Implementando especificações LOTOS na plataforma de distribuição ANSAware

Nelson S. Rosa[*] *Paulo R.F. Cunha*[†]
Djamel F.H. Sadok[‡]

Departamento de Informática
Universidade Federal de Pernambuco
50732-970 Recife. Pernambuco
E-mail: {nsr,prfc,djamel}@di.ufpe.br

Resumo

A fase de realização da trajetória de projeto de sistemas distribuídos baseado em métodos formais, consiste em mapear o modelo de implementação descrito em uma linguagem formal como LOTOS em elementos operacionais finais que apresentam um baixo grau de formalismo. Estes elementos têm sido construções de linguagens de programação como C, C++-ex e ADA. Neste artigo, no entanto, nós apresentamos uma metodologia para a realização de uma especificação LOTOS sobre a plataforma de distribuição ANSAware. Com isto, o mapeamento é feito da definição da implementação em construções das linguagens IDL e DPL. Consequentemente, todas as facilidades para comunicação e os mecanismos de transparências providos pela plataforma são utilizados. Para mostrarmos o uso da metodologia, utilizaremos a especificação de um sistema de segurança de um museu.

Abstract

An ultimate stage in the design of formally specified distributed systems, the realization, consists of mapping the implementation model described using a formal language such as LOTOS onto operational elements that have a weak formalism. These elements represent constructions from programming languages such as C, C++-ex and ADA. In this paper, a methodology for the realization of LOTOS specifications in the distributed platform ANSAware is presented. Herewith, the mapping is done from an implementation definition in constructions using ANSAware's Interface Description Language (IDL), its DPL language and C. The methodology exploits different facilities provided by this platform, including its communications services and transparency mechanisms. In order to explain and demonstrate the proposed methodology, the specification of a Museum Security System is used as an example.

[*] Mestrando do Departamento de Informática da UFPE
[†] Ph.d. Waterloo, 1981
[‡] Ph.d. Kent, 1990

1 Introdução

O projeto e a construção de sistemas distribuídos podem ser baseados em métodos formais. Esta tarefa pode ser projetada sobre uma trajetória de projeto que inicia com a captura dos requisitos do usuário e termina com um sistema real. A trajetória de projeto, de acordo com a metodologia de projeto Lotosphere[18, 7, 4, 2], está dividida em 4 fases e baseia-se na FDT(*Formal Description Technique*) LOTOS[13, 3]. Estas fases estão assim definidas: *captura de requisitos do usuário, arquitetural, implementação* e *realização*. Na primeira fase, são definidos os requisitos dos usuários sobre o sistema e um documento é gerado. Na segunda fase, a utilização de técnicas formais passa a ser possível, possibilitando uma descrição formal(a arquitetura) dos requisitos definidos na fase anterior. Na terceira fase, através de sucessivos refinamentos, obtém-se uma especificação do sistema que pode ser mapeada diretamente no ambiente de realização. Na última fase, a fase de realização, o modelo de implementação descrito em LOTOS é mapeado em elementos operacionais finais(linguagens de programação). Nesta fase, a especificação do sistema, que até então, era formalmente definida, necessita ser mapeada em construções do ambiente de realização, onde o grau de formalismo é baixo. A figura 1, apresenta uma visão simplificada da trajetória de projeto Lotosphere.

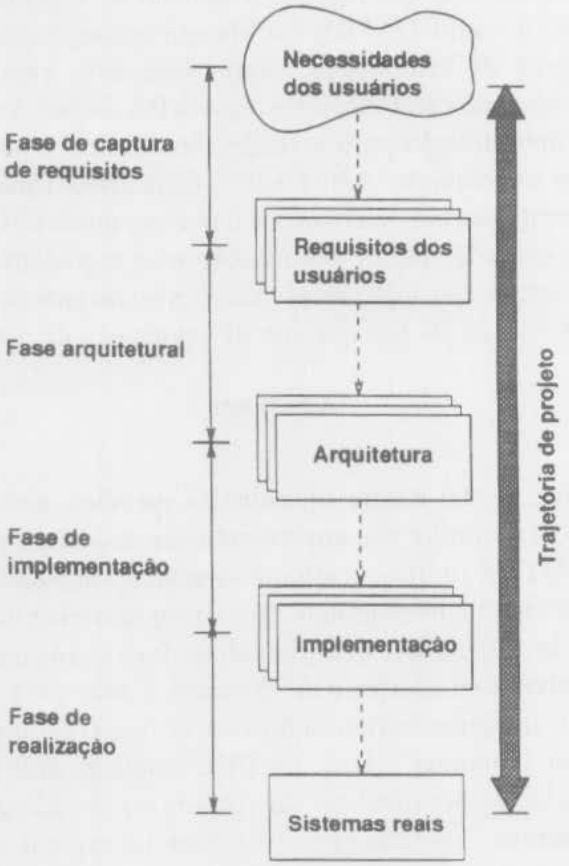

Figura 1: Trajetória de Projeto

Nesta figura, as duas primeiras fases são chamadas de fases iniciais e as duas últimas são ditas fases finais[18].

O mapeamento de especificações formais LOTOS em elementos de linguagens de programação se apresenta como uma tarefa muito complexa. Trabalhos desenvolvidos com o objetivo de executar esta tarefa se concentram no mapeamento de LOTOS em alguma linguagem de programação, como por exemplo, de LOTOS em C[15, 23], de LOTOS em C++-ex[24] e de LOTOS em ADA[14]. Com o surgimento das plataformas de distribuição como ANSAware[1], CORBA[8] e DCE[22], este mapeamento poderá se tornar mais fácil, pois estas plataformas fornecem um ambiente muito mais rico para o desenvolvimento de aplicações distribuídas do que as linguagens de programação de uso geral[21].

Neste artigo, nós apresentamos uma metodologia para ser aplicada na fase de realização das especificações LOTOS sobre a plataforma ANSAware, ou seja, nos concentramos no problema de mapear a especificação formal LOTOS em construções das linguagens IDL(*Interface Description Language*) e DPL(*Distributed Programming Language*). Para facilitar este mapeamento utilizaremos algumas transformações definidas no catálogo Lotosphere durante a fase de implementação.

Este artigo está organizado da seguinte forma: na seção 2, apresentamos uma breve introdução dos conceitos básicos de LOTOS; na seção seguinte, apresentaremos a plataforma de distribuição ANSAware, pois esta será o ambiente de realização da nossa especificação LOTOS; na seção 4, mostramos a metodologia que pode ser seguida para a construção de uma aplicação distribuída, à partir de sua especificação LOTOS, no ANSAware; na seção 5, aplicamos a metodologia definida na seção anterior para implementarmos um sistema de segurança de um museu; e por fim, na seção 6 apresentamos as conclusões obtidas com este trabalho.

2 A linguagem de especificação LOTOS

LOTOS(*Language Of Temporal Ordering Specification*) é uma técnica de descrição formal baseada na álgebra de processos e projetada para a especificação, análise e desenvolvimento de sistemas de processamento de informação distribuída e concorrente, em particular, serviços de comunicação e protocolos OSI.

Uma especificação LOTOS descreve um sistema através de uma hierarquia de componentes ativos, chamados de *processos*. Um processo é uma entidade capaz de realizar ações internas e não observáveis, e de interargir com outros processos através de ações externamente observáveis. A unidade atômica de interação entre processos é chamada de *evento*. Um evento corresponde a uma comunicação síncrona que pode acontecer entre dois processos capazes de interagir um com o outro efetuando este evento. Eventos são atômicos, no sentido de que, ocorrem instantaneamente, sem consumo de tempo, em um ponto de interação chamado *porta* e podem ou não, envolver troca de valores. A ação não observável é conhecida como ação interna ou evento interno. Cada processo tem associado um conjunto finito de portas que podem ser compartilhadas. Uma comunicação é uma ação sincronizada.

Uma especificação LOTOS tem a seguinte estrutura:

```
specification NomeEspec[porta1,....,portan](var1:sorte1,..... varm:sortep)
    : noexit          (* funcionalidade:  pode ser noexit, como neste caso,
                         ou exit                                          *)
lib
```

```
    ...                    (* referência a tipos da biblioteca de LOTOS      *)
endlib
type
    ...                    (* declarações-de-tipos                            *)
endtype
behaviour
    ...                    (* expressão-de-comportamento-da-especificação      *)
where
    process NomeProcesso [p1,..., pi](v1:s1,..., vk:sk)
        : exit     (* funcionalidade: pode ser exit, como neste caso,
                      ou noexit                                               *)
        := ...     (* expressão-de-comportamento-do-processo                  *)
    endproc
    :
endspec
```

O comportamento da especificação é visível do ambiente externo através da observação dos eventos que se passam em suas portas. A funcionalidade declarada para o sistema mostra se ele deve executar para sempre(*noexit*) ou não(*exit*). Como visto, uma especificação pode ser modular, no sentido de que existe uma expressão de comportamento geral descrevendo o comportamento de mais alto nível do sistema, a qual pode ser detalhada através de refinamentos sucessivos dos processos envolvidos. As expressões descrevem as ações(eventos) realizadas para atingir o comportamento global desejado e a declaração de tipos permite a definição de tipos abstratos de dados e suas operações[20, 3].

3 A plataforma de distribuição ANSAware

ANSAware é uma plataforma de distribuição desenvolvida a partir das especificações contidas no modelo de engenharia da arquitetura ANSA[6], em conformidade com o RM-ODP[9, 10, 11, 12]. O objetivo desta plataforma é apoiar o desenvolvimento e a execução de aplicações distribuídas. Para isto, ANSAware fornece dois grupos de componentes: o primeiro, constituído por um conjunto de ferramentas de programação, é utilizado para construção das aplicações; e o segundo, representando o ambiente de execução, é composto por programas que apóiam a execução das aplicações. As ferramentas de programação são as linguagens IDL(*Interface Description Language*) e DPL(*Distributed Programming Language*), o compilador *stubc*, o preprocessador *prepc* e algumas bibliotecas, e o ambiente de execução é formado pelo *trader*, *factory* e *node manager*[1].

A construção de uma aplicação em ANSAware consiste na definição de serviços e na construção de objetos, chamados de objetos computacionais, que fornecem e que utilizam estes serviços. Estes objetos são classificados em servidores, os que provêem os serviços, e clientes, os que utilizam os serviços. O acesso a estes serviços, é feito através de uma interface especificada em IDL e a comunicação entre os objetos é feita por passagem de mensagem.

A linguagem IDL, citada anteriormente, é utilizada para especificar as interfaces entre os componentes de uma aplicação distribuída e para definir uma representação de

dados comum o qual é usada para comunicar dados entre os sistemas heterogêneos. A especificação em IDL possui a estrutura mostrada na figura 2.

```
nomeTipo : INTERFACE =
IS COMPATIBLE WITH nomeTipo1
NEEDS nomeTipo2
BEGIN
-- tipos de dados especificos da interface
-- assinatura das operações
END.
```

Figura 2: Programa IDL

Uma vez que as especificações das interfaces para uma aplicação distribuída tenham sido definidas, os objetos que fornecem e usam estas interfaces devem ser codificados. A codificação é feita utilizando-se as linguagens C e DPL. A última, é uma linguagem fornecida por ANSAware para definir as interações entre os objetos. Após a codificação, os objetos computacionais são compilados e transformados em objetos de engenharia(cápsulas), os quais são a representação, em tempo de execução, dos objetos computacionais.

4 Metodologia de realização

Antes que possamos definir a proposta utilizada durante a fase de realização, devemos mostrar de que forma obtivemos o modelo de implementação(seção 1) que serviu como base para a realização da especificação no ambiente ANSAware.

4.1 O modelo de implementação

O modelo de implementação é o resultado que se obtém da fase de implementação da metodologia de projeto. Esta fase se inicia após a definição da arquitetura e tem o objetivo de gerar uma especificação, equivalente a primeira e ainda formal, mas que se encontre mais próxima de objetos reais. Sendo assim, esta tarefa consiste em obter uma especificação com um nível de abstração mais baixo, partindo-se de uma outra especificação mais abstrata, obtida nas fases iniciais da trajetória de projeto. Na figura 3, podemos ver os diferentes níveis de abstração de uma especificação.

Para conseguirmos obter uma especificação com um baixo nível de abstração, nós adotamos uma abordagem que faz transformações na especificação direcionadas para a implementação[5, 20]. Estas transformações fazem parte do catálogo de transformações desenvolvidas no projeto Lotosphere[2] e geram uma especificação LOTOMATON[17, 16, 19].

A adoção de um modelo como este, onde, a cada transformação é gerada uma especificação intermediária que se encontra mais próxima da implementação do que a especificação

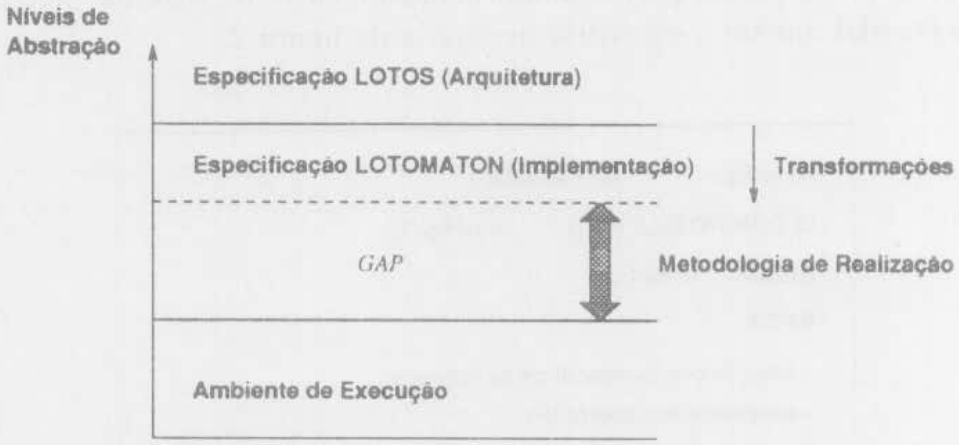

Figura 3: Níveis de abstração

anterior, facilita significativamente a complexa tarefa de realização de uma especificação formal. Neste caso, são removidas da especificação algumas construções não aceitas pelo ambiente de realização. Exemplos destes tipos de construções: a recursão infinita e o *multi-way rendez-vous*[2].

4.2 Passos da Metodologia

Uma vez que temos o nosso modelo de implementação definido, iniciamos a fase de realização. Esta fase envolve normalmente um processo de tradução(figura 4) e não uma transformação, como na fase de implementação. A especificação deve ser traduzida em construções das linguagens de programação providas pelo ANSAware: IDL e DPL.

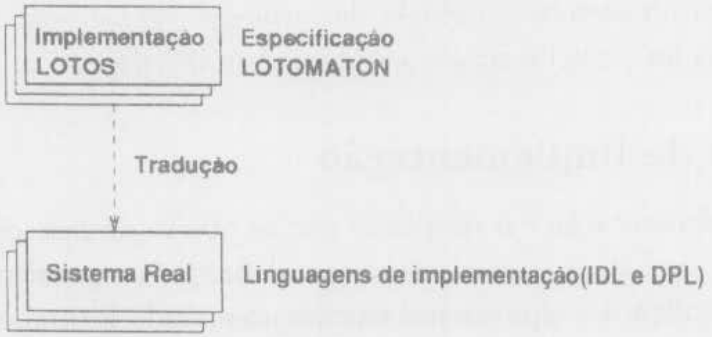

Figura 4: Fase de Realização

A metodologia guia o processo de mapeamento orientando as decisões que devem ser tomadas em cada uma das fases abaixo:

- A representação dos processos LOTOS no ANSAware;
- A implementação do comportamento dos processos LOTOS;
- Interface dos processos LOTOS; e

- A interação entre os processos LOTOS.

Fundamental para tratarmos as questões acima, foram as seguintes considerações feitas no ambiente ANSAware: cada objeto computacional(servidor) fornece um único serviço; cada objeto engenharia contém um único objeto computacional; e cada cápsula contém um único objeto de engenharia. Estas considerações foram feitas, para facilitar a identificação de elementos do ANSAware que pudessem representar as construções LOTOS, sem comprometimento no mapeamento.

4.2.1 Representação dos processos LOTOS

O componente ativo de uma especificação LOTOS é o processo, e a unidade de execução, ou seja, o elemento ativo de uma aplicação ANSAware, é a cápsula. Desta forma, cada processo LOTOS será representado em uma cápsula ANSAware. Esta cápsula, a exemplo do processo LOTOS, é capaz de se comunicar com outras cápsulas, de executar em paralelo com ou sem sincronização e, de maneira geral, interagir das mais diversas formas com outras cápsulas.

A representação de processos LOTOS como cápsulas se apresenta como uma característica fundamental deste mapeamento, haja vista, uma cápsula dentro do ANSAware pode ser distribuída, migrada, desativada e ativada, ou seja, todos os mecanismos de transparência providos por ANSAware são incorporados à cápsula. Sendo assim, a realização da especificação LOTOS sobre ANSAware terá um ambiente que já possui diversos mecanismos de transparência implementados, o que significa uma grande facilidade e importância para os programadores de aplicações distribuídas.

4.2.2 Implementação do comportamento dos processos LOTOS

Após as transformações da fase de implementação obtemos uma especificação LOTOMATON. Esta especificação contém uma máquina de estados finitos representando o comportamento de cada processo LOTOS. Estas máquinas são mapeadas diretamente em comandos DPL e C.

4.2.3 Interface dos processos LOTOS

A interação entre os componentes de uma aplicação ANSAware se baseia na definição da interface em IDL. Por outro lado, a interface de um processo LOTOS é o seu conjunto de portas, as quais, representam os pontos de interação entre os processos. Sendo assim, e considerando que cada cápsula só contém um único objeto computacional e este, por sua vez, só fornece um único serviço, a interface do processo é definida diretamente em IDL da seguinte forma:

Especificação LOTOS:

```
process NomeProcesso [p1, ..., pn]:exit
        := ...
endproc
```

Especificação IDL:

```
NomeProcesso : INTERFACE =
...
BEGIN
    p1 : OPERATION [...] RETURNS [...]
    ...
    pn : OPERATION [...] RETURNS [...]
END
```

O uso de uma linguagem declarativa como IDL para especificar as interfaces sobre as quais os componentes irão interagir, permite a transparência nas interações. Além disto, uma outra característica importante deste mapeamento para a plataforma ANSAware, reside no fato de termos uma linguagem de alto nível só para especificarmos as interfaces e, portanto, não precisamos usar uma linguagem de uso geral também para este fim.

4.2.4 Interação entre os processos LOTOS

Em LOTOS, os processos interagem através da execução de ações externamente observáveis, ou seja, de ações definidas na sua interface. No ANSAware, as interações são definidas em DPL e para que elas ocorram, uma referência de interface deve ser obtida. Por conseguinte, as interações definidas na especificação devem ser mapeadas em dois comandos DPL: o primeiro, para obtenção da referência de interface; e o segundo, para a chamada da operação na interface (equivalente à ação). Como exemplo, vamos considerar uma ação `act` de um processo **Prc** qualquer da especificação

Especificação LOTOS:

```
process Prc[act]:exit
    := act; ...
endproc
```

Comandos DPL:

```
1. ! { ref } <- traderRef$Import("Prc","","")
2. ! { } <- ref$act()
```

Com as interações mapeadas em DPL, todos os mecanismos de comunicação se tornam transparentes ao programador, ou seja, a linguagem permite que o programador trate a comunicação em um nível mais abstrato, sem que haja necessidade de se trabalhar com primitivas de comunicação de baixo nível. Com isto, as interações dos processos LOTOS, definidas na especificação, são mapeadas sem nenhuma dificuldade para o ambiente de realização.

Cada ação LOTOS será implementada como uma operação da interface definida em IDL. Logo, a estrutura geral da implementação da ação LOTOS `act` em DPL é:

```
int Nomeprocesso_act( _attr )
  ansa_InterfaceAttr *_attr;
  {  ...
    switch(Estado_Corrente)
      {
        case s:  ...
          Estado_Corrente = t;
        break;  ...
      }
    return 1;
  }
```

Na próxima seção, poderemos ver estes passos definidos acima aplicados a um exemplo concreto.

5 Estudo de caso: Sistema de Segurança de um Museu

O exemplo que utilizamos para aplicarmos a metodologia definida na seção 4.2 é de um sistema de segurança de um museu. A especificação LOTOS deste sistema e a sua implementação em C podem ser encontradas em [20]. A escolha deste exemplo foi feita por duas questões básicas: a primeira, pelo fato do exemplo ser bastante didático; e a segunda, por ser uma especificação LOTOS bem representativa do uso da linguagem.

A aplicação proposta, como exemplo, é a de um sistema para segurança de um museu. Quando um visitante entra no museu, um sensor sinaliza este evento, o mesmo ocorre quando um visitante deixa o museu. Desta maneira o sistema é capaz de informar o número de visitantes presentes no museu a cada instante. Este sistema opera em um de seus dois modos: o modo dia (**Day**) ou o modo noite (**Night**). Ele pode ser chaveado do modo dia, no qual visitantes podem entrar/sair do museu e um contador mostra o número de visitantes presentes, para o modo noite, no qual se supõe que nenhum visitante permanece no museu e nem poderá entrar ou sair dele. Um alarme será disparado se o modo noite for ativado e ainda restarem visitantes no museu. O alarme será também disparado quando um visitante entrar ou deixar o museu durante o modo noite.

A partir da especificação informal foi construída uma especificação formal em LOTOS. Esta especificação representa a definição da arquitetura do sistema. A seguir, vemos os principais componentes desta especificação:

```
SPECIFICATION Securitysystem [display,bell,switch,visitor_in,visitor_out]
      : NOEXIT
TYPE        (* Definições dos Tipos Abstratos de Dados *)
...
ENDTYPE
...
BEHAVIOUR
```

```
    HIDE channel IN
        Console [switch.display.bell.chan nel] (0)
    |[channel.switch]|
        Counter [visitor_in.visitor_out.switch.channel] (0)
WHERE
    PROCESS Console [switch.display.bell.channel] (x:nat) : NOEXIT
    :=
        ...
    ENDPROC
    PROCESS Counter [visitor_in.visitor_out.switch.channel] (x:nat) : NOEXIT
    :=
        ...
    ENDPROC
ENDSPEC
```

A expressão de comportamento de mais alto nível da especificação é definida pelos processos **Counter** e **Console** compostos em paralelo. Estes processos definem os dois principais componentes dos sistema de segurança e se encontram mais detalhados em [20]).

A partir da definição da arquitetura acima, a especificação passa por uma série de transformações que removem a recursão infinita e o *three-way rendez-vous* e produzem uma especificação LOTOMATON. Esta especificação se encontra menos abstrata e mais fácil de ser mapeada em DPL e C.

Devido ao pouco espaço disponível, as questões do mapeamento serão consideradas tomando-se o processo **Console** como exemplo. A especificação LOTOMATON para o processo **Console** pode ser vista a seguir:

Console [**switch.req.display.bell.channel**] (0) =

```
    AUT  (0, x:Nat=0 ::   display!x,          1)
         (1,              channel?x:Nat,     2)
         (2,              display!x,         1)
         (1,              switch,            3)
         (3,              req,               4)
         (4,              channel?x:Nat,     5)
         (3,              channel?x:Nat,     5)
         (5, [x ne 0] ::  bell,              6)
         (5, [x eq 0] ::  channel?x:Nat,     7)
         (5, [x eq 0] ::  switch,            8)
         (6,              channel?x:Nat,     7)
         (7,              bell,              6)
         (6,              switch,            8)
         (8,              req,               9)
         (9,              channel?x:Nat,     2)
         (8,              chanell?x:Nat,     2)
    CURST 0
    ENDAUT
```

Uma vez que temos a arquitetura e o modelo de implementação já definidos, podemos utilizar a metodologia para realizarmos a especificação no ANSAware. A seguir podemos ver de que forma a metodologia foi aplicada.

- De acordo com a seção 4.2.1, os processos LOTOS **Console** e **Counter** são mapeados em duas cápsulas de mesmo nome;
- Serão definidas duas interfaces em IDL, uma para cada processo LOTOS(seção 4.2.4). A especificação da interface de **Console** é mostrada a seguir:

Console : **INTERFACE** =

BEGIN

switch : **INTERROGATION OPERATION** [] **RETURNS** [];

req : **INTERROGATION OPERATION** [] **RETURNS** [];

display : **INTERROGATION OPERATION** [] **RETURNS** [];

bell : **INTERROGATION OPERATION** [] **RETURNS** [];

channel : **INTERROGATION OPERATION** [num : **CARDINAL**] **RETURNS** [];

END.

- O código DPL para ação ocorrida na porta *req* de **Console** é o seguinte(seção 4.2.4):

```
int Console_req( _attr )
    ansa_InterfaceAttr *_attr;
    {
      ansa_InterfaceRef ir_CT;
      ansa_Cardinal res;

      switch(Estado_Console)
      {
        case 3:
              ! { ir_CT } <- traderRefSImport("Counter","/","")
              ! { res } <- ir_CTSreq()
              Estado_Console = 4;
              Console_channel(_attr,res);
        break;
        case 8:
              ! { ir_CT } <- traderRefSImport("Counter","/","")
              ! { res } <- ir_CTSreq()
```

```
            Estado_Console = 9;
            Console_channel(_attr.res);
        break;
        }
    return 1;
    }
```

No código acima, vemos que a variável Estado_Console representa o estado, de acordo com a especificação LOTOMATON, do processo **Console**. Esta variável, define o estado da máquina de estados que representa o comportamento do processo. A cada ação temos uma mudança de estado. Podemos constatar do código acima que a implementação do comportamento do processo **Console** é direta(seção4.2.2).

A interação entre os processos **Console** e **Counter** na porta *req*. é feita utilizando os seguintes comandos DPL:

```
! { ir_CT } <- traderRef$Import("Counter","/","")
! { res } <- ir_CT$req()
```

6 Conclusão e trabalhos futuros

O uso de uma plataforma de distribuição como ANSAware, e não uma linguagem de programação de uso geral, facilita significativamente o desenvolvimento de uma aplicação distribuída. Isto ocorre, pelo fato da plataforma já incorporar, através de suas linguagens de programação e o seu ambiente de execução, diversos conceitos e mecanismos que são fundamentais para o desenvolvimento de uma aplicação distribuída. Baseado nisto, definimos uma metodologia que utiliza a construção de cápsulas em DPL, a definição das interfaces em IDL e os mecanismos de comunicação de alto nível fornecidos em DPL, para tornar a tarefa de realização menos complexa. Isto pode ser constatado com a implementação do sistema de segurança do museu, onde não houve dificuldade para mapear os processos LOTOS em cápsulas do ANSAware. Além disto toda a questão da comunicação foi tratada de forma abtsrata com comandos DPL. Logo, podemos advogar que o ambiente de apoio ao desenvolvimento de uma aplicação distribuída provido pela plataforma ANSAware torna a tarefa de realização de uma aplicação bem menos complexa do que se considerarmos uma linguagem de programação de uso geral como ambiente destino.

Referências

[1] APM. *ANSAware 3.0 Release Notes*. Architecture Projects Management Limited, March 1991.

[2] T. Bolognesi. Catalogue of LOTOS correctness preserving transformations - task 1.2 final deliverable. Technical Report Lo/WP1/T1.2/N0045/V03, Lotosphere Project, April 1992.

[3] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[4] J.P. Courtiat. The Lotosphere design methodology: illustrations - task 1.1 final deliverable. Technical Report Lo/WP1/T1.1/N0046/V03, Lotosphere Project, March 1992.

[5] P. Cunha, E. Najm, and J. Queiroz. Transformações de especificações LOTOS direcionadas para implementação. In *XI Simpósio Brasileiro de Redes de Computadores*, Campinas, São Paulo, Brasil, May 1993.

[6] R. van der Linden. An overview of ANSA. Technical report, APM, July 1993.

[7] S.P. Gomez. The Lotosphere design methodology: guidelines - task 1.1 final deliverable. Technical Report Lo/WP1/T.1/N0044/V04, Lotosphere Project, March 1992.

[8] Object Management Group. *The common object request broker: architecture and specification*. OMG Document Number 91.12.1, December 1991.

[9] ISO/IEC/JTC1/SC21/WG7. *Reference model of open distributed processing Part 1 - overview*. ISO, Helsink, Finland, July 1995.

[10] ISO/IEC/JTC1/SC21/WG7. *Reference model of open distributed processing Part 2 - descritive model*. ISO, Helsink, Finland, July 1995.

[11] ISO/IEC/JTC1/SC21/WG7. *Reference model of open distributed processing Part 3 - prescritive model*. ISO, Helsink, Finland, July 1995.

[12] ISO/IEC/JTC1/SC21/WG7. *Reference model of open distributed processing Part 4 - architectural semantic*. ISO, Helsink, Finland, July 1995.

[13] IS 8807 ISO/TC97/SC21/WG1. *LOTOS - A formal description technique based on the temporal ordering of observational behaviour*. ISO, 1988.

[14] M.J. Juffermans. Implementation of LOTOS specifications in ADA. Master's thesis, Twente University, June 1991.

[15] J.A. Manas and T. de Miguel. From LOTOS to C. In K.J. Turner, editor, *First International Conference on Formal Description Techniques*, pages 79–84, Stirling,Scotland, September 1988.

[16] E. Najm. Transforming contexts in LOTOMATON. In Lo/WP1/T1.2 /INRIA/N008, editor, *ESPRIT/Lotosphere Project*, October 1990.

[17] E. Najm, A. Lakas, E. Madelaine, A. Serhrouchni, and R de Simone. Interactive transformation tool for LOTOS and LOTOMATON. In *Lotosphere Workshop.3*, Pisa, Italy, September 1992.

[18] L.F. Pires. The Lotosphere design methodology: basic concepts - task 1.1/3.3 final deliverable. Technical Report Lo/WP1/T1.1/N0045/V04, Lotosphere Project, March 1992.

[19] J. Queiroz, A. Serhouchni, P. Cunha, and E. Najm. PIL: A tool for pre-implementation of LOTOS. In *International Conference on Formal Description Techniques*, Madri, Espanha, November 1990.

[20] J.A.M. Queiroz and P.R.F. Cunha. *Sistemas distribuidos: de especificações LOTOS à implementações*. IX Escola de Computação, Recife, Pernambuco, Brasil, July 1994.

[21] Nelson S. Rosa and P.R.F. Cunha. ANSAware : uma infra-estrutura para desenvolvimento de aplicações distribuídas. Technical report, Departamento de Informática, June 1995.

[22] W. Rosenberry, D. Kenney, and G. Fisher. *Understanding DCE*. O'Reilly & Associates,Inc., 1993.

[23] Peter van Eijk, Harro Kremer, and Marten van Sinderen. On the use of specification styles for automated protocol implementation from LOTOS to C. In R.L. Probert L. Logrippo and H. Ural, editors, *Tenth International IFIP WG6.1 Symposium of Protocol Specification, Testing and Verification*, pages 153-164, Ottawa,Ont.,Canada, June 1990.

[24] S. Westerdijk. Implementation of LOTOS specifications in C++-ex. Master's thesis, Twente University, August 1991.