UMA PROPOSTA DE ESPECIFICAÇÃO FORMAL EM SDL DE UMA SESSÃO DE ACESSO TINA

Rafael P. Guimarães, Walter C. Borelli
{rguima, borelli} @dt.fee.unicamp.br

DT – FEEC - Unicamp
Caixa Postal 6101
Campinas, SP, Brasil – Cep 13083-970

RESUMO

O trabalho apresentado neste artigo consiste na especificação formal orientada a objetos da Sessão de Acesso em uma arquitetura de serviços TINA (*Telecommunications Information Networking Architecture*) utilizando SDL (*Specification and Description Language*) com o objetivo de conferir um caráter comportamental ausente nas especificações em ODL (*Object Description Language*) descrita nas recomendações TINA. Sendo assim, é proposto neste artigo um novo mapeamento de ODL/IDL para SDL, com o qual é possível modelar os objetos componentes da arquitetura TINA de uma forma mais adequada.

Após a especificação proposta para a Sessão de Acesso TINA, são apresentados exemplos de simulações e sua completa validação.

Palavras-Chave: TINA, SDL, Arquitetura de Serviços

ABSTRACT

The work presented in this paper consists on the object oriented formal specification of the Access Session on a TINA Service Architecture using SDL, with the intention to provide a behaviour description which is not presented in the ODL specifications described on TINA recommendations. Then, it is proposed in this paper a new mapping from ODL/IDL to SDL, so that it is possible to model the TINA objects in a better way.

After the proposed specification for the TINA Access Session, it is presented examples of simulations and its full validation.

Keywords: TINA, SDL, Service Architecture

1. INTRODUÇÃO

O consórcio TINA (*Telecommunications Information Networking Architecture*) define uma arquitetura distribuída aberta para o provimento e gerenciamento de serviços de telecomunicações. A proposta TINA [1] é de disponibilizar um ambiente distribuído bem estruturado onde possa se oferecer desde os serviços mais simples até os mais complexos existentes.

Para simplificar a análise do problema, a arquitetura TINA é separada logicamente em 3 arquiteturas principais (que agrupam diferentes conceitos): a arquitetura de serviços, a arquitetura de rede e a arquitetura de computação.

A arquitetura de serviços [2], que é o foco principal deste artigo, define uma série de conceitos e princípios para a criação, implementação e gerência de serviços de telecomunicações. A arquitetura de rede define uma série de conceitos e princípios para a criação, implementação e gerência de redes de transporte. Finalmente, a arquitetura de computação define uma série de conceitos e princípios para a criação de sistemas distribuídos e de um ambiente de suporte aos sistemas.

Desta forma, uma série de objetos computacionais são especificados nas recomendações TINA afim de fornecer o suporte necessário para a realização desta arquitetura. Estes objetos são descritos através da linguagem ODL [3] (*Object Description Language*), que especifica as interfaces deste objetos, deixando suas características comportamentais em aberto.

Afim de fornecer uma especificação mais completa para estes objetos, conferindo a estas especificações um caráter comportamental e orientado a objetos, este artigo trata da aplicação da linguagem SDL [4] (*Specification and Description Language*) na especificação da Sessão de Acesso de uma Arquitetura de Serviços TINA. A escolha do SDL deve-se principalmente ao fato da mesma se tratar de uma linguagem para a especificação formal de sistemas e protocolos proposta pela ITU já consolidada e bastante utilizada no meio das empresas de telecomunicações, que contam com o suporte de ferramentas profissionais para o desenvolvimento, como o SDT[1] [5] por exemplo, utilizado no trabalho apresentado neste artigo.

Para a elaboração destas especificações em SDL, foi necessário o desenvolvimento de um mapeamento entre as estruturas ODL e as estruturas SDL, baseado em outros discutidos nos artigos [6] e [7], porém com algumas contribuições adicionais.

Este artigo discute os pontos principais da especificação em SDL da Sessão de Acesso TINA, finalizando com a apresentação de sua validação e de alguns exemplos de simulações de suas funcionalidades.

2. A ARQUITETURA DE SERVIÇOS TINA

A arquitetura de serviços define um conjunto de conceitos e princípios que possibilitam a concepção, a implementação, o uso e a operação de serviços de telecomunicações. Para isto, esta arquitetura define componentes reusáveis com os quais se constrói os serviços de telecomunicações.

A arquitetura de serviços TINA faz uso do conceito de sessão, o qual pode ser definido como uma relação temporária entre um grupo de recursos que irão conjuntamente efetuar alguma tarefa

[1] *SDT (SDL Design Tool) v.4.1: adquirido pelo DT/FEEC/UNICAMP através de um projeto temático FAPESP (Proc. 91/3660-0)*

por um período de tempo. São três os tipos principais de sessão: Sessão de Acesso, Sessão de Serviço e Sessão de Comunicação.

A Sessão de Acesso, que é especificada neste artigo, promove o acesso do usuário aos serviços disponibilizados pelo provedor. Para isso a sessão de acesso engloba a autenticação do usuário, troca de informações acerca das capacidades do usuário junto ao provedor, para que finalmente o usuário possa acessar serviços e se unir a sessões de serviço já existentes, quando convidado.

A Sessão de Serviço é responsável por prover um serviço dentre os disponíveis. É composta por uma instanciação do serviço a ser acessado e possui informações necessárias para o uso do mesmo.

Finalmente, a Sessão de Comunicação provê os recursos de comunicação necessários para o estabelecimento de conexões de fluxo de dados (*stream*) entre os participantes de uma sessão de serviço.

Na Figura 1 são apresentados os objetos participantes da Sessão de Acesso TINA e seus relacionamentos.

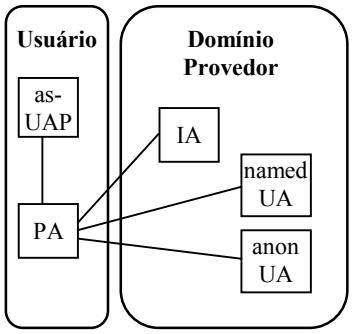

Figura 1. Interação entre componentes da Sessão de Acesso da Arquitetura de Serviços TINA

Os componentes relacionados com a sessão de acesso são independentes do serviço oferecido. O as-UAP (*access session User Application*) é responsável pela interface homem-máquina, provendo ao usuário mecanismos para se comunicar com o provedor. O PA (*Provider Agent*), por sua vez, é o representante do provedor no domínio do usuário, sendo responsável pela intermediação entre o as-UAP e o domínio do provedor. No domínio do provedor, o IA (*Initial Agent*) é o ponto de contanto inicial do usuário com o provedor, para que seja estabelecida uma sessão de acesso. O UA (*User Agent*), por sua vez, é o representante do usuário no domínio do provedor e suporta mecanismo de acesso aos diversos serviços disponibilizados – pode ser do tipo anonUA quando o usuário requisita um acesso anônimo, ou namedUA quando o usuário se identifica fornecendo, por exemplo, um *username* e uma senha. Apesar de um usuário poder ter mais de uma sessão de acesso com o provedor, existe um único UA por usuário no provedor.

3. ESPECIFICAÇÃO EM SDL DA SESSÃO DE ACESSO TINA

Nas recomendações TINA, os objetos apresentados na sessão anterior são descritos utilizando-se a linguagem ODL, que é uma linguagem de descrição das interfaces do objeto, praticamente igual à linguagem IDL (*Interface Description Language*).

Sendo assim, para que este sistema possa ser descrito em SDL, algumas regras de mapeamento da linguagem ODL (ou IDL) para a linguagem SDL foram estabelecidas (Tabela 1).

Uma das maiores vantagens do mapeamento proposto neste artigo sobre o apresentado em [6] é o fato dele permitir a existência de exceções no modelo SDL, o que é de fundamental importância em um sistema distribuído (como é o caso da arquitetura de serviços TINA). Além disso, trata-se de um modelo mais próximo do real quando comparado ao apresentado em [7], por utilizar procedimentos exportados para representar métodos dos objetos.

Estruturas ODL/IDL	Mapeamento em SDL
Group Type	Block Type
Object Type	Block Type
Interface Type	Process Type
Referência de objeto	Pid[2]
Operação *oneway* (assíncrona)	Sinal precedido por pCALL_
Operação (síncrona)	Procedimento exportado, onde o valore retornado é do tipo *choice*, ou seja, pode assumir valores de tipos diferentes, um para o resultado, outro para ocorrência de exceção
Enum	NewType com os literais correspondentes
Typedef	Syntype
Struct	NewType com a estrutura correspondente
Constant	Synonym

Tabela 1. Mapeamento de ODL para SDL

Sendo assim, seguindo este mapeamento, os objetos as-UAP, PA, IA, namedUA e anonUA são modelados como *Block Types* (Figura 2) e trocam sinais entre si da forma mostrada na figura 1.

Adicionalmente, no modelo SDL foram implementados os objetos NameServer, IdDispatcher e Creator. Estes objetos não estão dentro das especificações TINA, porém auxiliam a implementação da sessão de acesso.

O bloco NameServer é um Servidor de Nomes, cuja única funcionalidade é fornecer ao usuário uma referência ao objeto IA do provedor desejado quando solicitado, para que se possa estabelecer uma comunicação entre o usuário e o provedor. Para isto o objeto PA faz uso de um método do Servidor de Nomes chamado *getReference*. Para que o Servidor de Nomes possua referências para os objetos IA, sempre que um novo IA é instanciado, este se registra junto ao Servidor de Nomes através do método *registerReference*.

[2] *Pid: Process Identification*, corresponde a uma referência à localização do processo (identificador do processo)

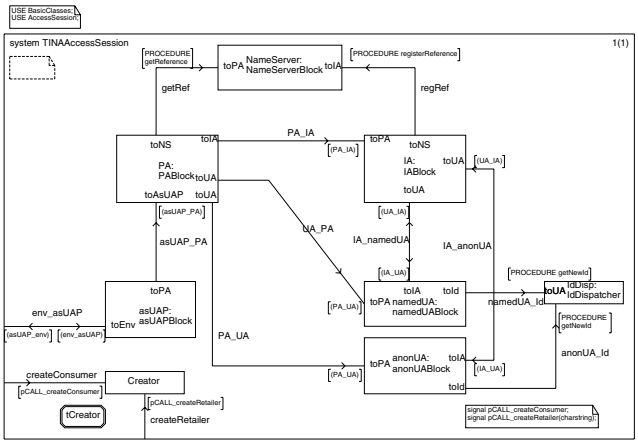

Figura 2. Modelo SDL da Sessão de Acesso TINA

O bloco IdDispacher, por sua vez, é responsável apenas por gerar números seqüenciais (através do método *getNewId*) que serão utilizados pelo namedUA e anonUA como identificadores de sessões de acesso. Finalmente, tem-se o objeto criador – *Creator* – (Figura 3), cujo único objetivo é tornar mais prática a instanciação de novos consumidores e provedores durante a simulação do modelo, através apenas de dois sinais: *createConsumer* e *createRetailer*.

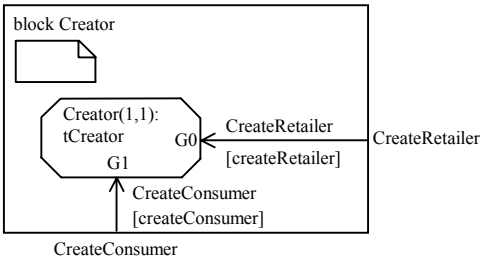

Figura 3. Funcionalidade do bloco (objeto) Creator

Quanto aos objetos pertencentes à arquitetura de serviços TINA, todos eles foram organizados da mesma forma. Foram mapeados em blocos SDL e suas interfaces em processos SDL. Entretanto, como um objeto especificado em ODL pode ser criado dinamicamente, enquanto um bloco SDL não pode ser instanciado (apenas processos SDL podem), para cada bloco SDL, tem-se além dos processos que representam suas interfaces, um processo pré-existente que é responsável pela "instanciação do bloco" – um processo criador (Figura 4). Este processo instancia cada um dos processos do bloco e após a criação dos mesmos envia a cada um deles as referências de todos os outros processos componentes deste objeto instanciado, de forma que os processos possam se comunicar (uma vez que conhecem os *Pid* uns dos outros), e assim atuar como uma unidade, um objeto.

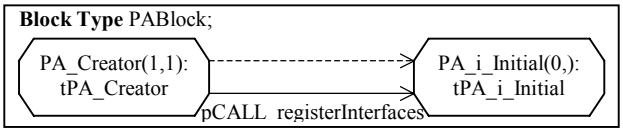

Figura 4. Solução para a instanciação de blocos SDL

A Figura 5 mostra a especificação em SDL bloco PA, com a presença do processo criador. Este processo contém apenas um procedimento responsável pela instanciação do objeto. Procedimento este que instancia as interfaces do objeto e envia a cada uma delas a referência das outras (através do sinal *pCALL_registerInterfaces*, que representa uma operação assíncrona) para que as interfaces possam interagir umas com as outras, se comportando como um único objeto. Além disso, o processo criador ainda pode fornecer outras informações relevantes às interfaces no momento de sua criação.

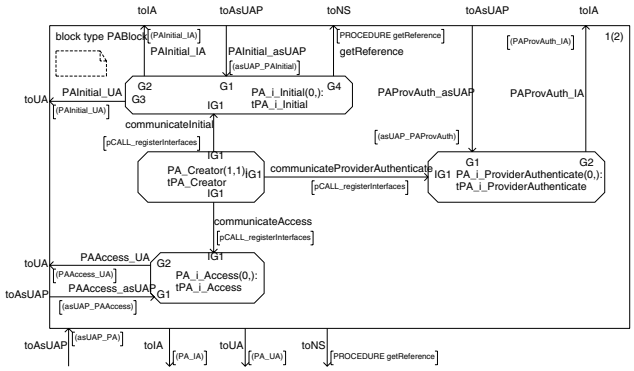

Figura 5. Bloco PA

A Figura 6 mostra o processo *PA_i_Initial*, que, assim como todos os demais processos, possui seu conjunto de procedimentos exportados (representando os métodos) que podem ser acessados por quaisquer outros blocos. Da mesma forma, todas as interfaces possuem um conjunto de procedimentos importados de outras interfaces (do mesmo objeto ou de outro), que representam os métodos que as mesmas podem acessar.

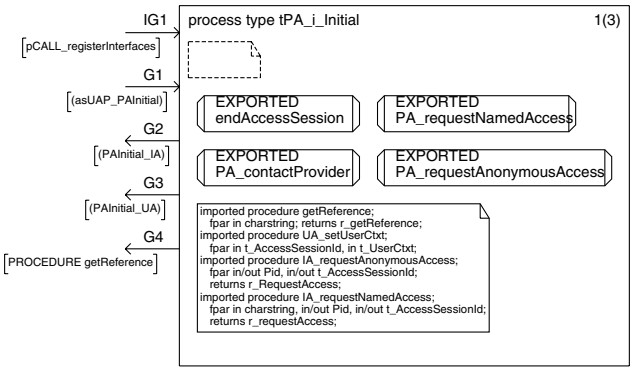

Figura 6. Interface i_Initial do objeto PA

A Figura 7 apresenta o procedimento exportado *PA_contactProvider*, no qual é definida a especificação comportamental do método por ele representado. Esta especificação não se encontra no modelo ODL, uma vez que o ODL se limita à especificar os objetos e suas interfaces, sem se preocupar com a dinâmica dos mesmos.

Este procedimento exportado é responsável por contactar o provedor cujo nome é fornecido através do parâmetro *desiredProviderParam*. Para isto, é chamado o procedimento

remoto *getReference* do bloco *NameServer* (o qual foi previamente importado pelo processo em questão, como pode ser visto na Figura 6). O resultado deste procedimento é retornado, informando o endereço do IA do provedor, ou então, retornando uma exceção (quando for o caso).

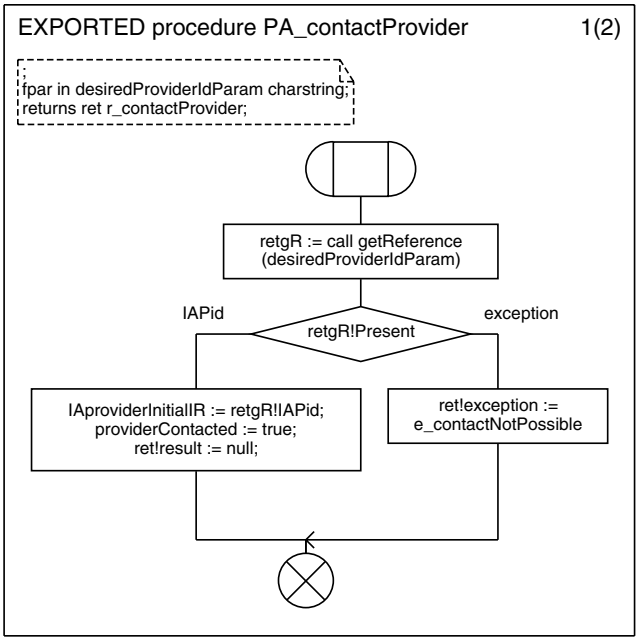

Figura 7. Procedimento exportado PA_contactProvider

O mapeamento proposto neste artigo se distingue de outras propostas ([6],[7]) ao estabelecer que para cada método (procedimento exportado) que possa gerar uma exceção, é definido um tipo de dados para o resultado do método. Este tipo é criado com o gerador *choice*, que permite que a uma mesma variável possam ser atribuídos valores de tipos distintos. Na Figura 8 tem-se um exemplo da definição do tipo do resultado do método (ou processo exportado) *PA_contactProvider*. O resultado pode assumir valores do tipo *null* ou do tipo *e_ContactProvider*. Neste exemplo, o método não retorna nenhum valor, assim, caso tudo ocorra corretamente, o resultado do método terá o seu componente *result* definido para *null*. Caso ocorra uma exceção, o resultado terá o seu componente *exception* definido para a exceção ocorrida (do tipo *e_ContactProvider*).

```
NewType r_contactProvider choice
   result null;
   exception e_ContactProvider;
EndNewType r_contactProvider;
```

Figura 8. Definição do tipo do resultado do método *PA_contactProvider*

O objeto que invocou este método, pode facilmente verificar a ocorrência da exceção, verificando qual componente do resultado está definido (Figura 9) e seguindo o fluxo normal ou executando os procedimentos pertinentes à ocorrência de uma exceção (a exceção pode ser identificada através do componente *exception*).

Todos os outros blocos (as-UAP, IA, anonUA e namedUA) foram organizados da mesma forma.

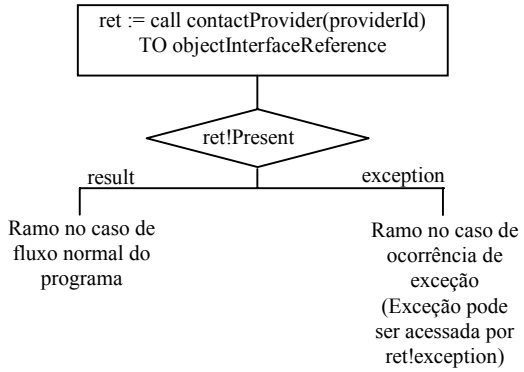

Figura 9. Detecção de ocorrência de exceção

Além disso, pode-se destacar o fato dos blocos namedUA e anonUA possuírem funcionalidades bastante similares, pois ambos representam os usuários (anônimo ou não) no domínio do provedor. Esta similaridade entre blocos, possibilitou a criação de um bloco chamado UABlock, o qual nunca é instanciado, e que possui as características (processos e procedimentos) comuns entre os blocos anonUA e namedUA (Figura 10). Desta forma, os blocos anonUA e namedUA são definidos como herdeiros do bloco UABlock, o que lhes permite partilhar de suas características comuns e adicionar às mesmas particularidades de seu funcionamento, através de redefinição de alguns dos procedimentos herdados.

Dentre os procedimentos comuns aos blocos namedUA e anonUA, que puderam ser definidos em UABlock, pode-se destacar os procedimentos responsáveis por listar os serviços disponíveis para o usuário e finalizar uma sessão de acesso. Dentre os procedimentos específicos dos blocos, pode-se citar o reponsável pelo estabelecimento de uma nova sessão de acesso, uma vez que no bloco namedUA é necessário realizar a autenticação do usuário, enquanto no bloco anonUA não, já que ele trata apenas das sessões de acesso anônimas.

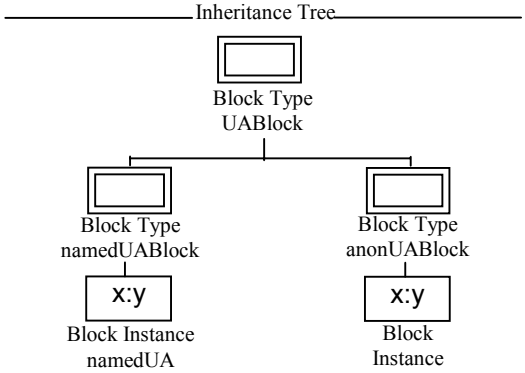

Figura 10. Herança dos blocos namedUA e anonUA

4. VALIDAÇÃO E SIMULAÇÕES DA SESSÃO DE ACESSO TINA

No modelo de sessão de acesso TINA desenvolvido, todas as funcionalidades esperadas foram cobertas e totalmente testadas:

- Contato inicial com o Provedor ou *Retailer*, através do IA, cuja referência é obtida junto ao Servidor de Nomes;
- Estabelecimento de sessões de acesso identificadas (namedUA), incluindo autenticação do usuário;
- Estabelecimento de sessões de acesso anônimas (anonUA);
- Obtenção de lista de serviços disponibilizados para o usuário que foi autenticado (ou usuário anônimo);
- Finalização das sessões estabelecidas (processo de *logout*).

O sistema foi completamente validado, utilizando-se a ferramenta *Validator* da suíte SDT da Telelogic para desenvolvimento em SDL, sendo possível a detecção de pequenos erros na especificação, como por exemplo, uma comparação errada que era realizada no processo *UA_endAccessSession* do bloco UA.

Tal erro pode facilmente ser descoberto após a execução do *Validator*, notando-se que nenhum símbolo abaixo de um determinado ramo da comparação havia sido alcançado (Figura 11). Após a correção desta comparação, o sistema foi novamente validado e todos os símbolos esperados foram alcançados.

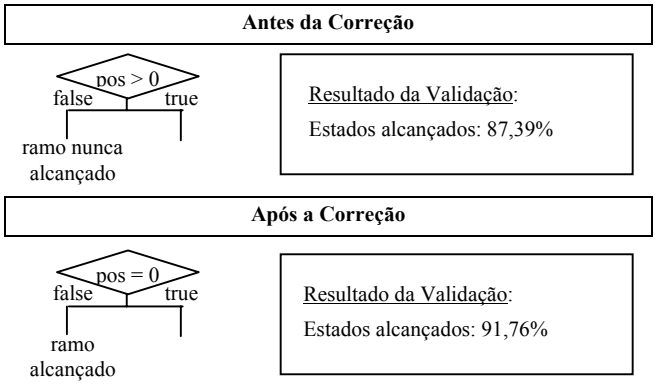

Figura 11. Detecção de erros através da validação

Nem todos os símbolos (aproximadamente 8%) foram alcançados durante a validação, pois alguns deles tratam situações de erro que poderiam ocorrer no sistema real, mas que não ocorrem em uma situação ideal. Um exemplo é a possibilidade de um PA se comunicar com o namedUA errado. Dada a especificação do PA, isto é impossível porém, há no namedUA um tratamento especial para este tipo de situação. Além disso, alguns símbolos do bloco UA (definidos como VIRTUAL) foram redefinidos em anonUA e namedUA, sendo assim, nunca são alcançados.

Após a completa validação do sistema, foram simuladas as principais funcionalidades para a garantia da validade do sistema. Para tanto foi utilizada a ferramenta *Simulator*, que também é parte integrante do SDT. Esta ferramenta permite o acompanhamento total da simulação gerando diagramas de trocas de mensagens (MSC – *Message Sequence Charts*), que permitem visualizar a interação entre os diversos componentes do sistema.

Na Figura 12, pode-se visualizar um destes diagramas gerados.

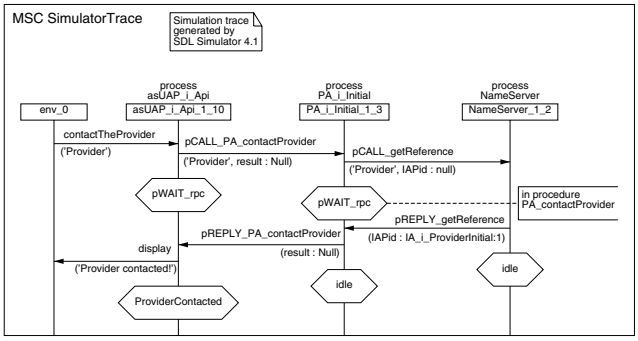

Figura 12. Contato inicial com o provedor

Durante o processo inicial, o usuário informa o desejo de contactar o provedor na inteface disponibilizada pelo as-UAP, aqui representado pelo envio do sinal *contactTheProvider* enviado à interface *i_Api* do objeto as-UAP. O objeto repassa esse pedido para o PA através de sua interface *i_Initial* e este contacta o Servidor de Nomes (através do procedimento *getReference*) afim de obter uma referência ao IA do provedor desejado. O Servidor de Nomes retorna a referência ao PA, que fica sabendo como contactar o provedor. Uma mensagem de sucesso é exibida ao usuário, a qual é aqui representada pelo sinal *display*.

Em um segundo passo, o usuário deve solicitar uma sessão de acesso ao provedor (que já pode ser contactado, uma vez que o PA possui uma referência para IA). Esta sessão pode ser anônima ou identificada. No caso de uma sessão anônima (Figura 13), o usuário solicita a sessão através de alguma interface gráfica – aqui representada pelo sinal *requestAccess* da interface *i_Api*. Essa solicitação é repassada para o PA através do método *PA_requestAnonymousAccess* da interface *i_Initia*l, que a repassa para o objeto IA através do método *IA_requestAnonymousAccess* da interface *i_ProviderInitia*l.

O método *IA_requestNamedAccess* solicita então ao processo anonUA_Creator a instanciação do objeto anonUA, que passará a ser o representante do usuário no domínio do provedor. Após a instanciação, o anonUA_Creator passa algumas informações relevantes às interfaces que compõem o anonUA (como por exemplo os endereços das outras interfaces).

O IA ainda solicita ao anonUA recém-criado o estabelecimento de uma sessão de acesso através de uma chamada ao método *UA_setupAccessSession* da interface *i_Initia*l. Este método solicita então ao objeto IdDispatcher um identificador para a sessão a ser estabelecida através do método *getReferenc*e.

Uma vez aceita a nova sessão, o PA repassa ao anonUA informações de contexto do usuário (capacidades do terminal usado, algumas interfaces importantes, etc) através do método *UA_setUserCtxt* da interface *i_ProviderNamedAccess*. Só aí o as-UAP recebe uma confirmação do estabelecimento da sessão de acesso (através do resultado do método *PA_requestAccess*) juntamente com um identificador da mesma, o *asI*d. Finalmente o as-UAP pode exibir ao usuário uma mensagem de sucesso do estabelecimento da sessão de acesso, representada aqui pelo sinal *displayAsI*d.

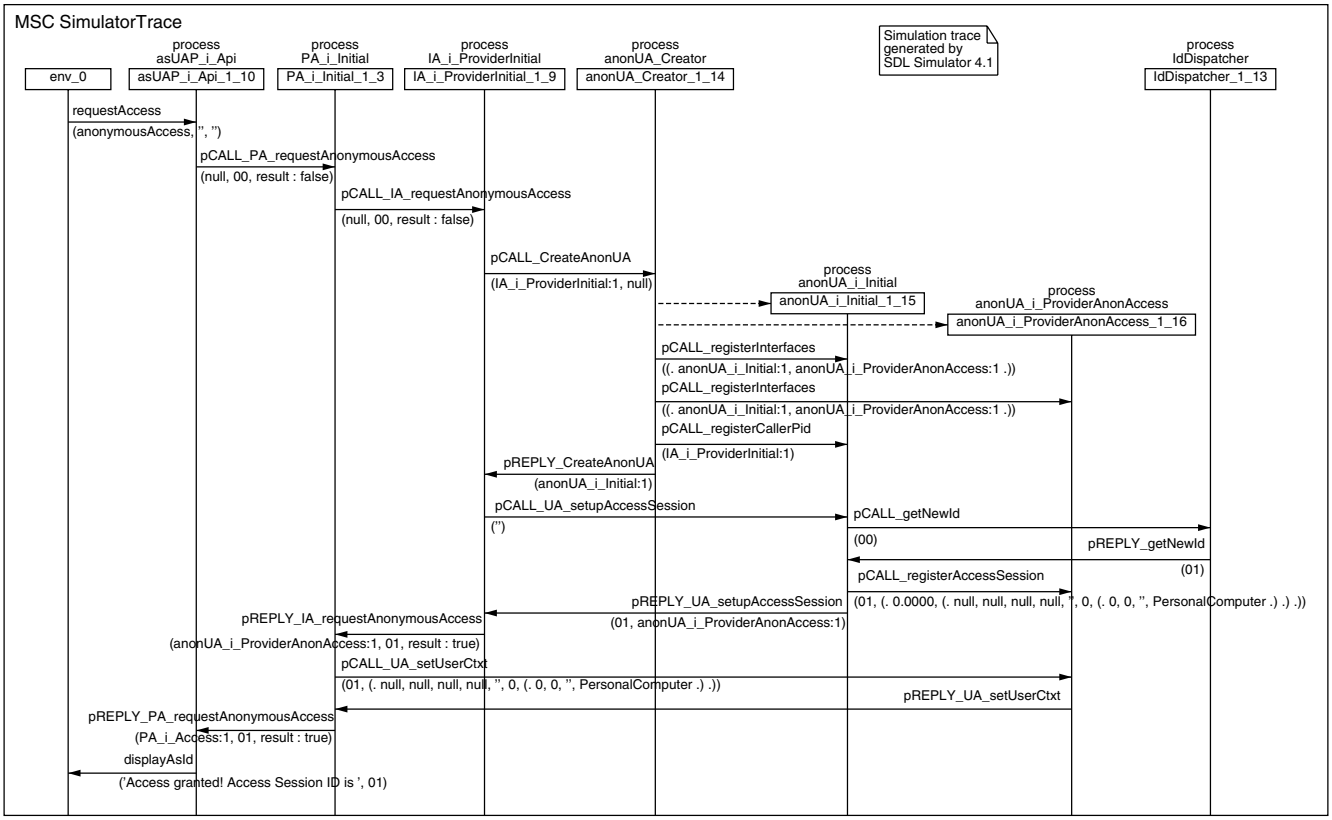

Figura 13. Estabelecimento de uma sessão de acesso anônima

5. CONCLUSÃO

O trabalho apresentado neste artigo propõe uma especificação formal da Sessão de Acesso em uma Arquitetura de Serviços TINA. Para tanto, é introduzido um mapeamento de ODL para SDL mais adequado, pois se aproxima mais do conceito de um objeto real com múltiplas interfaces, através do mapeamento de seus métodos para procedimentos exportados, com a possibilidade do tratamento das exceções ocorridas.

Utilizando este mapeamento, foi construído um modelo orientado a objetos em SDL da Sessão de Acesso TINA, conferindo à mesma uma formalidade também em seus aspectos comportamentais, frente à formalidade conferida pelo ODL (que se resume à especificação das interfaces dos objetos envolvidos). Sendo assim, a especificação em SDL permite a observação de toda a dinâmica envolvida no processos da sessão de acesso, possibilitando a simulação de diferentes funcionalidades do modelo e sua completa validação.

Para tanto, fez-se uso do conjunto de ferramentas SDT, utilizando-se o *Simulator* para a simulação dos casos de uso mais importantes da sessão de acesso e o *Validator* para a validação completa do sistema. Utilizando-se de tais ferramentas, todas as possíveis situações puderam ser testadas, o que garante a validade do modelo proposto. Pode-se desta forma, facilitar o processo de uma eventual implementação, uma vez que a lógica do sistema já está verificada e não possui erros.

6. REFERÊNCIAS

[1] Chapman, M., Montesi, S. *Overall Concepts and Principles of TINA*. TINA Consortium, February 1995.

[2] Abarca, C. et al. *Service Architecture – Version 5.0*. TINA Consortium, June 1997.

[3] Parhar, A. *TINA Object Definition Language Manual – Version 2.3*. TINA Consortium, July 1996.

[4] ITU-T (CCITT Recommendation Z.100). *Specification and Description Language*. Geneve, Switzerland, 1993.

[5] Telelogic AB. *Telelogic Tau 4.1 SDL Suite Getting Started*. Sweden, September 2000.

[6] Björkander, M. *Mapping IDL to SDL*. Telelogic AB, 1997.

[7] Kolberg, M., Sinnott, R., Magill, E. *Experiences modelling and using formal object-oriented telecommunication service frameworks*. Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 31, pp. 2577-2592, December 1999.

[8] Olsen A. et al. *The Pros and Cons of using SDL for creation of Distributed Services*. In: Proceedings of the Sixth International Conference on Intelligence in Services and Networks (IS&N'99), Barcelona, Spain, 1999.

[9] Sherratt, E., Loftus, C. *Designing distributed services with SDL*. IEEE Concurrency, vol. 08, n. 01, pp. 59-66, January-March 2000.