UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Luis Marco Cáceres Alvarez

Modelo de Segurança Multilateral e RBAC em um Ambiente de Serviço no Contexto de Gerenciamento de Contabilidade TINA

Tese submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Doutor em Ciência da Computação.

Prof. Dr. Carlos Becker Westphall
Orientador

Profa. Dra. Carla Merkle Westphall
Co-orientadora

Florianópolis, Agosto de 2004.

Modelo de Segurança Multilateral e RBAC em um Ambiente de Serviço no Contexto de Gerenciamento de Contabilidade TINA

Luis Marco Cáceres Alvarez

Tese submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Doutor em Ciência da Computação, Área de Concentração *Sistemas de Computação* e aprovada pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Raul Sidnei Wazlawick, Dr.
Coordenador do Curso

Banca Examinadora

_____

Prof. Carlos Becker Westphall, Dr.
Orientador

_____

Prof. Bruno Richard Schulze, Dr.

_____

Prof. Lisandro Zambenedetti Granville, Dr.

_____

Prof. Jorge Muniz Barreto, Dr.

_____

Prof. Vitório Bruno Mazzola, Dr.

Dedicatória

Por um descanso eterno:

“Para mis recordados

hermanos, Daniel y Raquel

Q.E.P.D.”

Por la vida y amor:

“Para mi hijo Marco André”

Agradecimentos

Á Deus Todo Poderoso, pela minha vida e tudo que conquistei, meu Muito Obrigado.

Gostaria de agradecer aos professores integrantes da banca examinadora pela apreciação do presente trabalho.

Em especial, agradecimento ao meu orientador, Dr. Carlos Becker Westphall pela sua confiança, apoio, dedicação a este trabalho, às suas atividades acadêmicas e pela sua maravilhosa pessoa.

Meus agradecimentos a minha co-orientadora, Dra. Carla Merkle Westphall pela sua colaboração e apoio neste trabalho.

Agradeço à vida e meus seres queridos, Marco André e Lia.

A meus colegas, amigos Wagner, Sandro, Marcelo e Ivanise por sua grande amizade e colaboração, muito obrigado.

A meus alunos que compõe a equipe do CTI&CEAD Francisco, Rafael, Andressa e Vanderlei pela sua força, apoio e colaboração, muito obrigado.

Ao UNICS por todo o apoio prestado e ser professor, muito obrigado.

Agradeço a minha irmã Luisa, fonte infinita de amor carinho e compreensão.

Finalmente, a este maravilhoso país, meu muito obrigado BRASIL.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

3Pty : Third Party

5W : What, hoW, When, Who, Where

ABR : Available Bit Rate

ACA : Access Control Agent

AccMgmtCtxt : Accounting Management Context

AS : Agent Security

API : Application Program Interface

asUAP : access session User Application

ATM : Asynchronous Transfer Mode

ATMF : ATM – Forum

Bkr : Broker

CADP : CAESAR/ALDEBARAN Development Package

CC : Connection Coordinator

CCITT : Consultive Committee for International Telegraph and Telephone

CCS : Calculus of Communicating Systems

CIM : Common Information Model

CO : Computational Objects

ConS : Connectivity Service

CORBA : Common Object Request Broker Architecture

CP : Connectivity Provider

CPr : Content Provider

CPE : Customer Premise Equipment

CSLN : Client – Server Layer Network

CSM : Communication Session Manager

DAVIC : Digital Audio – Visual Council

DDoS : Distributed Denial-of-Service

DPE : Distributed Processing Environment

ESTELLE : Extended Finite – State Machine Language

FCAPS : Failure, Configuration, Accounting, Performance and Security

GRM : General Relationship Model

GSEP : Generic Session End Point

GSM: : Global System of Mobile Communications
IA : Initial Agent
IDL : Interface Description Language
IETF : Internet Engineering Task Force
IN : Intelligent Networks
IP : Internet Protocol
IRTF : Internet Research Task Force
ISDN : Integrated Service Digital Network
ISO : International Standard Organization
IT : Information Technology
ITU : International Telecommunications Union
ITU-T : ITU – Telecommunications Sector
Ktn : Kernel Transportation Network
LOTOS : Language of Temporal Ordering Specification
LNC : Layer Network Coordinator
LNFed : Layer Network Federation
LRG : Laboratório de Redes e Gerência
LTS : Labelled Transition System
MAC : Message Authentication Code
NCCE : Native Computing and Communications Environment
NEC : Network Flow Connection
NFEP : Network Flow End Point
NIST : National Institute of Standards and Technology
NM-Forum : Network Management Forum
NO : Network Operator
ODL : Object Description Language
ODMA : Open Distributed Management Architecture
ODP : Open Distributed Processing
OMG : Object Management Group
OMT : Object Modeling Technique
OSI : Open System Interconnections
OSIMIS : OSI Management Information Service

PA : Provider Agent

PC : Personal Computer

PDA : Portable Digital Assistance

PeerA : Peer Agent

PeerUSM : Peer Usage Session Manager

PINT : PSTN and Internet Internetworking

PKI : Public Key Infrastructure

PSTN : Public Switched Telephone Network

PTO : Public Telecommunication Operators

QoP : Quality of Protection

QoS : Quality of Service

RBAC : Role – Based Access Control

RBACMgmtCtxt : Role-Based Access Control Management Context

RCM : Resource Configuration Management

Ret : Retailer Reference Point

RMI : Remote Method Invocation

RM-ODP : Reference Model for Open Distributed Processing

RSA : Revest, Shamir and Adleman

RtR : Retailer to Retailer

SA : Security Agent

SAE : SubscriberAssigment Entity

SAG : Subscription Assigment Group

SB : Service Broker

SC : Session Component

SecMgmtCtxt : Security Management Context

SDL : Specification and Description Language

SFC : Stream Flow Connection

SF : Service Factory

SFEP : Stream Flow End Point

SIM : Security Information Management

SLA : Service Level Agremment

SLCM : Service Life Cycle Management

SR : Session Relation
SSGCT : Sistema de Segurança de Gerenciamento de Contabilidade TINA
SSL : Socket Service Layer
SSM : Service Session Manager
ssUAP : service session User Application
ST : Service Transaction
STR : Service Transaction Record
SUB : Subscription Management Component
Tcon : Terminal connection
TCP : Transmission Control Protocol
TCSEC : Trusted Computer System Evaluation Criteria
TCSM : Terminal Communication Session Manager
TDF : Técnicas de Descrição Formal
TINA : Telecommunication Information Networking Architecture
TINA-C : TINA Consortium
TLA : Terminal Layer Adapter
TMF : Telemanagement Forum
TMN : Telecommunication Management Network
TMS : Transaction Management Service
UA : User Agent
UAP : User Application
UML : Unified Modeling Language
UNICS : Centro Universitário Diocesano do Sudoeste do Paraná
UPT : Universal Personnal Telecommunications
USM : User Service Session Manager
VoD : Video on Demand
VoIP : Voice over IP
VPN : Virtual Private Network
WBEM : Web-Based Enterprise Management
WWW : World Wide Web

RESUMO

Na área das telecomunicações, a crescente evolução e o constante desenvolvimento de novas tecnologias, aliado a fatores econômicos, tem proporcionado um grande impacto em praticamente todos os setores da sociedade. Além disso, a inexistência de padrões no setor de telecomunicações evidenciou as diferenças entre as estruturas das operadoras, onde existe uma constante introdução de novos serviços. Para atender rapidamente esta necessidade, faz-se necessário o uso de tecnologias avançadas que permitam conduzir a especificação e desenvolvimento desses serviços de uma maneira ágil e eficiente. Dentre tais tecnologias, pode se citar a orientação a objetos, reuso de componentes, sistemas distribuídos, arquitetura de serviços e, principalmente, TINA (*Telecommunications Information Networking Architecture*) devido à sua natureza aberta e independente de tecnologia. Os conceitos e os princípios de TINA foram elaborados com o objetivo de solucionar problemas existentes em IN (*Intelligent Network*), como o de controle de serviços centralizados e do modelo de dados de serviços. Nesse sentido, torna-se evidente que serviços mais complexos como multimídia e vídeo conferência precisam ser manipulados e gerenciados de uma forma mais rápida, eficiente e dinâmica. Neste contexto, TINA desenvolveu uma arquitetura detalhada para as redes de comunicações multi-serviços que permitem o intercâmbio de informações entre os usuários e os provedores em tempo real dentro de um ambiente seguro e confiável, onde o gerenciamento dos serviços TINA definidos pelas funções de gerenciamento FCAPS (*Failure, Configurations, Accounting, Performance* e *Security*) são ainda questões abertas para a pesquisa. Portanto, este trabalho de pesquisa tem como objetivo realizar um estudo e a análise deste contexto, propondo um modelo para responder as questões de segurança em um ambiente de serviço TINA. Este modelo esta principalmente relacionado com o gerenciamento de contabilidade em tempo real para múltiplos usuários e múltiplos provedores, onde os aspectos de segurança estão relacionados às políticas e mecanismos proporcionados pelos modelos de Segurança Multilateral e RBAC (*Role-Based Access Control*) e validado através do uso da técnica de descrição formal LOTOS e a implementação de um protótipo.

Palavras-Chaves

TINA, Gerenciamento de Contabilidade, Segurança Multilateral, RBAC.

ABSTRACT

In the telecommunications area, the increasing evolution and the constant development of new technologies, allied to economic factors, has proportionated a great impact in all the society sectors. Moreover, the inexistence of standards in the telecommunications sector has evidenced the differences among the structure of the operators, where there is a constant introduction of new services. To meet these necessity quickly, it becomes necessary the use of advanced technologies to allow the specification and development of these services in an agile and efficient way. Amongst such technologies, can be cited the object orientation, reuse of components, distributed systems, services architecture and, mainly, TINA (Telecommunications Information Networking Architecture) due to its openness and independent technology nature. The concepts and the principles of TINA was elaborated with the objective to solve existing problems in IN (Intelligent Network), as the control of centralized services control and the model of services data. Therefore, it becomes evident that more complex services as multimedia and video conference needs to be manipulated and managed in a faster, efficient and dynamic way. In this context, TINA developed a detailed architecture for the multi-services communication network that allow the real time interchange of information between users and suppliers in a trustworthy and secure environment, where the management of the TINA services specified by the FCAPS (Failure, Configurations, Accounting, Performance and Security) management functions are still open questions for research. Therefore, this work has as objective the study and the analysis of this context, with the proposal of a model to answer the questions of security in a TINA service environment. This model is mainly related with the real time accounting management using multiple users and multiple suppliers, where the security aspects are related with the policies and mechanisms provided by Multilateral Security and RBAC (Role-Based Access Control) models and validated through the use of the LOTOS of Formal Description Technique and the development of a prototype.

Key-Word

TINA, Accounting Management, Multilateral Security , RBAC, LOTOS.

1. INTRODUÇÃO

1.1 Introdução

Nas últimas décadas tem-se observado uma evolução exponencial no número de estações de trabalho. Este aumento considerável deve-se à ascensão das redes de computadores e de suas tecnologias subjacentes (tais como, o processamento de dados e das telecomunicações), que ocupam um lugar estratégico cada vez mais significativo nas indústrias, nas áreas comerciais e educacionais. Certamente, estas redes de comunicação transformam-se no suporte permanente da informação para tais organizações.

Assim sendo, resultam em estruturas de comunicação que contêm mais e mais sistemas heterogêneos, e que estão fortemente inter-conectados. Estas estruturas de comunicação, que são redes corporativas, estão inter-conectadas a redes de área local de sistemas pequenos, e redes a longas distâncias que suportam aplicações distribuídas, tornando-se cada vez mais necessárias.

Entretanto, a atual diversidade de redes inter-conectadas, evoluem a aplicações que requerem de Sistemas Distribuídos. Tal sistema é caracterizado pela sua distribuição heterogênea e a sua autonomia, a qual é oferecida de forma transparente ao acesso, localização, migração e à competição (concorrência) dos serviços. A todas essas características, é adicionada a necessidade de Qualidade do Serviço (*QoS - Quality of Service*), que se tornam uma das principais prioridades. Estas características requerem ainda a busca e o desenvolvimento para seu gerenciamento a fim de trazer uma resposta total, que trate ao mesmo tempo da gerência de redes e de serviços.

Hoje, as redes de telecomunicações e de informação não podem suportar uma infra-estrutura que permita acessar os serviços e às aplicações dos multifornecedores sem um gerenciamento de todo o sistema (redes e serviços), com uma visão global e coerente. Além disso, o desenvolvimento de novos serviços de telecomunicações vem impondo desafios aos provedores de equipamentos, operadoras e empresas fornecedoras de software (BRENNAN, 2000, PARK e BAEK, 2001, MAMPAEY e COUTURIER, 2000).

Assim, foram realizados esforços consideráveis pelas organizações tais como a ISO (*International Standard Organization*), ITU-T (*International Telecommunication*

Union - Telecommunications Sector) e o NM-Forum (*Network Management Forum*), para padronizar a tarefa de gerenciamento das redes heterogêneas, cuja finalidade é o serviço final estar em conformidade com a qualidade requerida; que novos serviços sejam representativos no contexto distribuído, como UPT (*Universal Personnal Telecommunication*), VoD (*Video on Demand*), entre outros, e que demandaem a representação de novos modelos conceituais para sua exploração e seu controle.

Nas últimas décadas, as redes de telecomunicações foram baseadas em soluções proprietárias, as quais são extremamente caras e tornaram-se lentas, pois as rotinas publicas de serviço eram fortemente dependentes do dispositivo e que foram integradas no software de controle. Hoje, o mercado das telecomunicações criou a necessidade de produtos mais flexíveis e inter-operáveis, ao mesmo tempo em que incentivou aos usuários a reivindicar o uso de serviços mais complexos e mais personalizados. Isto, no mercado, envolve um volume de software necessário aos novos serviços e uma pressão por tarifas cada vez mais reduzidas impondo novos paradigmas para o desenvolvimento, implementação e gerência dos serviços de telecomunicações (SEKKAKI, 2002).

Diante de tais fatos, o provedor de serviços de redes de telecomunicações em definitivo deve dar resposta às exigências do mercado: o objetivo é adaptar as infra-estruturas existentes de rede de comunicação às funcionalidades de novos serviços mais sofisticados.

Consideram-se atualmente dois modelos para abordar o problema de desenvolvimento de novos serviços:

- **Modelo IN** (*Intelligent Network*):

  Foi desenvolvido para permitir a criação suficientemente fácil e rápida de novos serviços nas telecomunicações. Foi adotada pela ITU, provando ser confiável em redes de telefonia comutada (PAVLOU, 1998).

- **Modelo TINA** (*Telecommunication Information Networking Architecture*):

  Foi criado por diversas e grandes companhias do setor das telecomunicações e de processo de dados. É uma arquitetura de software comum, baseada nos conceitos e princípios da programação distribuída orientada a objeto, como CORBA (*Common Object Request Broker Architecture*), da OMG (*Object Management Group*) e das especificações JAVA. O modelo TINA foi projetado para o mercado competitivo com o intuito de ser utilizado como uma base para a

criação e o gerenciamento de novos serviços para as redes fixas e móveis de serviços multimídia (TINA, 2000).

Depois do modelo IN, o ITU-T introduziu a solução TMN (*Telecommunication Management Network*) (ITU, 1993), que permite fornecer um ambiente adequado para a gerência dos serviços IN e do modelo de especificação da arquitetura TINA, que permite a gerência dos serviços de modo que a difusão e diversidade dos serviços da arquitetura TINA tem como objetivo solucionar os problemas de portabilidade e controle centralizado dos serviços no modelo IN (TINA, 2000, FINKELSTEIN, 2000).

Para descrever um serviço de telecomunicações no modelo TINA, deve-se identificar e diferenciar os aspectos que se relacionam ao acesso e gerenciamento do serviço. O termo ***serviço*** está relacionado com os seguintes conceitos: serviço da camada OSI, serviço de telecomunicações, serviço de gerência, serviço executado por um objeto, enquanto o termo ***gerência de serviço***, está relacionado com a atividade de análise, monitoramento e controle de serviço (ABARCA, 1998, PAVLOU, 1998).

Nesse contexto, a crescente demanda por serviços de comunicação e o rápido avanço da informática conduziram à criação de complexas estruturas de redes de telecomunicações, onde predominam o tráfego de informações digitais como voz, imagens e dados (BRENNAN, 2000). Sendo a Internet uma grande aliada na era da multimídia, seu desenvolvimento ocorreu de forma paralela a todo o desenvolvimento da telefonia e da telecomunicação tradicional. Elas usam diferentes tecnologias para fornecer diferentes características. Especificamente, a tecnologia da Internet emprega grande parte da inteligência nos terminais dos usuários, enquanto que no caso da telecomunicação tradicional, a inteligência se localiza principalmente dentro da própria rede. Quando integradas estas duas tecnologias podem oferecer uma vasta gama de oportunidades para prover serviços de informações e multimídia versáteis, além de inovar a maneira como estes serviços são criados e disponibilizados (TINA, 2000).

Desse modo, TINA permite disponibilizar uma arquitetura comum de software para serviços multimídia e de informação, no âmbito das telecomunicações, visando introduzir e controlar novos serviços de uma maneira rápida, ágil, padronizada e de elevada capacidade de gerência e configuração (PINTO, 2002).

1.2 Justificativa

O motivo desta pesquisa é propor e desenvolver um modelo de gerenciamento de contabilidade e segurança na arquitetura TINA, em relação ao fornecimento e manutenção de serviços, como por exemplo, serviços multimídia, que consiste no uso de todos os conceitos, dos princípios, dos padrões e dos meios para a construção, distribuição e exploração dos serviços que a arquitetura propõe na literatura disponível por TINA-C (*Consortium – TINA*), o qual interage com os padrões que incluem ATMF (*Asynchronous Transfer Mode Forum*), DAVIC (*Digital Audio-Visual Council*), ITU-T, TMF (*Telemanagement Forum*) e OMG (*Object Management Group*) (OMG, 1999), de forma a realizar os acordos das especificações e evitar duplicação de trabalhos (PINTO, 2002).

Desse modo, a contabilidade TINA é um serviço de gerência, essencial a todo os serviços de TINA. Sua natureza dinâmica, flexível e distribuída revolucionou os conceitos tradicionais de contabilidade onde as novas modalidades de pagamento, tais como a cobrança e a emissão da fatura pelos serviços fornecidos devem apresentar desafios relacionados a uma contabilidade confiável. A gerência de contabilidade possibilita definir para algum serviço, quem pode ser faturado, os limites contáveis e a cobrança relacionada a seu uso. O controle da fatura dinâmica e distribuída está relacionado a aspectos como a definição dos objetos e os eventos de contabilidade associados (HAMADA, 1996).

A segurança de contabilidade em TINA, permite a proteção da informação contável, tal como os objetos de contabilidade, eventos e estruturas de contabilidade, contra um intruso ou a usuários maliciosos. Considerando a natureza distribuída da gerência dos serviços do modelo TINA, a segurança também torna possível a proteção da informação contábil contra uma má gerência do provedor do serviço, a qual pode trazer um prejuízo econômico ao usuário. Esta proteção é realizada por um mecanismo de certificação mútua de eventos entre o usuário e o provedor, como por exemplo, um SLA (*Service Level Agremment*) (STAAMANN, 1997).

Nesse sentido, as principais preocupações com relação à segurança são a **confidencialidade**, onde as informações só são reveladas para pessoas autorizadas; **integridade**, onde a informação só pode ser modificada por usuários autorizados nas formas autorizadas; **responsabilidade**, onde os usuários são responsáveis por suas

ações relacionadas à segurança; e **acessibilidade**, onde usuários autorizados não podem ter seu acesso negado maliciosamente (OH e PARK, 2000).

O controle de acesso por si só não é uma solução completa para a obtenção de segurança em um sistema. Para que a segurança seja completa, é necessária a existência de outros serviços de segurança como **autenticação**, **auditoria** e **gerenciamento**. A **autenticação** é para garantir que um usuário é quem ele diz ser; **controles de acesso**, para controlar a que um usuário pode ter acessar em um sistema; **comunicações seguras**, para proteger informações em trânsito entre componentes; **auditoria de segurança**, para gravar e analisar o que o usuário faz no sistema; e **gerência de segurança**, para gerenciar informações de segurança incluindo as políticas de segurança (SANDHU e SAMARATI, 1994).

De acordo com um estudo do NIST (*National Institute of Standards and Technology*), essa necessidade de controle de acesso de acordo com os papéis que os usuários desempenham na organização define uma política de segurança denominada de RBAC (*Role-Based Access Control*), ou seja, um modelo de segurança baseado em papéis (WESTPHALL, 2000).

O RBAC é um termo utilizado para descrever políticas de segurança que controlam o acesso de usuários a recursos computacionais, baseado na construção de *roles* (papéis, funções). Esses papéis definem um conjunto de atividades concedidas para usuários autorizados (ARMANINI, 2003).

Estes princípios e políticas do modelo RBAC são outra motivação da linha de pesquisa deste trabalho, de forma a aplicar este modelo e suas extensões para o ambiente de serviço TINA, sendo possível realizar um desejado balanceamento entre a flexibilidade e segurança no gerenciamento de serviços de telecomunicação.

Outro aspecto importante na abordagem da segurança e que reflete no ambiente de serviço TINA: supõem que está completamente claro "*quem*" tem que ser protegido contra "*quem*". Por exemplo, o foco de TCSEC (*Trusted Computer System Evaluation Criteria*) na proteção de proprietários e operadores do sistema contra "atacantes" externos e aos usuários internos mal comportados ou mal intencionados. Também supõe-se freqüentemente que uma política de segurança pode definitivamente descrever que ações são autorizadas e devem ser reforçadas apropriadamente para manter um estado seguro. Estas considerações não se aplicam realmente quando diversos

participantes com interesses diferentes ou conflitantes estão envolvidos, por exemplo, em sistemas telefônicos ou de Internet:

- Subscritores necessitam proteção de outros, especialmente de operadores de redes ou dos provedores de serviços que monitoram sua comunicação; e
- Provedores necessitam de proteção contra fraudes, por exemplo, através de chamadas não pagas ou não contabilizadas, para as quais nenhum subscritor tomará responsabilidades (RANNENBERG, 2000).

A Segurança Multilateral visa conseqüentemente o balanceamento entre os diferentes participantes competindo pelos requisitos de segurança. Assim, considerando os requisitos de segurança de todos os participantes envolvidos e considerando-os também como atacantes potenciais. Isto é especialmente importante para sistemas de comunicação abertos, porque não pode esperar-se que os diversos participantes confiem uns nos outros. A Segurança Multilateral inclui, segundo (RANNENBERG, 2000):

1. Consideração de conflitos:
    a) Diversos participantes envolvidos em um sistema podem ter diferenças, talvez interesses e objetivos conflitantes de segurança.
2. Respeito dos interesses:
    a) Os participantes podem definir seus próprios interesses;
    b) Os conflitos podem ser reconhecidos e negociados; e
    c) Os resultados negociados podem reforçar a confiança.
3. Suporte à autoridade:
    a) Cada participante é minimamente requerido a confiar na *honestidade* de outros; e
    b) Cada participante é minimamente requerido a confiar na *tecnologia* de outros.

Desse modo, esta tendência reflete nos diversos desenvolvimentos, os quais causam problemas de segurança no mundo das telecomunicações tradicionais atuais. A diminuição de custo na transmissão na largura de banda permite aplicações distribuídas multimídia em tempo-real. O potencial CPE (*Customer Premises Equipment*) disponível para o uso progressivo da tecnologia de computadores para os usuários, disponibiliza uma grande quantidade de serviços para serem entregues via uma infraestrutura comum para terminais de usuário final multi-propósito. A de-regulação mundial do ambiente de

telecomunicações cria um mercado aberto para o fornecimento de serviços de telecomunicações. Assim, redes de telecomunicações não são só popularizadas para uma grande quantidade de usuários, mas também para uma grande quantidade de provedores de serviços. A cooperação, e ao mesmo tempo, concorrência de vários provedores na mesma rede física (esperando a rede atuar como uma infra-estrutura para serviços variando desde teleconferência e vídeo sob demanda para comércio eletrônico e bancos eletrônicos) elevam uma forte demanda por segurança multilateral e privacidade de uso de serviços e comunicação. Uma importante questão com respeito a segurança multilateral é a esperada heterogeneidade das políticas de segurança e segurança tecnológica de vários operadores de redes, provedores de serviços e usuários finais (BUTTYÁN, 1999), o que é compatível com os conceitos da arquitetura TINA.

Portanto, do ponto de vista conceitual, questões de segurança e de contabilidade no gerenciamento de serviço de telecomunicações em ambiente de serviço TINA devem estender-se a modelos que reforcem as políticas de segurança, as quais suportem a negociação em um contexto de segurança, como por exemplo, através de abordagem RBAC e Segurança Multilateral.

1.3 Objetivos da Tese

A TINA estabelece uma arquitetura baseada em tecnologias de computação distribuída, que permitem às redes de telecomunicações a introdução e o controle de novos serviços de uma maneira ágil e padronizada.

Na atualidade, serviços mais complexos precisam de funcionalidades de gerenciamento que permitam uma rápida, confiável e cuidadosa troca de informações entre os fornecedores e consumidores dos serviços. Nesse sentido, TINA especifica a Arquitetura de Gerenciamento de Contabilidade que permite oferecer um gerenciamento de contabilidade consistente e que garante uma contabilidade confiável através de um serviço distribuído em um ambiente multi-domínio TINA, onde o aspecto de segurança é parte essencial para a proteção das informações contábeis contra usuários maliciosos ou fornecedores mal intencionados, o que é um fator crítico em TINA e em sistemas distribuídos (HAMADA, 1996).

Nesse contexto é que surgiu a idéia deste trabalho, o qual tem como objetivo propor um modelo de segurança para o gerenciamento de contabilidade de TINA, que

permita solucionar estes fatores críticos através de políticas e mecanismos de segurança proporcionados pelos modelos da Segurança Multilateral e de Controle de Acesso baseado em Papéis (RBAC). Desse modo, em uma sessão de serviço estabelecida por um consumidor e um fornecedor, deve ser permitida uma negociação das características de segurança de ambos através de mecanismos de Segurança Multilateral, na parte relacionada ao processo de acesso, e à definição de papéis através de RBAC, na parte do processo de uso do serviço.

Para atingir estes objetivos propõe-se:

1. Realizar um estudo dos principais conceitos relacionados à TINA, mostrando um escopo geral de seu modelo de negócios, da sua arquitetura de serviços e do gerenciamento de serviço.
2. Realizar um estudo da Arquitetura de Contabilidade de TINA, mostrando os conceitos relacionados à transação de serviço dentro de um contexto de gerenciamento de contabilidade TINA e que envolvem aspectos relacionados à segurança.
3. Analisados os conceitos definidos acima, propor um modelo de segurança para o Gerenciamento de Contabilidade TINA, de modo a definir os contextos de segurança e de contabilidade através do estabelecimento de mecanismos e políticas de segurança para os modelos de Segurança Multilateral e RBAC.
4. Validar o modelo proposto, utilizando o modelo de descrição formal LOTOS e um protótipo para a validação dos resultados esperados de acordo com os padrões TINA para o:
    - Gerenciamento de contabilidade; e
    - Gerenciamento de segurança.

1.4 Organização da Tese

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta uma visão geral da arquitetura TINA; o Capítulo 3 apresenta o sistema de gerenciamento de contabilidade, os principais componentes e suas características; o Capítulo 4 analisa os serviços e mecanismos de segurança e apresenta considerações de segurança para contabilidade; no Capítulo 5 descreve-se a definição das políticas e mecanismos de segurança para o modelo de segurança. No Capítulo 6 descreve-se a proposta de

pesquisa da tese através do modelo de segurança que especifica o desenvolvimento de um modelo de contabilidade e de segurança em um ambiente de serviço baseado em TINA. O Capítulo 7 define a especificação formal do modelo de segurança e o Capítulo 8 mostra a implementação do protótipo. Finalmente, as conclusões e os trabalhos futuros aparecem no Capítulo 9.

2. VISÃO GERAL DA ARQUITETURA DE GERENCIAMENTO TINA

2.1 Introdução

A TINA fornece uma arquitetura aberta na qual novos serviços e funcionalidades podem ser facilmente construídos e adicionados, diferente da maneira tradicional onde novos serviços são adicionados à rede de maneira não padronizada. Isto soluciona não somente o problema do desenvolvimento de novas aplicações, mas também o problema de interoperabilidade entre sistemas de diferentes fabricantes (NIEHAUS, 1999).

Estas características apresentadas por TINA permitem definir três importantes objetivos, que são:

a) Facilitar a distribuição de serviços de multimídia e de informações;
b) Facilitar a criação e gerenciamento de novos serviços; e
c) Criar uma arquitetura aberta de componentes de telecomunicações e informações.

O consórcio TINA (TINA-C) procura cumprir estes objetivos através da utilização dos mais recentes desenvolvimentos tecnológicos, incluindo redes de banda larga, diversas formas de comunicação existentes e principalmente as tecnologias Internet e Intranet, com as quais TINA pretende contribuir com serviços tais como pontos de acessos e roteamento inteligente (NIEHAUS, 1999).

TINA é aplicável a todas as partes de um sistema de telecomunicações ou de informação:

- Terminais: computadores, pontos de acesso, etc.;
- Servidores de transporte: *switches*, roteadores;
- Servidores de serviço: Web, VoD (*Vídeo-on-Demand*), VoIP (*Voice over IP*); e
- Servidores de gerenciamento: autenticação, contabilidade e cobrança.

Para assegurar a interoperabilidade, portabilidade e reusabilidade dos componentes de software, é necessário existir uma independência de tecnologias específicas que permitam o compartilhamento na criação e gerenciamento de sistemas complexos. Isto consiste de objetivos comuns, princípios e conceitos abordando o gerenciamento de serviços, recursos e partes do Ambiente de Processamento Distribuído (DPE - *Distributed Processing Environment*). Para isto, TINA é baseada em quatro princípios fundamentais no gerenciamento da arquitetura:

1) **Orientado a objetos e distribuído**: a arquitetura de gerenciamento TINA toma vantagem do DPE, o qual conceitua-se como um ambiente orientado a objetos distribuído para o gerenciamento de recursos e de serviços. Está construído sobre uma abordagem orientada a objetos e distribuído de Redes de Gerenciamento de Telecomunicações - TMN (*Telecommunications Managements Networks*) e Arquitetura de Gerenciamento Distribuído Aberto - ODMA (*Open Distributed Management Architecture*);
2) **Orientado a serviço**: o gerenciamento de serviço deve garantir Qualidade de Serviço (QoS) como uma parte integrada do serviço. O desmembramento de determinados componentes de software não altera o funcionamento dos outros componentes;
3) **Dinâmico e flexível**: a natureza orientada a serviço de TINA requer que os recursos sejam necessariamente atribuídos e configurados mais dinamicamente, para suportar a flexibilidade requerida pelos serviços TINA; e
4) **Integrada**: o gerenciamento é uma parte integrada do fornecimento de serviço e inerente às diferentes camadas de TINA (serviço, recursos e DPE) (HAMADA, 1997).

A abordagem de gerenciamento TINA consiste de várias regras relacionadas e construídas sobre outras abordagens de gerenciamento, tais como TMN (ITU, 1993), ODP (ITU, 1995) e OmniPoint (OMNI, 1994). A Figura 2.1 ilustra a arquitetura de gerenciamento TINA, organizada ao redor de quatro conceitos associados ao gerenciamento de serviço:

- **Conceito de Particionamento**: TINA é particionado em três camadas: de serviço, de recursos e DPE. A arquitetura de gerenciamento é igualmente particionada, centrada sobre o gerenciamento de serviços e de recursos. Este também suporta o conceito de domínio e gerenciamento e entre domínios;
- **Conceito Funcional**: é representado pelas funções FCAPS (Falha, Configuração, Contabilidade, Desempenho e Segurança). Para suportar integralmente as funções FCAPS de uma sessão de serviço, são fornecidas construções como gerenciamento de contexto e transação de serviço;

- **Conceito Computacional**: representa o suporte computacional para as necessidades de gerenciamento. Este suporte computacional é oferecido pelo DPE; e
- **Conceito de Ciclo de Vida**: representa questões de ciclo de vida, que incluem o gerenciamento do ciclo de vida do serviço e o gerenciamento do ciclo de vida do usuário, como por exemplo, o gerenciamento do ciclo de vida do cliente (HAMADA, 1997).

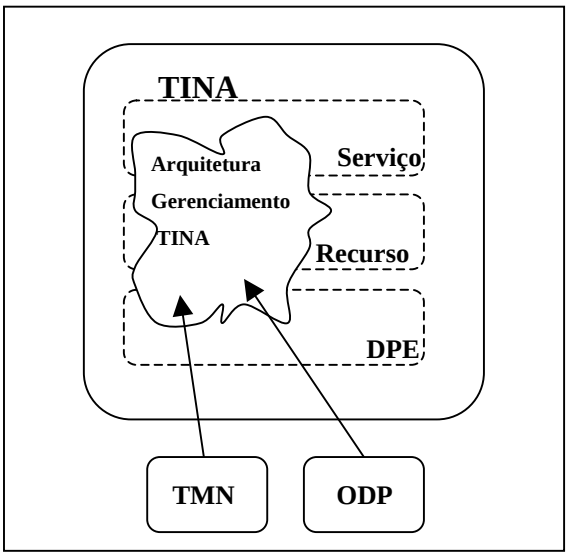


Figura 2.1 - Arquitetura de Gerenciamento TINA (HAMADA, 1997).

A arquitetura de gerenciamento TINA separa as aplicações de telecomunicações, isto é, softwares que implementam as funcionalidades providas pelo sistema do DPE, o qual suporta a execução distribuída de aplicações. Enquanto as aplicações e o DPE são implementados como um conjunto de objetos que interagem, TINA não obriga o uso de uma linguagem de programação orientada a objetos para sua implementação. Uma vez que TINA não existe isoladamente, é preciso que se tenha a habilidade de interagir com sistemas não-TINA, em um (ou ambos) nível(is) de serviço(s).

2.1.1 Arquitetura em Camadas

A arquitetura TINA é dividida em 4 camadas, como mostra a Figura 2.2. Iniciando pela camada do nível inferior, temos:

a) **Camada de Hardware**: envolve os processadores, memória, dispositivos de comunicação;

b) **Camada de Software**: envolve os sistemas operacionais, softwares de comunicações e outros softwares de suporte encontrados em recursos computacionais. Esta camada é chamada também de Ambiente de Comunicação e Computação Nativa (*NCCE - Native Computing and Communications Environment*). O NCCE é composto de um conjunto de nós computacionais interconectados, onde cada nó pode suportar diferentes tecnologias de software e hardware;

c) **Camada DPE**: fornece suporte para a execução das aplicações de telecomunicações distribuídas, além de possuir uma visão independente da tecnologia dos recursos computacionais. Uma vez que as aplicações distribuídas são implementadas como um conjunto de objetos que interagem e que podem estar localizados em diferentes nós, o DPE provê suporte para a localização do objeto e interação remota; e

d) **Camada de Aplicações**: implementa as funcionalidades oferecidas pelo sistema.

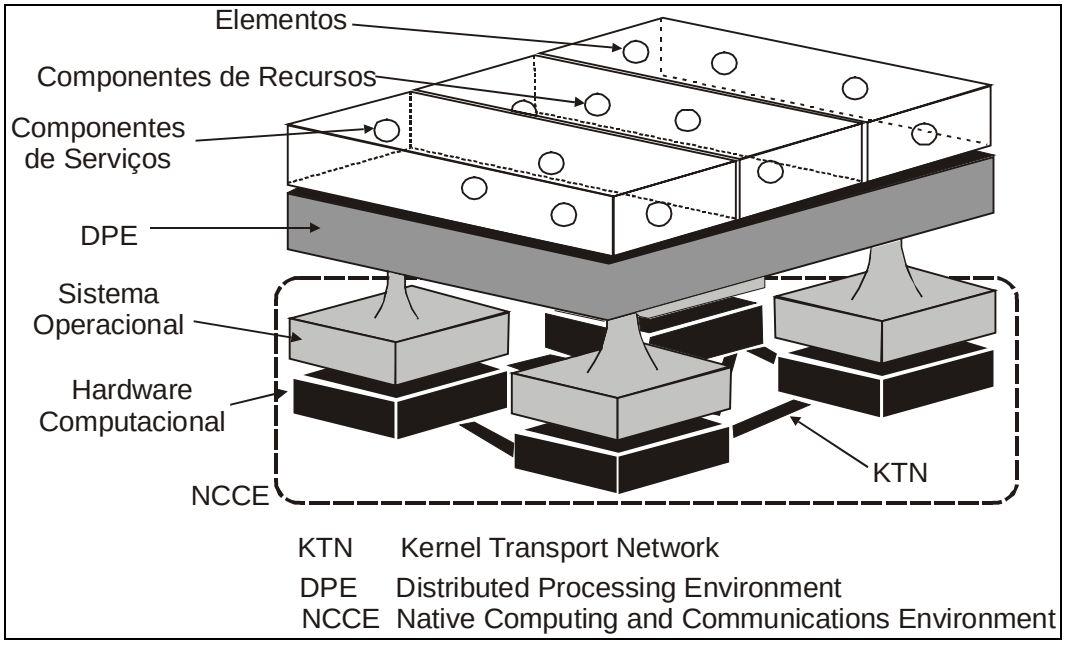


Figura 2.2 - Arquitetura em Camadas TINA (BUTTYÁN, 1999).

O sistema TINA em camadas consiste de nós separados no qual cada nó possui um NCCE e um DPE. A junção de todos os DPEs forma uma “superfície DPE”, onde cada nó pode ser de propriedade de diferentes administradores, consumidores e

provedores de serviço. A interface inter-DPE é definida em comum acordo entre todos os fabricantes ou por um padrão.

Em TINA, os serviços são realizados como aplicações distribuídas e consistem de componentes de serviços que interagem com os demais via DPE. O DPE é uma camada de software que opera sobre o NCCE, o qual é uma abstração do hardware computacional e o sistema operacional do nó de serviço. Enquanto o NCCE é dependente da tecnologia, o DPE oferece uma interface uniforme para o ambiente distribuído. O DPE consiste de implementações CORBA como o DPE *Kernel* e de serviços específicos TINA.

Os componentes de serviços são especificados em uma Linguagem de Descrição de Objeto TINA (ODL - *Object Description Language*), a qual é um superconjunto da Linguagem de Descrição de Interface CORBA (IDL - *Interface Description Language*). Elas consistem de objetos computacionais (CO - *Computational Objects*) ou grupos de CO. Os COs podem ter duas classes de interfaces: **interfaces operacionais**, as quais são comparáveis com interfaces de objetos em CORBA, e **interfaces de fluxo**, as quais são projetadas para transportar uma seqüência arbitrária de bytes entre dois componentes (por exemplo, fluxo de bits de áudio ou vídeo). As mensagens entre as interfaces operacionais são trocadas via a Rede de Transporte *Kernel* (KTN - *Kernel Transport Network*), enquanto os fluxos são transferidos via Rede de Transporte (BUTTYÁN, 1999).

2.1.2 A Arquitetura TINA

A arquitetura TINA é dividida em quatro sub-arquiteturas. Os relacionamentos entre estas quatro arquiteturas são demonstrados na Figura 2.3 e explicados a seguir:

1) **Arquitetura de Serviço**: consiste de um conjunto de conceitos, fundamentos, regras e padrões (*guidelines)* para a construção, desenvolvimento e operação dos serviços TINA;
2) **Arquitetura de Recursos de Rede**: estabelece um conjunto de conceitos que descrevem as redes de transporte de uma maneira independente da tecnologia e provê mecanismos para o estabelecimento, modificação e liberação de conexões de rede (estes mecanismos são definidos como um conjunto de interfaces disponíveis para a camada de recursos de rede), (ABARCA, 1997);

3) **Arquitetura Computacional**: define os conceitos de modelagem orientada a objetos e o ambiente de processamento distribuído (DPE) que fornece ao sistema o meio para que objetos possam se localizar e interagir. Estes conceitos estão baseados no RM-ODP (*Reference Model for Open Distributed Processing*) (ISO, 1994); e
4) **Arquitetura de Gerência**: especifica os princípios e conceitos gerais de gerenciamento para a arquitetura TINA. Estes conceitos são derivados dos padrões (ITU, 1993).

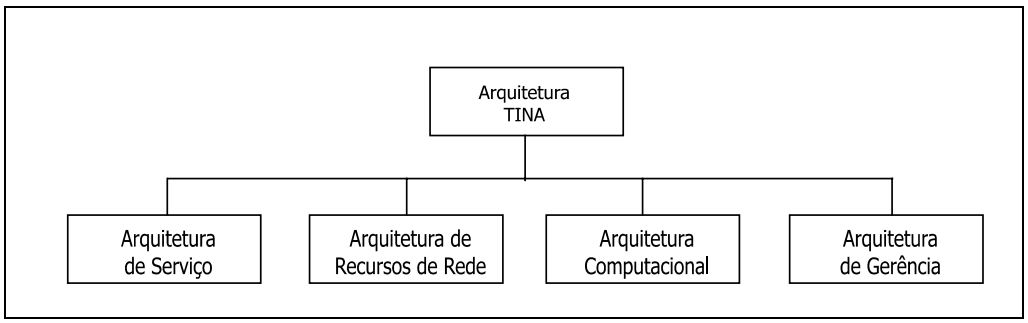


Figura 2.3 - Sub-Arquiteturas TINA (NIEHAUS, 1999).

2.2 O Modelo de Negócios TINA

O modelo de negócios define um *framework* para especificar pontos de referência[1] e propagar requisitos de negociação, provendo a estrutura para especificar, adicionar e modificar pontos de referência e componentes na TINA (MULDER, 1997).

Em TINA as implementações da arquitetura de serviço e da arquitetura de recursos de rede são aplicações executadas em um ambiente DPE. As especificações dos pontos de referência descrevem as interações entre estas aplicações e a plataforma DPE em todos os aspectos do TINA.

Na Figura 2.4 é ilustrado o modelo de negócios utilizado em TINA. Este modelo mostra cinco "áreas-chaves" (funções de negócios) com seus relacionamentos identificados. Estes relacionamentos no modelo de negócios são usados para identificar e definir os pontos de referências (por ex. *Bkr, Ret, Tcon, 3Pty,* etc) das aplicações.

[1] Um Ponto de Referência define um conjunto de interfaces associado com o modelo o qual é considerado potencialmente adequado para uma plataforma DPE.

Os consumidores (*Consumer*) solicitam serviços para os provedores (*Retailers*), porém, o serviço atual é fornecido pelo provedor de serviço terceirizado (*Third Party Service Providers*). Os provedores de conectividade (*Connectivity Provider*) oferecem a conectividade necessária para o transporte do fluxo *stream* de informação entre os participantes (S*takeholders*). O intermediário (*Broker*) atua como uma classe de serviço de página amarela e página branca (por exemplo, este entrega referências para serviços que podem ser descritos pelas características do serviço, porém também pelos nomes dos provedores) (BUTTYÁN, 1999).

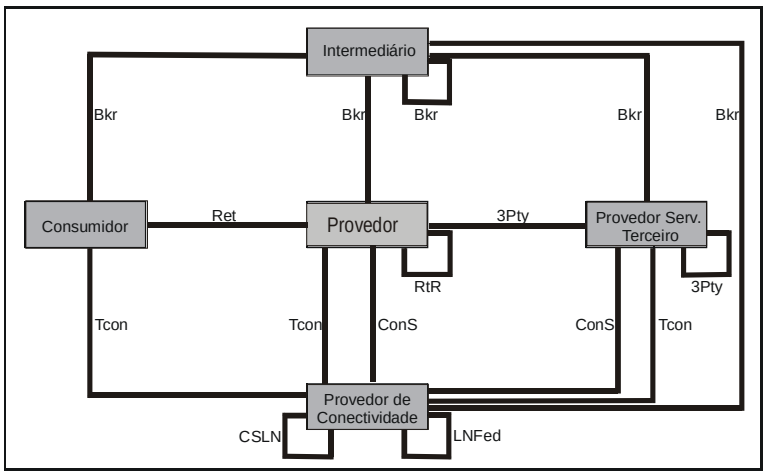


Figura 2.4 - Modelo de Negócios TINA (MULDER, 1997).

Cada participante tem seu próprio domínio administrativo e atua em um ou mais papéis. Qualquer relacionamento pelo uso de serviço é composto pela interação simples de dois participantes: usuário-provedor. O relacionamento usuário-provedor contém dois tipos de interações: **acesso** e **uso**. A parte de acesso está relacionada com o estabelecimento de um relacionamento temporário e confiável entre o domínio usuário e domínio provedor que é um pré-requisito para as interações de uso. A interoperabilidade entre os domínios está garantida pela definição de pontos de referência inter-domínio (BUTTYÁN, 1999).

2.2.1 Escopo do Modelo de Negócios

O modelo de negócios em TINA especifica:

- Um *framework* de negócios genérico para todos os integrantes da TINA. Ele define um conjunto de condições na qual os seguintes itens podem ser realizados:
  - a criação de novas funções de negócios e pontos de referência, onde estas novas funções podem interagir com as já existentes; e
  - a melhoria de funções de negócios e pontos de referências existentes para agregar novas regras e funcionalidades, a partir de pontos de referência já definidos e implementados.
- Um conjunto inicial de funções e relacionamentos de negócios para aplicar em TINA, utilizando *drivers* de serviços de telecomunicações e serviços de informações já existentes no mercado; e
- Requisitos impostos ao TINA para atender a um conjunto específico de serviços, propagando estes requisitos nas funções e relacionamentos de negócios nos pontos de referências dos participantes TINA proprietários do sub-sistema.

2.2.2 Definição do *Framework*

A base dos sistemas e subsistemas TINA é constituída pelos objetos de informações, objetos de computação e objetos de engenharia, sob um domínio administrativo de negócios e separados por pontos de referências. Desta maneira, a especificação de regras e interações entre domínios de gerência TINA se propagam através da especificação da visibilidade e direitos em cada tipo de objeto nos domínios relacionados. Estes direitos e visibilidades são colocadas em um contrato. Um contrato é um contexto definindo as restrições aonde o(s) ponto(s) de referência(s) irá(ão) operar. O contrato é estabelecido entre domínios de gerência de negócios e pode ser negociado *on-line* ou *off-line*.

2.2.3 Contrato

O contrato provê a base para os contextos definidos em outros *viewpoints*[2]. Com as restrições definidas no contrato, os contextos em outros *viewpoints* podem ser

[2] Uma forma de abstração alcançada usando um conjunto selecionado de conceitos de arquiteturas e regras de estruturação de maneira a focalizar um interesse em particular em um sistema. TINA provê conceitos de arquitetura para *viewpoints* de informação, computacional e de engenharia.

negociados *on-line* ou *off-line*. Entretanto, o contrato nunca pode ser modificado como resultado de negociações em outro *viewpoint*, uma vez que um simples *viewpoint* somente provê uma visão parcial das interações entre os domínios administrativos de negócios e pode violar as políticas negociadas para outros *viewpoints*.

2.2.4 Domínio Administrativo de Negócios

Um domínio administrativo de negócios é definido pelos requisitos de um *business role*. Um domínio administrativo de negócios irá interagir com outro através dos pontos de referência, os quais são as implementações dos relacionamentos de negócio.

Para permitir a interação entre os domínios administrativos de negócios, eles devem possuir um ponto de acesso, isto é, um objeto que aceite requisições de serviços provenientes de outros componentes. Informações sobre estes acessos são registradas e gerenciadas pelo *broker*[3] de negócios.

O conceito de domínio administrativo de negócios é baseado em *propriedades*, partindo do ponto de vista da empresa. A propriedade implica em um privilégio dominante na gerência de entidades dentro do domínio. Este privilégio pode ser delegado para domínios em outros *viewpoints.* Por exemplo, o domínio de gerência no *viewpoint* da informação pode resolver um problema específico (p. ex., no gerenciamento de falhas) ou um relacionamento como no gerenciamento de rede em uma rede particular de propriedade de um domínio de gerência de negócios, (Figura 2.5). Em outras visões o domínio administrativo de negócios pode ser dividido e agregado em outros tipos de domínios de maneira a facilitar a solução de problemas e relacionamentos.

2.2.5 Pontos de Referência

O ponto de referência é composto por diversas especificações de *viewpoints* relacionados por um contrato, com o propósito de favorecer a reutilização e a implementação modular das especificações. O ponto de referência contém os seguintes segmentos:

---

[3] É um *business role* que provê informações sobre como encontrar certos serviços e certos participantes em TINA.

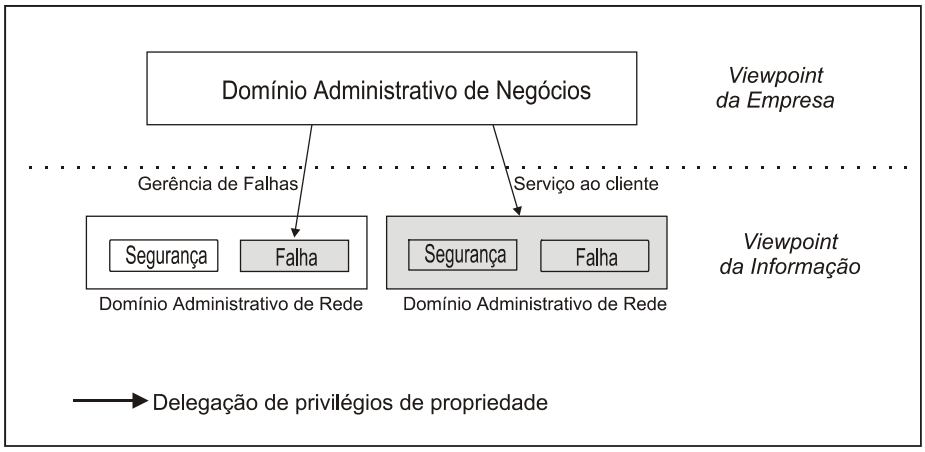


Figura 2.5 - Mapeamento de Domínios (MULDER, 1997).

1. **Segmento de negócios:** limitações de escopo, requisitos funcionais e não-funcionais atribuídos aos relacionamentos de negócios pelos componentes. Ele é derivado a partir dos requisitos dos componentes e suas interações;
2. **Segmento de informações:** define a informação a qual é compartilhada entre os domínios de gerência de negócios;
3. **Segmento computacional:** define as interfaces em objetos computacionais de maneira a serem acessíveis para outros domínios;
4. **Segmento de engenharia:** define a separação do DPE em nós DPE, *Ktn links* (*kernel Transportation Network*) e pilhas de protocolos necessários para a interoperabilidade entre os domínios administrativos de negócios; e
5. **Segmento de miscelânea:** define outras restrições como as limitações das especificações importadas para a especificação do ponto de referência.

2.2.6 Participantes[4]

Um conjunto inicial de componentes na arquitetura TINA foi identificado através da análise dos relacionamentos de negócios atualmente existentes nas telecomunicações e serviços de informações, como ilustrado na Figura 2.4. A seguir define-se cada um dos participantes.

[4] Pode ser um participante de qualquer tipo, empresa ou uma pessoa a qual possui um domínio (um ou mais domínios administrativos) em um sistema TINA.

a) Consumidor (*Consumer*)

Um participante com a função de negócios **consumidor** usufrui as vantagens dos serviços fornecidos por TINA. Este tipo de participante é a base econômica da TINA, uma vez que será ele quem irá pagar pela utilização dos serviços TINA. O perfil dos participantes com a função de negócios consumidor é variável, podendo ser desde grandes empresas ou corporações à usuários individuais, como por exemplo os usuários da Internet.

Os requisitos de alto nível para esta função de negócio em TINA são:

- obtenção da localização de provedores, provedores de serviços, e outros consumidores;
- registro e cancelamento de registro dos provedores;
- inicialização de relacionamentos de serviços que incluam os provedores de serviços e outros consumidores;
- sinalização de disponibilidade para os provedores (para a recepção de convites);
- aceitar convites para ingressar em sessões de outros consumidores ou provedores; e
- aceitar *downloads* de provedores para a atualização da capacidade de interação com o provedor (*upgrades*).

b) Provedor (*Retailer*)

O participante na função de negócios **provedor** provê serviços para os componentes na função de negócios **consumidor**. O número de provedores que podem estar relacionados em TINA pode variar de alguns provedores até milhares de provedores.

Um provedor pode desenvolver um novo serviço para uso imediato por qualquer consumidor em TINA, sem uma prévia autorização/padronização de outros provedores. Isto permite um rápido desenvolvimento de aplicações e faz com que TINA seja um sistema dinâmico para a integração com novas aplicações.

c) Broker

O participante na função de negócios ***broker*** possui a missão de prover informações que possibilitem que um participante encontre outros participantes ou serviços em TINA.

Em um sistema distribuído como TINA, existe a possibilidade de qualquer participante poder estabelecer um contato lógico com qualquer outro participante. A obtenção de informações e endereços de outros participantes são suportados por mecanismos genéricos do *broker*, isto é, o *broker* provê o serviço para se encontrar um objeto requisitado em TINA.

d) Provedores de Serviços Terceirizados (*Third Party Service Provider*)

O objetivo de um participante com a função de negócios **provedor de serviços terceirizados** é dar suporte em serviços para os provedores ou outros provedores de serviços terceirizados. Estes serviços podem ser considerados como "Serviços Negociáveis". O provedor de serviço terceirizado pode ser um servidor lógico, servidor de dados ou ambos.

A diferença entre o provedor de serviço terceirizado e um provedor é de que o provedor de serviço terceirizado não possui um vínculo contratual com os componentes na função de negócios consumidor. Entretanto, um participante pode se tornar um provedor ou um provedor de serviço terceirizado simultaneamente.

e) Provedor de Conectividade (*Connectivity Provider*)

O participante na função de negócio de **provedor de conectividade** gerencia a rede (switches, roteadores, links, etc). Esta rede pode ser a responsável pelo transporte que oferece suporte a ligações de fluxo ou podem constituir a rede de transporte de *kernel* para suportar ligações computacionais em TINA, provendo conexões entre nós de ambientes de processamento distribuídos.

A rede de transporte de um provedor de conectividade não é uma rede global que conecta todos os consumidores, provedores e provedores de serviços terceirizados em um sistema TINA. A rede de transporte global é segmentada em diversas sub-redes controladas por diferentes componentes. O gerenciamento de conexões de cada um destes segmentos fica a cargo de certos domínios administrativos de negócios. Para permitir a gerência das conexões roteadas através de dois ou mais segmentos de rede junto a diferentes provedores de conectividade, os provedores de conexão devem ser organizados entre si.

2.2.7 Relacionamentos de Negócios (*Business Relationship*)

Para permitir que os cinco tipos de funções de negócios anteriormente descritos possam interagir, um conjunto de tipos de relacionamentos de negócios foram definidos. Alguns relacionamentos aparecem mais que outros (Figura 2.4), denotando múltiplas ocorrências do mesmo tipo de relacionamento de negócios entre diferentes funções de negócios. Entretanto, apesar dos tipos de funções de negócios e as interações serem os mesmos, as informações enviadas podem ser completamente diferentes.

A seguir são listadas as interações que devem ser executadas antes que quaisquer outras interações sejam iniciadas:

- iniciar o diálogo entre os domínios de gerência de negócios;
- identificar os domínios administrativos de negócios entre as partes;
- estabelecer/liberar e gerenciar uma associação segura;
- descoberta de serviços e inicialização;
- estabelecimento do contexto de gerenciamento inicial; e
- negociação da utilização inicial de interações.

A seguir são descritas as interações genéricas para todos os relacionamentos de negócios, que permitem o suporte para o estabelecimento de relacionamentos gerenciáveis entre domínios de gerência de negócios.

1. **Relacionamento de Negócios do Fornecedor** (*Ret - Retailer Business relationship*): o relacionamento de negócios *Ret* é utilizado entre participantes com funções de negócios consumidor e participante*s* com funções de provedores (FARLEY,1998);
2. **Relacionamento de Negócios do Intermediário** (*Brk - Broker business relationship*): o relacionamento de negócios *Brk* provê acesso às informações de gerência controladas por qualquer outra função de Negócios TINA. O *broker* pode prover diferentes tipos de informações para diferentes funções de negócios e com finalidades diferentes;
3. **Relacionamento de Negócios Terceirizado** (*3Pty - Third Party business relationship*): um participante que possui a função de negócios provedor pode interagir com um participante que possui a função de negócios terceirizada para que possa fornecer uma grande variedade de serviços a seus consumidores sem ser realmente proprietário destes serviços;

4. **Relacionamento de Negócios Fornecedor para Fornecedor** (*RtR - Retailer to Retailer business relationship*): o relacionamento de negócios *RtR* reutiliza a funcionalidade dos relacionamentos de negócios *3Pty* e *Ret*, considerando o fato de que as informações que trafegam neste relacionamento podem ser diferentes, mas as interações não são diferentes;
5. **Relacionamento de negócios de Serviços de Conectividade** (*ConS - Connectivity service business relationship*): o relacionamento de negócios *ConS* é definido no provedor de conectividade, fornecendo os serviços de transporte de rede e as funções de negócios usando os serviços de conectividade de transporte. As conexões são estabelecidas entre pontos de acessos de redes arbitrários na camada de recursos de rede da TINA;
6. **Relacionamento de Negócios de Conexão de Terminais** (*Tcon - Terminal connection business relationship*): o relacionamento de negócios *TCon* provê a ligação de gerência entre a função de negócios do provedor de conectividade e as funções de negócios das partes envolvidas na Interface de Conexões Físicas (*Physical Connection Graph*). O relacionamento *TCon* está relacionado com o *Ret* ou com o *3Pty*, o qual executa a liberação da conexão. Uma vez que os pontos de terminação de rede (*Network Termination Points*) são dependentes da tecnologia, a implementação de interações do relacionamento *TCon* também serão dependentes da tecnologia;
7. **Relacionamento de Negócios da Camada de Redes Federativa** (*LNFed*: *Layer network federation business relationship*): o relacionamento *LNFed* é um relacionamento federativo entre provedores de conectividade; e
8. **Relacionamento da Camada de Rede Cliente-Servidor** (*CSLN*: *Client-server layer network relationship*): o relacionamento *CSLN* permite o uso de camadas entre domínios de gerência de negócios, executando funções de provedores de conectividade.

2.3 Arquitetura de Serviços TINA

A arquitetura de serviços TINA consiste de um conjunto de conceitos, fundamentos, regras e padrões para a construção, desenvolvimento e operação dos serviços TINA. O comportamento dos elementos em uma arquitetura TINA é modelado por componentes que operam em um ambiente de processamento distribuído (DPE) e

que interagem através de interfaces (KRISTIANSEN, 1997). A arquitetura de serviços TINA identifica os componentes para criar serviços e também descreve a maneira como eles serão combinados e de que maneira devem interagir. A arquitetura de serviços também examina quais componentes são necessários em um ambiente para instanciar, gerenciar e utilizar os serviços.

A Figura 2.6 apresenta a arquitetura de serviços TINA, onde mostra a combinação do modelo de negócios com o modelo computacional baseado nas especificações TINA. Pode-se observar a interação entre quatro tipos de participantes (consumidor, provedor, provedor de conectividade e o provedor terceirizado) e as estruturas necessárias em cada participante. Essa interação se dá através dos relacionamentos descritos no item 2.2.7.

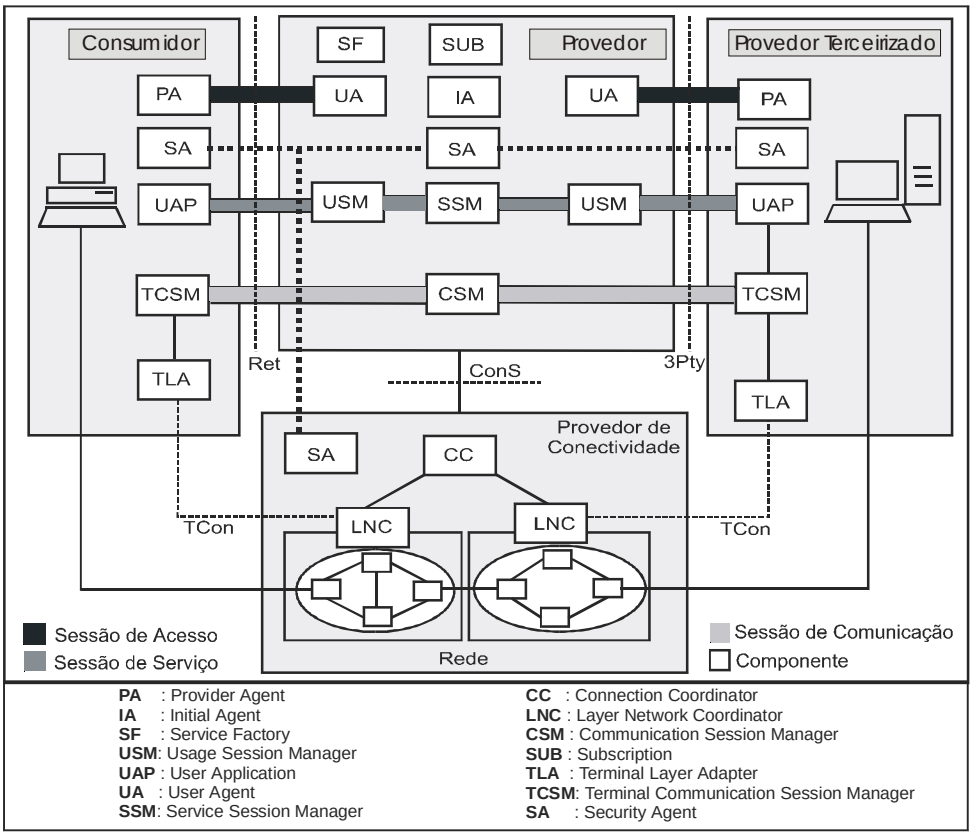


Figura 2.6 - Visão Geral da Arquitetura de Serviço (ABARCA, 1998).

2.3.1 Definição de Serviço

O termo serviço é usado em múltiplos sentidos. Uma definição tipicamente usada é que serviço é um conjunto de funções úteis oferecidos por um provedor de

serviço para um consumidor. Nas telecomunicações, um serviço pode ser visto como um conjunto “encapsulado” de capacidades disponíveis que é percebido por um usuário quando interage com uma rede de telecomunicações ou com um provedor de serviço e pela qual é realizada uma cobrança por este serviço.

Um serviço TINA é um conjunto de capacidades úteis fornecidas por um sistema existente para todas funções de negócios que o utilizem; cada função de negócios pode ver um mesmo serviço sob diferentes perspectivas.

Em TINA temos pelo menos estes três tipos de serviços (KRISTIANSEN, 1997):

- **Serviços de Telecomunicações**: são serviços baseados no transporte de bits de informações entre terminais conectados a uma rede de telecomunicações. O serviço de telecomunicações é responsável pelo estabelecimento de conexões e pelo processamento das informações relacionadas com as conexões. A rede de telecomunicações é transparente para as informações que trafegam entre os pontos da rede;
- **Serviços Gerenciáveis**: são serviços responsáveis pelo gerenciamento dos recursos TINA. Inclui funcionalidades de falha, configuração, contabilidade, performance e segurança, assim como o serviço de gerência do ciclo de vida e o serviço de gerência de instância; e
- **Serviços de Informações**: são serviços capazes de manipular informações de recursos como vídeo, audio ou documentos. Inclui o armazenamento (os dados) e a visualização (das aplicações capazes de interpretar estes recursos). Também compreende os serviços necessários entre o armazenamento e a visualização, tais como: cobrança, *buffer* e todos os serviços específicos.

2.3.2 Ambiente Comercial da Arquitetura de Serviço

O principal objetivo da arquitetura de serviço é de suportar o mais geral dos casos de interação entre domínios de gerência de negócios através de um DPE, de maneira a oferecer objetos de negócios para a captação de receita. Um exemplo deste ambiente comercial é apresentado na Figura 2.7.

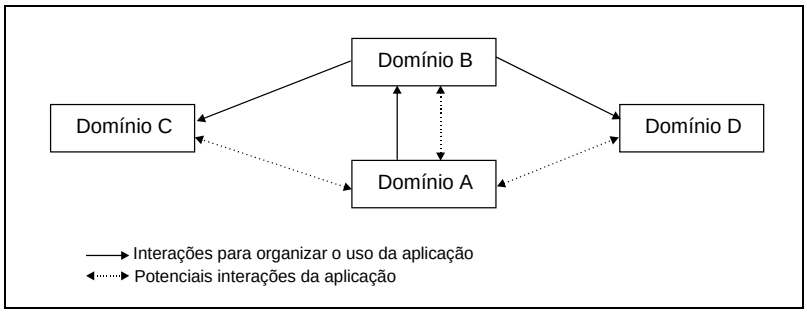


Figura 2.7 - Exemplo de Interações da Arquitetura de Serviços entre Domínios Administrativos de Negócios (KRISTIANSEN, 1997).

O domínio administrativo de negócios "A" deseja fazer uso de uma aplicação oferecida por um outro domínio administrativo "B". A aplicação pode ser totalmente fornecida por "B" ou "B" pode subcontratar esta aplicação (totalmente ou parcialmente) através de um ou vários domínios administrativos de gerência "C" e "D" (terceiros), cada um provendo contribuições únicas e diferentes para satisfazer "A". Os domínios "C" e "D" podem ou não interagir diretamente com "A". Estas interações representam os relacionamentos de negócio, onde a natureza e o ganho comercial de cada parte é único para cada aplicação e ocasião específica.

2.3.3 Divisão entre Acesso, Serviço e Comunicação

A arquitetura de serviços é dividida em dois conceitos: ***acesso*** e ***uso***. As interações necessárias para a descoberta e requisições de serviços são tratadas pelo *acesso*, enquanto que o controle do serviço e o envio do fluxo de conteúdo são tratados pelo *uso*. A Figura 2.8 ilustra a divisão da arquitetura.

a) O acesso

O *acesso* trata das interações necessárias para se estabelecer a comunicação entre os dois domínios. No exemplo da Figura 2.8, "B" armazena informações sobre "A", como autorizações e preferências já conhecidas. A funcionalidade de *acesso* é definida de maneira simples para permitir a fácil implementação em equipamentos TINA para o mercado. Todas as outras interações são manipuladas como *uso*.

b) O uso

O uso divide-se em:

- **Serviço**: as interações entre os componentes são necessárias para o controle do comportamento do serviço, como a troca de dados (conteúdo) e a manipulação de informações de gerenciamento; e
- **Comunicação**: as interações de comunicação são necessárias para estabelecer e manter conexões entre componentes que implementam o serviço. Também são tratadas a negociação de QoS, e a configuração e modificação de conexões.

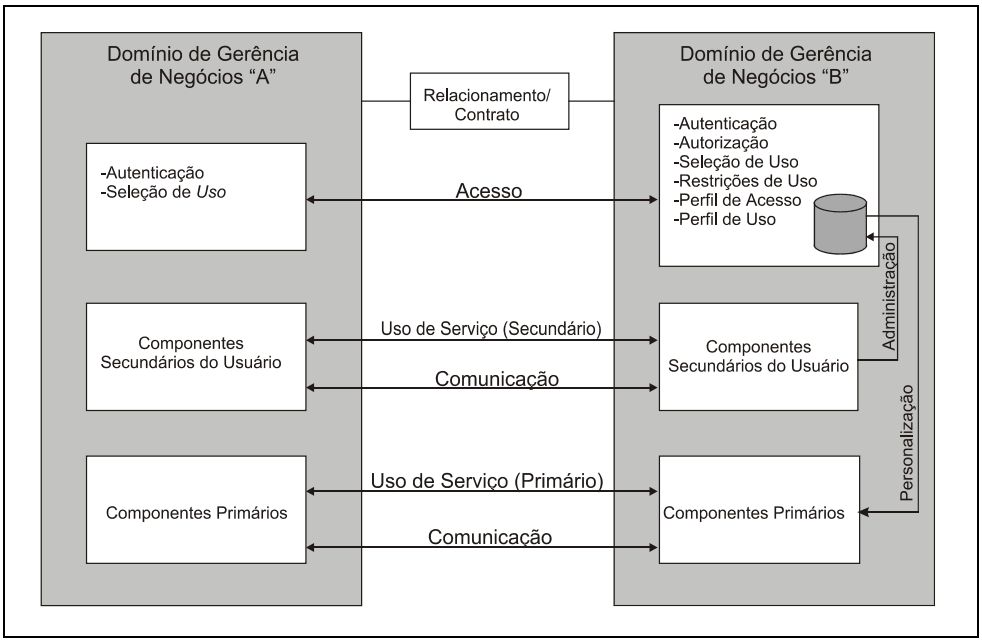


Figura 2.8 - Acesso, Uso Primário e Uso secundário (KRISTIANSEN, 1997).

c) Uso de serviço secundário

O serviço secundário é composto por interações de componentes desenvolvidos pelo provedor e que permitem ao cliente personalizar a apresentação e a utilização de componentes do provedor. Os limites do contrato do cliente são previamente estabelecidos entre as partes. Apesar dos serviços secundários não estarem implicitamente incluídos no contrato, eles agregam valores ao serviço primário.

d) Uso de serviço primário

O serviço primário trata de satisfazer o objetivo primário do contrato entre os dois domínios, como por exemplo, uma conferência multimídia, a provisão de informações de um recurso ou um serviço de gerenciamento.

2.4 Sessões

Em TINA uma sessão é definida como:

- um relacionamento temporário entre um grupo de objetos associados coletivamente para executar uma tarefa durante um período de tempo;
- uma sessão possui um estado que pode mudar durante o seu tempo de vida;
- uma sessão representa uma visão abstrata, simplificada do gerenciamento, de uso de objetos e de suas informações compartilhadas;
- os objetos em uma sessão estão sujeitos a políticas comuns que controlam a sessão, alguns objetos podem estar sujeitos a aspectos derivados destas políticas;
- uma sessão pode abranger múltiplos domínios de gerência de negócios; e
- o domínio da sessão define a política de um determinado domínio, objetos computacionais e informações que estão sujeitas a tal domínio de sessão.

Na Figura 2.9 é representado o escopo das sessões em TINA. Na figura é representado um exemplo onde o consumidor interage com outro participante em uma sessão de serviço oferecida por um provedor, cujas funções principais são:

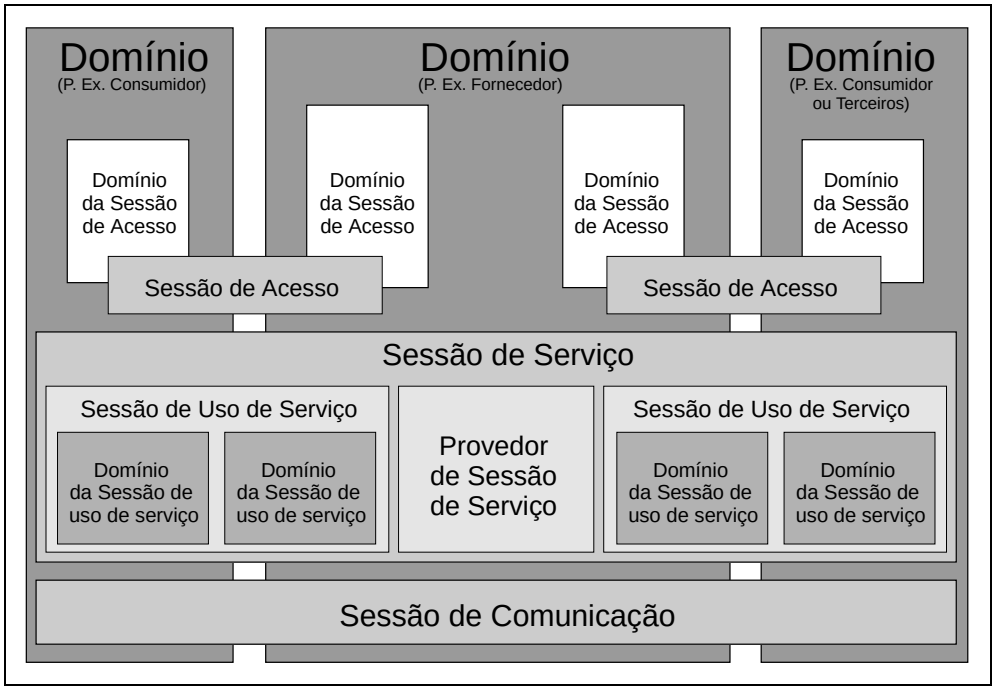


Figura 2.9 - Exemplo de Sessões TINA (KRISTIANSEN, 1997).

a) Sessão de Acesso

A sessão de acesso é estabelecida quando dois domínios da sessão são interconectados em um relacionamento seguro. O próximo estágio da sessão de acesso é a negociação dos termos entre os domínios para que se continue a interação e a autenticação. A partir de uma sessão de acesso muitas sessões de serviço podem ser invocadas, a qual é responsabilidade da sessão de acesso gerenciar estas sessões até que a sessão de serviço termine, no entanto, as demais seguem associadas à sessão de acesso (KRISTIANSEN, 1997).

b) Sessão de Serviço

Existem dois tipos de sessões de serviço:

- **Sessão de Serviço**: representa as informações e funcionalidades relacionadas aos serviços de execução, de controle e de gerência. Estes serviços incluem serviços primários (ex. conferência multimídia) e serviços secundários (ex. assinatura de um serviço on-line).
- **Provedor de Sessão de Serviço**: representa o centro do serviço lógico e de controle de um ou mais domínios que estejam participando do serviço.

c) Sessão de Comunicação

A sessão de comunicação representa uma visão geral do serviço de conexão e da rede, independente da tecnologia e dos recursos necessários para se estabelecer uma conexão fim-a-fim. Uma sessão de comunicação pode manipular múltiplas conexões. Estas conexões podem ser multiponto (videoconferência) e/ou multimídia (imagem e som).

A adoção do conceito de “sessão” para o controle das comunicações possui a vantagem de permitir que os serviços sejam instanciados dinamicamente e manipulados para manter uma configuração adequada dos recursos de comunicações desejados.

2.5 Gerenciamento de Serviço

O gerenciamento de serviço é uma parte importante da arquitetura de serviço TINA. O gerenciamento de serviço envolve funções de gerenciamento e controle das transações fim-a-fim dos serviços TINA, as quais são fornecidas pelas propriedades

FCAPS ditadas pelo contexto de gerenciamento da conexão. Os quatro conceitos associados com o gerenciamento de serviço são detalhados a seguir.

2.5.1 Conceito de Particionamento

O conceito de particionamento considera o aspecto de como o gerenciamento pode ser dividido em problemas de uma proporção mais gerenciável. TINA suporta dois tipos de particionamento: de camadas e de domínios. Estes dois conceitos são independentes, já que enquanto a camada é tratada como um particionamento "horizontal", o domínio é tratado como um particionamento "vertical".

- **Camada:** corresponde à diferença entre o gerenciamento de recursos e o gerenciamento de serviço. Os recursos são capacidades e equipamentos de redes. O gerenciamento de recurso define o gerenciamento dos recursos de rede. O gerenciamento de serviço relaciona-se com todas as atividades de gerenciamento dentro da camada de serviço.
- **Domínio:** a arquitetura TINA não supõe que um simples provedor de serviço seja o responsável para fornecer todos os serviços, muito pelo contrário, é assumido que irá existir um número considerável de participantes atuando em diversas funções de negócios. Participantes são na realidade entidades comerciais que provêem serviços ou recursos de comunicações. Para suportar um ambiente de múltiplos participantes, a arquitetura TINA reconhece o conceito de domínios. Estes podem ser domínios de gerência que são controlados por vários componentes. Com um domínio administrativo, vários domínios de gerência podem existir para auxiliar na organização de atividades de gerência necessárias (HAMADA, 1997, KRISTIANSEN, 1997).

2.5.2 Conceitos Funcionais

O escopo funcional considera as capacidades que o gerenciamento de serviço pode assumir. TINA abrange as áreas de gerenciamento FCAPS, a seguir descritas (KRISTIANSEN, 1997):

- **Gerência de Falhas**: trata de alarmes de falhas em geral, e também a correlação e distribuição de alarmes. Este tipo de gerenciamento é importante para que TINA seja confiável e tolerante a falhas, uma vez que consiste na maior parte de

elementos distribuídos semi-autônomos. Esta mesma natureza distribuída dificulta muito o controle e a gerência de falhas no sistema.

- **Gerência de Configuração**: trata da configuração de recursos, tais como: recursos de rede, recursos computacionais, recursos de software, etc. De modo que estes recursos configurados se tornem disponíveis para os serviços TINA. Como os tipos de recursos em TINA são diversificados, a gerência de configuração em TINA possui uma coleção de esquemas de gerenciamento para diferentes tipos de recursos.
- **Gerência de Contabilidade**: trata de questões como cobrança, objetos contábeis, e o envio de eventos de contabilidade. TINA possui opções flexíveis de contabilidade, como a cobrança on-line e cobrança por terceiros.
- **Gerência de Desempenho**: trata da monitoração, controle e administração da desempenho em TINA.
- **Gerência de Segurança**: trata de questões de segurança em nível de serviço, e subseqüentemente com todos os componentes que se situam no nível de recursos. A gerência de segurança TINA trata de questões como autenticação e autorização em um ambiente de domínio multi-provedor. A fim de estabelecer uma sessão de serviço em um ambiente multi-provedor, uma rede de confiança precisa ser mantida e gerenciada apropriadamente entre os domínios.

A arquitetura de gerenciamento TINA usa o c**ontexto de gerenciamento** e **transações de serviço** para suportar o gerenciamento FCAPS. O contexto de gerenciamento é um contrato de gerenciamento (serviço) entre o usuário e o provedor, o qual conduz atividades de gerenciamento (serviços) no domínio do provedor. A transação de serviço é construída para garantir a integridade da sessão de serviço com respeito a seu gerenciamento FCAPS. Isto é similar ao conceito de transação de base de dados, embora o propósito da transação de serviço é gerenciar a integridade da sessão de serviço, e não a integridade dos dados (HAMADA, 1997).

2.5.3 Conceitos computacionais

A arquitetura TINA adotou o conceito de *viewpoints* ODP, incluindo os *viewpoints* computacionais e de informações. Os *viewpoints* computacionais consideram um sistema como um conjunto de objetos que interagem e que podem ou não ser

distribuídos. Como TINA é baseado em uma arquitetura DPE, o aspecto computacional é muito importante. Ele provê a base para a estrutura do sistema gerenciado e de sistemas de gerência (KRISTIANSEN, 1997).

Um DPE oferece uma grande variedade de serviços de suporte para se manter um sistema de gerenciamento. Isto inclui vários serviços de gerenciamento de ciclo de vida que suportam: a criação e destruição de objetos; serviços de segurança como criptografia, autenticação e autorização; e outras funcionalidades como gerenciamento de eventos e *log*.

2.5.4 Conceitos de Ciclo de Vida

O conceito de ciclo de vida relaciona-se com o ciclo de vida do serviço e do usuário. O gerenciamento do ciclo de vida de serviço considera como os provedores podem desenvolver, disponibilizar e controlar a criação e eliminação de serviços. O gerenciamento de ciclo de vida de serviço TINA constrói software com ênfases no desenvolvimento de serviços em um ambiente orientado a objeto distribuído através de múltiplos domínios. O objetivo de TINA sobre as necessidades do usuário requer examinar o suporte disponível do usuário através *framework* TINA. O gerenciamento do ciclo de vida do usuário considera como os usuários são incluídos e suportados por um provedor (HAMADA, 1997).

2.6 Conclusão

Neste capítulo foi apresentada com detalhes a arquitetura TINA, a qual é uma arquitetura aberta para uma grande variedade de serviços de telecomunicações. Pode-se concluir que:

- A arquitetura e as especificações DPE podem ser usadas para especificar "*middleware*" - o software que provê o suporte para executar aplicações em um ambiente de computação distribuída;
- O modelo de negócios descreve vários papéis que ocorrem em uma operação de telecomunicações. A tendência através da desregulamentação e concorrência global, e que cada papel pode ter uma área de negócio separada e com muitos participantes. Os pontos de referência TINA especificam interfaces entre os vários papéis do modelo de negócio;

- TINA impõe um alto grau de controle e flexibilidade de usuário na camada de serviço. TINA também dá grande ênfase aos requerimentos de ambiente multi-provedor na camada de serviço e de recurso. A arquitetura de gerenciamento TINA permite satisfazer requerimentos de gerenciamento de serviços e recursos em um ambiente distribuído multi-participantes; e
- TINA usa o conceito de domínios para o suporte de múltiplos provedores e introduz políticas de gerenciamento como um meio de controle no gerenciamento de domínio.

Portanto, depois do estudo do escopo geral da arquitetura de gerenciamento de TINA, através das especificações do TINA-C, explanar-se em detalhes no seguinte capítulo o contexto da Arquitetura de Contabilidade TINA.

3. ARQUITETURA DE CONTABILIADADE TINA

3.1 Introdução

A disponibilidade e qualidade de serviços de telecomunicações são cada vez mais importantes, com a automação de negócios e a grande confiança em aplicações baseadas em computador. Na área de gerenciamento, o objetivo de todos os atores de telecomunicações é a rápida, cuidadosa e confiável troca de informações entre consumidores e provedores de serviços, para minimizar o tempo de resolução e para otimizar a satisfação dos consumidores em caso de violações de SLA.

Todos os provedores (incluindo o provedor de conectividade) oferecem serviços como serviços TINA. A informação é trocada entre os pontos de referência TINA, fazendo uso da tecnologia CORBA ou TMN. O uso do conceito de subscrição em TINA permite a conectividade, além de acrescentar valores aos recursos adicionais associados aos serviços, onde consumidores e/ou usuários da sessão serão afetados. A integração de sistemas de relatórios TINA e o gerenciamento de contabilidade disponibilizam os descontos em caso de violações de SLA (AGOULMINE, 2000).

O objetivo da arquitetura de contabilidade TINA é prover uma contabilidade flexível e confiável, com a adição de funcionalidades que não existem nas arquiteturas de contabilidade tradicionais. A gerência de contabilidade consiste de quatro ciclos, os quais são apresentados na Figura 3.1.

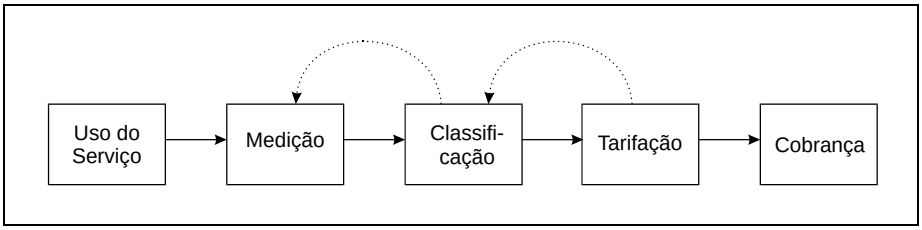


Figura 3.1 - Ciclo Básico da Contabilidade (HAMADA, 1996).

a) **Medição**: trata da monitoração e do registro dos recursos utilizados. Devido à natureza distribuída de TINA, este tipo de tarefa enfrenta muitas dificuldades, pois os objetos e recursos encontram-se distribuídos e muitas vezes migram de um local para outro. A medição é o primeiro passo e a base de todas as atividades da contabilidade;

b) **Classificação**: classifica as informações da medição em um conjunto de classes baseado na utilização dos serviços, dos recursos que estão sendo utilizados, da distância entre o usuário e o provedor, etc. A finalidade da classificação é categorizar e reduzir a quantidade de informações da medição;
c) **Tarifação**: nesta etapa é calculado o valor a ser cobrado baseado nas informações obtidas pelo ciclo da classificação. A estrutura tarifária é representada por uma tabela com os custos de cada categoria de serviço. Esta tabela pode ser alterada quando necessário e normalmente estas alterações dependem do provedor dos serviços; e
d) **Cobrança**: trata do processo de armazenamento das informações de cobranças e o envio destas cobranças ao consumidor (a entidade a qual o serviço está sendo entregue). O intervalo da cobrança pode ser mensal, diário, por horas ou por um intervalo que depende do acordo negociado entre o consumidor e o provedor de serviços.

3.2 Obstáculos no Contexto de Contabilidade TINA

A contabilidade em TINA é composta por complexas atividades dos objetos distribuídos, na qual uma arquitetura computacional e de engenharia ainda devem ser estudadas para garantir a flexibilidade e a confiabilidade. A seguir serão citados alguns dos problemas já identificados com relação à contabilidade TINA (HAMADA, 1996):

- **Contabilidade como parte do serviço de gerenciamento FCAPS**: refere-se às estruturas computacionais e de engenharia necessárias para a solução de problemas em nível de recursos do serviço de gerenciamento FCAPS. Por exemplo, a contabilidade (em particular, a medição) e a segurança (em particular, a auditoria) compartilham interesses comuns nas atividades de um objeto;
- **Contabilidade distribuída**: refere-se ao fato de que as entidades de contabilidade em TINA como o gerenciador de medição, cobrança, etc., são distribuídos, assim como os objetos da contabilidade. Um recurso pode ser compartilhado entre diferentes serviços e diferentes gerenciadores de medição. Neste contexto de gerenciamento distribuído, também é considerado como garantir que as informações de eventos sejam transformadas em informações de

medição consistentes. Esta natureza distribuída ainda impõe outros problemas, como por exemplo, a medição que é possivelmente executada pelo Servidor de Notificações do DPE ou pelo mecanismo de *EventReport/Notification* (ITU, 1993) do TMN, ou por algum outro mecanismo correspondente. Neste caso, informações de eventos não possuem garantia de entregas, podendo acontecer que estas informações se percam e as informações da medição resultante não sejam completas;

- **Aspectos dinâmicos da contabilidade**: refere-se à situação em que as estruturas tarifárias e categorias de classificação estão sujeitas a alterações devido às mudanças das necessidades do mercado, devido ao desenvolvimento de novos serviços e devido à remoção de serviços existentes. Adicionalmente, o ciclo da medição é dependente do serviço e da sua respectiva estrutura tarifária; e
- **Associação de funções flexíveis na contabilidade**: tradicionalmente existe uma diferença entre quem é o fornecedor de serviço e quem é o usuário. Este conceito deve ser reexaminado e redefinido para que suporte o conceito cliente-servidor distribuído de TINA. Por exemplo, no conceito em camadas TINA, um provedor de serviço pode ser um usuário (cliente) de outro provedor de serviço.

3.3 Arquitetura de Contabilidade Básica

Os problemas de contabilidade em TINA são associados aos conceitos do Contexto de Gerenciamento de Contabilidade (*AcctMgmtCtxt – Accounting Management Context*) e da Transação de Serviço. O propósito do *AcctMgmtCtxt* é de garantir que a contabilidade seja preservada através das atividades de um conjunto de objetos distribuídos, as quais constituem um serviço. É necessário enfatizar que a contabilidade não é uma propriedade ou um atributo de um simples objeto, mas sim um conjunto de grandezas medidas (ou calculadas) sobre as atividades dos objetos distribuídos durante o serviço (HAMADA, 1996).

Na Figura 3.2 é ilustrado o modelo de informações da Transação de Serviço e do Contexto de Gerenciamento de Contabilidade.

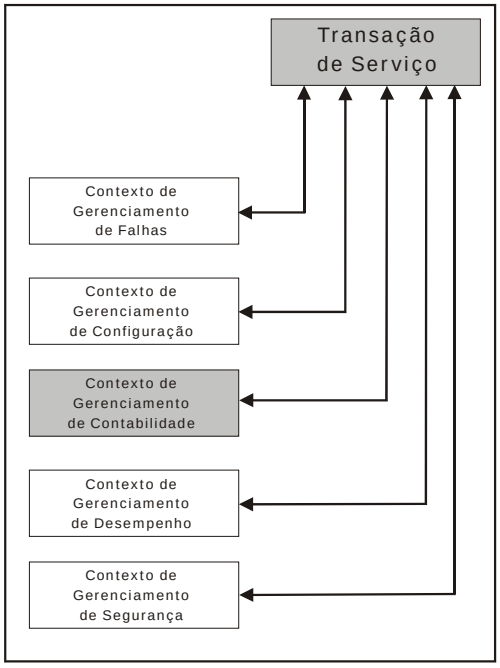


Figura 3.2 - Transação de Serviço e o Contexto de Gerenciamento FCAPS.

Cada componente do Contexto de Gerenciamento de Serviço FCAPS especifica os detalhes da qualidade e quantidade dos serviços de gerenciamento. O Contexto de Gerenciamento de Contabilidade particularmente especifica os 5Ws (o que, como, quando, quem e onde – *What, hoW, When, Who, Where*) da Gerência de Contabilidade. Quando uma Transação de Serviço é ativada pelo SSM (*Service Session Manager*), seu *AcctMgmtCtxt* é interpretado de acordo com sua Descrição de Componente de Sessão (unidade de serviço a ser contabilizado) e com a estrutura de tarifa dentro do domínio SSM. O SSM repassa o controle e os parâmetros necessários para os mecanismos em nível dos recursos computacionais, tais como, o CSM (*Communication Session Manager*), o *Notification Server* e os gerenciadores de medição.

A associação dos conceitos de Transação de Serviço e do *AcctMgmtCtxt* possui as seguintes conseqüências no processo de contabilidade TINA:

- **Contabilidade garantida**: a Transação de Serviço deve oferecer mecanismos que garantam a integridade do serviço;
- **Controle do processo de contabilidade flexível**: implica que o processo de contabilidade pode ser facilmente adaptado aos diferentes domínios através de mecanismos adequados de interpretação. Como a interpretação do *AcctMgmtCtxt* adiciona um maior nível de flexibilidade, pode-se dizer que o processo é

facilmente adequado às mudanças nas estruturas tarifárias ou no ambiente do serviço;

- **Contabilidade eficiente**: os parâmetros de controle da contabilidade em nível dos recursos, como do ciclo *Event Report*, podem ser otimizados na interpretação do *AcctMgmtCtxt* no SSM. Isto é possível devido à estrutura das informações da medição. A freqüência da medição deve ser determinada a partir das especificações do componente de sessão e do ambiente do domínio de serviço;
- **Contabilidade distribuída**: a tarefa da contabilidade pode ser dividida entre processos concorrentes ou dividida entre o usuário e o provedor do serviço. Em TINA, o processo de contabilidade utilizando o *AcctMgmtCtxt* possibilita que, com mecanismos de segurança apropriados, a tarefa da contabilidade seja dividida entre o usuário e o provedor de serviços, transferindo parte da responsabilidade de contabilidade para o usuário;
- **Contabilidade confiável**: as informações da contabilidade devem ser confiáveis. A necessidade da confiabilidade parte da natureza distribuída de TINA. Eventos podem ser perdidos devido aos congestionamentos temporários da rede ou informações da medição podem ser perdidas devido a uma falha no nó. Pode-se esperar que algumas destas questões sejam tratadas pela gerência de falhas, mas a consistência e a confiabilidade das informações de contabilidade devem ser tratadas pela própria gerência de contabilidade;
- **Contabilidade confiante**: as informações de contabilidade devem ser verídicas. Esta necessidade vem da natureza aberta de TINA. Um usuário e um provedor de serviços, totalmente desconhecidos entre si, podem se conectar utilizando algum tipo de catálogo eletrônico (*Yellow page*). Como o usuário pode confiar no provedor de serviços e como o provedor de serviços pode confiar no usuário? A primeira questão está mais relacionada com a gerência de contabilidade, enquanto a segunda está mais relacionada com a gerência de segurança. Informações de contabilidade devem ser confiáveis e gravadas em ambos os lados (opcionalmente) de uma maneira inalterável e irrefutável; e
- **Integridade das informações de contabilidade**: a integridade das informações da contabilidade deve ser preservada apesar de possíveis falhas na rede,

interrupções de serviços e do encaminhamento do serviço através de diferentes domínios de gerência.

3.4 Transação de Serviço (*Service Transaction*)

Nesta seção o conceito de transação de serviço é explanado. A transação de serviço é composta por três fases muito importantes para executar uma sessão (HAMADA, 1996):

a) **Fase *Setup***: o usuário apresenta seu esquema de contabilidade e o provedor de serviços também apresenta seu esquema de contabilidade (é possível o caso de negociação entre as partes). Quando os dois esquemas fornecidos estiverem compatíveis, um *AcctMgmtCtxt* consistente pode ser formado. O esquema negociado é submetido ao *AcctMgmtCtxt* da sua respectiva Transação de Serviço. O *AcctMgmtCtxt* é interpretado através da leitura das especificações do componente de sessão e do ambiente de contabilidade do domínio. Os mecanismos em nível de recursos são configurados de maneira a seguir as interpretações;

b) **Fase *Execution***: o serviço é oferecido ao usuário de acordo com as especificações da Transação de Serviço. Informações da medição são acumuladas nos respectivos componentes em nível de recursos (registros da conta do usuário, gerenciador da medição, registros da conta do provedor de serviços, entre outros), qual é especificada como parte do *AcctMgmtCtxt*; e

c) **Fase *Wrap-up***: o serviço é concluído e as informações da medição em localizações possivelmente distribuídas são coletadas e sumarizadas. Informações de cobrança podem ser enviadas ao usuário na conclusão da transação. A transação é concluída com sucesso, se as informações de contabilidade estão de acordo com o *AcctMgmtCtxt*, ou consideradas incompletas se não estiverem de acordo com o *AcctMgmtCtxt*. Neste último caso, ações corretivas podem ser tomadas.

3.5 Aninhamento de Transações de Serviços

Como um serviço pode ser encaminhado por agentes que não são controlados diretamente pelo usuário ou como um serviço pode se estender sobre diversos domínios de gerenciamento, é necessário que ao menos uma parte dos termos de contabilidade do

AcctMgmtCtxt do usuário sejam encaminhados juntamente com as atividades distribuídas iniciadas pelo usuário. Por exemplo, um usuário pode ativar um serviço encaminhado por um agente (por exemplo, *QoS broker*), que por sua vez usa um outro agente para executar certas funções (por exemplo, *bandwidth broker*), e assim por diante. Um conjunto de agentes pode ser ativado em cascata, onde o fim da corrente executa alguma função relacionada com o contexto de gerenciamento de serviço inicialmente configurado entre o usuário e o provedor de serviços (HAMADA, 1996).

O propósito do conceito de transação de serviço é oferecer um contexto de gerenciamento de serviço consistente através das atividades distribuídas dos objetos. A Figura 3.3 ilustra um serviço estendido por diversos domínios de gerência de serviço. O usuário submete um serviço dentro do domínio "A" através do GSEP1 (*Generic Session End Point*). Com o serviço encaminhado, ele pode eventualmente ativar um serviço em um domínio diferente (no caso, domínio "B") através do GSEP2. O usuário pode ou não saber da existência do GSEP2 ou de que um serviço do domínio "B" está sendo acessado pelo serviço que ele submeteu.

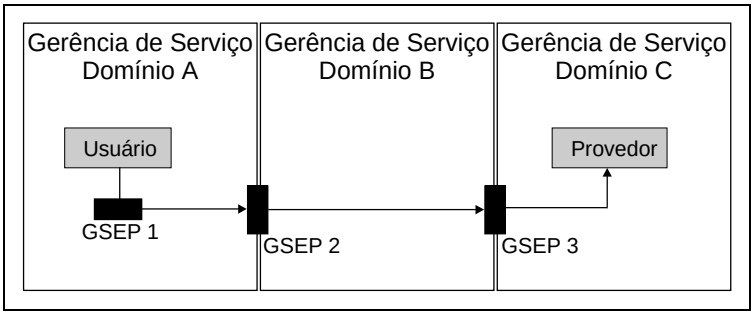


Figura 3.3 - Serviço Através de Múltiplos Domínios de Gerência de Serviço (HAMADA, 1996).

O serviço no Domínio "B" pode ser encaminhado por um SSM, ou pode ser encaminhado por um conjunto de agentes que atuam de maneira relativamente independente. Ele pode eventualmente alcançar GSEP3, que é o ponto de entrada para o Domínio "C".

Através do GSEP3, o provedor oferece um serviço. O provedor em si pode ou não estar visível a partir do Domínio "B", da mesma maneira que o provedor também pode ou não estar visível para o usuário.

Utilizando o conceito de aninhamento da transação de serviço a Figura 3.3 pode ser reestruturada como ilustra a Figura 3.4.

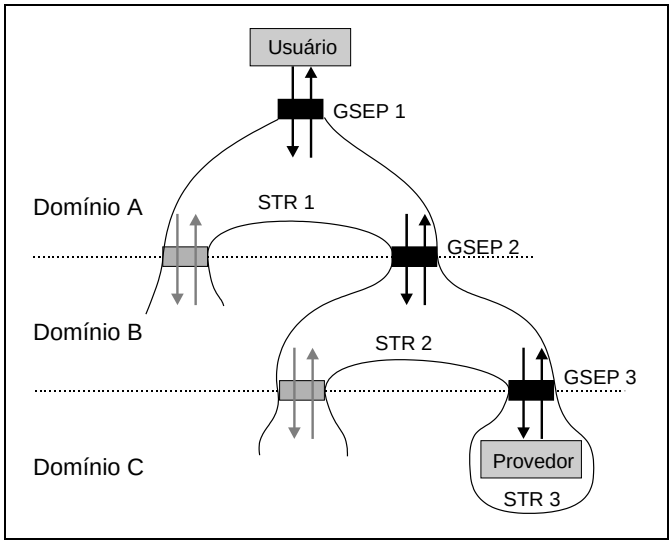


Figura 3.4 - Estrutura de Aninhamento da Transação de Serviço (HAMADA, 1996).

As curvas sólidas na Figura 3.4 representam os limites do escopo das transações de serviço. O escopo da transação de serviço geralmente coincide com seu respectivo domínio de gerência de serviço. Uma passagem por um domínio diferente (GSEP1, GSEP2) requer uma nova transação de serviço para iniciar um ponto de entrada (*entry point*).

O conceito da resolução de escopo no contexto de gerenciamento de serviço serve para se determinar a qual contexto a transação aninhada pertence antes que a transação inicie sua execução. Para explanar o problema do escopo, a seguir são dadas algumas propriedades do aninhamento de transações de serviço (utilizando a Figura 3.4 como exemplo):

- **A transação raiz (STR1) é concluída somente quando todas as transações aninhadas (STR2, STR3) concluírem suas funções**. Isto não quer dizer necessariamente que todas as transações aninhadas devem ser concluídas com sucesso. Por exemplo, se o esquema de *back-up* parcial está aplicado na transação raiz e se algumas das transações aninhadas falhar, o usuário pode ser capaz de utilizar apenas os resultados bem sucedidos e a transação raiz pode ser concluída com sucesso parcial.

- **Se a transação raiz abortar, todas as transações aninhadas devem abortar**. Este caso ocorre quando somente uma das transações aninhadas falhar e o esquema de *back-up all abort* é utilizado. Isto quer dizer que quando o *AcctMgmtCtxt* não pode se tornar visível, todo o serviço deve ser abortado.

3.6 Contexto do Gerenciamento da Contabilidade (*AcctMgmtCtxt*)

O *AcctMgmtCtxt* é um bloco de informações e parte fundamental da contabilidade TINA, a qual especifica o 5W (*What, hoW, When, Who, Where*) do ciclo básico da contabilidade. A Figura 3.5 ilustra o modelo do *AcctMgmtCtxt* (HAMADA, 1996):

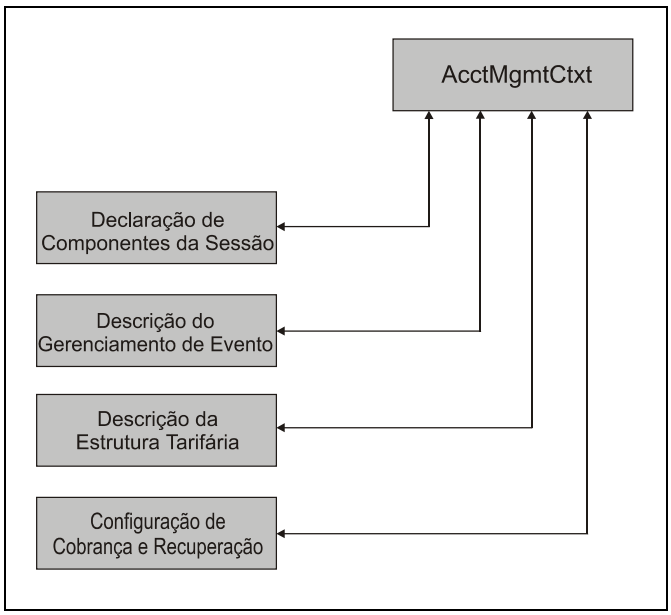


Figura 3.5 – Modelo do *AcctMgmtCtxt*.

a) **Declaração dos Componentes da Sessão**: esta parte define quais componentes de sessão serão usados ou se um componente de sessão específico será possivelmente usado na transação de serviço especificado pelo *AcctMgmtCtxt*. A função desta parte é declarar quais componentes de sessões estão disponíveis ou visíveis ao usuário, como parte distinguível do serviço, possibilitando que o usuário especifique ou negocie o gerenciamento de eventos, que será descrito a seguir na descrição de gerenciamento de eventos. A declaração dos componentes de sessão pode ser omitida caso ele já tenha herdado informações da transação de serviços. É preciso que o componente de sessão seja uma parte distinguível

do serviço, onde tanto o usuário quanto os provedores de serviços sejam capazes de identificar a sua existência e diferenciá-lo de outros componentes. Pode-se citar como exemplo, uma típica ligação telefônica a qual consiste das seguintes fases: fase-discar, fase-chamar, fase-falar, fase-desligar, fase-ocupado, entre outros. Estas fases são partes distintas do serviço e podem ser consideradas como componentes de sessão;

b) **Gerenciamento de Eventos**: esta parte especifica os 5Ws dos eventos gerados ou associados a cada componente de sessão. Por exemplo, no início de cada componente de sessão, o usuário pode desejar ser notificado ou o provedor de serviços pode necessitar de uma confirmação do usuário. Diferentes eventos podem ser gerados para cada componente de sessão ou agrupados em classes para simplificar as ações entre o usuário e o provedor de serviços. Eventos e configurações pré-definidas são derivados de outras partes do *AcctMgmtCtxt*, possibilitando desta maneira uma contabilidade com especificações pré-definidas sem que as descrições de gerenciamento de eventos sejam alteradas pelo usuário ou pelo provedor. Os eventos e ações aqui mencionados estão relacionados apenas com a contabilidade, ou seja, não estão relacionados com as atividades da sessão de serviço estabelecida entre o usuário e o provedor;

c) **Estrutura de Descrição da Tarifa**: esta parte especifica a estrutura tarifária negociada entre o usuário e o provedor. Esta estrutura tarifária proporciona ao *AcctMgmtCtxt* uma maior flexibilidade na tarifação, permitindo uma tarifação por usuário, por serviço ou por sessão. Se este tipo de flexibilidade não for necessária, a estrutura tarifária pode ser simplesmente herdada do contrato de assinatura (*subscription*) negociado entre o usuário e o provedor ou herdada da estrutura tarifária pré-definida do domínio de gerência de serviço; e

d) **Cobrança e Configuração de Recuperação**: esta parte especifica os 5Ws do ciclo de cobrança (*billing*) e das opções de recuperação, caso a transação de serviço não satisfaça os requisitos da contabilidade. Cobrança e recuperação são assuntos diferentes, mas eles estão combinados dentro de um bloco na especificação do *AcctMgmtCtxt*, pois eles possuem certas similaridades interessantes do ponto de vista do usuário. Por exemplo, a configuração da cobrança especifica “quem” (quem faz a chamada, cobrança de terceiros,

cobrança compartilhada, entre outros) e "quando" (mensal, semanal, cobrança on-line, entre outros.) do ciclo de cobrança. Como opções de recuperação existem a opção do esquema de *back-up* parcial e o esquema de *back-up all abort*.

3.7 Um Cenário de Serviço de Contabilidade Aplicável ao Contexto de Gerenciamento de Contabilidade TINA

A crescente complexidade dos serviços de telemática[5] (*telematics)* fornecidos atualmente freqüentemente relaciona a participação de muitos atores em uma simples sessão de serviço. Além disso, o desenvolvimento atual de negócios indica uma forte convergência de processos dos provedores de serviço através da integração de serviços sob o intermediador de serviço (*service brokers*). No estabelecimento de uma sessão, diferentes componentes de serviço fornecidos por diferentes atores têm que cooperar em ordem para estabelecerem a sessão de serviço fim-a-fim para o usuário. Uma forma segura de fornecer serviço fim-a-fim em um ambiente multi-provedor é aplicar um gerenciamento de sessão de serviço integral (VAN LE, 2002).

Uma arquitetura de serviço simples, por exemplo, pode consistir de um usuário e três provedores de serviço localizados em três diferentes domínios administrativos, isto é: um Intermediador de Serviço (SB – *Service Broker*), um Operador de Rede (NO – *Network Operator*) e um Provedor de dados (ou conteúdo) (CPr – *Content Provider*). Supõe-se que a subscrição do usuário é paga antecipadamente e fornecida por SB, de forma que, o SB assegura a relação comercial com o usuário. Isto implica o seguinte:

- O SB assegura a relação de negócios com NO e CPr (esta relação pode ser a longo e/ou curto prazo);
- O SB é responsável pelo correto fornecimento fim-a-fim do dado (conteúdo) solicitado e da correspondente transmissão QoS;
- O SB é responsável pelo monitoramento e atualização do crédito do usuário.
- O NO é responsável pela qualidade do dado transmitido desde CPr para o usuário final; e
- O CPr é responsável pelo fornecimento dos dados (conteúdo) solicitados.

[5] Interação de processamento de dados com dispositivos de comunicação, por exemplo, computadores, redes, etc.

Este cenário trata diretamente com questões relacionadas com o gerenciamento de contabilidade em tempo real. Isto se deve a:

i. quando se assume que cada provedor de serviço é responsável pelo gerenciamento de contabilidade dos componentes de serviço fornecido em seu domínio, significa definir uma arquitetura de gerenciamento de contabilidade local que deva disponibilizar a contabilidade para cada componente de serviço contábil na sessão; e

ii. do momento que SB é responsável por todo o gerenciamento de contabilidade da sessão de serviço, a troca de informação contábil entre os atores envolvidos deve ser feita de modo que o gerenciamento de contabilidade em tempo real deva ser realizado.

Para tratar com a complexidade do gerenciamento de contabilidade em tempo real, segundo (VAN LE, 2002), devem executar-se três passos:

1. definir uma arquitetura de contabilidade genérica que suporte contabilidade multi-provedor em tempo real. Este passo define os componentes necessários do sistema, como exemplo, a arquitetura de contabilidade TINA;
2. aplicar um modelo estrutural e procedimental para a arquitetura de contabilidade definida, como exemplo, uso do UML (*Unified Modeling Language*) (LARMAN, 2000) para descrever a estrutura da arquitetura, a qual expressa o relacionamento entre as entidades envolvidas e a forma como as entidades interatuam. Também, o uso da linguagem de especificação formal LOTOS (*Language of Temporal Ordering Specification*) (BRINKSMA, 1988) para descrever o comportamento das entidades individuais. Esta linguagem permite descrever o comportamento dos componentes individualmente, assim como também o comportamento de todas as partes do sistema como um todo; e
3. verificar a precisão do modelo para garantir a correta interação entre os componentes do sistema. Desse modo, as propriedades de contabilidade são verificadas para assegurar o bom funcionamento das propriedades de contabilidade cruciais, tais como: o usuário não deve pagar mais que o fornecido pelo serviço do provedor; um fornecimento de serviço deve parar quando o crédito do usuário alcance o valor zero, etc.

Por último, a propriedade de tempo real de um processo de contabilidade requer o processamento, a troca e a atualização da informação contábil em tempo real, durante todo fornecimento do serviço do provedor. Nesse contexto, pode-se concluir que estes passos são compatíveis e aplicáveis aos conceitos relacionados ao contexto de gerenciamento de contabilidade TINA, portanto, permitirão definir, verificar e validar um modelo de contabilidade que é o objetivo deste trabalho.

3.8. Conclusão

TINA apresenta uma arquitetura de gerenciamento de contabilidade que permite prover uma contabilidade flexível e confiável. É usada para denotar os quatro ciclos básicos da contabilidade: medição, classificação, tarifação e cobrança. Estes ciclos estão baseados na transação de serviço e no conceito do contexto de gerenciamento de contabilidade (*AcctMgmtCtxt*). Estes conceitos podem ser aplicados em diversas classes de serviços e aplicações distribuídos TINA, incluindo aquelas em que a sessão de serviço se expande por múltiplos domínios, sendo que esses domínios possuem múltiplos provedores de serviços com um número de usuários que pode ser alterado dinamicamente.

Para garantir um ambiente com múltiplos provedores, se faz necessário definir um gerenciamento de sessão de serviço integral, que permita assegurar todo o fornecimento do serviço fim-a-fim, isto também significa que cada provedor de serviço é responsável pelo apropriado fornecimento e contabilidade de seus próprios componentes de serviço, além de esperar um retorno financeiro em compensação pelo uso dos componentes de serviço.

Portanto, além de garantir uma contabilidade confiável, é de essencial importância considerar questões de segurança associadas ao gerenciamento de contabilidade. A segurança em contabilidade consiste de um serviço garantido e de informações contábeis integras. Nesse sentido, TINA apresenta uma abordagem de segurança na contabilidade que será analisada no próximo capítulo.

4. SEGURANÇA NA CONTABILIDADE TINA

4.1 Introdução

Embora tenha sido reconhecido que a segurança é uma das questões mais críticas em TINA e em computação distribuída em geral, é também reconhecido que grande parte destas questões estão sendo resolvidas. Em um sentido, a segurança envolve cada aspecto de TINA (HAMADA, 1997a). A seguir serão abordadas questões de segurança em TINA.

4.2 Análise do Domínio de Segurança em TINA

A segurança está envolvida em toda a arquitetura TINA, ela afeta todas as partes e não podem ser tratadas isoladamente. Para enfrentar esta complexidade, é necessário estruturar adequadamente os problemas no domínio de segurança de uma maneira adequada. Todos os serviços e recursos estão sujeitos a ataques e invasões (STAAMANN, 1997).

Os ataques podem ser o uso ilegítimo de componentes ou a modificação de dados, configurações ou programas. Estes ataques podem ocorrer através de acessos externos ao sistema, aos dados, aos serviços ou através da alteração das mensagens trocadas entre os componentes. Os invasores podem tanto ser agentes externos como também participantes dentro da própria rede TINA. Os motivos dos invasores podem ser: o uso ilegítimo dos serviços, fraude (negócios *on-line*), fraude de cobrança, observação de consumidores e provedores ou ataques de negação de serviço (*denial of service*).

O objetivo de um ataque pode ser alcançado direta ou indiretamente. No caso deste último, o invasor pode instalar um programa *backdoor*[6] durante o seu primeiro ataque bem sucedido, a qual permitirá que ele retorne mais tarde e continue o ataque. Exemplos de *backdoor* são: as alterações de programas ou a alteração de direitos de acesso (STAAMANN, 1997).

---

[6] *Backdoor*- Um programa ou falha existente no sistema que disponibiliza meios de acesso ao sistema. Um programa do tipo “Cavalo de Tróia” pode ser usado para se instalar um *backdoor* no sistema alvo.

Cada participante em uma rede TINA possui o seu próprio domínio administrativo. O domínio administrativo é um domínio confiável do participante, considerando o fato de que normalmente possui sua própria estrutura física (*hardware*) e que o software é instalado pelo próprio participante. Este domínio confiável pode ser composto de vários nós sobre o controle físico de um participante que são interconectados fisicamente por *links* inseguros. Estes *links* podem ser transformados em canais seguros através do uso de criptografia simétrica sem um gerenciamento sofisticado de chaves. Dentro de seu domínio, o participante confia no correto funcionamento do software instalado.

Para interagir com outros domínios (interações interdomínios), relacionamentos confiáveis devem ser estabelecidos. O canal de comunicação entre os domínios não pode ser assumido como seguro, tornando necessária a adição de segurança através de meios de criptografias. Todas as partes da arquitetura TINA que estão envolvidas nas interações entre domínios devem possuir segurança, caso contrário um componente sem segurança pode comprometer a segurança de todo o conjunto (STAAMANN, 1997).

4.3 Estrutura da Segurança TINA

A estrutura do domínio de segurança pode dividir-se de acordo com dois critérios: o nível estrutural definido em toda a arquitetura e o tipo de informação. Por exemplo, a mensagem de controle (*KTN*) ou conteúdo da comunicação (rede de transporte) (BUTTYÁN, 1999). Definem-se os seguintes sub-domínios do domínio de segurança:

4.3.1 Segurança do Sistema

A segurança do sistema deve assegurar que o sistema, principalmente o *hardware* e o sistema operacional, não estejam sujeitos a invasões. Estão envolvidos os recursos de rede (*switches*, roteadores) e recursos computacionais (Ambiente de Comunicação e Computação Nativa - NCCE, sistema operacional e portas de comunicação), uma vez que as invasões podem não ocorrer nas portas comumente usadas pelo DPE, mas sim, em outras portas do NCCE. Por último, é considerada a segurança no domínio do usuário final (consumidor), onde não se pode assumir que os CPEs (*Costumer Premises Equipment*) como computadores pessoais (PCs) ou *workstations* sejam de uso exclusivo como ponto final de redes TINA.

4.3.2 Segurança do Serviço

A segurança do serviço preocupa-se principalmente em preservar a integridade do controle do serviço. O controle do serviço inclui entre outras atividades a verificação de qual usuário possui permissão para usar um determinado serviço (assinatura) e a contabilidade com a finalidade de cobrança. Ambos confiam na identidade autenticada do usuário, ou seja, esta autenticação deve ser suportada por um protocolo para a autenticação do usuário. Usuários anônimos de um serviço pago podem ser autenticados utilizando identidades anônimas (por exemplo, serviço pré-pago). O protocolo de autenticação deve garantir que informações secretas não possam ser reveladas ou interceptadas por indivíduos não-autorizados. A integridade do controle de serviço inclui a integridade da verificação da assinatura e da contabilidade.

Os acessos às funcionalidades dos serviços são controlados em dois níveis, no nível DPE e no nível de serviço. No nível DPE, pode ser usado um simples controle de acesso baseado nas identidades autenticadas dos usuários envolvidos na sessão. No nível de serviço, a lógica de serviço implementada no componente de serviço controla o acesso às informações específicas de serviços e o acesso às funcionalidades baseadas em identidades autenticadas, contextos e informações de estado. A integridade e confidencialidade das informações trocadas entre as interfaces operacionais dos componentes de serviço são conseguidas através da ativação de características apropriadas dos serviços seguros do DPE. Estas características de segurança do DPE devem fornecer não somente a proteção, a integridade das mensagens e sua ordem temporal, mas também a proteção contra interrupções do próprio controle da conexão.

Os casos especiais de serviços são os serviços de gerência e os serviços de segurança. Ambos requerem um alto nível de segurança (poderosos mecanismos de autenticação, chaves criptográficas longas ou nós fisicamente seguros para sua implementação). Os serviços de segurança especializados fornecem características de segurança especializadas como o suporte para dinheiro digital (*digital cash*), que não está presente em todos os nós DPE, mas que são suportados por provedores dedicados (*Retailers* ou provedores terceirizados) no nível de serviço. Os serviços de gerenciamento estão relacionados com a gerência do sistema, dos serviços e do DPE. A segurança dos serviços de gerência é crucial, uma vez que acessos ilegais às funcionalidades de gerência podem ser usados para a implantação de *backdoors*.

4.3.3 Segurança do DPE

A segurança do DPE é principalmente relacionada com a prevenção de acessos ilegais aos objetos computacionais (CO) e aos grupos de objetos computacionais. As mensagens contendo argumentos, resultados, exceções, invocações de objetos e notificações também são protegidas. Os nós DPE também devem fornecer meios para auditorar e reportar eventos de segurança relevantes que ocorreram no nó de acordo com especificações de auditoria definidas pelo administrador. A segurança do DPE inclui a segurança das implementações do DPE e seus serviços básicos, como o serviço de objetos CORBA.

4.3.4 Segurança do Conteúdo da Comunicação

A segurança do conteúdo da comunicação está relacionada com a autenticação, integridade e confidencialidade das informações do serviço de conteúdo. Como todas as informações do serviço de conteúdo são enviadas sob a forma de fluxo (*stream*), este tipo de segurança é adequada somente para fluxos. Fluxos são protegidos através de mecanismos de criptografia, preferencialmente por cifras ou outros tipos especiais de cifras para certos formatos de informações, como voz e/ou vídeo. Se o serviço implementado pelo domínio do provedor não requerer nenhuma modificação do fluxo entre os dois pontos, então, pode-se estabelecer uma segurança ponto a ponto. Caso contrário, somente a segurança no lado do provedor pode ser estabelecida. O controle de serviço se encarrega da gerência das chaves necessárias (BUTTYÁN, 1999, STAAMANN, 1997).

Outros critérios de interesse a serem abordados nas questões de segurança, são os seguintes, segundo (HAMADA, 1997):

a. ***Framework* de autenticação**: a segurança inicia-se desde a autenticação, sendo que TINA precisa de um *framework* de autenticação pública. A padronização (X.509, PKI) (ISO, 1997) e tecnologia de cartões inteligentes estão disponíveis para uma autenticação confiável. O *framework* de autenticação é utilizado na parte de acesso da arquitetura de serviço;
b. **Interações complexas entre objetos**: as especificações de pontos de referência (FARLEY, 1998) são ilustradas usando diagramas de seqüências de eventos. Os quais servem praticamente como um modelo do comportamento dos pontos de

referência. Embora a seqüência seja dada como um exemplo, este não cobre o comportamento total do ponto de referência, a qual envolve interações complexas entre múltiplos objetos;

c. **Qualidade de Proteção (*QoP*) e integridade**: são vários níveis de qualidade de proteção, desde o mais básico (integridade dos dados, confidencialidade dos dados), até mais complexos (não repudiação, proteção contra a negação de serviços);
d. **Segurança na arquitetura de recursos**: os objetos de recursos de rede são controlados pelo provedor de serviço de uma comunicação e são vistos quase exclusivamente desde o ponto de vista de gerenciamento/controle;
e. **Privacidade e anonimato**: os interesses de segurança do usuário e do provedor não são os mesmos, o que pode motivar conflitos de privacidade e anonimato. Por exemplo, se todas as atividades do usuário são logadas, é provável violar a privacidade do usuário. O anonimato pode aparecer como uma alternativa aceitável por duas diferentes razões: o usuário não precisa mostrar sua identidade ou o usuário não está disposto a pagar;
f. **Componentes auto-testáveis**: os componentes de TINA, tais como agente provedor (PA- *Provider Agent*) e gerente da sessão de uma comunicação terminal (TCSM – *Terminal Communication Session Manager*) se localizam no domínio do usuário. Espera-se que ao executar as funções de gerenciamento e monitoração, estas complementem aquelas funcionalidades pertencentes ao domínio do provedor (*Retailer*), tal como o gerente da sessão de comunicação (CSM – *Communication Session Manager*). Por exemplo, os TCSMs no domínio do usuário e o CSM do provedor, coordenam a configuração de um enlace de fluxo fim-a-fim; e
g. **Questões ao nível de empresa**: isto envolve desde políticas de gerenciamento de segurança na empresa até gerenciamento de *firewall*, desde garantia de QoP até políticas de ajuste de *billing*, etc., que podem ser chamados de questões de segurança ao nível de empresa.

Para garantir estes conceitos de segurança em TINA, definidos anteriormente, pretende-se implementar estratégias e políticas de segurança, aplicando os conceitos de Segurança Multilateral (*Multilateral Security*) e Controle de Acesso Baseado em Papéis

(RBAC – *Role-based Acces Control*), de forma a realizar uma abordagem consistente, flexível e segura para o gerenciamento de serviços de telecomunicações TINA. Sendo esta a linha de pesquisa a desenvolver e o desafio a resolver. Na próxima sessão será apresentada uma descrição geral destes modelos tecnológicos.

4.4 O Modelo de Segurança Multilateral

A segurança multilateral tem por objetivo fornecer segurança para todos os participantes envolvidos, requerendo de cada participante um mínimo de confiança na honestidade dos outros participantes envolvidos. Segundo (PFITZMANN, 2002):

- Cada participante tem seus objetivos de proteção particulares;
- Cada participante pode formular seus objetivos de proteção;
- Conflitos de segurança são reconhecidos e os compromissos negociados; e
- Cada participante pode reforçar seus objetivos de proteção dentro do compromisso negociado.

A segurança multilateral não possibilita necessariamente que todos os participantes possam reforçar seus objetivos de segurança individuais, mas ao menos ela provê a transparência de todas as ações relacionadas à segurança de todos os participantes envolvidos.

4.4.1 Objetivos de Proteção, suas Sinergias e Interferências

Em discussões ocorridas principalmente nas esferas governamentais nos últimos 17 anos, a confidencialidade, a integridade, a disponibilidade e a contabilidade foram definidas como intimamente relacionadas com a segurança de um sistema. Fora da esfera governamental, os conceitos de anonimato (*anonymity*) e da não-observabilidade também se tornaram grandes itens relacionados à segurança, decorrente do avanço das tecnologias de armazenamento que tornou possível o registro permanente de informações pessoais por um custo muito baixo (PFITZMANN, 2001).

Para uma melhor compreensão dos objetivos de segurança no contexto de comunicação em redes de computadores, na Tabela 4.1 é dada a diferença entre conteúdo e circunstâncias da comunicação.

Existem algumas sinergias e interferências entre os objetivos de segurança, de modo que o relacionamento entre certos objetivos de segurança pode ocasionar

dificuldades para um (ou ambos) objetivo(s) de segurança. Também ocorrem casos onde o relacionamento entre dois objetivos de segurança termina por fortalecer um (ou ambos) objetivo(s) de segurança.

Tabela 4.1 - Objetivos de Proteção (PFITZMANN, 2001).

AMEAÇA\PROTEÇÃO	Conteúdo	Circunstâncias
Acesso não autorizado à informação	Confidencialidade Ocultação	anonimato não-observalidade
Modificação não autorizada da informação	Integridade	contabilidade
Alteração não autorizada das funcionalidades	Disponibilidade	alcançabilidade

Essa relação é ilustrada na Figura 4.1.

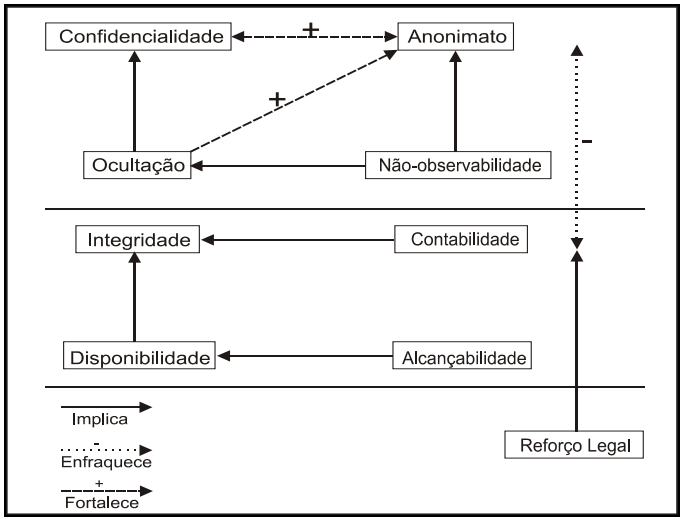


Figura 4.1 - Sinergias e Interferências entre Objetivos de Proteção (PFITZMANN, 2001).

Pftizmann em (PFITZMANN, 2001) introduz a classificação de tecnologias para a segurança multilateral de acordo com o número de participantes em um dado momento e também descreve as distinções entre tecnologias unilaterais, bilaterais, trilaterais e multilaterais. Estes conceitos são definidos e detalhados em (WATANABE, 2003).

4.4.2 Uma Abordagem de Segurança Multilateral: CrySTINA

Uma aplicação do modelo de arquitetura de telecomunicação baseado em *middleware* para segurança, tal como TINA, o qual usa as especificações de segurança

CORBA, é descrito em (BUTTYÁN, 1999) através da arquitetura de segurança ***CrySTINA***. As características de segurança em TINA são implementadas para vários níveis.

Na abordagem de CrySTINA, o DPE (*Distributed Processing Environment*) oferece funcionalidades de segurança gerais para as aplicações sobre cada nó DPE, como parte das funcionalidades DPE. Para integrar estas funcionalidades nas aplicações, deve ser fornecido um completo serviço de segurança DPE. Não obstante, o DPE deve também prover mecanismos de segurança de baixo nível para as aplicações de tratamento de tarefas de segurança de aplicações específicas. As camadas das funcionalidades de segurança são mostradas na Figura 4.2.

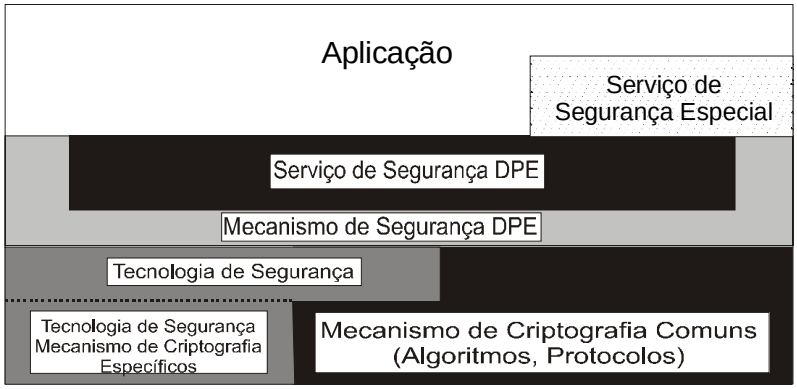


Figura 4.2 - Camadas das Características de Segurança CrySTINA (BUTTYÁN, 1999).

Os serviços de segurança DPE são exclusivamente baseados em mecanismos de segurança DPE. A implementação destes serviços pode usar mecanismos de criptografia diretamente ou pode ser construída sobre tecnologias de segurança de alto nível disponíveis, tais como Kerberos (NEUMAN e TS'O, 1994).

O uso de mecanismos criptográficos e/ou tecnologia de segurança de alto nível pode ser acompanhado através de interfaces padronizadas para facilitar a integração de produtos existentes dentro do DPE. Acima do nível DPE, há serviços de segurança especiais, que confiam exclusivamente nos serviços e nos mecanismos de segurança DPE. Os serviços de segurança especiais não são implementados em cada nó DPE. Exemplos de serviço de segurança especiais, podem ser o suporte a dinheiro eletrônico ou *billing* em um serviço multimídia.

4.5 O Modelo de Segurança RBAC

O termo Controle de Acesso Baseado em Papéis (RBAC – *Role-Based Access Control*) é utilizado para descrever mecanismos de segurança que controlam o acesso de usuários a recursos computacionais, baseado na construção de papéis. Esses papéis definem um conjunto de atividades concedidas para usuários autorizados. Pode-se imaginar um papel como se fosse um cargo ou posição dentro de uma organização, que representa a autoridade necessária para conduzir as tarefas associadas (JANSEN, 1998).

Os usuários pertencem a um papel de acordo com suas responsabilidades e qualificações e podem ser facilmente transferidos sem modificar a estrutura de acesso básica. Novas permissões, bem como a incorporação de novas aplicações e ações, podem ser concedidas aos papéis, e as permissões podem ser revogadas quando necessário (FERRAIOLO, 1995). Para muitos tipos de organização, o modelo RBAC fornece um modo mais intuitivo e eficaz de representar e gerenciar autorizações às informações em comparação a outras formas de controle de acesso (JANSEN, 1998).

Existe uma forte opinião entre os pesquisadores e especialistas de que muitos requisitos não são cobertos pelas políticas discricionárias[7] e obrigatórias[8]. As políticas obrigatórias se originam em ambientes rígidos, como o ambiente militar. Já as políticas discricionárias têm origem em ambientes acadêmicos. Nenhum destes dois tipos de políticas satisfaz as necessidades da maioria dos ambientes empresarias (SANDHU e SAMARATI, 1994). O modelo RBAC é uma alternativa às políticas de controle de acesso discricionária e obrigatória, e está cada vez mais atraindo a atenção, principalmente para aplicações comerciais.

As políticas baseadas em papéis se beneficiam por sua independência lógica, a qual especifica as autorizações de um usuário em duas partes: uma que relaciona os usuários aos papéis e outra que relaciona estes papéis aos direitos de acesso a um objeto. Isto simplifica muito a administração de segurança. Por exemplo, supondo o caso de um funcionário de uma empresa que recebe uma promoção, isto significa que suas

---

[7] Os direitos de acesso a cada recurso e a cada informação, são manipulados livremente pelo responsável do recurso ou da informação segundo sua vontade (WESTPHALL, 2000).

[8] Resume um conjunto de regras rígidas que expressam um tipo de organização envolvendo a segurança das informações no sistema como um todo e supõe que os usuários e objetos ou recursos do sistema estão todos etiquetados (WESPHALL, 2000).

atividades e responsabilidades irão mudar. Para realizar a mudança de autorizações deste funcionário, basta retirar as associações com os papéis atuais e associar novos papéis apropriados para o novo cargo. Se todas as autorizações fossem entre usuários e objetos diretamente, seria necessário retirar todos os direitos de acesso existentes ao usuário e associar os novos direitos de acesso. Esta é uma tarefa trabalhosa e que consome tempo (SANDHU e SAMARATI, 1994).

Nos últimos anos, os grandes fabricantes de software começaram implementar as características do modelo RBAC em bases de dados, para o gerenciamento de sistemas e sistemas operacionais, mas sem nenhuma concordância geral do que realmente constitui um conjunto apropriado de características do modelo RBAC (FERRAIOLO, 1999).

Com RBAC, a segurança é gerenciada em um nível muito próximo à estrutura da organização. Cada usuário está associado a um ou mais papéis, e cada papel está associado a um ou mais privilégios, que são concedidos aos usuários daquele papel. Os papéis podem possuir hierarquia. Por exemplo, alguns papéis em um banco podem ser as pessoas responsáveis pelo atendimento a clientes, caixas, gerente e diretor. A administração de segurança com RBAC consiste em determinar as operações que devem ser executadas por pessoas em tarefas específicas e associar os empregados aos papéis apropriados. Os fundamentos e conceitos do modelo RBAC são descritos em detalhe em (ARMANINI, 2003).

4.5.1 Uma Abordagem RBAC em Gerenciamento de Telecomunicações - TINA

Em uma abordagem de segurança no gerenciamento de serviço de telecomunicações, um dos problemas mais importantes é o *billing*. Em um ambiente de serviço multimídia, é necessário o *billing on-line*, já que o usuário é capaz de correlacionar sua carga de uso/serviço de recurso de rede com seu preço em tempo real. Para isto ser possível, o provedor do serviço necessita enviar relatórios de *billing* diretamente para a interface do usuário (HAMADA, 1998).

A Figura 4.3, ilustra uma sessão de vídeo sob demanda (VoD – *Vídeo on Demand*) de três participantes, com *billing* compartilhado. Na figura, um provedor terceirizado (*3 Pty*) fornece uma fonte de vídeo. A fonte de vídeo é entregue a dois consumidores usando uma conexão de banda larga de multi-difusão. A conta é a soma

da carga do serviço pelo provedor terceirizado, a carga do uso de rede pela comunicação do provedor de serviço (não mostrado na figura) e comissão do fornecedor.

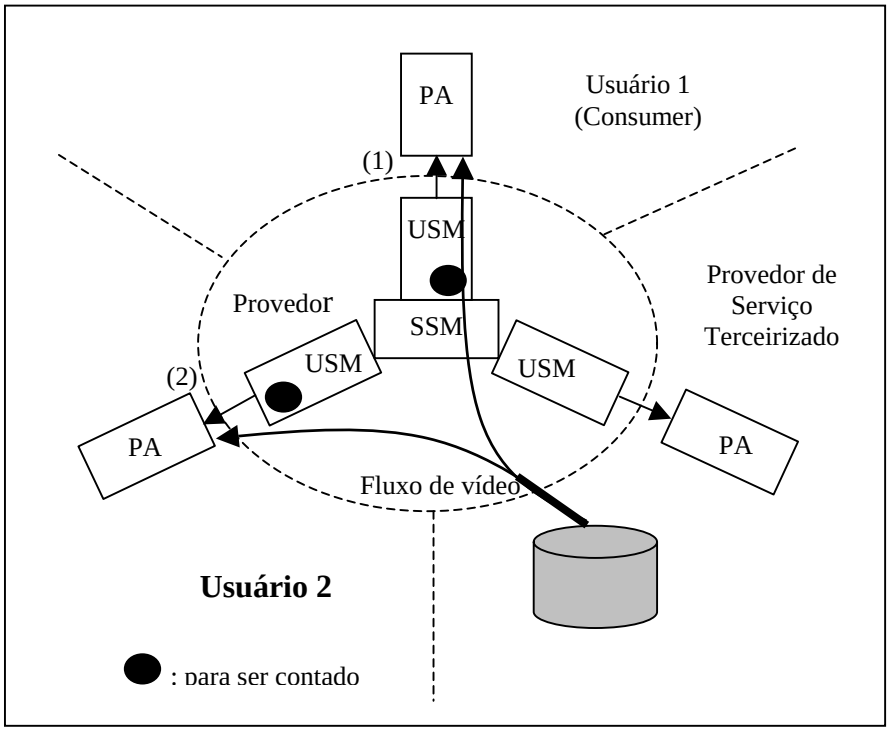


Figura 4.3 - Sessão de Três Participantes com *Billing* Compartilhado (HAMADA, 1998).

- PA (*Provider Agent*): é o agente provedor que representa a participação do provedor (*Retailer*) no domínio do usuário. Também serve como *placeholder*[9] para a informação *billing*;
- USM (*User Service Session Manager*): é o gerenciador de sessão do usuário que gerencia a parte específica do usuário da sessão de serviço; e
- SSM (*Service Session Manager*): é o gerenciador de sessão de serviço que gerencia a sessão de serviço para manter a integridade fim-a-fim do gerenciamento de serviço.

Os componentes de sessão de serviço (SSM e USMs) são criados dinamicamente, quando a sessão de serviço é iniciada. Dessa forma, é natural aplicar

[9] *Placeholder* - é um controle que contém uma área vazia onde, dinamicamente, pode-se incluir e excluir entre outras opções de controle.

RBAC para a sessão de serviço, de modo a fornecer o controle de acesso solicitado entre os objetos. No exemplo, os USMs servem como *placeholders* para os papéis (para ser contado), para ganhar acesso as correspondentes interfaces de relatório *billing* dos respectivos PAs ((1) e (2)).

4.6. Trabalhos Correlatos

Nesta seção serão descritos os trabalhos relacionados com a arquitetura de contabilidade, envolvendo também conceitos de segurança no gerenciamento de serviços de telecomunicações. Muitas linhas de pesquisas têm-se relacionado com o tema.

4.6.1 Na Área de Gerenciamento de Contabilidade

É de conhecimento o grande crescimento dos serviços de telecomunicações, sendo que dentro deste ambiente de comunicação, a contabilidade tem ganho um lugar destacado nas linhas de pesquisa dedicadas pela comunidade científica. Assim, a literatura apresenta uma variedade de trabalhos que serão descritos a seguir.

No artigo de (PRAS, 2001), proporciona-se uma introdução à contabilidade em Internet e discute-se o estado dos trabalhos dentro do IETF (*Internet Engineering Task Force*) e IRTF (*Internet Research Task force*), bem como, certos projetos de pesquisa. Explica como entender o conceito de contabilidade em Internet e certas questões importantes, como por exemplo, "*o que está sendo pago para*?", relacionado com a contabilidade de transporte e "*quem está sendo pago*?", relacionado com o conteúdo da contabilidade. O interesse da contabilidade se encontra no rápido crescimento comercial oferecido pela Internet.

Em (HELLEMANS, 1999), apresenta-se o projeto ACTS FlowThru, o qual visa desenvolver um sistema de gerenciamento que suporte o fluxo de informação gerenciável através dos domínios organizacionais e tecnológicos pela reutilização dos componentes, que têm sido desenvolvidos por outros projetos ACTS.

Os trabalhos de (SEKAKKI, 2001), (SEKAKKI, 2001b), (SEKAKKI, 2001c) e (WATANABE, 2001), discutem as características e requisitos de contabilidade dentro do contexto de gerenciamento de contabilidade, com o objetivo de integrá-lo em um ambiente de serviço baseado no paradigma TINA.

Em (EVLOGIMENOU e BOUTABA, 2002), aborda-se a notação de gerenciamento de contabilidade programável que é muito mais flexível, eficiente e escalável. Especificamente, propõe mecanismos de carga e preço em tempo real usando redes programáveis. Além disso, propõe uma nova arquitetura de contabilidade programável para Redes Privadas Virtuais (VPNs – *Virtual Private Networks*) com QoS disponibilizado.

Um outro trabalho é apresentado por (PINTO, 2002), que propõe uma arquitetura que contempla os aspectos positivos de TINA aliados a padrões abertos como WWW, Java e CORBA. Esta arquitetura suporta a implantação, acesso, uso e gerência de serviços oferecidos através da WWW. O artigo apresenta uma implementação da arquitetura de Serviços TINA, um serviço de laboratório virtual que faz uso desta arquitetura e uma extensão da arquitetura TINA, visando o suporte à ubiqüidade dos serviços de telecomunicações.

Em (RADISIC, 2002), apresenta uma analise dos processos de gerenciamento de contabilidade relevantes e descreve uma solução de contabilidade baseada em conceitos e em políticas para superar as desvantagens de soluções desenvolvidas previamente.

O trabalho de (HWANG, 2004), apresenta um sistema de gerenciamento de transações na Internet através do serviço de gerenciamento de transações (TMS – *Transaction Management Service*) para provedores de serviços de redes, que suportam esquemas de pagamento emissor/receptor em plataformas computacionais heterogêneas.

Em (VAN LE, VAN BEIJNUM e HUITEMA, 2004), propõe-se uma arquitetura de contabilidade e de carga baseada em componentes de serviços flexíveis para otimizar o fornecimento da sessão de serviço, num ambiente multi-domínio e apropriado para uma grande variedade de modelos de negócios.

E em (AGARWAL, KARNIK e KUMAR, 2004), apresenta um modelo de métricas baseado sobre o modelo de informação comum (CIM – *Common Information Model*) para a representação, troca e gerenciamento de informações de medição e contabilidade, porém não aborda questões de segurança dos dados medidos.

Em comparação a nosso trabalho com as experiências apresentadas na área de gerenciamento de contabilidade e em função aos atuais trabalhos de pesquisa apresentados, pode-se estabelecer que o contexto da linha de pesquisa se encontra no estado da arte. Este se baseia sobre o paradigma do gerenciamento de contabilidade e de

serviço da arquitetura TINA, como mencionado no trabalho de (SEKKAKI, 2001), que foi o início da linha de pesquisa desenvolvida no LRG (Laboratório de Redes e Gerência), o qual se tornou nos primeiros trabalhos publicados e de grande motivação para sua continuação.

4.6.2 Na Área de Gerenciamento de Serviços

Trabalhos relacionados com o gerenciamento de serviços são apresentados em (KORMANN, 1996), onde é descrita uma proposta que visa facilitar a interoperabilidade e interconectividade de sistemas de vários fornecedores. Fazendo referência aos padrões ISO e ITU-T para sistemas e gerenciamento de redes, em uma extensão da plataforma OSIMIS (*OSI Management Information Service*). Isto é realizado através da implementação de funções de medidas de uso (métricas), onde uma aplicação é descrita, demonstrando as características práticas da função implementada.

Em (PAVLOU e GRIFFIN, 1997), apresenta-se uma arquitetura de sistema RCM (*Resource Configuration Management*) que trata dos recursos de rede e gerenciamento. Este modelo genérico tem sido verificado através de uma implementação de um protótipo no contexto de uma prova no campo real.

O propósito do trabalho descrito por (PROZELLER, 1997) é mostrar como TINA pode ser posicionado em uma infraestrutura de software de redes de telecomunicações futuras, focalizado sobre as infraestruturas de PTOs (*Public Telecommunication Operators*). O conceito de redes programáveis é incluído para satisfazer as necessidades de PTOs para rapidamente provar e desenvolver novos serviços.

Em (AGOULMINE, 2000), apresenta uma rede integrada e um sistema de gerenciamento de falhas (*trouble*) de serviço, foi projetado para este ambiente dentro do *framework* do projeto European ACTS project FlowThru. O sistema de segurança de qualidade de serviço FlowThru, foi desenvolvido usando os conceitos TINA, TMN e processos de negócios do TeleManagement Fórum. Com o intuito de suportar provedores de serviço com a infraestrutura necessária, modelos e mecanismos para automatizar o gerenciamento de falha fim-a-fim em um ambiente multi-domínio.

O trabalho de (RAJAHALME, 1997), visa principalmente questões ao nível de sessão de serviço. Este descreve como o QoS pode ser dirigido por terminais

(começando desde o baixo nível de serviço até o nível de tecnologias de transporte atuais), como a negociação QoS é feita e como esta negociação afeta o controle de conectividade quando diferentes tecnologias estão envolvidas.

Em (LUNARDI e DOTTI, 2001), apresenta o desenvolvimento de uma camada de adaptação de QoS para incorporar QoS em aplicações multimídia, visando melhorar a qualidade dessas aplicações no cenário de operações atual na Internet.

Em (MORAES e FARINES, 2001), apresenta-se um modelo para a transmissão de vídeo sobre o serviço ABR/ATM (*Available Bit Rate*)/(*Asyinchronous Telecommunications Mode*) para aplicações de videoconferência sobre canais a baixa taxa de bits, baseado em um controlador de taxa que age sobre o codificador de vídeo, cuja saída alimenta um buffer de suavização de taxa.

Um outro trabalho apresentado por (HAMADA, 2004), mostra um simulador de gerenciamento de trafego P2P (*Peer-to-Peer*) usando J-SIM, uma ferramenta de simulação de redes baseada em JAVA para analisar, modelar e avaliar o trafego sobre P2P em redes metropolitanas.

E em (COHEN e RAZ, 2004), apresenta um estudo dos requerimentos e arquitetura básica necessária para um sistema distribuído, em um contexto aberto, de escalabilidade e modular, através de uma simulação baseada em suposições reais, de acordo ao serviço requerido para a informação.

Comparando as experiências apresentadas pela literatura, muitas destas características e conceitos se aproximam aos utilizados no nosso trabalho, no que refere-se a aspectos de gerenciamento de recursos, gerenciamento de métricas e de QoS dos serviços oferecidos e fornecidos pelos provedores. A diferença radica especificamente no uso do padrão de gerenciamento de serviços de TINA a qual apresenta uma arquitetura totalmente definida e estabelecida através de seus modelos conceituais de negócios e de serviço.

4.6.3 Na Área de Gerenciamento de Segurança

Nessa área importantes trabalhos relacionados com mecanismos e políticas de segurança têm sido realizados, proporcionando que os serviços garantam um uso confiável. Entre outros, o trabalho proposto pelos autores (SEKKAKI, 2001a), (ALVAREZ, 2001) e (WESTPHALL, 2003) é implementado um modelo de arquitetura

de segurança para o gerenciamento de contabilidade TINA baseado em objetos seguros, criptografia e etiquetas distribuídas, desenvolvido no LRG como parte da linha de pesquisa deste trabalho.

Em (REICHENBACH, 2002) apresenta um *framework* para o gerenciamento de riscos, que permite aos usuários utilizar instrumentos para avaliar eventuais riscos relacionados com sistemas de pagamento digital e para controlar estes riscos com instrumentos técnicos e econômicos.

Em (STAAMMANN e BUTTYÁN, 1997), apresenta um primeiro resultado do projeto CrySTINA. Analisa e estrutura problemas de segurança no domínio da arquitetura TINA-C. Apresenta uma abordagem para fornecer as funcionalidades na forma de serviços seguros, independente da aplicação auto-contida e mecanismos de segurança, como parte das funcionalidades DPE.

Atualmente pode-se observar diversos trabalhos referentes ao conceito de segurança multilateral, cujo, os principais conceitos são descritos em (RANNEMBERG, 2000), em (PFITZMANN, 2001) e em (PFITZMANN, 2002). Em (SAILER, 1998) é demonstrado como identificar problemas de segurança e assim, aplicar medidas de segurança adequadas.

A segurança multilateral se baseia no uso integrado de diversas tecnologias e abordagens, como em (CLAUB, 2001), que trata da gerência de identidades, em (ANDERSON, 1998), que demonstra o uso de mecanismos esteganográficos para ocultar informações e em (BLEUMER, 1999), que introduz o uso da autenticação biométrica, que fornece um alto nível de segurança por estar relacionada com um perfil físico-biológico único de cada indivíduo.

A aplicação da segurança multilateral em redes de telecomunicações é discutida em (BUTTÝAN,1999), com ênfase em arquiteturas que utilizem como base um DPE, como a arquitetura TINA, por exemplo. Em (SAILER, 1998) é descrita uma abordagem para prover serviços seguros em redes de telecomunicações ISDN/IN.

Para validar os conceitos da segurança multilateral em (PFITZMANN, 1998), um sistema de aplicação distribuída foi implementado, usando Java como linguagem para a programação e Java RMI (*Remote Method Invocation*) para dar suporte ao ambiente distribuído. (RANNENBERG, 2000) também apresenta um protótipo baseado

em PDA´s (*Portable Digital Assistance*) Newton conectados a telefones celulares GSM (*Global System of Mobile Communications*).

Em (ARMANINI, 2003a), apresenta a criação de um *framework* orientado a objeto, baseado no modelo de segurança RBAC, que pode ser utilizado no desenvolvimento de aplicações Web ou quaisquer outras que utilizam Java.

Em (GUSTAFSSON e SHAHMEHRI, 1996), apresenta um novo *framework* para a descrição de papéis, visando aumentar a segurança na informação. A generalidade desse *framework* permite um apropriado modelo de papéis em uma ampla faixa de aplicações, tal como, o controle de acesso baseado em papéis (RBAC) e o gerenciamento de fluxo de trabalho (*workflow*). O *framework* pode também ser usado como uma plataforma de modelo comum em sistemas de informação.

Um outro trabalho apresentado por (HAMADA, 1998), usa o modelo de segurança RBAC para o gerenciamento de serviços de telecomunicações. Extensões do modelo são desenvolvidas, como a representação de espaço de segurança e especificações de álgebra de papéis. Sendo essas, técnicas poderosas para analisar e projetar papéis em um ambiente dinâmico como TINA. Desse modo, permitem ambientes de serviços mais abertos e flexíveis para o futuro das aplicações de telecomunicações, levando em conta, a confiabilidade das telecomunicações e a flexibilidade dos serviços em Internet.

Em (PILZ, 2004), apresenta o conceito de "*Policy-Maker*" para o gerenciamento de segurança de redes heterogêneas. O qual é baseado sobre o modelo de informação CIM e da arquitetura de gerenciamento de empresas baseadas em WEB (WBEM – *Web-Based Enterprise Management*), com o objetivo de administrar políticas de segurança estruturadas hierarquicamente.

Assim em (KELLEY, 2004), apresenta o modelo de gerenciamento SIM (*Security Information Management*), o qual permite encontrar os problemas de negócios e as considerações técnicas relacionadas a eventos de segurança associados com qualquer IT (*Information Technology*), redes e recursos de segurança e que possam ser relatadas de uma maneira clara e concisa, com o objetivo de reduzir tempos de administração e riscos de incrementar a complacência das restrições definidas.

Por último, (KOUTEPAS, STAMATELOPOULOS e MAGLARIS, 2004) apresenta um *framework* distribuído que introduz a abordagem de inter-domínios

cooperativos, com objetivo de evitar os problemas da quantidade de ataques de negação de serviços distribuído (DDoS – *Distributed Denial-of-Service*) e em (THALER e RAVISHANKAR, 2004), apresenta um *framework* que permite coordenar os problemas inter-domínios através de uma abordagem de diagnóstico – localização, com o objetivo de permitir relatar os problemas e receber a realimentação de acordo a sua localização e identidade.

Em comparação a nosso trabalho, são muitos os trabalhos que abordam aspectos de segurança em ambientes de redes de telecomunicações, no qual aspectos específicos relacionados ao modelo de Segurança Multilateral e de Controle de Acesso Baseado em Papéis têm sido tratados como uma nova filosofia de políticas e mecanismos de segurança nos últimos anos, aspectos estes que se encontram no estado da arte. Esta abordagem tem se caracterizado como um tratamento diferente dentro da arquitetura TINA, já que muitos dos conceitos de segurança estão aplicados dentro do ambiente DPE, o qual ao ser um ambiente distribuído pode apresentar vulnerabilidades ao sistema. Isto proporciona características que não garantem funcionalidades de serviço seguras. Assim como os aspectos de segurança RBAC, (HAMADA, 1998) aborda este modelo para o gerenciamento de serviços de telecomunicações de modo de definir os papéis de uma forma dinâmica para os serviços. A diferença com nosso trabalho em primeiro plano radica especificamente que demonstramos que estes modelos de segurança podem ser estabelecidos num ambiente de serviço TINA de forma conjunta sem problemas de incompatibilidades das definições das políticas e mecanismos de seguranças, o que proporciona um acesso e uso de serviço confiável, seguro e flexível num ambiente multi-provedor e multi-usuário em tempo real e em segundo plano demonstramos através de ferramentas de validação que estes conceitos são válidos.

4.7 Conclusão

Devido a TINA ser uma arquitetura aberta para grande variedade de serviços de telecomunicações, isto apresenta possíveis vulnerabilidades a que o sistema está sujeito. Dessa forma, foram apresentados certos mecanismos e políticas que permitem fornecer confidencialidade e integridade na troca de informações entre os diversos participantes de uma sessão de serviço.

Estes mecanismos e políticas foram abordados através da análise de dois modelos tecnológicos, os quais mostram os conceitos de segurança multilateral e o controle de acesso baseado em papéis (RBAC). Estes são os mecanismos que permitirão desenvolver as políticas de segurança em um ambiente de gerenciamento de serviço na arquitetura de contabilidade TINA, onde a carga contábil deve ser realizada em tempo real no fornecimento de serviço fim-a-fim, em um ambiente multi-provedor.

Pode-se dizer que o número de trabalhos desenvolvidos na área de gerência de serviços, gerência de contabilidade e gerência de segurança demonstram a importância e validade da linha de pesquisa. No sentido que proporcionam um robusto acervo bibliográfico, o que permite incorporar e analisar novos conceitos que podem ser introduzidos em um ambiente de gerenciamento de serviço, através do paradigma TINA.

5. DEFINIÇÃO DAS POLÍTICAS E MECANISMOS DE SEGURANÇA

Nesta seção serão descritas as políticas e mecanismos de segurança relacionadas ao modelo de segurança proposto no gerenciamento de contabilidade da arquitetura TINA em um ambiente de serviço multimídia em tempo real.

5.1. Introdução

O estado da arte em relação aos avanços tecnológicos dos serviços de telecomunicações, tem evoluído muito rapidamente, acompanhando estes avanços, também as aplicações para executar estes serviços tem um lugar importante e destacado. Um exemplo atual é o contínuo crescimento de aplicações multimídia na Internet, o que motiva um grande interesse não só de pesquisadores, assim como também de profissionais da área de redes de computadores e usuários em geral. Entusiasmo esse justificado pela promessa de aplicações de grande utilidade como a educação à distância, rádio e TV sob demanda, videoconferências, entre outros. Aplicações como essas trazem requisitos de serviços normalmente expressos como Qualidade de Serviço (QoS – *Quality of Service*).

Apesar desses esforços em oferecer garantia de QoS fim-a-fim, buscando melhorar a atual arquitetura Internet, seja estendendo seu modelo incorporando reserva de recursos e oferecendo serviços diferenciados, ou ainda, fornecendo novos protocolos e novas arquiteturas de comunicação, ainda existem vários fatores que contribuem para que aplicações multimídia não tenham seus requisitos de QoS atendidos de forma satisfatória. Entre eles tem-se: a heterogeneidade dos sistemas finais, fazendo com que uma mesma aplicação se comporte de maneira diferente dependendo da máquina onde é executada; diferença de características (como por exemplo: confiabilidade, atraso, vazão, etc.) do conjunto das redes que compõe a Internet influenciando na percepção do usuário final da aplicação multimídia; oferecimento de reserva de recursos “tudo ou nada”, no qual, ou os recursos necessários para uma aplicação são oferecidos ou nada é oferecido; probabilidade de reservas de recursos não serem suportadas em grande parte na Internet ainda por algum tempo; reserva de recursos ser cara em termos de troca de mensagens e possuir tarifação agregada (LUNARDI e DOTTI, 2001).

Nesse contexto, a disponibilização destes novos serviços demanda um grande volume de aplicações quando comparados aos serviços de voz disponibilizados através

da rede pública de telefonia e rede inteligente (PSTN/IN). Assim, o Consórcio TINA (TINA-C) foi formado para definir uma arquitetura comum de software para serviços multimídia e de informação, no âmbito das telecomunicações. A arquitetura TINA visa o desenvolvimento e gerência de serviços de qualquer complexidade, com maior rapidez e confiabilidade.

O principal objetivo é estabelecer uma arquitetura para serviços de telecomunicações possibilitando seu rápido desenvolvimento, rápida introdução e elevada capacidade de gerência e configuração (PINTO, 2002).

No entanto, o mercado aberto das telecomunicações precisa integrar um efetivo gerenciamento de telecomunicações sobre múltiplos domínios organizacionais e tecnológicos. As aplicações de gerenciamento são freqüentemente projetadas de acordo com uma arquitetura específica e implementadas para serem executadas em plataformas tecnológicas específicas. Nesse cenário, o escopo de gerenciamento de sistema pode dividir-se em: Contabilidade, Desempenho e Segurança.

O sistema de desempenho concentra-se em aspectos associados com o planejamento de uma rede apropriada e com o fornecimento de atividades para a entrega de serviços. O sistema é usado na fase anterior ao ciclo de vida do serviço: como configuração e ordem/subscrição do sistema.

O sistema de segurança concentra-se em problemas relacionados à aspectos da fase inicial de serviço. Este faz uso de SLA (*Service Level Agremment*) estabelecidos na subscrição, como estabelecido no sistema de desempenho, para identificar violações do SLA. Já violações de SLA podem ter um impacto na contabilidade/*billing*, onde um cenário de desconto tem que ser considerado pelo sistema de contabilidade (HELLEMANS, 1999).

Baseado nos princípios e conceitos apresentados anteriormente, serão implementadas as políticas e mecanismos de segurança através das tecnologias de Segurança Multilateral e do Controle de Acesso Baseado em Papéis (RBAC) no Gerenciamento de Contabilidade TINA.

5.2 Questões de Segurança no Gerenciamento de Serviços de Telecomunicações

Um dos principais requisitos para o gerenciamento de contabilidade orientado para o serviço e o consumidor, é que este deve suportar aspectos dinâmicos com

respeito ao gerenciamento de serviço. Assim, deve-se desenvolver um gerenciamento que permita suportar cada atividade requerida para a contabilidade do serviço de uma forma apropriada e segura.

Segundo (RADISIC, 2002), as atividades (que formam o ciclo de vida do serviço) podem ser agrupadas em "*processos*" que acompanham a contabilidade do serviço como um todo. Estes processos descrevem sete fases do ciclo de vida.

i. **Fase de Oferta**: o provedor submete uma proposta consistindo da descrição do serviço e sua tarifa pelo uso do serviço. O consumidor está normalmente revendo as várias propostas de vários fornecedores ao mesmo tempo;
ii. **Fase de Negociação**: trata dos processos de conclusão de um contrato entre o provedor e o consumidor do serviço. Geralmente, um contrato contém detalhes sobre as funcionalidades do serviço, qualidade do serviço (QoS) e o curso da tarifa, assim como as penalidades (ex. um contrato SLA);
iii. **Fase de Implementação**: o provedor implementa o acordo estabelecido com o consumidor das funcionalidades do serviço;
iv. **Fase de Instalação e Prova**: todos os recursos apropriados, por exemplo, equipamento, sistema final, interfaces, aplicações, etc., que serão usados para fornecer as funcionalidades do serviço são instalados e provados, de tal modo que o serviço possa iniciar sua operação. Depois o consumidor expressa sua satisfação a respeito do fornecimento do serviço;
v. **Fase de Operação**: inicia o ciclo de vida do serviço com referência ao gerenciamento de contabilidade, em caso de tarifas baseadas pelo uso do serviço atualmente ativado, o qual deve ser medido e uma fatura deve ser compilada e enviada ao consumidor;
vi. **Fase de Alteração**: combina todas as interações que estão relacionadas às mudanças na funcionalidade do serviço, implementação do serviço e gerenciamento do serviço. Então esta fase é executada de forma paralela à fase de operação; e
vii. **Fase de Desinstalação**: o ciclo de vida do serviço é finalizado e as implementações dos recursos são liberadas.

Estas fases da contabilidade podem ser definidas em dois importantes processos: de **acesso** e de **uso**, como definido no item 2.3.3 e mencionado em (KRISTIANSEN,

1997). O primeiro processo está composto pela **fase de negociaçã**o e o segundo processo composto pela **fase de serviços**. Nesse contexto, a estes dois processos deve-se garantir níveis de segurança que permitam que as etapas de negociação e de uso do serviço sejam de forma transparente, segura e flexível.

Desse modo, em uma primeira etapa que corresponde ao processo de **acesso** em uma sessão de serviço TINA, pode-se introduzir os conceitos de Segurança Multilateral, de modo que as interações dos diversos participantes devem ser executas com um mínimo de segurança para que o serviço seja oferecido de forma confiável e com sucesso. Em uma segunda etapa que corresponde ao processo de **uso** do serviço, pode-se aplicar o conceito de segurança RBAC, que segundo (SANDHU e COYNE, 1996) pode ser aplicado ao gerenciamento de serviço de telecomunicações.

5.2.1. Aplicando os Mecanismos e Políticas de Segurança no Processo de Acesso

Para o processo de acesso será criado o conceito do Contexto de Gerenciamento de Segurança Multilateral (*SecMgmtCtxt*) que irá permitir a coordenação de todo o comportamento da Transação de Serviço com relação aos interesses de segurança definidos para um determinado *SecMgmtCtxt*. A estrutura do *SecMgmtCtxt* é muito semelhante ao do *AcctMgmtCtxt* (como mostra a Figura 5.1), tendo o mesmo domínio e tempo de vida. A estrutura do *SecMgmtCtxt* é constituída pelos seguintes componentes:

- **Declaração dos Componentes de Sessão**: A função desta parte é declarar quais Componentes de Sessão estão disponíveis ao usuário, quais são os componentes que já possuem uma relação segura e o nível de segurança que foi estabelecido com um determinado componente.
- **Objetivos de Segurança/Mecanismos de Segurança**: Discrimina quais objetivos de segurança o participante negociou. Também descreve quais mecanismos de segurança foram adotados para se efetivar os objetivos de segurança desejados.
- **Controle de *Logs***: Registra informações de eventos que ocorram durante a existência do *SecMgmtCtxt* para fins de auditoria . Estas informações podem ser o tipo de acesso que foi associado a um determinado objeto, quem acessou o objeto ou quem modificou o objeto. Quanto mais detalhado for o tipo de *Log* menor será o desempenho do sistema.

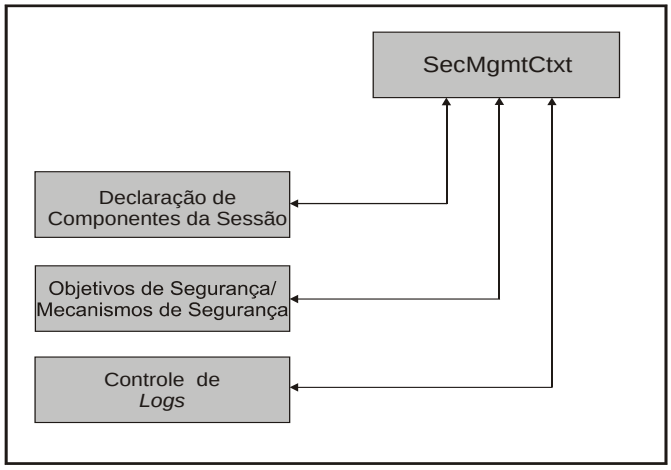


Figura 5.1 – O Modelo do *SecMgmtCtxt*.

Estes interesses de segurança serão previamente definidos entre os participantes. Como a segurança multilateral possibilita a negociação dos interesses de segurança, ao menos os limites mínimos de segurança dos participantes devem estar definidos antes mesmo de qualquer negociação.

Quando for necessário o estabelecimento de uma sessão de comunicação entre dois participantes (entre um consumidor e um provedor, por exemplo) é verificado se ambos os participantes pertencem ao mesmo Domínio de Gerência. Se eles pertencerem ao mesmo domínio, a sessão é estabelecida normalmente. Caso contrário os participantes fazem parte de Domínios de Gerência distintos e, conseqüentemente, de Domínios de Segurança distintos. Neste último caso, a negociação dos objetivos de segurança é necessária. O Agente de Segurança (SA) será o responsável por esta negociação que pode terminar com sucesso e a sessão de comunicação estabelecida ou por um erro, caso não seja possível negociar um conjunto mínimo de objetivos de segurança.

Alguns dos principais itens necessários para a segurança multilateral que deverão ser adicionados são:

- configuração externa ao usuário das características de segurança das aplicações para expressar suas preferências de segurança;
- negociação entre os sistemas dos participantes para acertar o problema de diferentes configurações; e

– uma visão abstrata dos mecanismos de segurança assim como uma API (*Application Program Interface*) abstrata para prover a flexibilidade necessária para a configuração externa.

A continuação pode-se definir estes mecanismos da seguinte maneira:

a. **Configuração:**

A configuração é a principal parte visível da plataforma para o usuário final. Uma atenção especial deve ser dada aos usuários com pouca experiência.

b. **Interface ao usuário**:

A interface ao usuário habilita o usuário final a uma fácil configuração de mecanismos de segurança e serviços para o seu sistema e suas aplicações distribuídas. A interface de configuração de segurança do sistema é parte do SA e provê a possibilidade de “entrar” dados de configuração centrais, as quais formarão a base para a configuração da segurança para toda a aplicação distribuída.

Para cada objetivo (requisito) de segurança como a confidencialidade, integridade, anonimato ou contabilidade, o usuário pode escolher o mecanismo de segurança de sua preferência. O sistema provê uma lista de mecanismos disponíveis. Novos mecanismos de segurança podem ser adicionados ao sistema posteriormente.

O usuário precisa se decidir sobre qual objetivo de segurança ele deseja alcançar para a sua aplicação.

c. **Negociação:**

Todos os participantes em uma aplicação distribuída podem configurar seus respectivos sistemas de acordo com os seus objetivos de segurança. Naturalmente, diferenças na configuração irão surgir. A melhor maneira para sobrepor estas diferenças é a negociação entre as partes envolvidas. De maneira a envolver o usuário o mínimo necessário nas interações, a negociação deve trabalhar o mais automáticamente possível.

A negociação em si de divide em duas partes. Primeiro, os participantes concordam com um objetivo de segurança geral, como confidencialidade ou contabilidade. Depois, os mecanismos utilizados para se atingir este objetivo de segurança são negociados.

Na Figura 5.2, o protocolo desenvolvido para a fase de negociação é ilustrado. Ele basicamente mostra que ambos os participantes criam uma proposta para uma possível configuração de segurança em comum. Estas propostas são trocadas e avaliadas. O iniciador da comunicação cria uma proposta que incorpora as propostas de todos os participantes. Se o sistema não for capaz de gerar esta proposta, o componente de negociação informa ao usuário que não foi possível chegar a um denominador comum para uma comunicação segura. Como resultado, ao menos um dos usuários deve mudar sua configuração de segurança ou a comunicação não poderá ser estabelecida.

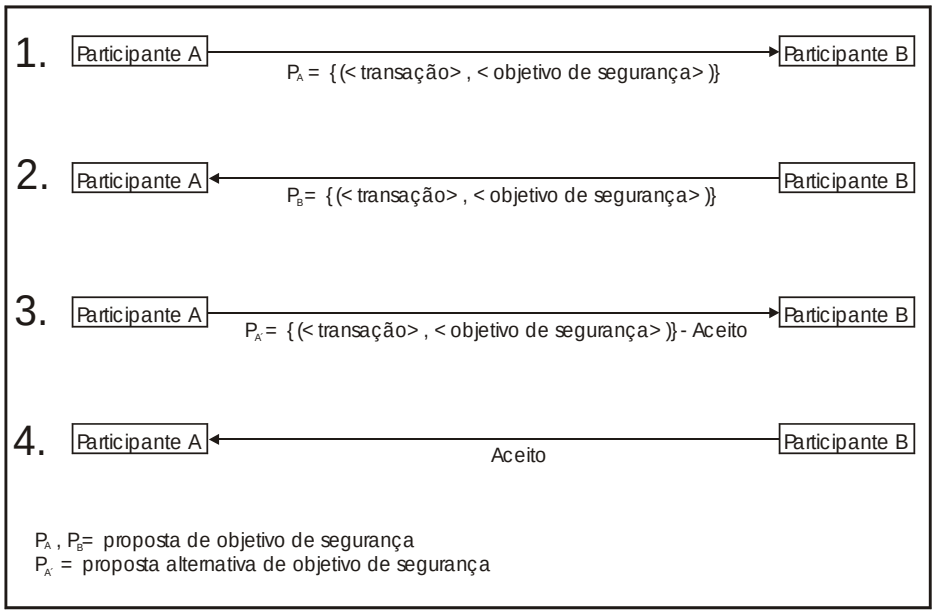


Figura 5.2 - Protocolo de Negociação da Fase 1 (WATANABE, 2003).

Desse modo, com os requisitos de segurança gerais alcançados, o sistema negocia os algoritmos de cifragem. O princípio é o mesmo como descrito acima, mas a solução para conflitos mais sérios se estende para a possibilidade da carga de novos mecanismos do servidor ou a incorporação de um *gateway* de segurança, que poderá executar as transformações necessárias entre diferentes mecanismos de segurança.

Um protocolo que permite estabelecer estas condições de negociação na primeira fase é denominado Protocolo SSL, o qual será utilizado como o mecanismo de segurança na etapa de negociação.

d) Protocolo SSL - *Socket Service Layer*:

O SSL é um protocolo de segurança que provê privacidade da comunicação sobre a Internet. O protocolo permite aplicações Cliente/Servidor de forma que é projetado para prevenir a escuta (*eavesdropping*), adulteração (*tampering*), ou falsificação de mensagem (*forgery*). O SSL (FREIRER, KARLTON e KOCHER, 1996) é um protocolo de propósito geral adequado para proteger conexões em sistemas distribuídos, prover autenticação, confidencialidade e integridade para comunicações através de conexões TCP/IP. Este protocolo está localizado entre o protocolo da camada de transporte (acima do TCP) e o protocolo da camada de aplicação, e é composto por suas camadas: o protocolo ***SSL Record***, que provê conexões seguras garantindo confidencialidade e integridade das mensagens transportadas, através de criptografia simétrica e de mensagens de integridade (MAC – *Message Authentication Code*); e o protocolo ***SSL Handshake***, que permite a autenticação de cliente e servidor, negocia algoritmos criptográficos e gera a chave simétrica usada pela camada SSL Record. Sua arquitetura é mostrada na Figura 5.3.

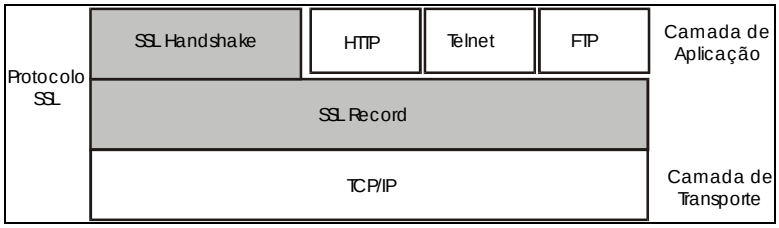


Figura 5.3 – Arquitetura do Protocolo SSL.

O protocolo ***SSL Record*** é usado para transferir dados de aplicação e de controle SSL entre cliente e servidor. Estes dados estão possivelmente fragmentados em pequenas unidades, e podem ser comprimidos, assinados e cifrados antes de serem transmitidos pelo protocolo de transporte confiável (TCP).

O SSL utiliza, para autenticação de cliente e servidor, certificados digitais e transferências com assinaturas digitais. A sessão SSL é estabelecida quando a negociação inicial (*handshake*) entre cliente e servidor é realizada. O fluxo de mensagens durante o *handshake* está representado na Figura 5.4.

A autenticação no SSL usa os certificados X.509 (ITU-TX509, 1993) para transferir informações sobre a chave pública de uma aplicação. O SSL v3 usa os certificados X.509 v3 para assegurar chaves públicas RSA (*Rivest*, *Shamir* e *Adleman*), e um

certificado X.509 modificado para assegurar chaves públicas, usadas pelo protocolo de distribuição de chaves do Departamento de Defesa dos EUA, *fortalezza/DMS*.

O X.509 *certificate* possui autoridades certificadoras localizadas no mundo todo. O X.509 é um padrão internacional que tem aplicabilidade em diversos campos, com forte aceitação internacional.

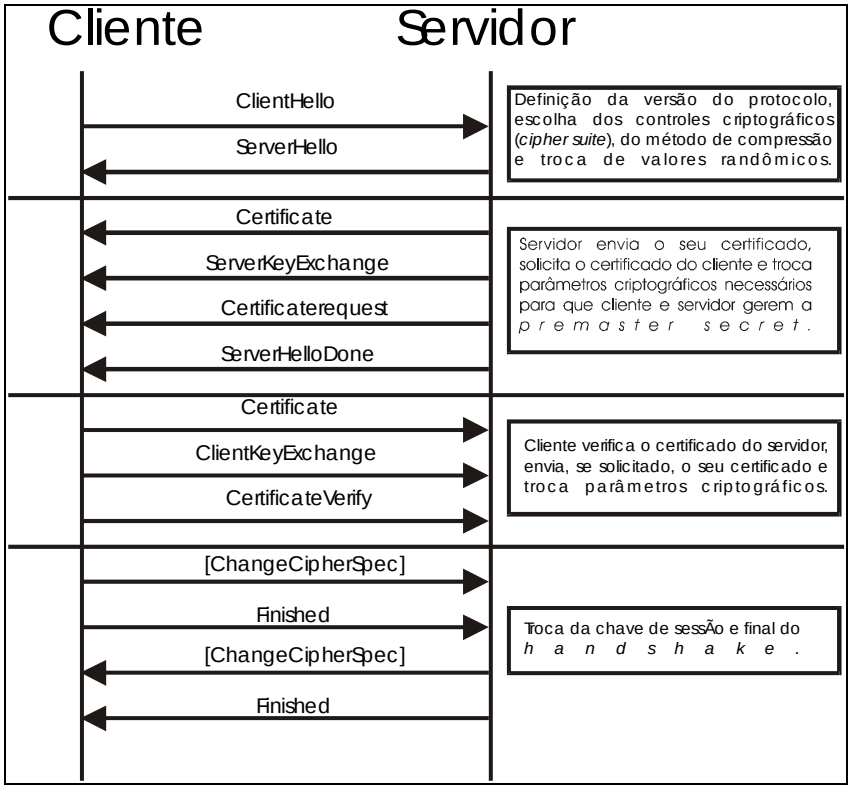


Figura 5.4 – Fluxo de Mensagens do *handshake*.

Portanto, fazendo um comparativo entre os requerimentos da etapa de negociação (que corresponde à primeira fase) e o protocolo SSL, podemos estabelecer mecanismos compatíveis para a definição de uma política de segurança multilateral utilizando o mecanismo de mensagens *handshake* e do certificado X.509 do protocolo SSL.

5.2.2. Aplicando os Mecanismos e Políticas de Segurança no Processo de Uso

Como mencionado no item 4.5.1, um dos problemas mais importantes do gerenciamento de serviço é o *billing*. Este ponto será tratado, analisado e estudado nesta seção, onde será abordado nosso modelo proposto na tese através da definição do modelo RBAC, como uma contribuição inédita deste trabalho. Na sequência analisamos

os conceitos relacionados à definição dos papéis num ambiente de gerenciamento de serviço.

5.2.2.1 Gerenciamento de Segurança e Espaço de Segurança

A sessão de serviço é executada por um conjunto de componentes de serviço, tais como SSM (*Service Session Management*) e USM (*User Service Session Management*), os quais são separados daqueles componentes de serviços correspondentes a outras sessões de serviço. Claramente, uma sessão de serviço representa por si mesma um interesse de segurança separado daquelas outras sessões de serviço, eventos através dela podem ser usados pelos mesmos membros da sessão.

Na arquitetura de serviço TINA, a autenticação ocorre na sessão de acesso (ambiente de segurança Multilateral, fase de negociação), e o resto do gerenciamento de segurança ocorre principalmente dentro da sessão de serviço (ambiente de segurança RBAC). Cada sessão de serviço é caracterizada por uma chave de sessão em sistemas simétricos, ou um par de chaves em sistemas assimétricos. Em outros casos, as chaves de sessão são usadas para separar interesse de segurança em uma sessão de serviço (e sessões de comunicação produzidas desde a sessão de serviço), das outras sessões de serviço.

Por exemplo, um SSM de uma sessão de serviço pode ser protegido de operações de invocação de objetos de outras sessões de serviço usando um certificado gerado das características da chave de sessão, eventos através de interfaces de referência do SSM podem ser acidentalmente revelados para outras sessões de serviço. Outros métodos de proteção tais como a criptografia de mensagem e verificação de integridade naturalmente seguem e derivam das características da chave de sessão.

Como mostrado no exemplo da Figura 4.3, o modelo RBAC apresentado por (SANDHU e COYNE, 1996) pode ser aplicado para o gerenciamento de serviço de telecomunicações. Estes têm diferenças no domínio de aplicação, não obstante, entre o modelo organizacional ao qual RBAC tem sido tradicionalmente associado, e o gerenciamento de serviços de telecomunicações, tais como:

1. Os papéis no gerenciamento de serviço de telecomunicações não são associados com papéis organizacionais; eles são melhor associados com tipos de serviço e classes de serviços.

2. Os papéis de serviços são mais transitórios que os papéis organizacionais; os usuários tomam seus papéis só quando uma sessão de serviço está sendo ativada.
3. A sessão de serviço visa ser melhor para múltiplos participantes, como mostrado na Figura 4.3, onde quatro participantes estão na sessão. Dentro do escopo da sessão de serviço, objetos e interfaces são dinamicamente instanciados. Os papéis interatuam com cada outro na sessão de serviço.
4. Uma sessão de serviço estende-se sobre múltiplos domínios administrativos. No exemplo da Figura 4.3, estende-se sobre quatro domínios.

Baseado sobre estas observações, recomenda-se que uma sessão de serviço seja vista como uma classe de espaço, definida como *Espaço de Segurança* (HAMADA, 1998). O espaço de segurança é povoado de várias classes de papéis, que interatuam com cada outro e são dinamicamente limitados por objetos ou interfaces.

A Figura 5.5 ilustra uma sessão de serviço de segurança entre um Usuário e Provedor:

1. A sessão começa com uma autenticação. Na figura duas chaves assimétricas (***Ku, Kp***) são usadas.
2. e 4. Os certificados de papéis para o Papel R_1 (para-ser-contado:***billed***) e o Papel R_2 (papel do usuário de VoD:***VoD_U***) do usuário são dados pelo provedor. Os certificados de papéis são temporizados (*timestamped*) e digitalmente atribuídos (sinalizado) pelo provedor ($\{\}_p$), os quais são criptografados pela chave de sessão (***k*** $_i$ e ***k*** $_{i+1}$, respectivamente).
3. e 5. Alguma chave da sessão necessita ser substituída por uma nova. Isto acontece quando o tamanho da chave é pequeno, e as chaves não se ajustam para um uso contínuo e a longo prazo.
6. Término da sessão, e os objetos da sessão devem ser destruídos.

Não obstante, em uma sessão de múltiplos participantes, estes requerem uma separação refinada dos interesses de segurança, porque simplesmente os interesses de segurança não são os mesmos.

Desse modo, cada interesse de segurança é protegido por uma seqüência de chaves de sessão e certificados de papéis, como ilustrado na Figura 5.5. Este conjunto de chaves e certificados são coletivamente chamados de chave **Generalizada para o**

Espaço de Segurança (HAMADA, 1998). ***K1*** é a chave generalizada do espaço de segurança, o qual representa exclusivamente o interesse de segurança do provedor, e ***K2*** é a chave generalizada do espaço de segurança, o qual representa o interesse de segurança comum para todos os participantes.

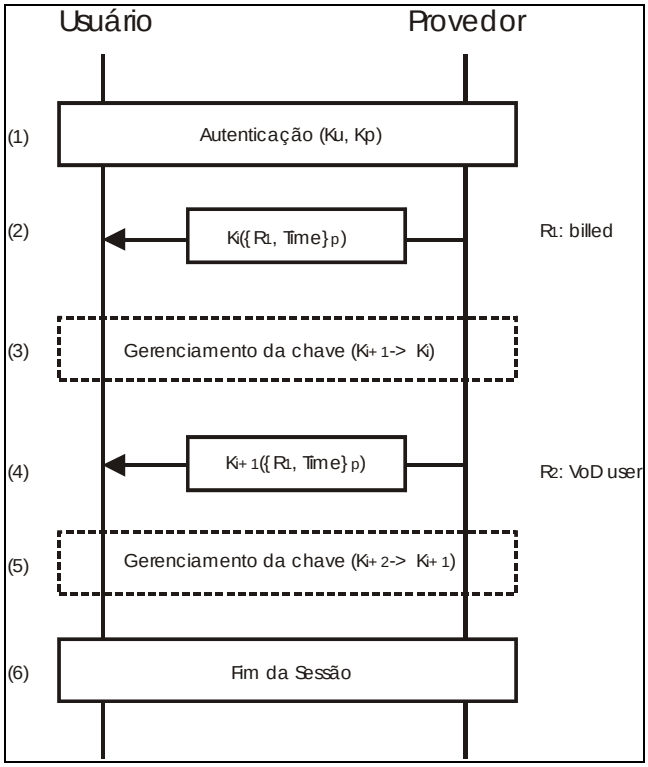


Figura 5.5 – Interação Usuário-Provedor em uma Sessão de Serviço (HAMADA, 1998).

Assim por exemplo, o SSM provê uma interface de controle, a qual permite ao provedor monitorar o desempenho da sessão, ou para suspender ou retomar a sessão de serviço toda. Esta interface deve ser acessada exclusivamente pelo provedor, e deve ser protegida dos outros membros participantes na sessão. A proteção desta interface, portanto, confia-se a ***K1***.

De outra forma, o SSM pode prover uma interface de monitoramento comum para todos os membros participantes na sessão, tal que qualquer um na sessão possa conhecer quem está na sessão e quantos membros estão na sessão. Esta interface necessita ser exclusivamente para os membros participantes, e necessita ser protegida de qualquer pessoa externa à sessão. A proteção desta interface, portanto, confia-se a ***K2***.

Obviamente, dois interesses de segurança podem ser diferenciados só pela escolha de diferentes chaves. Uma chave generalizada é definida como:

$$C = (\ IDSessão, \{\ ChaveSessão\ \}\ ).$$

O ***IDSessão*** é feito público, e é usado para designar o espaço de segurança. A ***ChaveSessão*** é usualmente mantida em secreto, e só é conhecida pelas partes interessadas. Baseado nestes conceitos é que se definirão, nas etapas de autenticação e atribuição de papéis, as políticas e mecanismos de segurança para o modelo arquitetural em um ambiente de sessão de serviço TINA.

5.2.2.2 Controle de Acesso Baseado em Papéis na Sessão de Serviço

Uma sessão de serviço segura entre um usuário e um provedor inicia-se através da negociação e autenticação. Posteriormente, são definidos os papéis e sua designação, usualmente isso ocorre quando for iniciada a sessão de serviço (fase de operação do serviço). Então, o provedor emite um certificado de papel que define os papéis do usuário, como por exemplo, um papel R_1 (*Billing*) e o papel R_2 (VoD – *Video on Demand*). O exemplo baseado em (HAMADA, 1998), pode ser ilustrado através da Figura 5.6 e descrito a seguir:

O componente SSM (*Service Session Manager*) preserva a Relação de Serviço (SR – *Session Relation*). O SR assegura o enlace relacionado ao papel do usuário na sessão, para manter sua integridade. Neste exemplo, três domínios de negócios estão participando na sessão de serviço (Provedor, Usuário e Provedor Terceirizado). O provedor terceirizado participa com o papel de provedor de conteúdo VoD (VoD – CP) e o usuário participa com o papel de usuário de VoD (VoD-U). Assim temos:

1. o USM (*User Session Manager*) correspondente ao usuário (rotulado como VoD-U) solicita um certificado de seu papel (VoD) desde o Agente de Controle de Acesso (ACA – *Access Control Agent*) no domínio do provedor;
2. o ACA retorna um certificado de papel ao USM solicitador;
3. o USM passa o certificado ao usuário final UAP (*User Application*); e
4. o usuário de VoD ativa a interface de controle de VoD do servidor de VoD do provedor terceirizado, usando o certificado obtido no passo anterior.

Neste cenário é ilustrado uma visão de um modelo centralizado do gerenciamento de papéis na sessão de serviço TINA, na qual a criação de papéis e

emissão de certificados por agente de controle de acesso é exclusivamente controlado pelo provedor. Este modelo centralizado apresenta uma melhor adaptação ao modelo de negócios TINA.

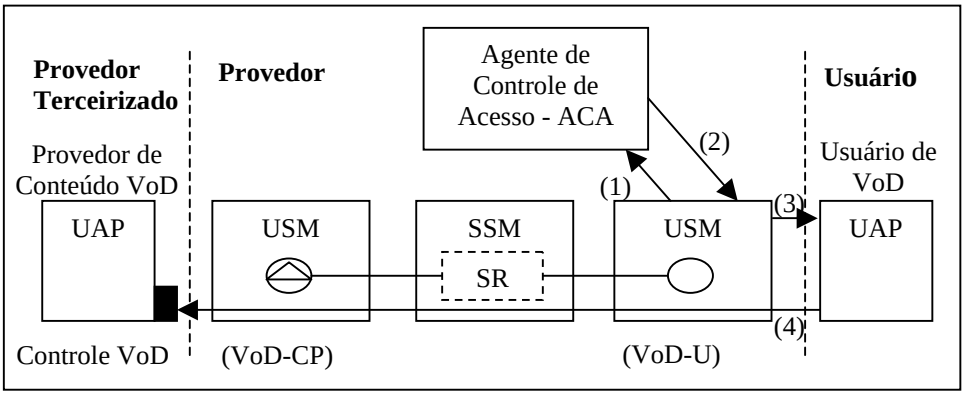


Figura 5.6 – Utilizando RBAC em uma Sessão de Serviço VoD.

5.2.2.3 Papéis e Privilégios de Acesso

Como foi mencionado previamente, os papéis interatuam com cada outro dentro da sessão de serviço. Isto pode ser expressado da seguinte maneira (HAMADA, 1998):

***R_import* x *R_export*: (*I_type* → *Privilégios*)**.

A fórmula implica que o produto de dois papéis, um dos quais é usado por uma entidade acessando (*R_import*) e outra usada por uma entidade acessada (*R_export*), define um operador (uma função), a qual retorna privilégios de acesso de um tipo de interface (*I_type*) num dado espaço de segurança. Este mapeamento é especificado pelas os seguintes 5-tuplas:

(***R_import, R_export, I_type, Privilégios, SecSpace***).

- ***R_import***: o papel da entidade acessando (objeto).
- ***R_export***: o papel de entidade acessada (objeto).
- ***I_type***: o tipo da interface sendo exportada desde *R_export* para *R_import*. Este pode representar um conjunto de interfaces associadas com o tipo de serviço.
- ***Privilégios***: privilégios de acesso dados para uma entidade acessando de *R_import* respeito a *I_type*. Um símbolo "*" implica que qualquer operação na interface é permitida.

- ***SecSpace***: designador de espaço de segurança. Um símbolo "*" implica a sessão de serviço toda.

Por exemplo, os papéis VoD da sessão de serviço da Figura 5.6 resultam a seguinte tupla:

(*VoD-U, VoD-CP, VoD_control, *, **).

5.2.2.4 Certificado de Papéis

Um certificado de papel assegura que ao proprietário do certificado é dado um papel pelo emissor do certificado. O emissor é normalmente o provedor da sessão de serviço. Este tem os seguinte campos:

- ***Role ID***: identificador do papel.
- ***Holder ID***: este deve conter a identidade do proprietário, para quem o certificado é emitido, para prevenir a reprodução do certificado.
- ***Security space***: para prevenir a reprodução do certificado pelo mesmo proprietário, este tem que mostrar o ID da sessão (uma parte pública da chave generalizada do espaço de segurança).
- ***Time stamp***: a existência de um certificado de papel é controlada dentro de um certo tempo no espaço de segurança, dentro do qual o papel designado é limitado para o usuário. O *Time stamp* combinado com o valor do tempo de vida pode servir para estes propósitos.
- ***Digital signature by the issuer***: um certificado é digitalmente assinado pelo emissor.

Baseado nestes conceitos, definiremos as políticas e mecanismos de segurança na segunda fase do modelo proposto através do uso do serviço, definindo os papéis num contexto de segurança RBAC e em um ambiente de multi-serviço e multi-usuário com *billing on-line*.

5.2.3 A Definição dos Papéis na Fase de Uso através do *RBACMgmtCtxt*

Finalizada a primeira fase de negociação que corresponde a uma relação de segurança definida pelo *SecMgmtCtxt*, será então necessário definir a relação de segurança que deve ser estabelecida para o processo de uso. Neste processo será criado o conceito do Contexto de Gerenciamento de Controle de Acesso Baseado em Papéis –

RBACMgmtCtxt, que permitirá definir e coordenar os papéis para um serviço com *billing on-line* no Contexto de Gerenciamento de Contabilidade. A estrutura será definida dinamicamente, similarmente ao *AccMgmtCtx*t, tendo o mesmo domínio (domínio Provedor) e tempo de vida. A estrutura do *RBACMgmtCtxt* é constituída pelos seguintes componentes como mostra a Figura 5.7.

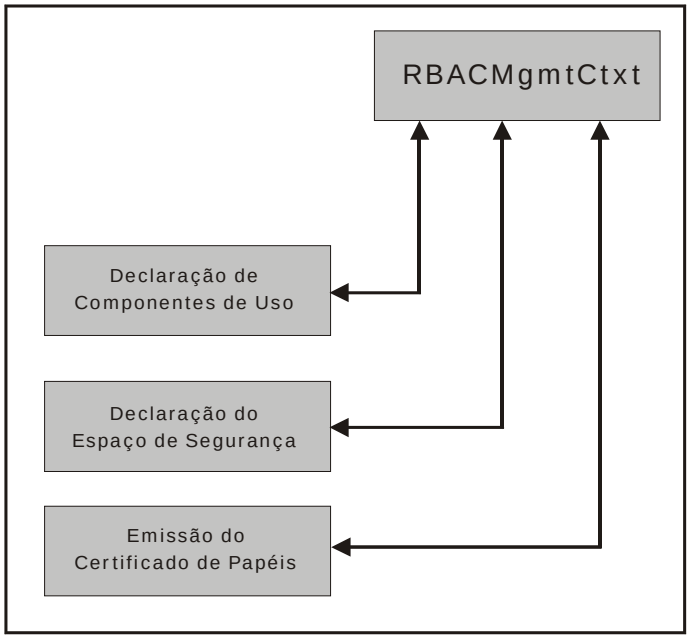


Figura 5.7 – O Modelo do *RBACMgmtCtxt*.

- **Declaração dos Componentes de Uso**: A função desta parte é declarar quais componentes de uso estão disponíveis ao usuário, quais serviços o provedor fornece em um ambiente de serviço multimídia com *billing on-line*, onde os recursos de rede fornecido pelo serviço relacionado a um usuário devem ser em tempo real. Também se incluem relatórios billing para a interface do usuário.
- **Declaração do Espaço de Segurança**: Como uma sessão de serviço estende-se por múltiplos domínios, se faz necessário definir um espaço de segurança que permita definir as classes de papéis de forma dinâmica, através de objetos e interfaces. Nesta etapa se definem as chaves do espaço de segurança as quais representam interesses de segurança para cada participante, usuário-provedor. A atribuição destas chaves para definir o

espaço de segurança dos participantes dos serviços é estabelecida através da etapa de negociação pelo protocolo SSL usando os certificados X.509.

- **Certificado de Papéis**: o certificado de papel atribuído ao usuário assegura o uso do serviço solicitado ao provedor, cuja definição é estabelecida através da declaração dos componentes de uso.

Este contexto é definido na segunda fase da sessão de serviço, onde primeiramente é realizada a negociação e posterior autenticação na fase de acesso, utilizando mecanismos de segurança que são negociados entre os participantes, e que cabe o interesse de segurança ser definido através do protocolo SSL, como mencionado no item 5.2.1.

Nesta segunda fase os componentes SSM e USM cumprem uma especial função, já que eles formam parte da sua interação com o contexto *AccMgmtCtxt* e que são criados dinamicamente quando é iniciada a sessão de serviço, assim deve-se definir claramente o espaço de segurança para proteger as operações de invocação de objetos usando os certificados que são gerados das características da sessão, como descrito no item 5.2.2.1.

Assim definido o espaço de segurança através das chaves obtidas e associadas especificamente pelo protocolo SSL, pode-se atribuir os papéis por meio do certificado de papéis que é responsabilidade do componente ACA no contexto *RBACMgmtCtxt*. Este componente interage diretamente com o USM definido para o usuário e gerenciado pelo componente SSM, para a emissão do certificado pelo provedor para o consumidor via o componente UAP no domínio do consumidor. Este certificado permitirá definir o papel do usuário no serviço, o qual é identificado através dos campos descritos no item 5.2.2.4 e que relaciona seu objeto *billing on-line* para estabelecer a coleta de dados contáveis que interagem com o contexto *AccMgmtCtxt* definido no gerenciamento de contabilidade TINA para um ambiente de serviço multi-provedor e multi-usuário em tempo real.

5.3 Conclusão

A contabilidade apresenta atividades agrupadas em processos que formam o seu ciclo de vida. O gerenciamento de serviço TINA também separa em dois importantes processos suas atividades, um primeiro processo que forma a fase de negociação e um

segundo processo que forma a fase de uso do serviço dentro de um contexto de gerenciamento de contabilidade das suas atividades.

A estas atividades foram incorporados políticas e mecanismos de segurança através dos modelos de Segurança Multilateral e de Controle de Acesso Baseado em Papéis. Como parte dos níveis de segurança estabelecidos e definidos, foram estes modelos incorporados em um primeiro nível no processo de negociação, em que se incorporou o Contexto de Gerenciamento de Segurança Multilateral (*SecMgmtCtxt*) que permite aos participantes através da negociação estabelecer seus interesses de segurança e num segundo nível definir através do Contexto de Gerenciamento RBAC (*RBACMgmtCtxt*) os certificados e papéis dos usuários definidos pelos serviços fornecidos pelo provedor dentro do processo de uso, o qual interatua com o Contexto de Gerenciamento de Contabilidade (*AcctMgmtCtxt*) para permitir um controle seguro e confiável dos eventos contáveis de um serviço que garantem um *billing on-line* para o usuário que está fazendo uso do serviço escolhido. Estes contextos terão o mesmo tempo de vida que o contexto de gerenciamento enquanto eventos contábeis estiveram sendo ativados pelo fornecimento de um serviço a um usuário.

Portanto, as políticas e mecanismos definidos permitiram um ambiente de segurança no contexto de gerenciamento de contabilidade TINA confiável, sem incompatibilidades das políticas de seguranças de ambos contextos. Formalizados os contextos de segurança, podemos definir e descrever a continuação o modelo de segurança para a arquitetura de contabilidade TINA.

6. DESCRIÇÃO DO MODELO DE SEGURANÇA NO GERENCIAMENTO DE CONTABILIDADE TINA

Neste capítulo descreveremos os propósitos de cada componente de serviço do modelo de segurança para o gerenciamento de contabilidade segundo o padrão da Arquitetura TINA (ABARCA, 1998), incorporando os componentes que possuem as funcionalidades e mecanismos de segurança do modelo. Esses componentes devem interagir com os componentes de Gerenciamento de Contabilidade TINA em um ambiente de contabilidade em tempo real com multi-provedores, de modo que os processos contábeis tornem o serviço de contabilidade integrado e confiável para os usuários.

O modelo se segurança da arquitetura de contabilidade denominado SSGCT – Sistema de Segurança do Gerenciamento de Contabilidade TINA, é apresentada na Figura 6.1, onde são mostrados os domínios que correspondem ao consumidor (quem solicita um tipo de serviço), ao provedor (quem fornece o serviço ao consumidor e gerencia a contabilidade do serviço) e ao provedor terceirizado (que fornece o serviço solicitado pelo provedor).

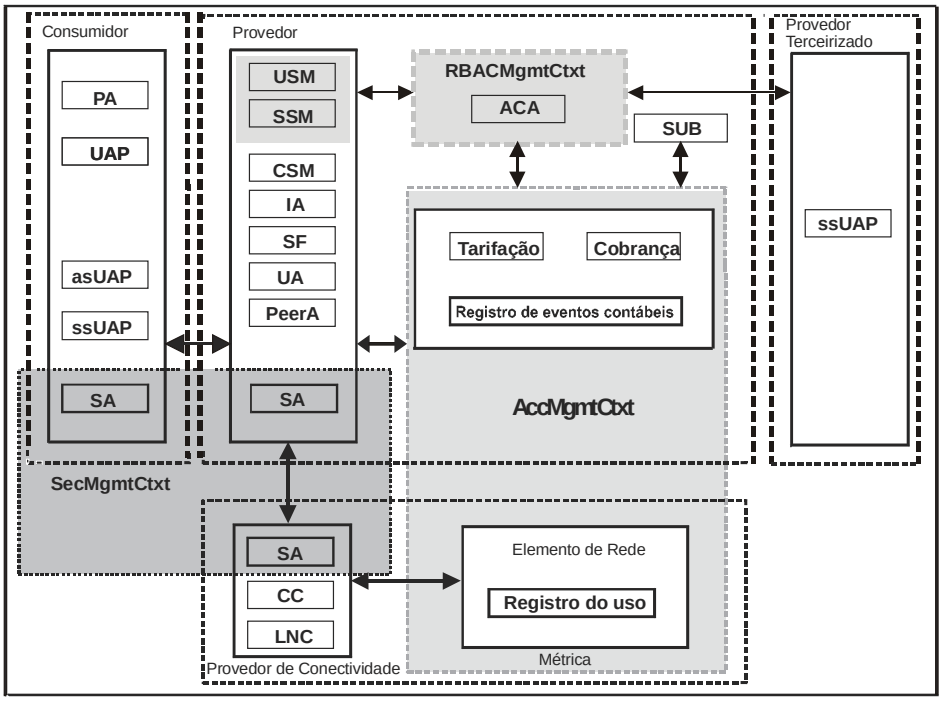


Figura 6.1 – Modelo de Segurança para o Gerenciamento de Contabilidade Usando Segurança Multilateral e RBAC - SSGCT.

Também na Figura 6.1 estão representados os contextos de segurança (*SecMgmtCtxt, RBACMgmtCtxt),* o contexto de contabilidade (*AccMgmtCtxt*) e os correspondentes componentes de cada domínio.

Definido o modelo de segurança para o gerenciamento de contabilidade, podemos a continuação descrever as funcionalidades de cada componente:

6.1. Descrição dos Componentes TINA e do Contexto de Segurança

As descrições dos componentes TINA, são especificadas como padrões através da Especificação de Componentes de Serviço (ABARCA, 1998), os quais descrevem as funcionalidades dos componentes computacionais da arquitetura TINA, como mostrado na Figura 6.1, e que correspondem a:

- asUAP - *Access Session User Application*.
- PA – *Provider Agent*.
- UA – *User Agent*.
- SUB – *Subscription Management Component*.
- ssUAP – *service session User Application*.
- SF – *Service Factory*.
- SSM – *Service Session Manager*.
- USM – *User Service Session Manager*.
- SLCM – *Service Life Cycle Management*.
- PeerA – *Peer Agent*.
- PeerUSM – *Peer Usage Session Manager*.

Os componentes do contexto de segurança do modelo correspondem a:

- *SecMgmtCtxt – Security Management Context*.
- *RBACMgmtCtxt – Role Based Access Control Management Context*.

A seguir descreveremos cada um destes componentes de maneira a entender suas funcionalidades e interações dentro de cada domínio, como definido para o modelo de negócio e a arquitetura de serviço TINA, incorporando os componentes dos modelos de segurança que permitiram definir um ambiente de serviço multi-usuário e multi-provedor com *billing* em tempo real, dentro de um contexto de gerenciamento de contabilidade (*AccMgmtCtxt*) seguro e confiável.

1. asUAP

O Componente de Serviço (SC – *Service Component*) da Aplicação do Usuário de uma sessão de acesso (*asUAP*) é definido para modelar uma variedade de aplicações no domínio do usuário (*User Domain*). Um asUAP representa uma ou mais destas aplicações e pode ser usado por usuários humanos e/ou outras aplicações no domínio usuário. Além de ser um SC relacionado a uma sessão de acesso ou uma sessão de serviço.

Quando um SC é relacionado a uma sessão de acesso, o asUAP permite a um usuário humano ou outra aplicação, fazer uso das capacidades (*capabilities*) de um Agente Provedor (PA - *Provider Agent*) ou Agente Pessoa (PeerA - *Peer Agent*), através de uma interface apropriada. Um asUAP relacionado à sessão de acesso suporta parte do domínio da sessão de acesso.

O asUAP provê as seguintes funcionalidades para:

- solicitar informações da autenticação do usuário, solicitadas pelo PA (ou PeerA) para configurar uma sessão de acesso com um Agente Usuário (UA – *User Agent*);
- solicitar ao usuário a criação de uma nova sessão de serviço;
- solicitar ao usuário unir-se a uma sessão de serviço existente; e
- avisar ao usuário de convites, o qual chega para PA ou PeerA.

O asUAP relacionado a uma sessão de acesso pode também suportar as seguintes capacidades opcionais, quando são também suportadas por PA ou PeerA:

- permitir ao usuário procurar por um provedor e registrar-se como um usuário dos serviços fornecidos pelo provedor; e
- permitir ao usuário procurar por serviços e identificar provedores fornecendo estes serviços.

Um domínio de usuário ou pessoa contêm um ou mais asUAPs relacionados à sessão de acesso, assim pode solicitar a PA ou PeerA o estabelecimento de uma sessão de acesso. Um ou mais asUAPs interagem com um PA ou PeerA para usar suas capacidades relacionadas à sessão de acesso.

Uma ocorrência asUAP pode suportar somente capacidades relacionadas à sessão de acesso ou somente capacidades relacionadas à sessão de serviço; ou pode

suportar ambos. Os asUAPs relacionados à sessão de acesso podem ser especializados por um domínio para interagir com um PA ou PeerA especializado.

2. PA

O Agente Provedor (PA) é um serviço independente do SC, definido como usuário terminal (*end-point*) de uma sessão de serviço. O PA é suportado no domínio atuando no papel de usuário no acesso, também suporta um usuário acessando seu UA e permite fazer uso de serviços através de uma sessão de acesso. O PA suporta a sessão de acesso, em conjunto com a sessão de acesso relacionada com os asUAPs e outras infraestruturas no domínio do usuário.

As capacidades suportadas pelo PA são:

- configurar um relacionamento confiável entre o usuário e o provedor (na sessão de acesso), interagindo com o Contexto de Gerenciamento de Segurança Multilateral (*SecMgmtCtxt*), e obtendo uma referência para o UA;
- dentro de uma sessão de acesso:
  - enviar solicitações (a partir de um usuário para UA) para criar novas sessões de serviço;
  - enviar solicitações para descobrir os serviços e seus modelos de sessão, de forma a garantir os direitos da Aplicação do Usuário na Sessão de Serviço (ssUAP – *service session User Aplication*) ou carregar este (para cada serviço, o provedor deve oferecer um serviço associado para carregar o ssUAP);
  - enviar solicitações para unir-se às sessões de serviços existentes;
  - receber convites para unir-se às sessões de serviço existentes (a partir de UA) e alertar ao usuário usando a sessão de acesso;
  - anonimamente fazer uso de serviços dos provedores;
  - desenvolver novos componentes dentro do domínio do usuário;
  - suportar o acesso para informações do terminal de configuração desde o domínio provedor; e
  - registrar-se para receber convites emitidos quando nenhuma sessão de acesso existir.

As operações suportadas pelo PA são serviços independentes. Para cada sessão de acesso simultânea um usuário tem um provedor, o que é uma ocorrência PA no domínio do usuário. Cada PA pode ser associado (através de uma sessão de acesso) com o mesmo UA, ou ocorrências separadas de UA. Um PA é sempre associado com um UA através da sessão de acesso (quando nenhuma sessão de acesso existir, um domínio usuário pode ainda suportar um PA, este pode ser usado para iniciar uma sessão de acesso e pode receber convites se registrado).

3. SecMgmtCtxt

O Contexto de Gerenciamento de Segurança (*SecMagmtCtxt*) é o ponto inicial do acesso para um domínio. Uma referência do *SecMgmtCtxt* é retornada para o domínio solicitador (PA ou PeerA) quando este deseja entrar em contato com o domínio. O *SecMgmtCtxt* suporta as seguintes capacidades:

- autenticar o domínio solicitador e configurar um relacionamento confiável entre os domínios (em uma sessão de acesso) que interagem com o PA ou PeerA;
- estabelecer uma sessão de acesso, mas permitindo ao domínio solicitador permanecer anônimo.

O *SecMgmtCtxt* suporta solicitações dos componentes PA/PeerA ao mesmo tempo. O PA/PeerA solicita entrar em contato com o domínio dando a referência para *SecMgmtCtxt*. Quando o PA/PeerA interage com o *SecMgmtCtxt* para estabelecer uma sessão de acesso com UA/PeerA, a referência para *SecMgmtCtxt* pode tornar-se inválida. Subseqüentemente, o *SecMgmtCtxt* pode entrar em contato com outro PA/PeerA. O *SecMgmtCtxt* confia nas interfaces sobre UA e PeerA para propósitos de inicialização e da interface sobre um servidor de autenticação.

Este contexto define o componente SA (*Security Agent*), que é componente de segurança multilateral responsável pela negociação entre os domínios do provedor e consumidor, o qual pode terminar com sucesso, sendo a sessão de comunicação estabelecida, ou com um erro, caso não seja possível negociar um mínimo de objetivos de segurança.

4. UA

O UA suporta as seguintes capacidades:

- dentro de uma sessão de acesso:
  - atua como um ponto de contato simples para o controle e gerência (criar/suspender/retomar/deletar) do ciclo de vida das sessões de serviço e sessões do usuário, considerando as restrições que possuem o subscritor e o usuário;
  - suspende/retoma sessões de serviço do usuário e sessões de serviços existentes. Este inclui o suporte para mobilidade da sessão;
  - gerencia as preferências dos usuários (escolha ou restrições) sobre o acesso de serviço e execução do serviço (isto é suportado iniciando uma sessão de serviços de um provedor específico);
  - resolve o ambiente de execução do serviço para o usuário, permitindo o uso de serviços desde diferentes tipos de terminais. Este requer informações de configuração de recursos do sistema do usuário (a qual inclui terminais e seus pontos de acesso sendo usados por ou disponíveis para o usuário; acesso para esta informação pode ser restringido pelo usuário/PA). Este inclui suporte para mobilidade pessoal;
  - registra usuários ao terminal para receber convites. Este inclui o suporte para mobilidade pessoal;
  - permite ao usuário definir políticas privadas/públicas (isto é, iniciando uma sessão de serviço de um provedor específico); e
  - negocia o modelo da sessão e conjunto de características suportadas pela sessão de serviço, de forma a interagir com o UAP no domínio usuário.
- aceita convites do usuário para unir-se a uma sessão de serviço; e
- libera convites para um terminal, previamente registrados pelo usuário por UA. Nenhuma sessão de acesso pode ser solicitada para permitir a liberação de convites.

O UA pode suportar as seguintes capacidades opcionais:

- executar ações no lado do usuário, quando o usuário não está na sessão de serviço com respeito a UA;
- iniciar uma sessão de acesso em relação a UA; e
- suportar a autenticação adicional do usuário. Isto pode adaptar-se ao usuário e ao contexto de uso.

5. SUB

O Componente de Gerenciamento de Subscrição permite o gerenciamento de subscritores, subscrições e usuários para todo o conjunto de serviços fornecidos pelo provedor. A funcionalidade principal oferecida por este componente é:

- criação, modificação, remoção e consulta de subscritores;
- criação, modificação, remoção e consulta de informações relacionadas a subscritores (associadas a usuários finais, grupos de usuários finais, etc.);
- criação, modificação, remoção e consulta de contratos de serviços (definição de perfis de serviços subscritos);
- recuperar listas de serviços, que estão disponíveis no domínio provedor ou os subscritores;
- recuperar o perfil do serviço para um usuário específico;
- criação e remoção de componentes de acesso; e
- notificação de mudanças para os componentes de acesso aos usuários afetados.

6. ssUAP

A Aplicação do Usuário da sessão de serviço interage com o PA e o USM (*User Service Session Manager*). A interação com PA permite ao ssUAP solicitar o início, recomeço ou união a uma sessão de serviço. Similar a PA pode invocar operações sobre S*ecMgmCtxt* para solicitar que o ssUAP inicie estes eventos.

7. SF

O Construtor de Serviço é um objeto específico do serviço, o qual gerencia o ciclo de vida dos objetos computacionais (CO – *Computational Object*) da sessão de serviço para um tipo de serviço.

Uma solicitação para criar uma sessão de serviço de um tipo de serviço em particular resultará na criação de uma ou mais instâncias de objetos (tipicamente, o USM e SSM). O SF criará e inicializará as instâncias de acordo as regras impostas na sua implementação. O SF retornará ao cliente uma ou mais referências às interfaces para aqueles componentes (o SF é usado para criar/gerenciar instâncias de todas as sessões de serviço relacionadas aos objetos definidos para USM, SSM e PeerUSM).

As solicitações para criar uma nova sessão de serviço são tipicamente feitas por UAs e PeerAs. O SSM é também cliente do SF, este pode solicitar ao SF criar um novo USM quando um novo usuário é unido a uma sessão existente ou pode solicitar a remoção/suspensão de objetos da sessão de serviço. Outro cliente do SF é o SLCM (*Service Life Cycle Management*), o qual inicializa e gerencia o SF. O cliente deve ter a referência da interface para o SF e emitir uma solicitação apropriada. O SF suporta mais de um tipo de serviço podendo prover interfaces separadas para cada tipo de serviço.

O SF suporta as seguintes funcionalidades:

- criar e inicializar objetos para um ou mais tipos de serviços sobre solicitação (este inclui a escolha do modelo da sessão suportado pela sessão de serviço, embora esta pode ser fixada pelo tipo de serviço);
- criar e inicializar um objeto (usualmente USM) para ser usado em conjunto com outros objetos (como SSM) criados por instâncias diferentes do construtor;
- continuar a gerenciar os objetos criados. Este pode prover uma lista de sessões gerenciadas, e pode limpar totalmente algumas sessões se solicitado; e
- suportar a suspensão/retomada de uma sessão de serviço.

Este pode opcionalmente incluir mecanismos para programar a ativação de uma sessão em uma data e hora específica.

O SF monta os recursos necessários para a existência de um componente quando este é criado. Portanto, o SF representa o escopo da alocação de recursos, o qual é o conjunto de recursos disponíveis pelo SF, além de, suportar uma interface que disponibilize ao cliente para restringir o escopo.

8. SSM

O Gerenciador da Sessão de Serviço suporta as seguintes funcionalidades:

- manter o monitoramento e controle de vários recursos compartilhados por múltiplos usuários em uma sessão de serviço. Isto pode ser feito para ter referências para outros objetos (como SSM) o qual realmente mantêm o contexto de uso para uma classe específica de recursos;
- manter o estado de um serviço e suportar a suspensão/retomada da sessão de serviço;
- suportar a inclusão/convite/modificação da ligação de fluxo (*stream binding*) e a participação do usuário neste;
- suportar a inclusão/convite/remoção de usuários a partir de a sessão de serviço para interagir com os UAs correspondentes;
- suportar a capacidade de negociação entre o usuário interagindo com os USMs. O SSM servirá como o centro de controle da construção de consensos (tais como o procedimento de votos); e
- suportar as capacidades de gerenciamento associadas com a sessão de serviço (por exemplo, a contabilidade).

9. USM

O Gerenciador da Sessão de Serviço do Usuário representa um participante em uma sessão e portanto oferece para o ssUAP o controle da sessão e controle específico de serviço. A participação de um participante na sessão de serviço pode também ser controlada pela sessão de acesso via o UA. O UA também controla a liberação de eventos contábeis enviados pelo USM ao UA, invocado sobre a interface *i_AccountingPushMgmt*.

As interações entre o USM e SSM são numerosas, dado que o USM deve invocar solicitações de controle de sessão sobre SSM e, o SSM deve indicar e informar o USM das mudanças no estado da sessão. O USM também suporta a interface *i_AccountingPush*, permitindo-lhe enviar informações contábeis.

O USM tem interações limitadas com o componente SF. Quando instanciado, o USM é inicializado com as propriedades necessárias de uso da sessão utilizando as

operações *i_Init* a qual também retorna a referência apropriada da interface USM para as interações com o ssUAP e UA.

OBS: as interfaces *i_AccountingPushMgmt, i_AccountingPush* e *i_Init* serão descrita em detalhes no Anexo 1.

10. SLCM

O Gerenciamento do Ciclo de Vida do Serviço provê as funcionalidades solicitadas para controlar o tipo de serviço e ocorrências do ciclo de vida do serviço. Assim como também permite a configuração da rede de serviço.

O SLCM é um serviço independente do SC, o qual gerencia o ciclo de vida dos serviços TINA. Este provê as seguintes funcionalidades:

- desenvolvimento, configuração e retirada da rede de serviço;
- gerenciar o tipo de serviço; e
- desenvolver, configurar, ativar, desativar e retirar ocorrências de serviços.

Este tem uma interação forte com os repositórios do ambiente de Processamento Distribuído (DPE – *Distributed Processing Environment*) adjacentes, e serviços e outras facilidades, semelhantes às facilidades de gerenciamento do nó, que permite o gerenciamento do nó de serviço que compõe a rede de serviço.

11. PeerA

O PeerA é um serviço independente do SC que representa um usuário (*Peer*) em outro domínio Peer´s. Deve suportar as capacidades externas combinadas do UA e do PA. Além destas capacidades, o PeerA deve suportar funcionalidades de gerenciamento interdomínio, como por exemplo, transferência de informação contábil.

É possível que o PeerA necessite suportar mecanismos para a troca de informação da interface de sessão entre a sessão de serviço em domínios diferentes, já que o nível de confiança entre dois provedores deve afetar o nível de detalhe visível para outros domínios. Ainda, este nível de confiança é claramente independente do serviço.

12. PeerUSM

O Gerenciador da Sessão de Uso do Peer é um SC o qual suporta o relacionamento *peer-to-peer* entre sessões de serviço em diferentes domínios.

O PeerUSM permite a uma sessão de serviço interagir com outra sessão de serviço em outro domínio. Ambas sessões de serviço são Peers, e ambas suportam uma PeerUSM para interagir até o fim.

13. RBACMgmtCtxt

Este é o contexto que define um espaço de segurança para uma sessão de serviço. Este espaço de segurança é similar a um *container*[10] e é povoado por várias classes de papéis[11]. Este contexto define o componente ACA (*Access Control Agent*) que permite emitir um certificado de papel, como definido no item 5.2.3.

6.2. Descrição das Interfaces que Interagem com cada Componente

Neste item descreveremos os propósitos de cada interface que interage com os SC no serviço do modelo de segurança para o gerenciamento de contabilidade segundo o padrão da Arquitetura TINA (ABARCA, 1998). A Figura 6.2 mostra o diagrama completo do modelo de segurança, onde se visualizam os domínios Consumidor (*Consumer*), Provedor (*Retailer*) e Provedor Terceirizado (*3PTy*) com seus respectivos componentes, interfaces e os contextos de contabilidade e de segurança (Obs.: uma melhor visualização da Figura 6.2 encontra no Anexo 2).

No Anexo 1, mostra-se de uma forma mas detalhadas as interfaces definidas na Figura 6.2, com o intuito de dar uma melhor compreensão e visualização das interfaces que interagem com os respectivos componentes computacionais e de segurança. Além de, descrever completamente cada interface que interage no SSGCT.

Até aqui foi definido e descrito o modelo de segurança da arquitetura de contabilidade TINA, na qual foram atingidos os seguintes objetivos:

- identificar e analisar as interfaces que interagem com os componentes de cada domínio;
- representar o modelo proposto através de um modelo geral, no qual estejam representados tanto os componentes como suas interfaces, formando um diagrama total do processo contábil; e

[10] Conceito que define que um objeto pode conter outros objetos.
[11] Neste espaço os papéis interagem com cada outro e são dinamicamente limitados diferenciando claramente um objeto ou uma interface, quando os objetos têm múltiplas interfaces.

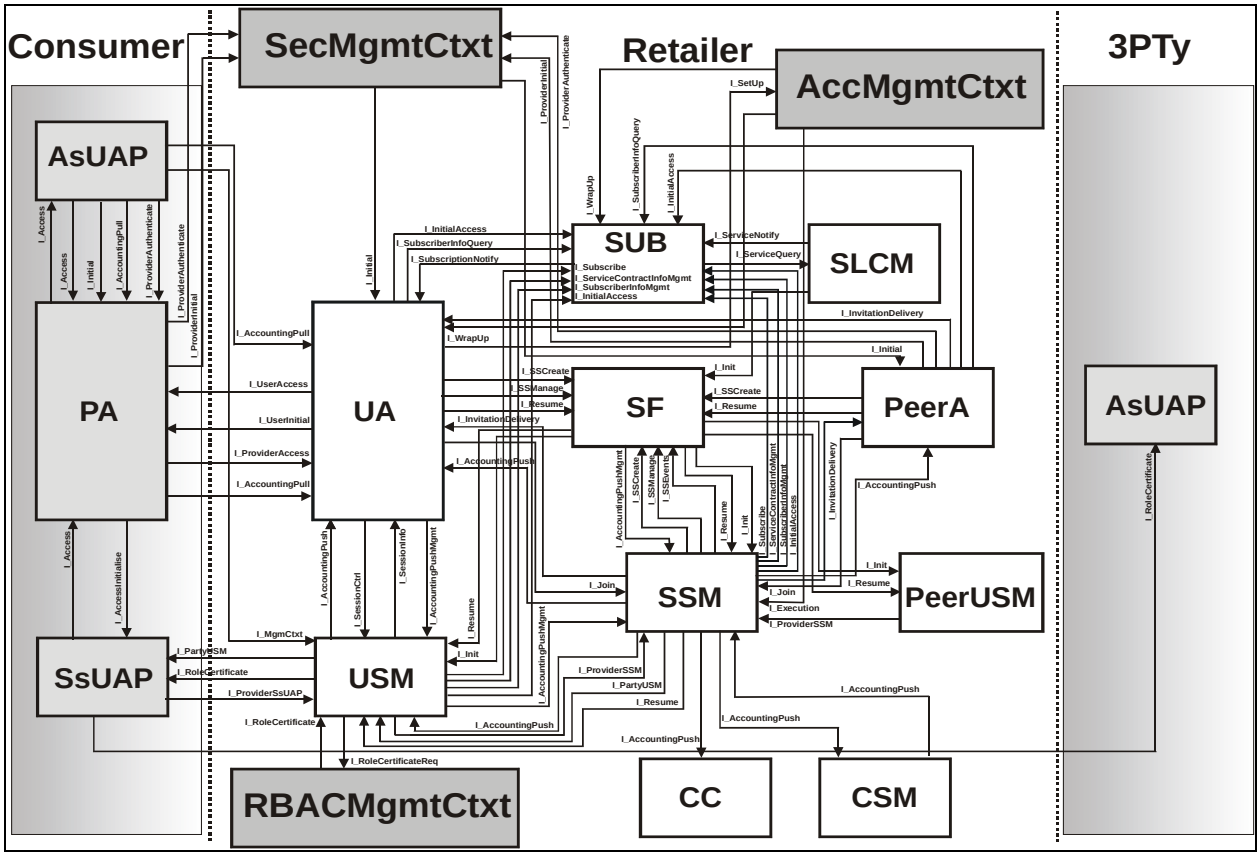


Figura 6.2 – Diagrama Esquemático do Modelo de Segurança no Gerenciamento de Contabilidade com seus respectivos Componentes e Interfaces.

- por último, deve-se segundo (VAN LE, 2002), verificar e validar corretamente o modelo estrutural e comportamental da arquitetura de contabilidade, para garantir a correta interação entre os componentes do sistema, usando:
    1. uma Linguagem de Descrição Formal, como LOTOS[12]; e
    2. a implementação de um protótipo de forma a validar o modelo proposto através de uma simulação de serviços oferecidos pelo fornecedor, em um ambiente de gerenciamento de contabilidade em tempo real, que demonstrará como são estabelecidas as políticas de segurança na fase de negociação usando o modelo de segurança multilateral e a fase de uso do serviço usando o modelo RBAC que

[12] LOTOS (*Language of Temporal Ordering Specification*) é uma técnica de especificação formal estruturada em métodos algébricos para a representação de processos. Esta ferramenta permite definir aplicações na área de gerenciamento de redes que são orientadas a objetos (BRINKSMA, 1988).

definirá em um contexto dinâmico para os papéis do serviço disponíveis ao consumidor com *billing on-line*.

6.3 Conclusão

Hoje em dia, o avanço e crescimento de aplicações multimídia através da Internet, têm motivado aos pesquisadores a propor novos serviços com a finalidade de fornecer aplicações aos usuários com requisitos de serviço de alta qualidade. Nesse sentido, apresentamos neste capítulo como TINA tem contribuído com tais requerimentos e como através desta arquitetura surgiu no Laboratório de Redes e Gerência uma linha de pesquisa baseado no modelo de Contabilidade, que permitiu o desenvolvimento de vários trabalhos, os quais validaram os conceitos e princípios de TINA.

Continuando nesta linha de pesquisa se apresenta um modelo de segurança que incorpora políticas e mecanismos num ambiente de Segurança Multilateral e do RBAC no gerenciamento de Contabilidade seguindo os padrões estabelecidos pelo TINA-C.

Por último, seguindo as recomendações da literatura, o modelo será validado através de um Linguagem de Descrição Formal, neste caso LOTOS através da sua ferramenta CADP (*CAESAR/ALDEBARAN Developments Package*) e a implementação de um protótipo em uma linguagem Orientada a Objeto, neste caso através de JAVA, que permitirá validar a arquitetura proposta.

7. ESPECIFICAÇÃO E VALIDAÇÃO FORMAL DO MODELO DE SEGURANÇA

7.1 Introdução

Este capítulo apresenta uma metodologia para a especificação e validação formal de sistemas distribuídos. A Técnica de Descrição Formal (TDF) utilizada é o padrão ISO 8807-LOTOS - *Language of Temporal Ordering Specification* (BRINKSMA, 1988). O principal objetivo do emprego desta técnica de alto rigor matemático é fornecer a prova formal de correções do sistema.

O modelo de segurança SSGCT apresentado na seção 6.2, é especificado formalmente, utilizando-se uma abordagem de refinamentos sucessivos (VISSERS, 1988), a qual permite validar seu desenvolvimento até sua especificação formal final. A validação do sistema utilizará a última versão beta da ferramenta *Eucaliptus ToolSet* 2.5/CADP 2003-e (GARAVEL, 2004) em ambiente Linux versão RedHad 8.0.

A próxima seção mostra uma breve descrição da Técnica de Descrição Formal LOTOS.

7.2 A Técnica de Descrição Formal LOTOS

Para especificar rigorosamente sistemas distribuídos, é conveniente utilizar metodologias de descrição formal a fim de prover maior confiabilidade a tais sistemas, geralmente complexos. As linguagens de especificação formal são baseadas em teorias matemáticas e estão associadas com métodos de especificação precisos, completos, consistentes e não ambíguos. Técnicas de Descrição Formal (TDFs) servem para definir os aspectos de comportamento e também os aspectos de dados de sistemas (PIRES, 1994 e QUEIROZ, 1994).

Pode-se destacar como características de uma boa técnica de descrição formal:

1. **A descrição mínima e precisa**: que permite fornecer meios para uma descrição precisa de todas as propriedades com um mínimo de informações extra;

2. **A facilidade de entendimento**: que permite prover descrições facilmente compreensíveis;
3. **O poder de expressão**: que permite ter a capacidade para exprimir uma quantidade muito grande de propriedades importantes para a descrição de sistemas;
4. **A fundamentação matemática**: que permite ter os seus elementos de definição e de especificação baseados em um modelo matemático; e
5. **A flexibilidade**: que permite fornecer meios para uma descrição suficientemente flexível para adaptar-se às pequenas alterações nas propriedades a serem especificadas (NOTARE, 2000).

LOTOS (BRINKSMA, 1988) e ESTELLE (*Extended Finite-State Machine Language*) são TDFs normalizadas pela ISO (*International Organization for Standardization*), enquanto SDL (*Specification and Description Language*) é uma TDF normalizada pelo CCITT (*Consultative Committe for International Telegraph and Telephone*), hoje ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*).

Neste trabalho optou-se pela metodologia LOTOS pelas seguintes razões:

- enquanto SDL e ESTELLE são baseados em linguagens de programação (Chill e Pascal, respectivamente), LOTOS é independente da linguagem de programação;
- existem ótimas ferramentas para a validação de especificações LOTOS, tais como CADP, com ambiente gráfico Eucaliptus Toolset para o ambiente Linux, desenvolvido pelo equipe VASY (Validação de Sistemas) do INRIA/Grenoble, com o qual o Laboratório de Rede e Gerência (LRG) e UNICS (Centro Universitário Diocesano do Sudoeste de Paraná) mantêm relações internacionais bilaterais;
- LOTOS é um padrão internacional (ISO 8807); e
- O nível de abstração é uma das mais importantes características para o desenvolvimento de sistemas. LOTOS apresenta além do elevado nível de abstração, outras características inerentes às TDF como o poder de expressão e a estruturação de especificações (QUEIROZ, 1994).

LOTOs é uma TDF que reúne duas álgebras: a primeira álgebra é utilizada para a descrição de aspectos de comportamento e corresponde a uma extensão de CCS – *Calculus of Communicating Systems* (Cálculo de Sistemas Comunicantes) (MILNER, 1980); a segunda álgebra é utilizada para a descrição dos aspectos de dados e corresponde à linguagem ACT ONE (EHRIG, 1985).

Em LOTOS Básico (a parte de LOTOS que representa apenas os aspectos de comportamento dos sistemas) uma especificação é constituída por uma hierarquia de definições de processos.

Pode-se ver um processo como uma caixa preta dotada de portas para a comunicação com o ambiente e com outros processos, via eventos de comunicação. Eventos de comunicação são ações que ocorrem nas portas de comunicação. Ações que não são observáveis externamente são chamadas de ações internas e são representadas pelo símbolo *i*.

A definição de especificações e de processos LOTOS tem o formato descrito a seguir:

specification <id> [lista_de_parâmetros>]:<funcionalidade>
behaviour <expressão_de_comportamento_da_especificação>
where process <id> [<lista_de_parâmetros>]:<funcionalidade>:=
<expressão_de_comportamento>
endproc
...
endspec

onde:

- **id**: identifica o nome da especificação ou o nome de um processo;
- **lista_de_parâmetros**: refere-se às portas de comunicação;
- **funcionalidade**: de um processo pode ser ***exit*** se ele termina com sucesso, habilitando um processo subseqüente, ou então, a **funcionalidade** pode ser ***noexit*** no caso de processos recursivos (infinitos), por exemplo; e
- **expressão_de_comportamento**: refere-se ao comportamento da especificação ou do processo.

7.2.1 Operadores LOTOS

Uma expressão de comportamento em LOTOS relata o que pode ser observado em termos de eventos. Nela podem ser usados operadores como:

;	seqüência de eventos;
[]	escolhas indeterminísticas entre comportamentos;
| | |	composição de processos independentes;
| |	composição de processos dependentes;
| **[** **]** |	composição geral;
hide . . . in . . .	ocultação de eventos;
>>	habilitação de processos; e
[>	interrupção de processos.

O operador **;** indica seqüenciamento de eventos, permitindo expressar que após a ocorrência de um primeiro evento ocorre um segundo evento. Por exemplo: *a1*; *a2*. Neste caso, o evento *a2* somente ocorre após a ocorrência de *a1*. O operador **;** também pode ser usado para expressar seqüenciamento de comportamentos. Por exemplo: *B1*; *B2*.

O operador **[]** indica uma escolha indeterminística entre dois comportamentos. Por exemplo: *B1***[]***B2*. Neste caso, após a escolha indeterminística, pode-se observar o comportamento descrito por *B1* ou então o comportamento descrito por *B2*.

Os operadores | | |, | | e | **[** **]** | possibilitam a representação de processos concorrentes. Com o operador **|** **|** **|** representam-se os processos executados concorrentemente que evoluem sem sincronização entre si. No caso de processos que compartilham nomes de eventos, esses sincronizam-se com seus ambientes comuns, mas não um processo com outro. Por exemplo: *calcula_nro_par*[a, b] **|** **|** **|** *calcula_nro_primo*[c, d]. Neste caso ambos os processos são executados em paralelo, mas sem sincronizarem suas portas entre si, isto é, os eventos nas portas a, b, c e d podem ocorrer sem que um processo interfira no outro.

Com o operador | | representam-se os processos executados concorrentemente que evoluem sincronizadamente e que existe dependência entre eles. Por exemplo: *calcula_nro_par*[a, b] **|** **|** *calcula_nro_primo*[b, c]. Neste caso os processos são executados em paralelo, porém somente o evento compartilhado na porta b pode

ocorrer. Nas portas a e c não podem ocorrer nenhum evento, pois não são portas compartilhadas.

Com o operador | **[ ]** | representam-se os processos executados concorrentemente que evoluem sem sincronização salvo onde há portas compartilhadas pelos processos. Tais portas são explicitadas pelo operador | **[ ]** |. Por exemplo: a expressão A | [a_1, a_2, ...,a_n] | B é usada quando os dois processos devem sincronizar-se somente com respeito às portas a_1, a_2, ...,a_n.

Com o operador **hide . . . in . . .** representa-se a ocultação de eventos, tornando-os invisíveis. Esta ocultação é importante para fins de análise e verificação de equivalências.

O operador >> representa seqüenciamento de processos. Em *B1 >> B2* o processo *B2* só é executado após o término com êxito do processo B1. Exemplo: *recebe*[. . .] >> *responde*[. . .]. Neste caso o processo *responde* só será executado após o término com êxito do processo *recebe*.

O operador [> representa interrupção de processo. Em B1 [> B2 o processo B2 pode interromper a qualquer instante o processo B1. Exemplo: *transmite_serviço*[. . .] [> *interrompe_serviço*[. . .]. Neste exemplo, o processo *interrompe_serviço* pode interromper o processo *transmite_serviço* a qualquer momento (antes ou durante a execução do processo *transmite_serviço*).

7.2.2 Processos LOTOS *stop, exit* e Infinitos

Os processos ***stop*** e ***exit*** pertencem à linguagem LOTOS. Nenhum deles possui qualquer evento. O processo ***stop*** representa a ausência completa de atividade. O processo ***exit***, por sua vez, indica uma terminação com sucesso, habilitando a realização de algum comportamento subseqüente. Exemplo:

process le_escreve[in, out] : ***noexit*** :=
    leitura[in] >> escrita[out]
    where
    process leitura[in] : ***exit*** := in; ***exit endproc***
    process escrita[out] : *exit* := in; ***stop endproc***
endproc

Nesse caso, após a execução do processo **exit** o controle é transferido para o estado inicial do processo escrita[out].

Outro meio de expressão de LOTOS é a chamada recursividade de processos. Tais chamadas permitem representar comportamentos infinitos. Exemplo:

process ciclo[a1, a2] : ***noexit*** :=

a1; a2; ciclo[a1, a2]

endproc

Neste exemplo o processo ciclo repete a seqüência de eventos a1; a2 infinitamente.

7.2.3 Modelo Semântico

O modelo da TDF LOTOS baseia-se em sistemas de transições rotulados (LTS – *Labelled Transition System*). Sistemas de transições rotuladas são constituídos de transições etiquetadas entre estados que evoluem no decorrer do tempo. As transições representam eventos observáveis ou eventos internos (BRINKSMA, 1988 e MILNER, 1980).

Um sistema de transição rotulada *Sys* é uma quádrupla <*S, Act, T,* S_o>, onde:

- *S* é um conjunto não vazio, e seus elementos referem-se aos estados;
- *Act* é um conjunto cujos elementos referem-se às ações;
- *T* é um conjunto de relações de transições que contém precisamente uma relação $a \rightarrow \subseteq S \times S$ para cada $a \in Act$; e
- S_o é o estado inicial de *Sys*.

Uma transição de um sistema de transições é uma tripla <*corrente*, *a, next*>, tal que <*corrente, next*> $\in a \rightarrow$. Sejam *B1* e *B2* expressões de comportamento, $B1 \rightarrow^a B2$ significa que do estado onde se tem a expressão de comportamento B1, após a ocorrência de um evento *a*, alcança-se o estado onde se tem a expressão de comportamento *B2*. Através desse modelo de transição, a semântica dos operadores LOTOS pode ser definida.

7.3 Especificação dos Serviços SSGCT

A seguir serão descritas, passo a passo, as especificações dos serviços SSGCT (Sistema de Segurança de Gerenciamento de Contabilidade TINA), começando de um

alto nível de abstração até a definição da especificação formal do modelo de segurança para um gerenciamento de contabilidade TINA.

7.3.1 Especificação Geral

No nível mais alto de abstração, o sistema SSGCT pode ser visto como uma caixa preta, formada pelos domínios Consumidor (*Consumer*) e Provedor (*Retailer*). As duas portas de comunicação entre os domínios são definidas de acordo com o ponto de referência padrão no modelo de negócios TINA, denominado Ret-RP (*Retailer Reference Point*) (descrito no item 2.2) que estabelece as portas de **acesso** e **uso**, como mostra a Figura 7.1.

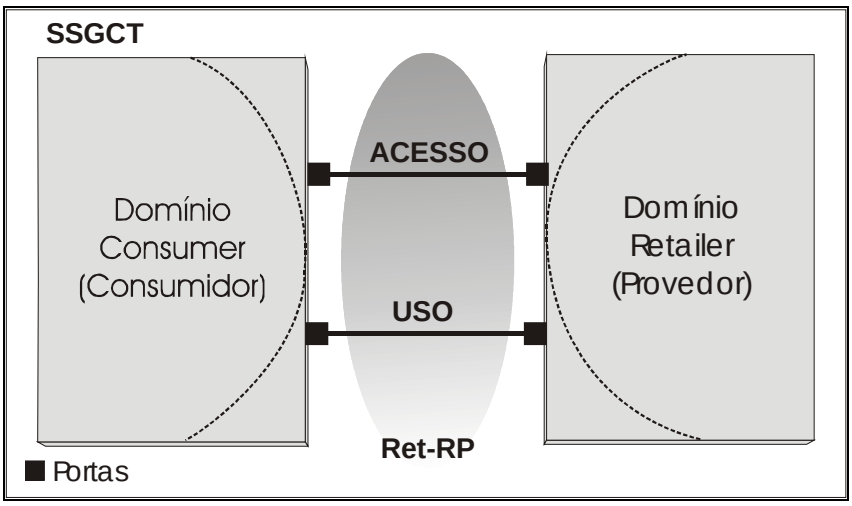


Figura 7.1 – Representação em Alto Nível do SSGCT.

A definição do ponto de referência é uma especificação semi-formal do relacionamento de negócios entre o papel de negócio Consumidor e o papel de negócio Provedor. O Ret-RP é separado em duas partes: de acesso e de uso, as quais definem um alto grau de abstração às portas de comunicação.

Para o Ret-RP, a parte de acesso permite descrever como o papel de negócio Consumidor acessa um papel de negócio Provedor para fazer uso dos serviços por ele fornecido; a parte de uso descreve as interações entre os papéis de negócios envolvidos durante o uso de um serviço. Cada parte é controlada independentemente (FARLEY, 1998). A independência destas partes (para fins do caso de estudo denomina-se de portas) permitem definir os contextos de segurança e de contabilidade também de forma independente, isto permitirá que os contextos de segurança sejam criados num ambiente

sem possibilidade de incompatibilidade com respeito as políticas de segurança, como definidas no item 5.2.

Desse modo a porta de acesso é utilizada para fazer uso de serviços e de negociação dos mecanismos de segurança dos domínios Consumidor e Provedor. A porta de uso permite usar o serviço selecionado previamente definido os papéis dentro do contexto RBAC, disponibilizando um contexto de gerenciamento de contabilidade com *billing on-line* confiável e em tempo real.

A Figura 7.2 ilustra o comportamento do sistema definido através da especificação LOTOS.

```
specification SsgctService [acesso, uso] : noexit behaviour
                processo [acesso, uso]

where

        process processo [acesso, uso] : noexit :=
                acesso; uso; processo [acesso, uso]
        endproc

endspec
```

Figura 7.2 – Especificação LOTOS de SSGCT.

O comportamento do sistema SSGCT é definido pelo processo *SsgctService*, o qual executa uma ação na porta de acesso, para permitir a negociação dos contextos de segurança e de serviço.

A segunda ação acontece na porta de uso e é realizada após a ação na porta de acesso. A ação na porta de uso permitirá definir os papéis relacionados ao tipo de serviço sendo fornecido para estabelecer um *billing on-line*.

A primeira ação pode ser chamada recursivamente pelo fato de que um Consumidor pode acessar mais de um serviço, podendo estabelecer um outro contexto.

A especificação do nível de abstração do SSGCT corresponde a uma formalização das solicitações de uso de serviço entre o Consumidor e o fornecimento do serviço do Provedor. A partir desta especificação inicial deverão ser realizados sucessivos refinamentos para utiliza-los como prova da correção da especificação final do sistema.

7.3.2 Especificação Formal do Processo de Acesso: definição do Contexto de Gerenciamento de Segurança Multilateral - *SecMgmtCtxt*

O modelo da Figura 7.3 apresenta o domínio de negócio do Consumidor e Provedor com seus respectivos componentes e interfaces de acordo com os padrões de TINA para o processo de Acesso. Neste modelo é incorporado o contexto de gerenciamento de segurança multilateral (*SecMgmtCtxt)* para estabelecer a negociação das políticas de segurança entre os domínios.

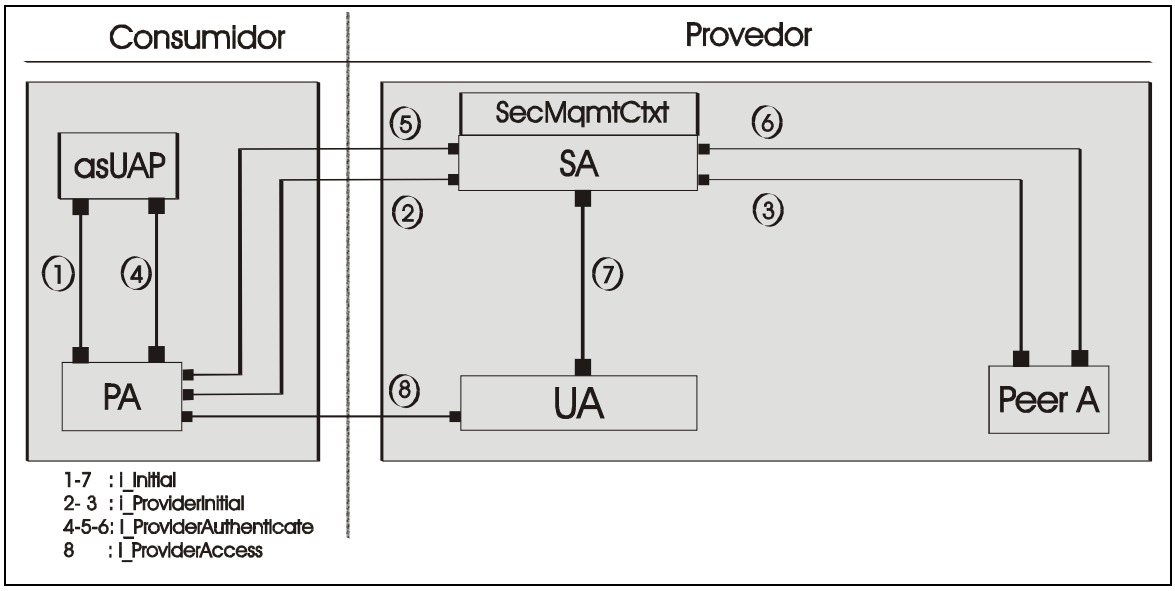


Figura 7.3 – Especificação Formal do Processo de Acesso.

O processo de acesso inicia-se através da seguinte seqüência de eventos:

1: a porta *i_Initial* inicia o estabelecimento de uma sessão de acesso permitindo ao Consumidor contatar-se e logar-se com um Provedor como usuário conhecido (PeerA) ou usuário anônimo (PA);

2 – 3: a porta *i_ProviderInitial* permite ao Consumidor e ao Provedor iniciar uma negociação para estabelecer um contexto de segurança, definido através do componente SA (*Security Agent*) do *SecMgmtCtxt*;

4 - 5 – 6: a Porta *i_ProviderAuthenticate* permite realizar a autenticação entre os domínios, neste nível o método de autenticação já deve estar definido através da porta *i_ProviderInitial*;

7: a porta *i_Initial* vai permitir referenciar a porta *i_ProviderAccess* que é necessária para dar início a uma sessão de serviço TINA subscrito ou a escolher; e

8: a porta *i_ProviderAccess* permite iniciar ou retomar a participação numa sessão de serviço TINA.

O comportamento do processo de acesso pode ser especificado formalmente em LOTOS como mostra a Figura 7.4.

```
specification SecMgmtCtxt [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                           i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2,
                           i_ProviderAccess] : noexit
behaviour
        negociacao [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                    i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_ProviderAccess]

where

    process negociacao [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                        i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_ProviderAccess] : noexit
:=
        i_Initial1; (i_ProviderInitial1; i_ProviderAuthenticate1; i_ProviderAuthenticate2; i_Initial2;
                        i_ProviderAccess; negociacao [i_Initial1, i_ProviderInitial1, i_ProviderInitial2,
                        i_ProviderAuthenticate1, i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2,
                        i_ProviderAccess]
        []
        i_ProviderInitial2; i_ProviderAuthenticate3; i_Initial2; i_ProviderAccess; negociacao [i_Initial1,
                        i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1, i_ProviderAuthenticate2,
                        i_ProviderAuthenticate3, i_Initial2, i_ProviderAccess])
    endproc

endspec
```

Figura 7.4 – Especificação Refinada do Processo de Acesso.

O processo de negociação "**negociacao**" que corresponde à especificação *SecMgmtCtxt* é combinado com o operador LOTOS de escolha indeterminística entre comportamentos "[]". Este operador indica opções entre comportamentos. Neste caso; temos uma primeira opção de comportamento estabelecida no domínio do Consumidor (*Consumer* na Figura 7.3, através do componente PA) como um usuário anônimo através das portas *i_ProviderInitial1; i_ProviderAuthenticate1; i_ProviderAuthenticate2; i_Initial2; i_ProviderAccess; negociação*[...] ou uma segunda opção que corresponde a um usuário conhecido ou subscrito (através do componente PeerA no domínio *Retailer*, como mostra a Figura 7.3) através das portas *i_ProviderInitial2; i_ProviderAuthenticate3; i_Initial2; i_ProviderAccess; negociação*[...] que permitirão negociar e estabelecer o contexto de segurança multilateral dos respectivos domínios e a escolha do tipo de serviço. É definido um

ambiente recursivo do processo de negociação devido a que vários usuários podem estar estabelecendo seu espaço de segurança para um determinado serviço.

7.3.3 Especificação Formal do Processo de Uso: definição do Contexto de Gerenciamento de Segurança RBAC - *RBACMgmtCtxt*

Após estabelecido o contexto de segurança multilateral através da negociação entre os domínios e especificado formalmente o processo de acesso, deve-se iniciar o processo de uso, que em primeira instância deve definir o contexto de gerenciamento de papéis que devem ser estabelecidos no domínio Provedor antes de fazer uso do serviço e realizar a execução dos eventos contábeis do serviço fornecido.

Neste caso, o processo de uso pode se dividir em duas etapas funcionais importantes: uma etapa de definição do contexto de gerenciamento de papéis (*RBACMgmtCtxt*) e a outra etapa de definição do contexto de gerenciamento de contabilidade TINA (*AccMgmtCtxt*).

7.3.3.1 Definição da Especificação Formal para *RBACMgmtCtxt*

Na Figura 7.5 pode-se observar os componentes que participam no contexto de segurança de papéis e as portas que interagem para a definição dos papéis no início de uma sessão de serviço.

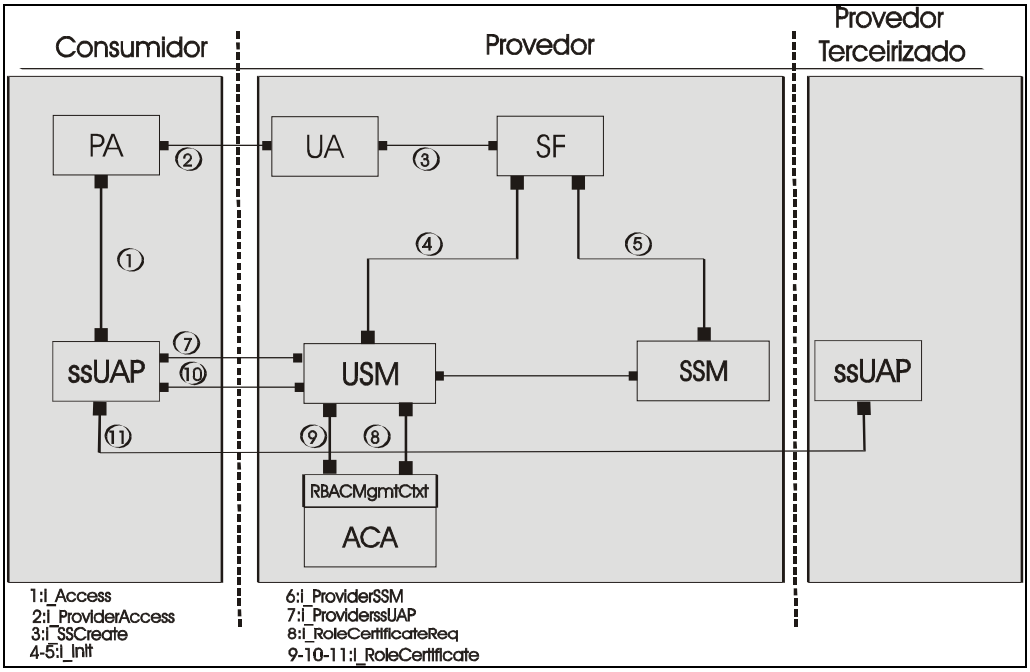


Figura 7.5 – Definição do Contexto de Segurança *RBACMgmtCtxt*.

O processo de definição de papéis inicia-se através da seguinte seqüência de eventos:

1: a porta *i_Access* permite iniciar uma sessão de serviço;

2: a porta *i_ProviderAccess* permite ao usuário acessar ao serviço subscrito;

3: a porta *i_SSCreate* permite criar os objetos de uma sessão de serviço, como por exemplo USM, SSM;

4 – 5: a porta *i_Init* permite iniciar a sessão de serviço e selecionar o tipo de serviço;

6: a porta *i_ProviderSSM* permite monitorar os eventos relacionados com o uso;

7: a porta *i_ProviderssUAP* permite finalizar uma sessão de serviço;

8: a porta *i_RoleCertificateReq* permite a solicitação de um certificado de papel ao contexto *RBACMgmtCtxt* através do componente ACA (Agente de Controle de Acesso), este contexto foi definido no item 5.2.2.1;

9: o ACA emite o certificado ao USM pertencente ao usuário solicitador;

10: o certificado é passado ao ssUAP do usuário final; e

11: o usuário ativa o serviço definido através do papel correspondente ao tipo de serviço selecionado usando o certificado e ativando o *billing* (papel) correspondente ao serviço solicitado pelo usuário.

A Figura 7.6 ilustra o comportamento do contexto de gerenciamento de papéis como definido no item 5.2.2.1 (espaço de segurança), especificado formalmente em LOTOS.

A especificação do processo *RBACMgmtCtxt* utiliza o operador de ocultação de portas **hide ... in** para as portas *i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2 e i_RoleCertificate3* em razão a que a emissão de certificados de papéis deve ser não observável para o domínio Consumidor e interno através do operador "**i**" para o domínio Provedor (ver a Figura 7.5); como mostra o processo de espaço de segurança "**espacoseguranca**", onde é definida a seqüência das portas para estabelecer o contexto de segurança para a definição dos certificados de papéis para um determinado tipo de serviço e seu respectivo *billing*.

```
specification RBACMgmtCtxt [i_Access, i_ProviderAccess, i_SSCreate, i_Init1, i_Init2, i_ProviderSSM,
                    i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2,
                    i_RoleCertificate3] : noexit
behaviour

hide i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2, i_RoleCertificate3 in
            espacoseguranca [i_Access, i_ProviderAccess, i_SSCreate, i_Init1, i_Init2, i_ProviderSSM,
                    i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2,
                    i_RoleCertificate3]
where

      process espacoseguranca [i_Access, i_ProviderAccess, i_SSCreate, i_Init1, i_Init2, i_ProviderSSM,
                    i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2,
                    i_RoleCertificate3] : noexit :=
            i; i_Access; i_ProviderAccess; i_SSCreate; i_Init1; i_Init2; i_ProviderSSM;
                    i_ProviderssUAP; i_RoleCertificateReq; i_RoleCertificate1; i_RoleCertificate2;
                    i_RoleCertificate3; espacoseguranca [i_Access, i_ProviderAccess, i_SSCreate, i_Init1,
                    i_Init2, i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1,
                    i_RoleCertificate2, i_RoleCertificate3]
      endproc

endspec
```

Figura 7.6 – Comportamento do Contexto Especificado Formalmente do Processo de USO.

7.3.3.2 Definição da Especificação Formal para *AccMgmtCtxt*

Antes de proceder a representação dos componentes do gerenciamento de contabilidade TINA, ilustraremos o uso do gerenciamento num cenário de serviço TINA típico e os relacionamentos dos componentes de serviço/rede.

A Figura 7.7 mostra o escopo geral do gerenciamento de contabilidade e *billing* num serviço TINA. Neste cenário dois UAPs no domínio do usuário se comunicam com outro através do enlace de um fluxo bidirecional. No cenário da Figura 7.7 tem se as seguintes suposições:

1. **Criação da sessão de serviço**: componentes de serviço como UA, USM, SSM, etc., já são criados e localizados.
2. **Configuração dos componentes de recursos de redes**: os componentes de recursos de rede como CC, LNC, etc., já são criados e localizados. Para finalidade de simplicidade, a Figura 7.7 não mostra todos os componentes.
3. **Configuração do enlace de fluxo**: necessariamente o ponto final de fluxo de rede (NFEP – *Network Flow End Point*) e o ponto final de fluxo (SFEP – *Stream Flow End Point*) já são fornecidos e limitados para o enlace de fluxo.

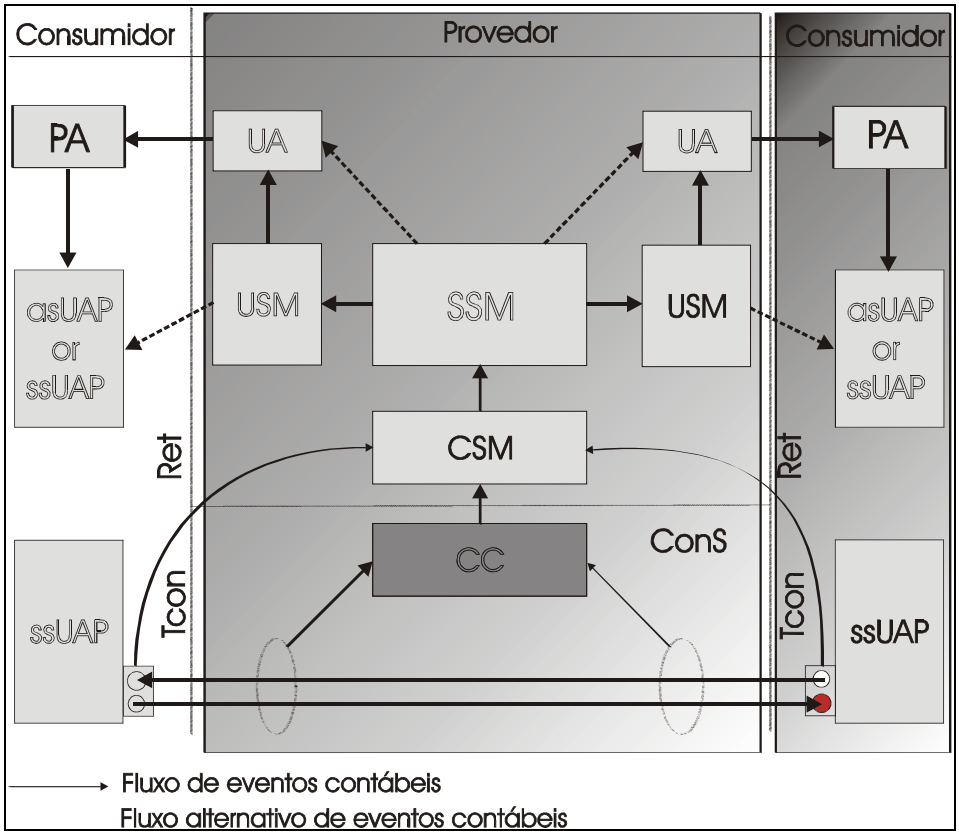


Figura 7.7 – Visão Geral do Gerenciamento de Contabilidade num Serviço TINA (ABARCA, 1998).

Embora todos estes passos são parte do gerenciamento de contabilidade, eles não se relacionam diretamente com o uso contábil, eles são mais utilizados para o gerenciamento de falhas e para propósitos de manutenção de sistemas (ABARCA, 1998).

A Figura 7.8 ilustra um exemplo da contabilidade de um provedor terceirizado. A estrutura do gerenciamento de contabilidade é exatamente similar à Figura 7.7, exceto que o fluxo de eventos contábeis (e o *billing*) são reservados para o ponto de referência 3Pty.

Assume-se que o conceito de transação de serviço é, por exemplo, informações *billing* que podem estar relacionadas com a monitoração da performance durante a transação. Assim o exemplo adota a seguinte suposição:

1. os dois usuários estão sobre duas transações de serviço separadas com respeito ao Provedor (Retailer), cujo contexto é passado através do Ret;

2. o Provedor está sobre uma transação de serviço, passado através da conectividade, cujo contexto é passado através do ConS. Esta transação de serviço corresponde a uma relação Cliente –Servidor entre a conexão de fluxo (SFC – *Stream Flow Connections*) e a conexão de fluxo de rede (NEC – *Network Flow Connection*) sobre o enlace de fluxo; e

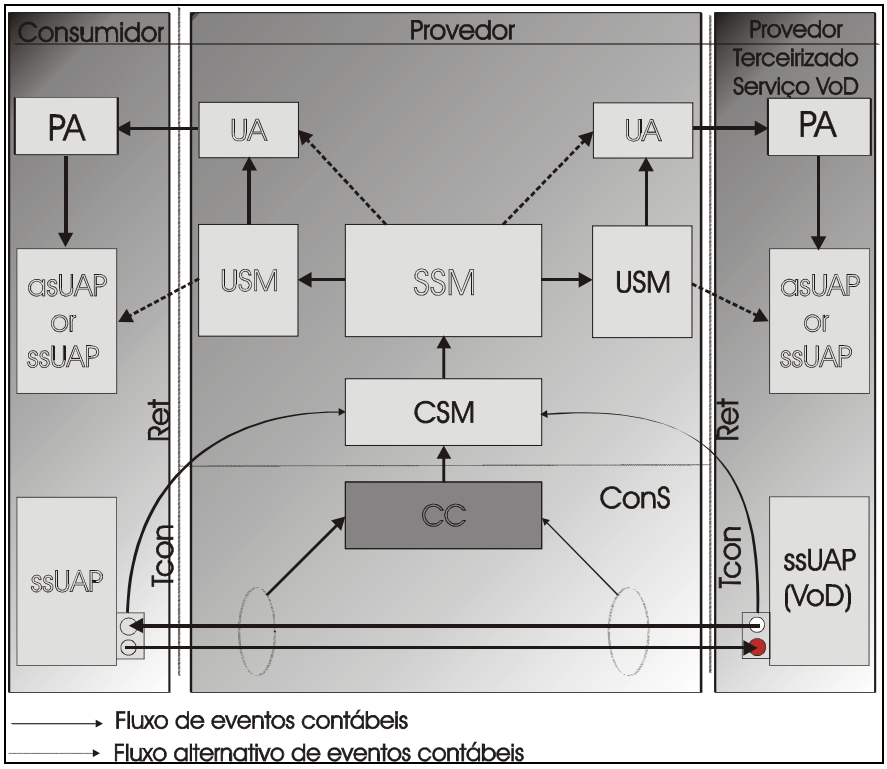


Figura 7.8 – Um exemplo de uma Contabilidade de um Provedor de Serviço Terceirizado (ABARCA, 1998).

3. o Provedor está atuando como um agente *billing* para o provedor de serviço de comunicação. Embora o Provedor é possível que não atue como um agente, é provável que na maioria do uso do serviço TINA, assume-se implicitamente na especificação do ponto de referências TINA. Um agente *billing* não necessariamente implica, que o provedor de conectividade (CP – *Connectivity Provider*) seja feito invisível[13] para o usuário. Este pode ser visível, por exemplo, um *bill* separado do CP, ou um *bill* combinado, integrado num único Provedor, dependendo do escopo do contexto de enlace.

[13] **Contexto de billing invisível**: o Provedor aparece como um integrador de todos os sub-serviços necessários, o qual inclui serviços suportados pelo CP. O Consumidor não trata diretamente com o CP, tal que o contexto não pode ser passado para o CP desde o Consumidor. O *bill* do CP pode aparecer com o *billing* do Provedor.

A suposição acima implica em uma manutenção da qualidade do serviço em TINA. No contexto de *billing* visível[14], ambas entidades de negócios, por exemplo, o Provedor e o Provedor de serviço de comunicação são visíveis desde o usuário. Visto que a visibilidade no *billing* deve ser traduzida dentro das responsabilidades sobre a manutenção da qualidade de serviço, o resultado do monitoramento de desempenho deve ser correlacionado com informações *billing* separadas em caso do contexto de *billing* visível para a conclusão da transação de serviço (ABARCA, 1998).

Baseado nestes conceitos pode se definir o contexto de gerenciamento de contabilidade TINA (*AccMgmtCtxt*) através dos domínios Consumidor (*Consumer*) e Provedor (*Retailer*) com seus respectivos componentes, como mostra a Figura 7.9.

O modelo apresentado na Figura 7.9, desenvolvido no LRG como parte da linha de pesquisa, foi baseado e aperfeiçoado em função das informações contidas nos documentos do Consórcio TINA, sendo que novas funções de gerenciamento de contabilidade foram criadas e incluídas no modelo.

A especificação no mais alto nível de abstração da arquitetura de serviço TINA são apresentadas nas Figuras 7.10 e 7.11 e que correspondem à formalização dos requerimentos dos usuários. Esta especificação é usada como base para refinamentos posteriores da concepção, no decorrer do projeto (SOUZA, 2001).

Cada especificação demonstra o comportamento entre o domínio usuário e o domínio do provedor através do ponto de referência Ret, o qual é dividido em acesso e serviço. A sessão de acesso foi totalmente especificada enquanto na sessão de serviço somente o gerenciamento de contabilidade foi especificado.

Em uma especificação formal LOTOS, deve-se descrever o modelo formalmente em duas etapas. Primeiramente, faz-se a descrição formal do serviço (1s) de acordo com as especificações mostradas na Figura 7.9.

Na Segunda etapa, descreve-se a especificação formal do protocolo (1p), o qual define os processos *Access* (Acesso) e *Accounting* (Contabilidade). O processo *Access*, permite a autenticação e acesso ao serviço do usuário. O processo *Accounting*, permite o controle e métricas contábeis dos recursos fornecidos no serviço. A especificação formal é mostrada na Figura 7.10.

[14] **Contexto de billing visível**: o Provedor e o Provedor de conectividade são vistos como duas entidades independentes pelo Consumidor. Assim como, dois *billing* separados são gerados desde as duas entidades.

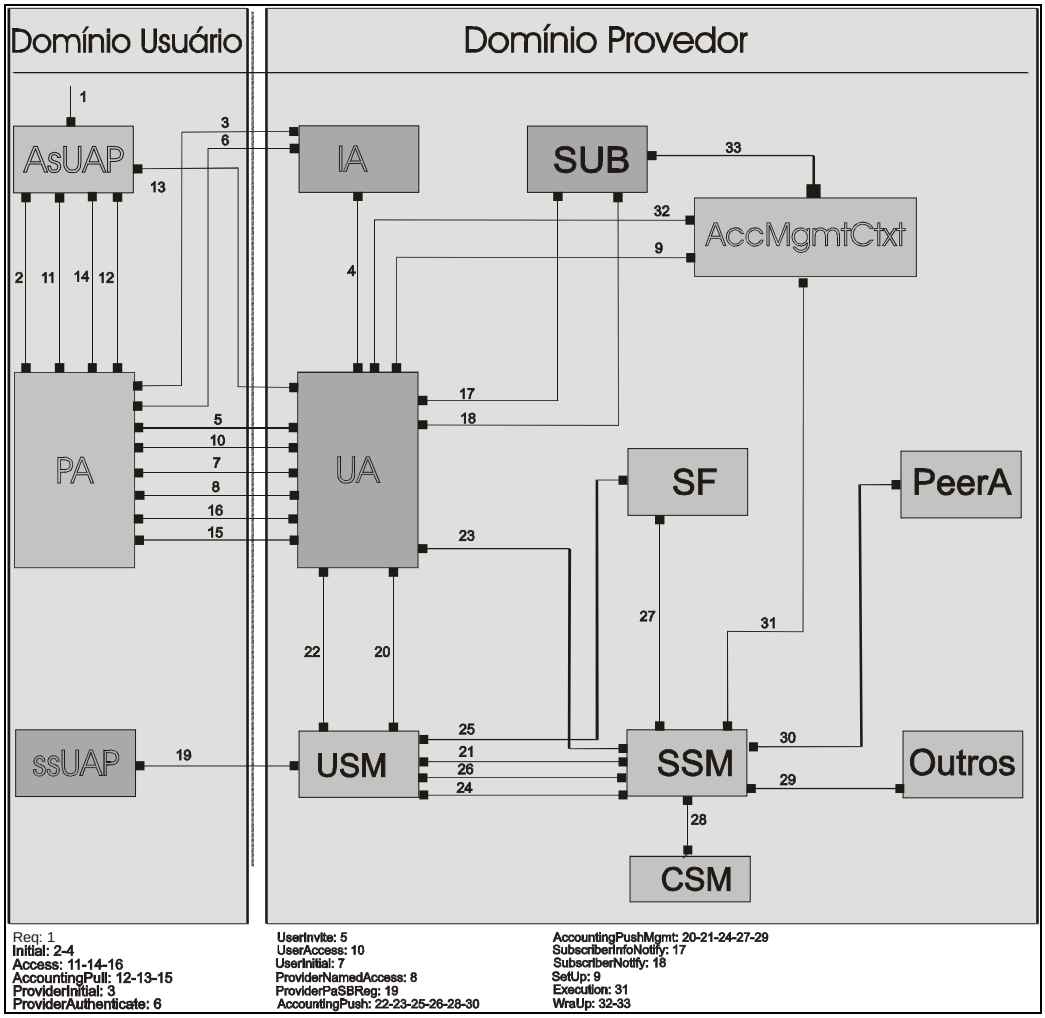


Figura 7.9 – Representação de Serviço do Modelo (SOUZA, 2001).

A verificação é uma prova formal da especificação que satisfaz as propriedades desejadas usando métodos matemáticos rigorosos. Ela pode ser de dois tipos, segundo (ARNOLD, 1989) e (GARAVEL, 2004):

- **Verificação com redução**, os gráficos gerados pelos compiladores são reduzidos de acordo com as relações de equivalência, que são, equivalência forte e fraca;
    - **Equivalência forte:** requer que cada evento em um sistema de transição corresponda a exatamente um evento igual no outro sistema de transição.
    - **Equivalência fraca:** um evento no sistema de transições, não precisa estar relacionado a um evento no outro sistema de transições.
- **Verificação por comparação**, uma especificação pode ser verificada através da comparação com outra especificação sob critérios, tais como equivalência forte e fraca.

```
specification
1s[req,initial1,providerinitial,initial2,userinvite,providerauthenticate,initial3,userinitial,providernamedaccess,setup,useraccess,access1,accountingpull1,accountingpull2,access2,accountingpull3,access3,subscriberinfonotify,subscribernotify,providerpasbreq,accountingpushmanagement1,accountingpushmanagement2,accountingpush1,accountingpush2,accountingpushmanagement3,accountingpush3,accountingpush4,accountingpushmanagement4,accountingpush5,accountingpushmanagement5,accountingpush6,execution,wrapup1,wrapup2]:noexit
        behaviour
        1s[. . .]
        where
        process  1s[. . .]:noexit:=
        req;(i;initial1;providerinitial;initial2;userinvite;providerauthenticate;initial3;userinitial;providernamedaccess;setup;useraccess;access1;accountingpull1;accountingpull2;access2;accountingpull3;access3;1s[. . .]
            []i;account[. . .]
            )
      where
        process account[. . .]:noexit:=
        req;(i;subscriberinfonotify;subscribernotify;providerpasbreq;accountingpushmanagement1;accountingpushmanagement2;accountingpush1;accountingpush2;accountingpushmanagement3;accountingpush3;accountingpush4;accountingpushmanagement4;accountingpush5;accountingpushmanagement5;accountingpush6;execution;wrapup1;wrapup2;account[. . .]
      []i; 1s[. . .]
          )
        endproc
        endproc
endspec
```

Figura 7.10 – Especificação de Serviço do Modelo.

Para validar a especificação formal foi usada a ferramenta ***EUCALYPTUS***, que integra **CADP** e **ALDEBARAN**. O método utilizado para comparar a especificação de serviço com a especificação de protocolo, foi a ***verificação por comparação***. Os resultados corretos sobre a análise sintática e semântica, e *"deadlocks"* da especificação foram obtidos. Os resultados da verificação por comparação são ilustrados na Figura 7.11.

O resultado "*TRUE*" significa que o protocolo executa o serviço esperado. A Figura 7.12, mostra este resultado como prova formal de equivalência quanto à observação entre as especificações 1s e 1p. O resultado "*TRUE*" obtido representa que a

especificação inicial dos requisitos do sistema é equivalente à especificação final refinada.

```
specification 1p [req,initial1, providerinitial, initial2, userinvite, providerauthenticate, initial3, userinitial,
providernamedaccess, setup, useraccess, access1, accountingpull1, accountingpull2, access2, accountingpull3,
access3, subscriberinfonotify, subscribernotify, providerpasbreq, accountingpushmanagement1,
accountingpushmanagement2, accountingpush1, accountingpush2, accountingpushmanagement3, accountingpush3,
accountingpush4, accountingpushmanagement4, accountingpush5, accountingpushmanagement5, accountingpush6,
execution, wrapup1, wrapup2] : noexit
behaviour
1p[. . .]
        where
        process 1p[. . .]:noexit:=
           acesso [. . .]
            >>
           account [. . .]
        endproc
        process acesso [. . .]:exit:=
                req;(i;initial1; providerinitial; initial2; userinvite; providerauthenticate; initial3; userinitial;
providernamedaccess; setup; useraccess; access1;accountingpull1; accountingpull2; access2; accountingpull3;
access3;acesso [. . .]
             []i;exit
                 )
        endproc
        process account[. . .]:noexit:=
        req;(i;subscriberinfonotify;subscribernotify;providerpasbreq;accountingpushmanagement1;accountingpush
management2;accountingpush1;accountingpush2;accountingpushmanagement3;accountingpush3;accountingpush4;ac
countingpushmanagement4;accountingpush5;accountingpushmanagement5;accountingpush6;execution;wrapup1;wra
pup2;account[. . .]
        []i;1p[. . .]
        )
        endproc
endspec
```

Figura 7.11 – Especificação de Protocolo do Modelo.

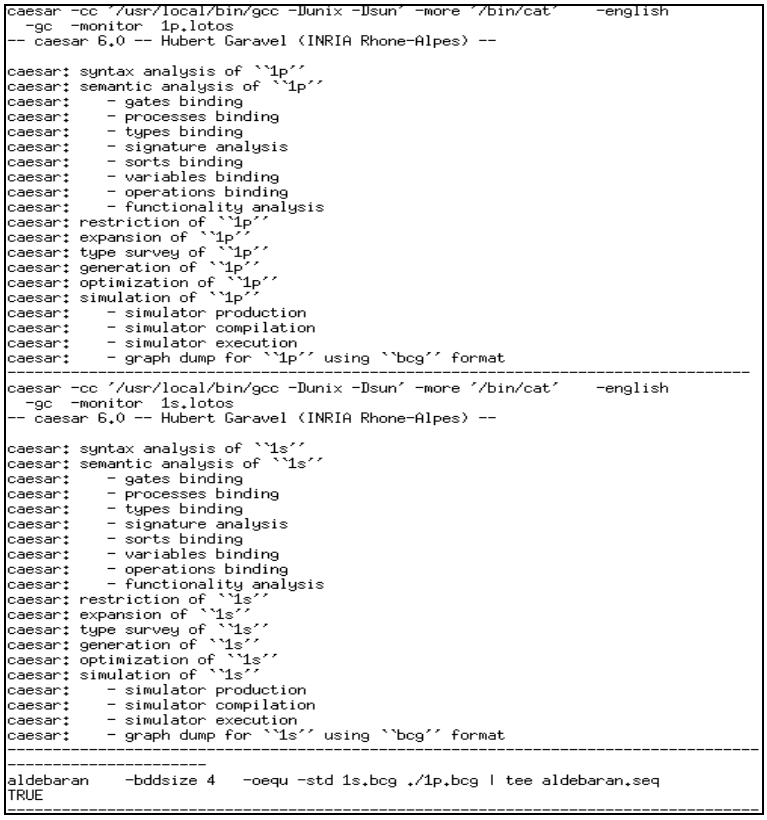


Figura 7.12 – Resultado da Verificação por Comparação de Equivalência Fraca entre a Especificação de Serviço e Protocolo.

A verificação proporciona a obtenção da prova formal (matemática) de correção do modelo de Gerenciamento de Contabilidade. Essa prova visa satisfazer os requisitos do nível máximo de segurança de acordo com o Departamento de Defesa dos EUA. Dessa maneira, o procedimento de análise formal e de obtenção da prova matemática de correção, visam garantir o modelo verificado, em conformidade com o mais alto nível de segurança (Class*A1*) (SIMON, 1996).

A partir desta primeira validação nasceu uma segunda especificação mas detalhada do contexto de contabilidade, onde através de uma análise mais minuciosa do modelo de contabilidade TINA conseguiu-se definir uma quantidade maior de componentes e interfaces que interatuam com aqueles já especificados no primeiro modelo e que dão uma visão mais completa e às vezes mais complexa da arquitetura de contabilidade definida pelo padrão TINA. A seguir na Figura 7.13 descreve-se a especificação do modelo de contabilidade *AccMgmtCtxt*, o qual forma parte do sistema geral do SSGCT.

```
specification AccMgmtCtxt [i_UserInitial, i_UserAccess, i_Access, i_AccountingPull1, i_AccountingPull2,
                           i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1, i_AccountingPush2,
                           i_AccountingPush3, i_AccountingPush4, i_AccountingPush5, i_AccountingPush6,
                           i_AccountingPush7, i_AccountingPushMgmt1, i_AccountingPushMgmt2,
                           i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1,
                           i_SSManage2, i_Resume1, i_Resume2, i_Resume3, i_Resume4, i_Resume5,
                           i_Resume6, i_PartyUSM2, i_InvitationDelivery1, i_InvitationDelivery2,
                           i_InvitationDelivery3, i_Join1, i_Join2, i_SSCreate, i_SSEvents, i_InitialAccess1,
                           i_InitialAccess2, i_InitialAccess3, i_InitialAccess4, i_SubscriberInfoQuery1,
                           i_SubscriberInfoQuery2, i_Subscribe1, i_Subscribe2, i_ServiceContractInfoMgmt1,
                           i_SubscriberInfoMgmt1, i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2,
                           i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2] : noexit

behaviour

        contabilidadeservico[i_UserInitial, i_UserAccess, i_Access, i_AccountingPull1, i_AccountingPull2,
                             i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1, i_AccountingPush2,
                             i_AccountingPush3, i_AccountingPush4, i_AccountingPush5, i_AccountingPush6,
                             i_AccountingPush7, i_AccountingPushMgmt1, i_AccountingPushMgmt2,
                             i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1,
                             i_SSManage2, i_Resume1, i_Resume2, i_Resume3, i_Resume4, i_Resume5, i_Resume6,
                             i_PartyUSM2, i_InvitationDelivery1, i_InvitationDelivery2, i_InvitationDelivery3,
                             i_Join1, i_Join2, i_SSCreate, i_SSEvents, i_InitialAccess1, i_InitialAccess2,
                             i_InitialAccess3, i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2,
                             i_Subscribe1, i_Subscribe2, i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1,
                             i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2, i_SubscriptionNotify,
                             i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2]

where

  process contabilidadeservico[i_UserInitial, i_UserAccess, i_Access, i_AccountingPull1, i_AccountingPull2,
                               i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1, i_AccountingPush2,
                               i_AccountingPush3, i_AccountingPush4, i_AccountingPush5, i_AccountingPush6,
                               i_AccountingPush7, i_AccountingPushMgmt1, i_AccountingPushMgmt2,
                               i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1,
                               i_SSManage2, i_Resume1, i_Resume2, i_Resume3, i_Resume4, i_Resume5,
                               i_Resume6, i_PartyUSM2, i_InvitationDelivery1, i_InvitationDelivery2,
                               i_InvitationDelivery3, i_Join1, i_Join2, i_SSCreate, i_SSEvents, i_InitialAccess1,
                               i_InitialAccess2, i_InitialAccess3, i_InitialAccess4, i_SubscriberInfoQuery1,
                               i_SubscriberInfoQuery2, i_Subscribe1, i_Subscribe2, i_ServiceContractInfoMgmt1,
                               i_SubscriberInfoMgmt1, i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2,
                               i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2] :
                               noexit :=
                      i; i_UserInitial; i_UserAccess; i_Access; i_AccountingPull1; i_AccountingPull2;
                      i_MgmtCtxt; i_AccountingPull3; i_Execution; i_AccountingPush1; i_AccountingPush2;
                      i_AccountingPush3; i_AccountingPush4; i_AccountingPush5; i_AccountingPush6;
                      i_AccountingPush7; i_AccountingPushMgmt1; i_AccountingPushMgmt2;
                      i_AccountingPushMgmt3; i_SessionCtrl; i_PartyUSM_1; i_SessionInfo; i_SSManage1;
                      i_SSManage2; i_Resume1; i_Resume2; i_Resume3; i_Resume4; i_Resume5; i_Resume6;
                      i_PartyUSM2; i_InvitationDelivery1; i_InvitationDelivery2; i_InvitationDelivery3;
                      i_Join1; i_Join2; i_SSCreate; i_SSEvents; i_InitialAccess1; i_InitialAccess2;
                      i_InitialAccess3; i_InitialAccess4; i_SubscriberInfoQuery1; i_SubscriberInfoQuery2;
                      i_Subscribe1; i_Subscribe2; i_ServiceContractInfoMgmt1; i_SubscriberInfoMgmt1;
                      i_ServiceContractInfoMgmt2; i_SubscriberInfoMgmt2;
                      i_SubscriptionNotify; i_ServiceQuery; i_ServiceNotify; i_WrapUp1; i_WrapUp2;
                      contabilidadeservico[i_UserInitial, i_UserAccess, i_Access, i_AccountingPull1,
                      i_AccountingPull2, i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1,
                      i_AccountingPush2, i_AccountingPush3, i_AccountingPush4, i_AccountingPush5,
```

```
                i_AccountingPush6, i_AccountingPush7, i_AccountingPushMgmt1,
                i_AccountingPushMgmt2, i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1,
                i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1, i_Resume2, i_Resume3,
                i_Resume4, i_Resume5, i_Resume6, i_PartyUSM2, i_InvitationDelivery1,
                i_InvitationDelivery2, i_InvitationDelivery3, i_Join1, i_Join2, i_SSCreate, i_SSEvents,
                i_InitialAccess1, i_InitialAccess2, i_InitialAccess3, i_InitialAccess4,
                i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1, i_Subscribe2,
                i_ServiceContractInfoMgmt1,i_SubscriberInfoMgmt1, i_ServiceContractInfoMgmt2,
                i_SubscriberInfoMgmt2,   i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify,
                i_WrapUp1, i_WrapUp2]
  endproc

endspec
```

Figura 7.13 – Descrição Formal do Modelo *AccMgmtCtxt* no SSGCT.

7.4 Especificação do Modelo SSGCT

A arquitetura geral do sistema SSGCT incluindo os principais processos refinados, podem ser observados na Figura 7.14 (Obs.: uma melhor visualização da Figura é apresentado no Anexo 2).

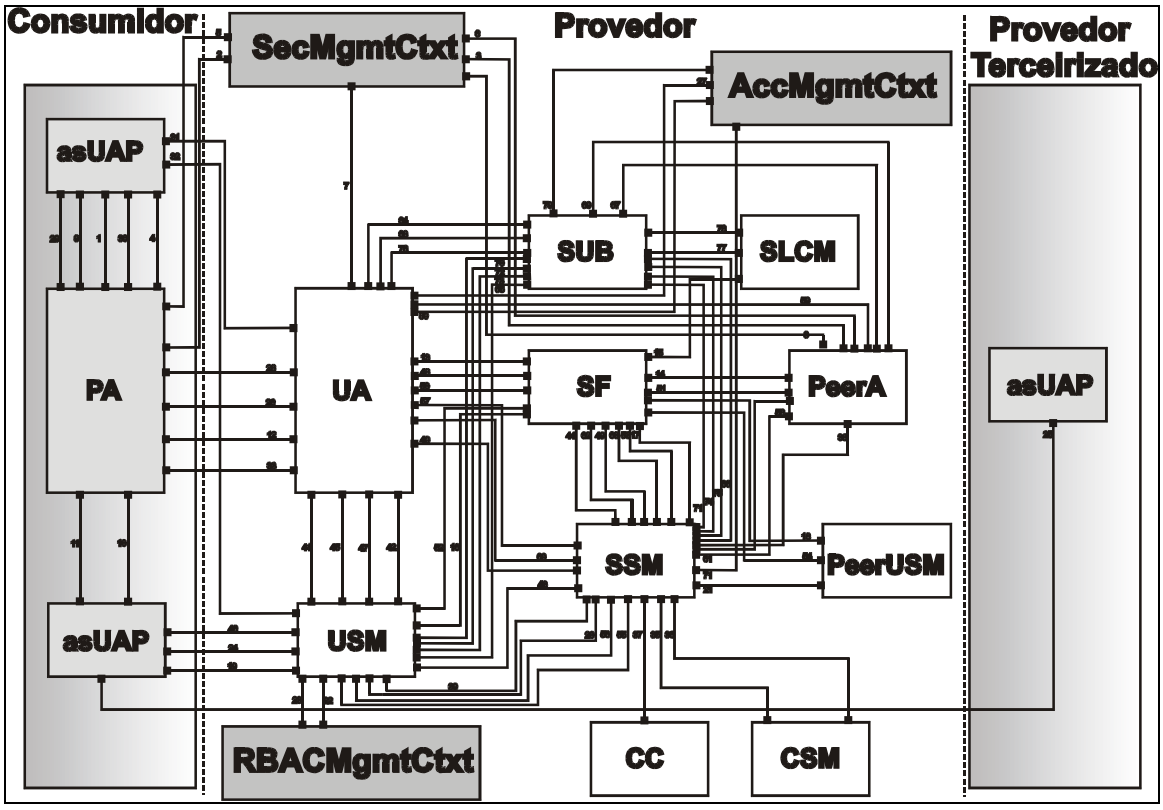


Figura 7.14 – Arquitetura Geral do SSGCT.

i_Initial: 1-7-8	**i_UserInitial: 26**	**i_InvitationDelivery: 57-58-59**
i_ProviderInitial: 2-3	**i_SetUp: 27**	**i_Join: 60-61**
i_Provider Authenticate: 4-5-6	**i_UserAccess: 28**	**i_SSEvents: 63**
i_Access: 9-11-29	**i_AccountingPull: 30-31-33**	**i_InitialAccess: 64-65-66-67**
i_AccessInitialise: 10	**i_Execution: 34**	**i_SubscriberInfoQuery: 68-69**
i_ProviderAccess: 12	**i_AccountingPush: 35-36-37-38-39-40-41**	**i_Subscribe: 70-71**

i_SSCreate: 13-14-62	**i_AccountingPushgmt: 42-43-44**	**i_ServiceContractInfoMgmt: 72-74**
i_Init: 15-16-17-18	**i_sessionCtrl: 45**	**i_SubscriberInfoMgmt: 73-75**
i_ProviderssUAP: 19	**i_PartyUSM: 46-47**	**i_SubscriptionNotify: 76**
i_ProviderSSM: 21-22	**i_SessionInfo: 47**	**i_ServiceQuery: 77**
i_RoleCertificateReq: 22	**i_SSManage: 48-49**	**i_ServiceNotify: 78**
i_RoleCertificate: 23-24-25	**i_Resume: 50-51-52-53-54-55**	**i_Wrap: 79-80.**

A arquitetura representada na Figura 7.14 pode ser descrita em LOTOS através da união das especificações refinadas descritas nos itens 7.3.2, 7.3.3.1 e 7.3.3.2 como apresentado na especificação formal da Figura 7.15.

```
specification SsgctProtocol [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                  i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_Access1, i_ProviderAccess,
                  i_SSCreate1, i_Init1, i_Init2, i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq,
                  i_RoleCertificate1, i_RoleCertificate2,i_RoleCertificate3, i_UserInitial, i_UserAccess,
                  i_Access2, i_AccountingPull1, i_AccountingPull2, i_MgmtCtxt, i_AccountingPull3,
                  i_Execution, i_AccountingPush1, i_AccountingPush2, i_AccountingPush3,
                  i_AccountingPush4, i_AccountingPush5, i_AccountingPush6, i_AccountingPush7,
                  i_AccountingPushMgmt1, i_AccountingPushMgmt2, i_AccountingPushMgmt3,
                  i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1,
                  i_Resume2, i_Resume3, i_Resume4, i_Resume5, i_Resume6, i_PartyUSM2,
                  i_InvitationDelivery1, i_InvitationDelivery2, i_InvitationDelivery3, i_Join1, i_Join2,
                  i_SSCreate2, i_SSEvents, i_InitialAccess1, i_InitialAccess2, i_InitialAccess3,
                  i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1,
                  i_Subscribe2, i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1,
                  i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2, i_SubscriptionNotify,
                  i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2] : noexit
behaviour
          hide i_RoleCertificateReq, i_RoleCertificate1, i_RoleCertificate2, i_RoleCertificate3 in
               modeloSsgct [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                  i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_Access1, i_ProviderAccess,
                  i_SSCreate1, i_Init1, i_Init2, i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq,
                  i_RoleCertificate1, i_RoleCertificate2,i_RoleCertificate3, i_UserInitial, i_UserAccess,
                  i_Access2, i_AccountingPull1, i_AccountingPull2, i_MgmtCtxt, i_AccountingPull3,
                  i_Execution, i_AccountingPush1, i_AccountingPush2, i_AccountingPush3,
                  i_AccountingPush4, i_AccountingPush5, i_AccountingPush6, i_AccountingPush7,
                  i_AccountingPushMgmt1, i_AccountingPushMgmt2, i_AccountingPushMgmt3,
                  i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1,
                  i_Resume2, i_Resume3, i_Resume4, i_Resume5, i_Resume6, i_PartyUSM2,
                  i_InvitationDelivery1, i_InvitationDelivery2, i_InvitationDelivery3, i_Join1, i_Join2,
                  i_SSCreate2, i_SSEvents, i_InitialAccess1, i_InitialAccess2, i_InitialAccess3,
                  i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1,
                  i_Subscribe2, i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1,
                  i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2, i_SubscriptionNotify,
                  i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2]

where

     process modeloSsgct [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                  i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_Access1, i_ProviderAccess,
                  i_SSCreate1, i_Init1, i_Init2, i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq,
                  i_RoleCertificate1, i_RoleCertificate2,i_RoleCertificate3,i_UserInitial, i_UserAccess,
                  i_Access2, i_AccountingPull1, i_AccountingPull2, i_MgmtCtxt, i_AccountingPull3,
                  i_Execution, i_AccountingPush1, i_AccountingPush2, i_AccountingPush3,
                  i_AccountingPush4, i_AccountingPush5, i_AccountingPush6, i_AccountingPush7,
                  i_AccountingPushMgmt1, i_AccountingPushMgmt2, i_AccountingPushMgmt3,
                  i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1,
```

```
                i_Resume2, i_Resume3, i_Resume4, i_Resume5, i_Resume6, i_PartyUSM2,
                i_InvitationDelivery1, i_InvitationDelivery2, i_InvitationDelivery3,i_Join1, i_Join2,
                i_SSCreate2, i_SSEvents, i_InitialAccess1, i_InitialAccess2, i_InitialAccess3,
                i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1,
                i_Subscribe2, i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1,
                i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2,
                i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2] : noexit
:=

          SecMgmtCtxt [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
                i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_ProviderAccess]
          >>
          RBACMgmtCtxt [i_Access1, i_ProviderAccess, i_SSCreate1, i_Init1, i_Init2,
                i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1,
                i_RoleCertificate2, i_RoleCertificate3]
          >>
          AccMgmtCtxt [i_UserInitial, i_UserAccess, i_Access2, i_AccountingPull1, i_AccountingPull2,
                i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1,
                i_AccountingPush2, i_AccountingPush3, i_AccountingPush4, i_AccountingPush5,
                i_AccountingPush6, i_AccountingPush7, i_AccountingPushMgmt1,
                i_AccountingPushMgmt2, i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1,
                i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1, i_Resume2, i_Resume3,
                i_Resume4, i_Resume5, i_Resume6, i_PartyUSM2, i_InvitationDelivery1,
                i_InvitationDelivery2, i_InvitationDelivery3, i_Join1, i_Join2, i_SSCreate2,
                i_SSEvents, i_InitialAccess1, i_InitialAccess2, i_InitialAccess3,
                i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1,
                i_Subscribe2, i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1,
                i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2,
                i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2]
     where

          process SecMgmtCtxt [i_Initial1, i_ProviderInitial1, i_ProviderInitial2, i_ProviderAuthenticate1,
               i_ProviderAuthenticate2, i_ProviderAuthenticate3, i_Initial2, i_ProviderAccess] : exit :=

               i_Initial1; (i_ProviderInitial1; i_ProviderAuthenticate1; i_ProviderAuthenticate2; i_Initial2;
                         i_ProviderAccess; exit
                    []
                    i_ProviderInitial2; i_ProviderAuthenticate3; i_Initial2; i_ProviderAccess; exit)


          endproc

          process RBACMgmtCtxt [i_Access1, i_ProviderAccess, i_SSCreate1, i_Init1, i_Init2,
               i_ProviderSSM, i_ProviderssUAP, i_RoleCertificateReq, i_RoleCertificate1,
               i_RoleCertificate2, i_RoleCertificate3] : exit :=

                    i; i_Access1; i_ProviderAccess; i_SSCreate1; i_Init1; i_Init2; i_ProviderSSM;
               i_ProviderssUAP; i_RoleCertificateReq; i_RoleCertificate1; i_RoleCertificate2;
                         i_RoleCertificate3; exit

          endproc


     process AccMgmtCtxt [i_UserInitial, i_UserAccess, i_Access2, i_AccountingPull1,
          i_AccountingPull2, i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1,
          i_AccountingPush2, i_AccountingPush3, i_AccountingPush4, i_AccountingPush5,
          i_AccountingPush6, i_AccountingPush7, i_AccountingPushMgmt1,
          i_AccountingPushMgmt2, i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1,
```

```
            i_SessionInfo, i_SSManage1, i_SSManage2, i_Resume1, i_Resume2, i_Resume3, i_Resume4,
            i_Resume5, i_Resume6, i_PartyUSM2, i_InvitationDelivery1, i_InvitationDelivery2,
            i_InvitationDelivery3, i_Join1, i_Join2, i_SSCreate2, i_SSEvents, i_InitialAccess1,
            i_InitialAccess2, i_InitialAccess3, i_InitialAccess4, i_SubscriberInfoQuery1,
            i_SubscriberInfoQuery2, i_Subscribe1, i_Subscribe2, i_ServiceContractInfoMgmt1,
            i_SubscriberInfoMgmt1, i_ServiceContractInfoMgmt2, i_SubscriberInfoMgmt2,
            i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1, i_WrapUp2] : noexit :=


            i; i_UserInitial; i_UserAccess; i_Access2; i_AccountingPull1; i_AccountingPull2; i_MgmtCtxt;
            i_AccountingPull3; i_Execution; i_AccountingPush1; i_AccountingPush2; i_AccountingPush3;
            i_AccountingPush4; i_AccountingPush5; i_AccountingPush6; i_AccountingPush7;
            i_AccountingPushMgmt1; i_AccountingPushMgmt2; i_AccountingPushMgmt3; i_SessionCtrl;
            i_PartyUSM_1; i_SessionInfo; i_SSManage1; i_SSManage2; i_Resume1; i_Resume2;
            i_Resume3; i_Resume4; i_Resume5; i_Resume6; i_PartyUSM2; i_InvitationDelivery1;
            i_InvitationDelivery2; i_InvitationDelivery3; i_Join1; i_Join2; i_SSCreate2; i_SSEvents;
            i_InitialAccess1; i_InitialAccess2; i_InitialAccess3; i_InitialAccess4; i_SubscriberInfoQuery1;
            i_SubscriberInfoQuery2; i_Subscribe1; i_Subscribe2; i_ServiceContractInfoMgmt1;
            i_SubscriberInfoMgmt1; i_ServiceContractInfoMgmt2; i_SubscriberInfoMgmt2;
            i_SubscriptionNotify; i_ServiceQuery; i_ServiceNotify; i_WrapUp1; i_WrapUp2;
          AccMgmtCtxt[i_UserInitial, i_UserAccess, i_Access2, i_AccountingPull1, i_AccountingPull2,
          i_MgmtCtxt, i_AccountingPull3, i_Execution, i_AccountingPush1, i_AccountingPush2,
          i_AccountingPush3, i_AccountingPush4, i_AccountingPush5, i_AccountingPush6,
          i_AccountingPush7, i_AccountingPushMgmt1, i_AccountingPushMgmt2,
          i_AccountingPushMgmt3, i_SessionCtrl, i_PartyUSM_1, i_SessionInfo, i_SSManage1,
          i_SSManage2, i_Resume1, i_Resume2, i_Resume3, i_Resume4, i_Resume5, i_Resume6,
          i_PartyUSM2, i_InvitationDelivery1, i_InvitationDelivery2, i_InvitationDelivery3, i_Join1,
          i_Join2, i_SSCreate2, i_SSEvents, i_InitialAccess1, i_InitialAccess2, i_InitialAccess3,
          i_InitialAccess4, i_SubscriberInfoQuery1, i_SubscriberInfoQuery2, i_Subscribe1, i_Subscribe2,
          i_ServiceContractInfoMgmt1, i_SubscriberInfoMgmt1, i_ServiceContractInfoMgmt2,
          i_SubscriberInfoMgmt2, i_SubscriptionNotify, i_ServiceQuery, i_ServiceNotify, i_WrapUp1,
          i_WrapUp2]

      endproc

endproc

endspec
```

Figura 7.15 – Especificação LOTOS do SSGCT.

Após os refinamentos para os diferentes contextos (*SecMgmtCtxt*, *RBACMgmtCtxt* e *AccMgmtCtxt*) que formam os processos do sistema SCGCT através da especificação *SsgctProtocol*, definem-se os operadores **hide** *i_RoleCertificateReq*, *i_RoleCertificate1*, *i_RoleCertificate2*, *i_RoleCertificate3* ..**in** *modeloSsgct*[...] com o objetivo de ocultar as portas aos usuários de um serviço contra o uso desonesto ou violações dos certificados definidos pelo provedor (Retailer) e o operador de habilitação de processos >> que vai a permitir executar um processo com sucesso através do processo predefinido **exit**. Neste caso, o processo *SecMgmtCtxt* deve terminar com sucesso para que o processo *RBACMgmtCtxt* possa ser executado, e este por sua vez deve terminar com sucesso para executar o processo *AccMgmtCtxt*.

Na seqüência, o sistema pode ser validado através do emprego de ferramentas apropriadas como **EUCALYTUS toolset** (GARAVEL, 2004).

7.5 Validação Formal do Modelo

A grande vantagem do uso de TDFs é a possibilidade de demonstrar a correção de uma especificação. A validação pode ser usada para descrever as atividades de demonstração de correção, ou apenas aumento de confiabilidade de uma especificação ou implementação (NOTARE, 2000). Algumas das técnicas de validação são descritas a seguir.

7.5.1 Teste

Testes podem ser aplicados em especificações realizadas em alto nível de abstração ou até mesmo versões de implementação. A maior desvantagem dessa técnica de validação é que os testes não são exaustivos, ou seja, são usados para encontrar erros e não para demonstrar correções (não constituindo, desse modo uma prova formal de correção).

7.5.2 Simulação

Na simulação, um modelo executável é desenvolvido e observado. Se o modelo não for muito complexo, a simulação descobre a maioria dos erros. Quando, porém, o modelo torna-se complexo, geralmente é impossível simular todos os casos importantes.

7.5.3 Verificação

A verificação é uma prova formal de que uma especificação ou implementação satisfaz propriedades desejáveis, usando métodos matemáticos rigorosos. A análise de especificações permite várias verificações. Por exemplo, comparar uma especificação original com outra mais refinada. Dois tipos de verificações podem ser especificadas: verificação por redução e verificação por comparação, como descrito no item 7.3.3.2.

7.6 Validação do Modelo SSGCT através da Ferramenta EUCALYPTUS

EUCALIPTUS é uma interface gráfica que inclui além do CADP (*Caesar/Aldebaran Development Package*) várias outras ferramentas, como mostra a Figura 7.16.

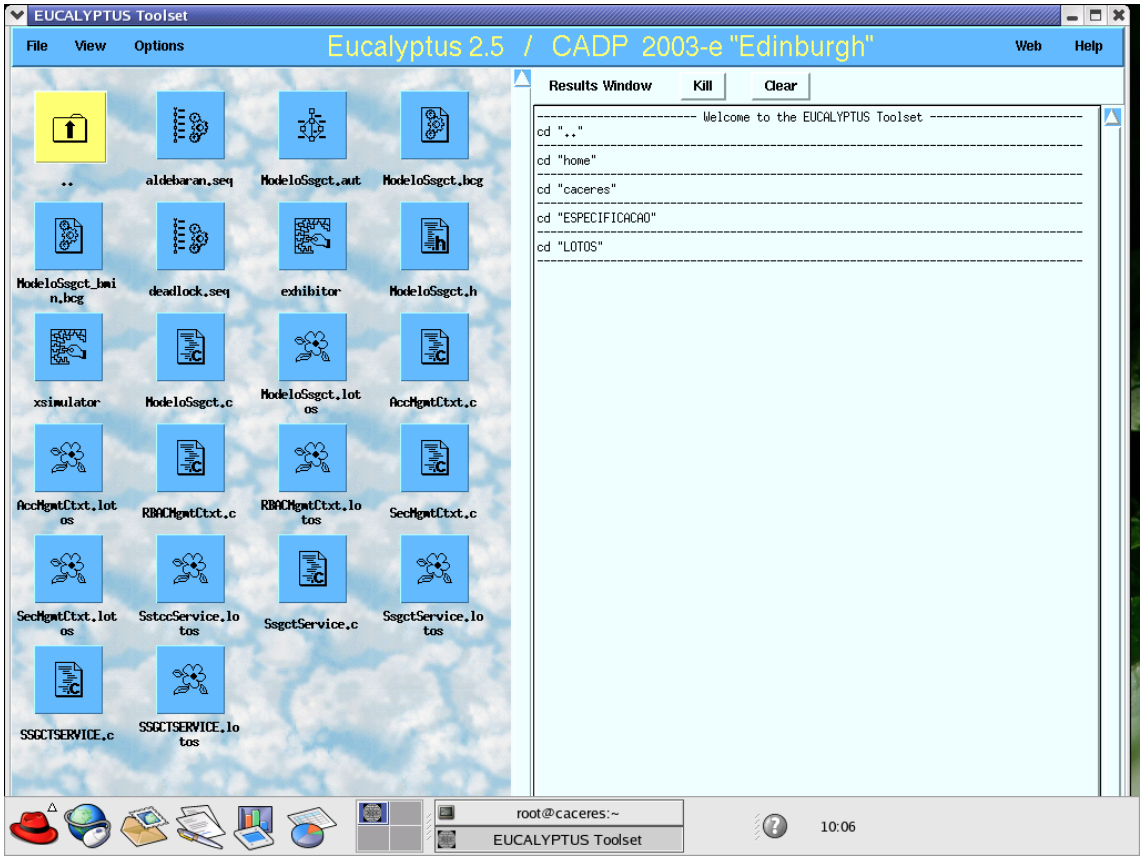


Figura 7.16 – A Interface *Eucalyptus ToolSet*.

As duas principais ferramentas utilizadas neste trabalho são:

1. CAESAR

CAESAR é um compilador que traduz uma especificação LOTOS em um programa C (para ser executado ou simulado) ou em um LTS - *Labelled Transition Systems* (para ser verificado utilizando ferramentas de bissimulação e/ou de lógica temporal). Por exemplo, é possível comparar o LTS de um protocolo com o LTS do serviço implementado pelo protocolo. Ambos LTSs são gerados utilizando CAESAR e comparados utilizando ALDEBERAN. Também é possível especificar propriedades do protocolo utilizando fórmulas de lógica temporal que podem ser avaliadas no LTS do protocolo. Os algoritmos de tradução CAESAR seguem vários passos. Primeiro, a descrição LOTOS é traduzida em uma álgebra de processos simplificada, chamada SUBLOTOS. Em seguida, num modelo intermediário de Redes de Petri, o qual provê uma representação compacta, estruturada e compreensível pelo usuário de ambos, controle e fluxo de dados. Eventualmente o LTS é produzido através de análise de

alcançabilidade sobre a Rede Petri. CAESAR aceita LOTOS completo (i. e., ambos, comportamento e dados) com a seguinte restrição em relação a parte de controle: recursão de processo não é permitida à esquerda e à direita de | [ ] |, nem na parte esquerda de >> e [>. Apesar destas restrições, o subconjunto de LOTOS suportado pelo CAESAR é grande e geralmente suficiente para as necessidades reais. A versão atual do CAESAR permite a geração de grandes LTSs em um razoável intervalo de tempo. A versão mais atual do CAESAR oferece uma funcionalidade chamada EXEC/CAESAR para a geração de código C. Este código pode ser inserido em aplicações, o que permite prototipação rápida das especificações LOTOS.

2. ALDEBARAN

ALDEBARAN é uma ferramenta para a verificação de sistemas de comunicação, representados por Sistemas de Transições Rotuladas (LTS - *Labelled Transition Systems*), i. e., as transições de estados são rotuladas por nomes de ações. Isto permite a redução de LTSs sob várias relações de equivalências (tais como bissimulação forte, equivalência quanto a observação, bissimulação de retardo e equivalência segura). ALDEBARAN provê ao usuário um diagnóstico quando dois LTS são considerados não equivalentes. Os algoritmos de verificação utilizados pelo ALDEBARAN são baseados em estudos, principalmente de Paige-Tarjan e Fernandez-Mounier (GARAVEL, 2004).

A seguir descreve-se os passos para a verificação de correção do sistema, isto é, a obtenção da prova formal de correção do sistema.

Passo 1: Geração do LTS

Para gerar o LTS (*Labelled Transition System*) correspondente à especificação do protocolo do arquivo *ModeloSsgct.lotos*, clique no arquivo e deve-se escolher a opção "***Generate labelled transition system...***", como mostra a Figura 7.17.

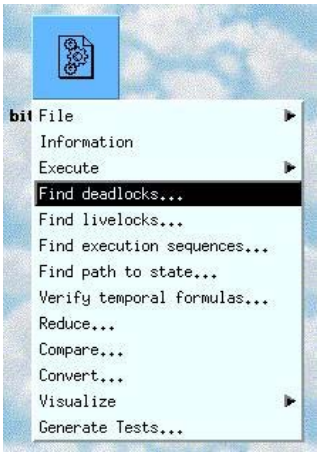


Figura 7.17 – Geração do LTS do Arquivo *ModeloSsgct.lotos*.

Ao clicar na opção "***Generate labelled transition system...***", uma segunda janela é aberta para escolher o tipo da geração do LTS, como mostra a Figura 7.18.

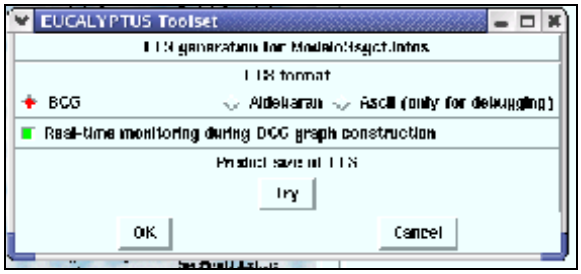


Figura 7.18 –Geração do LTS.

Durante a geração, o número de estados e transições são apresentados e uma vez completada, deve-se clicar no botão "***done***", como mostra a Figura 7.19.

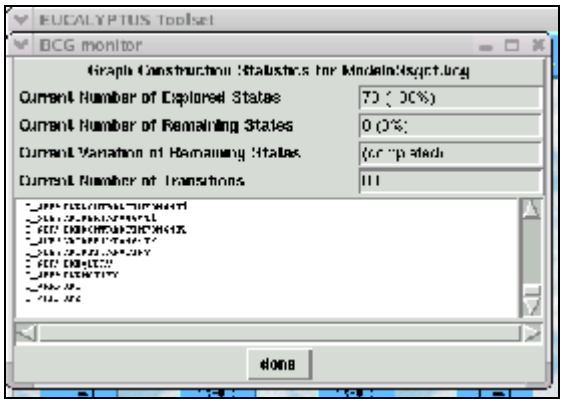


Figura 7.19 – Escolha na Geração do LTS.

A Figura 7.19 mostra que o LTS gerado, correspondente à especificação LOTOS *ModeloSsgct.lotos*, possui 79 estados e 80 transições. O grafo gerado é de formato

BCG[15]. Se a opção escolhida for ALDEBARAN ao invés de BGC, como neste caso, então a seguinte tela é apresentada como mostra a Figura 7.20.

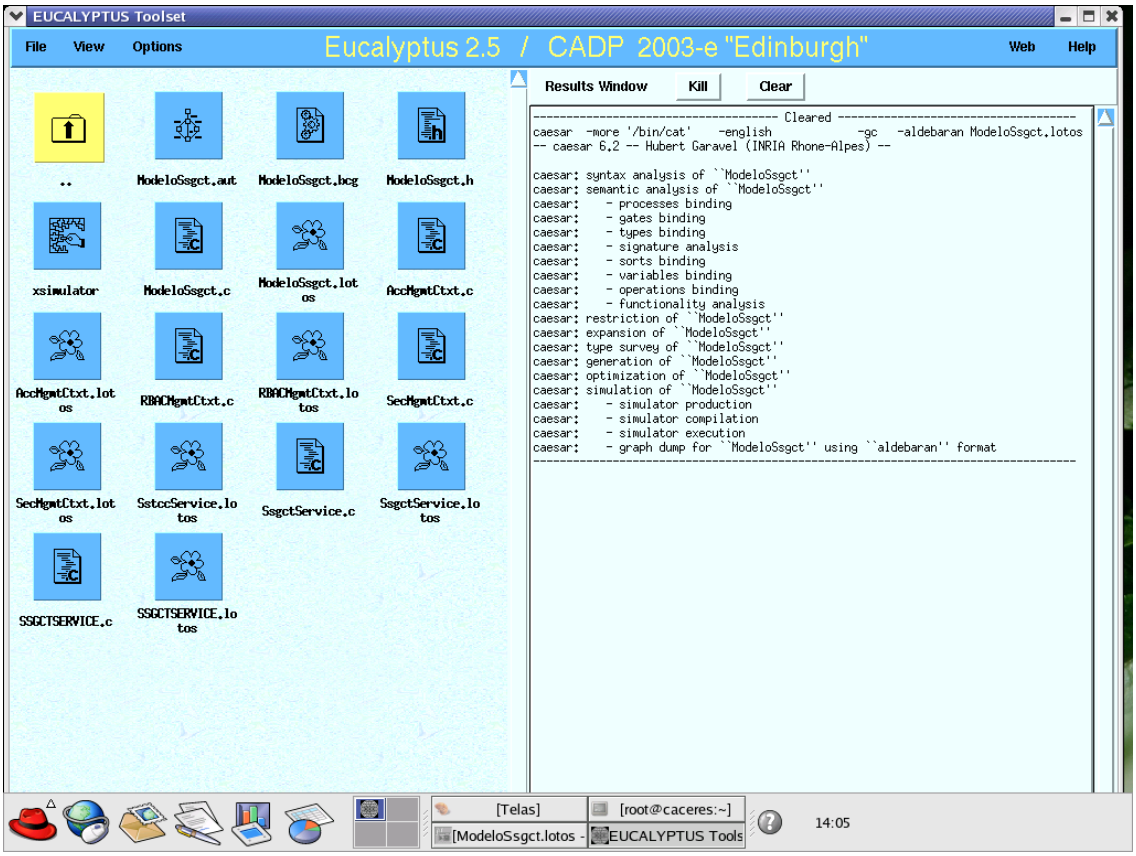


Figura 7.20 - Geração do LTS em Formato ALDEBARAN.

A seguir o seguinte grafo é gerado no formato ALDEBARAN (i.e., do-estado, evento, para-o-estado) como mostra a Figura 7.21.

```
des (0, 80, 79)
(0, I_INITIAL1, 1)
(1, I_PROVIDERINITIAL2, 2)
(1, I_PROVIDERINITIAL1, 3)
(2, I_PROVIDERAUTHENTICATE3, 4)
(3, I_PROVIDERAUTHENTICATE1, 5)
(4, I_INITIAL2, 6)
(5, I_PROVIDERAUTHENTICATE2, 7)
(6, I_PROVIDERACCESS, 8)
(7, I_INITIAL2, 9)
(8, i, 10)
(9, I_PROVIDERACCESS, 11)
(10, i, 12)
```

[15] BCG (*Binary-Coded Graphs*): é um formato de representação de LTSs que incluí uma coleção de bibliotecas e programas associados a este formato. Comparados aos formatos para LTSs baseados em ASCII, o formato BCG utiliza representação binária com técnicas de compressão que resultam em arquivos muito pequenos. O BGC é independente de qualquer linguagem, porém, mantém os objetos (tipos, funções, variáveis) definidos em programas fontes.

(11, i, 10)
(12, I_ACCESS1, 13)
(13, I_PROVIDERACCESS, 14)
(14, I_SSCREATE1, 15)
(15, I_INIT1, 16)
(16, I_INIT2, 17)
(17, I_PROVIDERSSM, 18)
(18, I_PROVIDERSSUAP, 19)
(19, i, 20)
(20, i, 21)
(21, i, 22)
(22, i, 23)
(23, i, 24)
(24, i, 25)
(25, I_USERINITIAL, 26)
(26, I_USERACCESS, 27)
(27, I_ACCESS2, 28)
(28, I_ACCOUNTINGPULL1, 29)
(29, I_ACCOUNTINGPULL2, 30)
(30, I_MGMTCTXT, 31)
(31, I_ACCOUNTINGPULL3, 32)
(32, I_EXECUTION, 33)
(33, I_ACCOUNTINGPUSH1, 34)
(34, I_ACCOUNTINGPUSH2, 35)
(35, I_ACCOUNTINGPUSH3, 36)
(36, I_ACCOUNTINGPUSH4, 37)
(37, I_ACCOUNTINGPUSH5, 38)
(38, I_ACCOUNTINGPUSH6, 39)
(39, I_ACCOUNTINGPUSH7, 40)
(40, I_ACCOUNTINGPUSHMGMT1, 41)
(41, I_ACCOUNTINGPUSHMGMT2, 42)
(42, I_ACCOUNTINGPUSHMGMT3, 43)
(43, I_SESSIONCTRL, 44)
(44, I_PARTYUSM_1, 45)
(45, I_SESSIONINFO, 46)
(46, I_SSMANAGE1, 47)
(47, I_SSMANAGE2, 48)
(48, I_RESUME1, 49)
(49, I_RESUME2, 50)
(50, I_RESUME3, 51)
(51, I_RESUME4, 52)
(52, I_RESUME5, 53)
(53, I_RESUME6, 54)
(54, I_PARTYUSM2, 55)
(55, I_INVITATIONDELIVERY1, 56)
(56, I_INVITATIONDELIVERY2, 57)
(57, I_INVITATIONDELIVERY3, 58)
(58, I_JOIN1, 59)
(59, I_JOIN2, 60)
(60, I_SSCREATE2, 61)
(61, I_SSEVENTS, 62)
(62, I_INITIALACCESS1, 63)
(63, I_INITIALACCESS2, 64)
(64, I_INITIALACCESS3, 65)
(65, I_INITIALACCESS4, 66)
(66, I_SUBSCRIBERINFOQUERY1, 67)
(67, I_SUBSCRIBERINFOQUERY2, 68)
(68, I_SUBSCRIBE1, 69)

```
(69, I_SUBSCRIBE2, 70)
(70, I_SERVICECONTRACTINFOMGMT1, 71)
(71, I_SUBSCRIBERINFOMGMT1, 72)
(72, I_SERVICECONTRACTINFOMGMT2, 73)
(73, I_SUBSCRIBERINFOMGMT2, 74)
(74, I_SUBSCRIPTIONNOTIFY, 75)
(75, I_SERVICEQUERY, 76)
(76, I_SERVICENOTIFY, 77)
(77, I_WRAPUP1, 78)
(78, I_WRAPUP2, 24)
```

Figura 7.21 – Arquivo *ModeloSsgct.aut* – LTS no Formato ALDEBARAN.

A primeira linha do gráfico apresentado na Figura 7.21 mostra que o primeiro estado é “0”, o número de transições é 80 e o número de estados é 79.

Passo 2: Visualização do LTS

Um arquivo chamado *ModeloSsgct.bcg* é criado. Para visualiza-lo basta clicar em “*Visualise*” e a opção “*Draw*” e o seguinte grafo é visualizado, como mostra a Figura 7.22.

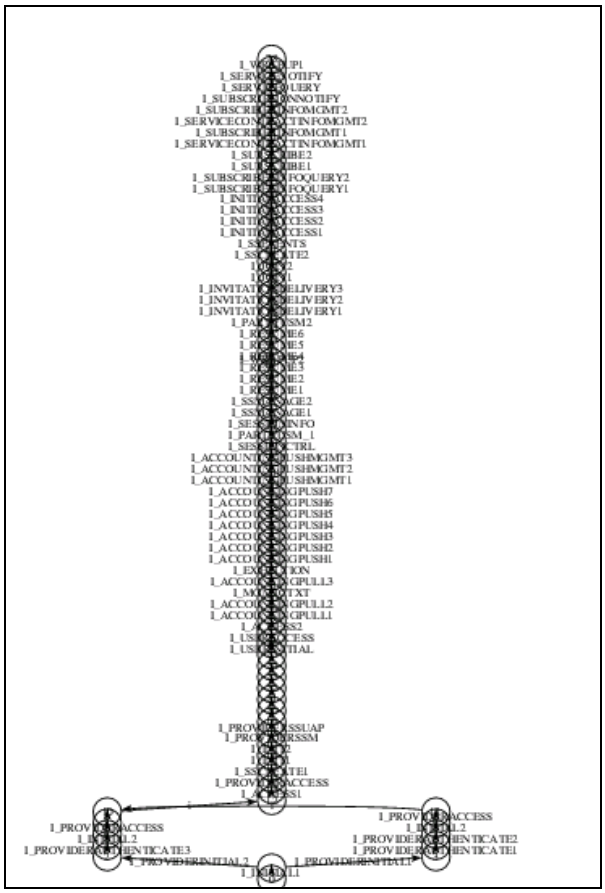


Figura 7.22 – LTS com Transições.

Como pode ser observado, o LTS tem um número de estados e transições, segundo o grafo não muito denso, devido a que as transições são quase de forma seqüencial. Se este fosse muito complexo, pode-se utilizar a opção de redução. Neste caso não se faz necessário já que o resultado é o mesmo.

Passo 3: Geração do LTS

Da mesma maneira como apresentado no Passo 1, neste passo é gerado o LTS associado a especificação *SsgctService.lotos*. Este LTS é pequeno e desta forma não é necessário reduzi-lo. Veja a Figura 7.23.

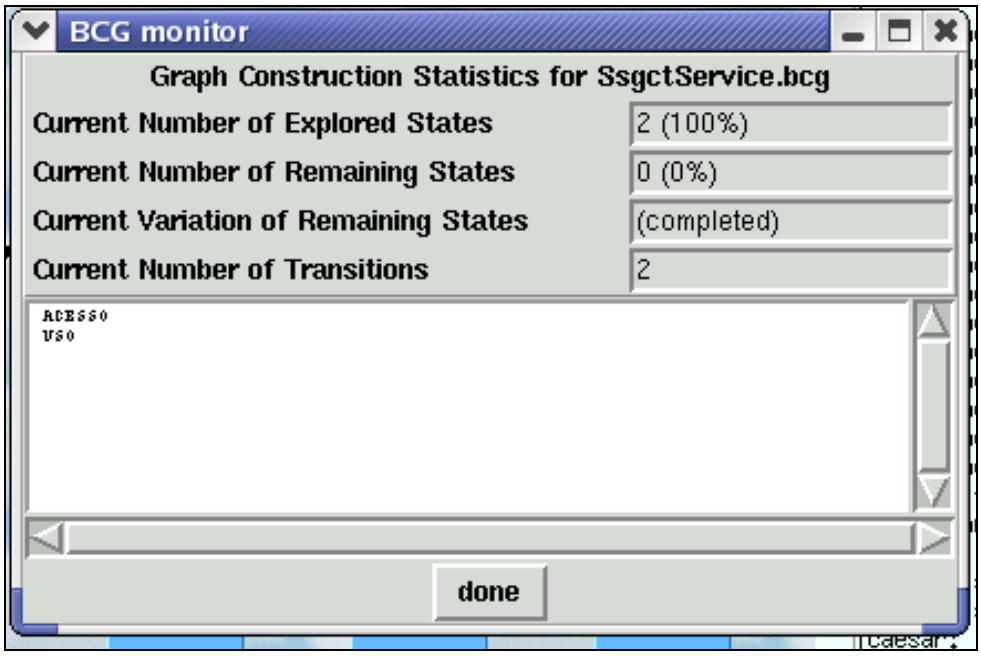


Figura 7.23 – Grafo Gerado para *SsgctService.lotos*.

Na Figura 7.23 pode-se observar que o LTS associado à especificação contém somente 2 estados e 2 transições. Se ao invés de BCG fosse utilizado como opção de grafo ALDEBARAN, as informações seriam apresentadas como mostra a Figura 7.24.

Como pode ser observado, cada passo da análise sintática e semântica foi realizado com sucesso, sem apresentar erros. Veja a Figura 7.25 o LTS no formato ALDEBARAN através do arquivo *SsgctServices.aut*.

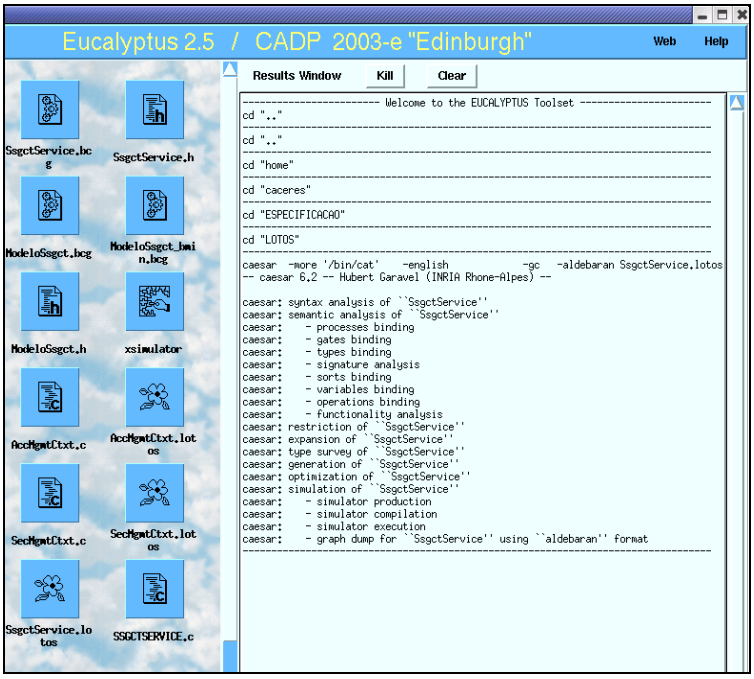


Figura 7.24 – Grafo do *SsgctService.lotos* no Formato ALDEBARAN.

```
des (0, 2, 2)
(0, ACESSO, 1)
(1, USO, 0)
```

Figura 7.25 – *SsgctService.aut* no Formato ALDEBARAN.

Como pode ser observado na Figura 7.25, o LTS inicia no estado “0”, inclui 2 estados e 2 transições.

A seguir mostra-se na Figura 7.26 o tamanho dos arquivos relacionado às especificações de serviço e protocolo.

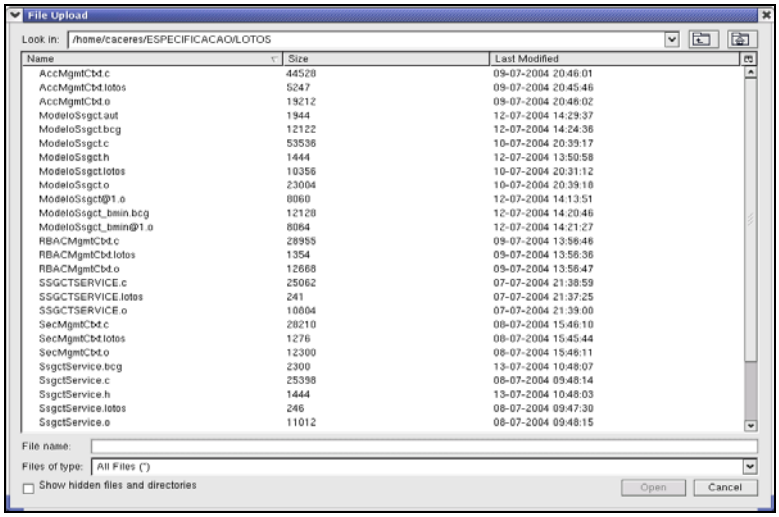


Figura 7.26 – Arquivos Relacionados com os Tamanhos das Especificações.

Passo 4: Comparação de LTS

Neste passo estamos em condições de comparar o LTS que representa o LTS do protocolo com o LTS do serviço esperado. Para conseguir a comparação, posiciona-se no arquivo *ModeloSsgct_bmin.bcg* e selecione "*Compare...*", a seguir aparece a seguinte janela representada pela Figura 7.27.

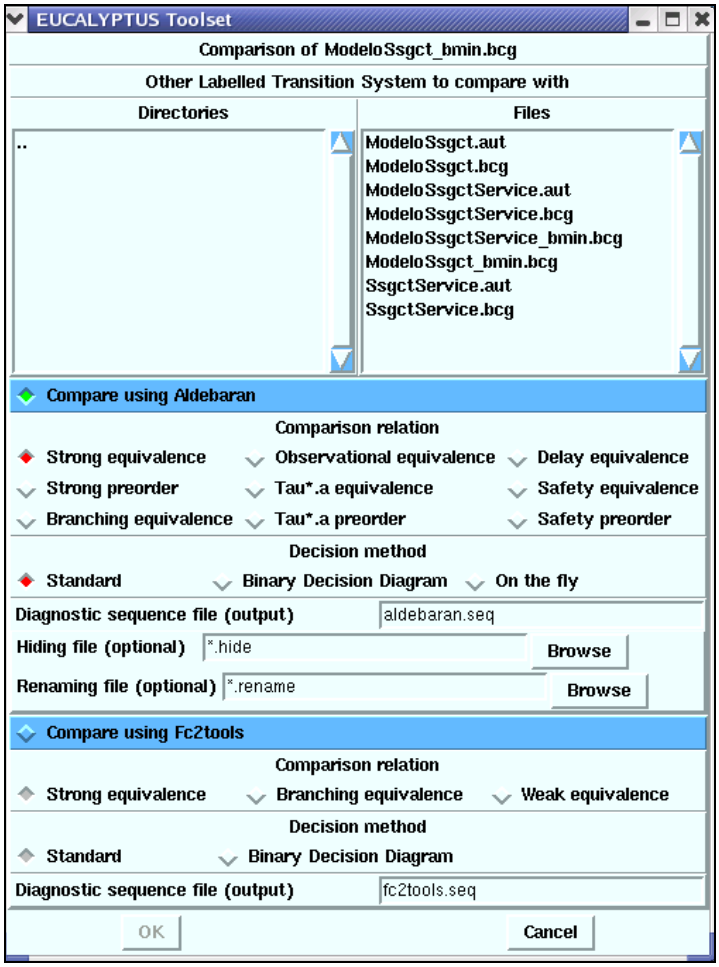


Figura 7.27 – Comparação do LTS.

Nesta janela, é possível escolher a ferramenta para comparar os LTSs (*ALDEBARAN* ou *Fc2tools*[16]), bem como a relação de comparação (*Strong equivalence bisimulation, Observationnal equivalence bissimulation, ...*). Neste caso se selecionou:

- *ModeloSsgcService.bcg* para o LTS a ser comparado;

[16] *Fc2tools*: permite a implementação de processos algébricos para a sintaxes e semânticas. Também permite a verificação por redução e de abstração, permite o uso alternativo ou combinado de estilos de implementação explicita e implícita para uma melhor eficiência.

- *ALDEBARAN* para a ferramenta a ser utilizada;
- *Observational Equivalence* para a relação de comparação; e
- *Standard* para o método de decisão.

Depois de confirmar com "***OK***", o resultado da comparação aparece na parte direita da janela, como mostra a Figura 7.28.

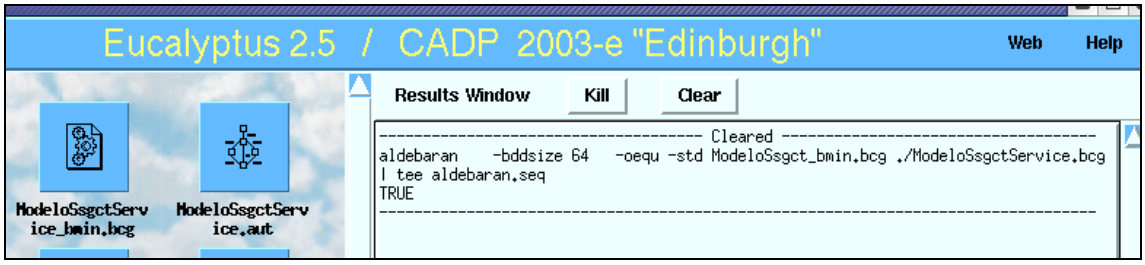


Figura 7.28 – Resultado da Equivalência de Observação.

O resultado "***TRUE***", significa que o protocolo executa o serviço esperado. Na Figura 7.29, apresenta a prova formal de equivalência entre as especificações de observação do *ModeloSsgc*t e o *ModeloSsgctService*, com resultado "*TRUE*" que indica a equivalência entre a especificação inicial do sistema e as especificações finais refinadas.

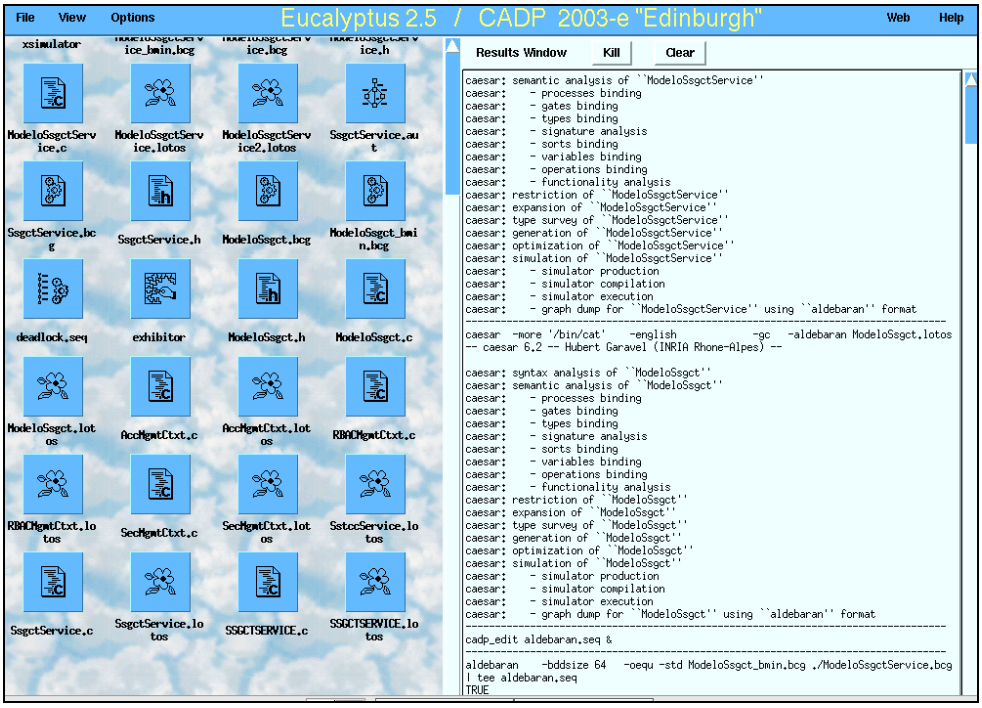


Figura 7.29 – Prova Formal de Equivalência do SSGCT.

Portanto, os resultados mostram que o estudo, análise da arquitetura proposta, mostra estar corretamente especificada e formalmente validada, onde os componentes computacionais e interfaces definidas por TINA, os contextos de segurança *SecMgmtCtxt* e *RBACMgmtCtxt* e de contabilidade *AcctMgmtCtxt* definidos, interagem em um ambiente de serviço multi-domínio, multi-usuário e multi-provedor para o estabelecimento de uma sessão de serviço, visando uma contabilidade confiável, integra e que garantem através das políticas e mecanismos de segurança especificados um ambiente seguro através de um protocolo de negociação e a definição de papéis de uma forma dinâmica com *billing on-line*. Conclui-se também que a contribuição deste trabalho está direcionada ao uso de mecanismos e políticas de segurança de modelos diferentes, que através da arquitetura de serviço TINA podem ser introduzidos em dois processos, de acesso e uso, sem entrar em conflitos de incompatibilidades, o que garante uma contabilidade confiável durante uma sessão de serviço e que demonstra ser um trabalho de pesquisa inédito e que se encontra no estado da arte.

7.7 Conclusão

Neste capítulo se definiu através da técnica de descrição formal LOTOS as especificações do Sistema de Segurança de Gerenciamento de Contabilidade TINA – SSGCT – o qual mostrou-se ser eficiente no desenvolvimento do sistema, garantindo os níveis de segurança estabelecidos, provando ser uma técnica de alto rigor matemático para a especificação e análise e projeto de sistemas distribuídos complexos como o apresentado.

Usou-se para a validação a ferramenta *EUCALYPTUS toolset* que permitiu validar os diferentes refinamentos das especificações. As validações utilizadas foram as simulações, testes para encontrar erros e as verificações que permitem provar formalmente a correção do sistema.

O procedimento utilizado para obter a prova de correção foi a geração de sistemas de transição rotuladas (LTS) para a especificação mais abstrata *SsgctService* associado ao serviço e o *ModeloSsgctService* associado ao protocolo. Por último, a prova formal de correção obtida é garantida através do protocolo ISO 8807 e a seguir no seguinte capítulo descreve-se e mostra-se o protótipo desenvolvido para o modelo validado.

8. RESULTADOS DA IMPLEMENTAÇÃO DO SISTEMA SSGCT

8.1 O Modelo Arquitetural de Contabilidade

A Figura 6.1 da seção 6, mostra o modelo arquitetural do contexto de gerenciamento de contabilidade (*AccMgmtCtxt*) com as interações entre os principais componentes da arquitetura de serviço TINA e contextos de segurança.

O *AcctMgmtCtxt* é composto por componentes que abrangem diferentes aspectos do serviço, do controle de rede e do gerenciamento, a saber:

a) Coletor de Eventos Contábeis

Este componente recebe, coleta e registra os eventos contábeis associados com o estado do Componente de Sessão ou com as mudanças de estado do Componente de Sessão. São registrados também eventos associados com a informação gerada pelo SSM (*Service Session Manager*). O Coletor de Eventos Contábeis é dividido em dois módulos: *EventManagememt* e *SessionComponent*.

b) Tarifação

Este componente é representado por dois módulos:

Módulo *Recovery*: permite fornecer a taxa de cobrança como uma função do estado do componente atual. Este converte os eventos contábeis coletados em registros de cobrança e armazena os mesmos dentro de uma base de dados. Em adição, ele permite restaurar a informação coletada quando ocorre uma falha no serviço.

Módulo *Tariff*: entrega a cobrança atual acumulada como uma função da seqüência de eventos. Este calcula a tarifa, usando a fórmula de cobrança de acordo com o contrato estabelecido e armazena as informações dentro de um registro *billing*.

c) Cobrança

Este componente pode ser automaticamente emitido ao final de um período negociado e definido no contrato com o cliente. Este componente é gerado através da informação de cobrança do registro *billing*. Existem quatro tipos de configurações *billing*: *On-line charging, Shared billing, Third-party billing* e *Credit-debit billing*. A implementação da cobrança na aplicação desenvolvida (protótipo) foi configurada de forma *on-line charging*.

d) Registro de Uso

Durante o tempo de vida da conexão, este componente coleta e controla a aquisição de informações relacionadas com o uso dos recursos da rede gerados pelo LNC (*Layer Network Coordinator*). Este componente também registra os dados coletados para os processos futuros (gerenciamento de Falha e Desempenho).

8.2 Definição dos Componentes no AcctMgmtCtxt

A implementação da aplicação usa o ambiente distribuído CORBA com Visibroker 3.04 (VISIBROKER, 1998) e linguagem de programação JAVA 2.0.2. Os objetos são definidos como interfaces em IDL, assim, eles podem acessar e ser acessados por qualquer outro objeto independentemente da linguagem de programação (OMG, 1999). A definição dos componentes na IDL dentro do contexto AcctMgmtCtxt são definidos na Figura 8.1.

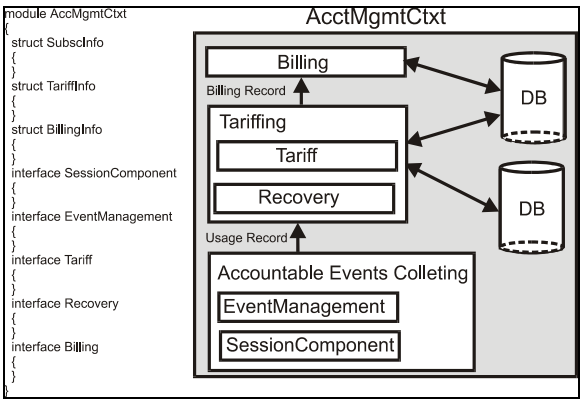


Figura 8.1 - Definição dos Componentes no *AcctMgmtCtxt*.

A construção de uma IDL para a implementação do protótipo permite que as interfaces criadas se comuniquem entre elas. A base de dados definidas na IDL estão divididas em "*SubscripInfo.mdb*" que define o usuário assinante, a "*TariffInfo.mdb*" que registra os eventos dos componentes *Tariffing* enquanto ocorre a execução do serviço e "*BillingInfo.mdb*" que define a tarifa-cobrança do usuário.

O *AcctMgmtCtxt* é o componente mais importante nesta classe, este é gerado na fase de *Set-up* da ST e é destruído quando a ST é concluída (fase *Wrap-up*). A interface *SessionComponent* é usada para o controle e gerenciamento do SC, além de iniciar e encerrar as ações métricas do objeto contábil. O SC referencia um serviço para um

usuário específico. *EventManagement* define o gerenciamento de eventos (por exemplo: *delivery*).

8.3 Implementação e Execução do Protótipo

Devido a alta complexidade necessária para a implementação de todo o conceito da segurança multilateral na arquitetura de contabilidade TINA, decidiu-se limitar a implementação do protótipo na negociação dos objetivos de segurança e na simulação da vídeo conferência com toda a estrutura de contabilidade.

Esta implementação fornece a base para a simulação de uma vídeo-conferência entre diversos usuários. Ela é composta por um módulo servidor e um módulo cliente. O módulo servidor é responsável pelo controle dos usuários e por prover o serviço de vídeo-conferência para estes usuários, que por sua vez, utilizam o módulo cliente para interagir na vídeo-conferência. Diversos usuários podem se conectar ao servidor simultaneamente.

Ao iniciar uma sessão, o usuário deve configurar as suas preferências de segurança que serão utilizadas na negociação para o estabelecimento da sessão. A Figura 8.2 mostra a interface onde o usuário pode escolher quais objetivos de segurança ele irá usar e quais mecanismos existentes ele poderá usar para alcançar o objetivo de segurança desejado.

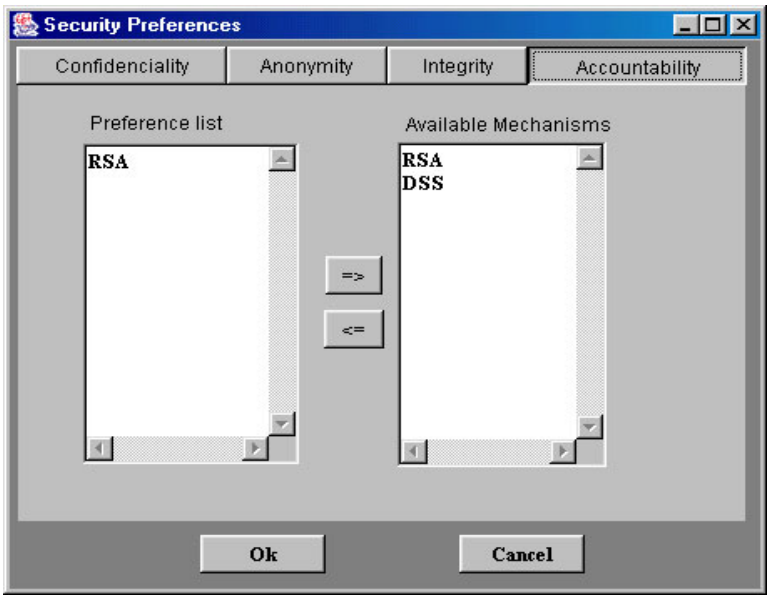


Figura 8.2 - Caixa de Diálogo para Seleção de Preferências de Segurança.

Neste exemplo ilustrado, para alcançar uma contabilidade eficiente, o usuário selecionou o algoritmo RSA. Com as preferências do usuário já definidas, é verificada a necessidade de se fazer a negociação com o outro participante, caso seja necessário, a negociação ocorre como descrito na Figura 5.2.

Após a negociação ser concluída com sucesso, uma segunda interface é visualizada, como mostra a Figura 8.3, sub-dividada nos seguintes módulos:

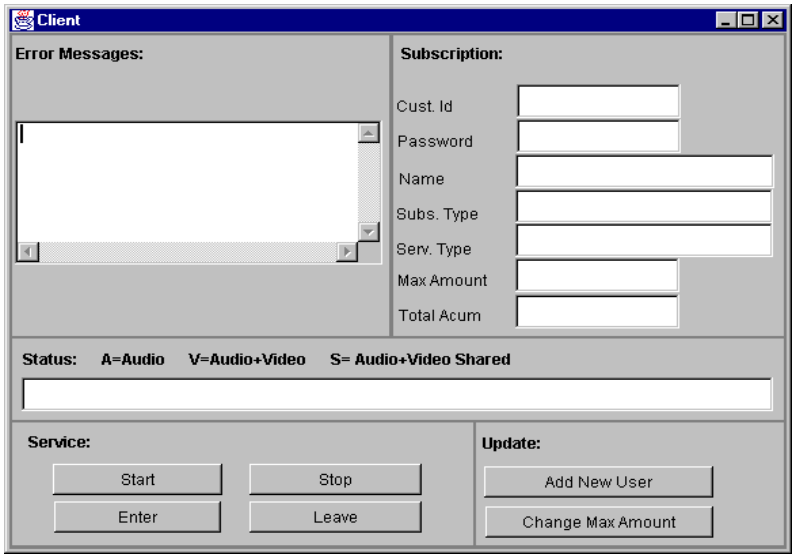


Figura 8.3 - Interface Gráfica do Módulo Usuário.

- ***Subscription*:** login do usuário e visualização dos dados do usuário;
- ***Update*** : atualização do usuário (*Insert, Modify*, *Delete*, *etc*);
- ***Monitoring service***: monitorização do serviço de vídeo conferência para cada usuário; e
- ***Error messages***: visualização de erros que podem ocorrer durante a sessão de serviço.

O módulo cliente está divido em duas fases: fase de Inicialização e a fase de Monitoração.

a) fase de Inicialização

- O protótipo inicia um serviço quando é solicitado por um usuário (módulo *Service*, através do botão "*Start*" na Figura 8.3) que registra seu log, ingressando seu Id e password (módulo *Subscription* na Figura 8.3).

- Imediatamente, uma verificação (acesso à base de dados "*SubscripInfo.mdb*"). O usuário autorizado pode ser classificado em dois tipos:

  1. usuário "*com assinatura*", o qual tem um contrato com o provedor e sua prioridade no uso do serviço é completa: áudio (*fala-escuta*), vídeo (*visão-imagem*), por exemplo, A+V;

  2. usuário "*sem assinatura*", o qual usa o serviço pela primeira vez, não tem um contrato com o provedor, sua prioridade de uso do serviço é parcial e ele só assiste ao serviço: áudio (*escuta*), vídeo (*visão*), por exemplo, A. O novo usuário será registrado na base de dados (módulo *Update*: "*Add New User*" na Figura 8.3) para assegurar que sua quantidade de crédito disponível é suficiente para fazer uso do serviço ou de outra forma, dar um *log-out*, não permitindo o uso do serviço.

b) Fase de Monitoração

- Para cada usuário autorizado, seu próprio *AcctMgmtCtxt* é criado, iniciando o serviço (módulo *Service*, botão "*Enter*" na Figura 8.3).
- Automaticamente é criado o espaço de segurança para a definição dos certificados de papéis dos serviços escolhidos pelo cliente, neste caso a simulação de um vídeo-conferência com seu correspondente objeto *billing* associado ao papel, como descrito no item 5.2.2.1.
- Definido os papéis, automaticamente também, o *AcctMgmtCtxt* cria os objetos contábeis (através das interfaces *SessionComponent* e *EventManagement*) que identificam os eventos gerenciados durante a execução e que estão relacionados diretamente com os objetos definidos no contexto de definição de papéis.

Interface *SessionComponent*	**Interface *EventManagement***
{ SubscInfo SubsValid(in SubscInfor informacoes); ***Void*** AddNewUser(in SubscInfo informacoes); }	{***boolean*** RegService(in ***string*** cid, in ***long*** sertype); ***long*** GetTipoServico(in ***string*** cid, in ***long*** tipo_servico); ***long*** EnterEvent(in ***string*** csid, in ***long*** tipo_serv); ***long*** LeaveEvent(in ***string*** csid, in ***long*** tipo_serv); }

Ao mesmo tempo, um vetor de estado de sessão é gerado (SSVec) onde cada posição (pos) registra a informação relacionada aos eventos do serviço:

1. tempo inicial (ti);
2. tempo final (tf);

3. tempo acumulado pelo "*uso de serviço completo*": A+V (fs – full service) ou "*uso de serviço parcial*": (ps – partial service); e

4. tempo acumulado pelo "*uso de serviço compartilhado*" dos usuários (ts– time shared).

- Ao clicar no botão "*Enter*", o "ti" é inicializado enquanto "fs" ou "ps" são ativados. Como o vetor é visto como uma estrutura dinâmica e volátil, qualquer falha que ocorra durante o serviço é registrado como informação na base de dados. O *backup* é necessário para armazenar a informação (eventos) na base de dados ("*TariffInfo.mdb*") porque sua interação com o módulo Tariffing (interface *Tariff*) é periódica.

```
Interface Tariff
{  Tarval Tariff_val();  }
```

- Durante o período de serviço, o provedor deve estar verificando se o tempo consumido pelo usuário não excedeu 80% do tempo limite disponível. Se isto ocorrer, o provedor notifica ao usuário e o consulta se ele deseja aumentar a quota disponível (módulo *Update*: "*Change Max Amount*" na Figura 8.3), ou fazer *log-out* do serviço (ativa a interface *Billing* para sua cobrança).

```
Interface Billing
{
  void BillingSave(in string cid, in string name, in string date, in long totalsecs, in long totalunits,
          in string inittime, in string endtime, in long asecs, in long avsecs, in long ssecs);
}
```

- O usuário que faz uso do serviço pode liberar a sessão (módulo *Service*, botão "*Leave*" na Figura 8.3). Nesse momento, o usuário entra no estado de áudio (A). Ele também tem a opção de retomar o serviço (módulo *Service*, botão "*Enter*" na Figura 8.3) ou finalizar a sessão (módulo *Service*, botão "*Stop*" na Figura 8.3).
- Existe também a opção de compartilhar o serviço entre vários usuários ao mesmo tempo. Isto acontece quando for ativado o botão "*Enter*" e então são disparados (inicializados) os temporizadores "ts" e "fs" de cada usuário. Para liberar o estado de compartilhamento, é ativado o botão "*Leave*" para ativar "fs" ou o botão "*Stop*" para terminar a sessão.
- Se o usuário está no estado "*Stop*", os temporizadores são desativados, e "tf" é registrado. Finalmente, a cobrança *on-line* (interface *Billing*) é ativada pelos

serviços fornecidos ao usuário e a informação é armazenada na base de dados "*BillingInfo.mdb*", (Figura 8.3).

Uma importante parte do serviço ocorre quando acontecem falhas. Por exemplo, eventos podem ser perdidos durante a sessão devido à perda da sincronização dos temporizadores. O estado contábil do SC pode ser interrompido por falhas na rede, falta de energia, etc. Nesse momento, a interface *Recovery* armazena e salva a informação na base de dados ("*TariffInfo.mdb*"). Se ocorrer uma falha e se o usuário desejar, a sessão é reiniciada e automaticamente se reinicializam os temporizadores para continuar a contabilização dos eventos, ou a sessão é encerrada. Para simular uma falha, clique "*X*" no canto superior direito da janela, e clique "*Start*" para continuar, como mostra a Figura 8.3.

```
Interface Recovery
{  boolean  CheckStatus (in string csid);  }
```

Na Figura 8.4 foram considerados na execução do protótipo três usuários que compartilham a mesma sessão de um "*serviço de vídeo conferência múltiplo*" fornecido pelo *Retailer*, mostrando um exemplo da interface gráfica "*Client*" com sua respectiva interface gráfica "*Billing*" que representa a cobrança on-line de um usuário pelo uso do serviço fornecido pelo *Retailer*.

Os usuários tem três opções de acesso ao serviço de acordo com o contrato estabelecido entre as partes: V (Vídeo+Áudio); A (Áudio); e S (Vídeo+Áudio Compartilhado).

O uso da opção "V" custa 3$/unit, da opção "A" custa 1$/unit e da opção "S" custa 2$/unit.

(Obs.: no Anexo 3 encontra-se o código fonte do protótipo)

8.4 Conclusão

Neste Capítulo foi apresentado o resultado obtido com a implementação do modelo proposto para o gerenciamento de contabilidade TINA. Um tratamento especifico na definição do serviço foi definido, neste caso de vídeo-conferência, devido à alta complexidade da implementação, dando ênfase à negociação inicial entre os participantes num ambiente de segurança multilateral e na execução do gerenciamento

de contabilidade que envolve aspectos de segurança relacionados com a definição dos certificados de papéis para o serviço.

Neste contexto pode-se concluir que os aspectos de segurança incorporados no contexto de gerenciamento de contabilidade são válidos e compatíveis com as especificações definidas no sistema. A seguir serão apresentadas as conclusões finais e trabalhos futuros.

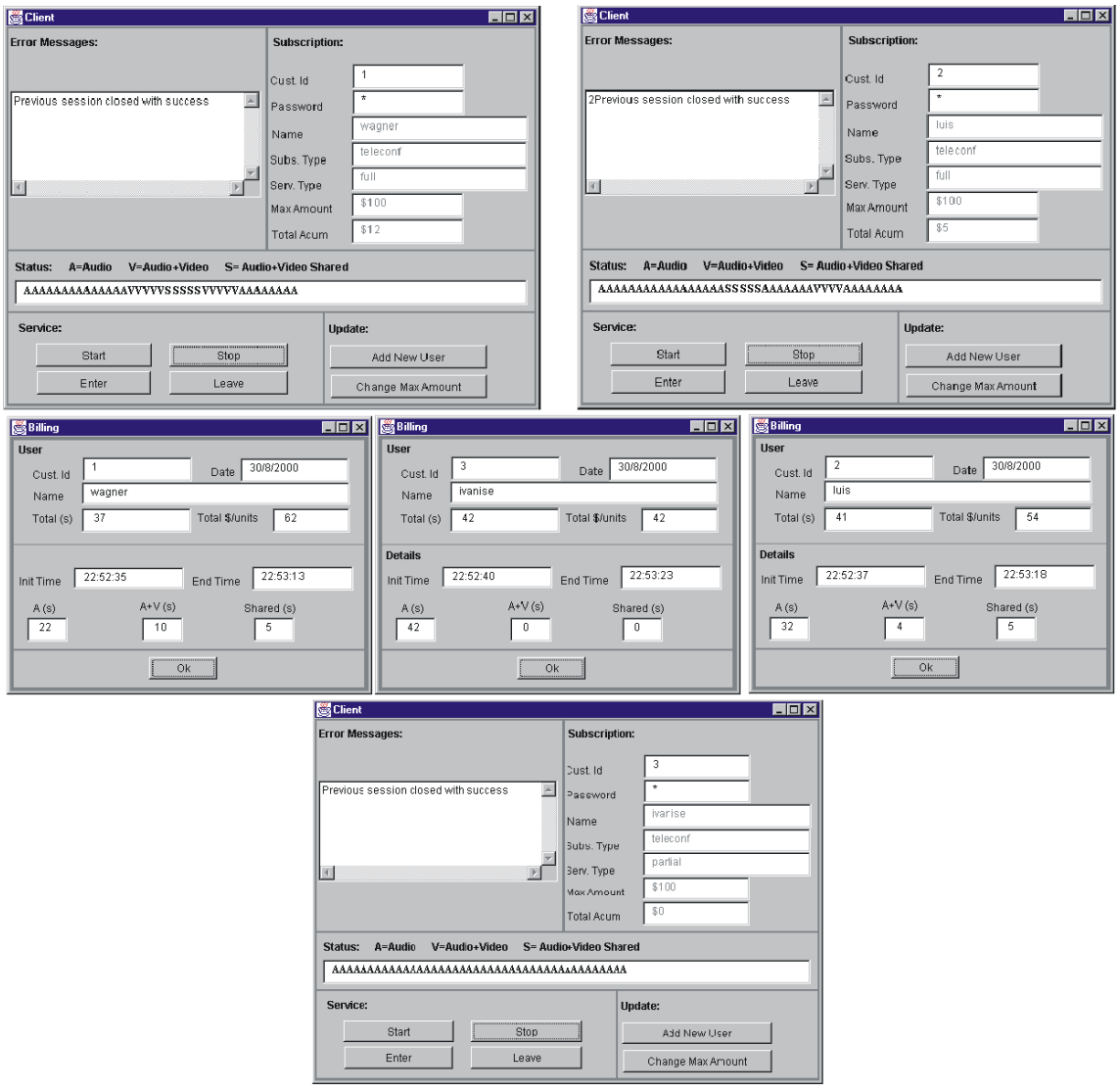


Figura 8.4: Exemplo de Execução do Protótipo.

9. CONCLUSÕES E TRABALHOS FUTUROS

9.1 Conclusões Finais

O desenvolvimento de novos serviços de telecomunicações está impondo cada vez mais desafios aos fornecedores de equipamentos, operadoras e aplicações em um mercado extremamente competitivo. Assim, o volume de aplicações necessárias para suportar os novos serviços e a competitividade pela redução de tarifas impõem novas tendências para o desenvolvimento, implantação e gerência dos serviços de telecomunicações. Nesse sentido, a arquitetura aberta TINA estabelece uma arquitetura para os novos serviços de telecomunicações de forma flexível e de rápida adaptação.

Neste trabalho apresentou-se uma visão geral dos conceitos TINA que especifica o contexto de gerenciamento de contabilidade. Descreve-se as características da contabilidade, requisições e as especificações de gerenciamento de contabilidade. Além disso, apresentam-se os conceitos relacionados à segurança na gerência de contabilidade de TINA.

A partir destes conceitos, se propõe um modelo de segurança para o gerenciamento de contabilidade dos serviços TINA. Este modelo representa uma arquitetura de contabilidade em tempo real no fornecimento de serviços fim-a-fim em um ambiente multi-usuário/multi-provedor. Esta arquitetura segue as recomendações e padrões estabelecidos por TINA, onde claramente são definidos os domínios para o consumidor, provedor e provedor terceirizado, juntamente com seus correspondentes componentes e interfaces que permitem a interação através dos domínios. Para garantir a segurança em uma sessão de serviço, são introduzidos conceitos de segurança através de mecanismos e políticas de segurança estabelecidas pelas tecnologias de Segurança Multilateral e do Controle de Acesso baseado em Papéis (RBAC).

Estes mecanismos de segurança são incorporados em dois processos definidos pela arquitetura de serviços TINA: de **acesso** e de **uso**. O acesso de gerenciamento de serviço em TINA possui muitas funções independentes do serviço, tais como: a autenticação, controle de acesso, contabilização, entre outros. Em uma primeira etapa de negociação, que corresponde ao processo de acesso, foram incorporados os conceitos de Segurança Multilateral, que se ajustam bem ao ambiente de uma sessão de um serviço TINA (o qual corresponde as quatro primeiras fases do ciclo de vida de um processo de

contabilidade e é definido pelo Contexto de Gerenciamento de Segurança (*SecMgmtCtxt*)). Uma segunda etapa corresponde ao processo de uso, onde são incorporados os conceitos de RBAC, que definem o espaço de segurança no qual são analisados e elaborados os papéis, em um ambiente de serviço dinâmico TINA. A designação dos papéis apenas ocorre quando é iniciada a sessão de serviço (que correspondem as três ultimas fases do ciclo de vida do processo contábil e é definido pelo Contexto de Espaço de Segurança (*RBACMgmtCtxt*)). Definidos os contextos de segurança, um ambiente contábil é estabelecido através do contexto de contabilidade *AcctMgmtCtxt*, que visa garantir eventos de contabilidade eficientes e que garantem que o *billing* definido no espaço de segurança seja de acordo ao serviço fornecido pelo provedor em tempo real.

Para validar a complexidade da arquitetura de gerenciamento de contabilidade em tempo real aplicaram-se os passos seguintes:

i) Definiu-se a arquitetura de contabilidade como um todo, onde foram incluídos todos seus componentes e interfaces que interagem, através dos domínios estabelecidos pelos padrões TINA.

ii) Incorporou-se à arquitetura de contabilidade os contextos de segurança multilateral e de controle de acesso baseado em papéis para determinar os requerimentos e políticas de segurança que foram estabelecidos na arquitetura, com o intuito de fornecer um serviço e uma contabilidade de forma eficiente e que garantem que as informações e eventos contábeis não sejam maliciosamente manipulados ou violados pelos participantes ou agentes externos.

iii) Em uma primeira etapa que é definida pela arquitetura de gerenciamento de serviço TINA, chamado de processo de **acesso**, foram incorporadas e estabelecidas as negociações dos requisitos de segurança dos participantes (domínios Consumidor e Provedor) através do contexto de segurança multilateral *SecMgmtCtxt*. Em uma segunda etapa chamada de processo de **uso,** foram incorporados e estabelecidos os espaços de segurança para a emissão de certificados de papéis para o serviço fornecido através do contexto de segurança *RBACMgmtCtxt* no domínio Provedor.

iv) Definidos os contextos, um modelo foi desenvolvido para representar todo um ambiente de serviço e de gerenciamento de contabilidade multi-usuário/multi-provedor em tempo real.

v) Usou-se uma técnica de modelagem de especificação formal LOTOS – ISO 8807 - para a validação da arquitetura de contabilidade de maneira que descreva o comportamento do sistema de acordo com os níveis de gerenciamento, como: gerenciamento de sessão de serviço, gerenciamento dos componentes do serviço e gerenciamento de contabilidade do serviço, que conformam o ciclo de vida.

vi) A ferramenta utilizada é o *EUCALYPTUS toolset 2.5/CADP* que permitiu especificar e verificar formalmente o sistema de segurança para o gerenciamento de contabilidade TINA, o qual através do modelo formal permitiu descrever a forma estrutural do sistema e garantir matematicamente sua correção. Desse modo, verificou-se o correto comportamento do sistema, onde a contabilidade em tempo real é freqüentemente realizada atualizando o crédito do usuário durante uma sessão de serviço.

vii) A implementação do protótipo através de ODP/OMG CORBA e da linguagem JAVA para permitir o suporte e segurança dos sistemas distribuídos num ambiente de serviço multimídia em tempo real com o intuito de garantir a segurança e a privacidade dos participantes de acordo aos mecanismos de segurança estabelecidos.

Por último, depois destes anos de trabalho e de resultados obtidas nesta linha de pesquisa podemos concluir que estes indicam que o sistema proposto a partir do modelo de segurança definido para o gerenciamento de contabilidade de TINA, demonstrou que modelos de segurança diferentes, os quais proporcionam políticas e mecanismos que podem ser incorporados no sistema, podem ser implementados num nível de negociação e de definição de papéis respectivamente, sem ter a desvantagem de incompatibilidades entre eles. Isto permite garantir a segurança num ambiente de serviço TINA através do gerenciamento de contabilidade em tempo real e cujo *billing on-line* é definido especificamente para o cliente que está usando um serviço determinado através do certificado de papel, visando garantir que não se produzam fraudes ou manipulações das informações produzidas pelos eventos contábeis. Isto fica demonstrado através da

aplicação de técnicas de especificação formal que permitiram validar o correto comportamento funcional e estrutural do sistema.

Portanto, este trabalho contribui com aspectos que se encontram no estado da arte, apesar de usar o padrão TINA, do consórcio TINA-C, que foi encerrado no ano 2000, mas que produziu um amplo conjunto de especificações que permitiram desenvolver o presente trabalho.

9.2 Trabalhos Futuros

Uma vez validada a arquitetura de segurança para o gerenciamento de contabilidade através das técnicas mencionadas, é interessante desenvolver uma simulação real de vários tipos de serviços, como videoconferência, teleconferência, ensino à distância, bibliotecas virtuais, etc, fornecida por vários provedores e disponíveis a vários usuários finais, de modo a interagir em um ambiente multi-usuário em tempo real.

Outra proposta de pesquisa seria incluir outros conceitos de gerenciamento FCAPS, como de falhas em uma sessão de serviço fornecido por múltiplos usuários, na qual a qualidade de serviço (QoS) no gerenciamento de serviço seria um dos tópicos mais aprofundados a serem analisados e que, com certeza, influenciam no gerenciamento de contabilidade.

Uma outra linha de pesquisa seria analisar a compatibilidade da arquitetura TINA com outros tipos de arquitetura, como em uma plataforma *Ad Hoc* para a prestação de tais serviços ou de novos, que podem ser incorporados sem afetar sua adaptação.

Por último, à arquitetura pode-se incorporar os tipos abstratos de dados definidos nas especificações LOTOS, de modo que o código C gerado automaticamente possa contribuir e garantir mais segurança e rapidez de implementação.

10. REFERÊNCIAS BIBLIOGRÁFICAS

(ABARCA, 1997) ABARCA, C.; et al. ***Network Resource Architecture***. V. 3.0, TINA-C, [http://www.tinac.com], 1997.

(ABARCA, 1998) ABARCA, C.; et al. ***Service Component Specification Computacional Models and Dynamics***. V.1.0b, TINA-C, [http://www.tinac.com], 1998.

(AGARWAL, KARNIK e KUMAR, 2004) AGARWAL, V.; KARNIK, N; KUMAR, A. ***An Information Model for Metering and Accounting***. 2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(AGOULMINE, 2000) AGOULMINE, N.; et al. ***Trouble Management for Multimedia Service in Multi-Provider Environments***. In: Journal of Network and System Management, Vol. 8, No 1, 2000. Págs.

(ALVAREZ, 2001) ALVAREZ, L. M. C.; et. al. ***Development of a prototype based on TINA Accounting and Security Management Architecture***. XXI International Conference of the Chilean Society of Computer Science. SCCC 2001, IEEE CS Press. Punta Arenas - Chile, Novembro, 2001. Págs. 50-57.

(ANDERSON, 1998) ANDERSON, R.; et al. ***The Steganographic File System***. Cambridge University. [http://www.cl.cam.ac.uk/ftp/users/rja14/sfs3.pdf], 1998.

(ARMANINI, 2003) ARMANINI, K. ***Seguraweb: Um Framework RBAC para Aplicações Web***. Dissertação de Mestrado (Programa de Pós-Graduação em Ciência da Computação). Laboratório de redes e Gerência. Universidade Federal de Santa Catarina. Florianópolis - SC, 2003.

(ARMANINI, 2003a) ARMANINI, K.; et al. ***SeguraWeb: RBAC Framework for Web Applications***. GRES 2003 - Colloque Francophone Sur la Gestion de Réseuax et Service. Fortaleza - CE, 2003. Págs. 161-176.

(ARNOLD, 1989) ARNOLD, A. ***Systèmes de Transitions Finis et Sémantique des Processus Communicants***. Technique et Science Informatiques, Université Bordeaux, 1989. Págs. 193 – 216.

(BLEUMER, 1999) BLEUMER, G. ***Biometric Authentication and Multilateral Security***. In: Multilateral security in Communications, Stuttgard, Alemanha. 1999.

(BRENNAN, 2000) BRENNAN, R.; et al. ***Evolutionary Trends in Intelligent Networks***. In: IEEE Communications Magazine, Junho 2000. Págs. 86 – 93.

(BRINKSMA, 1988) BRINKSMA, E. ***A Tutorial on LOTOS***. ISO 8807 Information Processing Systems – Open System Interconnection – LOTOS. A formal description technique on the temporal ordering of observational behaviour, 1988.

(BUTTYÁN, 1999) BUTTYÁN, L.; et al. ***Multilateral Security in Middleware-Based Telecommunication Architectures***. In: G. Mueller and K. Rannenberg, editors, Multilateral Security in Communications, Volume 3: Technology, Infrastructure, Economy, Addison-Wesley, 1999.

(CLAUB, 2001) CLAUB, S.; et al. ***Identity Management and its Support of Multilateral Security***. Computer Network 37, 2001. Págs. 205 – 219.

(COHEN e RAZ, 2004) COHEN, R., RAZ, D. ***An Open and Modular Approach for a Context Distribution System***. 2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(EHRIG, 1985) EHRIG, H.; MAHR, B. ***Fundamentals of Algebraic Specifications***. Springer-Verlag. 1985.

(EVLOGIMENOU e BOUTABA, 2002) EVLOGIMENOU, A.; BOUTABA, R. ***Programmable Accounting Management for Virtual Private Network***. In: Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002). IEEE Publishing. Florence – Itália. April, 2002.

(FARLEY, 1998) FARLEY, P.; et al. ***Ret Reference Point Specification***. V.1.0, TINA-C, [http://www.tinac.com], 1998.

(FERRAIOLO, 1995) FERRAIOLO, D. F.; et al. ***Role Based Access Control (RBAC): Features and Motivations***. U.S. Department of Commerce. NIST – National Institute of Standards and Technology. 1995.

(FERRAIOLO, 1999) FERRAIOLO, D. F.; et al. ***A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet***. NIST - National Institute of Standards and Technology. 1999.

(FINKELSTEIN, 2000) FINKELSTEIN, M.; et al. ***The Future of the Intelligent Network***. In: IEEE Communications Magazine, Junho 2000. Págs. 100 – 106.

(FREIRE, KARLTON e KOCHER, 1996) Freirer, A. O. ; Karlton, P.; Kocher, P. C. ***Secure Socket Layer 3.0***. Internet Draft. November 1996. http://wp.netscape.com/eng/ssl3/draft302.txt.

(GARAVEL, 2004) GARAVEL, H. ***CADP(CAESAR/ALDEBARAN DEVELOPMENT PACKAGE): A Software Engineering Toolbox for Protocol and Distributed Systems – Version 2003-e***. INRIA/VASY. Grenoble, França, 2004. http://www.inrialpes.fr/vasy/cadp.

(GUSTAFSSON e SHAHMEHRI, 1996) GUSTAFSSON, M.; SHAHMEHRI, N. ***A Role Description Framework and its Applications to Role-Based Access Control***. IEEE WET '96 – International Workshop on enterprise Security at Stanford University, Palo Alto. Junho, 1996.

(HAMADA, 1996) HAMADA, T.; et al. ***Accounting Management Architecture***. TINA-C, [http://www.tinac.com], 1996.

(HAMADA, 1997) HAMADA, T.; et al. ***An Overview of the TINA Management Architecture***. In: Journal of Network and System Management, Vol. 5, No 4, 1997.

(HAMADA, 1997a) HAMADA, T. ***Dynamic Role Creation from Role Class Hierarchy – Security Management of Service Session in Dynamic Service Environment***. In: Proceeding of the TINA '97 – Global Convergence of Telecommunication and Distributed Object Computing, 1997. IEEE.

(HAMADA, 1998) HAMADA, T. ***Role-Based Access Control in Telecommunication Service Management – Dynamic Role Creation and Management in TINA Service Environment***. In: Proceeding of the Third ACM Workshop on Role-Based Access Control, Outubro 1998, USA. Págs. 105 – 113.

(HAMADA, 2004) HAMADA, T.; et al. ***Peer-to-Peer Traffic in Metro Networks: Analysis, Modeling, and Policies***. 2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(HELLEMANS, 1999) HELLEMANS, P.; et al. ***Accounting Management in a TINA-Based Service and Network Environment***. Intelligence in Services and Networks - Paving the Way for an Open Service Market, 6th International Conference on Intelligence and Services in Networks, IS&N'99, Barcelona - Espanha, Abril, 1999. Págs. 13 – 24.

(HWANG, 2004) HWANG, J., et al. ***Transaction Management for Sender/Receiver – Payment Schemes in Charging and Accounting Systems for Interconnected Networks***.

2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(ISO, 1994) ISO/IEC 10764/ITU-T X.900. ***Information Processing – Open Distributed Processing – Basic Reference Model of ODP***. Part 1: Overview and Guide to use, 1994.

(ISO, 1995) ISO/IEC 10165-7. ***Structure of Management Information: General Relationship Model***. (ITU-T Recommendation X.725), 1995.

(ISO, 1997) ISO/IEC 9594-8. Information Technology – Open System Interconnections. ***The directory: Authentication Framework*** (ITU-T Recommendation X.509), 1997.

(ITU, 1993) ITU-T M.3010. ***Principles for a Telecommunication Management Network***. 1993.

(ITU-TX509, 1993) ITU-T. ***Information Technology – The Open System Interconnection – The Directory. Authentication Framework***. ITU-T Recommendation X.509, November 1993. http://www.itu.int/rec/recommendation.asp.

(ITU, 1995) ITU-T Rec.X.903ISO/IEC 10746-3. ***Basic reference Model of Open Distributed Processing***. Part 3: Architecture. 1995.

(JANSEN, 1998) JANSEN, W. A. ***A Revised Model for Role-Based Access Control***. NIST 6192, 1998.

(KELLEY, 2004) KELLEY, D. ***Security Management Convergence via SIM (Security Information Management) – A Requirements Perspective***. Journal of Network and Systems Management, Vol. 12 - Report, No. 1, Março 2004. págs. 137 – 144.

(KORMANN, 1996) KORMANN, L. F.; et al. ***OSI Usage Metering Function for the OSIMIS Management Platform***. In: Journal of Network and Systems Management. Vol. 4, No. 3, 1996.

(KOUTEPAS, STAMATELOPOULOS e MAGLARIS, 2004) KOUTEPAS, G.; STAMATELOPOULOS, f.; MAGLARIS, B. ***Distributed Management Architecture for Cooperative Detection and Reaction to DDoS Attacks***. Journal of Network and Systems Management, Vol. 12, No. 1, Março 2004. págs. 73 – 94.

(KRISTIANSEN, 1997) KRISTIANSEN, L.; et al. ***TINA Service Architecture 5.0***. TINA-C, [http://www.tinac.com], 1997.

(LARMAN, 2000) LARMAN, C. ***Utilizando UML e Padrões: uma Introdução à análise e ao Projeto Orientado a Objeto***. Trad. Luiz A Meirelles Salgado - Porto Alegre: Bookman, 2000. ISBN 85-730-651-8.

(LUNARDI e DOTTI, 2001) LUNARDI, S.; DOTTI, F. ***Uma Camada de Adaptação à Qualidade de Serviço na Internet para Aplicações Multimídia***. In: 19º Simpósio Brasileiro de redes de Computadores – SRBC 2001. Florianópolis – SC. Maio, 2001.

(MAMPAEY e COUTURIER, 2000) MAMPAEY, M.; COUTUIRER, A. ***Using TINA Concepts for IN Evolution***. In: IEEE Communications Magazine, Junho 2000 – págs 94 – 99.

(MILNER, 1980) MILNER, R. ***A Calculus of Communicating System***. Lecture Notes in Computer Science, 824. Springer-Verlag. Berlin, 1980.

(MORAES e FARINES, 2001) MORAES, A; FARINES, J-M. ***Transmissão de Vídeo sobre o Serviço ABR/ATM para Aplicações de Videoconferência, em Canais de Baixa Taxa de Bits***. In: 19º Simpósio Brasileiro de redes de Computadores – SRBC 2001. Florianópolis - SC. Maio, 2001.

(MULDER, 1997) MULDER, H.; et al. ***TINA Business Model and Reference Points***. TINA-C, [http://www.tinac.com], 1997.

(NEUMAN e TS'O, 1994) NEUMAN, C.; TS'O, T. ***Kerberos: Na authentication Service for Computer Networks***. IEEE Communications Magazine, Septembro 1994. Págs. 33 - 38.

(NIEHAUS, 1999) NIEHAUS, D. ***Telecommunications Information Networking Architecture***. In: ITTC – Information and Telecommunication Technology Center. University of Kansa. [http://Hegel.ittc.ukans.Edu/projects/tina-c], 1999.

(NOTARE, 2000) NOTARE, M.S.M.A. ***Concepção, Desenvolvimento e Análise de um sistema de Gerência de Segurança para Redes de Telecomunicações***. Florianópolis, 2000. Tese de Doutorado em Ciências da Computação – UFSC.

(OH e PARK, 2000) OH, S.; PARK S. ***Enterprise Model as a Basis of Administration on Role-Based Access Control***. Dept. of Computer Science, Sogang University, Seoul, Korea. IEEE – Cooperative Database Systems for Advanced Applications, 2001, CODAS 2001. The proceedings of the Third Intenational Symposium on 2000. Págs. 150 – 158.

(OMG, 1999) OMG. ***The Common Object Request Broker: Architecture and Specification – Version 2.3.1.***. Document formal/99-10-07, Object Management Group. [http://www.omg.org.], Outubro 1999.

(OMNI, 1994) OMNI Point Technical Architecture. ***Delivering a Management System Framework to Enable Service Management Solutions***. In: Network Management Fórum, 1994.

(PAVLOU e GRIFFIN, 1997) PAVLOU, G.; GRIFFIN, D. ***Realizing TMN-like Management Services in TINA***. In: Journal of Network and Systems Management. Vol. 5, No. 4, 1997.

(PAVLOU, 1998) PAVLOU, G.; et al. ***An Evolutionary Approach towards the Future Integration of IN and TMN***. In: Interoperable Communication Networks 1, 1998.

(PARK e BAEK, 2001) PARK, J.; BAEK, J. ***Management of Service Level Agreements for Multimedia Internet Service Using a Utility Model***. In: IEEE Communications Magazine, Junho 2001 – págs 100 – 1006.

(PFITZMANN, 1998) PFITZMANN, A.; et al. ***A Java-Based Distributed Platform Multilateral Security***. In: Lectures Notes in Computer Science, 1998.

(PFITZMANN, 2001) PFITZMANN, A. ***Multilateral Security: Enabling Technologies and treir Evolution***. In: Informatics: 10 Years Back. 10 yerars Ahead. LNCS, 2001.

(PFITZMANN, 2002) PFITZMANN, A.; et al. ***Striking a Balance between Cyber-Crime Prevention and Privacy***. IPTS – Institute for Prospective Technological Studies. Report vol. 57, 2002.

(PILZ, 2004) PILZ, A. ***"Policy-Maker": a Toolkit for Policy-Based Security Management***. 2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(PINTO, 2002) PINTO, R. P.; et al. ***Uma Arquitetura para Disponibilização e Gerência de Serviços na Internet***. In: 20º Simpósio Brasileiro de redes de Computadores – SRBC 2002. Búzios - RJ. Maio, 2002.

(PIRES, 1994) PIRES, L. F. ***Architectural Notes: a framework for distributed systems development***. Enschede. The Netherlands: CIP – Gegevens Koninklijke Bibliotheek, 1994.

(PRAS, 2001) PRAS, A.; et al. ***Internet Accounting***. In: IP-Oriented Operations and Management. IEEE Communications Magazine, Maio 2001. Págs. 108 – 113.

(PROZELLER, 1997) PROZELLER, P. ***TINA and the Software of the Telecom Network of the Future***. In: Journal of Network and Systems Management. Vol. 5, No. 4, 1997.

(QUEIROZ, 1994) QUEIROZ, J. A M. de; CUNHA, P. R. F. ***Sistemas Distribuídos: de especificações LOTOS a implementações***. IX Escola de Computação. Recife. Julho 1994.

(RADISIC, 2002) RADISIC, I. ***Using Policy-Based Concepts to Provide Service Oriented Accounting Management***. In: Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002). IEEE Publishing. Florence – Itália. Abril, 2002.

(RAJAHALME, 1997) RAJAHALME, J.; et al. ***Quality of Service Negotiation in TINA***. In: Proceeding of the TINA '97 – Global Convergence of telecommunications and Distributed Object Computing. IEEE, 1997.

(RANNENBERG, 2000) RANNENBERG K. ***Multilateral Security – A Concept and Examples for Balanced Security***. In: New Security Paradigms Workshop 2000. Irland, 2000.

(REICHENBACH, 2002) REICHENBACH, M.; et al. ***Individual Risk Management for Digital Payment System***. In: Computational Economics from Economics Workimg Paper Archive at WUSTL. [http://econpapers.hhs.se/paper/wpawuwpco/0204001.htm], 2002.

(RUMBAUGH, 1991) RUMBAUGH, J.; et al. ***Object-Oriented Modeling and Design***. Prentice-Hall, 1991.

(SAILER, 1998) SAILER, R. ***An Evolutionary Approach to Multilaterally Secure Services in ISDN/IN***. In: Seventh International Conference on Computer Communications and Networks, Louisiana, 1998. Págs. 276 – 283.

(SANDHU e SAMARATI, 1994) SANDHU, R.; SAMARATI, P. ***Acces Control: Principle and Practice***. IEEE Communications Magazine, Volume 32 Issue: 9 Sept. 1994. Págs. 40 – 48.

(SANDHU e COYNE, 1996) SANDHU, R.; COYNE, E. ***Role-Based Acess Control Models***. IEEE Computer. Vol. 29, No 2. Fevereiro, 1996. Págs. 38 – 47.

(SEKKAKI, 2001) SEKKAKI, A.; et al. ***Development of a Prototype Based on TINA Accounting Management Architecture***. In: IFIP/IEEE International Simposium on Integrated Network Management, 2001. V.1. Págs. 100-120.

(SEKKAKI, 2001a) SEKKAKI, A., et al. ***Security within TINA Accounting Architeture Management***. In: ICC2001 – International Conference on Communications, Helsinki – Finlândia.. Junho, 2001.

(SEKKAKI, 2001b) SEKKAKI, A., et al. ***Accounting Management based Service Environment in a TINA Architecture***. In: ICC2001 – International Conference on Communications, Helsinki – Finlândia.. Junho, 2001.

(SEKKAKI, 2001c) SEKKAKI, A., et al. ***Development of Accounting Management based Service Environment in TINA, JAVA and CORBA Architectures***. International Conference on Networking – CREF. Colmar - França. Julho, 2001.

(SEKAKKI, 2001d) SEKKAKI, A.; et. al. ***A Formal Specification and Verification of TINA Architecture Service with the Accounting Management Module***. GRES 2001 - Colloque Francophone sur la Gestion de Réseuax et Service. Marrakech – Marrocos. Dezembro, 2001. Págs. 204-218.

(SEKKAKI, 2002) SEKKAKI, A. ***Gestion dês Systemes Distribues: Cas de la Comptabilite et de la Securite***. Tese de Doutorado de Estado. Spécialé: Informatique. Faculte Dês Sciences – Aïn Chock. Université Hassan II Aïn Chock, Casablanca, 2002.

(SIMON, 1996) SIMON, E. ***Distributed Information Systems: from client/server to distributed multimedia***. Maidenhead, Berkshire, England: McGraw-Hill, 1996. 414 p.

(SOUZA, 2001) SOUZA, I. V.; et. al. ***Formal Specification and Verification of the Accounting Model of TINA Architecture using LOTOS and ALDEBARAN***. Congresso Brasileiro de Computação. Itajaí – SC. Agosto, 2001.

(STAAMANN, 1997) STAAMANN, S.; et al. ***Security in the Telecommunications Information Networking Architecture – The CrysTINA Approach***. In: Proceedings of the TINA ‘97 – Global Convergence of Telecommunication and Distributed Object Computating, 1997. IEEE.

(STAAMMANN e BUTTYÁN, 1997) STAAMMANN, S.; BUTTYÁN, L. ***Security in the Telecommunication Information Networking Architecture – the CrySTINA Approach***. In: Proceedings of the TINA ‘97 – Global Convergence of Telecommunication and Distributed Object Computating, 1997. IEEE.

(THALER e RAVISHANKAR, 2004) THALER, D., RAVISHANKAR, Ch. ***An Architecture for Inter-Domain Troubleshooting***. Journal of Network and Systems Management, Vol. 12 - Report, No. 1, Março 2004. págs. 155 - 189.

(TINA, 2000) TINA. ***About TINA-C***. [http://www.tinac.com/about/about.html], 2000.

(VAN LE, 2002) VAN LE, M.; et al**.** ***Formal Modeling of Service Session Management***. IFIP/IEEE International Conference on Management of Multimedia Networks and Services, MMNS 2002. Santa Bárbara – USA, 2002. Págs. 36 – 48.

(VAN LE, VAN BEIJNUM e HUITEMA, 2004) VAN LE, M.; VAN BEIJNUM, B. J. F.; HUITEMA, G. B. ***A Service Component – Based Accounting and Charging Architecture to Support Interim Mechanisms across Multiple Domains***. 2004 IEEE/IFIP Network Operations&Management Symposium. NOMS 2004. COEX Convention Center Seoul, Korea. 19 – 23 April 2004.

(VISIBROKER, 1998) VISIBROKER 3.01. http://www.inspire.com/

(VISSERS, 1988) VISSERS, C. A.; SCOLLO, G.; SINDEREN, M. V. ***Architecture and Specification Style in Formal Descriptions of Distributed Systems***. University of Twente. The Netherlands, 1988.

(WATANABE, 2001) WATANABE, W. T.; et al. ***Accounting Management based on TINA Service and CORBA Architecture***. XIX Simpósio Brasileiro de Telecomunicações. Fortaleza – CE. Setembro, 2001.

(WATANABE, 2003) WATANABE, W. T. ***Desenvolvimento de um Modelo de Segurança no Gerenciamento de Contabilidade Baseado na Arquitetura TINA***. Dissertação de Mestrado (Programa de Pós-Graduação em Ciência da Computação). Laboratório de Redes e Gerência. Universidade Federal de Santa Catarina. Florianópolis - SC, 2003.

(WESTPHALL, 2000) WESTPHALL, C. M. ***Um Esquema de Autorização para a Segurança em Sistemas Distribuídos de Larga Escala***. 2000. Curso de Pós – Graduação em Engenharia elétrica. Universidade Federal de Santa Catarina.

(WESTPHALL, 2003) WESTPHALL, C. B.; et. al. ***Extending TINA with Secure On-Line Accounting Services***. Journal of Network and Systems Management. Plenum Publishing Corporation. Meddletown, USA (2003, Vol.11 , No. 4), 2003. Págs. 379 – 397.

ANEXO 1 - DEFENIÇÃO DAS INTERFACES QUE INTERAGEM COM OS COMPONENTES COMPUACIONAIS DEFINIDOS POR TINA

1.1 Introdução

Este documento permitirá a ajudar a entender as interações através das interfaces especificadas na Arquitetura de Serviço TINA (ABARCA, 1998), com o propósito de mostrar que interfaces suportam os componentes computacionais e quais são as operações definidas para cada uma delas.

Desse modo, enquanto a Arquitetura de Serviço só descreve os conceitos e os componentes de serviço (como descrito no Capítulo 2), este documento permitirá descrever de forma mas detalhadas quais são, o que fazem e que operações realizam.

1.2 Definição das Interfaces

1. ***i_Access()***:

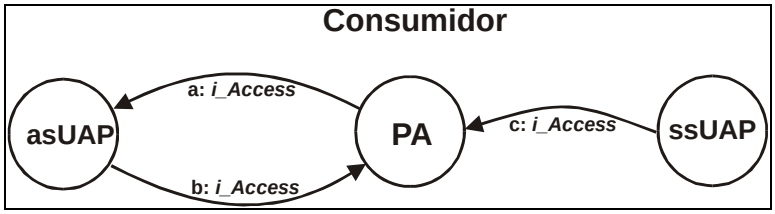


Figura 1 – Interface *i_Acces*:

a) PA → asUAP;

b) asUAP → PA; e

c) ssUAP → PA.

a. PA → asUAP

- Permite ao PA informar ao Consumidor (*Consumer*) de eventos associados com sua sessão de acesso, tais como, convites, nova sessão, etc.
- Convida um usuário a unir-se a uma sessão.

b. asUAP → PA

- Esta interface permite a um consumidor conhecido acessar a seus serviços subscritos.
- O consumidor usa esta interface para todas as operações dentro de uma sessão de acesso simples com o provedor (*Retailer*).

- Esta interface é retornada (ativada) quando o consumidor tem sido autenticado pelo provedor e uma sessão de acesso tem sido estabelecida.
- As operações principais são:
  - sair desde um provedor;
  - listar serviços subscritos;
  - retomar uma sessão de serviço;
  - retomar a participação numa sessão de serviço;
  - convidar um usuário a unir-se à sessão;
  - registrar-se para receber convites fora de uma sessão de acesso; e
  - iniciar uma sessão de serviço e selecionar o modelo da sessão.

c. ssUAP → PA

- Inicia uma sessão de serviço e seleciona o modelo do serviço.

2. ***i_Initial()***:

a. asUAP → PA

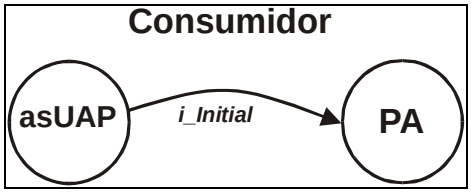


Figura 2 – Interface *i_Initial*:
asUAP → PA.

- O PA oferece esta operação para o asUAP, para que o consumidor entre em contato com um provedor e assim permitir ao consumidor estabelecer uma sessão de acesso.
- Permite ao consumidor identificar ele mesmo um provedor e estabelecer uma sessão de acesso. Um contexto de segurança deve ter sido configurado entre o consumidor e o provedor usando os serviços de segurança CORBA (*Common Object Request Broker Architecture*). Neste caso, esta operação retorna uma referência para a interface *i_ProviderAccess*. Se o cliente foi autenticado, então uma exceção *e_AuthenticateError* será emitida. Esta contém uma referência à interface *i_ProviderAuthenticate*, a qual pode ser usada para autenticar e configurar o contexto de segurança. Então esta operação deve ser invocada novamente para recuperar a referência à interface *i_ProviderAccess*.

- Permite ao cliente estabelecer uma sessão de acesso com o provedor sem revelar sua identidade. A sessão de acesso proverá acesso a alguns serviços, embora o consumidor necessite negociar com o provedor quais serviços estão disponíveis.
- As operações principais são:
  - contatar um provedor; e
  - Logar-se em um provedor como usuário conhecido ou anônimo.

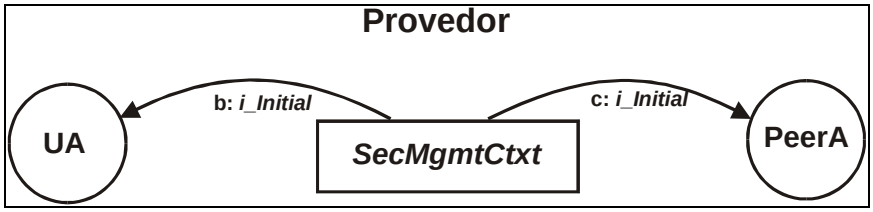


Figura 3 – Interface *i_Initial*:
(b) *SecMgmtCtxt* → UA; e
(c) *SecMgmtCtxt* → PeerA.

b. *SecMgmtCtxt* → UA
- Permite obter a referência da interface *i_ProviderAccess* que é necessária para dar início à interação de uma sessão de acesso.

c. *SecMgmtCtxt* → PeerA
- Permite a verificação de um usuário subscrito num serviço.

3. ***i_ProviderAuthenticate()***:

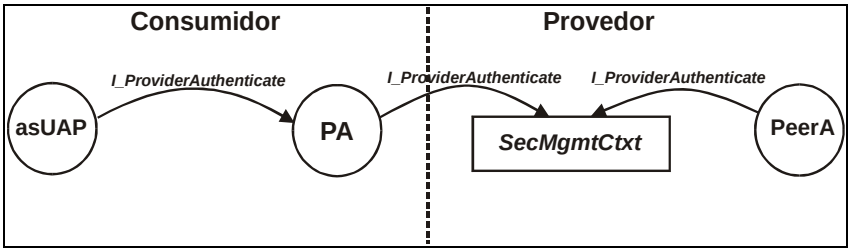


Figura 4 – Interface *i_ProviderAuthenticate*.

- Esta interface provê um conjunto genérico de operações que podem ser usadas para "transportar" informações de autenticação entre os domínios autenticando-se (usuário e provedor). De forma a usar este método de autenticação, ambos domínios devem conhecer o método de autenticação que usaram para gerar os dados de autenticação transportados.
- As operações principais são:

- Logar em um provedor como usuário conhecido (PeerA) ou anônimo (PA).

4. ***i_ProviderInitial()***:

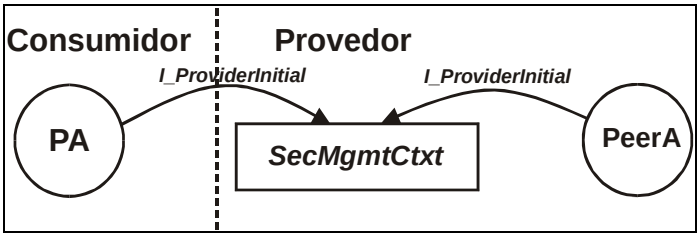


Figura 5 – Interface *i_ProviderInitial*.

- O PA invoca esta interface aos provedores, para estabelecer uma sessão de acesso que permita ao usuário acessar aos serviços que está subscrito.
- Neste estado, um contexto de segurança entre o usuário e provedor deve ter sido estabelecido através do serviço de segurança *SecMgmtCtxt*. Se for o caso, não é necessária a autenticação do usuário e provedor, então esta operação retorna uma referência à interface *i_ProviderAccess*.
- Se o controle de segurança entre o usuário e provedor não foi estabelecido ainda, o provedor não permitirá que uma sessão de acesso seja configurada, assim o *SecMgmtCtxt* retornará uma exceção *e_AuthenticateError* que contem uma referência à interface *i_ProviderAuthenticate*. Esta interface pode ser usada para autenticar os domínios do usuário e provedor e estabelecer um contexto de segurança (Contexto definido como de Segurança Multilateral).
- A invocação a PA retorna uma exceção também para asUAP.
- As operações principais são:
  - Logar-se em um provedor como usuário conhecido ou anônimo.

5. ***i_ProviderAccess()***:

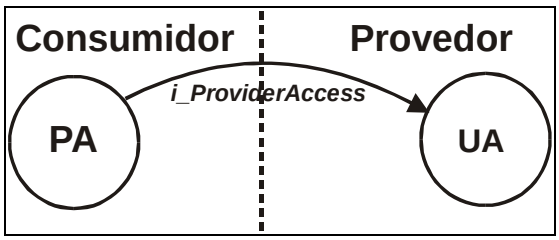


Figura 6 – Interface *i_ProviderAccess*.

- Esta interface permite ao usuário acessar a seus serviços subscritos. O usuário usa esta interface para todas as operações dentro de uma sessão de acesso com o provedor.
- Esta interface é retornada quando o usuário foi autenticado pelo provedor e uma sessão de acesso foi estabelecida.
- As operações principais são:
  - logar-se em um provedor como usuário conhecido ou anônimo;
  - sair de um provedor;
  - retomar a participação numa sessão de serviço; e
  - retomar uma sessão de serviço

6. ***i_UserInitial()***:

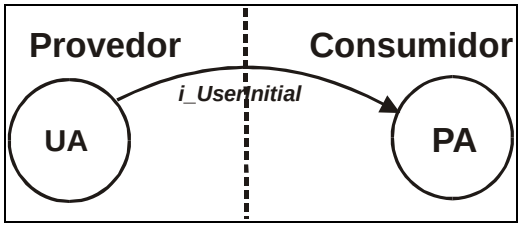


Figura 7 – Interface *i_UserInitial*.

- Esta interface permite entregar o convite ao terminal se o UA possui a referência atual de PA para usar fora de uma sessão de serviço.
- As operações principais são:
  - convida um usuário a unir-se à sessão.

7. ***i_UserAccess()***:

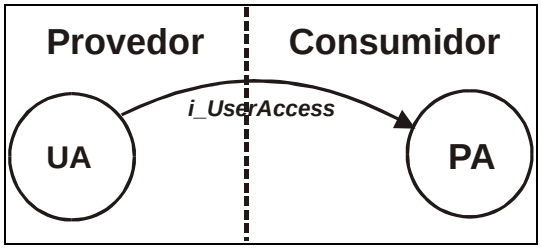


Figura 8 – Interface *i_UserAccess*.

- As operações principais são:
  - cancela a sessão de acesso;
  - cancela o convite ao usuário; e
  - permite buscar informações da sessão de acesso.

8. ***i_AccessInitialize()***:

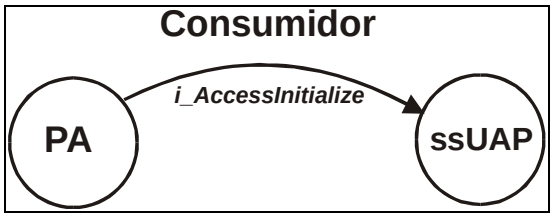


Figura 9 – Interface *i_AccessInitialize*.

- Esta interface é oferecida para o PA de forma que a aplicação possa ser solicitada para responder a convites para a sessão de serviço a qual suporta o ssUAP.
- As operações principais são:
  - inicia uma sessão de serviço e seleciona o modelo da sessão; e
  - une-se à sessão de serviço com convite.

9. ***i_MgmtCtxt()***:

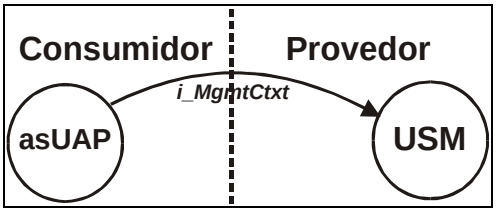


Figura 10 – Interface *i_MgmtCtxt*.

- Esta interface permite unir, liberar e recuperar um conjunto de contexto de gerenciamento para a sessão de serviço em questão.

10. ***i_AccountingPull()***:

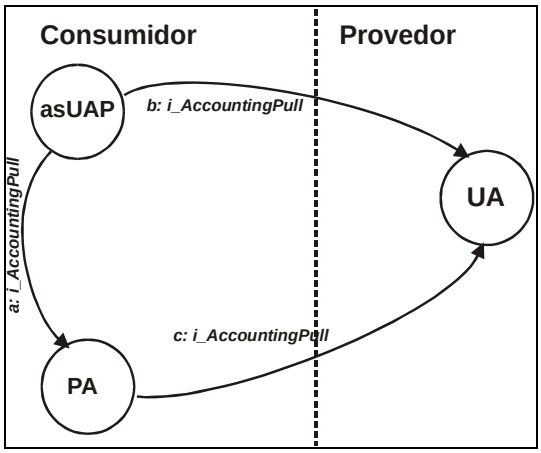


Figura 11 – Interface *i_AccountingPull*.

a. asUAP → PA

- Esta interface permite ao asUAP recuperar dados contábeis desde PA.
- Permite a asUAP enviar uma solicitação a PA, o qual retorna uma listagem dos registros *billing* a asUAP como resposta. O asUAP é capaz de consultar o estado do *billing* do usuário.
- Esta interface provê as seguintes operações:
  - permite ao asUAP recuperar dados contábeis sobre o usuário especificando o intervalo de tempo; e
  - permite ao asUAP para recuperar dados contábeis sobre uma sessão de serviço específica do usuário que tem tomado parte nele.

b. asUAP → UA

- Esta interface permite ao cliente recuperar dados contábeis desde UA.
- Esta interface provê as seguintes operações:
  - permite ao cliente recuperar dados contábeis sobre o usuário UA especificando o intervalo de tempo; e
  - permite ao cliente recuperar dados contábeis sobre uma sessão de serviço específica do usuário UA que tem tomado parte nele.

c. PA → UA

- Esta interface permite periodicamente mostrar informações de uma sessão de serviço ativa via o correspondente PA, quando é usado o *billing* on-line.O usuário pode consultar o estado do *billing* usando o PA, o qual solicita informações de *billing* desde UA.
- Também retorna uma listagem dos registros *billing* para PA como resposta.

11. ***i_AccountingPush()***:

a. USM → UA ← SSM

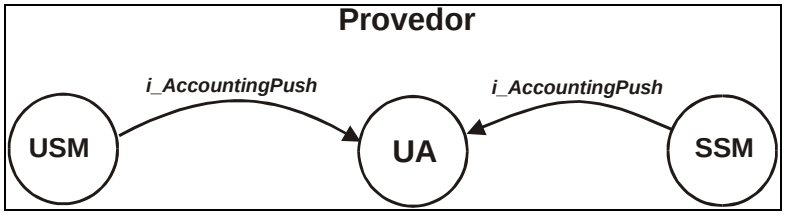


Figura 12 – Interface *i_AccountingPush*:

USM → UA ← SSM.

- Esta interface permite ao cliente armazenar dados contábeis em UA.

- Este provê as seguintes operações:
  - permite ao cliente armazenar um evento de *billing* específico no UA;
  - permite ao cliente armazenar uma lista de eventos de *billing* no UA;
  - permite ao cliente remover um evento específico de *billing* previamente armazenado no UA;
  - permite ao cliente remover uma lista de eventos *billing* previamente armazenado no UA; e
  - permite ao cliente remover entradas *log* correspondendo a um intervalo de tempo específico.

b. USM → SSM, CC → SSM, CSM → SSM, PeerA→ SSM

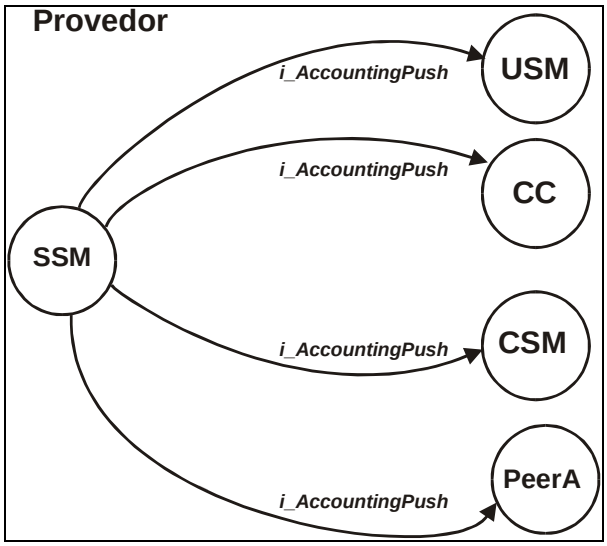


Figura 13 – Interface *i_AccountingPush*:
USM → SSM, CC → SSM, CSM → SSM, PeerA→ SSM.

- Definido sob o gerenciamento de contabilidade.
- A interface permite eventos contábeis que são colocados no USM desde outros objetos de sessão, o SSM, como conseqüência da atividade da sessão.

12. ***i_AccountingPushMgmt()***:

a. USM → SSM ← SF

- Definido sob o gerenciamento de contabilidade.

- A interface permite gerenciar os eventos contábeis relatando o que é colocado na interface de contabilidade pelo SSM como uma conseqüência da atividade da sessão.

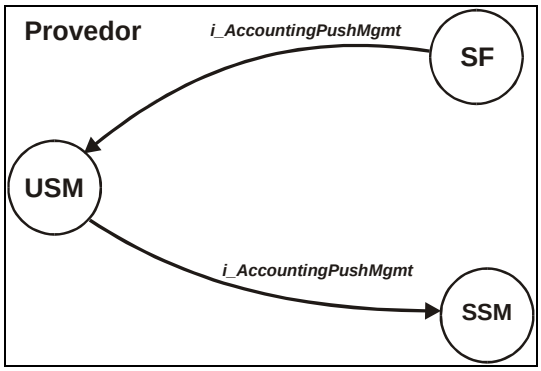


Figura 14 – Interface *i_AccountingPushMgmt*:

USM → SSM ← SF.

b. UA → USM

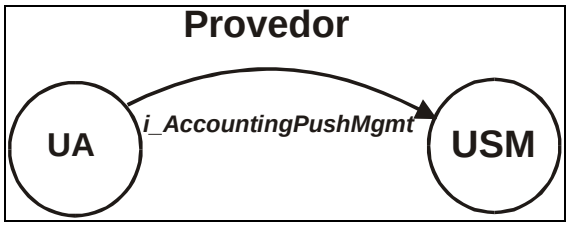


Figura 15 – Interface *i_AccountingPushMgmt*:

UA → USM.

- A interface permite gerenciar os eventos contábeis relatando o que é colocado na interface de contabilidade pelo USM como uma conseqüência da atividade da sessão.

13. ***i_PartyUSM() e i_ProviderssUAP()***:

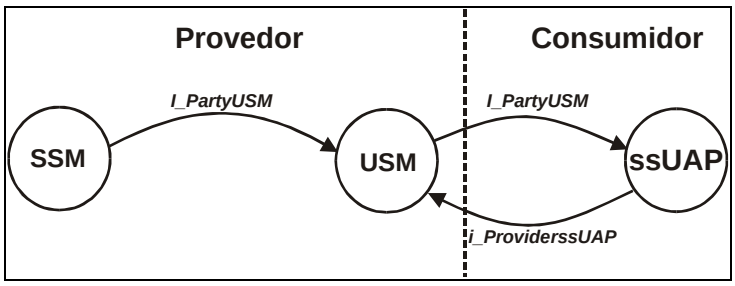


Figura 16 – Interfaces *i_PartyUSM e i_ProviderssUAP*.

a. USM ←→ ssUAP

- Estas interfaces permitem as seguintes operações:

- finalizar uma sessão de serviço;
- finalizar a sessão de serviço via a sessão de acesso;
- suspender uma sessão de serviço;
- suspender a participação numa sessão de serviço; e
- convidar um usuário a unir-se à sessão.

b. SSM → USM

- Esta interface permite todos os usos.
- Esta interface também é usada pelo SSM para informar das mudanças da sessão, outros comandos e informações.

14. ***i_ProviderSSM()***:

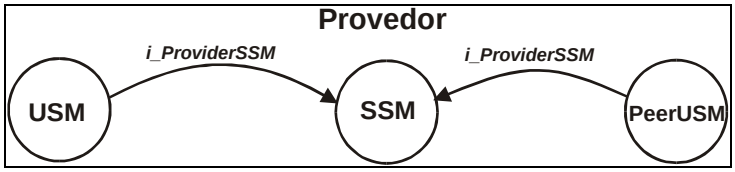


Figura 17 – Interface *i_ProviderSSM*:
USM → SSM, PeerUSM → SSM.

- Esta interface permite diagnosticar e monitorar os vários eventos relacionados com o uso.

15. ***i_SessionCtrl()***:

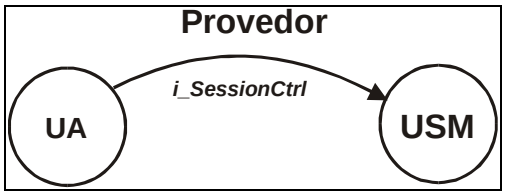


Figura 18 – Interface *i_SessionCtrl*:
UA → USM.

- Esta interface permite a UA dar comandos de controle de sessão que têm sido solicitados através de uma sessão de serviço.
- Esta interface provê as seguintes operações:
  - permite que toda a sessão seja terminada pelo UA no domínio do *Retailer*.;

- permite ao UA terminar a participação dos participantes representados pelo USM;
- permite ao UA suspender toda a sessão através do USM dos participantes representados pelo UA; e
- permite ao UA suspender toda a participação na sessão dos participantes representados pelo UA.

16. ***i_SessionInfo()***:

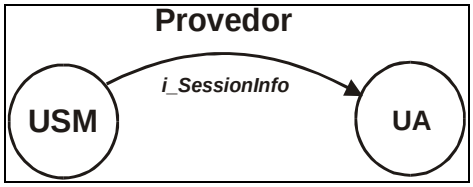


Figura 19 – Interface *i_SessionInfo*:
USM → UA.

- Esta interface permite ao cliente dar ao UA as informações necessárias para manter uma lista atualizada de sessões correspondentes à participação do usuário e ao estado da sessão e da participação do usuário.
- Esta interface provê as seguintes operações:
  - suspender a participação na sessão de serviço;
  - finalizar a sessão de serviço via a sessão de acesso;
  - suspender uma sessão de serviço; e
  - finalizar uma sessão de serviço.

17. ***i_InvitationDelivery()***:

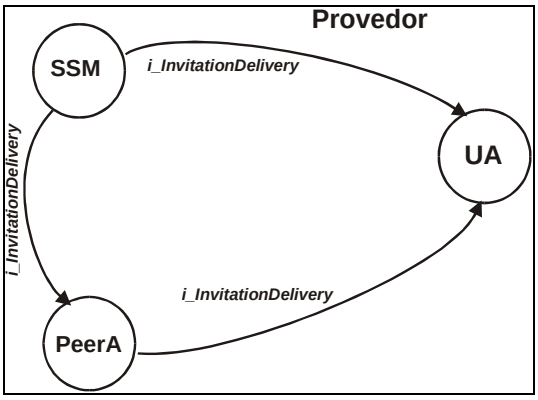


Figura 20 – Interface *InvitationDelivery*:
SSM → UA, SSM → PeerA, PeerA → UA.

- Esta interface permite ao cliente enviar solicitações para o usuário UA ou para cancelar eles.
- Esta interface provê as seguintes operações:
  - convida ao usuário a unir-se à sessão; e
  - cancela um convite previamente emitido por razões de término da sessão de serviço;

18. ***i_SubscriptionNotify()***:

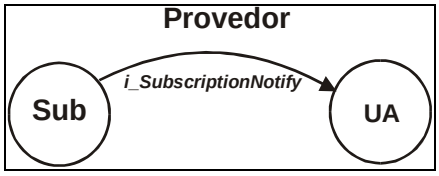


Figura 21 – Interface *i_SubscriptionNotify*:
Sub → UA.

- Esta interface permite ao cliente notificar ao UA sobre serviços novos, modificados ou removidos na pasta do UA correspondente ao usuário.
- Esta interface provê as seguintes operações:
  - subscrever um novo cliente;
  - modificar um novo serviço;
  - modificar o contrato do serviço; e
  - cancelar a subscrição.

19. ***i_Resume()***:

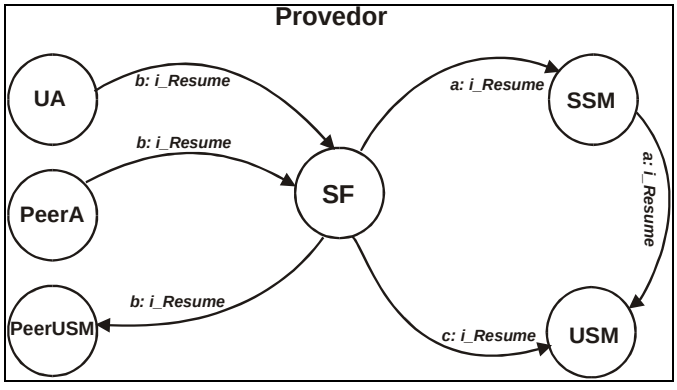


Figura 22 – Interface *i_Resume*:
a) SSM → USM, SF → USM;
b) UA → SF, PeerA → SF, SF → PeerUSM; e
c) SF → SSM.

a. SSM → USM, SF → USM

- Esta interface dá uma referência nova quando a sessão é suspensa ou a participação representada por USM é suspensa.

b. UA → SF, PeerA → SF, SF → PeerUSM

- Esta interface permite suportar as operações de retomada de um serviço e a participação de um usuário em uma sessão de serviço.

c. SF→ SSM

- Esta interface é criada sobre a suspensão de uma sessão de serviço e a referência da interface é armazenada em SF.
- As operações são:
  - inicia uma sessão de serviço e seleciona o modelo da sessão;
  - recupera uma sessão serviço; e
  - recupera a participação de uma sessão de serviço.

20. ***i_Init()***:

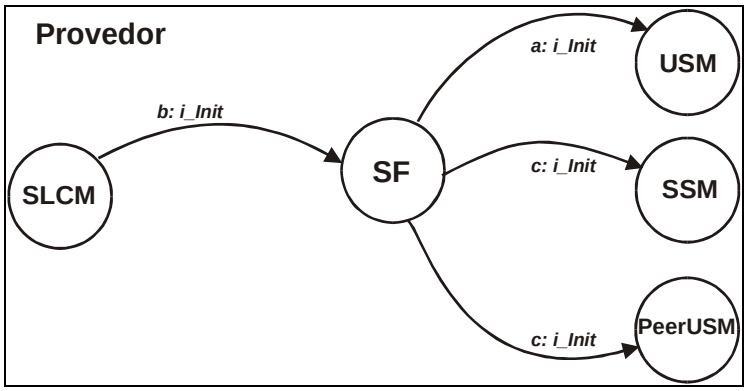


Figura 23 – Interface *i_Init*:
a) SF → USM;
b) SLCM → SF, SF → USM; e
c) SF → SSM, SF → PeerUSM.

a. SF → USM

- Esta interface é usada por SF para inicializar o serviço com propriedades apropriadas quando o componente foi novamente instanciado para uma nova sessão. A operação é usada uma vez e dá acesso a outras interfaces sobre o componente USM.

- Este provê as seguintes operações:
  - inicia uma sessão de serviço e seleciona o modelo da sessão;
  - retoma uma sessão de serviço;
  - retoma a participação numa sessão de serviço; e
  - une-se à sessão de serviço com convite.

b. SLCM → SF

- Esta interface permite ao SLCM inicializar, configurar e gerenciar o SF.
- Este provê as seguintes operações:
  - registrar um novo serviço;
  - retirar um serviço; e
  - modificar um serviço existente.

c. SF→ SSM, SF→ PeerUSM

- Esta interface permite as seguintes operações:
  - inicia uma sessão de serviço e seleciona o modelo da sessão;
  - retoma uma sessão serviço; e
  - retoma a participação numa sessão de serviço.

21. ***i_Join()***:

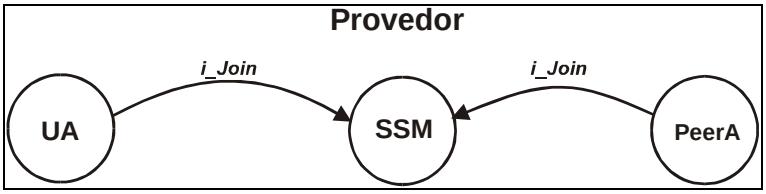


Figura 24 – Interface *i_Join*:

UA → SSM, PeerA → SSM.

- Esta interface permite ao UA (ou PeerA) enviar solicitações desde um consumidor para unir-se a uma sessão de serviço com convite ou notificação, este também suporta réplicas de convites.
- Esta interface realiza as seguintes operações:
  - une-se a uma sessão de serviço com convite.

22. ***i_SSCreate()***:

- Esta interface suporta as operações para criar os objetos de uma sessão de serviço (USM, SSM e PeerUSM).

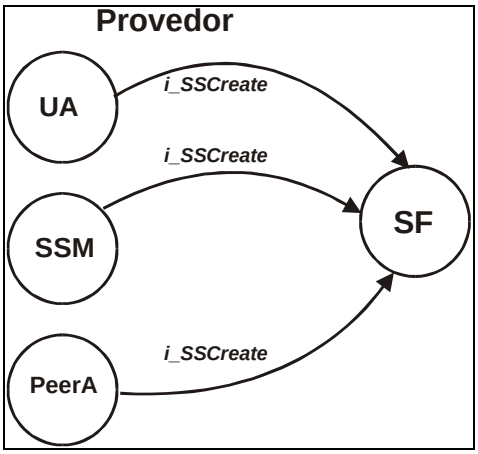


Figura 25 – Interface *i_SSCreate*:

UA → SF, SSM → SF, PeerA → SF.

- Esta interface realiza as seguintes operações:
  - permite ao UA ou PeerA solicitar, a criação de uma nova sessão de serviço. Este resulta na criação do USM e SSM (ou PeerUSM);
  - permite solicitar a inclusão de um novo usuário para um serviço existente. Este resulta na criação do USM; e
  - permite ao SSM solicitar a criação de PeerUSM para suportar uma sessão federada.

23. ***i_SSManage()***:

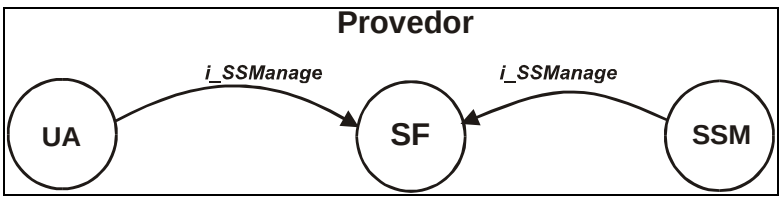


Figura 26 – Interface *i_SSManage*:

UA → SF, SSM → SF.

- Esta interface suporta as operações de gerenciamento (terminar, suspender e obter informações) da sessão de serviço e objetos relacionados (USM, SSM e PeerUSM).
- Uma sessão de serviço é identificada unicamente por um identificador dentro do SF.
- Cada USM gerenciado por um SF é identificado pelo par: *SessionId* e *UserSessionId.*
- Esta interface realiza as seguintes operações:

- suspender uma sessão de serviço;
- terminar a sessão de serviço via a sessão de acesso; e
- terminar uma sessão de serviço.

24. ***i_SSEvents()***:

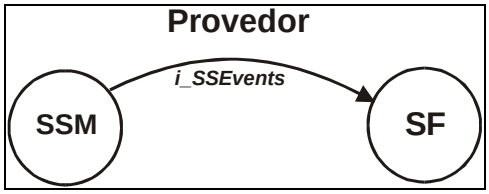


Figura 27 – Interface *i_SSEvents*:

SSM → SF.

- Esta interface é usada pelo SSM, USM e PeerUSM para enviar notificações para o SF sobre mudanças que ocorrem na sessão de serviço.

25. ***i_InitialAccess()***:

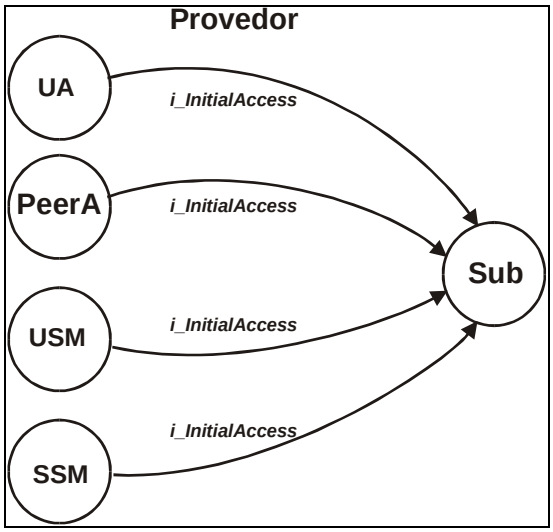


Figura 28 – Interface *i_InitialAccess*:

UA → Sub, PeerA → Sub, USM → Sub, SSM → Sub.

- Esta interface permite a um cliente solicitar uma interface para a funcionalidade do gerenciamento de subscrição.
- No caso do cliente é um componente de acesso, este retorna uma referência à interface *i_SubscriberInfoQuery*, e no caso de SSM, uma referência à interface *i_Subscribe*.

- Uma operação terminar é fornecida para liberar os recursos que foram alocados na operação *i_init*.

26. ***i_SubscriberInfoQuery()***:

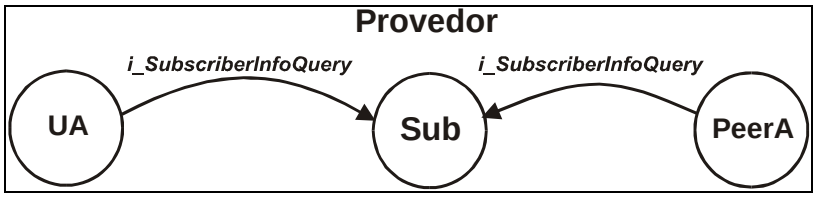


Figura 29 – Interface *i_SubscriberInfoQuery*:
UA → Sub, PeerA → Sub.

- Esta interface é oferecida para os componentes de acesso UA e PeerA para permitir-lhes recuperar informações da subscrição associada a um usuário particular.
- Esta interface realiza as seguintes operações:
  - listar serviços subscritos (assinados).

27. ***i_Subscribe()***:

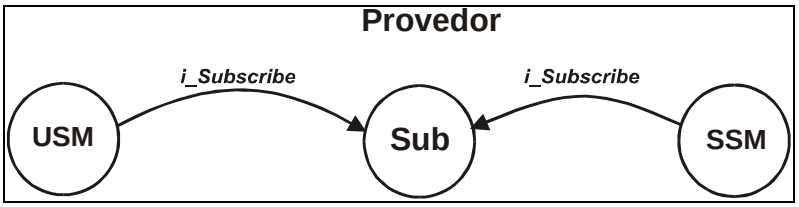


Figura 30 – Interface *i_Subscribe*:
USM → Sub, SSM → Sub.

- Esta interface permite criar um contrato de subscrição para o assinante.
- Esta interface realiza as seguintes operações:
  - subscreve um novo consumidor;
  - modifica informações do subscritor;
  - cancela a subscrição;
  - contrata um novo serviço; e
  - modifica o contrato do serviço.

28. ***i_ServiceContractInfoMgmt()***:

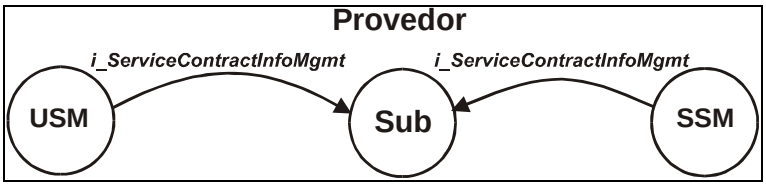


Figura 31 – Interface *i_ServiceContractInfoMgmt*:
USM → Sub, SSM → Sub.

- Esta interface provê funções para definir e modificar um contrato de serviço.
- Este permite a modificação e consulta às informações de um contrato de serviço.
- Esta interface realiza as seguintes operações:
  - subscrever um novo consumidor;
  - contratar um novo serviço;
  - modificar o contrato do serviço; e
  - cancelar a subscrição;

29. ***i_SubscriberInfoMgmt()***:

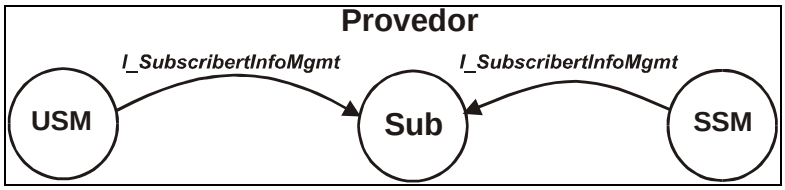


Figura 32 – Interface *i_SubscriberInfoMgmt*:
USM → Sub, SSM → Sub.

- Esta interface permite incluir, modificar e deletar informações relacionadas aos subscritores como entidades associadas (usuários, terminais ou NAPs – *Network Access Point*), grupos de atribuição de subscrição e associações de serviço para o SAG – *Subscriber Assigment Entity*.
- Desta interface o cliente pode acessar só informações de um subscritor particular.
- As principais operações são:
  - subscreve um novo consumidor;
  - modifica informações do subscritor;

- cancela a subscritor;
- contrata um novo serviço; e
- modifica o contrato do serviço.

30. ***i_ServiceNotify()***:

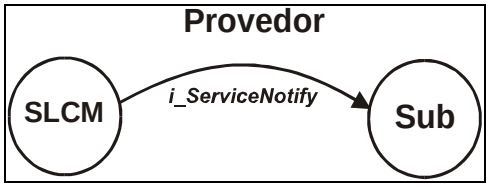


Figura 33 – Interface *i_ServiceNotify*:

SLCM → Sub.

- Esta interface permite ao SLCM notificar ao SUB sobre novos serviços a serem usados e disponíveis para a subscrição e uso, ou sobre a modificação ou remoção de um serviço existente.
- As operações principais são:
  - registrar um novo serviço;
  - modificar um serviço existente; e
  - retirar um serviço.

31. ***i_ServiceQuery()***:

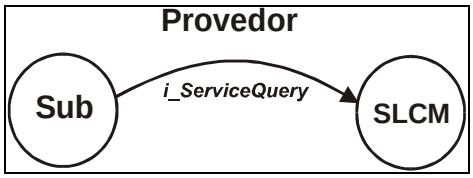


Figura 34 – Interface *i_ServiceQuery*:

Sub → SLCM.

- Esta interface provê de informações necessárias do serviço de subscrição do Sub.

31. ***i_SetUp(), i_WrapUp() e i_Execution()***:

- A interface *i_SetUp* permite aos participantes (usuário e provedor) apresentar seus esquemas de contabilidade para uma possível negociação para criar seu *AccMgmtCtxt*.
- A interface *i_Execution* permite que um serviço seja oferecido ao usuário de acordo com sua Transação de Serviço e as informações contábeis são ativadas.

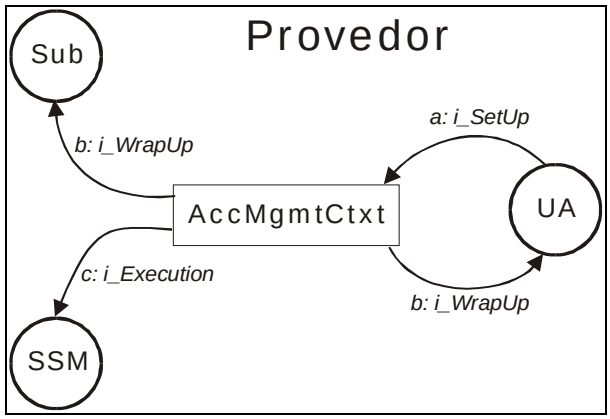


Figura 35 – Interfaces *i_SetUp, i_WrapUp e i_Execution:*

a) UA → *AccMgmtCtxt*;

b) *AccMgmtCtxt* → UA, *AccMgmtCtxt* → Sub; e

c) *AccMgmtCtxt* → SSM.

- A interface *i_WrapUp* permite terminar o serviço oferecido e as medições contábeis são coletadas e sumarizadas.
- Estas interfaces foram descritas sucintamente no item 3.4 que definem os conceitos da transação de serviço em um contexto de gerenciamento de contabilidade TINA.

32. ***i_RoleCertificate()*** e ***i_RoleCertificateReq()***:

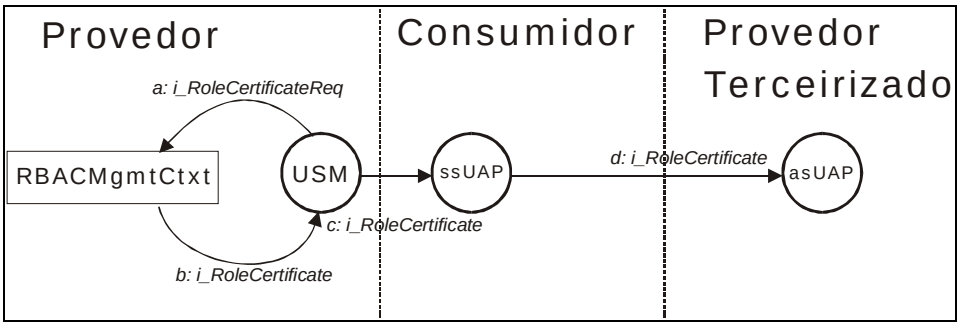


Figura. 36 – Interfaces *i_RoleCertificate* e *i_RoleCertificateReq*:

a) USM → *RBACMgmtCtxt*;

b) *AccMgmtCtxt* → USM;

c) USM → ssUAP, e

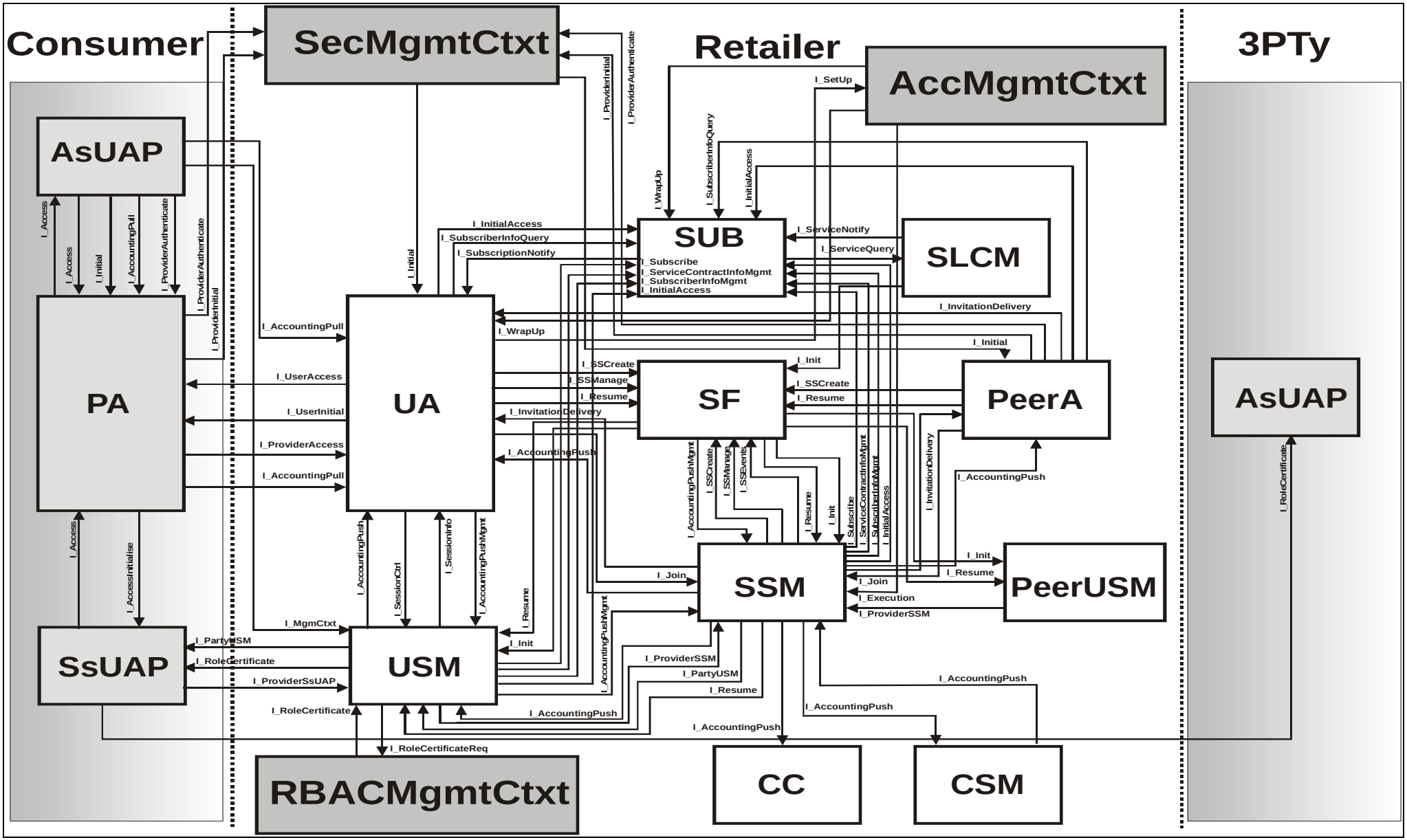
d) ssUAP → asUAP.

- A interface *i_RoleCertificateReq* permite solicitar ao contexto de gerenciamento RBAC o certificado de papel que é emitido ao USM através do Agente de Controle de Acesso (ACA).

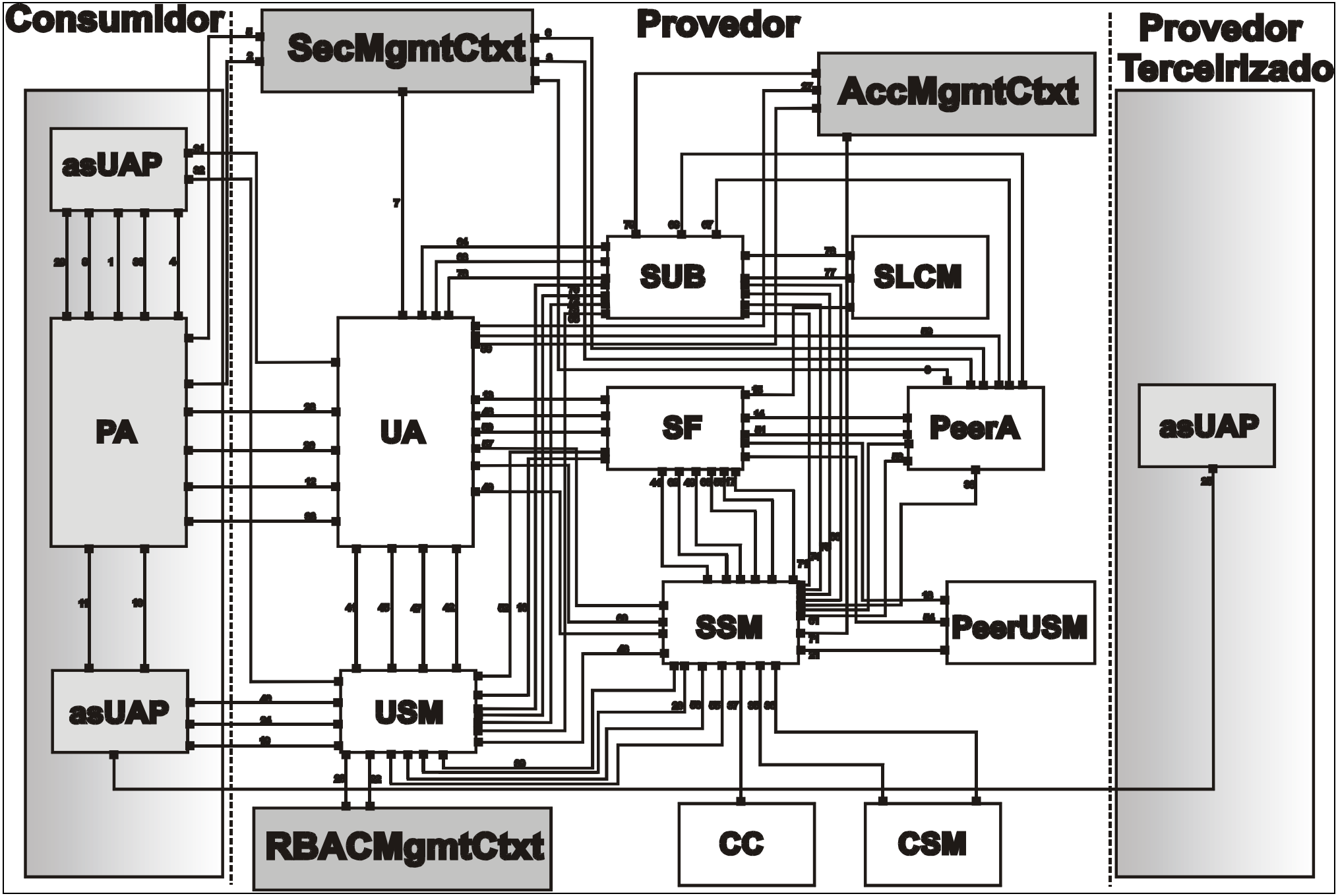
- A interface *i_RoleCertificate* permite entregar o certificado ao usuário final para ativar a interface de controle que permite ao servidor fornecer o serviço requerido pelo usuário usando o certificado.
- Estas interfaces foram descritas sucintamente no item 5.2 que definem as políticas do modelo RBAC para uma sessão de serviço TINA.

ANEXO 2 – AMPLIAÇÃO DAS FIGURAS DA ARQUITETURA SSGCT AMPLIAÇÂO

1.1 Ampliação correspondente a Figura 6.2



1.2 Ampliação correspondente a Figura 7.14



ANEXO 3 – CORBA/JAVA

3.1 Introdução

Este documento apresenta o código fonte do protótipo implementado em JAVA e em um ambiente distribuído através do padrão CORBA.

```
// {$R client.JFM}
import java.awt.*;
import java.lang.*;
import java.util.*;
// Class client
public class client extends Frame
{
   final int MenuBarHeight = 0;
   String args1[];
   int tipo_servico = 1; // 1 = escuta
                    // 2 = escuta,fala
                    // 3 = escuta,fala/compartilhado
   boolean start = false; // false= stop, true = start
   boolean enter = false; // false= leave, true = enter

   String csid ="0";
   String name = "";
   String data_inicio = "";
   int    seg_total   = 0;
   int    valor_total = 0; //
   String hora_inicio = "";
   String hora_final  = "";
   int    seg_tipo1   = 0; //escuta
   int    seg_tipo2   = 0; //escuta,fala
   int    seg_tipo3   = 0; //escuta,fala/compartilhado

  // int valor1 = 0;//5; //escuta
   int price1 = 1;
   int price2 = 3;//15;//escuta,fala
   int price3 = 2;//10;//escuta,fala/compartilhado

   //AccCtxt Ctxt = new AccCtxt(csid);
   org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args1,null);
     AccMgmtCtxt.EventManager                                    eventmanager=
AccMgmtCtxt.EventManagerHelper.bind(orb,"EventManager_AccMgmt");
      AccMgmtCtxt.Billing                                              billing=
AccMgmtCtxt.BillingHelper.bind(orb,"Billing_AccMgmt");

      //AccMgmtCtxt.Tariff                                               tariff=
AccMgmtCtxt.TariffHelper.bind(orb,"Tariff_AccMgmt");
```

```
 //  AccMgmtCtxt.Tarval rifa = new AccMgmtCtxt.Tarval() ;
//   rifa = tariff.Tariff_val();

     //valor1 = rifa.type1;
     //price1 = 5; //tarifa.type1;
     //price2 = 15; //tarifa.type2;
     //price3 = 10; //tarifa.type3;

 //  System.out.println(tarifa.type1); //+" "+price2+" "+price3);

//AccCtxt Ctxt = new AccCtxt(csid);

   Thread Ctxt;
   // Component Declaration
   public Panel Panel1;
   public Panel Panel2;
   public Panel Panel3;
   public Label Label1;
   public Label Label2;
   public Label Label3;
   public Label Label4;
   public Label Label5;
   public Label Label6;
   public TextField CustIdTf;
   public TextField NameTf;
   public TextField SubsTypeTf;
   public TextField ServTypeTf;
   public TextField MaxAmTf;
   public TextField TotalAcTf;
   public Button AddUserBt;
   public Button ChgMaxBt;
   public Button StartBt;
   public Button StopBt;
   public Button EnterBt;
   public Button LeaveBt;
   public TextField PwdTf;
   public Label Label10;
   public Label Label7;
   public Label Label8;
   public Panel Panel4;
   public TextField StatusTf;
   public Label Label9;
   public Label Label11;
   public Panel Panel5;
   public TextArea MsgTa;
   public Label Label12;
   // End of Component Declaration

   // Constructor
   public client()
```

```
{
    // Frame Initialization
    setForeground(Color.black);
    setBackground(Color.gray);
    setFont(new Font("Dialog",Font.BOLD,12));
    setTitle("Client");
    setLayout(null);
    // End of Frame Initialization

    // Component Initialization
    Panel1 = new Panel();
    Panel1.setLayout(null);
    Panel1.setForeground(Color.black);
    Panel1.setBackground(Color.lightGray);
    Panel1.setFont(new Font("Dialog",Font.BOLD,12));
    Panel2 = new Panel();
    Panel2.setLayout(null);
    Panel2.setForeground(Color.black);
    Panel2.setBackground(Color.lightGray);
    Panel2.setFont(new Font("Dialog",Font.BOLD,12));
    Panel3 = new Panel();
    Panel3.setLayout(null);
    Panel3.setForeground(Color.black);
    Panel3.setBackground(Color.lightGray);
    Panel3.setFont(new Font("Dialog",Font.BOLD,12));
    Label1 = new Label("Cust. Id",Label.LEFT);
    Label1.setFont(new Font("Helvetica",Font.PLAIN,12));
    Label2 = new Label("Name",Label.LEFT);
    Label2.setFont(new Font("Helvetica",Font.PLAIN,12));
    Label3 = new Label("Subs. Type",Label.LEFT);
    Label3.setFont(new Font("Helvetica",Font.PLAIN,12));
    Label4 = new Label("Serv. Type",Label.LEFT);
    Label4.setFont(new Font("Helvetica",Font.PLAIN,12));
    Label5 = new Label("Max Amount",Label.LEFT);
    Label5.setFont(new Font("Helvetica",Font.PLAIN,12));
    Label6 = new Label("Total Acum",Label.LEFT);
    Label6.setFont(new Font("Helvetica",Font.PLAIN,12));
    CustIdTf = new TextField("");
    CustIdTf.setForeground(Color.black);
    CustIdTf.setBackground(Color.white);
    CustIdTf.setFont(new Font("Dialog",Font.PLAIN,12));
    NameTf = new TextField("");
    NameTf.setForeground(Color.black);
    NameTf.setBackground(Color.white);
    NameTf.setFont(new Font("Dialog",Font.PLAIN,12));
    SubsTypeTf = new TextField("");
    SubsTypeTf.setForeground(Color.black);
    SubsTypeTf.setBackground(Color.white);
    SubsTypeTf.setFont(new Font("Dialog",Font.PLAIN,12));
    ServTypeTf = new TextField("");
```

```
        ServTypeTf.setForeground(Color.black);
        ServTypeTf.setBackground(Color.white);
        ServTypeTf.setFont(new Font("Dialog",Font.PLAIN,12));
        MaxAmTf = new TextField("");
        MaxAmTf.setForeground(Color.black);
        MaxAmTf.setBackground(Color.white);
        MaxAmTf.setFont(new Font("Dialog",Font.PLAIN,12));
        TotalAcTf = new TextField("");
        TotalAcTf.setForeground(Color.black);
        TotalAcTf.setBackground(Color.white);
        TotalAcTf.setFont(new Font("Dialog",Font.PLAIN,12));
        AddUserBt = new Button("Add New User");
        AddUserBt.setFont(new Font("Dialog",Font.PLAIN,12));
        ChgMaxBt = new Button("Change Max Amount");
        ChgMaxBt.setFont(new Font("Dialog",Font.PLAIN,12));
        StartBt = new Button("Start");
        StartBt.setFont(new Font("Dialog",Font.PLAIN,12));
        StopBt = new Button("Stop");
        StopBt.setFont(new Font("Dialog",Font.PLAIN,12));
        EnterBt = new Button("Enter");
        EnterBt.setFont(new Font("Dialog",Font.PLAIN,12));
        LeaveBt = new Button("Leave");
        LeaveBt.setFont(new Font("Dialog",Font.PLAIN,12));
        NameTf.enable(false);
        SubsTypeTf.enable(false);
        ServTypeTf.enable(false);
        MaxAmTf.enable(false);
        TotalAcTf.enable(false);
        PwdTf = new TextField("");
        PwdTf.setForeground(Color.black);
        PwdTf.setBackground(Color.white);
        PwdTf.setFont(new Font("Dialog",Font.PLAIN,12));
        Label10 = new Label("Password",Label.LEFT);
        Label10.setFont(new Font("Helvetica",Font.PLAIN,12));
        PwdTf.setEchoCharacter('*');
        Label7 = new Label("Update:",Label.LEFT);
        Label7.setFont(new Font("Dialog",Font.BOLD,12));
        Label8 = new Label("Service:",Label.LEFT);
        Label8.setFont(new Font("Dialog",Font.BOLD,12));
        Panel4 = new Panel();
        Panel4.setLayout(null);
        Panel4.setForeground(Color.black);
        Panel4.setBackground(Color.lightGray);
        Panel4.setFont(new Font("Dialog",Font.BOLD,12));
        StatusTf = new TextField("");
        StatusTf.setForeground(Color.black);
        StatusTf.setBackground(Color.white);
        StatusTf.setFont(new Font("TimesRoman",Font.BOLD,12));
        Label9 = new Label("Status:    A=Audio    V=Audio+Video    S= Audio+Video
Shared",Label.LEFT);
```

```
Label9.setFont(new Font("Dialog",Font.BOLD,12));
Label11 = new Label("Subscription:",Label.LEFT);
Label11.setFont(new Font("Dialog",Font.BOLD,12));
Panel5 = new Panel();
Panel5.setLayout(null);
Panel5.setForeground(Color.black);
Panel5.setBackground(Color.lightGray);
Panel5.setFont(new Font("Dialog",Font.BOLD,12));
MsgTa = new TextArea("");
MsgTa.setForeground(Color.black);
MsgTa.setBackground(Color.white);
MsgTa.setFont(new Font("Dialog",Font.PLAIN,12));
Label12 = new Label("Error Messages:",Label.LEFT);
Label12.setFont(new Font("Dialog",Font.BOLD,12));
// End of Component Initialization

// Add()s
Panel5.add(Label12);
Panel5.add(MsgTa);
add(Panel5);
Panel3.add(Label11);
Panel4.add(Label9);
Panel4.add(StatusTf);
add(Panel4);
Panel2.add(Label8);
Panel1.add(Label7);
Panel3.add(Label10);
Panel3.add(PwdTf);
Panel2.add(LeaveBt);
Panel2.add(EnterBt);
Panel2.add(StopBt);
Panel2.add(StartBt);
Panel1.add(ChgMaxBt);
Panel1.add(AddUserBt);
Panel3.add(TotalAcTf);
Panel3.add(MaxAmTf);
Panel3.add(ServTypeTf);
Panel3.add(SubsTypeTf);
Panel3.add(NameTf);
Panel3.add(CustIdTf);
Panel3.add(Label6);
Panel3.add(Label5);
Panel3.add(Label4);
Panel3.add(Label3);
Panel3.add(Label2);
Panel3.add(Label1);
add(Panel3);
add(Panel2);
add(Panel1);
// End of Add()s
```

```
        InitialPositionSet();
    }

    public void InitialPositionSet()
    {
        // InitialPositionSet()
        reshape(196,109,581,439);
        Panel1.reshape(348,334+MenuBarHeight,227,98);
        Panel2.reshape(6,334+MenuBarHeight,340,98);
        Panel3.reshape(287,25+MenuBarHeight,288,240);
        Label1.reshape(2,47,51,19);
        Label2.reshape(4,105,51,19);
        Label3.reshape(2,133,82,19);
        Label4.reshape(2,161,76,19);
        Label5.reshape(3,186,99,19);
        Label6.reshape(4,214,80,19);
        CustIdTf.reshape(91,39,121,27);
        NameTf.reshape(90,95,191,27);
        SubsTypeTf.reshape(90,123,190,27);
        ServTypeTf.reshape(90,150,190,27);
        MaxAmTf.reshape(90,178,121,27);
        TotalAcTf.reshape(90,207,121,27);
        AddUserBt.reshape(6,35,169,25);
        ChgMaxBt.reshape(7,67,168,25);
        StartBt.reshape(30,33,125,25);
        StopBt.reshape(175,33,127,25);
        EnterBt.reshape(31,63,123,25);
        LeaveBt.reshape(175,63,127,25);
        PwdTf.reshape(91,67,121,27);
        Label10.reshape(3,75,73,19);
        Label7.reshape(5,7,63,19);
        Label8.reshape(12,6,76,19);
        Panel4.reshape(6,267+MenuBarHeight,569,65);
        StatusTf.reshape(7,31,554,27);
        Label9.reshape(8,7,454,19);
        Label11.reshape(6,5,107,19);
        Panel5.reshape(6,25+MenuBarHeight,278,240);
        MsgTa.reshape(2,68,271,115);
        Label12.reshape(3,5,126,19);
        // End of InitialPositionSet()
    }

    public boolean handleEvent(Event evt)
    {
        // handleEvent()
        if (evt.id == Event.WINDOW_DESTROY && evt.target == this)
client_WindowDestroy(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == MaxAmTf)
MaxAmTf_Action(evt.target);
```

```
        else if (evt.id == Event.ACTION_EVENT && evt.target == AddUserBt)
AddUserBt_Action(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == ChgMaxBt)
ChgMaxBt_Action(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == StartBt)
StartBt_Action(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == StopBt)
StopBt_Action(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == EnterBt)
EnterBt_Action(evt.target);
        else if (evt.id == Event.ACTION_EVENT && evt.target == LeaveBt)
LeaveBt_Action(evt.target);
        // End of handleEvent()

        return super.handleEvent(evt);
      }
   /*=============================================================
=======================*/
      public void paint(Graphics g)
      {
        // paint()
        // End of paint()
      }
   /*=============================================================
=======================*/

     // main()
     public static void main(String args[])
     {
       client client = new client();
       client.show();
     } // End of main()
   /*=============================================================
=======================*/

      // Event Handling Routines
      public void client_WindowDestroy(Object target)
      {
        System.exit(0);
      }
   /*=============================================================
=======================*/

      public void AddUserBt_Action(Object target)
      {
       //   System.exit(0);
       //          public static void main(String args[])
       //{
        AddUser AddUser = new AddUser();
        AddUser.show();
```

```
        //  } // End of main()

        //

      }
   /*=================================================================
=======================*/

      public void MaxAmTf_Action(Object target)
      {
      ////
      }
   /*=================================================================
=======================*/

      public void ChgMaxBt_Action(Object target)
      {
       ///
      }
   /*=================================================================
=======================*/

     //  public void BlockUsrBt_Action(Object target)
     //  {
     //  /////
     //  }
   /*=================================================================
=======================*/

      public void StartBt_Action(Object target)
      {
       ////
       start= true;
      boolean lockstatus = true;  //locked= true, unlocked= false;

       AccMgmtCtxt.SubscInfo vai = new AccMgmtCtxt.SubscInfo() ;
       AccMgmtCtxt.SubscInfo volta = new AccMgmtCtxt.SubscInfo() ;


       vai.custid = CustIdTf.getText();
       vai.pwd = PwdTf.getText();
       vai.name = " ";
       vai.subtype = " ";
       vai.servtype = " ";
       vai.maxamnt = 0;
       vai.totalacm = 0;

         //colocar um try
```

```
        AccMgmtCtxt.SessionComponent sessioncomponent=
AccMgmtCtxt.SessionComponentHelper.bind(orb,"SessionComponent_AccMgmt");
        volta = sessioncomponent.SubsValid(vai);

       if (Integer.parseInt(vai.custid) == Integer.parseInt(volta.custid))
        {
                if (Integer.parseInt(vai.pwd) == Integer.parseInt(volta.pwd))
                {
          NameTf.setText(volta.name);
          name = volta.name;
          PwdTf.setText(volta.pwd);
          SubsTypeTf.setText(volta.subtype);
          ServTypeTf.setText(volta.servtype);
          MaxAmTf.setText("$"+volta.maxamnt);
          TotalAcTf.setText("$"+volta.totalacm);
          ///////////////
          csid= volta.custid;
         //AccCtxt Ctxt = new AccCtxt(csid);
           AccMgmtCtxt.Recovery recovery=
AccMgmtCtxt.RecoveryHelper.bind(orb,"Recovery_AccMgmt");
           lockstatus = recovery.CheckStatus(csid);
           if (lockstatus)
           {
            MsgTa.appendText("Previous session closed with success"+"\n");
            }
           else
           {
            MsgTa.appendText("Previous session closed abnormally. Display previous billing
info now. "+"\n");
            Bill Bill = new Bill(csid);
            Bill.show();
            }
          Ctxt = new AccCtxt(csid);
          Ctxt.start(); //volta.custid);

          ////////////////////
                }
                else
                 {
                        MsgTa.appendText("Invalid Password!!"+"\n");
                  }

        }
        else
        {
                 MsgTa.appendText("Inexistent Id User. Try Add User."+"\n");
                  AddUser AddUser = new AddUser();
           AddUser.show();

        }
```

```
        }
    /*=============================================================
=======================*/

        public void StopBt_Action(Object target)
        {
         start = false;
         Ctxt.suspend();
         seg_total = seg_tipo1+seg_tipo2+seg_tipo3;
         hora_final = hora();
         // provisorio Ctxt.suspend();
         System.out.println(data_inicio);

         this.trigger();

         System.out.println(hora_inicio);
    System.out.println(hora_final);
    System.out.println(seg_total);
    System.out.println(valor_total);
    seg_tipo1   = 0; //escuta
    seg_tipo2   = 0; //escuta,fala
    seg_tipo3   = 0; //escuta,fala/compartilhado
    seg_total   = 0;
    valor_total = 0; //

          Bill Bill = new Bill(csid);
          Bill.show();

        ////

        }
    /*=============================================================
=======================*/

        public void EnterBt_Action(Object target)
        {
         enter = true;
         tipo_servico = 2;
         tipo_servico = eventmanager.EnterEvent(csid,tipo_servico);
         ////
        }
    /*=============================================================
=======================*/

        public void LeaveBt_Action(Object target)
        {
         enter = false;
         tipo_servico = 1;
         tipo_servico = eventmanager.LeaveEvent(csid, tipo_servico);
         System.out.println("Leave"+tipo_servico);
```

```
        /////
       }
    /*=============================================================
=======================*/
       class AccCtxt extends Thread
       {
         String cid;
         public AccCtxt(String custid)
          {
           cid = csid; //custid;//inic = inicio;
          }

          public void run()
          {
                System.out.println("thread");
                boolean reg_success = false;
                String txt = "";
           String  servtype = "-";
          //       AccMgmtCtxt.EventManager                                    eventmanager=
AccMgmtCtxt.EventManagerHelper.bind(orb,"EventManager_AccMgmt");

            //registra tipo_servico
            reg_success = eventmanager.RegService(cid,tipo_servico);
                 int num =0;
                 String second = sec();
            String anterior = second;
            String houra = hora();
            //String dauta = data();
            data_inicio = data();
            hora_inicio = hora();
            while (true)
            {
                 if (second.equals(anterior))
                  {
                          second = sec();

                          try
                           {
                                  sleep(300);
                            }
                           catch(Exception e)
                       {
                           System.err.println(e);
                           }
                             }
                            else
                            {
                                  txt = StatusTf.getText();
                                  if (txt.length() > 60)
                                  {
```

```
                        StatusTf.setText("");
                    }
                    else
                    {
                        tipo_servico = eventmanager.GetTipoServico(cid,tipo_servico);
                        if (tipo_servico == 1)
                         {
                            servtype = "A";
                            valor_total= valor_total+price1;
                            seg_tipo1++;
                          }
                        if (tipo_servico == 2)
                         {
                            servtype = "V";
                            valor_total= valor_total+price2;
                            seg_tipo2++;

                          }
                        if (tipo_servico == 3)
                        {
                            servtype = "S";
                            valor_total= valor_total+price3;
                            seg_tipo3++;

                        }


                    StatusTf.setText(StatusTf.getText()+servtype);
                    anterior=second;

                        }
                }

            }

         }
        }

/*=================================================================
====================*/
      public String hora()
      {
         Date d = new Date();
         String hour = "";
         String min  = "";
         String secon= "";
         hour = formatfor2(d.getHours());
         min  = formatfor2(d.getMinutes());
         secon= formatfor2(d.getSeconds());
         String hora_junto = hour+":"+min+":"+secon;
```

```
        return hora_junto;
       }

/*=================================================================
====================*/

     public String formatfor2(int n)
     {
        String ret;
        if (n < 10 )
                 ret = new String ("0"+n);
                else
                 ret = new String(""+n);
                 return ret;
     }

/*=================================================================
====================*/
    public String data()
    {
        Date d = new Date();
        int dia = d.getDate();
        int mes = d.getMonth();
        int ano = 2002;//d.getYear();

        String ret = dia+"/"+(mes+1)+"/"+ano;
        return ret;
    }
    /*=================================================================
=======================*/

      public String sec()
          {

            long time= System.currentTimeMillis();
            String teste = Long.toString(time);
            String sub =teste.substring(8,10);
            //System.out.println(sub);
            //int segund= Integer.parseInt(sub);
            //return segund;
            return sub;
          }
    /*=================================================================
=======================*/

      public void  trigger()
          {
```

```
billing.BillingSave(csid,name,data_inicio,seg_total,valor_total,hora_inicio,hora_final,seg_tip
o1,seg_tipo2,seg_tipo3);

        }



      // End of Event Handling Routines

    } // End of Class client

    ================================================================

    //Server.java

    public class Server
    {
        public static void main(String[] args)
        {
                //Inicializa o ORB
                org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

                //Inicializa o BOA
                org.omg.CORBA.BOA                                                boa
=((com.visigenic.vbroker.orb.ORB)orb).BOA_init();// orb.BOA_init();
                ////////////////////////////////////////////////
                //1 Cria um novo objeto sessioncomponent
                SessionComponentImpl sessioncomponent = new
SessionComponentImpl("SessionComponent_AccMgmt");
                //Exporta o novo objeto criado
                boa.obj_is_ready(sessioncomponent);
                System.out.println(sessioncomponent + "está pronto.");

                ////////////////////////////////////////////////
                //2 Cria um novo objeto EventManager
                EventManagerImpl eventmanager = new
EventManagerImpl("EventManager_AccMgmt");//"EventManager_AccMgmt");
                //Exporta o novo objeto criado
                boa.obj_is_ready(eventmanager);
                System.out.println(eventmanager + "está pronto.");

                ////////////////////////////////////////////////
                //3 Cria um novo objeto Tariff
                AccMgmtCtxt.Tariff tariff = new TariffImpl("Tariff_AccMgmt");
                //Exporta o novo objeto criado
                boa.obj_is_ready(tariff);
                System.out.println(tariff + "está pronto.");

                ////////////////////////////////////////////////
```

```
            //4 Cria um novo objeto recovery
            AccMgmtCtxt.Recovery recovery = new
RecoveryImpl("Recovery_AccMgmt");
            //Exporta o novo objeto criado
            boa.obj_is_ready(recovery);
            System.out.println(recovery + "está pronto.");

            ////////////////////////////////////////////////
            //5 Cria um novo objeto billing
            AccMgmtCtxt.Billing billing = new BillingImpl("Billing_AccMgmt");
            //Exporta o novo objeto criado
            boa.obj_is_ready(billing);
            System.out.println(billing + "está pronto.");

            //espera por requisições que irão chegar
            boa.impl_is_ready();
        }

    }
```