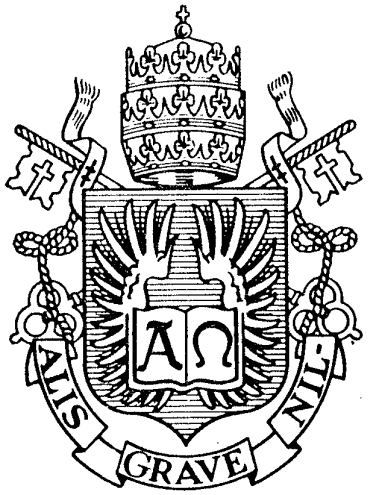

PUC

Série : Monografias em Ciência da Computação
No.4/89

FORMALIZAÇÃO DA ABORDAGEM ORIENTADA A OBJETOS EM REDES DE PETRI

Wagner Teixeira da Silva
Gernot Richter

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

A MARQUÊS DE SÃO VICENTE, 225 – CEP 22453

RIO DE JANEIRO – BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série : Monografia em Ciências da Computação, No. 4/89.
Editor : Paulo A. S. Veloso Abril, 1989.

FORMALIZAÇÃO DA ABORDAGEM ORIENTADA A OBJETOS EM REDES DE PETRI *

Wagner Teixeira da Silva

Gernot Richter

* Trabalho apresentado como Qualificação ao Prof. Gernot Richter (GMD-PUC/Rio)

Para obter cópias :
Rosane T. L. Castilho
Assessoria de Biblioteca, Documentação e Informação
Rua Marquês de São Vicente, 225 - Gávea
22.453 - Rio de Janeiro, RJ.
Brasil

SUMÁRIO

RESUMO

O paradigma de orientação a objetos vem ganhando repercussão como norteador na concepção e desenvolvimento de sistemas. Entretanto, a terminologia e os conceitos do enfoque não estão devidamente padronizados, o que dificulta a leitura e comparação do número explosivo de propostas que estão surgindo na área. Este trabalho propõe uma formalização dos principais conceitos de linguagens orientadas a objetos. Para tanto, usa-se um tipo de rede de Petri de alto nível chamado Predicado/ Transição (Pr/T).

PALAVRAS-CHAVE: Rede predicado/transição, rede de Petri de alto nível, rede elementar, orientado a objeto, objeto, classe de objetos, exemplar de uma classe, instância de uma classe, concorrência, formalização de comportamento de sistemas.

ABSTRACT

The object-oriented paradigm is gaining increasing impact as a guiding conception for systems design and development. However, the lack of an appropriate standard terminology based on clearly defined concepts difficults comprehension and comparison of the growing number of proposals which refer to the object-oriented approach. This paper proposes a formalization of the main concepts of object-oriented languages in term of high-level Petri nets called Predicate/Transition (Pr/T) Nets.

KEY-WORDS: Predicate/transition net, high-level Petri net, elementary net, object-oriented, object, object class, instance of a class, concurrency, formalization of systems behaviour.

1. INTRODUÇÃO

1.1 CONCEITO DE OBJETO

Objeto é tudo aquilo que se apresenta a nossa vista ou espírito. A esta entidade podemos atribuir propriedades ou verificar propriedades que lhe são inerentes. Queremos ver objetos como entidades autônomas. Para cada objeto há um conjunto de possíveis estados internos (não visíveis de fora). Um objeto, em um dado instante, está em um destes estados internos. A cada objeto está associado um conjunto de operações. Um objeto pode mudar de um estado interno para outro unicamente através das suas próprias operacões. Nenhum objeto tem a capacidade de alterar, diretamente, o estado interno de outro objeto. Os objetos se comunicam através de mensagens que é a única forma de interação entre eles. Assim um objeto muda de estado em função do contexto externo dado pela comunidade de objetos a qual ele pertença.

Na figura 1 temos um exemplo que mostra, ainda informalmente, uma possível interação entre objetos. Lá, suponhamos, o objeto Vendedor solicita o estoque ao objeto Almo-pneu (almoxarifado de pneus) o que lhe responde 150 unidades. O mesmo Vendedor solicita ao objeto Almo-câmara o custo unitário. Almo-câmara então responde Cz$150,00. E ainda o Vendedor manda uma mensagem para Almo-roda informando-lhe ter vendido 2 unidades. O objeto Almo-roda, altera seu estado interno, dando baixa no seu estoque, e responde ao Vendedor OK.

1.2 CONCEITO DE CLASSE

Uma classe é um conjunto de pessoas ou coisas (concretas ou abstratas) com características comuns. E cada objeto será um exemplar de uma dada classe. Aqui classe será definida por um gabarito de dados e serviços. Um gabarito de serviço é a descrição de um procedimento. O conjunto de seviços gabaritado em uma particular classe caracteriza o comportamento possível de cada objeto exemplar dessa classe. Quando um objeto x, exemplar da classe **X**, é criado ele passa a ser uma entidade autônoma com memória local e procedimentos definidos conforme o gabarito da classe **X**. Agora estes procedimentos são específicos do objeto x e

só eles têm acesso a sua memória local.

Uma classe é, portanto, uma entidade genérica representada pelo gabarito que a define, ou seja, o gabarito da classe é quem dá as características de seus objetos exemplares (instâncias). Podemos dizer ainda, por abuso de expressão, que os objetos herdam as características da classe.

1.3 SISTEMA DE SUPORTE

Um sistema de suporte é, para nós, uma entidade que gerencia os objetos e realiza a comunicação na comunidade de objetos. Cabe a ele, a pedido, criar ou delir objetos. Sempre que o sistema de suporte recebe um pedido de criação, ele cria um objeto a imagem e semelhança do gabarito da classe solicitada no pedido e devolve uma mensagem para o solicitante dizendo que o atendeu. A partir deste instante o objeto criado já pode postar ou receber mensagens. Para suportar a comunicação entre objetos, o sistema de suporte faz o papel de correio. Coleta uma mensagem de um objeto remetente e se encarrega de entregá-la ao objeto destinatário.

Se o sistema de suporte recebe um pedido de deleção de um objeto e todas as condições para a deleção são verificadas, então o objeto será delido. E a partir do instante em que um objeto é delido ele deixa de existir para a comunidade de objetos.

O sistema de suporte, na literatura técnica, também é considerado como um objeto, a despeito de ele não ter um gabarito de uma classe que descreva suas características e nem ter sido criado por nenhum outro objeto.

1.4 REDES DE PETRI

Redes de Petri de alto nível, tipo Predicado/Transição (Pr/T), são grafos dirigidos e biparticionados com dois tipos de nós, denominados lugares e transições, e arcos ligando lugares a transições ou transições a lugares. Uma transição junto com os arcos significa uma conexão causal entre os lugares ligados à transição. Os arcos ao redor de um nó do tipo transição são denominados os ramos da conexão, que indicam a maneira da participação de um lugar em uma conexão: os arcos que ligam

lugares a transições representam ramos de entrada, e aqueles que ligam transições a lugares, ramos de saída, indicando assim os lugares de entrada e os lugares de saída de cada conexão. Um lugar pode participar em uma conexão de várias maneiras diferentes, neste trabalho usamos apenas duas. Uma rede Pr/T tem anotado, em seus lugares, arcos e transições, fórmulas e termos de uma linguagem de primeira ordem (linguagem de anotação).

Um termo anotado em um lugar designa um conjunto de entidades do universo considerado. Ele define o **domínio** do lugar. Os elementos do domínio definem as **marcas** que podem estar presentes ou ausentes em um estado do sistema, sendo que tal estado está representado pela **marcação** dos lugares da rede (um termo dentro de um lugar significa a presença da marca correspondente). Nas redes Pr/T, os lugares são interpretados como predicados de extensão variável, ou seja, predicados que não pertencem ao universo da linguagem de anotação. Nessa interpretação, a **presença** de uma marca significa que o predicado está verdadeiro para a entidade, a **ausência** de uma marca significa o contrário. Em outras palavras, a marcação atual de um lugar representa a extensão atual do predicado variável a ele associado.

Uma fórmula anotada em uma transição define, junto com os termos anotados nos seus arcos, um conjunto de alterações das marcações nos lugares conectados. Os termos designam conjuntos de entidades que são subconjuntos dos domínios desses lugares. No caso de conjuntos com cardinalidade fixa usa-se, frequentemente, variáveis para os elementos dos conjuntos e omite-se as chaves do termo de enumeração no arco: escreve-se, então, x,y em lugar de {x,y}, ou x em lugar de {x}. Determina-se as alterações definidas, através da **valorização** das variáveis (livres) constantes da fórmula e dos termos: Uma valorização das variáveis com entidades conforme os termos nos arcos, para a qual a fórmula é verdadeira, define uma **alteração**, indicando, nos lugares de entrada, as marcas que desaparecem (as marcas de entrada) e, nos lugares de saída, as marcas que aparecem (as marcas de saída) quando de uma ocorrência da alteração.

Neste trabalho vamos representar uma alteração pelo identificador da transição (p. ex. t_1, t_2, t_3) e pela valorização

daquelas variáveis, cujos valores não são definidos por uma função (p.ex. $r=r_1$, $m=m_1$). Assim, a alteração definida pela transição t_2 e os valores $r=r_1$, $m=m_2$ e outros, que resultam da aplicação de funções, será representada pela expressão $(t_2; r = r_1, m = m_1)$.

Uma alteração só pode ocorrer se ela estiver **habilitada**, ou seja, se todas as marcas de entrada estiverem presentes e todas as marcas de saída estiverem ausentes.

Uma **ocorrência** de uma ou várias alterações representa uma mudança de estado do sistema, alterando a marcação dos lugares de entrada ou saída da rede, ou seja, as extensões atuais dos predicados variáveis: as marcas nos lugares de entrada tornam-se ausentes, as marcas nos lugares de saída tornam-se presentes. Na interpretação Pr/T isso significa, que alguns predicados deixaram de estar verdadeiros ou passaram a estar verdadeiros para as entidades consideradas.

O comportamento completo de um sistema é dado por uma **marcação inicial**, que faz parte do modelo. O estado atual de um sistema é dado por uma marcação alcançável a partir da marcação inicial. Uma marcação é uma função variável que determina, para cada marca da rede, a sua atual presença ou ausência.

Uma introdução mais detalhada a redes tipo Pr/T pode ser achada em Genrich & Lautenbach(1981) e Genrich(1986); uma introdução mais didática é dada por Richter(1983) e Reisig(1985b); e ainda uma boa introdução em português está no trabalho de Heuser(1989).

1.5 REDES Pr/T ENCAIXADAS

Na presente aplicação das redes Pr/T, usamo-las de forma encaixada: o sistema de suporte, modelado por uma rede Pr/T, tem algumas entidades que são também redes Pr/T as quais modelam os **objetos**. As redes que modelam os objetos têm como entidades as mensagens (pedidos de serviços, respostas etc), bem como outras estruturas de informação para uso interno. O fragmento simplificado de uma rede maior, ilustrado pelas figuras 2 e 3, respectivamente o objeto (rede Pr/T com marcação inicial) e o fragmento da rede maior (interação entre a rede abrangente e a rede encaixada) visa ilustrar a idéia básica usada nesta abordagem de formalização. As funções ENTR e COLET, na figura 3, definem

transformações da rede encaixada (ENTR:de "sem" para "com" pedido, COLET:de "com" para "sem" resposta), e a função RESP retorna uma mensagem de resposta que está no lugar MsgSaida da rede encaixada (objeto).

1.6 RESUMO DAS PRÓXIMAS SEÇÕES

Após uma visão geral do tema que vamos abordar neste trabalho, dada pelo resumo, introduzimos, informalmente, conceitos básicos ligados ao paradigma de objetos e demos uma descrição sucinta do instrumento formal, redes de Petri de alto nível, que utilizamos para formalizar os principais conceitos da abordagem por objetos.

Na seção 2 modelamos em redes de Petri, tipo Pr/T, um sistema de suporte e um objeto exemplar de uma classe genérica. O sistema de suporte tem como função: (a) Criar objetos (b) Delir objetos e (c) Prover a comunicação entre os objetos. O objeto modelado exibe a interface com o sistema de suporte, sua memória local, de acesso privativo das operações do objeto, um conjunto de serviços oferecido pelo objeto, um serviço de recuperação de mensagens destinadas a outros serviços do próprio objeto e um serviço rejeitador de mensagens com pedidos de serviços não oferecidos.

Na seção 3 especificamos a título de exemplificação dois objetos: um objeto inversor, cuja função é receber sequências de elementos de outros objetos e imprimi-las em ordem inversa, e um objeto pilha. Vários destes objetos pilhas poderão ser usados pelo objeto inversor. A fim de facilitar a compreensão da dinâmica do objeto inversor, fizemos uma modelagem simplificada da interação deste objeto com os demais objetos que usam seus serviços ou que lhe prestam algum tipo de serviço. Esta última modelagem foi feita em rede de Petri elementar (uma marca por lugar, cuja presença é indicada por uma ficha), considerando apenas um objeto solicitante do serviço de inversão do objeto inversor.

Fazemos a conclusão na seção 4, reafirmando os conceitos utilizados neste trabalho e realçando a nossa preocupação essencialmente semântica, sem compromisso com aspectos de implementação.

Na seção 5 referimos a uma bibliografia introdutória a redes de Petri e a bibliografia referente a Programação Orientada a

Objetos, a qual teve maior vigor a partir do principal evento ocorrido nesta área: primeira conferência do grupo OOPSLA, em Portland, conhecida como OOPSLA'86. Os anais desta conferência foram publicados em SIGPLAN Notices.

2. MODELO EM REDES DE PETRI

Com a preocupação de dar uma semântica exata aos conceitos de linguagens de programação orientada a objetos é que nos propusemos a esta modelagem. Redes de Petri são muito adequadas para modelar sistemas concorrentes, e o paradigma de orientação a objetos induz naturalmente a concorrência, mas não a exige. Modelaremos primeiro o sistema de suporte que estabelece as relações entre os objetos e tem a capacidade de criar e delir objetos. Em seguida modelaremos um exemplar ilustrativo de objeto com sua memória local e o seu comportamento.

2.1 SISTEMA DE SUPORTE

Neste modelo o sistema de suporte tem quatro lugares e quatro conexões, t_1 a t_4. Na figura 4 temos a rede do modelo e nas tabelas 2.1 a 2.3 temos a descrição dos conjuntos de entidades que formam os domínios dos lugares, bem como dos predicados constantes e das funções que aparecem nas expressões inscritas nos lugares e nas conexões.

Na anotação dos lugares foi adotado a convenção tradicional nas redes Pr/T de juntar o termo que designa o domínio do lugar (p.ex. MSG) com o símbolo que designa a sua interpretação, ou seja, o predicado variável associado a ele (p.ex. MsgPostada( )). Assim MsgPostada(MSG) significa que para cada elemento x de MSG, presente no lugar, a sentença MsgPostada(x) é considerada verdadeira.

A rede do sistema de suporte tem uma marcação inicial μ_o dada pela tabela 2.4 e representada na figura 4. No lugar MsgPostadas não temos nenhuma mensagem, o lugar ObjetoVivo tem um objeto inicial O_o, o lugar ClasseDef contém o conjunto das descrições de todas as classes já definidas, e o lugar IdLivre contém o conjunto de todos os identificadores para os objetos que venham a ser

Tabela 2.1 Descrição dos domínios dos lugares

Conjunto	Descrição
CLASSE	As descrições de todas as classes. Cada descrição de classe é constituida de: . Identificador de classe . Gabarito de dados e serviços (rede Pr/T) . Registro dos objetos vivos que são exemplares da clase. . Outras informações pertinentes à classe (p.ex. número máximo de exemplares)
MSG	Todas as mensagens possíveis. Cada mensagem é constituida de: . Identificadores do objeto de orígem, de destino e do serviço. . Tipo de mensagem: pedido de serviço ou resposta. . Parâmetros (msg tipo pedido) ou resultado (msg tipo resposta)
OBJETO	Todos os exemplares possíveis das classes definidas. Cada objeto é constituido de: . Identificadores do objeto e da classe. . Memória local e procedimentos . Capacidade de comunicação (receber e enviar mensagens) . Estado atual
ID	Todos os identificadores de objetos possíveis

Tabela 2.2 Descrição dos predicados (constantes)

Predicado	Descrição
PedCriar (m)	É verdadeiro se a mensagem **m** for um pedido de criação de um novo objeto. É falso em caso contrário.
PedDelir (m)	É verdadeiro se a mensagem **m** for um pedido para eliminar um dado objeto. É falso em caso contrário.
LimObjetOk (c)	É verdadadeiro se a classe **c** ainda não atingiu o número máximo de exemplares (este limite pode ser aplicado a objetos físicos, tais como impressora, unidade de fita magnética etc.)

Tabela 2.3 Descrição das funções

Função	Descrição
CLASSPED (m)	Se a mensagem m for um pedido de criação, retorna o identificador da classe do objeto a ser criado.
COLETA (o)	Retorna um par formado pela mensagem m e pelo objeto o, sem a mensagem m no lugar MsgSaida (novo estado do objeto).
CRIAR (m,d)	Retorna um objeto o com identificador d. O objeto o tem dados e procedimentos gabaritados pela classe descrita em m e está no estado inicial.
ENTREGA (m,o)	Retorna o objeto o sem a mensagem m no lugar MsgEntrada (novo estado do objeto).
ID (o)	Retorna o identificador de objeto do objeto o.
IDCL (x)	Retorna o identificador de classe da descrição da classe x ou do objeto x.
IDOBJD (m)	Retorna o identificador do objeto de destino da mensagem m.
OBJCRIADO (c,d)	Retorna a descrição da classe c com o registro dos objetos exemplares vivos acrescido do identificador d.
OBJDELIDO (c,d)	Retorna a descrição da classe c com o registro dos objetos exemplares vivos decrescido do identificador d.
PAR (m)	Retorna o parâmetro da mensagem m.
RESPOSTA (m,d)	Retorna a mensagem de resposta da mensagem m com o identificador d (do objeto criado ou delido).

Tabela 2.4 Marcação inicial μ_o

Lugar	Entidades
MsgPostada	—
ObjetoVivo	O_o
ClasseDef	$C_o, C_1, \ldots, C_n$
IdLivre	$d_1, d_2, \ldots, d_k$

criados. Esta marcação é apenas uma das possíveis marcações iniciais (p.ex. poderíamos ter uma marcação inicial com mais de um objeto no lugar ObjetoVivo).

Lembramos que as entidades do domínio do lugar ObjetoVivo são objetos exemplares das classes definidadas. Estes objetos são modeiadas por redes Pr/T, as quais ficam encaixadas na rede do sistema de suporte, conforme comentamos no item 1.5.

Na marcação inicial μ_0, uma sequência possível de alterações, que poderão ficar habilitadas e ocorrer, são:

(1) $(t_4 : o=O_0)$

A ocorrência desta alteração faz aparecer uma mensagem no lugar MsgPostada e altera o estado do objeto O_0 no lugar ObjetoVivo.

Uma hipótese possível para sua ocorrência seria: o objeto O_0 manda uma mensagem m_1 ao sistema de suporte solicitando a criação de um objeto da classe C_1, gerando a marcação μ_1, conforme tabela 2.5.

(2) $(t_1 : m=m_1, d=d_1, c=C_1)$

Dado que m_1 é um pedido de criação de outro objeto, e as demais condições se verificam então esta alteração fica habilitada.

A ocorrência desta alteração faz desaparecer as marcas m_1 do lugar MsgPostada e d_1 do lugar IdLivre, alterar a descrição da classe C_1 no lugar ClasseDef, e aparecer o objeto O_1 no lugar ObjetoVivo.

A interpretação é: a vista da mensagem m_1, no lugar MsgPostada, o sistema de suporte cria O_1, e posta a mensagem m_2 com a resposta de criação para o objeto O_0. Isto gera a marcação μ_2, na tabela 2.6.

(3) $(t_3 : m=m_2, o=O_0)$

Dado que m_2 é uma resposta ao pedido de criação de um objeto e o objeto de orígem existe no lugar ObjetoVivo, então esta alteração fica habilitada.

A ocorrência desta alteração faz desaparecer a marca m_2 do lugar MsgPostada e alterar o objeto O_0 no lugar ObjetoVivo.

A interpretação: o sistema de suporte entrega a mensagem de resposta da criação do objeto O_1 ao objeto O_0, isto gera a

marcação μ_3, na tabela 2.7.

(4) $(t_4 ; o=O_o)$

A ocorrência desta alteração faz aparecer uma mensagem no lugar MsgPostada e altera o estado do objeto O_o no lugar ObjetoVivo.

Suponhamos que a mensagem seja m_3 na qual o objeto O_o solicita que objeto O_1 lhe preste algum serviço. O sistema de suporte coleta esta mensagem gerando a marcação μ_4, na tabela 2.8.

Tabela 2.5 Marcação μ_1

Lugar	Entidades
MsgPostada	m_1
ObjetoVivo	O_o (sem m_1)
ClasseDef	C_o, C_1, ..., C_n
IdLivre	$d_1, d_2, \ldots, d_k$

Tabela 2.6 Marcação μ_2

Lugar	Entidades
MsgPostada	m_2
ObjetoVivo	O_o, O_1
ClasseDef	C_o, C_1(com ID(O_1)), ..., C_n
IdLivre	d_2, d_3, ..., d_k

Tabela 2.7 Marcação μ_3

Lugar	Entidades
MsgPostada	—
ObjetoVivo	O_o(com m_2), O_1
ClasseDef	C_o, C_1, ..., C_n
IdLivre	d_2, d_3, ..., d_k

Tabela 2.8 Marcação μ_4

Lugar	Entidades
MsgPostada	m_3
ObjetoVivo	O_o(sem m_3), O_1
ClasseDef	C_o, C_1, ..., C_n
IdLivre	d_2, ..., d_k

O objeto O_0 na tabela 2.5 tem um estado diferente em relação à tabela 2.4. Ele tinha a mensagem m_1 no lugar MsgSaida e o sistema de suporte coletou-a, deixando-o com a ausência de m_1. Possivelmente outros lugares internos ao objeto O_0, também, tenham sido alterados. O mesmo aconteceu com a descrição da classe C_1, da tabela 2.5 para a tabela 2.6, com a criação do objeto O_1, que passou a ter o identificador do objeto O_1 constando no seu registro de objetos vivos. Contudo, só indicamos informalmente estes estados diferentes de objetos ou descrições de classe. O leitor, porém, fica ciente do fato. Fica implícito que quando um objeto recebe ou envia uma mensagem, ele muda de estado. E quando um objeto é criado ou delido, a descrição de classe daquele objeto, também, sofre mudança de estado. Observamos que o **estado interno** de um objeto não é alterável pelo sistema de suporte. O sistema de suporte tem acesso apenas à interface com o objeto (representada pelos lugares MsgEntrada e MsgSaida).

2.2 OBJETOS EXEMPLARES

Um objeto exemplar (também: objeto instância) de uma classe é uma entidade autônoma, dispondo de memória local, procedimentos e interface de comunicação com o sistema de suporte. Em uma comunidade de objetos a única forma de comunicação entre eles é via mensagens. Um objeto pode solicitar ao sistema de suporte a criação ou deleção de outros objetos. O sistema de suporte é o único com capacidade de criar ou delir objetos. Um objeto pode solicitar que outro objeto lhe preste algum serviço. Este último se tiver dentre os seus procedimentos a capacidade de realizar o serviço pedido então ele o executa e devolve uma mensagem com o resultado ao objeto solicitante. Em caso contrário ele também devolve uma mensagem com a negativa da execução. Cada objeto tem identificação única. O identificador de cada objeto criado fica registrado na descrição da classe e é eliminado de lá quando ele é delido. Algumas classes especiais podem ter um limite de exemplares vivos. Este limite pode ser controlado pela quantidade de identificadores de objetos registrados na descrição da classe.

De um modo geral temos: se um objeto O_1 solicita um serviço ao objeto O_2, então o objeto O_2 ativa o seu procedimento relacionado

ao serviço pedido. Este procedimento para realizar o serviço pedido pode se socorrer dos serviços de outros objetos. Quando o objeto O_2 receber os resultados dos serviços que ele próprio pediu a outros objetos, ele devolve o resultado do serviço que o objeto O_1 solicitou-lhe. A rede da figura 5 mostra uma estrutura básica possível de um objeto exemplar de uma classe. As tabelas 2.8 e 2.9 relacionam as funções e predicados (constantes) adicionais usados na anotação da rede da figura 5.

O domínio do lugar MemoriaLocal, veja figura 5, é dado pelo conjunto DADOS, que é o conjunto de todas entidades especificadas no gabarito de dados da descrição da classe do objeto.

Tabela 2.8 Descrição dos predicados (constantes)

Predicado	Descrição
NeServ (m)	É verdadeiro se a mensagem **m** especificar um serviço que não seja previsto pelo objeto
$Serv_x(a,b,d,e)$	É verdadeiro se os parâmetros **a,b,c** e **d** forem compatíveis com o serviço **x**.

Tabela 2.9 Descrição das funções

Função	Descrição
MYSELF	Retorna o identificador do próprio objeto.
ERRO (m)	Retorna a mensagem de erro de serviço inexistente.
SERVICO (m)	Retorna a identificação do serviço contido na mensagem **m**.

Um objeto exemplar de uma classe pode ou não ter uma marcação inicial. Na rede da figura 5, nos estamos supondo que tal marcação exista.

Considerando uma classe especificada por uma rede gabarito, um objeto exemplar desta classe tem a mesma rede. Se um objeto, uma vez criado, tem a rede com uma marcação inicial, então é possível que haja alterações habilitadas, na ausência de mensagens no lugar MsgEntrada, e que elas ocorram mudando o estado interno inicial do objeto, ocorrendo até envio de mensagens para outros objetos. Em

caso contrário o objeto fica inativo aguardando a chegada de mensagens de outros objetos quando então ele ativa os seus procedimentos para atender as solicitações que chegaram.

A rede representada pela figura 5 é um esquema de objeto.Assim sua rede tem três lugares e k+2 conexões. Os lugares denominados MsgEntrada e MsgSaida formam a interface com o sistema de suporte. Toda mensagem que o objeto recebe é colocada pelo sistema de suporte ou pela transição t_o, do próprio objeto, no lugar MsgEntrada. E toda mensagem que o objeto manda para objetos é colocada no lugar MsgSaida e o sistema de suporte a tira dali e a entrega ao objeto de destino, se este for diferente do objeto de orígem. Uma vez que o objeto pode mandar uma mensagem para si, em situações em que um serviço deste objeto requer um outro serviço do próprio objeto, a transição t_o recupera estas mensagens de MsgSaida e as coloca no lugar MsgEntrada, evitando assim uma participação desnecessária do sistema de suporte. O lugar MemóriaLocal neste esquema de rede está representando a área de dados gabaritada pelo gabarito da classe do objeto e a memória local de cada procedimento. Os procedimentos de t_1 a t_k caracterizam os serviços que o objeto pode oferecer. A transição t_{k+1} especifica a rejeição de pedidos de serviços que o objeto não prevê. Assim um objeto ao receber uma solicitação de serviço verifica se ele o oferece e se todas as demais condições estão atendidas (p.ex. se uma das fórmulas, correspondente ao serviço pedido, $Serv_x(m, m', d, d')$, para x de 1 até k, anotadas nas conexões da rede da figura 5, é verdadeira). Em caso afirmativo o serviço é iniciado. Caso contrário o procedimento t_{k+1} manda uma mensagem ao objeto de origem rejeitando o serviço.

Devemos salientar o potencial paralelismo entre os procedimentos de um dado objeto. Aqui temos basicamente dois casos:

- Suponha que existam várias mensagens para o mesmo procedimento. Desde que o procedimento, a vista das mensagens, não use dados comuns para atende-las, então é possível que todas as mensagens sejam atendidas concorrentemente.
- Suponha que existam várias mensagens para procedimentos

diferentes. Desde que os procedimentos, a vista das mensagens, não usem dados comuns, então todas as mensagens podem ser atendidas concorrentemente.

O paralelismo entre objetos diferentes se dá de forma muito natural. Como cada objeto tem memória privativa não há nenhum impedimento para que vários objetos simultaneamente possam estar realizando suas tarefas. É claro que neste caso cada objeto deve ter seus próprios mecanismos de controle para garantir sincronismo, onde ele for necessário.

A recursão no mundo dos objetos é também muito natural. Em uma visão sequencial podemos ter um encadeamento de mensagens do objeto O_1 para o objeto O_2 e este para o objeto O_3, etc, e do objeto O_{k-1} para o objeto O_k. As mensagens podem ser solicitações de serviços. Os resultados dos serviços realizados são comunicados na ordem inversa . Se todos os objetos forem exemplares de uma mesma classe, isto é, todos tiverem os mesmos procedimentos mas sobre dados distintos e os serviços solicitados envolverem um procedimento comum aos objetos, então temos a clássica recursão: um procedimento invocando a si mesmo com uma condição de contorno. Os demais tipos de recursão também são trivialmente verificados.

3. MODELO EXEMPLO

A guisa de exemplificação especificaremos em rede de Petri o comportamento de um objeto cujo objetivo é inverter sequências (não vazias) de itens recebidas de outros objetos e imprimi-las. Objetos exemplares desta classe "Inversor" fazem uso de objetos exemplares da classe "Pilha" e da classe "Impressora". A especifição semântica dos dois primeiros objetos é dada pelo modelo em rede definido pelas figuras 6 e 7, objeto inversor e objeto pilha, respectivamente.

3.1 MODELO ELEMENTAR

De modo a tornar mais fácil a compreensão da dinâmica do objeto inversor (modelo da figura 6), na sua interação com o sistema de suporte, com os objetos solicitantes dos seus serviços e com os demais objetos de apoio, isto é, aqueles objetos que prestam

serviços ao objeto inversor, nós modelamos em rede de Petri elementar (figura 8) o comportamento do objeto inversor, levando em conta apenas um único solicitante do serviço de inversão. Este modelo é todo sequencial e o leitor não terá (esperamos) muita dificuldade em acompanhar os eventos que ocorrem a partir da solicitação de um serviço de inversão ao objeto inversor.

O processo começa (veja figura 8) quando o objeto solicitante envia um pedido de inversão ao objeto inversor. Este recebe o pedido (conexão x_1) e manda uma mensagem para o sistema de suporte pedindo a criação de um objeto pilha. Quando o objeto inversor recebe a resposta da criação do objeto pilha ele autoriza o solicitante a mandar o primeiro item da sequência a ser invertida (conexão x_2). E enquanto o solicitante tiver item para inversão, o inversor recebe um item e manda uma mensagem para o objeto pilha pedindo o empilhamento deste item (conexão x_3). Após o objeto pilha realizar o seu serviço, o inversor recebe a resposta do empilhamento e autoriza o solicitante a mandar outro item (conexão x_4). Quando o solicitante informa fim de itens o inversor recebe esta mensagem e manda uma mensagem ao sistema de suporte pedindo a criação de um objeto impressor (conexão x_5). Quando o inversor recebe a resposta de criação do objeto impressor ele pede ao objeto pilha para desempilhar um item (conexão x_6). E agora enquanto houver itens na pilha o inversor recebe um item e pede ao objeto impressor para imprimi-lo (conexão x_7). Depois da impressão o inversor recebe a resposta do objeto impressor e manda uma mensagem para o objeto pilha pedindo para desempilhar outro item (conexão x_8). E assim por diante, até que a pilha fique vazia. Quando o objeto inversor recebe a resposta de "pilha vazia" do objeto pilha, ele manda uma mensagem para o solicitante informando o término do serviço e pede ao sistema de suporte para delir os objetos pilha e impressor (conexão x_9).

Para entendermos o objeto inversor completo, modelado na figura 6, podemos pensar nele, grosseiramente, como sendo modelado por várias redes daquelas descritas na figura 8, umas empilhadas sobre as outras. Assim cada rede, sendo independente uma das outras, representa um atendimento de um solicitante de serviço de inversão. Isto é, vários serviços de inversão poderiam estar sendo

concorrentemente prestados pelo objeto inversor.

Na modelagem do comportamento do objeto inversor, não exploramos a possível concorrência entre os objetos pilha e impressor ligados a um mesmo atendimento de um solicitante de serviço. Com isto ganhamos maior simplicidade no modelo (figura 6), ao preço de uma sequencialização não necessária.

3.2 EXEMPLAR DE INVERSOR

As tabelas 3.1 e 3.2 descrevem os predicados (constantes) e as funções da linguagem de anotação usada na rede do objeto inversor. Só descreveremos aquelas funções e predicados ainda não descritos na rede do sistema de suporte. Todos os domínios dos lugares já foram descritos no sistema de suporte. A interpretação dos lugares como predicados variáveis é dada na tabela 3.3.

A idéia do objeto inversor é pedir a criação de um objeto pilha (conexão t_1 e t_2) para cada objeto solicitante do serviço de inversão, receber uma sequência, item a item, de cada um dos objetos solicitantes do serviço (conexão t_3), e a medida que for recebendo os itens de cada objeto pedir a pilha do respectivo objeto (conexão t_3 e t_4) que empilhe aquele item. E quando um objeto solicitante informar o final de sua sequência (conexão t_5), então o objeto inversor solicita a criação de um objeto impressor (conexão t_5) ao sistema de suporte e guarda a identificação deste solicitante e da pilha a ele relacionada (lugar s_7). Após receber a confirmação de que foi criado um objeto impressor, o inversor associa um dos atendimentos constantes do lugar s_7 ao objeto impresssor criado (conexão t_6) e pede o desempilhamento do primeiro item ao objeto pilha relacionado ao atendimento (conexão t_6). Quando recebe a resposta de desempilhamento de um item (conexão t_7) o inversor identifica a pilha e o objeto impressor a ela relacionado e então pede a impressão do item recebido. Após receber a resposta do objeto impressor (conexão t_8), pede o desempilhamento de um novo item ao objeto pilha relacionado a aquele objeto impressor. O processo agora se repete com a conexão t_7 recebendo o resultado de desempilhamento e mandando uma mensagem para o objeto impressor, pedindo a impressão do item recebido, e a conexão t_8 recebendo o resultado da

Tabela 3.1 Descrição dos predicados (constantes)

Predicado	Descrição
PedInv (m)	É verdadeiro se m for uma mensagem pedindo serviço de inversão.
RespCrPil (m)	É verdadeiro se m for uma resposta de criação de um objeto pilha.
EnvItem (m)	É verdadeiro se m for uma mensagem enviando um item para o serviço de inversão.
FimItens (m)	É verdadeiro se m for uma mensagem informando fim dos itens para o serviço de inversão.
RespCrImp (m)	É verdadeiro se m for uma resposta ao pedido de criar um objeto impressor.
RespEmp (m)	É verdadeiro se m for uma resposta ao pedido de empilhar um item a um objeto pilha.
RespDesemp (m)	É verdadeiro se m for uma resposta ao pedido de desempilhar um item a um objeto pilha.
RespImp (m)	É verdadeiro se m for uma resposta ao pedido de imprimir um item a um objeto impressor
RespPilVaz (m)	É verdadeiro se m for uma resposta "pilha vazia" ao pedido de desempilhar um item a um objeto pilha.

impressão, pedida por t_7, e mandando uma mensagem ao objeto pilha, pedindo o desempilhamento de outro item. E assim por diante, até que a pilha do atendimento em questão fique vazia. Neste caso o objeto inversor (conexão t_9) recebe a mensagem de "pilha vazia", quando então é enviado ao objeto solicitante uma mensagem de final de atendimento e duas mensagens ao sistema de suporte pedindo a deleção dos objetos pilha e impresssor asssociados ao solicitante atendido.

Deve ser observado que o objeto inversor solicita que seja criado um outro objeto (pilha ou impressor) e fica aguardando a resposta de criação (veja conexões t_2 e t_6); e só então ele solicita serviços a estes objetos. Pode haver limites quantitativos de exemplares de objetos vivos em determinadas

Tabela 3.2 Descrição das funções

Função	Descrição
MSGCRPIL	Retorna uma mensagem solicitando ao sistema de suporte criar um objeto pilha.
MSGEMP (i,o)	Retorna uma mensagem solicitando ao objeto o empilhar o item i.
MSGCRIMP	Retorna uma mensagem solicitando ao sistema de suporte criar um objeto impressor.
MSGDESEMP (o)	Retorna uma mensagem solicitando ao objeto o desempilhar um item.
MSGIMP (i,o)	Retorna uma mensagem solicitando ao objeto o imprimir o item i.
MSGDEL (o)	Retorna uma mensagem solicitando ao sistema de suporte delir o objeto o.
MSGIT (o)	Retorna uma mensagem solicitando ao objeto o mandar um item.
IDOBJCR (m)	Retorna o identificador do objeto criado, se m for a resposta a solicitação de criar um objeto.
IDOBJO (m)	Retorna o identificador do objeto de origem da mensagem m.
RESPINV (o)	Retorna uma mensagem ao objeto o comunicando-lhe o término do serço.

(Obs.: o é um identificador de objeto)

classes (por exemplo unidades de impressão, unidades de fitas magnéticas etc.), daí os cuidados tomados neste modelo, pois o sistema de suporte que modelamos, figura 4, só cria um objeto (suponhamos) se o limite de exemplares da classe não for excedido.

Um único objeto inversor poderá atender a vários pedidos de inversão concorrentemente (no nosso exemplo a ordem de chegada dos pedidos não é observada). Entretanto como foi modelado um objeto solicitante só poderá submeter um pedido de inversão por vez. Isto fica garantido pela conexão t_1 a qual só fica habilitada se o objeto solicitante não figurar no lugar s_9. Isto é, se o objeto solicitante já não estiver correntemente usando os serviços de

Tabela 3.3 Descrição dos predicados variáveis

Predicados	Interpretação
Atendimento (a)	O objeto a está sendo atendido pelo objeto inversor.
Aguarda criar objeto pilha (a)	O objeto inversor está aguardando a criação de um objeto pilha para o atendimento do objeto a.
Aguarda item (a,p)	O objeto inversor está aguardando um item do objeto a, o qual será empilhado pelo objeto p.
Aguarda empilhar (a,p)	O objeto inversor está aguardando o objeto p empilhar um item, enviado pelo objeto a.
Aguarda criar objeto impressor (a,p)	O objeto inversor está aguardando a criação de um objeto impressor para o atendimento do objeto a cujos itens foram empilhados pelo objeto p.
Aguarda desempilhar (a,p,l)	O objeto inversor está aguardando o objeto p desempilhar um item, o qual será impresso pelo objeto l.
Aguarda imprimir (a,p,l)	O objeto inversor está aguardando o objeto l imprimir um item, oriundo da pilha p do atendimento do objeto a.

(Obs.: a, p, e l são identificadores de objetos)

inversão. E tão logo um objeto tiver o seu serviço de inversão concluido ele será eliminado do lugar s_9 pela transição t_9, ficando agora em condições de ter um novo pedido de inversão atendido.

3.3 OBJETO PILHA

O objeto inversor faz uso de um objeto pilha para realizar seu serviço de inversão de listas. A figura 7 mostra a rede de um objeto pilha. Este modelo simplificado possui três lugares e quatro conexões. A tabela 3.4 descreve as funções usadas na anotação da rede deste objeto.

Tabela 3.4 Descrição das funções

Função	Descrição
CONSULTARTOPO (p)	Retorna o item do topo da pilha p.
DESEMPILHAR (p)	Retorna a pilha p sem o item de topo
EMPILHAR (i,p)	Retorna a pilha p com o item i no topo.
ESVAZIAR (p)	Retorna a pilha p vazia.
RESULTADO (m,r)	Retorna a resposta ao pedido de serviço m com o resultado r.
SERVIÇO (m)	Retorna a identificação do serviço especificado em m.

(Obs.: p é uma entidade do tipo pilha)

O objeto pilha neste modelo, além da interface com o sistema de suporte MsgEntrada e MsgSaida, possui um lugar MemóriaLocal que apenas os seus procedimentos acessam. Neste caso a memória local contém uma estrutura de informação do tipo PILHA, inicialmente vazia. Em uma estrutura deste tipo os itens só são inseridos, consultados e/ou eliminados pelo topo. Nenhuma operação é realizada com itens que não estejam no topo da pilha. As conexões na rede do objeto modelado representam os serviços que este objeto pode realizar. Quando uma mensagem é colocada no lugar MsgEntrada, uma alteração, definida por uma das quatro conexões, t_1, t_2, t_3 ou t_4, fica habilitada, dependendo apenas do serviço que a mensagem vem solicitando (ESVAZIAR, CONSULTARTOPO, DESEMPILHAR ou EMPILHAR). O resultado da ocorrência da alteração habilitada, além de poder mudar a marca do lugar MemóriaLocal, é colocado, na forma de mensagem, no lugar MsgSaida. Daí cabe ao sistema de suporte entregar a mensagem ao objeto destinatário, no caso o solicitante do serviço realizado.

4. CONCLUSÕES

Procuramos formalizar os principais conceitos do paradigma de objetos sem referência ao jargão usual encontrado na literatura e nem a aspectos de implementação, com o objetivo de evitar enviesamento semântico.

O conceito de herança fica patente pelo processo de se criar

objetos exemplares que têm o mesmo gabarito de dados e serviços definidos para a classe.

A independência entre os objetos é assegurada pelo fato de eles não terem área de dados em comum e de se comunicarem exclusivamente através de mensagens. Esta independência torna os objetos entidades autônomas que podem realizar serviços de forma naturalmente concorrente. Cada objeto só "tem conhecimento" do seu estado local e do estado da sua interface com o sistema de suporte. E o comportamento do objeto só depende deste contexto e de nada mais.

A população de objetos é dinâmica: objetos surgem, interagem com a sua comunidade e desaparecem quando seus serviços se tornam desnecessários. A qualquer instante um objeto vivo pode solicitar a criação de um outro objeto exemplar de uma dada classe, solicitar serviços a este objeto e até mesmo pedir a sua deleção. Um objeto, uma vez criado, pode enviar e receber mensagens, realizar serviços e pedir que outros objetos lhe prestem algum outro serviço complementar aos seus próprios serviços.

A combinação dos conceitos expostos aqui para criar novas visões de objetos parece muito natural. Assim, o conceito de hierarquia de classes e herança múltipla são extensões que podem ser baseadas nesse trabalho.

A principal idéia que norteou este trabalho foi a de que os conceitos do paradigma de objetos não deveriam ser restringidos ou enviesados por aspectos de implementação, mas sim definidos, de uma forma neutra, com o maior rigor possível.

5. BIBLIOGRAFIA

BOOCH, g. Object-Oriented Development. **IEEE Transaction on Software Engineering.** 12(2):211-21, Feb 1986.

BRUNO, G. & BALSAMO, A. Petri Net-based object-Oriented Modelling of distributed systems. **ACM SIGPLAN Notices** 21(11):284-93, October, 1986.

COHEN, A.T. Data Abstraction, Data Encapsulation and Object Oriented Programming. **ACM SIGPLAN Notices,** 18(1):31-35, Jan. 1984.

CUNNIGHAM, W. & BECK, K. A Diagram for Object-Oriented Programs. **ACM SIGPLAN Notices,** 21(11):361-7, Nov 1986.

GENRICH, H.J. & LAUTENBACH, K. System modelling with high-level Petri nets. Theorical Computer Science, 13:109-36, North-Holland Publishing, 1981.

GENRICH, H.J. Predicate/Transition Nets. Lecture Notes in Computer Science, vol 254: Petri Nets: Central models and Their Proprieties. Advances in Petri Nets. pp 207-47, 1986.

HEUSER, C.A. Análise Estruturada de Sistemas com Redes de Petri. XX Congresso Nacional de Informática: 668-75, SUCESU, São Paulo, 1987.

HEUSER, C.A. Modelagem Conceitual de Sistemas. IV EBAI, Termas de Rio Hondo, Argentina, Janeiro de 1989. Kapelusz, Buenos Aires, 1989. 94 pp.

HEWITT, C. et alii. A Universal Modular Actor Formalism for Artificial Intelligence. Proceedings of 3rd IJCAI, 1973.

JONATHAN, M. Orientação para Objetos em SMALTALK-80: Uma abordagem eficaz para a construção de sistema de software. Primeiro Simpósio Brasileiro de Engenharia de Software, Itaipava, RJ:32-44, 1987.

MATICH, G. H. Princípios e Polêmicas sobre a "Orientação a Objetos". Trabalho de pesquisa (Orientador: Sergio de Carvalho), Informática/PUC-RJ, 1988.

NGUYEN, Van et alii. A Generalized Object Model. ACM SIGPLAN Notices, 21(10):78-87, Oct 1986.

REISIG, W. Petri Nets -- An Introduction. EATCS Monographs on Computer Science, vol 3. Springer-Verlag, Heidelberg, 1985a.

REISIG, W. A Primer on System Thinking in Terms of Net Theory. faor/model/gofor/g0048, anexo 1, GMD, 1985.

RENTSCH, T. Object-Oriented Programming. ACM SIGPLAN Notices, 17(9):76-87, Sep. 82.

RICHTER, G. O sentido e o valor do banco de dados. Dados e Idéias, 2(6):2-14, Rio, 1977.

RICHTER, G. Netzmodelle fuer die Buerokommunikation. Informatik-Spektrum (1983) 6:210-20 (em alemão).

RICHTER, G. et alii. Generic Office Frame of Reference. In: G. Schaefer (ed.), Functional Analysis of Office Requirements: A Multiperspective Approach. Wiley, 1988.

ROSSI, G. Programacion Orientada a Objetos: Inpacto en Tecnicas de Descripcion de Software baseados en Rede de Petri. Projeto ETHOS, II EBAI, Petrópolis, Abril 1987.

ROTENBERG, H.B. Programação Orientada a Objetos-Um enfoque de

Engenharia de Software. Tese de Mestrado, Informática/PUC-RJ, Abril de 1987.

STROM, R. A comparison of the Object-Oriented and Process Paradigms. ACM SIGPLAN Notices, 21(10):88-97, Oct 1986.

TAKAHASHI, T. Introdução a programação Orientada a Objetos. III EBAI, Curitiba, Janeiro de 1988. 148 pp.

TOKORO, M. & ISHIKAWA, Y. Concurrent Programming in Orient84/K: An Object-Oriented Knowledge Representation Language. ACM SIGPLAN Notices, 21(10):39-48, Oct 1986.

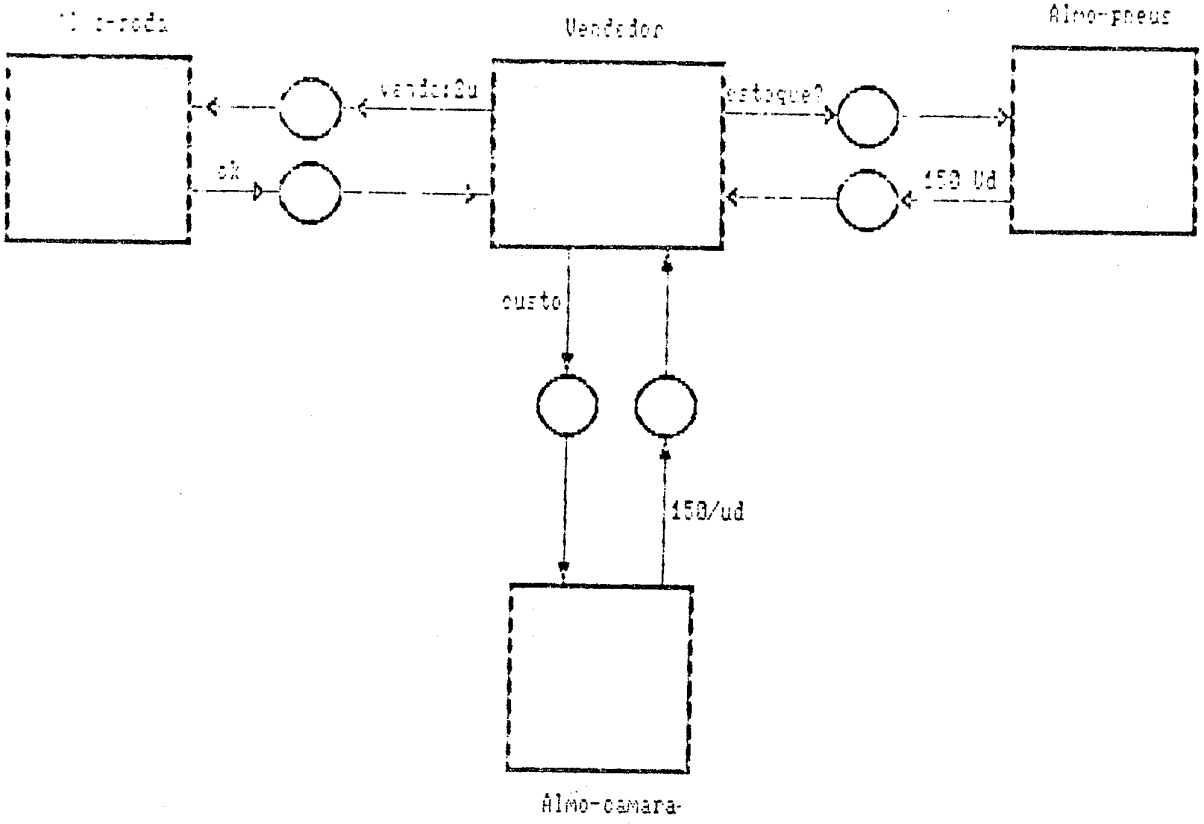

Figura 1: Interação entre quatro objetos

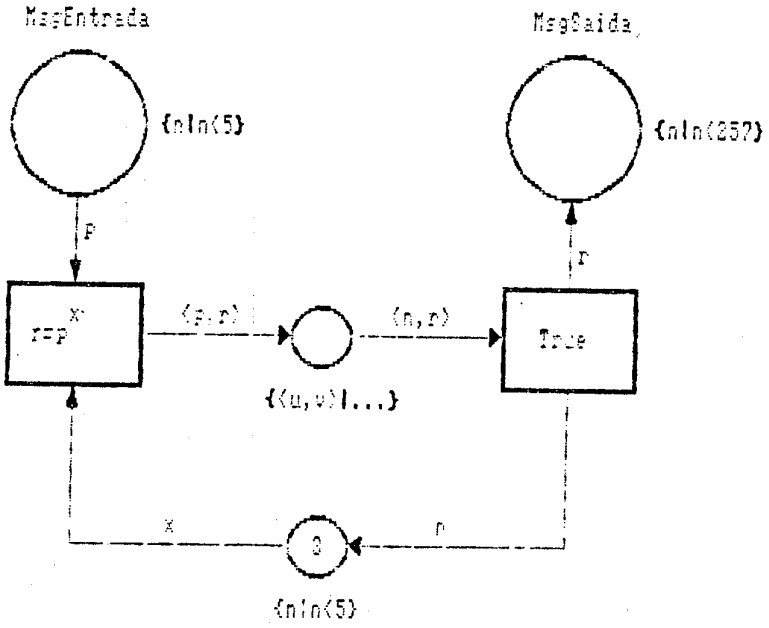

Figura 2: Um objeto (rede Pr/T com marcação inicial)

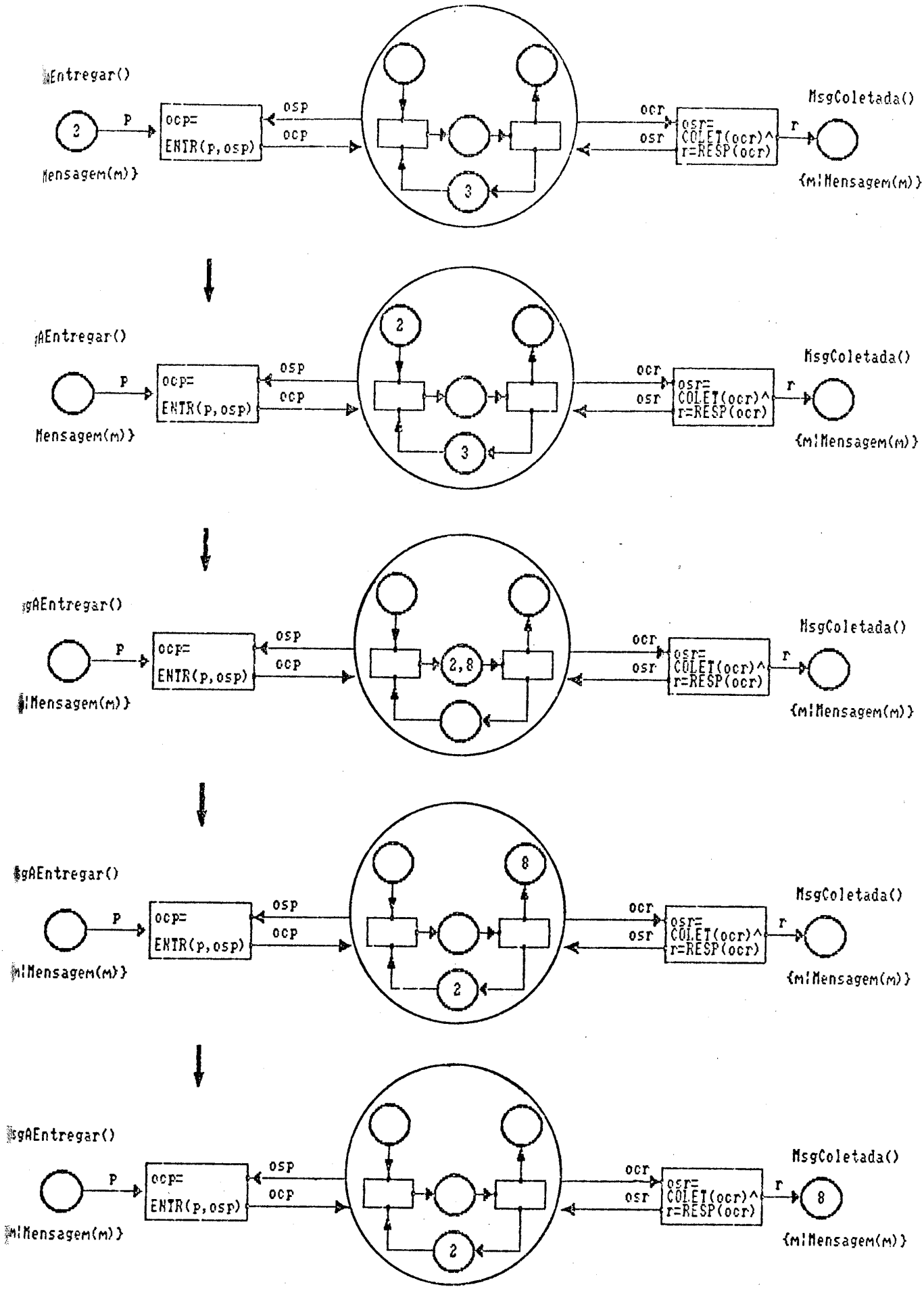

Figura 3: Interação entre a rede abrangente e a rede encaixada (objeto, figura 2)
(p=pedido, r=resposta, os/oc=objeto sem/com ...)

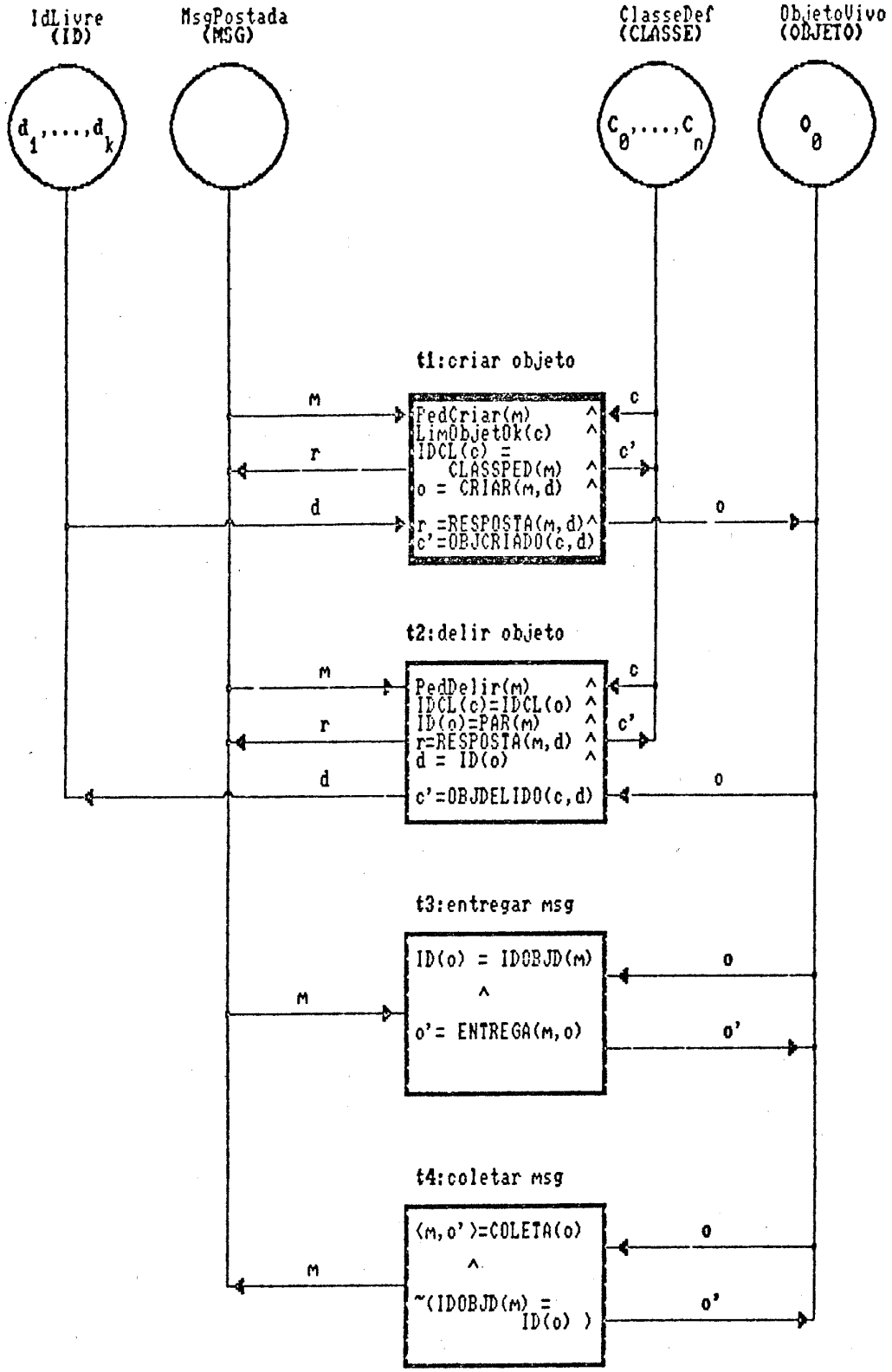

Figura 4: Sistema de suporte

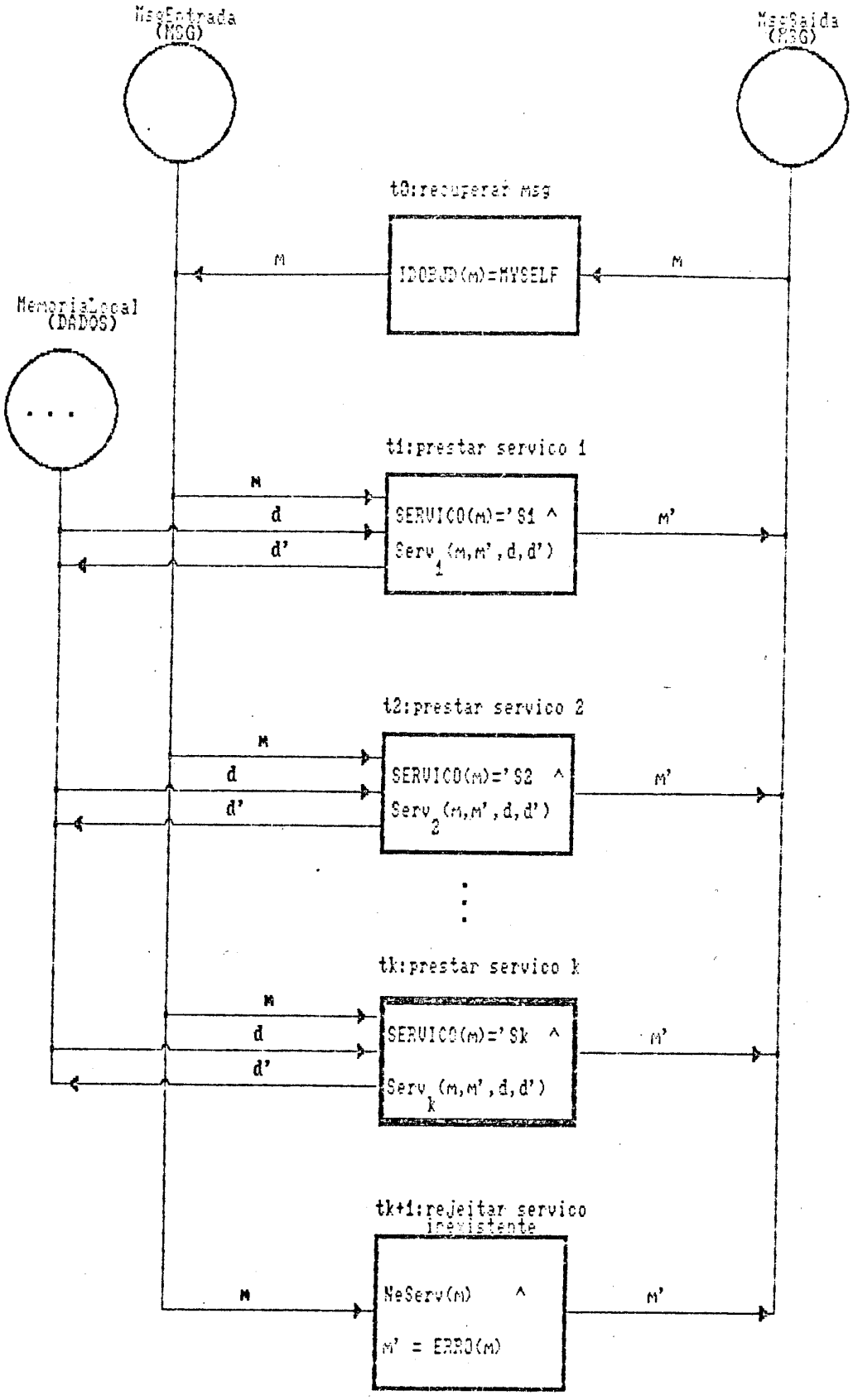

Figura 5: Objeto exemplar

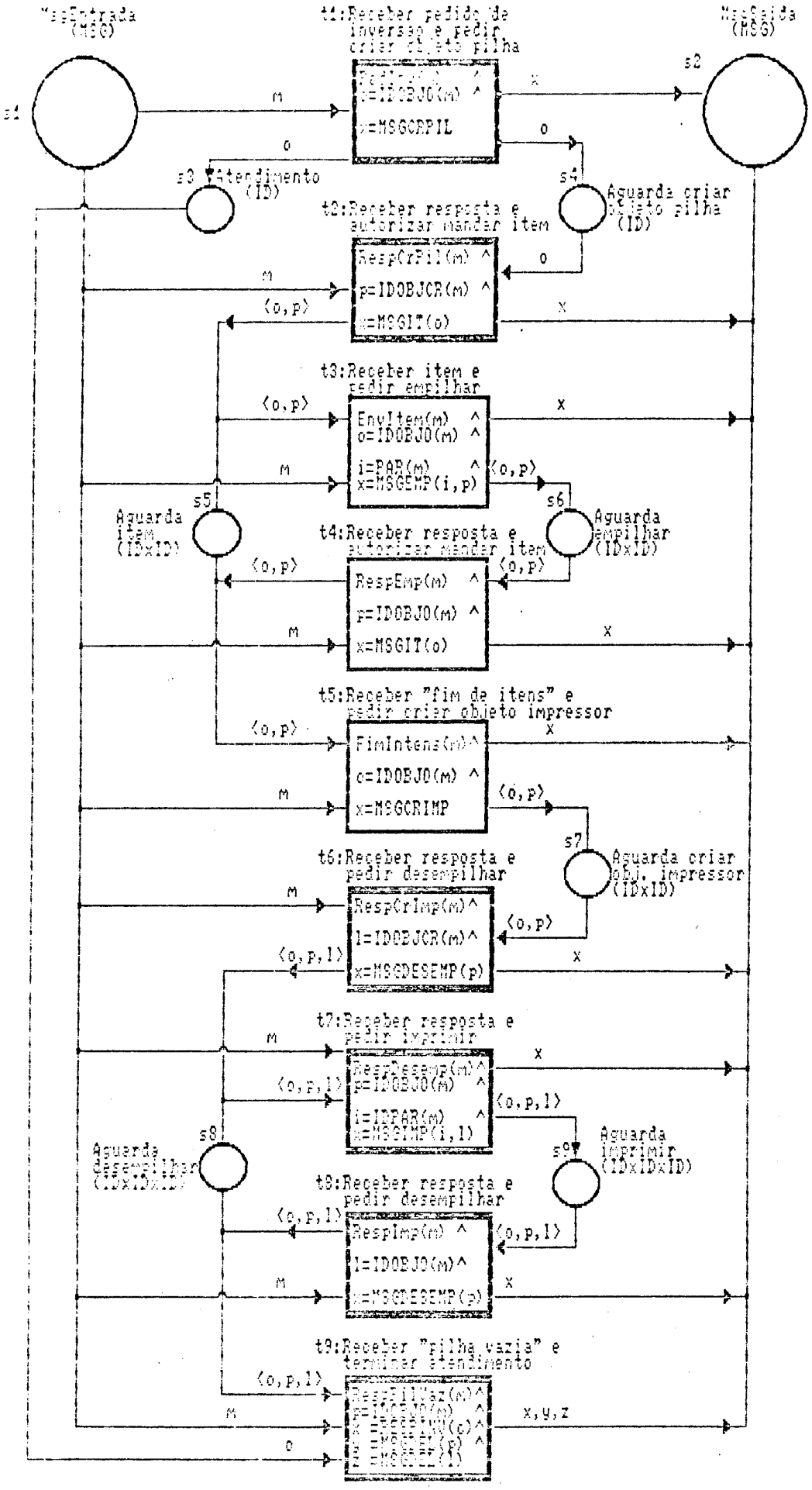

Figura 6: Exemplar de inversor (um objeto)

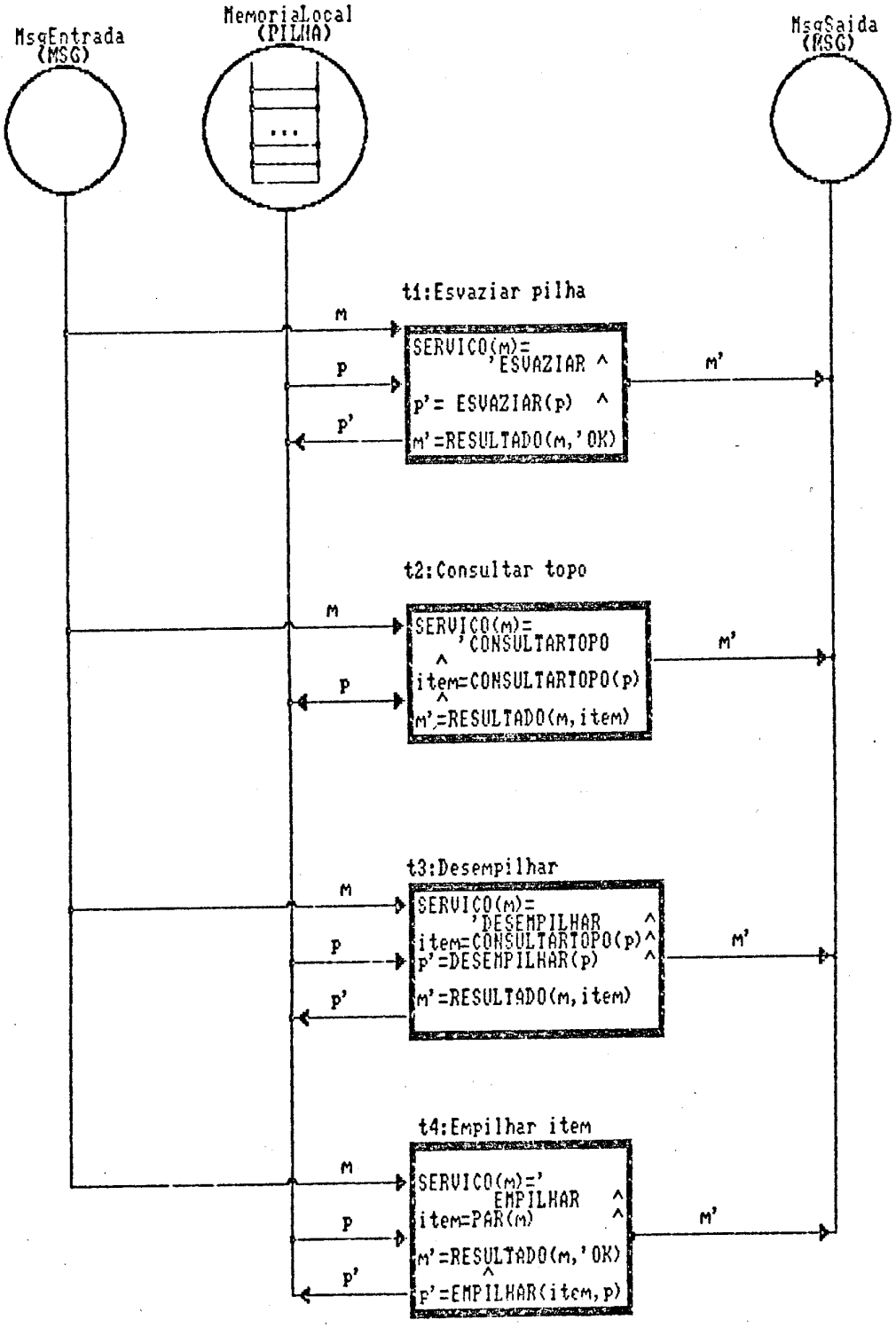

Figura 7: Exemplar de pilha (um objeto)

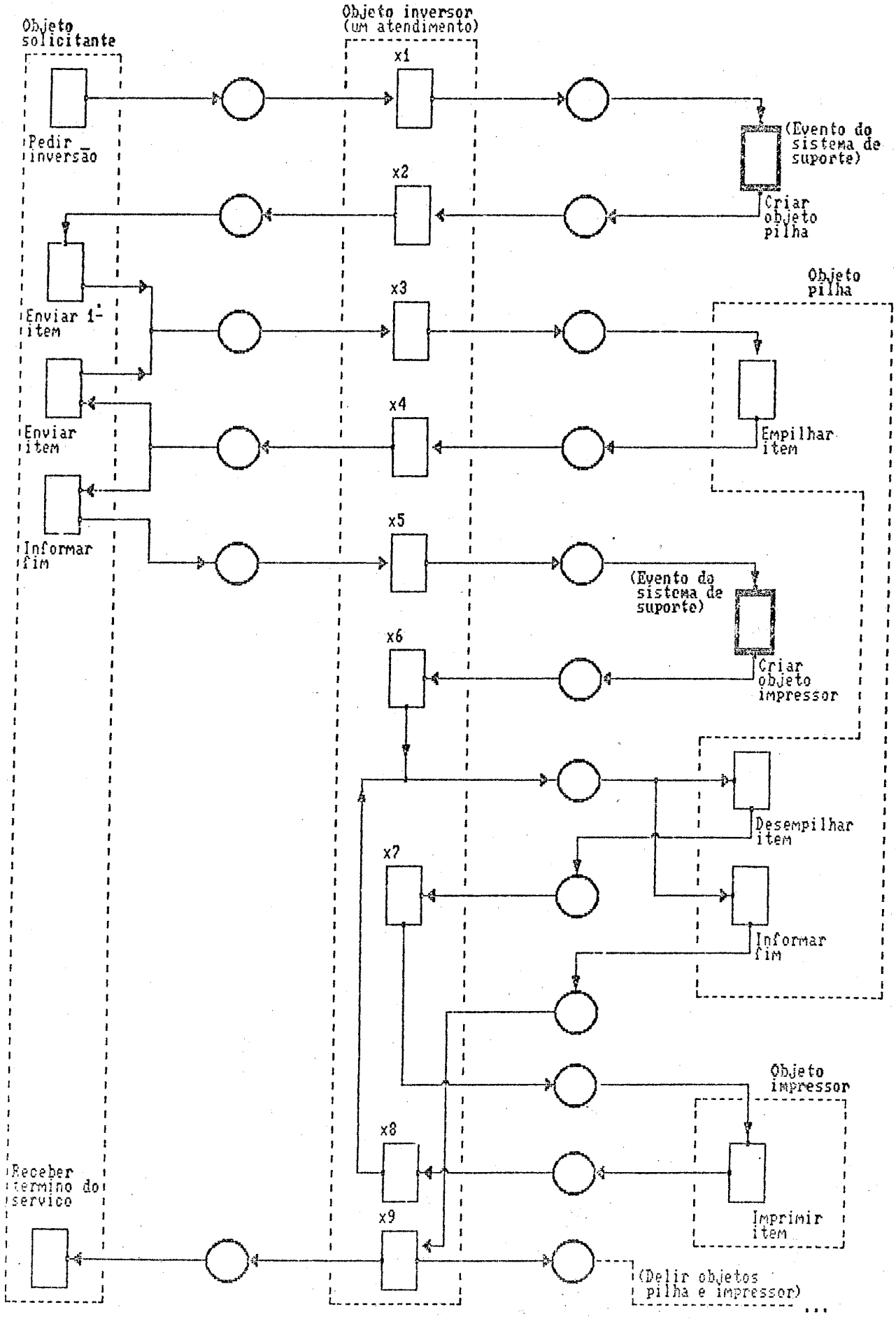

Figura 8: Rede elementar (1 marca/lugar)