AMBER: uma linguagem para o desenvolvimento de sistemas distribuídos

Luís Ferreira Pires
Cléver Ricardo Guareis de Farias

Telematics Systems and Services, University of Twente
P.O. Box 217, 7500 AE Enschede, the Netherlands
{pires, farias}@cs.utwente.nl

Abstract: This paper presents AMBER, the architectural modelling language for the design of distributed systems developed at the University of Twente. This language allows the specification of distributed systems in terms of structures of functional entities and their corresponding behaviour. AMBER was originally developed to support the design of telematic systems, but nowadays its major application area is focused in the re-engineering of business systems and in the development of component-based services. This paper introduces the main concepts of AMBER, its graphical notation, application areas and tool support.

Resumo: Este artigo apresenta AMBER, a linguagem de modelagem arquitetônica para o projeto de sistemas distribuídos desenvolvida na Universidade de Twente. AMBER possibilita a especificação de sistemas em termos de estruturas de entidades funcionais com os seus comportamentos correspondentes. Esta linguagem foi originalmente desenvolvida para suportar o projeto de sistemas telemáticos, mas atualmente sua maior área de aplicação é na re-engenharia de sistemas empresariais e no desenvolvimento de sistemas baseados em componentes. Este artigo introduz os principais ingredientes do modelo arquitetônico de AMBER, sua notação gráfica, áreas de aplicação da linguagem e ferramentas de suporte.

Palavras-chave: Especificação, verificação, implementação e teste de sistemas distribuídos; LOTOS; AMBER.

1. Introdução

No início dos anos 90 a comunidade científica estava trabalhando ativamente na aplicação e avaliação de Técnicas de Descrição Formal. A TDF LOTOS ([7], 'Language Of Temporal Ordering Specification') recebeu naquela oportunidade uma atenção especial, devido ao seu alto nível de abstração e elegância de sua semântica formal. O projeto Lotosphere [1] do programa europeu ESPRIT II, por exemplo, desenvolveu uma série de especificações LOTOS de protocolos do modelo OSI, especialmente da camada de aplicação, juntamente com uma metodologia de projeto suportada por LOTOS e um conjunto de ferramentas para LOTOS chamado *lite* ('LOTOS Interactive Tool Environment').

Baseado na nossa experiência no projeto Lotosphere, decidimos então trabalhar no desenvolvimento de uma base de conceitos arquitetônicos que facilitasse o desenvolvimento de sistemas distribuídos. O trabalho começou com uma análise das deficiências de LOTOS [15], e culminou no desenvolvimento de um modelo arquitetônico abstrato que resolve algumas dessas deficiências, gerando ao mesmo tempo um conhecimento mais profundo das necessidades de modelagem de sistemas distribuídos [4,9,11,12]. Esse modelo foi originalmente desenvolvido para suportar o projeto de sistemas telemáticos, como, por exemplo, serviços e protocolos do modelo OSI. Uma metodologia para o projeto de serviços e protocolos baseado nesse modelo é apresentada em [10,16].

Este artigo tem como objetivo mostrar que um modelo básico bem projetado pode ser usado no desenvolvimento de linguagens, métodos e ferramentas que podem ser aplicados em diversas áreas. Neste sentido, este artigo descreve alguns ingredientes desse modelo arquitetônico juntamente com a linguagem gráfica utilizada para representar os principais

elementos do modelo, chamada ‘Architectural Modelling Box for Enterprise Redesign’ (AMBER).

O restante deste artigo está organizado da seguinte forma: a seção 2 apresenta os conceitos básicos do modelo de AMBER, a saber, entidades funcionais, ações, interações e suas relações; a seção 3 discute a definição de comportamentos, através da combinação de relações básicas entre ações; a seção 4 introduz mecanismos que permitem estruturar comportamentos em termos de composições de comportamentos mais simples; a seção 5 descreve algumas áreas de aplicação de AMBER, bem como as ferramentas atualmente disponíveis; e finalmente a seção 6 apresenta as conclusões deste artigo.

2. Conceitos básicos

Os conceitos básicos identificados em AMBER são agrupados em dois domínios: entidades e comportamento. No domínio de entidades encontramos entidades funcionais e pontos de interação, enquanto no domínio de comportamento encontramos comportamentos consistindo de ações, interações e relações entre elas. Estes conceitos são discutidos a seguir.

2.1. Entidades funcionais

Um sistema pode ser modelado em termos de uma estrutura de entidades funcionais que interagem entre si, sendo que cada uma dessas entidades representa uma parte lógica ou física do sistema total. Uma entidade funcional é um conceito abstrato que modela um sistema, ou parte de um sistema.

Um sistema interage com o seu ambiente através de mecanismos que são desenvolvidos especialmente para este fim. O conceito de ponto de interação nos possibilita representar de modo abstrato mecanismos que suportam a interação entre sistemas. Uma entidade funcional só pode ter acesso ao comportamento de outra entidade funcional através de um de seus pontos de interação. Os pontos de interação de uma entidade funcional, na realidade, delimitam-na do ponto de vista do ambiente do sistema representado por essa entidade funcional.

A Figura 1(i) representa uma rede de comunicação e seus usuários A e B. Graficamente, uma entidade funcional é representada através de um retângulo com as bordas cortadas, enquanto que um ponto de interação é representado por uma elipse.

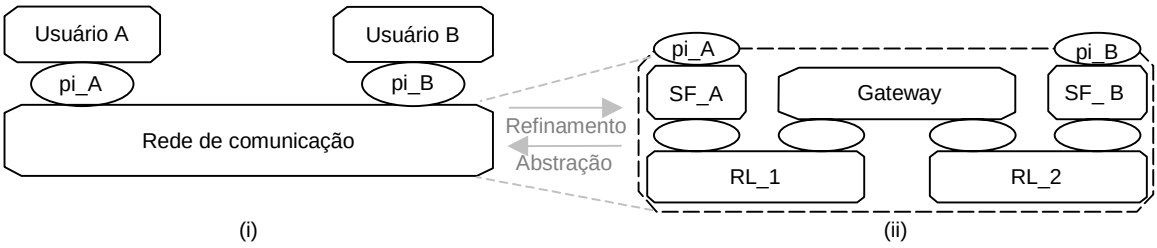


Figura 1. Entidades representando uma rede de comunicação e seus usuários.

Ao invés de representar uma rede de comunicação como uma única entidade, poderíamos tê-la representado em termos de sua estrutura interna como mostra a Figura 1(ii). Essa estrutura pode consistir, por exemplo, de entidades que representam duas redes locais (RL_1 e RL_2), dois sistemas finais (SF_A e SF_B), uma ponte (Gateway) e os pontos de interação internos à rede de comunicação para a interação entre essas entidades. Esta nova estrutura de entidades é um refinamento da entidade que representa a rede de comunicação.

2.2. Ações

Em um modelo de um sistema cada entidade funcional tem um comportamento, o qual representa a funcionalidade dessa entidade. Um comportamento representa uma coleção de atividades que a entidade pode executar isoladamente ou em cooperação com outras entidades.

Uma ação é um conceito abstrato introduzido para representar uma atividade executada por um único sistema. Consequentemente, no nosso modelo uma ação é associada a uma única entidade em um certo nível de abstração.

Atributos de ações. Atividades são executadas visando a obtenção de um resultado ou produto. Embora exista uma variedade infinita de atividades, um único conceito de ação é suficiente para modelar todas essas atividades, pois esse conceito de ação representa as características essenciais de uma atividade através de seus atributos. Uma ação tem os seguintes atributos: *informação*, que modela o produto obtido na execução de atividade que está sendo modelada; *tempo*, que modela o momento no tempo em que a atividade faz o seu produto ser disponível; e *local*, que modela o local (físico ou lógico) onde o produto de uma atividade está disponível.

Atividades podem produzir um número infinito de produtos. Três tipos de produtos são interessantes no nosso modelo: *produto concreto*, tal como carros, pães, televisores, etc., *informação*, tal como dados pessoais, mensagens, etc., e *serviço prestado*, tal como um paciente que foi atendido em um hospital. Todos esses tipos de produtos são representados em nosso modelo como valores do atributo de informação. Dessa forma, o atributo de informação de uma ação contém informação sobre o produto da atividade modelada pela ação.

Uma atividade tem uma duração de tempo, mas existe um certo momento de tempo no qual esta termina. No nosso modelo estamos interessados somente no momento no qual a atividade termina, pois antes desse momento não podemos ter certeza de seu produto. O atributo de tempo de uma ação representa o momento em que a atividade modelada pela ação termina e faz com que o seu produto seja disponível. Por exemplo, uma mensagem eletrônica é considerada como recebida quando o último byte da mensagem é recebido. Assim, o atributo de tempo de uma ação que representa a atividade de recepção de uma mensagem eletrônica contém o momento da recepção do último byte.

Uma atividade acontece em um local físico ou lógico, no qual o produto da atividade é feito disponível quando a atividade termina. O atributo de local de uma ação representa esse local para a atividade que a ação modela.

Representação de ações. Para podermos nos referir a uma ação, necessitamos identificá-la de modo inequívoco. Ocorrências de ações são identificadas em nosso modelo através de *identificadores de ação*.

A Figura 2 mostra uma ação enviar, que representa uma atividade na qual uma mensagem eletrônica é enviada. Uma ação é representada graficamente por um círculo, enquanto o identificador da ação é colocado dentro desse círculo ou em um quadrado de texto ligado ao círculo por uma linha. Atributos da ação são representados dentro de um quadrado de texto. Os símbolos ι, τ e λ representam os atributos de informação, tempo e local de uma ação, respectivamente. A ação enviar modela o envio da mensagem "Olá Cléver" através do endereço de correio eletrônico pires@cs.utwente.nl às 15 horas do dia 26 de janeiro de 2001.

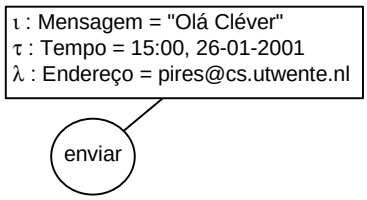


Figura 2. Ação que modela o envio de uma mensagem eletrônica.

A ação enviar na Figura 2 quando executada pode receber um único valor para cada um dos seus atributos. Porém, de forma geral queremos representar atividades que podem estabelecer diferentes valores alternativos. Por exemplo, talvez quiséssemos modelar o envio de uma mensagem entre 14 e 16 horas do dia 26 de janeiro de 2001. Em AMBER isso pode ser feito através do uso de uma condição para o atributo de tempo, representada da seguinte forma:

$$\tau : \text{Tempo} \mid 14{:}00,\ 26\text{-}01\text{-}2001 < \tau < 16{:}00,\ 26\text{-}01\text{-}2001$$

Essa condição representa que a mensagem pode somente ser enviada em algum momento de tempo dentro do intervalo entre às 14 horas e às 16 horas do dia 26 de janeiro. Isso significa que essa atividade termina no momento indicado pelo valor do seu atributo de tempo resultante, o qual somente pode receber um valor dentro desse intervalo.

A notação utilizada para os atributos de uma ação pressupõe que os tipos Mensagem, Tempo e Endereço tenham sido definidos usando uma linguagem para representação de tipos abstratos de dados. Nosso modelo se concentra na modelagem de aspectos dinâmicos de sistemas, e não considera a parte de modelagem de dados. Entretanto, nosso modelo assume que seja possível definir tipos de dados e operações envolvendo esses tipos de dados, por exemplo, de forma semelhante a linguagem ACT ONE utilizada em LOTOS.

Propriedades de ações. Uma atividade na vida real ou termina com sucesso, produzindo seus resultados esperados, ou falha. A ocorrência de uma ação no nosso modelo representa o término com sucesso de uma atividade. Assim, de forma semelhante a uma atividade na vida real, uma ação no nosso modelo ocorre ou não; ou seja, nosso modelo não admite resultados parciais ou execução parcial de ações. Se houver necessidade de representar resultados parciais em atividades, isso pode ser feito no nosso modelo usando ações mais refinadas (nível de abstração mais baixo), no qual esses resultados parciais são modelados por ações distintas. Além disso uma ação no nosso modelo ocorre uma única vez ou não ocorre nunca.

Uma ação no nosso modelo é uma unidade indivisível de atividade em um determinado nível de abstração, sendo portanto também uma unidade de comportamento. Essa propriedade de ações é geralmente chamada de *atomicidade*. Uma ação é atômica, no sentido de que ela não pode ser decomposta, a menos que o nível de abstração no qual a atividade que essa ação está sendo considerada seja abaixado. Por exemplo, a ação que representa o envio de uma mensagem eletrônica só pode ser decomposta se considerarmos a atividade em termos de sub-atividades, tais como aquelas que possibilitam o envio da mensagem (preenchimento do corpo da mensagem, pressionar o botão de mandar, etc.).

A atomicidade de uma ação impõe também que os mecanismos que executam atividades sejam confiáveis. Dessa forma, podemos utilizar as seguintes propriedades de ações na composição de comportamentos: se uma ação ocorre, outras ações podem se referir à ocorrência dessa ação e aos resultados estabelecidos por essa ação; se uma ação não ocorre, outras ações não podem se referir aos resultados dessa ação (esses não foram estabelecidos).

2.3. Interações

Uma interação no nosso modelo representa uma atividade que é executada por dois ou mais sistemas em cooperação. Consequentemente, uma interação é comum às entidades que modelam os sistemas que participam da atividade modelada pela interação.

Por serem representações de atividades, interações têm os mesmos atributos que ações. A diferença entre uma interação e uma ação é que os participantes de uma interação podem influenciá-la de formas diferentes, o que já não ocorre em uma ação. Assim, precisamos representar a participação de cada entidade em uma interação de modo separado, em termos de uma *contribuição à interação*, sendo que o resultado final da interação (o que realmente ocorre) é determinado por todas as contribuições de todos os participantes.

Representação gráfica de interações. Interações são representadas em nosso modelo ressaltando a distribuição de responsabilidades na execução de uma interação entre as entidades participantes. A Figura 3 representa o envio de uma mensagem eletrônica como uma interação entre um usuário de uma aplicação de correio eletrônico e a aplicação propriamente dita (sistema).

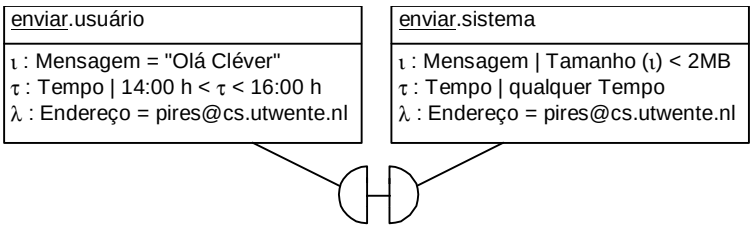


Figura 3. Envio de uma mensagem representada como uma interação.

Propriedades de interações. De modo semelhante a uma ação, uma interação é atômica e deve ser realizada de modo confiável. Dessa forma, se uma interação ocorre, todas as entidades envolvidas podem se referir à ocorrência dessa interação e aos seus resultados (valores de seus atributos); mas se uma interação não ocorre, nenhuma entidade pode se referir a sua ocorrência e aos seus resultados.

Em outras palavras, ou uma interação ocorre para todas as entidades participantes ou não ocorre para nenhuma delas.

Valores de atributos em interações. Considerando as negociações que podem ocorrer no estabelecimento de valores de atributos de interações, no caso especial de interações entre duas entidades podemos identificar as seguintes formas básicas para o estabelecimento de valores:

- *checagem de valores*: uma entidade impõe o estabelecimento de um valor específico x, enquanto a outra entidade impõe o estabelecimento de um valor específico y. Essa interação somente pode ocorrer se $x = y$, de tal modo que as entidades sincronizam em seus valores;
- *passagem de valores*: uma entidade impõe o estabelecimento de um valor específico x, enquanto a outra entidade permite o estabelecimento de qualquer valor pertencente a um conjunto de valores Y. Essa interação somente pode ocorrer para um valor x tal que $x \in Y$. Nesse caso, o valor x é originado por uma entidade e passado a outra entidade como resultado da interação;
- *geração de valores*: uma entidade permite o estabelecimento de qualquer valor pertencente a um conjunto de valores X, enquanto a outra entidade permite o

estabelecimento de qualquer valor pertencente a um conjunto de valores Y. Essa interação somente pode ocorrer para um valor x, tal que $x \in X$ e $x \in Y$. Nesse caso, um valor x é gerado por ambas as entidades de forma aleatória como resultado da interação.

Essas formas de interação podem ser generalizadas para interações entre múltiplas entidades, e podem ser combinadas de forma arbitrária.

2.4. Ações como interações integradas

Em algumas ocasiões uma ação pode ser utilizada para modelar uma *interação integrada*, a qual é uma abstração das contribuições das interações individuais das entidades envolvidas.

A Figura 4 apresenta um exemplo de uma ação integrada a que modela uma interação a com três contribuições de interação, por exemplo, de três entidades distintas. As restrições dos atributos de informação e tempo da ação a correspondem à superposição das restrições dos atributos de informação e tempo definidos pelas três contribuições de interação a. A interação a generaliza as formas básicas de estabelecimento de valores discutidas anteriormente para o caso de três entidades participando de uma interação. Assumimos no exemplo que tanto a ação a quanto as contribuições de interação a podem referir-se a algum valor de tempo τ_0, o qual foi estabelecido anteriormente.

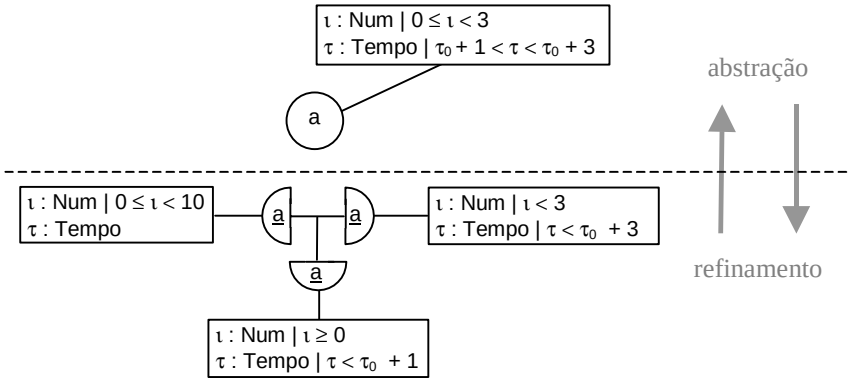


Figura 4. Exemplo de uma interação integrada.

Uma interação é considerada um refinamento de uma ação quando esta interação define uma distribuição de responsabilidades específica para a execução da ação por múltiplas entidades. Por questões de simplicidade, o termo ação é usado para referir-se tanto a ações quanto a interações no restante deste artigo. O termo interação será usado somente quando o assunto a ser abordado aplicar-se exclusivamente a interações.

2.5. Relações entre ações

O comportamento de um sistema é formado pelas atividades que são executadas por este sistema e pelo relacionamento entre estas atividades. Dessa forma, além dos conceitos de projeto para a modelagem de atividades (ações e interações), também necessitamos de conceitos de projeto para a modelagem de relacionamentos entre atividades.

A Figura 5 apresenta um exemplo de uma relação simples entre duas ações enviar e receber, as quais representam as atividades de enviar e receber uma mensagem eletrônica, respectivamente. A seta representa que a ação receber somente pode ocorrer depois que a ação enviar ocorreu. Além disso, os atributos de informação, tempo e endereço de ambas as ações estão relacionados, pois os atributos da ação receber fazem referência aos valores dos atributos que são estabelecidos na ação enviar. Na Figura 5, por exemplo, ι_{enviar} e $\iota_{receber}$ representam os valores dos atributos de informação das ações enviar e receber,

respectivamente. Os demais atributos são representados de forma semelhante em nossa notação gráfica.

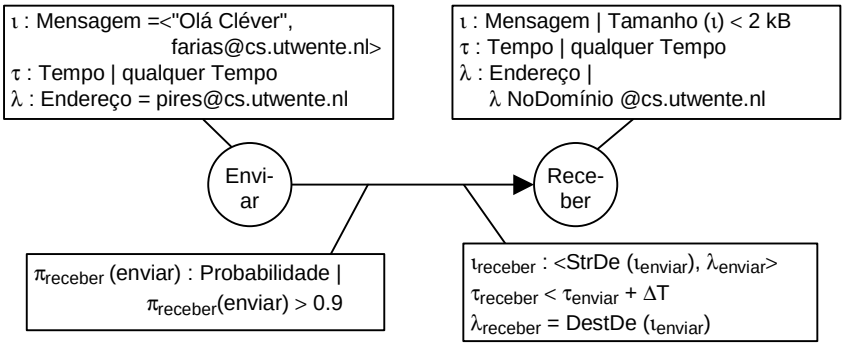


Figura 5. Relação entre duas ações.

A Figura 5 ilustra alguns tipos de restrições que podem ser impostas sobre os valores de atributos de uma ação. Por exemplo:

- a condição Tamanho(ι) < 2 kB restringe o tamanho do valor de informação estabelecido na ação receber. Esta restrição é independente da ação enviar;
- a condição $\tau_{receber} < \tau_{enviar} + \Delta T$ limita o atraso de recepção da mensagem em um máximo de ΔT. Dessa forma, o atributo de tempo de receber fica dependente do atributo de tempo de enviar;
- a condição $\iota_{receber} : < StrDe(\iota_{enviar}), \lambda_{enviar} >$ determina que o valor do atributo de informação de receber deve consistir do texto contido no valor do atributo de informação de enviar, acrescido do valor do atributo de local de enviar. Dessa forma o atributo de informação de receber depende dos atributos de informação e local de enviar.

O exemplo da Figura 5 ilustra também o *atributo de probabilidade* que é um atributo da relação entre ações. A relação entre as ações enviar e receber define implicitamente que receber pode ocorrer somente depois que enviar tenha ocorrido. O atributo de probabilidade $\pi_{receber}$(enviar) define a probabilidade condicional que receber ocorra sabendo-se que enviar ocorreu. No exemplo essa probabilidade é um valor superior a 90%.

3. Definição de comportamento

Um comportamento é definido em termos de suas ações e suas relações. No nosso modelo essas relações são chamadas de *relações de causalidade.*

3.1. Relações de causalidade

De maneira geral, a ocorrência de uma ação depende da satisfação de uma condição. Esta condição envolve a ocorrência ou não ocorrência de outras ações. Contudo, algumas ações em um comportamento não dependem de outras ações, podendo ocorrer a qualquer momento depois do começo da execução de um comportamento. Tais ações são chamadas de *ações iniciais.*

Uma ação pode ocorrer tanto sem restrições a partir do início da execução de um comportamento (ação inicial), quanto a sua ocorrência pode se tornar possível dependendo de uma condição, que envolve a ocorrência ou não ocorrência de outras ações durante a execução de um comportamento. Uma relação de causalidade define a condição para a ocorrência de uma ação. Uma ação que depende de outras para ocorrer é chamada de *ação resultante* em uma relação de causalidade.

Uma condição definida em uma relação de causalidade consiste de:

- uma *condição de causalidade*, a qual define como a ocorrência da ação resultante depende da ocorrência ou não ocorrência de outras ações;
- *restrições dos atributos de ação*, as quais definem como os valores dos atributos de informação, tempo e endereço estabelecidos pelas ações da condição de causalidade influenciam a ocorrência da ação resultante e os valores dos seus atributos de informação, tempo e endereço, e;
- um *atributo de probabilidade*, o qual define a probabilidade de ocorrência da ação resultante, caso a condição de causalidade e as restrições dos atributos de ação sejam satisfeitas na execução de um comportamento.

A ocorrência de uma ação é possível somente quando a condição de causalidade e as restrições dos atributos de ação definidos na relação de causalidade desta ação são ambas satisfeitas. Ignorando os atributos de ação, uma relação de causalidade tem a forma $\gamma_\mu \rightarrow a$, onde a é o identificador da ação resultante a, γ é a condição de causalidade da ação a, e μ é o valor do atributo de probabilidade associado a γ.

3.2. Condições básicas de causalidade

Uma relação de causalidade permite a modelagem da ordem temporal das ocorrências de ações. Consideremos duas ações a e b em uma execução de um comportamento, as quais ocorrem nos instantes τ_a e τ_b, respectivamente. Existem três possibilidades para a ordem temporal destes instantes: $\tau_a < \tau_b$, $\tau_a = \tau_b$ ou $\tau_a > \tau_b$. Estas possibilidades são disjuntas, ou seja, se uma delas se aplica a uma execução de comportamento, as outras duas não se aplicam. Dessa forma, a condição de causalidade γ da ação a pode ser definida em termos das seguintes condições básicas de causalidade:

- $\gamma = b$, a qual define que a ocorrência da ação b é uma condição para a ocorrência da ação a, de tal forma que a ocorrência de b deve preceder a ocorrência de a. Assumindo-se que tanto a quanto b ocorram, $\tau_a > \tau_b$. Esta condição é chamada de *condição de habilitação*;
- $\gamma = \neg b$, a qual define que a não ocorrência da ação b é uma condição para a ocorrência da ação a, até que a ocorra. A ação a pode ocorrer desde que a ação b não tenha ocorrido e desde que esta não ocorra simultaneamente com a. Assumindo-se que b ocorra, ou a é desabilitada por b ou a ocorre antes de b, tal que $\tau_a < \tau_b$. Esta condição é chamada de *condição de desabilitação* e a ação b é chamada de *ação de desabilitação*;
- $\gamma = \bar{b}$, a qual define que a ocorrência da ação b é uma condição para a ocorrência da ação a, tal que ação b deve acontecer simultaneamente com a. Assumindo-se que tanto a quanto b ocorram, $\tau_a = \tau_b$. Esta condição é chamada de *condição de sincronização*.

Atributo de incerteza. O exemplo da Figura 5 mostra que o atributo de probabilidade pode ser utilizado para modelar a probabilidade da ocorrência de uma ação. Neste artigo, consideramos um caso particular do atributo de probabilidade, o chamado de *atributo de incerteza*. O atributo de incerteza define, para cada condição básica de causalidade, a incerteza de que uma ação ocorra caso esta condição seja satisfeita. Dois valores de incerteza alternativos podem ser associados a uma condição básica de causalidade: deve (!), o qual define que a ação resultante deve ocorrer caso a condição associada seja satisfeita, ou pode (?), o qual define que a ação resultante pode (ou não pode) ocorrer caso a condição associada seja satisfeita.

O valor de incerteza deve corresponde ao valor de probabilidade 1 (100%), enquanto que o valor de incerteza pode corresponde a um valor de probabilidade menor que 1 (< 100%). O atributo de incerteza permite-nos somente modelar a presença ou ausência de incerteza. Os atributos de probabilidade nos permitem quantificar a incerteza da ocorrência de uma ação. Atributos de probabilidade foram investigados em detalhe em [9]; nesse artigo somente consideramos atributos de incerteza.

Uma condição básica de causalidade γ da ação a deve ser associada a um valor de incerteza deve ou pode. A associação de incerteza $\upsilon_a(\gamma)$ relaciona um valor de incerteza a uma condição básica de causalidade γ, de tal modo que $\upsilon_a(\gamma)$ = deve ou $\upsilon_a(\gamma)$ = pode, com $\gamma \in \{b, \neg b, =b\}$.

Exemplos de comportamentos simples. Ações iniciais não dependem de outras ações. Para podermos definir a condição de uma ação inicial de forma semelhante às outras ações, nós introduzimos a *condição de início*. A condição de início, representada pelo símbolo $\surd$, é sempre satisfeita a partir de um certo momento determinado pelos mecanismos que instanciam (criam) o comportamento. Contudo, tais mecanismos não são representados explicitamente no nosso modelo.

A Figura 6 apresenta algumas relações simples entre duas ações a e b, e como relações de causalidade são representadas nas notações gráfica e textual.

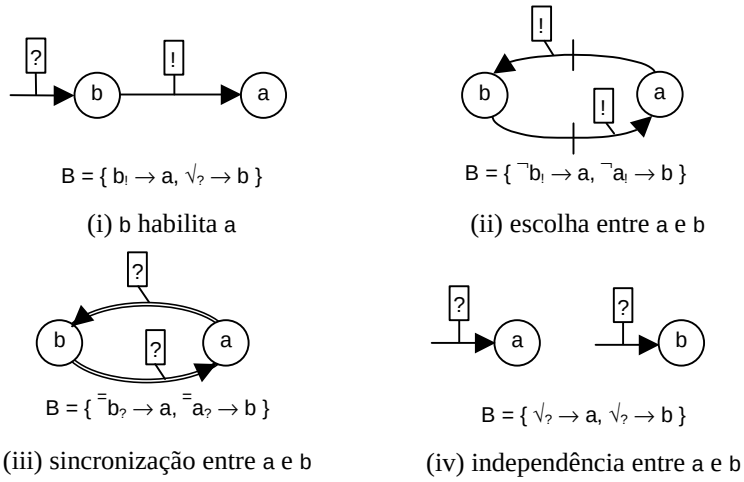


Figura 6. Exemplos de relações entre ações.

Na Figura 6 (i), b é uma ação inicial, e a condição de habilitação de a é representada por uma seta sólida de b até a. A Figura 6 (ii) modela uma desabilitação mútua (escolha), ou seja, a ocorrência de a desabilita a ocorrência de b, e vice-versa. A desabilitação de a por b (b por a) é representada por uma seta sólida de b até a (a até b), entrecortada por uma barra vertical. A Figura 6 (iii) modela a sincronização entre a e b. A condição de sincronização $=b$ ($=a$) da ação a (b) é representada por um seta sólida de linhas duplas de b até a (a até b). A Figura 6 (iv) modela a independência entre as ações a e b, a qual é representada implicitamente através da ausência de relações entre estas ações.

3.3. Conjunção de condições de causalidade

Em alguns comportamentos, múltiplas condições básicas envolvendo ações diferentes devem ser satisfeitas para que uma determinada ação possa ocorrer. Este tipo de situação pode ser representado através da conjunção de duas ou mais condições de habilitação, desabilitação ou sincronização envolvendo diferentes ações. O operador $\wedge$ (conjunção) permite a representação

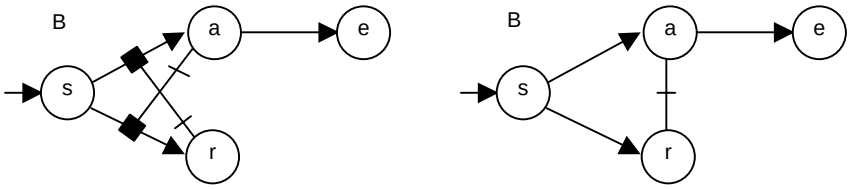
da conjunção de duas condições de causalidade. A conjunção de múltiplas condições pode ser representada através da aplicação sucessiva deste operador.

A conjunção de duas condições de causalidade γ_1 e γ_2 de uma ação resultante a é definida por $\gamma = \gamma_1 \wedge \gamma_2$, de tal modo que ambas as condições γ_1 e γ_2 são condições necessárias para a ocorrência de a. Dessa forma, a somente pode ocorrer caso ambas γ_1 e γ_2 sejam satisfeitas durante a execução de um comportamento.

Alguns exemplos de conjunção de condições são:

- $\gamma = b \wedge c$, na qual ambas as ações b e c têm de ocorrer antes que a ação a possa ocorrer;
- $\gamma = b \wedge \neg c$, na qual a ação a pode ocorrer após a ocorrência de b, e caso c não ocorra antes ou ao mesmo tempo que a;
- $\gamma = b \wedge \neg c \wedge \overline{=} d$, na qual a ação a somente pode ocorrer simultaneamente com d, após a ocorrência de b e caso c não ocorra antes ou ao mesmo tempo que a.

A Figura 7 mostra um exemplo de conjunção de condições de habilitação e desabilitação. Nessa figura a ação s modela a solicitação de um produto. Depois da ocorrência da ação s, uma escolha é feita entre a ocorrência da ação a ou a ocorrência da ação r, que modelam a aceitação e a recusa da solicitação, respectivamente. A ocorrência de a exclui a ocorrência de r, e vice-versa. Depois da ocorrência da ação a, a ação e, que modela a entrega do produto solicitado, pode ocorrer. O símbolo ■ representa o operador de conjunção na nossa notação gráfica.



(i) notação original (ii) notação simplificada

Figura 7. Conjunção de condições de habilitação e desabilitação.

O comportamento da Figura 7 pode ser representado na nossa notação textual como $B = \{\surd \rightarrow s, s \wedge \neg r \rightarrow a, s \wedge \neg a \rightarrow r, a \rightarrow e\}$. A Figura 7 (ii) apresenta uma notação gráfica simplificada para a relação de escolha entre as ações a e r. Neste texto assumimos que quando o valor do atributo de incerteza é omitido, o valor deve é associado à condição como 'default'.

3.4. Disjunção de condições de causalidade

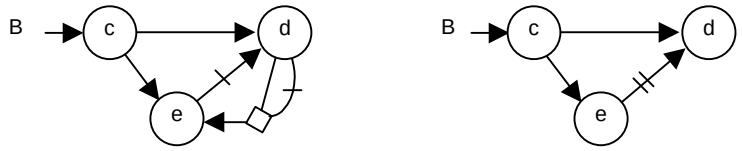
Em alguns comportamentos pelo menos uma condição básica ou uma conjunção de condições deve ser satisfeita para que uma determinada ação possa ocorrer. Este tipo de situação pode ser representado através da disjunção de duas ou mais condições de causalidade básicas ou conjunções de condições. O operador $\vee$ (disjunção) permite a representação da disjunção de duas condições, sendo que estas podem ser condições básicas, conjunções ou disjunções. Estas condições são chamadas de *condições de causalidade alternativas*. A disjunção de múltiplas condições pode ser representada aplicando-se sucessivamente o operador $\vee$.

A disjunção de duas condições de causalidade Γ_1 e Γ_2 de uma ação resultante a é definida por $\Gamma = \Gamma_1 \vee \Gamma_2$, de tal modo que as condições de causalidade alternativas Γ_1 e Γ_2 são condições mínimas e suficientes para a ocorrência da ação resultante a. Isso implica que a pode ocorrer caso pelo menos uma destas condições sejam satisfeitas durante a execução de um comportamento.

Em uma execução de comportamento, a ocorrência de uma ação resultante é causada por somente uma das suas condições de causalidade alternativas. Desse modo optamos por uma *interpretação de ou-exclusivo* para a causa efetiva da ocorrência de uma ação. No nosso modelo, supomos que características específicas da implementação do comportamento determinam qual condição efetivamente causa a ocorrência da ação resultante, caso múltiplas condições de causalidade alternativas sejam satisfeitas durante a execução de um comportamento. Assim, abstraímo-nos de características de implementações, de tal modo que a escolha de condições efetivas, do ponto de vista abstrato do modelo, é aleatória.

A interpretação de ou-exclusivo implica que todas as possíveis alternativas para a ocorrência da ação resultante devem ser definidas explicitamente. A condição de causalidade alternativa que causa a ocorrência da ação resultante em uma execução é chamada de *condição de causalidade resultante*.

A Figura 8 mostra um exemplo de desabilitação entre duas ações, definida usando disjunção de condições. Nessa figura a ação c modela o estabelecimento de uma conexão, a ação d modela a transferência de dados e a ação e modela o encerramento da conexão. O símbolo □ representa o operador de disjunção na notação gráfica.



(i) notação original (ii) notação simplificada

Figura 8. Exemplo de relação de desabilitação entre duas ações.

O comportamento da Figura 8 pode ser representado na nossa notação textual como B = {√ → c, c ∧ ¬e → d, (c ∧¬d) ∨ (c ∧ d) → e}. A condição c ∧ ¬e define que d pode ocorrer depois da ação c, desde que a ação e não ocorra antes ou simultaneamente com d. Consequentemente, a recíproca é válida, ou seja, d não pode ocorrer simultaneamente com e. Assim, a condição ¬d ∨ d, implicitamente presente na condição de causalidade (c ∧¬d) ∨ (c ∧ d), define que e não pode ocorrer simultaneamente com d. Quando d ocorre, a ocorrência de e é desabilitada no momento τ_d, mas torna-se habilitada novamente depois de τ_d graças à condição c ∧ d. Dessa forma, a ocorrência de e em uma execução é efetivamente causada pela condição c ∧¬d ou pela condição c ∧ d, a qual nesse caso fica sendo a condição de causalidade resultante de e nesta execução. A Figura 8 (ii) apresenta a notação gráfica simplificada para a relação de desabilitação entre as ações e e d.

Atributo de incerteza. Em uma relação de causalidade um valor de incerteza deve ser associado a cada condição de causalidade alternativa. Por exemplo, na relação de causalidade (¬b ∧ ¬c) ∨ (b ∧ ¬d) → a, devemos definir duas associações de incerteza υ_a(¬b ∧ ¬c) e υ_a(b ∧ ¬d).

A Figura 9 apresenta algumas relações entre duas ações a e b que são freqüentemente encontradas na prática, juntamente com algumas regras para a definição de atributos de incerteza. Comportamentos que consistem de múltiplas ações podem ser definidos através da conjunção e disjunção destas relações.

4. Estruturação de comportamentos

Um comportamento complexo pode ser melhor entendido se o mesmo for definido em termos de uma estrutura que consiste de uma composição de comportamentos mais simples. Além disso, essa estruturação de comportamentos permite a reutilização de definições de comportamento e facilita o refinamento [9].

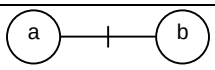
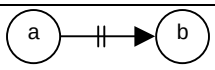
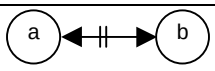
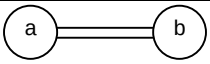
Relação entre ações	Representação textual	Representação gráfica	Regra de consistência
a habilita b	$\{ \sqrt{}\mu1 \rightarrow a,$ $a_{\mu2} \rightarrow b \}$		
Escolha	$\{ \neg b_{\mu1} \rightarrow a,$ $\neg a_{\mu2} \rightarrow b \}$		$\mu1 = \mu2$
a desabilita b	$\{ \neg b_{\mu1} \vee b_{\mu3} \rightarrow a,$ $\neg a_{\mu2} \rightarrow b\}$		$\mu1 = \mu2$
Entrelaçamento	$\{ \neg b_{\mu1} \vee b_{\mu3} \rightarrow a,$ $\neg a_{\mu2} \vee a_{\mu4} \rightarrow b\}$		$\mu1 = \mu2$
Sincronização	$\{ =b_{\mu1} \rightarrow a,$ $=a_{\mu2} \rightarrow b\}$		$\mu1 = \mu2$

Figura 9. Relações comuns entre ações.

Existem duas técnicas básicas para a estruturação de comportamentos no nosso modelo: a estruturação usando causalidade entre comportamentos e a estruturação usando restrições aplicadas a ações comuns a comportamentos. Comportamentos podem ser estruturados de forma arbitrária através do uso combinado dessas duas técnicas.

4.1. Estruturação usando causalidade

A estruturação usando causalidade é baseada na decomposição de uma relação de causalidade, de tal modo que uma ação e a condição para a sua ocorrência são definidas em comportamentos distintos. Os seguintes elementos sintáticos foram introduzidos para possibilitar essa estruturação: *ponto de entrada* (entry), que consiste de um ponto em um comportamento a partir do qual ações deste comportamento podem ser habilitadas através de condições envolvendo ações de outros comportamentos; e *ponto de saída* (exit), que consiste em uma condição de causalidade em um comportamento que pode ser utilizado para habilitar ações em um outro comportamento.

A Figura 10 ilustra a decomposição da relação de causalidade $b \wedge c \rightarrow a$ nas pseudo-relações de causalidade $b \wedge c \rightarrow$ exit e entry $\rightarrow a$, as quais são definidas nos comportamentos B1 e B2, respectivamente. O termo pseudo-relação é utilizado para ressaltar que pontos de entrada e de saída não são ações pois eles não modelam atividades. Um ponto de entrada e um ponto de saída são representados pelo símbolo $\triangleright$ na nossa notação gráfica.

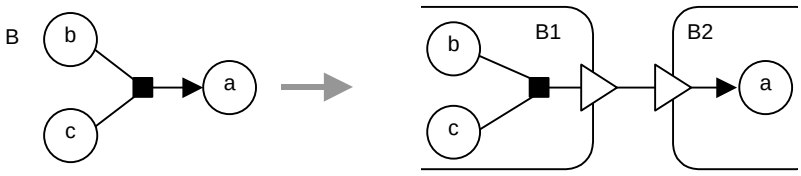


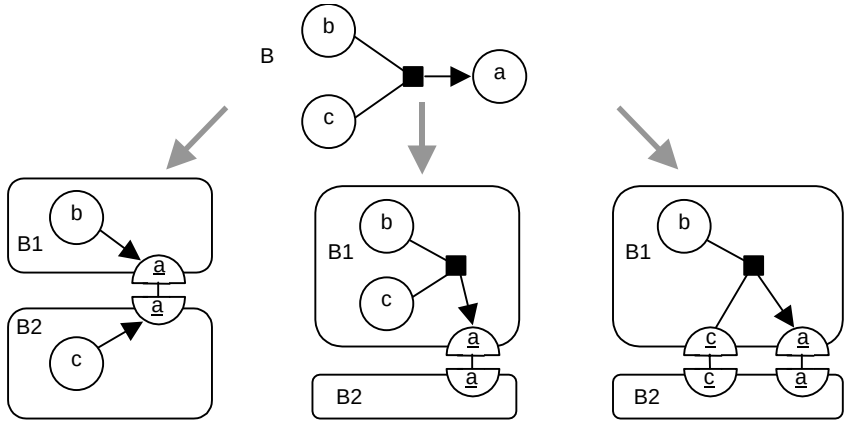
Figura 10. Decomposição da relação de causalidade $b \wedge c \rightarrow a$.

4.2. Estruturação usando restrições

Na seção 2.4 mostramos que uma ação pode ser considerada como uma interação integrada, ou vice-versa, de tal forma que diferentes contribuições para uma interação podem representar uma ação de forma distribuída. A estruturação de um comportamento usando restrições é análoga à estruturação de uma ação usando diferentes contribuições. Nessa estruturação um comportamento é definido em termos uma composição de comportamentos mais simples com

ações comuns, onde cada comportamento contribui de uma determinada forma para a ocorrência dessas ações.

A Figura 11 mostra três alternativas para a estruturação da relação de causalidade $b \wedge c \rightarrow a$, obtidas através de diferentes distribuições das condições b e c.



(i) b e c em comportamentos distintos

(ii) b e c no mesmo comportamento

(iii) c distribuída (comum a B1 e B2)

Figura 11. Alternativas para a estruturação da relação $b \wedge c \rightarrow a$ usando restrições.

Na Figura 11 (i) a ação b se encontra em B1 enquanto a ação c se encontra em B2. Uma ação distribuída somente pode ocorrer caso todas as contribuições dessa ação em todos os comportamentos aos quais essa ação é comum sejam satisfeitas. Nesse exemplo, a ação distribuída a somente pode ocorrer depois da ocorrência de ambas as ações b e c, de tal modo que a condição para a ocorrência da ação integrada a também se aplica à ação distribuída a. A Figura 11 (ii) mostra uma estruturação onde b e c se encontram em B1 (B2 não pode restringir a ocorrência de a) e a Figura 11 (iii) mostra uma estruturação onde b se encontra em B1 e c é comum a B1 e B2.

Uma análise exaustiva das decomposições de relações de causalidade, tanto conjuntivas como disjuntivas, inclusive para ações comuns a mais de um comportamento, pode ser encontrada em [4]. Neste artigo, somente introduzimos os objetivos e requisitos dessas decomposições.

5. Aplicações

As aplicações de AMBER tem sido principalmente nas áreas de reengenharia de processos empresariais e desenvolvimento de sistemas telemáticos. Essas aplicações e o suporte de ferramentas existente são discutidos a seguir.

5.1. Reengenharia de processos empresariais

No âmbito do projeto Testbed [14] AMBER tem sido utilizada na reengenharia de processos empresarias. O processo de reengenharia começa com a especificação de um processo empresarial em seu estado atual. A partir dessa especificação, o arquiteto de processo empresariais pode utilizar as regras de abstração desenvolvidas em [4,9] para determinar o serviço oferecido pelo processo que está sendo re-projetado, ou seja, para determinar o que os usuários desse processo observam.

O passo seguinte é a análise do serviço atual e a sugestão de melhorias no mesmo. A partir da definição do novo serviço, e usando a situação interna atual do processo, uma nova estrutura

interna do processo é definida e implantada [5]. A Figura 12 ilustra os passos da reengenharia de processos empresariais.

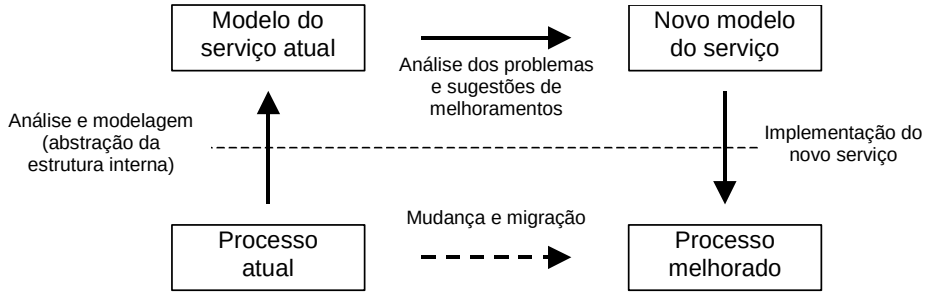


Figura 12. Reengenharia de processos empresariais.

Um outro aspecto da aplicação de AMBER no escopo de processos empresariais é o mapeamento de modelos AMBER em modelos de 'workflow' que podem ser implementados por 'workflow engines'. Nesse aspecto o modelo arquitetônico de AMBER se mostrou mais geral do que os modelos de workflow, de tal modo que usando um sub-conjunto de AMBER pôde-se criar modelos que podem ser diretamente traduzidos para modelos de workflow.

5.2. Desenvolvimento de serviços telemáticos

No âmbito do projeto Friends [6] AMBER tem sido utilizada no projeto de serviços telemáticos. AMBER é utilizada na modelagem completa do comportamento externo de um componente de software, definindo-se tanto as operações que podem ser invocadas na interface do componente quanto as operações invocadas pelo mesmo em outros componentes, assim como o relacionamento entre estas operações e os seus parâmetros.

AMBER também é utilizada no teste da implementação do serviço modelado. Dentro do projeto Friends foi também desenvolvida uma aplicação que monitora a execução dos comportamentos dos componentes que provêem o serviço modelado e gera um modelo AMBER contendo a ordem de execução das operações invocadas e a relação causal entre as mesmas [3]. Desta forma, pode-se comparar o modelo da especificação com o modelo da execução e verificar se a implementação corresponde ao serviço modelado.

5.3. Ferramentas de suporte

O desenvolvimento de modelos utilizando AMBER é suportado por um conjunto integrado de ferramentas chamado Testbed Studio. O Testbed Studio foi desenvolvido na âmbito do projeto Testbed e possui ferramentas para a simulação passo-a-passo e a análise funcional de modelos comportamentais, incluindo as seguintes operações:

- trilha, i.e., verificação se uma dada sequência de ações é sempre executada, nunca executada ou eventualmente executada;
- vivacidade, i.e., verificação se a invocação de um dado conjunto de ações resulta na invocação de pelo menos uma ou de todas as ações de um outro conjunto;
- ocorrência combinada, i.e., verificação se a invocação das ações de um conjunto ou desabilita a invocação das demais ações pertencentes ao conjunto ou todas as ações pertencentes ao conjunto são sempre, algumas vezes ou nunca invocadas;
- garantia, i.e., verificação se a invocação de pelo menos uma ação ou todas as ações de um dado conjunto exige a invocação de pelo menos uma ação ou de todas a ações de um outro conjunto.

Além disso, o Testbed Studio também oferece um editor e uma biblioteca de modelos que facilitam o desenvolvimento de modelos e a reutilização dos mesmos. A Figura 13 ilustra a simulação de um comportamento usando o Testbed Studio.

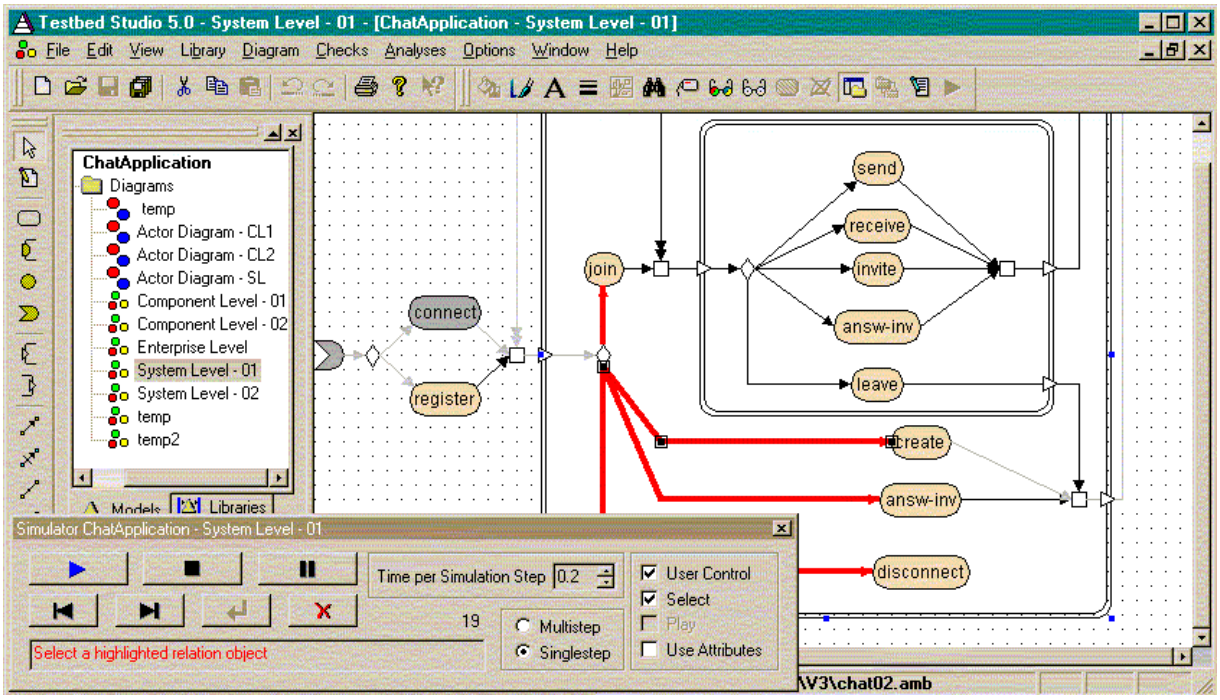


Figura 13. Simulador de comportamentos do Testbed Studio.

6. Conclusões

Neste artigo introduzimos uma linguagem para a modelagem de sistemas distribuídos e seu modelo arquitetônico abstrato. Devido a complexidade do modelo, restringimo-nos a discutir algumas de suas características, sem a pretensão de sermos completos.

A partir do ano letivo de 1997/1998 na Universidade de Twente nós expusemos os nossos alunos do curso de Projeto de Sistemas Telemáticos (‘The Design of Distributed Systems’, [16]) a essa linguagem, em substituição à TDF LOTOS que era utilizada anteriormente no curso. Os resultados dos exercícios e a reação dos estudantes, especialmente daqueles que repetiram o curso, foram muito satisfatórias. Um dos problemas do uso de LOTOS era que os alunos tinham a sua atenção desviada para as ferramentas e especificações, esquecendo-se dos conceitos e aspectos arquitetônicos que formam o núcleo desse curso. Esse problema foi completamente sanado através desta experiência. Desse modo não só continuamos a apresentar o curso dessa forma nos anos seguintes, como estamos editando o material do nosso curso em um livro, que deve aparecer em 2001.

Dentro do projeto ProTeM-CC Fase III Design de Aplicações Multimídia Distribuídas (DAMD) [8], foi também investigado o uso de AMBER na modelagem abstrata de serviços multimídia, incluindo um serviço de vídeo sob demanda [2] e de uma videoconferência [13].

Embora não tenhamos mencionado neste artigo, o modelo apresentado aqui foi desenvolvido tendo em mente a necessidade de se refinar modelos de um sistema, para oferecer suporte a implementação. Técnicas para o refinamento de comportamentos e ações foram desenvolvidas em [4,9]. Nesse aspecto o nosso modelo apresenta um alto grau de flexibilidade, permitindo que em um passo de refinamento ações sejam substituídas por estruturas de ações, e que a decomposição de ações seja realizada a partir de seus atributos.

Agradecimentos

Muitas das idéias e exemplos deste artigo foram desenvolvidas dentro do 'Architecture group' do CTIT, Universidade de Twente. De modo especial as contribuições de Chris Vissers, Marten van Sinderen e Dick Quartel na maturação dessas idéias devem ser mencionadas. Cléver Ricardo Guareis de Farias é bolsista de doutorado financiado pelo CNPq sob o processo número 200003/97-6.

Referências

1. T. Bolognesi, J. van de Lagemaat, and C.A. Vissers, editors. *LOTOSphere: software development with LOTOS*. Kluwer Academic Publishers, the Netherlands, 1995.
2. A. R. Camolesi. *Uma Metodologia para o Design de Serviços de TV-Interativa*. Dissertação de Mestrado, Programa de Pós Graduação em Ciência da Computação (PPG-CC), Universidade Federal de São Carlos (UFSCar), 2000.
3. N. Diakov, H. Batteram, H. Zandbelt and M. van Sinderen. Design and implementation of a framework for monitoring distributed component interactions. In *Proceedings of the 7th International Conference on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS 2000)*, LNCS 1905, Springer, pp. 227-240, 2000.
4. L. Ferreira Pires. *Architectural notes: a framework for distributed systems development.* Ph.D. Thesis. University of Twente, Enschede, the Netherlands, 1994.
5. H.M. Franken, M.K. de Weger, D.A.C. Quartel and L. Ferreira Pires. On engineering support for business process modelling and redesign. In *Proceedings of the International Workshop on Modelling Techniques, Business Process Re-Engineering and Benchmarking*, pp. 81-98, Bordeaux, France, April 1996.
6. FRIENDS project: http://friends.gigaport.nl/.
7. ISO. *LOTOS: a formal description technique based on the temporal ordering of observational behaviour*. IS 8807, 1989.
8. Projeto DAMD: http://www.dc.ufscar.br/pesquisa/damd/.
9. D. Quartel. *Action relations: basic design concepts for behaviour modelling and refinement.* Ph.D. Thesis. University of Twente, Enschede, the Netherlands, 1998.
10. D.A.C. Quartel, L. Ferreira Pires, M.J. van Sinderen, H.M. Franken, and C.A. Vissers. On the role of basic design concepts in behaviour structuring. *Computer Networks and ISDN Systems*, 29, pp. 413-436, 1997.
11. M. van Sinderen. *On the design of application protocols.* Ph.D. Thesis. University of Twente, Enschede, the Netherlands, 1995.
12. M. van Sinderen, L. Ferreira Pires, C. A. Vissers, and J.-P. Katoen. A design model for open distributed processing systems. *Computer Networks and ISDN Systems*, 27, pp. 1263-1285, 1995.
13. M. A. Teixeira. *Modelagem de Conceitos Arquitetônicos para a Especificação de Aplicações Multimídia Distribuídas*. Dissertação de Mestrado, Programa de Pós Graduação em Ciência da Computação (PPG-CC), Universidade Federal de São Carlos (UFSCar), 1999.
14. Testbed project: http://www2.telin.nl/testbed/.
15. C.A. Vissers, M. van Sinderen, and L. Ferreira Pires. What makes industries believe in formal methods. In A. Danthine, G. Leduc, and P. Wolper (editors), *Protocol Specification, Testing, and Verification XIII*, pp. 3-26, Elsevier Science Publishers B.V. (North-Holland), the Netherlands, 1993.
16. C.A. Vissers, L. Ferreira Pires, D.A.C. Quartel and M.J. van Sinderen. *The architectural design of distributed systems*. Lecture notes for The design of telematic systems. University of Twente, Enschede, the Netherlands, March 2001.