a escolha da base de usuários, e.g., LDAP, OpenID, através de arquivo de configuração. Nos projetos anteriores, a configuração ocorria de forma intrusiva, sendo necessária a manipulação de arquivos internos do sistema. Para simplificar a parametrização do módulo servidor, a configuração do RASEA ocorre através do arquivo de parametrização armazenado na pasta de configurações do Servidor de Aplicação (APÊNDICE A – Arquivo de configuração do servidor RASEA).

Recurso: **Gerenciamento de sistemas**

Grupos	Excluir	Inserir	Visualizar	Alterar
Equipe de Desenvolvimento	✗	✗	✗	✗
Equipe de Sustentação	✗	✗	✗	✗
Gerente de Projetos	✗	✗	✗	✗
Gerente de Sistemas	✓	✓	✓	✓
Administradores	✓	✓	✓	✓
Usuário Master	✗	✗	✗	✗

Figura 3.7 – Exemplo de apresentação das autorizações de forma matricial

A interface gráfica do módulo administrativo do RASEA foi inicialmente projetada conforme o APÊNDICE D – Antigo módulo administrativo do servidor RASEA. Entretanto, com base nos relatos e sugestões dos usuários, percebeu-se a necessidade de disponibilizar ferramentas para tornar eficiente o gerenciamento das permissões, que são: (i) disponibilização de consultas com filtros e; (ii) exibição das autorizações em forma matricial (Figura 3.7). O processo de refatoração contou com o envolvimento dos usuários nas atividades de especificação e validação do novo projeto de interface gráfica (APÊNDICE E – Atual módulo administrativo do servidor RASEA).

Durante o processo de desenvolvimento, os usuários envolvidos elogiaram a evolução do trabalho e evidenciaram a simplicidade proporcionada nas atividades de controle de acesso e gerenciamento de permissões. Todas as experiências adquiridas durante os 30 (trinta) meses de estudo, implementação e validação, envolvendo diretamente 12 (doze) usuários responsáveis pela administração de acesso em diversas organizações, contribuíram para a maturidade da arquitetura do projeto RASEA, explanada no capítulo a seguir.

4 PROJETO RASEA

Projetado como uma solução para controle de acesso unificado às aplicações comerciais, o RASEA implementa os conceitos RBAC, SOA e *Web Services*, elucidados no capítulo 2, para garantir a maturidade da solução e a independência de plataforma. O RASEA foi construído com base nas experiências adquiridas nos projetos AvanSEG e S4A. Este capítulo apresenta a arquitetura da solução proposta bem como outros aspectos de alto nível do projeto técnico. Os detalhes de codificação não serão abordados, haja vista que o código-fonte está disponível publicamente através do repositório *open source* (http://rasea.svn.sourceforge.net/viewvc/rasea/).

4.1 VISÃO GERAL

A Arquitetura de *Software* auxilia na identificação dos componentes e seus relacionamentos de alto nível (SHAW; GARLAN, 1994), facilitando o entendimento da solução como um todo. O projeto RASEA fundamenta sua arquitetura sobre dois elementos principais tratados sob duas perspectivas para assegurar o controle de acesso *cross*-plataforma às aplicações comerciais, que são, conforme a Figura 4.1, respectivamente: (i) servidor; (ii) agente; (iii) independência de plataforma e; (iv) integração.

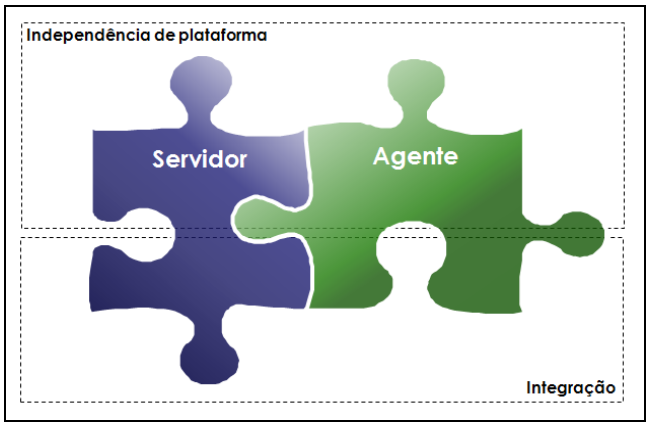

Figura 4.1 – Principais elementos e perspectivas da arquitetura do projeto RASEA

Os elementos da arquitetura atuam como um quebra-cabeça, completando o modelo com o encaixe de todas as peças. A perspectiva ‘independência de plataforma’ agrega padrões e

tecnologias, possibilitando que aplicações comerciais em diferentes plataformas beneficiem-se da solução proposta. A perspectiva 'integração' promove a interação entre o RASEA, as informações da organização e as aplicações clientes. Enquanto o elemento 'servidor' preocupa-se com disponibilidade dos serviços, o 'agente' – consumidor dos serviços – aborda aspectos específicos das plataformas das aplicações. Os elementos 'servidor' e 'agente' devem garantir a independência de plataforma e a integração, que serão tratados em detalhes mais adiante.

O RASEA é uma evolução do código-fonte e as experiências adquiridas nos projetos AvanSEG e S4A (s. 3), acrescentando-os melhorias, tais como: (i) seleção da base de usuários através de arquivo de configuração; (ii) código-fonte aberto e; (iii) conformidade com o padrão ANSI INCITS 359-2004: RBAC. Nenhuma característica ou funcionalidade dos projetos AvanSEG e S4A foi removida do RASEA, e sim melhoradas. Com base nos critérios estabelecidos neste trabalho para avaliação das ferramentas (s. 2.4), o projeto RASEA atende de forma completa a todos os requisitos, conforme ilustrado na Figura 4.2.

	Autenticação	Autorização	RBAC	Padrões abertos	Código aberto / livre	Suporte	Web	Módulo de gerenciamento	Unificação	Independência de plataf or
rasea	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legenda
✓ Atende completamente

Figura 4.2 – Avaliação do projeto RASEA

Objetivando a padronização de vocabulário, segue a relação de todos os elementos envolvidos na arquitetura do projeto RASEA e suas respectivas descrições:

a) **Base de controle de acesso** – Repositório unificado que armazena dados estruturados conforme o Modelo de Referência do padrão RBAC (Figura 2.3, p. 22), excetuando-se a entidade usuário. São exemplos de dados armazenados na base de controle de acesso: papéis, objetos, operações e permissões;

b) **Credenciais de usuário** – Dados que identificam os usuários da organização, tais como: nome, senha, e-mail, etc;

c) **Base de usuários** – Repositório unificado para armazenamento dos usuários da organização. A base de usuários é composta por instâncias do elemento '*user*' do Modelo de Referência do padrão RBAC (Figura 2.3, p. 22);

d) **Servidor** – Um dos principais elementos da arquitetura, responsável por unificar o controle de acesso, manter a conexão com as bases de dados e prover funcionalidades através do módulo administrativo e do módulo de serviços;

e) **Módulo administrativo** – Interface gráfica com o usuário que possibilita o gerenciamento das permissões e autorizações às aplicações, conforme a Especificação Funcional do RBAC para operações de administrativas (Figura 2.4, p. 23);

f) **Módulo de serviços** – Disponibiliza interfaces para integração de aplicações clientes com os serviços de controle de acesso e gerenciamento;

g) **Serviços de controle de acesso** – Agrupamento de serviços (ou *EndPoint*) consumido pelos agentes. Disponibilizam informações necessárias para controle de acesso conforme a Especificação Funcional do RBAC para operações de consulta (Figura 2.4, p. 23);

h) **Serviços de gerenciamento** – Agrupamento de serviços (ou *EndPoint*) acessado por consumidores que desejam gerenciar o servidor RASEA. Disponibiliza as mesmas funcionalidades do módulo administrativo;

i) **Agente** – Um dos principais elementos da arquitetura, responsável por interceptar e analisar as requisições às aplicações, concedendo ou revogando o acesso com base nas informações obtidas através do serviço de controle de acesso. Executa as operações de apoio ao sistema, conforme com a Especificação Funcional do RBAC (Figura 2.4, p. 23);

j) **Aplicação parceira** – É toda aplicação conectada a um agente que utiliza o RASEA como dispositivo para controle de acesso unificado. As aplicações parceiras podem, adicionalmente, consumir os serviços de gerenciamento;

k) **Aplicação autônoma** – É toda aplicação que não está conectada a um agente e, conseqüentemente, não utiliza o RASEA como dispositivo para controle de acesso

unificado. As aplicações autônomas podem consumir os serviços de gerenciamento, caso necessitem integrar com o RASEA;

l) **Container** – Representa um Servidor de Aplicação ou Servidor *Web* que hospeda as aplicações comerciais.

De acordo com Kruchten (1995), a arquitetura define os elementos do modelo, suas conexões e relacionamentos, conforme ilustrado na Figura 4.3. O elemento 'A' representa o servidor, o qual disponibiliza um módulo administrativo (B) para que o administrador do acesso (C) gerencie as permissões aos sistemas. O servidor provê o módulo de serviços (D), agrupado em dois *EndPoints*: (i) controle de acesso e; (ii) gerenciamento. Os serviços para controle de acesso respaldam os agentes com informações suficientes para executar as operações de apoio ao sistema. Os agentes são conectados às aplicações parceiras (E e F). As aplicações que não utilizam os serviços para controle de acesso são chamadas de aplicações autônomas (G), que podem acessar os serviços de gerenciamento caso desejem integrar com o RASEA.

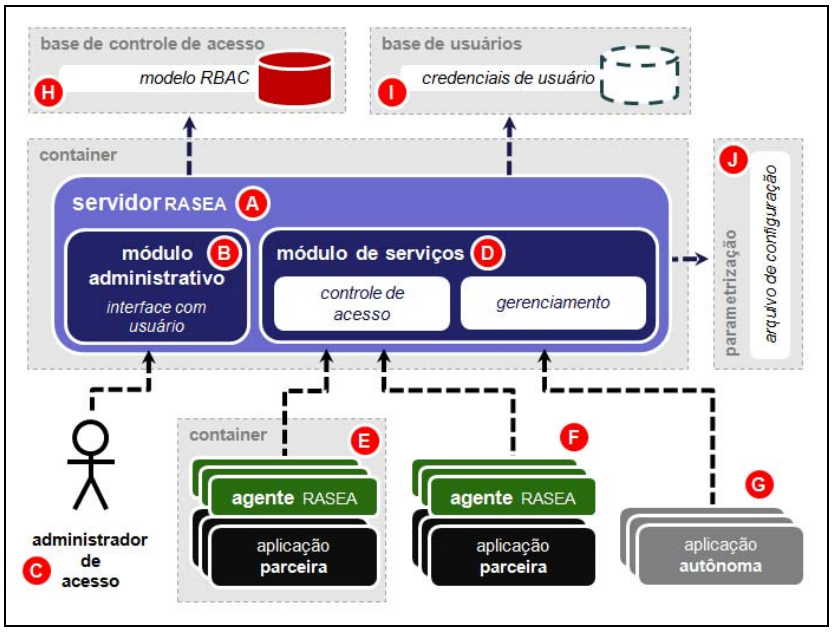

Figura 4.3 – Visão geral da arquitetura do projeto RASEA

Na Figura 4.3, o servidor (A) mantém a base de dados (H), que contém informações estruturadas conforme o Modelo de Referência do padrão RBAC, possibilitando o controle de acesso às aplicações comerciais. A base de dados (I) detém credenciais dos usuários existentes na organização. O servidor (A) acessa a base de dados de usuários (I) para executar tarefas de autenticação. Almejando a flexibilidade da solução, o servidor (A) pode ser parametrizado a partir de um arquivo de configuração (J).

Nos próximos tópicos, as perspectivas (independência de plataforma e integração) e os principais elementos (servidor e agente) da arquitetura serão abordados em detalhes. Serão considerados requisitos de performance, disponibilidade, concorrência, distribuição e tolerância a falhas, assuntos imprescindíveis para a definição da arquitetura do *software* (KRUCHTEN, 1995). Para tanto, faz-se necessário conhecer a estrutura dos dados mantidos pelo RASEA, tratada a seguir.

4.2 MODELO DE DADOS

Os dados mantidos pelo RASEA foram modelados seguindo duas abordagens: (i) diagrama de entidade-relacionamento e; (ii) diagrama de classes de domínio. O diagrama de entidade-relacionamento é traduzido em representação relacional dos dados persistentes, enquanto o diagrama de classes de domínio utiliza a Unified Modeling Language (UML). De acordo com Eden e Kazman (2003, p. 9), a UML é comumente empregada como notação de projeto técnico. Os diagramas estão de acordo com o padrão RBAC (FERRAIOLO, 2001, p. 232) acrescendo-se a entidade '*application*' para suportar o controle de acesso às aplicações comerciais.

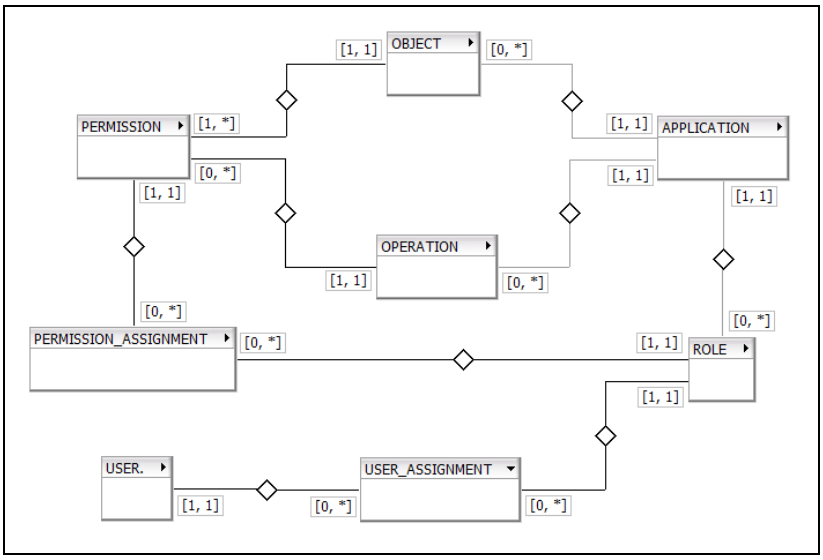

Figura 4.4 – Diagrama de Entidade-relacionamento do RASEA

A Figura 4.4 apresenta o Diagrama de Entidade-relacionamento, onde é possível observar os elementos do RBAC. A entidade *object* representa um recurso que está sob gerência do controle de acesso. As operações (*operation*) são associadas aos recursos, dando origem à

entidade *permission*. As permissões podem ser associadas aos grupos de usuário, ilustradas pela entidade *permission_assignment*. Os papéis funcionais são representados por *role*. Um papel agrupa usuários (*user*) a partir da associação *user_assignment*. Uma aplicação (*application*) é formada por recursos (*object*). Cada aplicação possui suas próprias operações (*operation*) e papéis (*role*). Assim, é possível que o papel 'R' da aplicação 'A1', por exemplo, seja composto por usuários distintos do papel 'R' da aplicação 'A2'. Ou seja, 'R (A1) ≠ R (A2)'.

O Diagrama de Classes de Domínio (Figura 4.5) representa a visão orientada a objetos (OO) do padrão RBAC utilizando UML (RAY, 2004). Por se tratar de uma percepção OO, algumas entidades relacionais sofreram adaptações na sua nomenclatura, aproximando-se mais da realidade das aplicações comerciais (GAEDKE; MEINECKE; NUSSBAUMER, 2005, p. 1156). A associação *permission_assignment* é representada pela classe *Authorization*. A entidade *object* ganhou um nome mais intuitivo: *Resource*, que pode ser traduzido como 'recurso'. A associação *user_assignment* é representada pela classe *Member*.

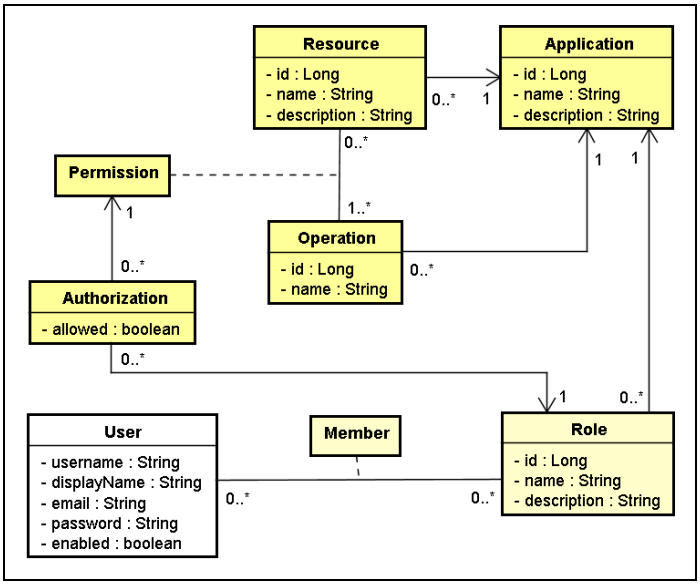

Figura 4.5 – Diagrama de Classes de Domínio do RASEA

4.3 INDEPENDÊNCIA DE PLATAFORMA E INTEGRAÇÃO

Uma das premissas do RASEA é garantir a interoperabilidade com aplicações parceiras em diferentes plataformas. Para possibilitar a independência de tecnologia, optou-se por duas

estratégias: (i) garantir a independência de plataforma; e (ii) empregar *Web Services* no canal de comunicação. De acordo com Li e Karp (2007, p. 9), os padrões abertos, adotados pela especificação Web Services, possibilitam o desacoplamento das aplicações, facilitando o desenvolvimento de componentes independentes de tecnologias.

A Figura 4.6 destaca a utilização de padrões abertos na conexão entre o servidor (C) e os agentes. O item 'A' representa aplicações parceiras acopladas aos agentes. Cada aplicação parceira possui sua respectiva base de dados (B), que armazena informações relativas ao negócio da aplicação. Os dados de controle de acesso (D) e as credenciais dos usuários (E) são mantidos pelo servidor RASEA, abstraindo todos os detalhes técnicos para as aplicações parceiras. As informações das bases 'D' e 'E' são transferidas para os agentes exclusivamente por meio de *Web Services* via SOAP, assegurando a independência de plataforma.

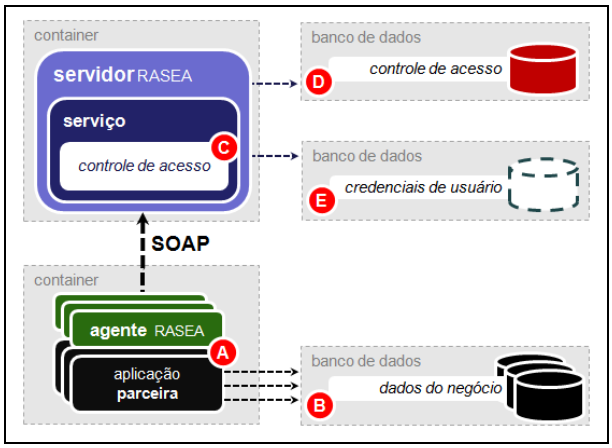

Figura 4.6 – Comunicação utilizando padrões abertos

Na solução RASEA, os serviços foram agrupados em dois *EndPoints* (Figura 4.3): (i) controle de acesso e; (ii) gerenciamento. O *EndPoint* de controle de acesso contém serviços consumidos pelos agentes e provê informações para o efetivo controle de acesso. O consumo do *EndPoint* de controle de acesso ocorre no processo de autenticação e verificação de autorização. O *EndPoint* de gerenciamento reúne os serviços de administração do servidor RASEA, acessados por aplicações (parceiras ou autônomas) que necessitam modificar informações do servidor.

Considerando que todos os serviços devem ser autônomos (PAPAZOGLOU; HEUVEL, 2007, p. 390) e estar em conformidade com a Especificação Funcional do *Core* RBAC (ANSI, 2004; FERRAIOLO, 2001, p. 242–243), segue a descrição dos serviços do *EndPoint* de controle de acesso:

a) **assignedRoles (application, user): role[]** – Retorna a listagem contendo todos os papéis que o usuário possui em uma determinada aplicação;

b) **assignedUsers (application, role): user[]** – Retorna a listagem contendo todos os usuários que possuem o papel em uma determinada aplicação;

c) **roleOperationsOnObject (application, role, object): operation[]** – Retorna todas as operações que um papel pode executar sobre um determinado recurso (objeto) de uma aplicação;

d) **rolePermissions (application, role): permission[]** – Retorna as permissões autorizadas ao papel em uma determinada aplicação. Uma permissão é composta pelo par 'operação × recurso' (BARKER, 2008, p. 144);

e) **userOperationsOnObject (application, user, object): operation[]** – Retorna todas as operações que um usuário pode executar sobre um determinado recurso (objeto) de uma aplicação. A verificação é feita com base nos papéis associados ao usuário;

f) **userPermissions (application, user): permission[]** – Provê a listagem das permissões autorizadas ao usuário em uma determinada aplicação. A verificação é feita com base nos papéis associados ao usuário;

g) **authenticate (username, password): boolean** – Executa o processo de autenticação do usuário tomando como base o seu nome e senha. Este serviço não é exigido pelo padrão RBAC;

h) **userDetail (username): user** – Obtém as credenciais do usuário com base no seu nome. Este serviço não é exigido pelo padrão RBAC.

Ao contrário do *EndPoint* de controle de acesso, o *EndPoint* de gerenciamento provê operações de modificação de dados no servidor. Em concordância ao padrão RBAC, segue a descrição dos serviços de gerenciamento:

a) **addUser (user)** – Adiciona um novo usuário na base unificada de usuários;

b) **deleteUser (user)** – Remove um usuário existente da base unificada de usuários;

c) **assignUser (application, user, role)** – Associa o usuário ao papel de uma aplicação;

d) **deassignUser (application, user, role)** – Desassocia o usuário ao papel de uma aplicação;

e) **changePassword (username, oldPassword, newPassword)** – Efetua a modificação da senha do usuário. Este serviço não é exigido pelo padrão RBAC;

f) **addRole (application, role)** – Adiciona um novo papel em uma aplicação;

g) **deleteRole (application, role)** – Remove um papel existente de uma determinada aplicação;

h) **grantPermission (application, permission, role, allowed)** – Concede permissão ao papel para executar uma operação sob um determinado recurso da aplicação. Uma autorização pode ser positiva ou negativa (BERTINO, 2001, p. 42);

i) **revokePermission (application, permission, role)** – Remove a permissão de um papel a um recurso do sistema.

Todos os serviços foram adaptados para suportar o controle de acesso aos múltiplos sistemas da organização, acrescentando-se o parâmetro '*application*'. Este assunto foi discutido nas seções "2.2 RBAC" (p. 20) e "4.2 Modelo de dados" (p. 50). As funções de apoio ao sistema (ANSI, 2004) não estão descritas nos serviços, pois são implementadas nos agentes RASEA, responsáveis pelo controle da sessão do usuário. O mesmo ocorre com as funções de consulta *SessionRoles* e *SessionPermissions* (Figura 2.4), que exibem os papéis e as permissões associadas à sessão do usuário, respectivamente.

O RASEA não pretende substituir tecnologias, arquiteturas, aplicações ou fontes de informação existentes na organização, e sim possibilitar a integração utilizando padrões abertos. Os problemas de integração (ANAYA; ORTIZ, 2005, p. 27) são analisados e tratados sob três pontos de vista: (i) aplicação; (ii) informação e; (iii) tecnologia.

A interação com aplicações pode acontecer de duas formas: (i) comunicação com aplicações parceiras ou; (ii) comunicação com aplicações autônomas. A Figura 4.7 destaca a integração com aplicações autônomas (B) por meio dos serviços de gerenciamento (A), possibilitando que sistemas interajam ativamente com o servidor RASEA manipulando informações e modificando seu comportamento (CUMMINS, 2002, p. 317).

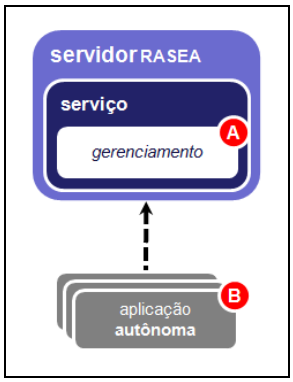

Figura 4.7 – Integração com o serviço de gerenciamento

Além da integração com seus serviços, o RASEA possibilita a conexão com repositórios de credenciais de usuários. Como exemplo, tem-se o LDAP, adotado por diversas organizações com o propósito de unificar o armazenamento e a autenticação dos usuários. A Figura 4.8 ilustra a integração entre o servidor (A) e a base de usuários (B). A estratégia de armazenamento das credenciais, representada pelos itens 'C', 'D' e 'E', é definida no arquivo de configuração do servidor, detalhada na subseção a seguir. Para garantir a extensibilidade da solução, o RASEA permite que novas estratégias (F) sejam acopladas, atendendo às necessidades específicas que possam ocorrer. Este processo será detalhado na seção 4.4 – Servidor RASEA.

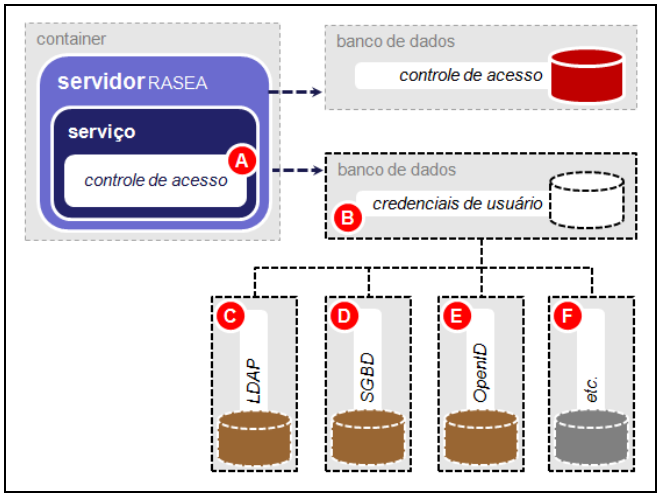

Figura 4.8 – Integração com base de usuários legada

É fundamental que todos os níveis de integração (CUMMINS, 2002, p. 316) estabeleçam conexões seguras pra garantir integridade, autenticidade e confidencialidades da informação. A integração entre as aplicações e os serviços RASEA empregam o *HyperText Transfer Protocol Secure* (HTTPS), utilizado por instituições financeiras para efetuar transações via Internet. A conexão com a base de credenciais utiliza dutos criptografados, tal como o *Secure*

LDAP (LDAPS). Não faz parte do escopo desde trabalho detalhar protocolos e técnicas de segurança.

Integrar soluções existentes ao invés de reinventá-las. Essa característica está evidenciada nos agentes RASEA, que se beneficiam de diversas tecnologias para desempenhar o seu trabalho, encaixando-as como peças de um quebra-cabeça. Todos os detalhes sobre os agentes serão explicitados na seção 4.5 – Agente RASEA.

4.4 SERVIDOR RASEA

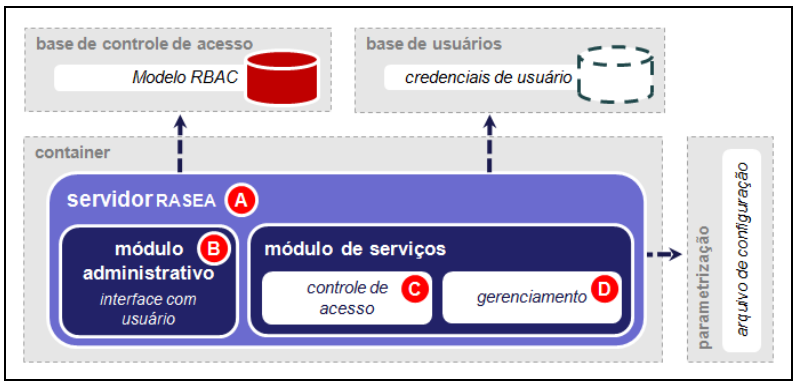

Figura 4.9 – Arquitetura do servidor RASEA

O servidor é o ponto central da arquitetura RASEA, responsável por: (i) implementar o modelo RBAC; (ii) prover serviços por meio de *Web Services*; (iii) integrar com dados da organização e; (iv) disponibilizar uma interface gráfica para gerenciamento do controle de acesso. A Figura 4.9 destaca o servidor (A) interagindo com elementos essenciais para o seu funcionamento. O item 'B' representa a interface para gerenciamento do controle de acesso às aplicações parceiras registradas no RASEA. Os itens 'C' e 'D' representam os serviços providos para aplicações parceiras ou autônomas.

Todas as informações fornecidas pelos serviços do RASEA têm sempre como fonte duas bases de dados: (i) base de autorização, que implementa o modelo RBAC e; (ii) base de autenticação, que contém as credenciais dos usuários. A base de autorização é mantida única e exclusivamente pelo servidor RASEA, sendo possível efetuar modificações nos dados através do módulo administrativo ou via serviços de gerenciamento. A escolha da base de usuários é feita no arquivo de configuração do servidor.

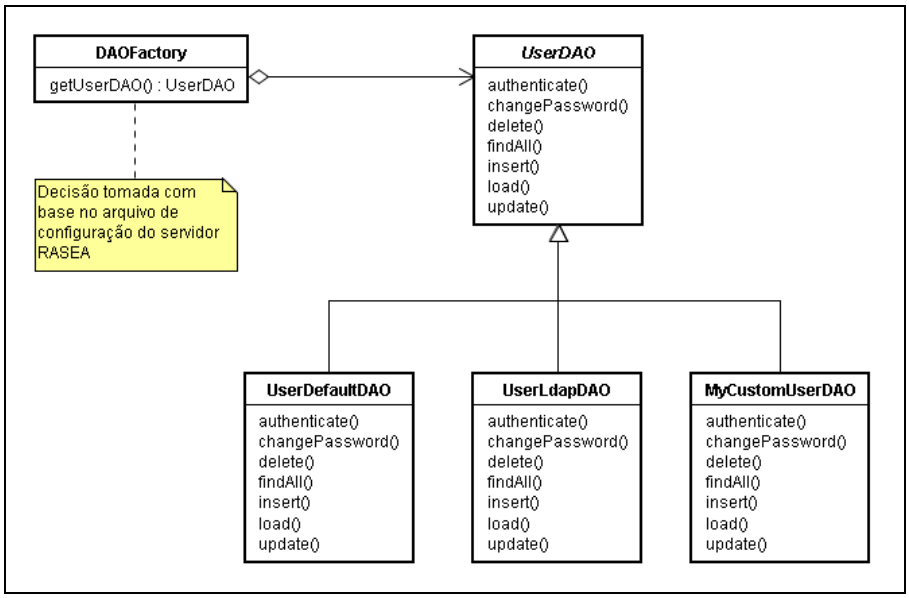

Figura 4.10 – Estratégia de armazenamento das credenciais de usuários

A relação entre o servidor e a base de credenciais é feita de forma homogênea independente da solução específica adotada. Isto permite a expansão do sistema sem a necessidade de se alterar o código do servidor. A interação com a base de credenciais dos usuários implementa o padrão de projeto *Strategy* (GAMMA, 2000, p. 292), onde o elemento decisor toma o arquivo de configuração como fonte de parâmetros. Sempre que o servidor necessita acessar a base de credenciais, uma chamada é feita à fábrica de objetos de acesso a dados (DAO, do inglês *Data Access Object*): o *DAOFactory*. Parametrizado com o arquivo de configuração do servidor (APÊNDICE A – Arquivo de configuração do servidor RASEA), a fábrica *DAOFactory* decide qual das implementações da API padrão deve-se utilizar. A Figura 4.10 ilustra os elementos envolvidos na estratégia de seleção da base de credenciais dos usuários.

O servidor RASEA tem a estrutura interna dividida em camadas, delegando responsabilidades aos seus elementos. Basicamente, o servidor está modularizado em três blocos: (i) camada de apresentação; (ii) camada de negócio e; (iii) camada de acesso aos dados. A responsabilidade da camada de apresentação é possibilitar a interação entre os consumidores e as funcionalidades do servidor, apresentando as informações de forma adequada ao canal de comunicação, tal como interface gráfica ou serviços. A camada de negócio é o núcleo do servidor, composto pelas operações do RBAC. A camada de acesso aos dados detém a lógica de persistência e manutenção das informações armazenadas nas bases de dados.

A Figura 4.11 apresenta a arquitetura em camadas do servidor RASEA, tendo como principal elemento o seu núcleo (A), envolto pelas camadas de apresentação e acesso aos dados. Os

elementos ‘B’ e ‘D’ fazem parte da camada de apresentação, que expõem as funcionalidades do servidor para os consumidores ‘C’ e ‘E’. A camada de acesso aos dados é composta pelo conjunto de DAO (F) que manipula a base de autorização (G) e pelos DAO (H) que compõem a estratégia de manipulação das credenciais do usuário (I).

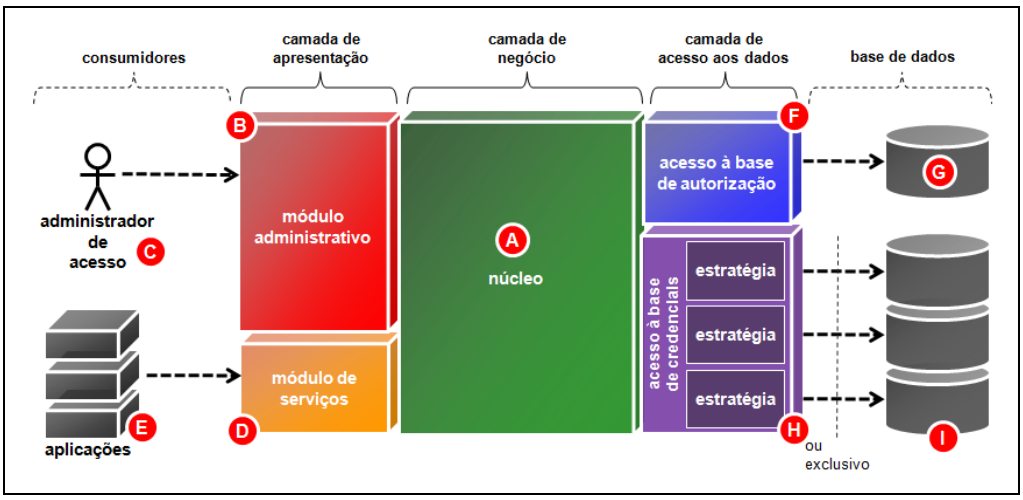

Figura 4.11 – Arquitetura em camadas do servidor RASEA

As operações providas pela interface gráfica com os usuários (GUI) devem ser restritas aos administradores de acesso da organização. Segue abaixo as funcionalidades do módulo administrativo, disponíveis via GUI no servidor RASEA:

a) **Cadastro de aplicações** – Possibilita a inclusão, alteração e exclusão de aplicações parceiras que serão controladas pelo RASEA;

b) **Cadastro de operações** – Após o cadastro de uma aplicação parceira, é preciso informar quais as operações (ou ações) fazem parte do seu controle de acesso. Esta funcionalidade contempla a inclusão, alteração e exclusão das operações de uma determinada aplicação parceira;

c) **Cadastro de recursos** – Os usuários executam operações sobre os recursos de uma aplicação parceira. Esta funcionalidade prevê a inclusão, alteração e exclusão dos recursos de uma aplicação parceira, bem como a associação com as operações que podem ser executadas sobre o recurso (ou permissões);

d) **Cadastro de usuários** – Possibilita a manipulação da base única de usuários da organização, possibilitando a inclusão, alteração e exclusão de usuários e suas credenciais;

e) **Cadastro de papéis** – Provê a inclusão, alteração e exclusão de papéis (ou grupos de usuários) de uma aplicação parceira, bem como a associação dos usuários aos papéis;

f) **Concessão de autorização** – Tendo efetuado o cadastro da aplicação parceira, suas ações, recursos, permissões e papéis; é possível conceder ou revogar acesso ao sistema. A concessão da autorização é a associação de uma permissão (recurso × operação) a um papel;

g) **Configurações** – Exibe as parametrizações do servidor definidas no arquivo de configuração. Também possibilita a seleção de idioma das telas do servidor RASEA.

Como resposta ao modelo descentralizado, a arquitetura proposta concentra todas as funcionalidades e serviços necessários em um único elemento do modelo: o servidor RASEA. Todas as aplicações parceiras dependem dos serviços para executar tarefas de controle de acesso, portanto é fundamental que o servidor possua: (i) alta disponibilidade; (ii) performance e; (iii) escalabilidade. Enquanto a alta disponibilidade assegura o atendimento às requisições dos agentes, a performance garante a execução em tempo hábil, evitando impactos no negócio da organização (KERN, 2004, p. 90). A estrutura deve ser escalável, possibilitando o crescimento sem prejuízos à alta disponibilidade e performance da solução.

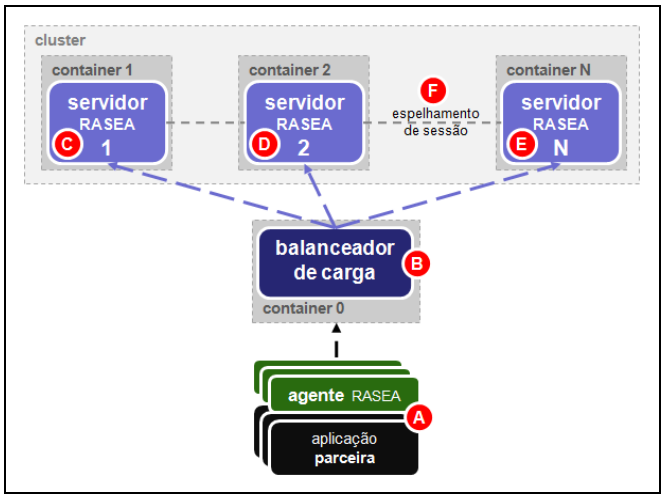

Figura 4.12 – Balanceamento de carga

Para garantir a alta disponibilidade, performance e escalabilidade, o servidor RASEA suporta o uso de *cluster* com balanceamento de carga, conforme ilustrado na Figura 4.12. O elemento 'A' representa aplicações parceiras acopladas aos agentes, que acessam os serviços do RASEA. O balanceador de carga (B) intercepta as chamadas dos agentes e decide, com base em uma heurística, qual o servidor atenderá as requisições. Os elementos 'C', 'D' e 'E'

ilustram os servidores RASEA, responsáveis por tratar as requisições delegadas pelo balanceador de carga. Cada servidor é um nó que compõe o *cluster*. A sessão de cada nó é replicada (F), garantindo o remanejamento de servidores sem interrupções no serviço. Não é necessária nenhuma configuração especial no RASEA, e sim no container, tal como o suporte do JBoss AS a *cluster* de aplicações (STANSBERRY; ZAMARRENO, 2009). O balanceamento de carga foi exaustivamente testado no RASEA.

4.5 AGENTE RASEA

Toda aplicação parceira deve estar conectada a um agente. O agente RASEA atua como um conversor do modelo independente de plataforma para a tecnologia específica de cada aplicação (IBM, 2009). Além disso, o agente executa efetivamente as tarefas de controle de acesso às aplicações, prevenindo o acesso não autorizado ao sistema. Nesta seção serão abordadas as duas principais funções do agente RASEA: (i) especializar a tecnologia e; (ii) controlar o acesso à aplicação parceira.

O agente RASEA é uma das principais contribuições deste trabalho e o grande diferencial em relação às demais soluções de controle de acesso apresentadas na subseção 2.4 – Ferramentas existentes. Os agentes simplificam o desenvolvimento de aplicações parceiras, que passam a dispor de uma API de controle de acesso local e não precisam se preocupar com detalhes de acesso ao serviço Web. Além disto, os agentes ampliam a atuação do RASEA dentro das aplicações parceiras, oferecendo suporte completo à interceptação das tentativas de acesso, prevenindo que usuários não autorizados acessem os recursos protegidos das aplicações.

Como elemento da arquitetura proposta, o agente agrega extensibilidade à solução, possibilitando que aplicações projetadas em qualquer tecnologia com suporte ao consumo de *Web Services* integrem com o RASEA, garantindo o controle de acesso unificado de forma independente de plataforma. A Figura 4.13 destaca a especialização do agente RASEA em diferentes tecnologias. O item ‘A’ representa os serviços, providos pelo servidor, que são acessados pelos agentes. Os itens ‘B’, ‘C’ e ‘D’ representam especializações de agentes, construídos em tecnologias específicas, tais como: Java, .NET e PHP. O item ‘E’ representa agentes desenvolvidos em qualquer tecnologia que suporte o consumo de *Web Services*, proporcionando flexibilidade à solução.

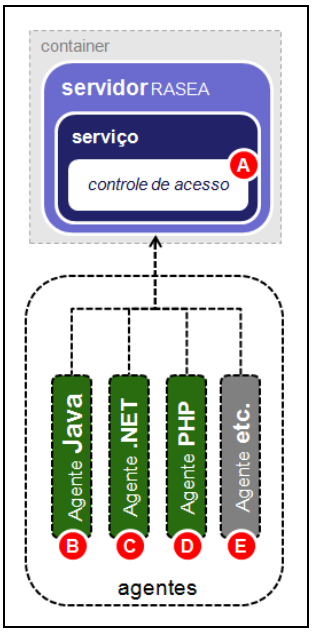

Figura 4.13 – Tipos de agente

Os agentes devem se especializar ao máximo, garantindo o encaixe perfeito com a tecnologia da aplicação parceira. Como exemplo, pode-se citar a utilização de Java na construção de aplicações comerciais. Existem diversas ferramentas, paradigmas, *frameworks*, bibliotecas e tecnologias que podem ser utilizadas na construção de aplicações na plataforma Java, tais como: (i) *JavaServer Faces* (JSF); (ii) *Apache Struts*; (iii) *Google Web Toolkit* (GWT); (iv) *Abstract Window Toolkit* (AWT); dentre outras. É fundamental que exista um agente para cada tecnologia específica.

O agente é responsável por interceptar todas as requisições feitas às aplicações parceiras, conforme estabelecido pelo padrão de projeto *Front Controller* (ALUR; CRUPI; MALKS, 2003, p. 164). Todas as tentativas de acesso devem ser tratadas com base nas informações providas pelo servidor RASEA. De acordo com Kern (2004, p. 90), vagarosidade nas decisões de acesso podem ser críticas para o negócio da organização, portanto é fundamental armazenar em *cache* todas as permissões associadas ao usuário. Enquanto a Figura 4.14 representa o processo executado pelo agente para realizar a autenticação e o armazenamento dos privilégios de acesso na sessão, a Figura 4.15 descreve as etapas do controle de acesso, executadas pelo agente RASEA, ao interceptar requisições aos recursos da aplicação parceira.

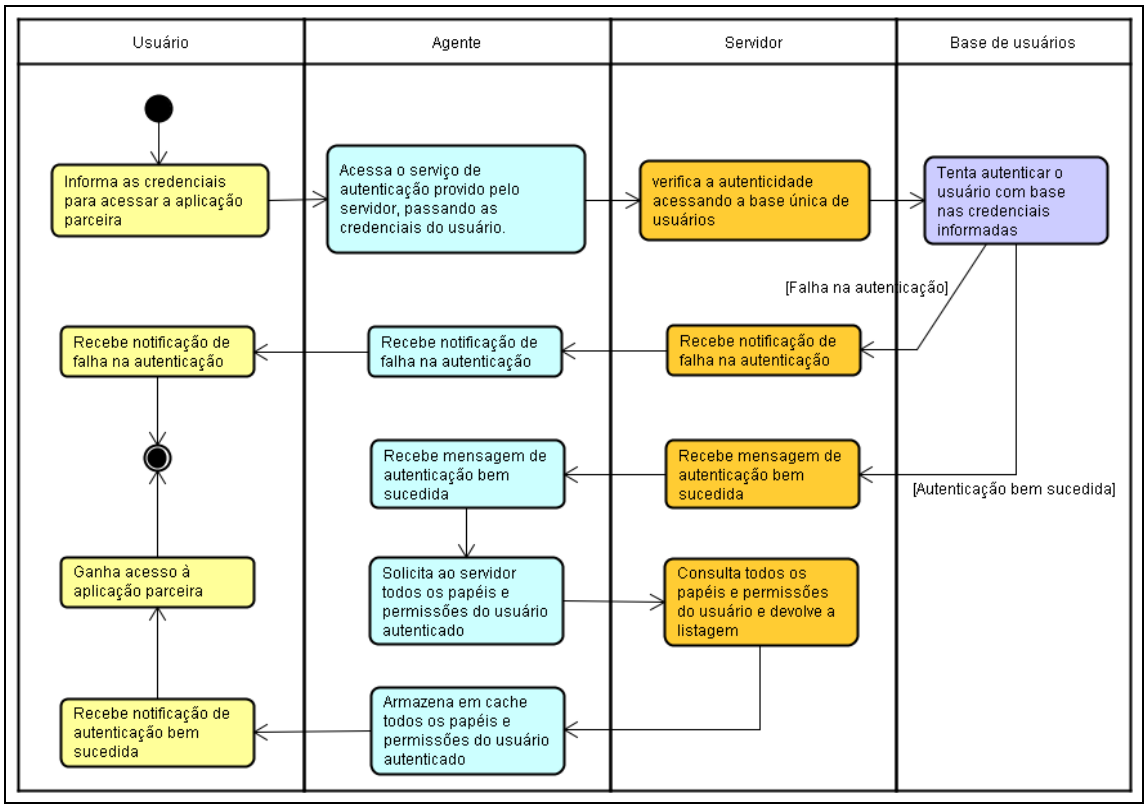

Figura 4.14 – Processo de autenticação e armazenamento dos privilégios de acesso

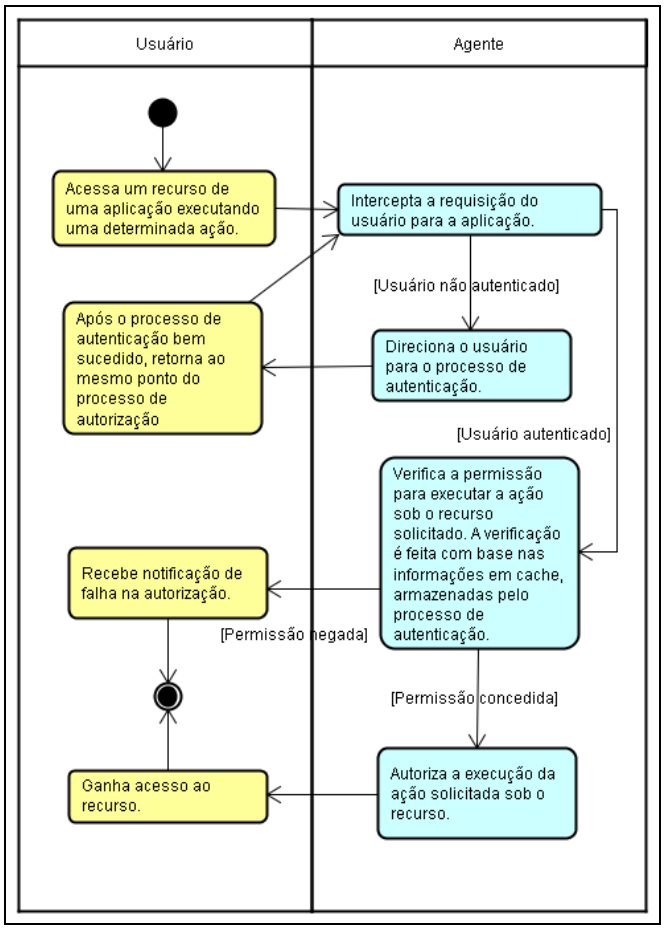

Figura 4.15 – Processo de verificação e autorização de acesso

O servidor RASEA é também uma aplicação que requer o controle de acesso às suas funcionalidades. Este requisito possui um caráter recursivo (DAVISON; HINKLEY, 1997), visto que o servidor acessa os seus próprios serviços para garantir o controle de acesso a si mesmo, caracterizando-o como uma aplicação parceira. A Figura 4.16 ilustra o agente acoplado ao servidor (B) acessando os serviços (D) por meio de uma conexão (C). Percebe-se que a estrutura do servidor é a mesma adotada pelas aplicações parceiras (A).

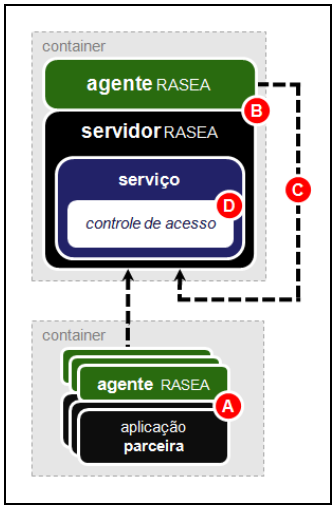

Figura 4.16 – Agente do servidor RASEA consumindo os serviços de controle de acesso

O agente conectado ao servidor RASEA foi implementado na tecnologia *Seam Security* (SEAM, 2009), visto que o servidor é compatível com o *JBoss Seam Framework* para plataforma Java. O processo de interceptação das requisições à aplicação está de acordo com a especificação oficial da ferramenta (SEAM, 2009), materializada na classe *RaseaPermissionResolver*, que implementa a API *PermissionResolver* responsável por tratar todas as requisições de acesso aos recursos da aplicação. O código-fonte do agente utilizado pelo servidor RASEA está disponível publicamente no repositório *open source* do projeto (http://rasea.svn.sourceforge.net/svnroot/rasea/agent/java/seam/) regido pela licença LGPL (2009), como uma das contribuições deste trabalho.

O agente na tecnologia *Seam Security* está sendo utilizado na Fábrica de *Software* da Avansys Tecnologia para integrar as aplicações produzidas com a solução unificada de controle de acesso: o RASEA. Atualmente 6 (seis) aplicações comerciais construídas pela Fábrica de *Software* da Avansys utilizam o agente em ambiente de produção. O agente *Seam Security* foi inicialmente desenvolvido para a versão 2.0.2 do *JBoss Seam Framework*, e foi completamente reprojetado para atender aos requisitos de segurança da especificação *Seam Security* para as versões do posteriores à 2.0.2, tais como: (i) implementação do *IdentityStore*

e; ii) criação do *PermissionResolver*. Atualmente o *JBoss Seam Framework* encontra-se disponível na versão 2.2.0. O APÊNDICE G – Configuração do agente *Seam Security* em uma aplicação parceira – ilustra a configuração de uma aplicação que utiliza o agente *Seam Security*.

Os clientes da Avansys Tecnologia, que adquiriram mais de uma aplicação comercial construída pela Fábrica de *Software*, solicitaram que, ao acessar a aplicação, suas credenciais não fossem solicitadas novamente, caso já houvessem informado-as previamente. Como exemplo, os usuários citaram os serviços *Web* da empresa Google: ao acessar o serviço *Mail* (http://mail.google.com) informando as credenciais, é possível alternar entre os serviços *Calendar* (http://calendar.google.com) e *Docs* (http://docs.google.com) sem necessidade de informar os dados da conta novamente. Esta funcionalidade foi disponibilizada no agente RASEA para *Seam Security* a partir da versão 0.1.0.RC4.

O agente *Seam Security* é parametrizado com base no arquivo de configuração que contém informações a respeito do servidor RASEA. Desta forma, o agente deverá ser re-parametrizado sempre que ocorrerem mudanças na infra-estrutura do servidor, tais como: (i) alteração de porta; (ii) mudança de endereço; (iii) mudança do protocolo. O detalhamento das configurações está descrito no APÊNDICE B – Arquivo de configuração do agente RASEA.

Existe no mercado uma grande variedade de tecnologias e linguagens de programação. Deve-se construir um agente para cada tecnologia que pretenda utilizar o RASEA. O projeto RASEA está aberto à colaboração da comunidade para a criação de novos agentes, seguindo o modelo *Bazaar* proposto por Raymond (2001). Durante o desenvolvimento do projeto, diversos codificadores e arquitetos contribuíram com sugestões para a evolução do projeto *open source* RASEA, principalmente do elemento 'agente'.

A premissa para todo agente é atuar no ponto central, onde todas as requisições aos recursos das aplicações sejam interceptadas. Para conseguir isto, existem soluções simplórias e soluções arrojadas. Um exemplo de solução simples para agentes *Web* é a inserção da *tag* *<include>* em todas as páginas, invocando um determinado procedimento que atua como o controlador de acesso, buscando informações no servidor RASEA. Esta é uma solução simples, porém pouco elegante. Como exemplo de solução robusta para a plataforma Java, tem-se o *Java Authentication and Authorization Service* (JAAS), que atua diretamente no

núcleo da máquina virtual, possibilitando o controle refinado das requisições feitas pelos usuários às aplicações.

A Fábrica de *Software* da Avansys utiliza o sistema de versionamento *Subversion* (SVN) (http://subversion.tigris.org) para controlar a evolução dos artefatos gerados durante o ciclo de desenvolvimento do *software*. Para controlar o acesso ao repositório, a Avansys utiliza o recurso *authzfile* do SVN, que possui a listagem das permissões a todos os repositórios em um único arquivo, dificultando o gerenciamento do controle de acesso. Com isso, surgiu a idéia de implementar gatilhos do SVN (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2007, p. 116), que atuariam como agentes RASEA.

Como o SVN não é caracterizado como uma aplicação comercial (s. 1.1), e por não fazer parte do escopo deste trabalho, o agente não foi implementado. Contudo, foram feitos estudos baseados no material oficial da ferramenta (COLLINS-SUSSMAN; FITZPATRICK; PILATO, 2007), que indicam a possibilidade de concepção do agente SVN. A partir de então, percebeu-se que a solução e a arquitetura propostas neste trabalho, graças aos agentes, podem beneficiar não somente aplicações comerciais, e sim sistemas em geral que necessitam controlar o acesso aos seus recursos.

5 CONSIDERAÇÕES FINAIS

Neste capítulo, serão apresentados os resultados obtidos a partir de análises elaboradas durante o desenvolvimento deste trabalho. As considerações finais abrangem desde as ferramentas existentes, passando pela evolução do projeto, até a especificação da arquitetura do projeto RASEA. Não obstante, foram detectadas oportunidades de melhoria da solução proposta, motivando a sua evolução em trabalhos futuros. Por fim, conclui-se que o projeto RASEA é capaz de promover: (i) redução de custos no desenvolvimento de aplicações comerciais e; (ii) simplificação na administração do controle de acesso dos sistemas corporativos.

5.1 RESULTADOS OBTIDOS

Almejando o cumprimento do objetivo deste trabalho, os resultados obtidos foram avaliados sob três perspectivas: (i) análise da evolução do projeto; (ii) comparação das ferramentas existentes no mercado e; (iii) construção do projeto RASEA. A análise da evolução do projeto validou as propostas deste trabalho baseadas na extensa experiência acumulada, enquanto a pesquisa comparativa das ferramentas existentes ponderou os pontos fortes e fracos de cada tecnologia.

Os cenários apresentados na evolução do projeto foram estrategicamente selecionados para experimentar e validar o objetivo geral deste trabalho, que é "prover uma solução orientada a serviços que suporte o controle de acesso unificado de usuários às aplicações, fundamentando-se em conceitos e tecnologias existentes para garantir a interoperabilidade em ambientes heterogêneos" (s. 1.3). O Cenário A – Sistemas Internos (s. 3.1) – integrou aplicações em diferentes plataformas, possibilitando o controle de acesso unificado às aplicações, atestando o seguinte objetivo específico: "garantir que a arquitetura proposta beneficie aplicações em diferentes plataformas" (s. 1.3).

No Cenário B – Projeto Inscrição Online (s. 3.2) – evidenciou a abordagem orientada a serviços atendendo aos requisitos de alta disponibilidade e performance em cenários geograficamente distribuídos. Demonstrando a viabilidade econômica do projeto, o Cenário C

– Fábrica de *Software* (s. 3.3) – apresentou números importantes para tomada de decisões estratégicas, respaldados pela métrica de Análise de Pontos de Função. Houve uma economia de até 31% (trinta e um por cento) no custo do desenvolvimento de aplicações comerciais em regime de Fábrica de *Software*.

A análise das ferramentas existentes no mercado (s. 2.4), consolidadas na matriz comparativa (Figura 2.10, p. 30), mostra que as ferramentas MACA e OES possibilitam a unificação do controle de acesso *cross*-plataforma de aplicações comerciais, no entanto ambas atendem parcialmente os critérios estabelecidos. O projeto RASEA tem como objetivo suprir esta demanda, tornando-se referência de mercado, atendendo a todos os requisitos estabelecidos, conforme a Figura 5.1.

	Autenticação	Autorização	RBAC	Padrões abertos	Código aberto / livre	Suporte	Web	Módulo de gerenciamento	Unificação	Independência de platafor
Seam Security	✓	✓	!	✓	✓	✓	!		!	✗
MACA	✓	✓	!	✓	✓	!	!	✓	✓	✓
OES	✓	✓	!	✓	✗	✓	✓	✓	✓	✓
rasea	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Legenda

- ✓ Atende completamente
- ! Atende parcialmente
- ✗ Não atende
- Não se aplica / impossível avaliar

Figura 5.1 – Comparativo entre as ferramentas existentes e o RASEA

O projeto RASEA foi concebido como contribuição prática deste trabalho, conforme o seguinte objetivo específico: "implementar a solução proposta e disponibilizá-la à comunidade por meio de uma licença *open source*" (p. 15). O RASEA é dividido em dois subprojetos: (i) **servidor**, regido pela licença GPL e; (ii) **agente**, regido pela licença LGPL. O projeto está em constante evolução e pode ser acompanhado pela rede social *Twitter* (http://www.twitter.com) através da conta *RaseaProject* (http://twitter.rasea.org). Os detalhes da elaboração e construção do RASEA podem ser acessados através do *blog* do projeto (http://blog.rasea.org), disponível publicamente para toda a comunidade. No decorrer deste

trabalho, programadores autônomos ofereceram sugestões e contribuições, bem como algumas empresas demonstraram interesse na implantação do RASEA em suas instalações.

Desde o primeiro semestre de 2009, a Avansys Tecnologia apóia o projeto RASEA, dedicando esforços na evolução da ferramenta. De acordo com o planejamento da organização, seu produto S4A será completamente substituído pelo RASEA em toda linha de produção da sua Fábrica de *Software* até o final de 2009. Além disso, a Avansys Tecnologia adotou o RASEA como ferramenta padrão para controle de acesso às suas aplicações internas, que suportam as atividades de *backend* da empresa.

Como principais contribuições deste trabalho, duas se destacam: (i) o *design* de uma solução que concentra os elementos essenciais de controle de acesso baseado nas experiências adquiridas durante a evolução do projeto e; (ii) o elemento 'agente' da arquitetura proposta. No decorrer do desenvolvimento dos projetos AvanSEG, S4A e o RASEA – idealizados, gerenciados, projetados pelo autor deste trabalho e implementados por ele e sua equipe – a solução adquiriu uma extensa experiência graças, principalmente, ao envolvimento de 12 (doze) administradores de acesso de organizações distintas. Ao final do processo com duração superior a 30 (trinta) meses, os usuários elogiaram a simplicidade provida pela solução para controle de acesso e gerenciamento das permissões. O elemento 'agente' agregou extensibilidade à solução, promovendo-se o grande diferencial em relação às soluções de mercado tomando com base o objetivo deste trabalho.

A simplicidade no *design* do RASEA tem sido uma decisão de projeto que tem permitido não apenas a atuação de profissionais não especialistas em segurança na administração de acesso a aplicações, como também pode melhorar o controle na segurança do sistema, dado que o administrador não se perde em demasiados detalhes, concentrando-se nos elementos essenciais do controle de acesso. Alcançar a simplicidade foi resultado de um longo processo interativo de projeto, implementação e teste, garantindo que a solução implementada atenda às principais necessidades do público-alvo do RASEA.

5.2 TRABALHOS FUTUROS

Durante o desenvolvimento deste trabalho foram identificadas diversas oportunidades de melhoria e evolução do projeto ainda não implementadas, enumeradas a seguir: (i) implementação do *Hierarchical* e *Constrained* RBAC; (ii) criação de agentes em diferentes tecnologias; (iii) seleção de outros métodos de autenticação e; (iv) utilização dos padrões SAML e XACML para segurança dos *Web Services*.

Conforme descrito no tópico 'RBAC' (s. 2.2), o projeto concentra-se no *Core* RBAC (Figura 2.3). Sabendo-se da fundamental importância dos componentes *Hierarchical* RBAC (FERRAIOLO, 2001, p. 234) e *Constrained* RBAC (FERRAIOLO, 2001, p. 238) para aplicações comerciais, sugere-se o aprofundamento teórico e a implantação formal destes conceitos na solução RASEA. O *Hierarchical* RBAC define a hierarquia de papéis (Figura 2.5), enquanto o *Constrained* RBAC é utilizado para resolver os conflitos de interesse dos papéis de uma organização (FERRAIOLO, 2001, p. 238).

Para garantir a integração com aplicações parceiras em plataformas e tecnologias distintas, o agente atua como um motor de especialização. Como contribuição prática deste trabalho, construiu-se o agente para a plataforma Java integrado com a ferramenta *Seam Security*. É fundamental a criação de novos agentes em outras plataformas e tecnologias, tais como: .NET e PHP. De acordo com o planejamento de colaboradores autônomos, funcionários de um órgão público, um agente PHP está em fase de construção e será concluído até dezembro de 2009.

Este trabalho focou no processo de autorização e controle de acesso de aplicações comerciais (s. 2.1), não aprofundando em técnicas e procedimentos de autenticação. Sugere-se a elaboração e construção de funcionalidades que suportem distintos métodos de autenticação, tais como: OpenID e Certificado Digital. Durante os testes de integração do RASEA com tecnologias de mercado, concebeu-se um agente responsável por delegar tarefas de autenticação para a ferramenta *Java Open Single Sign-On* (JOSSO) (http://www.josso.org), que está fase de estudo e testes.

Tal como o produto OES, uma melhoria consistirá na utilização da *Security Assertion Markup Language* (SAML) e da *eXtensible Access Control Markup Language* (XACML) nos serviços

expostos da ferramenta RASEA. Os padrões SAML e XACML, mantidos pela OASIS, são largamente utilizados no mercado para segurança de *Web Services*. A ferramenta RASEA está implantando gradativamente tais padrões na definição das políticas de acesso aos seus serviços. A identificação, avaliação, implantação e acompanhamento de oportunidades de melhoria são fundamentais para a evolução contínua do projeto.

Durante a concepção do agente RASEA, notou-se a possibilidade de elaboração de um cliente para o sistema de controle de versões SVN. Notou-se então que é possível integrar não somente com aplicações comerciais, e sim com sistemas em geral que necessitem restringir o acesso aos seus recursos ou às suas partes. Desta maneira, tornou-se evidente que o projeto RASEA possibilita a criação de extensões para diversas tecnologias, necessitando um estudo aprofundado neste sentido para delimitação de possíveis integrações e reuso da solução proposta neste trabalho em novos cenários.

REFERÊNCIAS

ABNT: ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 17799**: tecnologia da informação: técnicas de segurança: código de prática para a gestão da segurança da informação. Rio de Janeiro, 2005.

_____. **NBR ISO/IEC 27001**: tecnologia da informação: técnicas de segurança: sistemas de gestão de segurança da informação: requisitos. Rio de Janeiro, 2006.

ALONSO, Gustavo et al. **Web services**: concepts, architectures and applications. [S.I.]: Springer-Verlag, 2004. ISBN: ISBN 3-540-44008-9.

ALUR, Deepak; CRUPI, John; MALKS, Dan. **Core J2EE Design Patterns**: Best practices and Design Strategies. 2 ed. Upper Saddle River: Prentice Hall PTR, 2003. ISBN: 0-13-142246-4.

ANAYA, Victor; ORTIZ, Angel. How enterprise architectures can support integration. In: PROCEEDINGS OF THE FIRST INTERNATIONAL WORKSHOP ON INTEROPERABILITY OF HETEROGENEOUS INFORMATION SYSTEMS, 2005, Bremen. **Anais...** New York: ACM, 2005. p. 25–30. ISBN: 1-59593-184-5.

ANSI: AMERICAN NATIONAL STANDARD INSTITUTE. **ANSI INCITS 359-2004**: Information Technology: Role Based Access Control. New York, 2004.

ARKER, Steve. Access control by action control. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 13, 2008, Estes Park. **Anais...** New York: ACM, 2008. p. 143–152. ISBN: 978-1-60558-129-3.

BARKER, Steve. Access control by action control. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 2008, Estes Park. **Anais...** New York: ACM, 2008. p. 143–152. ISBN: 978-1-60558-129-3.

BERTINO, Elisa et al. A logical framework for reasoning about access control models. In: ACM WORKSHOP ON ROLE BASED ACCESS CONTROL, 2001, Chantilly. **Anais...** New York: ACM, 2001. p. 41–52. ISBN: 1-58113-350-2.

BEZNOSOV, Konstantin; DENG, Yi. A framework for implementing role-based access control using CORBA security service. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 1999, Fairfax. **Anais...** New York: ACM, 1999. ISBN: 1-58113-180-1.

BOX, Don. **A Brief History of SOAP**. [S.I.]: O´ Reilly, 2001. Disponível em: <http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html>. Acesso em: 13 de fev. 2009.

BUEGE, Brian; LAYMAN, Randy; TAYLOR, Art. **Segurança contra Hackers J2EE e Java**: Desenvolvendo aplicações seguras com a tecnologia Java. São Paulo: Futura, 2003. ISBN: 85-7413-156-3.

BUTEK, Russell; GALLARDO, Nicholas. **Web services hints and tips**: JAX-RPC versus JAX-WS, Part 1. [S.I.]: IBM developerWorks, 2006. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-tip-jaxwsrpc.html>. Acesso em: 07 de fev. 2009.

COLLINS-SUSSMAN, Ben; FITZPATRICK, Brian W.; PILATO, C. Michael. Version **Control with Subversion**: For Subversion 1.4. [S.I.]: Tigris.org, 2007.

CUMMINS, Fred A. **Integração de Sistemas**: EAI – Enterprise Application Integration. Rio de Janeiro: Campus, 2002. ISBN: 85-352-0975-1.

DAN, Asit; JOHNSON, Robert D.; CARRATO, Tony. SOA service reuse by design. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2008, Leipzig. **Anais…** New York: ACM, 2008. p. 25–28. ISBN: 978-1-60558-029-6.

DAVIES, Jeff; KRISHNA, Ashish; SCHOROW, David. **The Definitive Guide to SOA**: BEA AquaLogic Service Bus. New York: Apress, 2007. ISBN-13: 978-1-59059-797-2.

DAVISON, Anthony C.; HINKLEY, D. V. **Bootstrap methods and their application**. [S.I.]: Cambridge University Press, 1997. ISBN: 0521574714.

DIRGAHAYU, Teduh; QUARTEL, Dick; SINDEREN, Marten van. Designing Interaction Behaviour in Service-Oriented Enterprise Application Integration. In: SYMPOSIUM ON APPLIED COMPUTING, 2008, Fortaleza. **Anais…** New York: ACM, 2008. p. 1048–1054. ISBN: 978-1-59593-753-7.

EDEN, Amnon H.; KAZMAN, Rick. Architecture, Design, Implementation. In: CONFERENCE ON SOFTWARE ENGINEERING, 2003. Disponível em: <http://www.eden-study.org/articles/2003/icse03.pdf>. Acesso em: 20 de fev. 2009.

FERRAIOLO, David F. et al. Proposed NIST standard for role-based access control. **ACM Transactions on Information and System Security (TISSEC)**, v. 4, n. 3, p. 224–274, 2001. ISSN:1094-9224.

GAEDKE, Martin; MEINECKE, Johannes; NUSSBAUMER, Martin. A modeling approach to federated identity and access management. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE, 14, 2005, Chiba. **Anais...** New York: ACM, 2005. p. 1156–1157. ISBN: 1-59593-051-5.

GAMMA, Erich et al. **Padrões de Projeto**: Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2000. ISBN: 85-7307-610-0.

GPL: GNU General Public License. Version 3. [S.I.]: Free Software Foundation, 2007. Disponível em: <http://www.gnu.org/licenses/gpl.html>. Acesso em: 1 mai. 2009.

HULL, Richard; SU, Jianwen. Tools for Composite Web Services: A Short Overview. **ACM SIGMOD Record**, New York: ACM , v. 34, n. 2, p. 86–95, jun. 2005. ISSN: 0163-5808.

IBM**. The PIMA Project**: Platform-Independent Model for Applications. [S.I.]: IBM Research. Disponível em <http://www.research.ibm.com/PIMA/>. Acesso em: 28 de fev. 2009.

ISO: INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 10181-3**: information technology, open systems interconnection, security frameworks for open systems: access control framework. [S.I.], 1996.

JAAS, **Java Authentication and Authorization Service**: Reference Guide. Version 1.5.0. [S.I]: Sun Microsystems, 2001. Disponível em: <http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html>. Acesso em: 27 jan. 2009.

JONES, Steve. **Enterprise SOA Adoption Strategies**: Using SOA to Deliver IT to the Business. [S.I.]: InfoQ, 2006. ISBN: 978-1-84728-398-6.

JOSUTTIS, Nicolai M. **SOA in Practice**: The Art of Distributed System Design. 1. ed. Sebastopol: O´ Reilly, 2007. ISBN-10: 0-596-52955-4

JSE, **Java 2 Platform Standard Edition 5.0**: New Features and Enhancements. Version 1.5.0. [S.I]: Sun Microsystems, 2004. Disponível em: <http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html>. Acesso em: 12 mai. 2009.

JSR: JAVA SPECIFICATION REQUESTS. **JSR-000314**: JavaServer Faces 2.0 (Final Release). [S.I.], 2009. Disponível em: <http://jcp.org/aboutJava/communityprocess/final/jsr314/index.html>. Acesso em: 11 jul. 2009.

KERN, Axel et al. A meta model for authorisations in application security systems and their integration into RBAC administration. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 9, 2004, Yorktown Heights. **Anais...** New York: ACM, 2004. p. 87–96. ISBN: 1-58113-872-5.

KRUCHTEN, Philippe. Architectural Blueprints – the 4+1 View Model of Software Architecture. **IEEE Software**, [S.I.], v. 12, n. 6, p. 42–50, nov. 1995. Disponível em: <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf >. Acesso em: 20 de fev. 2009.

LGPL: **GNU Lesser General Public License**. Version 3. [S.I.]: Free Software Foundation, 2007. Disponível em: <http://www.gnu.org/licenses/lgpl.html>. Acesso em: 15 jan. 2009.

LI, Jun; KARP, Alan H. Access control for the services oriented architecture. In: WORKSHOP ON SECURE WEB SERVICES, 2007, Fairfax. **Anais...** New York: ACM, 2007. p. 9–17. ISBN: 978-1-59593-892-3.

MOTTA, Gustavo H. **Um modelo de autorização contextual para o controle de acesso ao prontuário eletrônico do paciente em ambientes abertos e distribuídos**. São Paulo: Universidade de São Paulo, Escola Politécnica, 2004. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3142/tde-05042004-152226/>. Acesso em: 15 abr. 2009.

ORACLE. **Oracle Entitlements Server**: White Paper. Redwood Shores: Oracle Corporation, 2008. Disponível em: <http://www.oracle.com/technology/products/id_mgmt/oes/pdf/oes_entitlements.pdf>. Acesso em: 27 nov. de 2008.

PAPAZOGLOU, Mike P.; HEUVEL, Willem-Jan. Service oriented architectures: approaches, technologies and research issues. **The International Journal on Very Large Data Bases**, Springer-Verlag, v. 16, n. 3, p. 389–415, jul. 2007. ISSN: 1066-8888.

PILIPCHUK, Denis. **Adapting JAAS to SOA Environments**: SOA Security Service. Disponível em: <http://today.java.net/pub/a/today/2007/02/06/adapting-jaas-to-soa-environments.html>. Acesso em: 27 nov. de 2008.

PRESMAN, Roger S. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002. ISBN: 85-86804-25-8.

RAY, Indrakshi et al. Using uml to visualize role-based access control constraints. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES, 2004, New York. **Anais...** New York: ACM, 2004. p. 115–124. ISBN: 1-58113-872-5.

RAYMOND, Eric S. **The Cathedral & the Bazaar**: Musings on Linux and Open Source by an Accidental Revolutionary. [S.I.]: O´ Reilly, 2001. ISBN: 0-596-00108-8.

SAMPAIO, Cleuton. **SOA e Web Services em Java**. Rio de Janeiro: Brasport, 2006. ISBN: 85-7452-267-8.

SANDHU, Ravi et al. Role-Based Access Control Models. **IEEE Computer**, [S.I.], v. 29, n. 2, p. 38–47, fev. 1996. Disponível em: <http://csrc.nist.gov/rbac/sandhu96.pdf>. Acesso em: 03 fev. 2009.

SANDHU, Ravi; FERRAIOLO, David; KUHN, Richard. **The NIST Model for Role-Based Access Control**: Toward A Unified Standard. [S.I]: National Institute of Standards and Technology, 2000. Disponível em: <http://cs-www.ncsl.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf>. Acesso em: 10 jan. 2009.

SCHEMA, Mike. **Hack Notes**: Segurança na Web. Rio de Janeiro: Elsevier, 2003. ISBN: 85-352-1350-3.

SEAM Security: **The Seam Security API**. Version 2.1.1.GA. [S.I], 2009. Disponível em: <http://docs.jboss.org/seam/2.1.1.GA/reference/en-US/html/security.html>. Acesso em: 15 abr. 2009.

SHAW, Mary; GARLAN, David. **An Introduction to Software Architecture**. School of Computer Science – Carnegie Mellon University: Pittsburgh, 1994.

_____. **Software Architecture**: Perspectives on an Emerging Discipline. Upper Saddle River: Prentice Hall, 1996. ISBN: 978-0131829572.

SILVA, Lino S. **Public Key Infrastructure PKI**: Conheça a Infra-estrutura de Chaves Púbicas e a Certificação Digital. São Paulo: Novatec, 2004. ISBN: 85-7522-046-2.

SMITH-FERRIER, Guy. **.NET Internationalization**: The Developer's Guide to Building Global Windows and Web Applications. [S.I.]: Addison-Wesley Professional, 2006. ISBN: 0-321-34138-4.

SONG, Eunjee et al. Verifiable composition of access control and application feature. In: SYMPOSIUM ON ACCESS CONTROL MODELS AND TECHNOLOGIES ARCHIVE, 2005, Stockholm. **Anais...** New York: ACM, 2005. p. 120–129. ISBN: 1-59593-045-0.

STANSBERRY, Brian; ZAMARRENO, Galder. **JBoss Application Server Clustering Guide**. [S.I.]: JBoss Labs. Disponível em: <http://labs.jboss.com/file-access/default/members/jbossas/freezone/docs/Clustering_Guide/4/html/index.html>. Acesso em: 15 abr. 2009.

VAZQUEZ, Carlos E.; SIMÕES, Guilherme S.; ALBERT, Renato M. **Análise de Pontos de Função**: Medição, Estimativas e Gerenciamento de Projetos de Software. 4. ed. São Paulo: Érica, 2003. ISBN: 85-7194-899-2.

VERHANNEMAN,Tine et al. Implementing a modular access control service to support application-specific policies in CaesarJ. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 118, 2005, Grenoble. **Anais...** New York: ACM, 2005. ISBN: 1-59593-265-8.

W3C, **SOAP Specifications**. Version 1.2. [S.I], 2007. Disponível em: <http://www.w3.org/TR/soap/>. Acesso em: 16 jun. 2009.

WANG, Hua et al. A framework for role-based group delegation in distributed environments. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 171, 2006, Hobart. **Anais...** Darlinghurst: Australian Computer Society, 2006. p. 321–328. ISBN ~ ISSN: 1445-1336, 1-920682-30-9.

XMLRPC: Arquitetura do XML-RPC. xmlrpc.jpg. 1999. Altura: 270 pixels. Largura: 527 pixels. 96 dpi. 24 bit. 22.9 Kb. Formato Arquivo JPG. Compactado. Disponível em: <http://static.userland.com/images/XML-RPC/xmlrpc.jpg>. Acesso em: 16 jun. 2009.

YU, Qui et al. Deploying and managing Web services: issues, solutions, and directions. **The International Journal on Very Large Data Bases**, Springer-Verlag, v. 17, n. 3, p. 537–572, mai. 2008. ISSN: 1066-8888.

APÊNDICE A – Arquivo de configuração do servidor RASEA

O servidor RASEA deve ser configurado a partir do arquivo de configuração, que deve estar disponível no *classpath* do servidor de aplicação. No JBoss AS, este arquivo deve estar na pasta ***<JBOSS_HOME>/server/default/conf/*** com o nome ***rasea-server.properties***. O arquivo consiste em diversas entradas 'atributos × valor' separadas por quebras de linha. Os atributos são separados pelo sinal '=' dos seus respectivos valores.

1. Configurações administrativas

Parâmetro	Descrição	Valor padrão
admin.password (opcional)	Senha do usuário administrador.	rasea
admin.email (opcional)	E-mail do usuário administrador.	admin@rasea.org

Exemplo:

admin.password=secret
admin.email=my@domain.com

2. Configurações de e-mail

Parâmetro	Descrição	Valor padrão
mail.host (opcional)	Nome ou IP do *host* SMTP para envio de mensagens.	
mail.tls (opcional)	Indica se a conexão é segura ou não: - **true**: conexão segura; - **false**: conexão não segura;	false
mail.port (opcional)	Porta SMTP para envio de mensagens.	25 (ou 456, caso TLS seja true)
mail.username (opcional)	Nome do usuário para autenticação no servidor SMTP.	
mail.password (opcional)	Senha do usuário para autenticação no servidor SMTP.	

Exemplo:

admin.password=secret
admin.email=my@domain.com
store.type=default
strore.readonly=false
mail.host=smtp.gmail.com
mail.port=465
mail.tls=true

mail.username=hello@gmail.com
mail.password=123456

3. Configurações da base de usuários

Parâmetro	Descrição	Valor padrão
store.type (opcional)	Seleção da base única de usuários: - **default**: conexão com a base embutida; - **ldap**: conexão LDAP; - **custom**: base de usuários customizada;	default
store.readonly (opcional)	Indica se a base de usuários é apenas para leitura: - **true**: apenas leitura; - **false**: leitura e escrita;	False

Exemplo:

admin.password=secret
admin.email=my@domain.com
store.type=default
strore.readonly=false

4. Configurações da base de usuários customizada

Parâmetro	Descrição	Valor padrão
custom.class	Indica a classe customizada que implementa a *interface* UserDAO. A classe deve estar disponível no *classpath* do Servidor de Aplicação.	

Exemplo:

admin.password=secret
admin.email=my@domain.com
store.type=custom
strore.readonly=false
mail.host=smtp.gmail.com
mail.port=465
mail.tls=true
mail.username=hello@gmail.com
mail.password=123456
custom.class=com.domain.MyCustomUserDAO

5. Configurações da base de usuários LDAP

Parâmetro	Descrição	Valor padrão
ldap.server.protocol (opcional)	Protocolo utilizado na conexão com o servidor LDAP: - **ldap**: conexão sem segurança; - **ldaps**: conexão segura;	ldap
ldap.server.host	Nome ou IP do *host* do servidor LDAP.	
ldap.bind.dn	Nome distinto do usuário do LDAP com privilégios de leitura e escrita.	
ldap.bind.password	Senha do usuário especificado no atributo 'ldap.bind.dn'.	
ldap.user.search-context-dn	Base de pesquisa dos usuários na estrutura do LDAP.	
ldap.user.insert-pattern-dn	Padrão a ser utilizado como nome distinto na inserção de novos usuários na base ldap.	
ldap.user.name-attribute (opcional)	Nome do atributo que identifica o nome (*login*) do usuário.	sAMAccountName
ldap.user.displayName-attribute (opcional)	Nome do atributo que identifica o nome de exibição do usuário.	givenName
ldap.user.mail-attribute (opcional)	Nome do atributo que identifica o e-mail principal do usuário. As notificações do RASEA para o usuário utilizarão este e-mail, tal como entrega da senha gerada automaticamente pelo RASEA.	
ldap.user.alternateMail-attribute (opcional)	Nome do atributo que identifica o e-mail alternativo do usuário. Possui a mesma função do atributo 'ldap.user.mail-attribute', atuando como contingência.	
ldap.user.password-attribute (opcional)	Nome do atributo que identifica a senha do usuário.	userPassword
ldap.user.enabled-attribute (opcional)	Nome do atributo que identifica se o usuário está ativo ou não.	userAccountControl
ldap.user.enabled-value (opcional)	Valor do atributo 'ldap.user.enabled-attribute' para determinar que o usuário está ativo.	544
ldap.user.disabled-value (opcional)	Valor do atributo 'ldap.user.enabled-attribute' para determinar que o usuário está inativo.	546
ldap.user.objectClass-attribute (opcional)	Nome do atributo que identifica a categoria do usuário.	objectClass
ldap.user.object-classes (opcional)	Valor do atributo 'ldap.user.objectClass-attribute' que determina as categorias de um usuário no LDAP.	top,person,organizationalPerson,user

Parâmetro	Descrição	Valor padrão
ldap.user.additional-valued-attributes (opcional)	Combinação atributo=valor utilizado na inserção de novos usuários na base LDAP.	
ldap.user.password-charset (opcional)	Padrão *encoding* para armazenamento da senha do usuário, e.g., UTF-8, UTF-16, UTF-32, etc.	

Exemplo:

admin.password=secret
admin.email=my@domain.com
store.type=ldap
strore.readonly=false
mail.host=smtp.gmail.com
mail.port=465
mail.tls=true
mail.username=hello@gmail.com
mail.password=123456
ldap.server.protocol=ldaps
ldap.server.host=ldap.domain.com
ldap.bind.dn=cn=admin,cn=Users,dc=domain,dc=com
ldap.bind.password=surprise
ldap.user.search-context-dn=ou=Accounts,dc=domain,dc=com
ldap.user.insert-pattern-dn=cn={username},ou=Accounts,dc=domain,dc=com
ldap.user.name-attribute=sAMAccountName
ldap.user.displayName-attribute=sn,givenName
ldap.user.mail-attribute=userPrincipalName,mail
ldap.user.alternateMail-attribute=personalMail
ldap.user.password-attribute=unicodePwd
ldap.user.objectClass-attribute=objectClass
ldap.user.enabled-attribute=userAccountControl
ldap.user.enabled-value=544
ldap.user.disabled-value=546
ldap.user.object-classes=top,person,organizationalPerson,user
ldap.user.additional-valued-attributes=scriptPath=logon.vbs
ldap.user.password-charset=UTF-16LE

APÊNDICE B – Arquivo de configuração do agente RASEA

O agente RASEA para a tecnologia *Seam Security* deve ser configurado a partir do arquivo de configuração, que deve estar disponível no *classpath* do servidor de aplicação. No JBoss AS, este arquivo deve estar na pasta ***<JBOSS_HOME>/server/default/conf/*** com o nome ***rasea-agent.properties***. O arquivo consiste em diversas entradas 'atributos × valor' separadas por quebras de linha. Os atributos são separados pelo sinal '=' dos seus respectivos valores.

Parâmetro	**Descrição**	**Valor padrão**
protocol (opcional)	Protocolo utilizado na conexão com o servidor RASEA: - **http**: conexão sem segurança; - **https**: conexão segura;	http
host	Nome ou IP do *host* do servidor RASEA.	
port (opcional)	Porta do servidor RASEA.	80 para o protocolo HTTP e 443 para o protocolo HTTPS
mock	Indica se a o agente deve se conectar com o servidor ou deve simular as informações geradas pelo serviço. Ideal para executar as aplicações parceiras em ambiente de desenvolvimento sem recursos de infra-estrutura: - **true**: modo de simulação de dados; - **false**: modo para ambiente de homologação e produção;	false

APÊNDICE C – Módulo de serviços do Servidor RASEA

Com o intuito de apresentar o módulo de serviços do servidor RASEA, seguem os WSDL dos serviços de controle de acesso e gerenciamento.

1. WSDL de controle de acesso: AccessControl.wsdl

```
<definitions name='AccessControlService' targetNamespace='http://rasea.org/ws/AccessControl'
xmlns='http://schemas.xmlsoap.org/wsdl/' xmlns:ns1='http://rasea.org/types'
xmlns:ns2='http://rasea.org/commons' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns:tns='http://rasea.org/ws/AccessControl' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
 <types>
  <xs:schema targetNamespace='http://rasea.org/ws/AccessControl' version='1.0'
xmlns:ns1='http://rasea.org/types' xmlns:ns2='http://rasea.org/commons'
xmlns:tns='http://rasea.org/ws/AccessControl' xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:import namespace='http://rasea.org/types'/>
   <xs:import namespace='http://rasea.org/commons'/>
   <xs:element name='RaseaServiceException' nillable='true' type='ns2:raseaServiceFault'/>
   <xs:element name='assignedRoles' type='tns:assignedRoles'/>
   <xs:element name='assignedRolesResponse' type='tns:assignedRolesResponse'/>
   <xs:element name='assignedUsers' type='tns:assignedUsers'/>
   <xs:element name='assignedUsersResponse' type='tns:assignedUsersResponse'/>
   <xs:element name='authenticate' type='tns:authenticate'/>
   <xs:element name='authenticateResponse' type='tns:authenticateResponse'/>
   <xs:element name='changePassword' type='tns:changePassword'/>
   <xs:element name='changePasswordResponse' type='tns:changePasswordResponse'/>
   <xs:element name='listRoles' type='tns:listRoles'/>
   <xs:element name='listRolesResponse' type='tns:listRolesResponse'/>
   <xs:element name='listUsers' type='tns:listUsers'/>
   <xs:element name='listUsersResponse' type='tns:listUsersResponse'/>
   <xs:element name='raseaServiceHeader' nillable='true' type='ns2:raseaServiceHeader'/>
   <xs:element name='requestConfirmationCode' type='tns:requestConfirmationCode'/>
   <xs:element name='requestConfirmationCodeResponse'
type='tns:requestConfirmationCodeResponse'/>
   <xs:element name='resetPassword' type='tns:resetPassword'/>
   <xs:element name='resetPasswordResponse' type='tns:resetPasswordResponse'/>
   <xs:element name='roleDetail' type='tns:roleDetail'/>
   <xs:element name='roleDetailResponse' type='tns:roleDetailResponse'/>
   <xs:element name='roleEnabled' type='tns:roleEnabled'/>
   <xs:element name='roleEnabledResponse' type='tns:roleEnabledResponse'/>
   <xs:element name='roleOperationsOnResource' type='tns:roleOperationsOnResource'/>
   <xs:element name='roleOperationsOnResourceResponse'
type='tns:roleOperationsOnResourceResponse'/>
   <xs:element name='rolePermissions' type='tns:rolePermissions'/>
   <xs:element name='rolePermissionsResponse' type='tns:rolePermissionsResponse'/>
   <xs:element name='unassignedRoles' type='tns:unassignedRoles'/>
   <xs:element name='unassignedRolesResponse' type='tns:unassignedRolesResponse'/>
   <xs:element name='userDetail' type='tns:userDetail'/>
   <xs:element name='userDetailResponse' type='tns:userDetailResponse'/>
   <xs:element name='userEnabled' type='tns:userEnabled'/>
   <xs:element name='userEnabledResponse' type='tns:userEnabledResponse'/>
   <xs:element name='userOperationsOnResource' type='tns:userOperationsOnResource'/>
   <xs:element name='userOperationsOnResourceResponse'
type='tns:userOperationsOnResourceResponse'/>
   <xs:element name='userPermissions' type='tns:userPermissions'/>
   <xs:element name='userPermissionsResponse' type='tns:userPermissionsResponse'/>
   <xs:complexType name='authenticate'>
    <xs:sequence>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
     <xs:element minOccurs='0' name='password' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='authenticateResponse'>
    <xs:sequence>
     <xs:element name='authenticated' type='xs:boolean'/>
    </xs:sequence>
```

```
   </xs:complexType>
   <xs:complexType name='rolePermissions'>
    <xs:sequence>
     <xs:element minOccurs='0' name='application' type='xs:string'/>
     <xs:element minOccurs='0' name='role' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='rolePermissionsResponse'>
    <xs:sequence>
     <xs:element maxOccurs='unbounded' minOccurs='0' name='permissions'
type='ns1:permission'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='userEnabled'>
    <xs:sequence>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='userEnabledResponse'>
    <xs:sequence>
     <xs:element name='enabled' type='xs:boolean'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='changePassword'>
    <xs:sequence>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
     <xs:element minOccurs='0' name='password' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='changePasswordResponse'>
    <xs:sequence>
     <xs:element name='changed' type='xs:boolean'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='listUsers'>
    <xs:sequence/>
   </xs:complexType>
   <xs:complexType name='listUsersResponse'>
    <xs:sequence>
     <xs:element maxOccurs='unbounded' minOccurs='0' name='users' type='ns1:user'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='listRoles'>
    <xs:sequence>
     <xs:element minOccurs='0' name='application' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='listRolesResponse'>
    <xs:sequence>
     <xs:element maxOccurs='unbounded' minOccurs='0' name='roles' type='ns1:role'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='userPermissions'>
    <xs:sequence>
     <xs:element minOccurs='0' name='application' type='xs:string'/>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='userPermissionsResponse'>
    <xs:sequence>
     <xs:element maxOccurs='unbounded' minOccurs='0' name='permissions'
type='ns1:permission'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='assignedRoles'>
    <xs:sequence>
     <xs:element minOccurs='0' name='application' type='xs:string'/>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='assignedRolesResponse'>
    <xs:sequence>
     <xs:element maxOccurs='unbounded' minOccurs='0' name='roles' type='ns1:role'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='assignedUsers'>
    <xs:sequence>
```

```
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='assignedUsersResponse'>
 <xs:sequence>
  <xs:element maxOccurs='unbounded' minOccurs='0' name='users' type='ns1:user'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='requestConfirmationCode'>
 <xs:sequence>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='requestConfirmationCodeResponse'>
 <xs:sequence>
  <xs:element name='requested' type='xs:boolean'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='resetPassword'>
 <xs:sequence>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
  <xs:element minOccurs='0' name='confirmationCode' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='resetPasswordResponse'>
 <xs:sequence>
  <xs:element name='reseted' type='xs:boolean'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleDetail'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleDetailResponse'>
 <xs:sequence>
  <xs:element minOccurs='0' name='role' type='ns1:role'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleEnabled'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleEnabledResponse'>
 <xs:sequence>
  <xs:element name='enabled' type='xs:boolean'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleOperationsOnResource'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
  <xs:element minOccurs='0' name='resource' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='roleOperationsOnResourceResponse'>
 <xs:sequence>
  <xs:element maxOccurs='unbounded' minOccurs='0' name='operations' type='ns1:operation'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='unassignedRoles'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='unassignedRolesResponse'>
 <xs:sequence>
  <xs:element maxOccurs='unbounded' minOccurs='0' name='roles' type='ns1:user'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='userDetail'>
 <xs:sequence>
```

```
      <xs:element minOccurs='0' name='username' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='userDetailResponse'>
     <xs:sequence>
      <xs:element minOccurs='0' name='user' type='ns1:user'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='userOperationsOnResource'>
     <xs:sequence>
      <xs:element minOccurs='0' name='application' type='xs:string'/>
      <xs:element minOccurs='0' name='username' type='xs:string'/>
      <xs:element minOccurs='0' name='resource' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='userOperationsOnResourceResponse'>
     <xs:sequence>
      <xs:element maxOccurs='unbounded' minOccurs='0' name='operations' type='ns1:operation'/>
     </xs:sequence>
    </xs:complexType>
   </xs:schema>
   <xs:schema targetNamespace='http://rasea.org/commons' version='1.0'
xmlns:xs='http://www.w3.org/2001/XMLSchema'>
    <xs:complexType name='raseaServiceFault'>
     <xs:sequence>
      <xs:element name='errorCode' type='xs:int'/>
      <xs:element minOccurs='0' name='source' type='xs:string'/>
      <xs:element minOccurs='0' name='additionalInfo' type='xs:anyType'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='raseaServiceHeader'>
     <xs:sequence>
      <xs:element minOccurs='0' name='username' type='xs:string'/>
      <xs:element minOccurs='0' name='password' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
   </xs:schema>
   <xs:schema targetNamespace='http://rasea.org/types' version='1.0'
xmlns:xs='http://www.w3.org/2001/XMLSchema'>
    <xs:complexType name='permission'>
     <xs:sequence>
      <xs:element minOccurs='0' name='resource' type='xs:string'/>
      <xs:element minOccurs='0' name='operation' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='user'>
     <xs:sequence>
      <xs:element minOccurs='0' name='username' type='xs:string'/>
      <xs:element minOccurs='0' name='displayName' type='xs:string'/>
      <xs:element minOccurs='0' name='email' type='xs:string'/>
      <xs:element minOccurs='0' name='alternateEmail' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='role'>
     <xs:sequence>
      <xs:element minOccurs='0' name='name' type='xs:string'/>
      <xs:element minOccurs='0' name='description' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='operation'>
     <xs:sequence>
      <xs:element minOccurs='0' name='name' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
   </xs:schema>
  </types>
  <message name='AccessControl_roleEnabledResponse'>
   <part element='tns:roleEnabledResponse' name='roleEnabledResponse'></part>
  </message>
  <message name='AccessControl_roleOperationsOnResource'>
   <part element='tns:roleOperationsOnResource' name='roleOperationsOnResource'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
  </message>
  <message name='RaseaServiceException'>
   <part element='tns:RaseaServiceException' name='RaseaServiceException'></part>
  </message>
  <message name='AccessControl_userEnabled'>
```

```
  <part element='tns:userEnabled' name='userEnabled'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_listRoles'>
  <part element='tns:listRoles' name='listRoles'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_userPermissions'>
  <part element='tns:userPermissions' name='userPermissions'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_userEnabledResponse'>
  <part element='tns:userEnabledResponse' name='userEnabledResponse'></part>
 </message>
 <message name='AccessControl_assignedRolesResponse'>
  <part element='tns:assignedRolesResponse' name='assignedRolesResponse'></part>
 </message>
 <message name='AccessControl_roleEnabled'>
  <part element='tns:roleEnabled' name='roleEnabled'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_assignedUsers'>
  <part element='tns:assignedUsers' name='assignedUsers'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_changePassword'>
  <part element='tns:changePassword' name='changePassword'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_userDetailResponse'>
  <part element='tns:userDetailResponse' name='userDetailResponse'></part>
 </message>
 <message name='AccessControl_unassignedRoles'>
  <part element='tns:unassignedRoles' name='unassignedRoles'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_assignedUsersResponse'>
  <part element='tns:assignedUsersResponse' name='assignedUsersResponse'></part>
 </message>
 <message name='AccessControl_assignedRoles'>
  <part element='tns:assignedRoles' name='assignedRoles'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_roleDetail'>
  <part element='tns:roleDetail' name='roleDetail'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_rolePermissions'>
  <part element='tns:rolePermissions' name='rolePermissions'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_roleOperationsOnResourceResponse'>
  <part element='tns:roleOperationsOnResourceResponse'
name='roleOperationsOnResourceResponse'></part>
 </message>
 <message name='AccessControl_userOperationsOnResourceResponse'>
  <part element='tns:userOperationsOnResourceResponse'
name='userOperationsOnResourceResponse'></part>
 </message>
 <message name='AccessControl_unassignedRolesResponse'>
  <part element='tns:unassignedRolesResponse' name='unassignedRolesResponse'></part>
 </message>
 <message name='AccessControl_resetPasswordResponse'>
  <part element='tns:resetPasswordResponse' name='resetPasswordResponse'></part>
 </message>
 <message name='AccessControl_userPermissionsResponse'>
  <part element='tns:userPermissionsResponse' name='userPermissionsResponse'></part>
 </message>
 <message name='AccessControl_resetPassword'>
  <part element='tns:resetPassword' name='resetPassword'></part>
 </message>
 <message name='AccessControl_authenticate'>
  <part element='tns:authenticate' name='authenticate'></part>
 </message>
 <message name='AccessControl_listUsersResponse'>
  <part element='tns:listUsersResponse' name='listUsersResponse'></part>
 </message>
```

```
 <message name='AccessControl_listUsers'>
  <part element='tns:listUsers' name='listUsers'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_changePasswordResponse'>
  <part element='tns:changePasswordResponse' name='changePasswordResponse'></part>
 </message>
 <message name='AccessControl_authenticateResponse'>
  <part element='tns:authenticateResponse' name='authenticateResponse'></part>
 </message>
 <message name='AccessControl_userDetail'>
  <part element='tns:userDetail' name='userDetail'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_listRolesResponse'>
  <part element='tns:listRolesResponse' name='listRolesResponse'></part>
 </message>
 <message name='AccessControl_roleDetailResponse'>
  <part element='tns:roleDetailResponse' name='roleDetailResponse'></part>
 </message>
 <message name='AccessControl_userOperationsOnResource'>
  <part element='tns:userOperationsOnResource' name='userOperationsOnResource'></part>
  <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
 </message>
 <message name='AccessControl_rolePermissionsResponse'>
  <part element='tns:rolePermissionsResponse' name='rolePermissionsResponse'></part>
 </message>
 <message name='AccessControl_requestConfirmationCode'>
  <part element='tns:requestConfirmationCode' name='requestConfirmationCode'></part>
 </message>
 <message name='AccessControl_requestConfirmationCodeResponse'>
  <part element='tns:requestConfirmationCodeResponse'
name='requestConfirmationCodeResponse'></part>
 </message>
 <portType name='AccessControl'>
  <operation name='assignedRoles' parameterOrder='assignedRoles raseaServiceHeader'>
   <input message='tns:AccessControl_assignedRoles'></input>
   <output message='tns:AccessControl_assignedRolesResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='assignedUsers' parameterOrder='assignedUsers raseaServiceHeader'>
   <input message='tns:AccessControl_assignedUsers'></input>
   <output message='tns:AccessControl_assignedUsersResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='authenticate' parameterOrder='authenticate'>
   <input message='tns:AccessControl_authenticate'></input>
   <output message='tns:AccessControl_authenticateResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='changePassword' parameterOrder='changePassword raseaServiceHeader'>
   <input message='tns:AccessControl_changePassword'></input>
   <output message='tns:AccessControl_changePasswordResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='listRoles' parameterOrder='listRoles raseaServiceHeader'>
   <input message='tns:AccessControl_listRoles'></input>
   <output message='tns:AccessControl_listRolesResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='listUsers' parameterOrder='listUsers raseaServiceHeader'>
   <input message='tns:AccessControl_listUsers'></input>
   <output message='tns:AccessControl_listUsersResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='requestConfirmationCode' parameterOrder='requestConfirmationCode'>
   <input message='tns:AccessControl_requestConfirmationCode'></input>
   <output message='tns:AccessControl_requestConfirmationCodeResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='resetPassword' parameterOrder='resetPassword'>
   <input message='tns:AccessControl_resetPassword'></input>
   <output message='tns:AccessControl_resetPasswordResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='roleDetail' parameterOrder='roleDetail raseaServiceHeader'>
   <input message='tns:AccessControl_roleDetail'></input>
```

```
    <output message='tns:AccessControl_roleDetailResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='roleEnabled' parameterOrder='roleEnabled raseaServiceHeader'>
    <input message='tns:AccessControl_roleEnabled'></input>
    <output message='tns:AccessControl_roleEnabledResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='roleOperationsOnResource' parameterOrder='roleOperationsOnResource
raseaServiceHeader'>
    <input message='tns:AccessControl_roleOperationsOnResource'></input>
    <output message='tns:AccessControl_roleOperationsOnResourceResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='rolePermissions' parameterOrder='rolePermissions raseaServiceHeader'>
    <input message='tns:AccessControl_rolePermissions'></input>
    <output message='tns:AccessControl_rolePermissionsResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='unassignedRoles' parameterOrder='unassignedRoles raseaServiceHeader'>
    <input message='tns:AccessControl_unassignedRoles'></input>
    <output message='tns:AccessControl_unassignedRolesResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='userDetail' parameterOrder='userDetail raseaServiceHeader'>
    <input message='tns:AccessControl_userDetail'></input>
    <output message='tns:AccessControl_userDetailResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='userEnabled' parameterOrder='userEnabled raseaServiceHeader'>
    <input message='tns:AccessControl_userEnabled'></input>
    <output message='tns:AccessControl_userEnabledResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='userOperationsOnResource' parameterOrder='userOperationsOnResource
raseaServiceHeader'>
    <input message='tns:AccessControl_userOperationsOnResource'></input>
    <output message='tns:AccessControl_userOperationsOnResourceResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
   <operation name='userPermissions' parameterOrder='userPermissions raseaServiceHeader'>
    <input message='tns:AccessControl_userPermissions'></input>
    <output message='tns:AccessControl_userPermissionsResponse'></output>
    <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
   </operation>
  </portType>
  <binding name='AccessControlBinding' type='tns:AccessControl'>
   <soap:binding style='document' transport='http://schemas.xmlsoap.org/soap/http'/>
   <operation name='assignedRoles'>
    <soap:operation soapAction=''/>
    <input>
     <soap:body parts='assignedRoles' use='literal'/>
     <soap:header message='tns:AccessControl_assignedRoles' part='raseaServiceHeader'
use='literal'></soap:header>
    </input>
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
   <operation name='assignedUsers'>
    <soap:operation soapAction=''/>
    <input>
     <soap:body parts='assignedUsers' use='literal'/>
     <soap:header message='tns:AccessControl_assignedUsers' part='raseaServiceHeader'
use='literal'></soap:header>
    </input>
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
   <operation name='authenticate'>
```

```
   <soap:operation soapAction=''/>
   <input>
    <soap:body use='literal'/>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='changePassword'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='changePassword' use='literal'/>
    <soap:header message='tns:AccessControl_changePassword' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='listRoles'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='listRoles' use='literal'/>
    <soap:header message='tns:AccessControl_listRoles' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='listUsers'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='listUsers' use='literal'/>
    <soap:header message='tns:AccessControl_listUsers' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='requestConfirmationCode'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body use='literal'/>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='resetPassword'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body use='literal'/>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
```

```
  <operation name='roleDetail'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='roleDetail' use='literal'/>
    <soap:header message='tns:AccessControl_roleDetail' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='roleEnabled'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='roleEnabled' use='literal'/>
    <soap:header message='tns:AccessControl_roleEnabled' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='roleOperationsOnResource'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='roleOperationsOnResource' use='literal'/>
    <soap:header message='tns:AccessControl_roleOperationsOnResource'
part='raseaServiceHeader' use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='rolePermissions'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='rolePermissions' use='literal'/>
    <soap:header message='tns:AccessControl_rolePermissions' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='unassignedRoles'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='unassignedRoles' use='literal'/>
    <soap:header message='tns:AccessControl_unassignedRoles' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='userDetail'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='userDetail' use='literal'/>
    <soap:header message='tns:AccessControl_userDetail' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
```

```
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
   <operation name='userEnabled'>
    <soap:operation soapAction=''/>
    <input>
     <soap:body parts='userEnabled' use='literal'/>
     <soap:header message='tns:AccessControl_userEnabled' part='raseaServiceHeader'
use='literal'></soap:header>
    </input>
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
   <operation name='userOperationsOnResource'>
    <soap:operation soapAction=''/>
    <input>
     <soap:body parts='userOperationsOnResource' use='literal'/>
     <soap:header message='tns:AccessControl_userOperationsOnResource'
part='raseaServiceHeader' use='literal'></soap:header>
    </input>
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
   <operation name='userPermissions'>
    <soap:operation soapAction=''/>
    <input>
     <soap:body parts='userPermissions' use='literal'/>
     <soap:header message='tns:AccessControl_userPermissions' part='raseaServiceHeader'
use='literal'></soap:header>
    </input>
    <output>
     <soap:body use='literal'/>
    </output>
    <fault name='RaseaServiceException'>
     <soap:fault name='RaseaServiceException' use='literal'/>
    </fault>
   </operation>
  </binding>
  <service name='AccessControlService'>
   <port binding='tns:AccessControlBinding' name='AccessControlPort'>
    <soap:address location='http://zycnote:8080/rasea/services/AccessControl'/>
   </port>
  </service>
</definitions>
```

2. WSDL de gerenciamento: Management.wsdl

```
<definitions name='ManagementService' targetNamespace='http://rasea.org/ws/Management'
xmlns='http://schemas.xmlsoap.org/wsdl/' xmlns:ns1='http://rasea.org/types'
xmlns:ns2='http://rasea.org/commons' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns:tns='http://rasea.org/ws/Management' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
 <types>
  <xs:schema targetNamespace='http://rasea.org/commons' version='1.0'
xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:complexType name='raseaServiceHeader'>
    <xs:sequence>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
     <xs:element minOccurs='0' name='password' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='raseaServiceFault'>
    <xs:sequence>
     <xs:element name='errorCode' type='xs:int'/>
     <xs:element minOccurs='0' name='source' type='xs:string'/>
     <xs:element minOccurs='0' name='additionalInfo' type='xs:anyType'/>
    </xs:sequence>
   </xs:complexType>
  </xs:schema>
  <xs:schema targetNamespace='http://rasea.org/ws/Management' version='1.0'
xmlns:ns1='http://rasea.org/types' xmlns:ns2='http://rasea.org/commons'
xmlns:tns='http://rasea.org/ws/Management' xmlns:xs='http://www.w3.org/2001/XMLSchema'>
   <xs:import namespace='http://rasea.org/types'/>
   <xs:import namespace='http://rasea.org/commons'/>
   <xs:element name='RaseaServiceException' nillable='true' type='ns2:raseaServiceFault'/>
   <xs:element name='addRole' type='tns:addRole'/>
   <xs:element name='addRoleResponse' type='tns:addRoleResponse'/>
   <xs:element name='addUser' type='tns:addUser'/>
   <xs:element name='addUserResponse' type='tns:addUserResponse'/>
   <xs:element name='assignUser' type='tns:assignUser'/>
   <xs:element name='assignUserResponse' type='tns:assignUserResponse'/>
   <xs:element name='deassignUser' type='tns:deassignUser'/>
   <xs:element name='deassignUserResponse' type='tns:deassignUserResponse'/>
   <xs:element name='deleteRole' type='tns:deleteRole'/>
   <xs:element name='deleteRoleResponse' type='tns:deleteRoleResponse'/>
   <xs:element name='deleteUser' type='tns:deleteUser'/>
   <xs:element name='deleteUserResponse' type='tns:deleteUserResponse'/>
   <xs:element name='disableRole' type='tns:disableRole'/>
   <xs:element name='disableRoleResponse' type='tns:disableRoleResponse'/>
   <xs:element name='disableUser' type='tns:disableUser'/>
   <xs:element name='disableUserResponse' type='tns:disableUserResponse'/>
   <xs:element name='enableRole' type='tns:enableRole'/>
   <xs:element name='enableRoleResponse' type='tns:enableRoleResponse'/>
   <xs:element name='enableUser' type='tns:enableUser'/>
   <xs:element name='enableUserResponse' type='tns:enableUserResponse'/>
   <xs:element name='grantPermission' type='tns:grantPermission'/>
   <xs:element name='grantPermissionResponse' type='tns:grantPermissionResponse'/>
   <xs:element name='raseaServiceHeader' nillable='true' type='ns2:raseaServiceHeader'/>
   <xs:element name='revokePermission' type='tns:revokePermission'/>
   <xs:element name='revokePermissionResponse' type='tns:revokePermissionResponse'/>
   <xs:complexType name='addUser'>
    <xs:sequence>
     <xs:element minOccurs='0' name='user' type='ns1:user'/>
     <xs:element minOccurs='0' name='password' type='xs:string'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='addUserResponse'>
    <xs:sequence/>
   </xs:complexType>
   <xs:complexType name='addRole'>
    <xs:sequence>
     <xs:element minOccurs='0' name='application' type='xs:string'/>
     <xs:element minOccurs='0' name='role' type='ns1:role'/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name='addRoleResponse'>
    <xs:sequence/>
   </xs:complexType>
   <xs:complexType name='deleteUser'>
    <xs:sequence>
     <xs:element minOccurs='0' name='username' type='xs:string'/>
```

```
 </xs:sequence>
</xs:complexType>
<xs:complexType name='deleteUserResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='enableUser'>
 <xs:sequence>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='enableUserResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='disableUser'>
 <xs:sequence>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='disableUserResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='deleteRole'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='deleteRoleResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='grantPermission'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='permission' type='ns1:permission'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
  <xs:element name='allowed' type='xs:boolean'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='grantPermissionResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='revokePermission'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='permission' type='ns1:permission'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='revokePermissionResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='assignUser'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='assignUserResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='deassignUser'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='username' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name='deassignUserResponse'>
 <xs:sequence/>
</xs:complexType>
<xs:complexType name='disableRole'>
 <xs:sequence>
  <xs:element minOccurs='0' name='application' type='xs:string'/>
  <xs:element minOccurs='0' name='role' type='xs:string'/>
 </xs:sequence>
</xs:complexType>
```

```
    <xs:complexType name='disableRoleResponse'>
     <xs:sequence/>
    </xs:complexType>
    <xs:complexType name='enableRole'>
     <xs:sequence>
      <xs:element minOccurs='0' name='application' type='xs:string'/>
      <xs:element minOccurs='0' name='role' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='enableRoleResponse'>
     <xs:sequence/>
    </xs:complexType>
   </xs:schema>
   <xs:schema targetNamespace='http://rasea.org/types' version='1.0'
xmlns:xs='http://www.w3.org/2001/XMLSchema'>
    <xs:complexType name='user'>
     <xs:sequence>
      <xs:element minOccurs='0' name='username' type='xs:string'/>
      <xs:element minOccurs='0' name='displayName' type='xs:string'/>
      <xs:element minOccurs='0' name='email' type='xs:string'/>
      <xs:element minOccurs='0' name='alternateEmail' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='role'>
     <xs:sequence>
      <xs:element minOccurs='0' name='name' type='xs:string'/>
      <xs:element minOccurs='0' name='description' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
    <xs:complexType name='permission'>
     <xs:sequence>
      <xs:element minOccurs='0' name='resource' type='xs:string'/>
      <xs:element minOccurs='0' name='operation' type='xs:string'/>
     </xs:sequence>
    </xs:complexType>
   </xs:schema>
  </types>
  <message name='Management_disableUserResponse'>
   <part element='tns:disableUserResponse' name='disableUserResponse'></part>
  </message>
  <message name='Management_deassignUserResponse'>
   <part element='tns:deassignUserResponse' name='deassignUserResponse'></part>
  </message>
  <message name='RaseaServiceException'>
   <part element='tns:RaseaServiceException' name='RaseaServiceException'></part>
  </message>
  <message name='Management_addRole'>
   <part element='tns:addRole' name='addRole'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
  </message>
  <message name='Management_enableUser'>
   <part element='tns:enableUser' name='enableUser'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
  </message>
  <message name='Management_addUserResponse'>
   <part element='tns:addUserResponse' name='addUserResponse'></part>
  </message>
  <message name='Management_disableUser'>
   <part element='tns:disableUser' name='disableUser'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
  </message>
  <message name='Management_grantPermission'>
   <part element='tns:grantPermission' name='grantPermission'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
  </message>
  <message name='Management_deleteUserResponse'>
   <part element='tns:deleteUserResponse' name='deleteUserResponse'></part>
  </message>
  <message name='Management_grantPermissionResponse'>
   <part element='tns:grantPermissionResponse' name='grantPermissionResponse'></part>
  </message>
  <message name='Management_enableRoleResponse'>
   <part element='tns:enableRoleResponse' name='enableRoleResponse'></part>
  </message>
  <message name='Management_deleteRole'>
   <part element='tns:deleteRole' name='deleteRole'></part>
   <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
```

```
</message>
<message name='Management_revokePermissionResponse'>
 <part element='tns:revokePermissionResponse' name='revokePermissionResponse'></part>
</message>
<message name='Management_revokePermission'>
 <part element='tns:revokePermission' name='revokePermission'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_enableUserResponse'>
 <part element='tns:enableUserResponse' name='enableUserResponse'></part>
</message>
<message name='Management_deleteUser'>
 <part element='tns:deleteUser' name='deleteUser'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_addUser'>
 <part element='tns:addUser' name='addUser'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_disableRoleResponse'>
 <part element='tns:disableRoleResponse' name='disableRoleResponse'></part>
</message>
<message name='Management_deassignUser'>
 <part element='tns:deassignUser' name='deassignUser'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_assignUserResponse'>
 <part element='tns:assignUserResponse' name='assignUserResponse'></part>
</message>
<message name='Management_addRoleResponse'>
 <part element='tns:addRoleResponse' name='addRoleResponse'></part>
</message>
<message name='Management_enableRole'>
 <part element='tns:enableRole' name='enableRole'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_deleteRoleResponse'>
 <part element='tns:deleteRoleResponse' name='deleteRoleResponse'></part>
</message>
<message name='Management_disableRole'>
 <part element='tns:disableRole' name='disableRole'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<message name='Management_assignUser'>
 <part element='tns:assignUser' name='assignUser'></part>
 <part element='tns:raseaServiceHeader' name='raseaServiceHeader'></part>
</message>
<portType name='Management'>
 <operation name='addRole' parameterOrder='addRole raseaServiceHeader'>
  <input message='tns:Management_addRole'></input>
  <output message='tns:Management_addRoleResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
 </operation>
 <operation name='addUser' parameterOrder='addUser raseaServiceHeader'>
  <input message='tns:Management_addUser'></input>
  <output message='tns:Management_addUserResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
 </operation>
 <operation name='assignUser' parameterOrder='assignUser raseaServiceHeader'>
  <input message='tns:Management_assignUser'></input>
  <output message='tns:Management_assignUserResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
 </operation>
 <operation name='deassignUser' parameterOrder='deassignUser raseaServiceHeader'>
  <input message='tns:Management_deassignUser'></input>
  <output message='tns:Management_deassignUserResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
 </operation>
 <operation name='deleteRole' parameterOrder='deleteRole raseaServiceHeader'>
  <input message='tns:Management_deleteRole'></input>
  <output message='tns:Management_deleteRoleResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
 </operation>
 <operation name='deleteUser' parameterOrder='deleteUser raseaServiceHeader'>
  <input message='tns:Management_deleteUser'></input>
  <output message='tns:Management_deleteUserResponse'></output>
  <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
```

```
  </operation>
  <operation name='disableRole' parameterOrder='disableRole raseaServiceHeader'>
   <input message='tns:Management_disableRole'></input>
   <output message='tns:Management_disableRoleResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='disableUser' parameterOrder='disableUser raseaServiceHeader'>
   <input message='tns:Management_disableUser'></input>
   <output message='tns:Management_disableUserResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='enableRole' parameterOrder='enableRole raseaServiceHeader'>
   <input message='tns:Management_enableRole'></input>
   <output message='tns:Management_enableRoleResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='enableUser' parameterOrder='enableUser raseaServiceHeader'>
   <input message='tns:Management_enableUser'></input>
   <output message='tns:Management_enableUserResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='grantPermission' parameterOrder='grantPermission raseaServiceHeader'>
   <input message='tns:Management_grantPermission'></input>
   <output message='tns:Management_grantPermissionResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
  <operation name='revokePermission' parameterOrder='revokePermission raseaServiceHeader'>
   <input message='tns:Management_revokePermission'></input>
   <output message='tns:Management_revokePermissionResponse'></output>
   <fault message='tns:RaseaServiceException' name='RaseaServiceException'></fault>
  </operation>
 </portType>
 <binding name='ManagementBinding' type='tns:Management'>
  <soap:binding style='document' transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='addRole'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='addRole' use='literal'/>
    <soap:header message='tns:Management_addRole' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='addUser'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='addUser' use='literal'/>
    <soap:header message='tns:Management_addUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='assignUser'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='assignUser' use='literal'/>
    <soap:header message='tns:Management_assignUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='deassignUser'>
```

```
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='deassignUser' use='literal'/>
    <soap:header message='tns:Management_deassignUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='deleteRole'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='deleteRole' use='literal'/>
    <soap:header message='tns:Management_deleteRole' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='deleteUser'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='deleteUser' use='literal'/>
    <soap:header message='tns:Management_deleteUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='disableRole'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='disableRole' use='literal'/>
    <soap:header message='tns:Management_disableRole' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='disableUser'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='disableUser' use='literal'/>
    <soap:header message='tns:Management_disableUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='enableRole'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='enableRole' use='literal'/>
    <soap:header message='tns:Management_enableRole' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
```

```
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='enableUser'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='enableUser' use='literal'/>
    <soap:header message='tns:Management_enableUser' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='grantPermission'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='grantPermission' use='literal'/>
    <soap:header message='tns:Management_grantPermission' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
  <operation name='revokePermission'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body parts='revokePermission' use='literal'/>
    <soap:header message='tns:Management_revokePermission' part='raseaServiceHeader'
use='literal'></soap:header>
   </input>
   <output>
    <soap:body use='literal'/>
   </output>
   <fault name='RaseaServiceException'>
    <soap:fault name='RaseaServiceException' use='literal'/>
   </fault>
  </operation>
 </binding>
 <service name='ManagementService'>
  <port binding='tns:ManagementBinding' name='ManagementPort'>
   <soap:address location='http://zycnote:8080/rasea/services/Management'/>
  </port>
 </service>
</definitions>
```

APÊNDICE D – Antigo módulo administrativo do servidor RASEA

Com o intuito de apresentar o antigo módulo administrativo do RASEA, seguem as imagens da interface gráfica da aplicação implantada na Avansys Tecnologia. Em julho de 2009, a interface gráfica do módulo administrativo foi completamente reformulada (APÊNDICE E – Atual módulo administrativo do servidor RASEA).

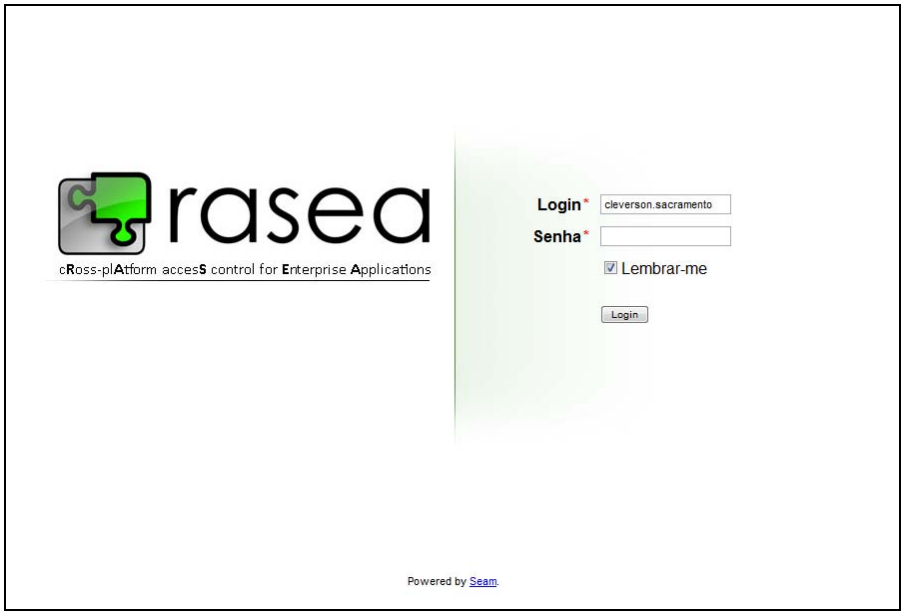

Imagem 1 – Tela de autenticação do sistema módulo administrativo do RASEA.

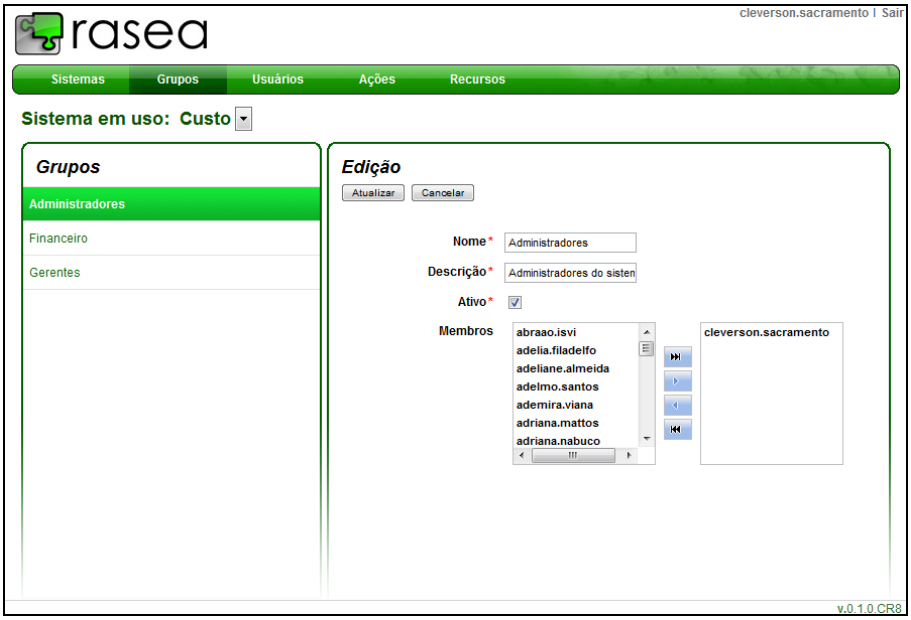

Imagem 2 – Gerenciamento dos papéis de um determinado sistema. Neste caso, a aplicação em uso é o sistema para controle de custos dos projetos da Avansys.

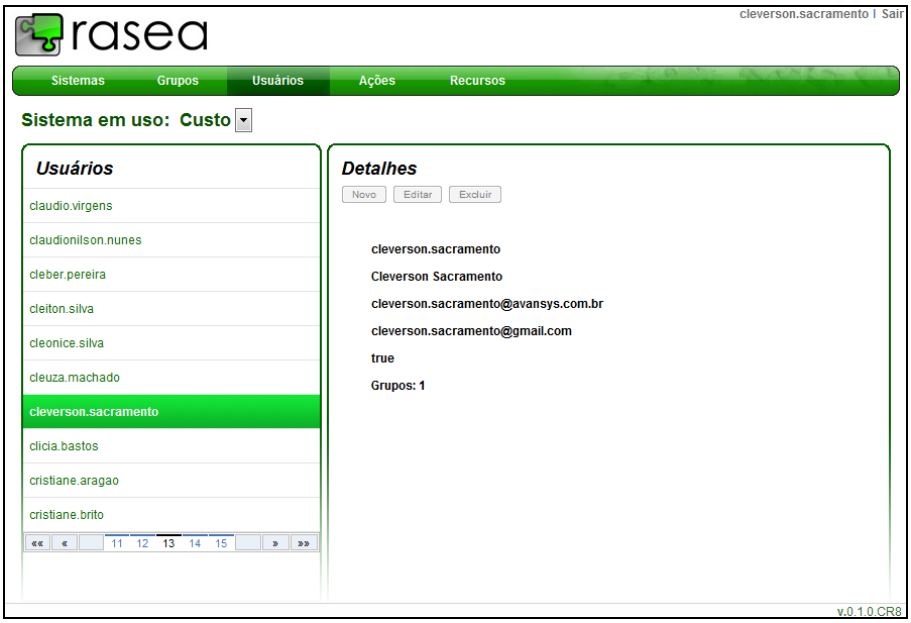

Imagem 3 – Listagem de todos os usuários da empresa, mostrando a integração do RASEA com a base unificada de usuários da organização.

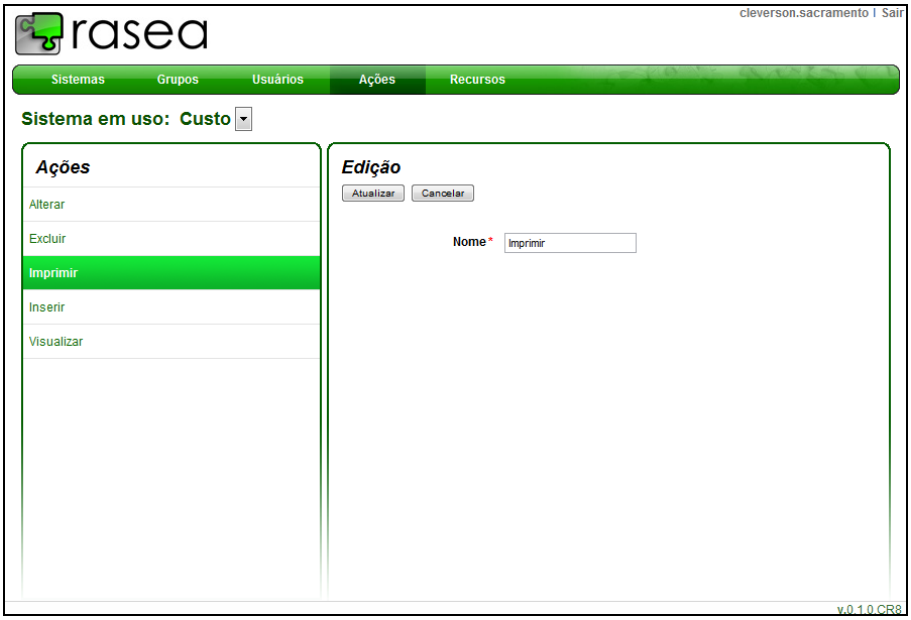

Imagem 4 – Gerenciamento das operações possíveis de serem executadas no sistema de custos.

Imagem 5 – Gerenciamento dos recursos do sistema de custos e associação das possíveis operações ao recurso (permissões).

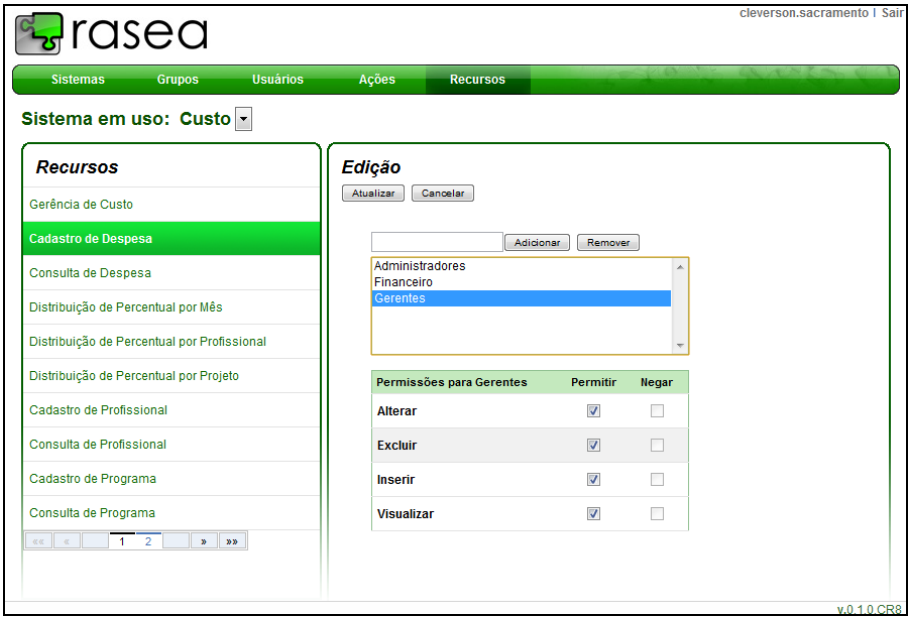

Imagem 6 – Concessão de permissão a um determinado papel do sistema de custos, autorizando que uma operação seja executada sob um recurso do sistema por um determinado papel.

APÊNDICE E – Atual módulo administrativo do servidor RASEA

O RASEA é utilizado na Avansys Tecnologia para controlar o acesso das aplicações internas da organização e na linha de produção da sua Fábrica de *Software*. A interface gráfica do RASEA foi completamente reformulada (APÊNDICE D – Antigo módulo administrativo do servidor RASEA) para atender as solicitações dos usuários, almejando a facilitação e simplificação no gerenciamento de permissões.

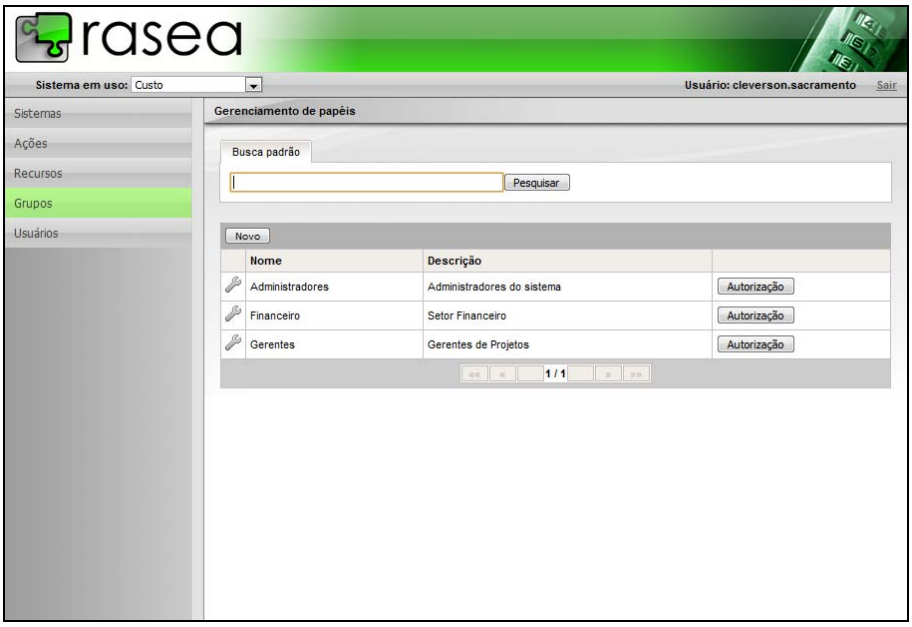

Imagem 1 – Foram incluídos filtros nas listagens para facilitar a busca, conforme solicitações dos usuários.

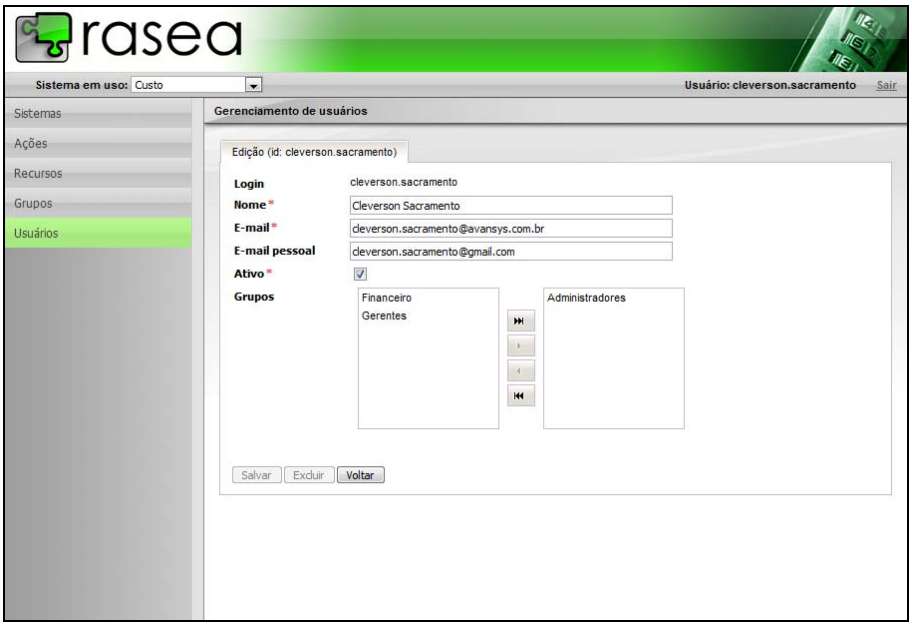

Imagem 2 – Cada listagem antecede a tela de cadastro ou alteração de registros.

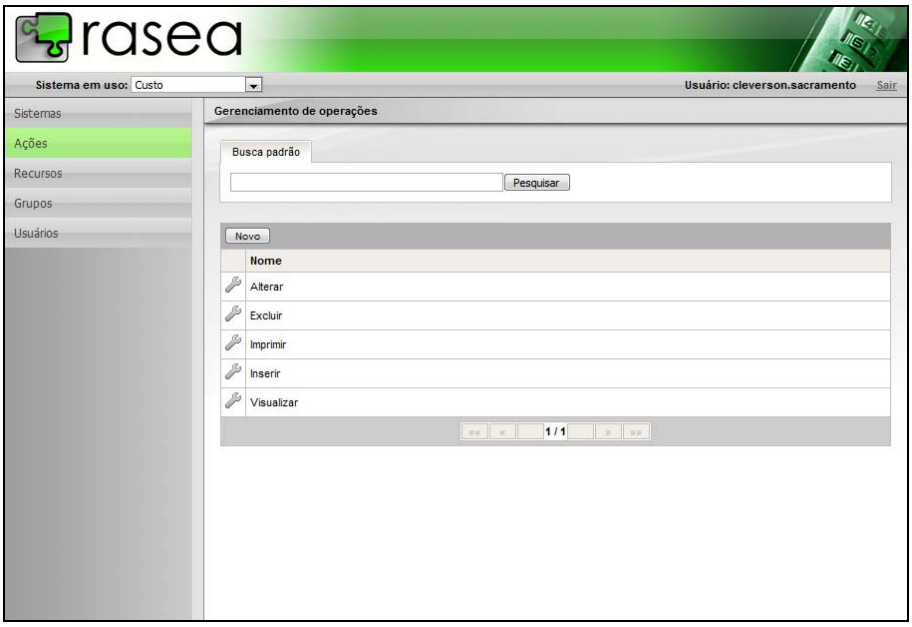

Imagem 3 – Listagem de todas as ações associadas ao sistema interno da Avansys para gerenciamento de custos de projetos.

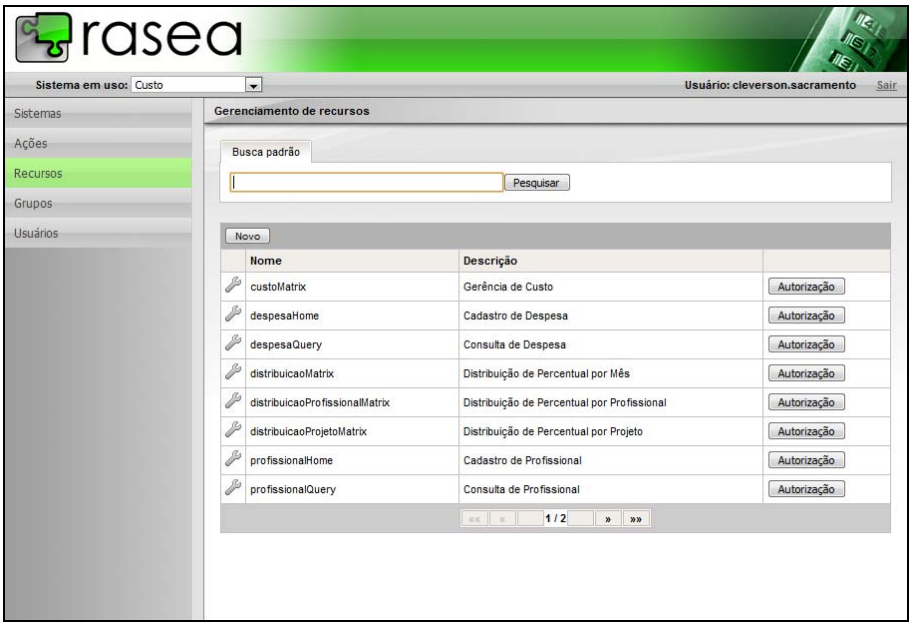

Imagem 4 – Listagem de todos os recursos do sistema de custos.

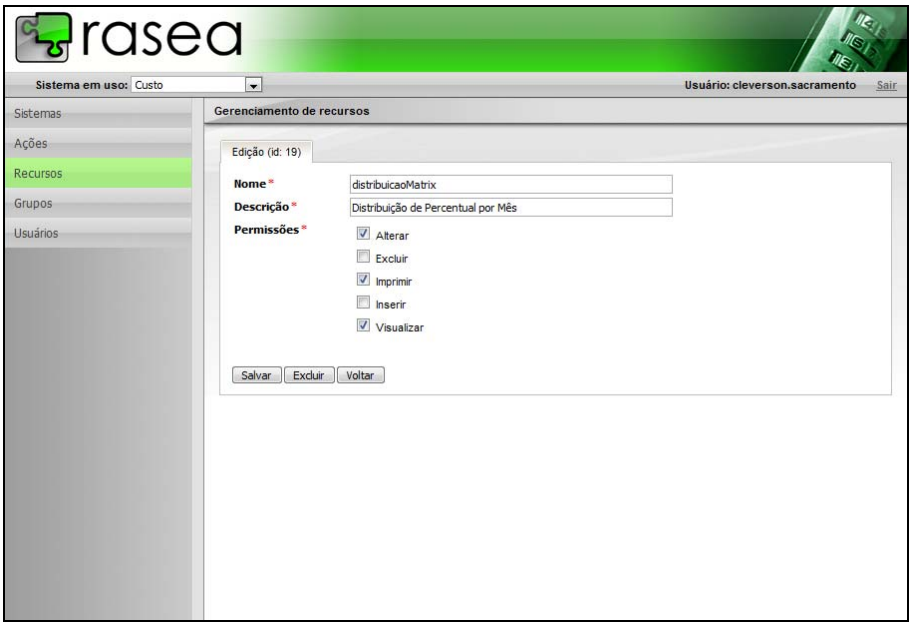

Imagem 5 – Alteração dos dados de um recurso e associação das possíveis operações, também conhecida como permissões.

Imagem 6 – Concessão das permissões de um determinado recurso para os papéis do sistema. O *layout* desta funcionalidade foi completamente reprojetado para atender às solicitações dos usuários, almejando a facilitação e simplificação no gerenciamento de permissões.

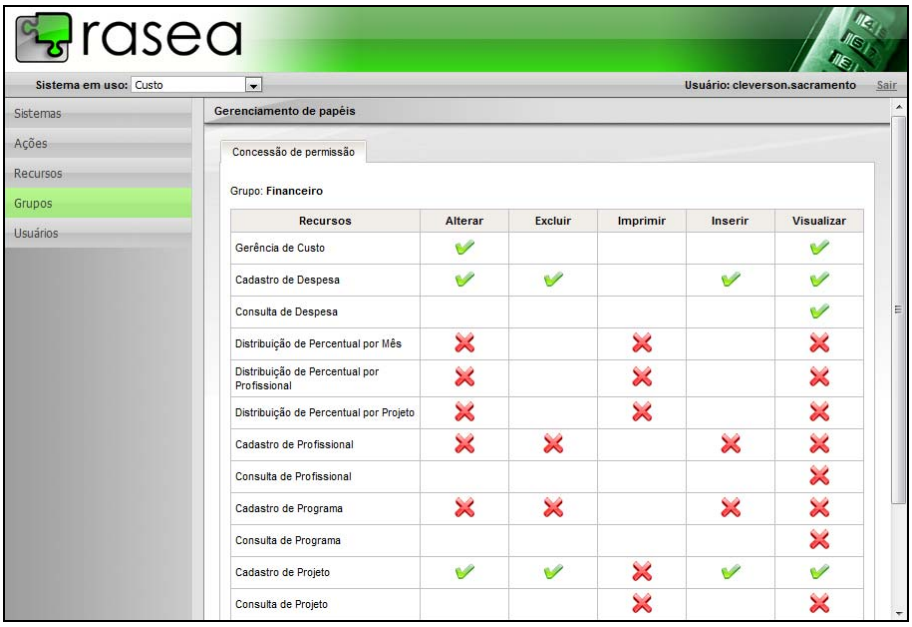

Imagem 7 – Mesma funcionalidade da tela anterior, entretanto esta tela possibilita o gerenciamento das permissões concedidas sob a perspectiva de um determinado papel, neste caso o papel 'financeiro'.

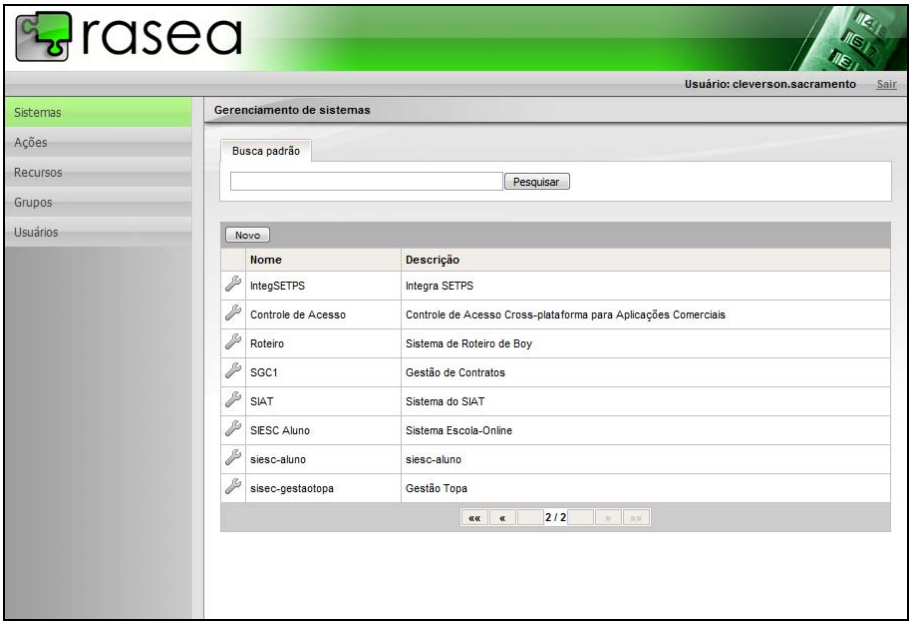

Imagem 8 – Listagem dos sistemas cadastrados na Avansys que possuem o seu controle de acesso gerenciado pelo RASEA.

APÊNDICE F – *Script* de migração do S4A para o RASEA

Segue o *script* utilizado para a migração dos dados do S4A para a estrutura do RASEA, seguindo o Modelo de Referência do RBAC.

```
CREATE PROCEDURE MIGRACAO_S4A_RASEA (ID_SISTEMA NUMBER) AS
VAR
    SEQ_APPLICATION  NUMBER;
BEGIN
    SELECT SEG.HIBERNATE_SEQUENCE.NEXTVAL
      INTO SEQ_APPLICATION
      FROM DUAL;

    INSERT INTO SEG.R_APPLICATION  (ID, NAME, DESCRIPTION)
    SELECT SEQ_APPLICATION, SIGLA, NOME
      FROM SEG.SISTEMA
     WHERE ID = ID_SISTEMA;

    INSERT INTO SEG.R_RESOURCE (ID, DESCRIPTION, NAME, APPLICATION_ID)
    SELECT SEG.HIBERNATE_SEQUENCE.NEXTVAL, NOME, URI, SEQ_APPLICATION
      FROM SEG.TELA
     WHERE SISTEMA_ID = ID_SISTEMA;

    INSERT INTO SEG.R_OPERATION (ID, NAME, APPLICATION_ID)
    SELECT SEG.HIBERNATE_SEQUENCE.NEXTVAL, NOME, SEQ_APPLICATION
      FROM SEG.ACAO
     WHERE ID IN (SELECT ACAO_ID FROM SEG.FUNCIONALIDADE WHERE TELA_ID IN (SELECT ID FROM
            SEG.TELA WHERE SISTEMA_ID = ID_SISTEMA));

    INSERT INTO SEG.R_PERMISSION (OPERATION_ID, RESOURCE_ID)
    SELECT O.ID, R.ID
      FROM SEG.FUNCIONALIDADE F, SEG.ACAO A, SEG.TELA T, SEG.R_RESOURCE R, SEG.R_OPERATION O
     WHERE F.ACAO_ID = A.ID
       AND F.TELA_ID = T.ID
       AND T.URI = R.NAME
       AND A.NOME = O.NAME
       AND T.SISTEMA_ID = ID_SISTEMA
       AND R.APPLICATION_ID = SEQ_APPLICATION
       AND O.APPLICATION_ID = SEQ_APPLICATION;

    INSERT INTO SEG.R_USER (USERNAME, ALTERNATE_EMAIL, DISPLAYNAME, EMAIL, ENABLED, PASSWORD)
    SELECT A.ID, A.EMAIL, A.NOME, A.EMAIL, 1, SENHA
      FROM SEG.ATOR A, SEG.USUARIO U
     WHERE A.ID = U.ID
       AND A.ID NOT IN (SELECT USERNAME FROM SEG.R_USER);

    INSERT INTO SEG.R_ROLE (ID, DESCRIPTION, ENABLED, NAME, APPLICATION_ID)
    SELECT SEG.HIBERNATE_SEQUENCE.NEXTVAL, NOME, 1, ID, SEQ_APPLICATION
      FROM SEG.ATOR
     WHERE ID IN (SELECT ID
                    FROM SEG.GRUPO G
                   WHERE ID IN (SELECT ATOR_ID
                                  FROM SEG.PERMISSAO
                                 WHERE TELA_ID IN (SELECT ID
                                                     FROM SEG.TELA
                                                    WHERE SISTEMA_ID = ID_SISTEMA)));

    INSERT INTO SEG.R_USER_ASSIGNMENT (USERNAME, ROLE_ID)
    SELECT M.USUARIO_ID ,R.ID
      FROM SEG.MEMBRO M, SEG.R_ROLE R
     WHERE M.GRUPO_ID = R.NAME
       AND R.APPLICATION_ID = SEQ_APPLICATION;

    INSERT INTO SEG.R_PERMISSION_ASSIGNMENT (ALLOWED, ROLE_ID, OPERATION_ID, RESOURCE_ID)
    SELECT PERMITIR, RO.ID, O.ID, RE.ID
      FROM SEG.PERMISSAO P,
           SEG.ACAO A, SEG.TELA T, SEG.GRUPO G,
           SEG.R_OPERATION O, SEG.R_RESOURCE RE, SEG.R_ROLE RO
```

```
     WHERE P.ACAO_ID = A.ID
       AND P.ATOR_ID = G.ID
       AND P.TELA_ID = T.ID
       AND A.NOME = O.NAME
       AND T.URI = RE.NAME
       AND G.ID = RO.NAME
       AND T.SISTEMA_ID = ID_SISTEMA
       AND RO.APPLICATION_ID = SEQ_APPLICATION
       AND RE.APPLICATION_ID = SEQ_APPLICATION
       AND O.APPLICATION_ID = SEQ_APPLICATION;



    INSERT INTO SEG.R_ROLE (ID, DESCRIPTION, ENABLED, NAME, APPLICATION_ID)
    SELECT SEG.HIBERNATE_SEQUENCE.NEXTVAL, 'S4A_'||ATOR_ID, 1, 'S4A_'||ATOR_ID,
            SEQ_APPLICATION
    FROM (SELECT DISTINCT ATOR_ID
      FROM SEG.PERMISSAO
     WHERE ATOR_ID NOT IN (SELECT ID FROM SEG.GRUPO)
       AND TELA_ID IN (SELECT ID
                         FROM SEG.TELA
                        WHERE SISTEMA_ID = ID_SISTEMA));

    INSERT INTO SEG.R_USER_ASSIGNMENT (USERNAME, ROLE_ID)
    SELECT DISTINCT 'S4A_'||P.ATOR_ID, RO.ID
      FROM SEG.PERMISSAO P, SEG.R_ROLE RO
     WHERE RO.NAME = 'S4A_'||P.ATOR_ID
       AND P.ATOR_ID NOT IN (SELECT ID FROM SEG.GRUPO)
       AND P.TELA_ID IN (SELECT ID
                         FROM SEG.TELA
                        WHERE SISTEMA_ID = ID_SISTEMA)
       AND RO.APPLICATION_ID = SEQ_APPLICATION;

    INSERT INTO SEG.R_PERMISSION_ASSIGNMENT (ALLOWED, ROLE_ID, OPERATION_ID, RESOURCE_ID)
    SELECT PERMITIR, RO.ID, O.ID, RE.ID
      FROM SEG.PERMISSAO P,
           SEG.ACAO A, SEG.TELA T, SEG.GRUPO G,
           SEG.R_OPERATION O, SEG.R_RESOURCE RE, SEG.R_ROLE RO
     WHERE P.ACAO_ID = A.ID
       AND P.ATOR_ID = G.ID(+)
       AND G.ID IS NULL
       AND P.TELA_ID = T.ID
       AND A.NOME = O.NAME
       AND T.URI = RE.NAME
       AND 'S4A_'||P.ATOR_ID = RO.NAME
       AND T.SISTEMA_ID = ID_SISTEMA
       AND RO.APPLICATION_ID = SEQ_APPLICATION
       AND RE.APPLICATION_ID = SEQ_APPLICATION
       AND O.APPLICATION_ID = SEQ_APPLICATION;
END;
```

APÊNDICE G – Configuração do agente *Seam Security* em uma aplicação parceira

Segue a configuração do *JBoss Seam Framework* (components.xml) para o Sistema de Custos da Avansys Tecnologia, utilizando o agente para a tecnologia *Seam Security*. O Sistema de Custos faz referência à biblioteca **rasea-seam-agent.jar**, que é ativada a partir da configuração do arquivo **components.xml**. Destaque para a simplicidade na configuração.

```
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components" (...)
      xmlns:rasea="http://rasea.org/agent/seam"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="(...)">

      <!-- Configurações do Seam sem relação com o agente RASEA -->
      <core:init debug="false" jndi-pattern="custo/#{name}/local" />
      <core:manager concurrent-request-timeout="20000" conversation-timeout="120000"
                    conversation-id-parameter="cid" />
      <international:locale-selector cookie-enabled="true" />
      <web:context-filter url-pattern="/services/*" />

      <mail:mail-session session-jndi-name="java:/Mail" />
      <document:document-store use-extensions="true" />
      <persistence:managed-persistence-context name="entityManager" auto-create="true"
                    persistence-unit-jndi-name="java:/custoEntityManagerFactory" />

      <event type="org.jboss.seam.security.notLoggedIn">
            <action execute="#{redirect.captureCurrentView}" />
      </event>

      <event type="org.jboss.seam.security.loginSuccessful">
            <action execute="#{redirect.returnToCapturedView}" />
      </event>

      <!-- Configurações do agente Seam Security com suporte ao SSO -->
      <rasea:partner-application app-name="custo" />
      <rasea:single-sign-on-identity />
</components>
```