Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência da Computação

Uma Proposta de Modelagem Conceitual de Sistemas Dirigida por Comportamento

por

Guillermo Bustos Reinoso

Tese submetida como requisito parcial para
a obtenção do grau de Doutor em
Ciência da Computação

Prof. Carlos A. Heuser
Orientador

Porto Alegre, abril de 1996.

CIP- CATALOGAÇÃO NA PUBLICAÇÃO

Bustos Reinoso, Guillermo

Uma proposta de modelagem conceitual de sistemas dirigida por comportamento / Guillermo Bustos Reinoso.- Porto Alegre: CPGCC da UFRGS, 1996.

286p.: il.

Tese (doutorado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1996. Orientador: Heuser, Carlos A.

1. Análise Orientada a Objetos. 2. Modelagem Orientada a Objetos. 3. Modelagem Dinâmica. 4. Modelagem Comportamental. 5. Modelagem Conceitual. 6. Especificação de Requisitos. 7. Projeto Orientado a Objetos. I. Heuser, Carlos A. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Hélgio Trindade
Pró-Reitor de Pesquisa e Pós-Graduação: Cláudio Scherer
Diretor do Instituto de Informática: Prof. Roberto Tom Price
Coordenador do CPGCC: Prof. Flávio Rech Wagner
Bibliotecária-Chefe do Instituto de Informática: Zita Prates de Oliveira

CNPq 1.03.03.00-6

AGRADECIMENTOS

Em primeiro lugar, quero agradecer à *Universidad Católica de Valparaíso*, no Chile, e, em particular, à *Escuela de Ingeniería Industrial* (que sempre confiou em mim), à *Agencia de Cooperación Internacional - Chile* e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), no Brasil, pelos diversos auxílios financeiros recebidos durante minha permanência neste curso de pós-graduação.

Agradeço ao meu orientador, Prof. Carlos A. Heuser, pelo seu generoso e desinteressado apoio e orientação, tanto no aspecto humano, quanto no aspecto acadêmico. Suas sugestões e críticas construtivas sempre foram muito valiosas no desenvolvimento de meus trabalhos e pesquisas, que concluiram felizmente com esta tese de doutorado.

Agradeço, também, aos meus colegas e amigos do CPGCC, que me proporcionaram um ambiente caloroso e fraterno que muito facilitou minha adaptação.

Aos membros da banca de defesa da tese: Prof. Dr. Bruno Maffeo, da PUC/RJ; Prof. Dr. Paulo Masiero, do ICMSC/USP; Profª Dra. Nina Edelweiss e Prof. Dr. Roberto Price, ambos do II/UFRGS, pelas diversas observações e perspectivas que muito melhoraram a qualidade final do texto aqui apresentado.

Finalmente, à minha família: aos meus pais, no Chile, que sempre me apoiaram quando eu quis progredir; à minha esposa Margarita que, com muito amor e compreensão, aceitou compartilhar o pouco tempo disponível e às minhas filhas, Camila e Tamara, que vieram ao mundo, a primeira, quando esta tese não existia nem como a semente de uma idéia, e a segunda, quando esta tese já estava parcialmente no papel.

A todos, meu muito obrigado!

"O que poderia ser mais antropomórfico na modelagem humana do que afirmar ser tudo uma máquina? As máquinas são, específica e cabalmente, criações humanas."

Rupert Sheldrake

SUMÁRIO

LISTA DE ABREVIATURAS

AOO: **A**nálise **O**rientada a **O**bjetos

HLS: *High-Level Statecharts*

MOO: **M**odelagem **O**rientada a **O**bjetos

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A **M**odelagem **O**rientada a **O**bjetos (MOO) é o processo de construção de modelos de sistemas através da identificação e definição de um conjunto de objetos relacionados, que comportam-se e colaboram entre si conforme os requisitos estabelecidos para o sistema. Esta definição inclui os três aspectos ortogonais, ou dimensões, deste tipo de modelagem: a dimensão estrutural dos objetos, a dimensão dinâmica do comportamento e a dimensão funcional dos requisitos.

Conforme a importância relativa dada a cada uma destas dimensões, podem ser definidas três estratégias possíveis para conduzir a MOO. Estas estratégias são as dirigidas por dados, por comportamento e por processos. A estratégia dirigida por processos já está superada. Atualmente, a estratégia dirigida por dados domina na maioria das técnicas de MOO.

A estratégia dirigida por comportamento propõe que a estrutura dos objetos em um sistema pode ser determinada a partir do comportamento externo e interno que o sistema deve apresentar. Esta idéia é interessante, porque permite introduzir tardiamente o encapsulamento na MOO. Conforme é argumentado neste trabalho, as vantagens atribuídas à orientação a objetos são de implementação, isto é, a decisão de orientar ou não a objetos é, na realidade, uma decisão de *design*. Ao introduzir o encapsulamento na modelagem inicial do sistema, ganha-se o benefício da continuidade estrutural ao custo de colocar a MOO mais perto do *design*.

Neste contexto, este trabalho apresenta um processo de modelagem conceitual de sistemas do ponto de vista comportamental que introduz tardiamente o encapsulamento da orientação a objetos como primeiro passo de *design*. Em outras palavras, é proposta uma técnica de modelagem sob uma estratégia dirigida por comportamento (privilegiando, assim, o aspecto dinâmico dos sistemas) com o suficiente poder de expressão para, ao mesmo tempo, permitir a modelagem de sistemas de informação no nível conceitual e derivar dos modelos dinâmicos obtidos uma representação estrutural orientada a objetos.

O sistema, na concepção desta proposta, é composto por um conjunto de processos concorrentes, cada um dos quais recebe um estímulo do ambiente, realiza um

tratamento específico sobre ele e gera para o ambiente uma resposta. Os estímulos externos são decompostos em conjuntos de eventos concorrentes tratados no interior do processo. As ações realizadas no interior do mesmo são compostas nas respostas geradas para o exterior.

Os processos são modelados comportamentalmente, utilizando o formalismo proposto *High-Level Statecharts* (HLS). HLS é uma extensão dos *statecharts* de Harel. As principais extensões propostas são a introdução de estados "parametrizados" usando variáveis e a representação genérica de conjuntos de estados concorrentes e exclusivos.

O modelo de processos é desintegrado em unidades de comportamento que tratam das mesmas variáveis. Estas unidades são integradas em um modelo de ciclos de vida para estas variáveis.

Finalmente, após a aplicação da técnica de modelagem conceitual, é obtido um modelo estrutural orientado a objetos. Este modelo é derivado utilizando unicamente informações contidas nos modelos dinâmicos gerados no processo da técnica proposta. No modelo estrutural são identificadas classes, objetos, atributos, associações estáticas, hierarquias de herança e operações.

Todo o processo é exemplificado utilizando o problema padrão de preparação de congressos da IFIP.

Palavras-chave: modelagem conceitual, modelagem comportamental, modelagem dinâmica, análise orientada a objetos, modelagem orientada a objetos, especificação de requisitos, projeto orientado a objetos.

TITLE: "A Proposal of Behavior-Driven Systems Conceptual Modeling"

ABSTRACT

Object-**O**riented **M**odeling (OOM) is the process of construction of systems models, through an identification and definition of a set of relating objects. These objects have a collaborative behavior according to the system requirements previously defined. This definition includes three modeling aspects or dimensions: object structural dimension, behavior dynamic dimension and requirements functional dimension.

Depending on a relative importance of each dimension, three possible strategies to drive OOM are defined. The strategies are: data-driven, behavior-driven and process-driven. Process-driven strategy is obsolete. Nowadays, data-driven is the dominant strategy in the world of OOM techniques.

Behavior-driven strategy suggests both internal and external system behaviors define its object structure. This idea is attractive because it allows a late encapsulation in the OOM. As explained in this work, the main advantage to use object-orientation is for implementation. So, to object-orient or not to object-orient is a design decision. If encapsulation is introduced in the very beginning of systems modeling, the structural continuity is achieved at the cost of pulling OOM closer to design.

In this context, the work presents a process of systems conceptual modeling using a behavioral point of view. This process introduces object-oriented encapsulation lately as a first step in the design phase. In other words, this work is a proposal of a modeling technique under a behavior-driven strategy (focusing the dynamic aspect of the systems) with enough expression power to model information systems at conceptual level and, at the same time, to derive of an object-oriented structural representation from the dynamic models.

As conceived in the proposal, a system is composed by a set of concurrent processes. Each process receives a stimuli from the environment, makes a specific treatment on it and generates a response to the environment. The external stimuli is decomposed into a set of concurrent events which are internally handled by the process.

Actions internally performed by the process are composed into a response which is sent outside the process.

Processes are behaviorally modeled using a proposed formalism called *High-Level Statecharts* (HLS). HLS is a extension of Harel's *statecharts*. The main extensions proposed are parameterized states using variables and generic representation of concurrent and exclusive sets of states.

Process model is disintegrated into behavior units handling the same variables. The units are integrated into a life cycle model for these variables.

Finally, after the modeling technique has been applied, an object-oriented structural model is obtained. This model is derived exclusively using information from the dynamic models constructed during the modeling process. Classes, objects, attributes, static associations, inheritance hierarchies and operations in the structural model are identified.

Examples used in all the modeling process are taken from the standard problem of IFIP conference.

Keywords: conceptual modeling, behavioral modeling, dynamic modeling, object-oriented analysis, object-oriented modeling, requirements specification, object-oriented design.

INTRODUÇÃO

O paradigma de desenvolvimento com orientação a objetos não é novo. A programação orientada a objetos surgiu com a linguagem Simula em 1967. Porém, apenas na segunda metade da década de 80 (quase 20 anos depois), ela deixou de ser uma corrente marginal para tornar-se tema central de discussão em programação. Muito ajudou, nessa época, o lançamento da linguagem C++, uma extensão orientada a objetos da linguagem C.

Aproximadamente nessa mesma época, começou a vir à luz a necessidade de contar com técnicas que permitissem conduzir a fase anterior à programação, o *design*, também de forma orientada a objetos. Em menos de 5 anos, já era evidente que a fase de análise também deveria considerar, pelo menos parcialmente em um primeiro momento, os conceitos da orientação a objetos. Desta forma, a orientação a objetos, de maneira análoga à dita "revolução estruturada", completava seu desenvolvimento "reverso": da implementação para o *design* e daí para a análise.

Na segunda metade dos anos 90, a **A**nálise **O**rientada a **O**bjetos (AOO) está em uma etapa de consolidação. A situação atual da AOO, no que diz respeito a pesquisas acadêmicas e aplicações na área industrial, pode ser descrita em termos de um conjunto de transformações simultâneas. Estas transformações podem ser descritas sucintamente como segue:

- Existência de muitas propostas alternativas para realizar a AOO. Inicialmente apresentadas na forma de artigos, agora a maioria é descrita em livros. Esta grande diversidade pode parecer benéfica, porém, muitas vezes, dificulta a escolha de alguma em particular e pode, inclusive, confundir na hora de definir quais os conceitos verdadeiramente relevantes neste tipo de análise.
- Aproveitamento de idéias, conceitos e representações sugeridas pelas propostas pioneiras em AOO para configurar técnicas denominadas de segunda geração.
- Crescente interesse na possibilidade de padronização e/ou convergência das metodologias de AOO.
- Mudança na forma de conduzir a AOO, de uma estratégia dirigida por processos (usando, por exemplo, os DFD's da análise estruturada) para uma estratégia dirigida por dados (usando os conceitos da modelagem semântica de dados) e, mais

recentemente, para uma estratégia dirigida por comportamento (usando modelos de interação do sistema com agentes/sistemas/usuários externos).

- Relativa estabilidade nos conceitos mais importantes da modelagem do aspecto estrutural dos objetos. Isto foi possível pela significativa contribuição da modelagem semântica de dados para este aspecto.
- Relativa instabilidade dos conceitos e modelos para o aspecto dinâmico dos objetos. Existem diferentes considerações sobre como abordar a dimensão do comportamento. Se a modelagem nesta dimensão é feita antes, depois ou em paralelo com a da dimensão estrutural, qual(is) o(s) modelo(s) a ser(em) aplicado(s), como gerenciar a complexidade do comportamento do sistema, como distribuir o fluxo de controle entre os objetos etc.?
- Aumento da adoção pela indústria de metodologias orientadas a objetos que consideram a análise, o *design* e a implementação ou, pelo menos, algumas destas fases.
- Aumento das opções de ferramentas CASE que introduzem, pelo menos parcialmente, os conceitos da orientação a objetos nas diversas fases do desenvolvimento de software.

Considerando este cenário atual, o presente trabalho propõe-se a aproveitar algumas destas tendências e descrever um processo de modelagem conceitual de sistemas que contribua para abordar e resolver alguns dos seus problemas.

Essencialmente, o objetivo principal deste trabalho é *desenvolver um processo de modelagem conceitual de sistemas do ponto de vista comportamental, que permita introduzir posterior e tardiamente o encapsulamento da orientação a objetos.* Ou, em outras palavras, propor uma técnica de modelagem sob uma estratégia dirigida por comportamento (privilegiando, assim, o aspecto dinâmico dos sistemas) com o suficiente poder de expressão para, ao mesmo tempo, permitir a modelagem de sistemas de informação no nível conceitual e poder derivar dos modelos obtidos uma representação estrutural orientada a objetos.

É necessário destacar que a modelagem conceitual no conceito desta proposta e particularmente o modelo dinâmico utilizado dentro deste processo, é independente, precede e fundamenta a estrutura da orientação a objetos.

Como objetivos secundários podem ser citados os seguintes:

1. Analisar criticamente o estado da arte na AOO, considerando as dimensões estrutural, dinâmica e funcional de modelagem, e as estratégias dirigidas por estas mesmas dimensões para detectar as principais deficiências que estas técnicas apresentam.
2. Introduzir na proposta de modelagem alguns elementos que venham a resolver algumas das deficiências indicadas.
3. Combinar o enfoque sistêmico, que permite contextualizar e conceber os sistemas de informação, com uma modelagem dinâmica/comportamental, que permite estruturar o sistema de maneira a gerar respostas aos estímulos que recebe.
4. Desenvolver um modelo dinâmico ou uma extensão de algum modelo existente, que forneça construtores de suficiente abstração e notações associadas para modelar conceitual e comportamentalmente os sistemas de informação.
5. Evitar a construção de modelos estruturais de objetos ou qualquer forma de encapsulamento orientado a objetos nos passos iniciais da modelagem proposta.
6. Introduzir semântica suficiente no modelo comportamental do sistema de tal forma que seja possível derivar, com relativa simplicidade, uma estrutura de objetos a partir dele.

Para poder atingir os objetivos acima, o presente trabalho, que descreve em detalhe a técnica proposta, foi organizado conforme os seguintes capítulos.

O capítulo 1 descreve o panorama atual do que se conhece como AOO ou, mais precisamente, a **M**odelagem **O**rientada a **O**bjetos (MOO). São descritos os três aspectos ou dimensões deste tipo de modelagem, quais sejam: estrutural, dinâmica e funcional. Também são descritas as atividades mínimas que devem ser desempenhadas para que exista uma MOO. Usando a perspectiva das três dimensões da modelagem, é indicado que estas atividades mínimas podem ser organizadas de diversas maneiras, definindo assim três estratégias para conduzir a MOO: dirigida por dados, dirigida por comportamento e dirigida por processos. Finalmente, é feita uma análise critica da situação da MOO como um todo e, em particular, para cada uma das estratégias mencionadas.

No capítulo 2, é apresentada uma visão geral da proposta de modelagem, que é descrita em maior detalhe nos seguintes capítulos. Neste capítulo, primeiramente, são descritas brevemente algumas bases teóricas utilizadas na concepção e desenvolvimento da técnica proposta. A seguir, são apresentados os três grandes passos que constituem o processo de modelagem proposto: a modelagem inicial do sistema, a construção de ciclos de vida e a derivação do modelo estrutural dos objetos. Cada passo é descrito através de suas atividades componentes e dos documentos e modelos de entrada e saída para estas atividades. Finalmente, o processo de modelagem é analisado à luz das atividades mínimas da MOO.

O capítulo 3 está dedicado integralmente a descrever a sintaxe e a semântica do modelo dinâmico fundamental para toda a proposta: os *High-Level Statecharts* (HLS), uma extensão do formalismo conhecido como *statecharts*. Inicialmente, são apresentadas as representações dos *statecharts* mantidas nos HLS e as representações excluídas. A seguir, são descritas as extensões relativas às variáveis usadas nos HLS, os estados simples e compostos e as transições e seus elementos. Exemplos para todos os casos são apresentados, a grande maioria tomada da definição do problema da IFIP.

No capítulo 4, é descrito o primeiro passo da técnica proposta: a modelagem inicial de sistemas. Neste passo são construídos dois modelos. O modelo da visão global é o mais abstrato dos dois e descreve o sistema em um contexto sistêmico, relacionado-o com outros sistemas por meio de estímulos e respostas. O outro modelo descrito é de processos. Este modelo é construído usando os HLS definidos no capítulo anterior. É descrita a forma de construir detalhadamente os diagramas HLS de cada processo componente do sistema. Praticamente todos os exemplos para este capítulo são tomados da definição do problema da IFIP.

O capítulo 5 descreve o seguinte passo dentro da técnica proposta: a construção de ciclos de vida. Este passo é descrito em função de suas atividades que permitem integrar o modelo de ciclos de vida (também usando HLS) a partir do modelo de processos obtido no passo anterior. Os casos de sinônimos e homônimos, próprios da integração de visões (*view integration*), também são considerados. Como exemplo de aplicação, continua sendo usado o problema da IFIP.

O capítulo 6 apresenta o terceiro e último passo da técnica de modelagem proposta: derivação do modelo estrutural de objetos. São indicadas heurísticas detalha-

das (com base em regras e/ou tabelas de decisão) para, a partir da informação contida em todos os modelos gerados nos passos anteriores, extrair um modelo estrutural que inclui classes, objetos, atributos, associações estáticas, hierarquias de herança e operações. O problema da IFIP continua sendo usado como exemplo de aplicação.

Finalmente, o capítulo 7 apresenta as conclusões e considerações finais para o presente trabalho. Este capítulo é dividido em: a verificação do cumprimento dos objetivos indicados nesta introdução; as contribuições gerais e particularidades deste trabalho; algumas considerações gerais sobre o trabalho; a descrição e um breve comparativo para destacar as diferenças e semelhanças com outras técnicas que também seguem a estratégia dirigida por comportamento e alguns lineamentos gerais para futuros desenvolvimentos a partir do trabalho aqui exposto.

Adicionalmente, são incluídos os quatro seguintes anexos: a descrição original em inglês do problema da IFIP, que é largamente usado como exemplo de aplicação neste trabalho; a notação dos HLS, que é o modelo dinâmico fundamental na técnica de modelagem proposta; o modelo completo do problema da IFIP, que mostra os diagramas resultantes da aplicação de cada um dos passos da proposta a este problema, e um resumo da notação OMT para o modelo estrutural de objetos.

1 MODELAGEM ORIENTADA A OBJETOS

A análise orientada a objetos, ou, mais precisamente[1], a Modelagem Orientada a Objetos (MOO), é uma área de pesquisa de recente data. Apenas nos anos 90 é que têm aparecido na literatura propostas mais elaboradas (na forma de livros) que visam abordar esta fase inicial do desenvolvimento de software sob o paradigma da orientação a objetos. A diversidade de técnicas cresce, criando com isto dificuldades para identificar:

- quais os conceitos verdadeiramente relevantes a serem considerados na construção de modelos orientados a objetos,
- quais as estratégias alternativas mais apropriadas para conduzir a MOO,
- quais as técnicas mais poderosas e adequadas aos diversos domínios de problemas e
- quais as atividades e procedimentos geralmente aceitos que devem ser realizados na MOO.

Neste sentido, este capítulo dá um panorama do que se conhece atualmente como MOO, indicando algumas características julgadas como inconvenientes do ponto de vista da modelagem conceitual.

O capítulo está organizado como segue: a seção 1.1 descreve o estado da arte em termos das dimensões ou aspectos da MOO e as atividades a serem realizadas ao construir modelos deste tipo; a seção 1.2 mostra as estratégias alternativas para conduzir a MOO; finalmente, na seção 1.3, são indicadas as principais considerações em relação a este tipo de modelagem.

---

[1] Os termos de *análise* e *orientação a objetos* parecem se contrapor. A noção de decomposição e exame crítico visando entendimento da *análise* não casa com a idéia de síntese (encapsulamento de atributos e operações nos objetos) e de alternativa de *design*/implementação que existe na orientação a objetos.

1.1 Estado da Arte na MOO

No contexto do desenvolvimento de sistemas de software com orientação a objetos, a Modelagem Orientada a Objetos[2] (MOO) é a construção de modelos de um sistema (problema) através da identificação e especificação de um conjunto de objetos relacionados, que se comportam e colaboram entre si conforme os requisitos estabelecidos para o sistema de objetos.

Aplicar a MOO significa obter um modelo de especificação do sistema com orientação a objetos.

1.1.1 As Dimensões da MOO

A definição anterior já permite distinguir, dentro do processo de MOO, três dimensões ou aspectos mais ou menos ortogonais para descrever um sistema de objetos: a dimensão estrutural dos objetos ("identificação e especificação de um conjunto de objetos relacionados"), a dimensão dinâmica do comportamento ("comportam-se e colaboram entre si") e a dimensão funcional dos requisitos ("requisitos estabelecidos para o sistema de objetos").

A *dimensão estrutural dos objetos* centra-se nas propriedades estáticas ou passivas dos sistemas. Está relacionada com a estrutura estática do sistema de objetos. A estrutura inclui a *identidade* de cada objeto, sua *classificação*, seu *encapsulamento* (seus *atributos* e suas *operações*[3]) e suas *associações* estáticas (hierarquias de herança, agregação, composição e associações específicas derivadas do domínio do problema).

A *dimensão dinâmica do comportamento* centra-se nas propriedades ativas e descreve o comportamento individual e a colaboração entre os objetos que constituem o sistema. O comportamento é refletido por meio de *estados* (estágios dentro do ciclo de vida dos objetos), *transições* entre estes estados, *eventos* (fatos relevantes para os objetos), *ações* (representadas pelas operações dos objetos) e comunicações (mensagens) entre os objetos.

---

[2] Definição adaptada da definição da análise orientada a objetos de [MON 92].

[3] As operações são apenas as assinaturas dos métodos, isto é, não incluem nenhuma descrição algorítmica.

No caso da *dimensão funcional dos requisitos*, são consideradas as propriedades relativas à função de transformação do sistema de objetos, isto é, os processos de conversão de entradas em saídas. Esta transformação é refletida por *funções* (que transformam valores) e *fluxos de dados* (entradas e saídas destas funções). As funções e os fluxos de dados são organizados em redes funcionais que podem ser construídas considerando os processos como encapsulados (modelagem *end-to-end*) ou, ainda, através de uma decomposição funcional pura, violando assim o princípio de encapsulamento da orientação a objetos[4].

Estas dimensões não devem ser consideradas como rigorosamente ortogonais, isto é, que cada uma delas considera seus próprios elementos de modelagem de maneira independente e não redundante. Cada dimensão é, na realidade, uma visão que enfatiza ou realça algumas das propriedades desconsiderando outras. Desta forma, a interseção entre os elementos descritos em cada uma das dimensões não é vazia.

A tabela 1.1 mostra quais os modelos que algumas técnicas de MOO utilizam para a modelagem considerando estas três dimensões.

1.1.2 Atividades da MOO

O processo de MOO pode ser dividido em um conjunto mínimo de atividades. A relação a seguir mostra estas atividades sem nenhuma seqüência específica. Ordens específicas dependerão das estratégias utilizadas na MOO (vide seção 1.2). As atividades são:

- identificar os objetos, os atributos e as classes,
- associar estaticamente os objetos,
- descrever o comportamento dos objetos,
- definir a colaboração do comportamento dos objetos,
- organizar as classes em hierarquias de herança e
- agregar e/ou particionar as classes por níveis de abstração.

---

[4] Esta discussão é detalhada mais adiante.

TABELA 1.1 - Dimensões da MOO vs. modelos de algumas técnicas de MOO (adaptado e estendido de [BUS 94c]).

DIMENSÃO / TÉCNICA	ESTRUTURAL	DINÂMICA	FUNCIONAL
BAILIN [BAI 89]	Diagrama de entidade-relacionamento e dicionário de entidades.		Diagrama de fluxo de dados de entidades.
COAD & YOURDON [COA 92]	Diagrama de classes-&-objetos (incluindo os níveis de classes-&-objetos, estruturas, assuntos e atributos).	Diagrama objeto-estado e nível de serviço.	Diagrama de serviço.
COLEMAN ET AL. [COL 94]	Modelo de objetos (incluindo classes, relacionamentos e atributos).	Modelo de interface (incluindo modelo de ciclo de vida do sistema e modelo de operações).	
EMBLEY ET AL. [EMB 92]	Modelo objeto-relacionamento.	Modelo objeto-comportamento e modelo objeto-interação.	
HENDERSON-SELLERS & EDWARDS [HEN 94]	Modelo O/C (objeto/classe), especificação de classes e modelo de herança.	Modelo de eventos e *objectcharts*.	Modelo de estrutura de serviço.
JACOBSON ET AL. [JAC 92]	Modelo de objetos do domínio (com nomes, atributos lógicos, associações estáticas de instâncias e herança).	Modelo de casos de uso (como interação), interfaces e modelo de objetos do domínio (com associações dinâmicas de instâncias e operações).	Modelo de casos de uso (como função).
KAPPEL & SCHREFL [KAP 91]	Diagrama de objetos.	Diagramas de comportamento e de especificação de atividades.	
MARTIN & ODELL [MAR 94]	Diagrama de categorização de objetos, diagrama de composição e diagrama de objeto-relacionamento.	Diagrama de eventos, diagrama de fluxo de objetos e diagrama de transição de estados.	
NERSON [NER 92]	Modelo estático (incluindo as cartas de classes e *clusters*).	Modelo dinâmico que mostra cenários de interação.	
RUBIN & GOLDBERG [RUB 92]	Modelos de objetos (incluindo cartões de modelagem de objetos, relacionamentos organizacionais e contratuais, e glossários de serviços e atributos).	Modelos da dinâmica do sistema (incluindo ciclos de vida dos objetos, seqüenciação de operações e glossário de estados).	
RUMBAUGH ET AL. [RUM 91]	Modelo de objetos.	Modelo dinâmico (incluindo diagramas de fluxo de eventos e diagramas de estado).	Modelo funcional (incluindo diagramas de fluxo de dados e descrição das funções).
SHLAER & MELLOR [SHL 92]	Diagrama de domínios, diagrama de estrutura de informação, descrição de objetos e atributos, especificação de relacionamentos e modelo de relacionamento de subsistemas.	Modelo objeto-acesso, modelo de acesso a subsistemas, modelo objeto-comunicação, modelo de comunicação de subsistemas e modelo de estados (incluindo diagrama de transição de estados e tabela de transição de estados).	Diagrama ação-fluxo de dados e descrição de processos.

1.1.2.1 A Identificação dos Objetos, Classes e Atributos

De acordo com [YOU 94]: "Não existe nada mais central, mais crucial, em qualquer metodologia orientada a objetos, do que o processo de descoberta de *quais* as classes e objetos que devem ser incluídos no modelo." É grande a importância que esta atividade possui, pois é ela que determinará os elementos de modularização (os objetos) sobre os quais será construído todo o restante do(s) modelo(s).

A identificação dos objetos de um domínio de problema é conduzida segundo os seguintes enfoques principais (para maiores detalhes vide [DEC 93], [FIR 93] e [HEN 94]):

- **Enfoque dos nomes de Abbott**: Baseado na técnica original de [ABB 83]. Os objetos são identificados como nomes (substantivos) em uma especificação textual do sistema em linguagem natural. Esta especificação deveria descrever seqüências de interação no contexto do sistema. Este enfoque pode falhar para localizar classes relevantes. As técnicas de MOO de [PRE 87], [BOO 83], [EVB 86] e [MAT 88] utilizam este enfoque.
- **Enfoque estruturado**: Usar diagramas de fluxo de dados (DFDs) contextuais da análise estruturada (vide por exemplo [YOU 90]) para identificar terminadores que podem corresponder a classes ou objetos. Também podem ser identificados os depósitos de dados dos DFDs de mais baixo nível, como potenciais classes ou objetos. As técnicas de MOO de [BAI 89], [BOO 86], [LEE 91], [ALA 88] e [BUL 89] utilizam este enfoque.
- **Enfoque de estado**: Estados desejados podem ser definidos em um documento de especificação. É possível então definir um objeto que exiba estes estados. Como "estado" pode ser usado em um sentido impreciso, este enfoque pode falhar na correta identificação de classes. A técnica de [HEN 94] sugere usar este enfoque entre outros, porém não é tratado especificamente.
- **Enfoque de atributo/operação**: Conforme Firesmith [FIR 93] (citado por [DEC 93], p.184), o enfoque estabelece que "para cada abstração de dados (ou abstração funcional), identificar a correspondente... classe da qual é um atributo (ou uma operação)."
- **Enfoque de relacionamento**: Entidades em um diagrama entidade-relacionamento [CHE 76] podem ser vistas como uma primeira aproximação de classes/objetos. Os nodos de descrições de domínios com redes semânticas (relacionamentos estáticos) ou diagramas de mensagem/interação (relacionamentos dinâmicos) podem ser identificados como classes/objetos candidatos. As técnicas de [NAV 89] e [MAN 89] utilizam este enfoque com relacionamentos estáticos.
- **Enfoque de decomposição**: Quando são definidos objetos muito complexos, é indispensável identificar os objetos componentes. Estes objetos podem estar relacionados ao objeto complexo em termos espaciais, temporais ou ainda em ter-

mos de associações metafóricas. A proposta de [BOO 91] usa parcialmente esta idéia.

- **Enfoque de reuso**: Uso de repositórios, que capturam elementos identificados de sistemas anteriores similares, tais como bibliotecas de classes (possivelmente produzidas por análise de domínio) e/ou *frameworks* de domínios. Este enfoque não é tratado como parte integrante de nenhuma das técnicas de MOO mais conhecidas.
- **Enfoque de abstração**: Utiliza itens do conhecimento do domínio sob modelagem, tais como: agregações, dispositivos ou objetos físicos, pessoas, papéis, unidades organizacionais, locais geográficos, eventos e interações, como classes e objetos potenciais. Este enfoque é utilizado pelas técnicas de MOO de [COA 92], [HEN 94], [YOU 94], [COL 94], [MAR 94], [RUM 91], [EMB 92], [SHL 90] e [KAP 91], entre outras.

Propriedades tidas como relevantes, atômicas, intrínsecas e determinantes dos objetos são consideradas como atributos dos mesmos. Os atributos são relações binárias entre uma classe e um domínio específico.

A identificação de tipos distintos de objetos determina também as classes às quais estes objetos pertencem. Geralmente, este processo de classificação é realizado conjuntamente com a identificação dos objetos do domínio no momento de descrevê-los estaticamente.

1.1.2.2 A Associação Estática dos Objetos

Os objetos devem configurar uma estrutura que expresse relacionamentos ou associações[5] estáticos dependentes do domínio do problema. Estes relacionamentos são conexões entre instâncias ou ocorrências (objetos) das classes, no sentido de uma tupla tipificada. Uma tupla tipificada é aquela em que cada elemento da tupla pertence a um domínio específico (uma classe). Este conceito é tomado inicialmente da modelagem semântica de dados, tal como no modelo entidade-relacionamento [CHE 76].

---

[5] Os termos *associação* e *relacionamento* são tratados aqui em forma equivalente.

Observe-se que a definição anterior permite a existência de relacionamentos entre mais de duas classes. Este tipo de relacionamento é chamado de relacionamento múltiplo. Nem todas as técnicas permitem relacionamentos múltiplos, entre elas [COA 92], [YOU 94], [RUB 92] e [NER 92].

As associações entre os objetos podem apresentar diversas características, entre elas:

- **Nomeação**: as associações podem levar nome ou não. A maioria das propostas de MOO utilizam nomeação, com exceção das de [COA 92] e [YOU 94].
- **Cardinalidade**: número de instâncias que participam na associação (incluindo opcionalidade). A cardinalidade é utilizada nas associações da maioria das técnicas de MOO, com exceção das de [BAI 89], [NER 92] e [RUB 92].
- **Direcionabilidade**: as associações podem ser dirigidas ou não, isto é, possuem um sentido, indo de uma classe para a outra. Entre as propostas de MOO que incluem a direcionabilidade nas associações estão as de [NER 92] e [EMB 92].
- **Padronização de associações mais comuns**: composição, agregação e participação (*membership*). A maioria das técnicas de MOO padroniza alguns relacionamentos, sendo a de composição a mais comum. As propostas de [BAI 89] e [NER 92] estão entre as que não padronizam associações.
- **Qualificadores**: permitem descrever as propriedades reflexiva, simétrica ou transitiva. Apenas as propostas de [DEC 93], [EMB 92] e em menor grau [RUM 91] estão entre as propostas de MOO que utilizam qualificadores.
- **Restrições**: aplicação de regras que melhor definem a semântica da associação. Entre as técnicas de MOO que permitem restrições estão as de [COA 92], [YOU 94], [RUM 91] e [DEC 93].
- **Coleções**: tais como conjuntos, listas, seqüências ou *arrays*. As técnicas de [EMB 92] e [DEC 93] estão entre as que permitem introduzir algumas destas formas de organização dos objetos.

As propriedades dos objetos conhecidas como atributos podem também ser modeladas como relacionamentos. Neste caso, o processo de MOO dá maior ênfase às associações, e tanto os objetos quanto os seus atributos (que se tornariam objetos) correspondem a nodos em uma rede de relacionamentos. As técnicas propostas por [EMB 92] e [BOO 91] optam por não definir atributos nos objetos na MOO.

1.1.2.3 A Descrição do Comportamento dos Objetos

O comportamento de um objeto dentro de uma classe é melhor representado utilizando os conceitos básicos de estado, regra de transição (ou transição simplesmente), evento e ação. Os *estados* são estágios dentro do ciclo de vida do objeto, e podem ser representados, por exemplo, por um subconjunto específico de valores do domínio dos atributos do objeto[6]. Uma mudança do valor de um ou mais atributos pode determinar uma mudança de comportamento do objeto que possui tais atributos, o que necessariamente deve ser refletido como o passo de um estado a outro. As *transições* representam, neste contexto, mudanças nos valores dos atributos, implicando assim mudanças de estados. Estas transições são produzidas como um reflexo a incidentes ou fatos relevantes acontecidos dentro ou fora do próprio objeto. Neste último caso, os fatos podem ser gerados por outros objetos ou ainda podem ser gerados no exterior do sistema. Estes fatos são conhecidos como *eventos*. As *ações*, que podem acontecer durante uma transição ou durante a permanência em um estado[7], são atividades ou operações que devem ser realizadas pelo objeto, tais como cálculos, modificação de atributos ou geração de eventos para outro(s) objeto(s) do sistema e/ou agente(s) externo(s).

Um aspecto relevante a ser considerado ao descrever o comportamento dos objetos diz respeito ao paralelismo ou concorrência interna. É uma questão em discussão na literatura se se deve simplificar os modelos de objetos não incluindo concorrência, como em [SHL 92], [DEC 93], [COA 92] ou [YOU 94] ou se se deve optar por modelos mais poderosos que a incluam, como [EMB 92], [MAR 94], [RUM 91] e [RUB 92].

Outras características dos modelos de comportamento dos objetos são :

- Eventos parametrizados e condicionados: possibilidade de incluir parâmetros nos eventos e usar *guards*. Técnicas de MOO tais como as de [RUM 91], [KAP 91] e [DEC 93] os incluem.

---

[6] Esta definição corresponde, na verdade, a um estado *passivo*, segundo [DEC 93], pois sua definição é precisa e baseada em valores de atributos.

[7] Neste caso, segundo [DEC 93], o estado é dito *ativo*, porque refere-se a uma situação em que o objeto está envolvido na execução de uma ação.

- Verificação dos efeitos para todas as combinações possíveis de estados e transições, tratada apenas na proposta de [SHL 92].
- Uso de identificadores para os objetos, como nas propostas de [SHL 92] e [KAP 91].
- Definição de *threads* de comportamento de exceção, como nas técnicas de [RUM 91] e [EMB 92].
- Hierarquização da complexidade através de construtores de abstração. As propostas de MOO de [RUM 91], [DEC 93], [RUB 92], [EMB 92], [HEN 94], [MAR 94] e [KAP 91] oferecem diversos conceitos para hierarquizar os modelos.

O modelo básico mais utilizado para conceituar o comportamento dos objetos é a máquina de estados finitos, que graficamente é representado por diagramas de transição de estados. As propostas de [SHL 92], [COA 92], [YOU 94], [MAR 94], [DEC 93] e [EMB 92] usam este tipo de diagramas. Outras alternativas usadas são as redes de Petri (vide [HEU 90]), como em [KAP 91], e os *statecharts* (extensões da máquina de estados finitos) de [HAR 87], como nas técnicas de [RUM 91] e [RUB 92]. Em [BUS 95] encontra-se um comparativo de técnicas para descrever o comportamento de objetos.

1.1.2.4 A Definição da Colaboração Entre os Objetos

Além de ser necessária a descrição do comportamento de cada objeto no sistema modelado, também é necessário identificar os eventos que produzem as transições, as seqüências dos eventos, os estados que definem o contexto para estes eventos, e a organização temporal destes eventos e estados. Assim, torna-se necessário transpor os limites do comportamento dos objetos para refletir a interação ou colaboração[8] entre eles e com os agentes externos ao sistema, considerando o fluxo de eventos ou mensagens entre os objetos e agentes. Deve ser definida, portanto, a forma em que a interação entre os objetos (e agentes externos) é realizada, isto é, quais os objetos (ou agentes) que participam, como os objetos agem na interação, e qual a natureza de tal interação.

---

[8] Os termos *interação* e *colaboração* são utilizados indistintamente.

Segundo [DEC 93], as interações entre os objetos e entre os objetos e os agentes externos podem ser categorizadas em:

- Um objeto gerador de um evento e um ou mais objetos receptores do evento, isto é, uma interação produtor-consumidor que implica uma comunicação assíncrona unidirecional. As propostas de modelagem de [RUM 91], [KAP 91], [EMB 92], [DEC 93], [MAR 94] e [SHL 92] caem nesta categoria de interação.
- Um objeto requer uma operação a ser realizada por outro objeto e espera pela sua resposta. Isto implica uma comunicação sincronizada bidirecional, associada ao conceito de *mensagem* da programação orientada a objetos. As técnicas de MOO de [COA 92], [YOU 94] e [DEC 93] tratam as comunicações desta forma.

Adicionalmente, as colaborações podem considerar:

- Possibilidade de comunicação apenas entre classes, entre classes e instâncias e apenas entre instâncias. Esta distinção é feita nas técnicas de [COA 92], [YOU 94] e [EMB 92].
- Mensagens geradas por múltiplos objetos origens e/ou recebidas por múltiplos objetos destinos, consideradas nas propostas de [COA 92], [YOU 94], [MAR 94], [SHL 92], [EMB 92] e [KAP 91].
- Mensagens qualificadas, sujeitando a condições os objetos que geram e/ou recebem as mensagens, suportadas pelas técnicas de [KAP 91], [SHL 92], [MAR 94], [EMB 92] e [DEC 93].
- Seqüenciação das mensagens, suportada pelas técnicas de [COA 92], [HEN 94] e [YOU 94].
- Mensagens de exceção, consideradas pelas propostas de [RUM 91] e [EMB 92].
- Identificadores dos objetos envolvidos na comunicação, incluídos pelas técnicas de MOO de [SHL 92] e [KAP 91].
- Agentes externos ao sistema de objetos, considerados explicitamente nos modelos de [RUM 91], [SHL 92] e [EMB 92].
- Hierarquização da complexidade da colaboração em níveis de abstração, tratada por [EMB 92] e [DEC 93].

Em [BUS 95] pode ser encontrado um comparativo de técnicas para definir as colaborações entre objetos.

1.1.2.5 A Organização das Classes em Hierarquias de Herança

As hierarquias de herança de classes podem ser construídas através de dois processos: generalização (abstração de propriedades comuns de uma ou mais subclasses para uma superclasse) e especialização (reuso de propriedades de uma superclasse em uma ou mais subclasses).

As propriedades herdadas entre um nível e o seguinte dentro de uma hierarquia podem ser atributos/operações (da superclasse), relacionamentos (da superclasse com outras classes), comportamento (da superclasse) e colaboração (entre a superclasse e outras classes ou agentes externos). Estas propriedades eventualmente podem ser modificadas ou estendidas na subclasse que herda tais propriedades. Nem todas as técnicas de MOO permitem herança de todas as propriedades (como por exemplo [RUB 92], [NER 92], [BAI 89] e [RUM 91]), assim como, também, não todas permitem modificar as propriedades herdadas (como por exemplo [COA 92], [YOU 94], [RUM 91], [SHL 92] e [EMB 92]).

A herança também pode ocorrer em forma múltipla, isto é, uma subclasse pode herdar de mais de uma superclasse simultaneamente. Isto é conhecido como herança múltipla. A maioria das propostas de MOO permite herança múltipla.

Segundo [DEC 93], as relações que podem ser estabelecidas entre as subclasses que herdam de uma superclasse podem ser de:

- **Exclusão**: As subclasses são por definição disjuntas, e portanto não podem apresentar objetos (instâncias) em comum. As técnicas de [RUM 91], [COL 94], [HEN 94] e [EMB 92] estão entre as que reconhecem este tipo de relação.
- **Cobertura**: Qualquer instância da superclasse deve pertencer a pelo menos uma das subclasses. Esta propriedade não é considerada pelas propostas de MOO, com exceção das de [DEC 93] e [EMB 92].
- **Particionamento**: Quando existe em forma simultânea exclusão e cobertura, tem-se o *particionamento* da superclasse.

1.1.2.6 A Agregação e o Particionamento das Classes por Níveis de Abstração

A própria natureza *plana* da colaboração entre os objetos (e a construção *middle-out* e *bottom-up* de modelos devida a esta natureza) causa sérias dificuldades para a modelagem de sistemas grandes e complexos. A orientação a objetos não fornece explicitamente mecanismos de particionamento ou decomposição de tarefas, equivalentes àqueles encontrados, por exemplo, na análise estruturada clássica [DEM 78]. Na MOO é mais apropriado utilizar mecanismos de agregação ou composição.

Entre as características mais relevantes dos mecanismos para agregar e/ou particionar as classes por níveis de abstração, conforme apresentado em [BUS 94a], estão:

- **Estratégia**: Se compõe ou decompõe, ou seja, se é essencialmente *top-down* e portanto particiona previamente o sistema de objetos ou é predominantemente *bottom-up* e agrega as classes já identificadas. Técnicas como as de [REE 92], [RUB 92] e [SHL 92] são indicadas para o particionamento. Para a agregação, as propostas de [COA 92] e [YOU 94], [NER 92], [BAI 89], [DEC 93], [RUM 91] e [BOO 91] são as mais indicadas. Já as propostas de [JAC 92] e [SHL 92], [BUS 94b] e [EMB 92] combinam ambas as estratégias.
- **Dimensão**: Se utiliza elementos de uma ou mais dimensões da modelagem para guiar o particionamento/agregação. A maioria das propostas de MOO centra-se na dimensão estrutural para orientar o particionamento/agregação, tais como as de [RUM 91], [COA 92], [BUS 94b], [YOU 94], [NER 92], [BAI 89] e [BOO 91]; porém outras introduzem alguns elementos da dimensão dinâmica (geralmente em relação à colaboração entre os objetos) para realizar o particionamento. As técnicas de [DEC 93], [SHL 92], [REE 92], [JAC 92] e [EMB 92], em diversos graus, podem ser consideradas como deste tipo.
- **Abrangência**: Abrangência que o conceito ou construtor de particionamento/agregação possui ao longo do ciclo de vida de desenvolvimento orientado a objetos. A maioria das técnicas de MOO apresenta construtores que não transcendem as fases de *design* e implementação. Entre estas técnicas estão as de [REE 92], [COA 92], [YOU 94], [BAI 89], [BUS 94b], [SHL 92], [RUM 91], [JAC 92] e [EMB 92]. Os construtores de outras propostas cobrem também o *design* e sua implementação orientados a objetos, tais como as de [DEC 93], [NER 92] e [BOO 91], entre outras.

1.2 Estratégias da MOO

As seis atividades descritas na seção anterior podem ser ordenadas de diversas maneiras, conforme os diferentes procedimentos propostos pelas técnicas de MOO. Contudo, estes procedimentos podem ser categorizados em três estratégias gerais, refletindo a divisão por dimensões da MOO. Estas estratégias são: dirigida por dados (*data-driven*), dirigida por comportamento (*behavior-driven*) e orientada por tarefas ou processos (*process-driven*).

Estas estratégias podem ser consideradas como alternativas para conduzir a MOO. Contudo, existem certas tendências que levam, de maneira geral, a preferir uma às outras. Isto será tratado em detalhe mais adiante.

1.2.1 A Estratégia Dirigida por Dados

Nesta estratégia, a MOO é dirigida principalmente pela informação estática sobre o domínio do problema. Inicia-se a modelagem pelos elementos associados à dimensão estrutural dos objetos, para continuar depois com os elementos da dimensão dinâmica do comportamento.

A idéia central por trás desta estratégia é a de que a estrutura dos objetos do sistema é determinante para o comportamento que este apresenta. Os objetos e as classes e suas respectivas associações conformam a base dos modelos comportamentais.

Segundo [JAC 95b]: "Esta estratégia começa com a identificação de entidades do mundo real do domínio do problema e a [posterior] alocação de dados e comportamento a estas entidades."

As perguntas-chave a serem formuladas nesta estratégia são [YOU 94]: Quais os objetos que "existem" no domínio do problema? Quais as entidades próprias do vocabulário do domínio? Quais os dados que precisam ser mantidos pelo sistema?

A identificação dos objetos nesta estratégia pode ocorrer sob os enfoques dos nomes de Abbott, de atributo, de relacionamento (estático), de decomposição, de reuso e de abstração.

A figura 1.1(a) apresenta a seqüência geral para este tipo de estratégia.

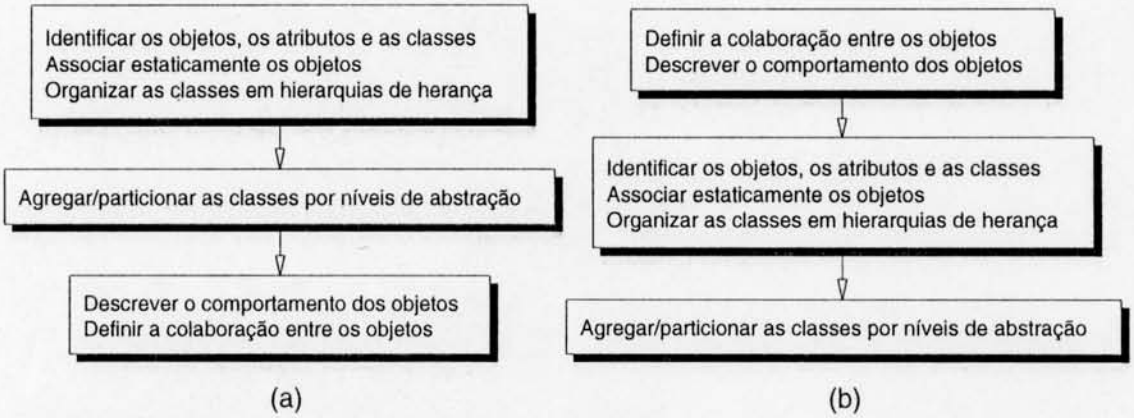


FIGURA 1.1 - Seqüência de atividades da MOO de acordo com: (a) estratégia dirigida por dados; (b) estratégia dirigida por comportamento.

A grande maioria das técnicas de MOO seguem esta estratégia, com algumas pequenas variações e adições nos seus procedimentos. Entre estas técnicas encontram-se as de [BOO 91], [COA 92], [COL 94], [DEC 93], [EMB 92], [KAP 91], [HEN 94], [MAR 94], [NER 92], [RUM 91], [SHL 92] e [YOU 94].

1.2.2 A Estratégia Dirigida por Comportamento

A estratégia dirigida por comportamento é baseada em informação sobre o comportamento esperado do sistema de objetos. A MOO, sob esta estratégia, inicia com a definição do comportamento global do sistema e segue, depois, com a definição da colaboração de algum tipo de componentes do interior do sistema. Estes componentes ou entidades se tornarão objetos candidatos. Assim, esta estratégia começa com os elementos da dimensão dinâmica do comportamento, para seguir com os elementos da dimensão estrutural dos objetos.

O pressuposto básico é que a estrutura dos objetos do sistema é derivada do comportamento que este deve apresentar. Em outras palavras, a estrutura é determinada pelo comportamento do sistema. A forma pela qual o sistema e seus componentes agem e reagem colaborando define as associações estáticas entre os objetos do sistema.

Segundo [YOU 94], perguntas-chaves nesta estratégia são: Como os objetos se comunicam? Com quem? O que fazem para responderem aos eventos internos ou externos do sistema? Qual o comportamento dos componentes do sistema?

Os objetos são identificados, nesta estratégia, por um único enfoque ou uma combinação dos enfoques de estado, de relacionamento (dinâmico), de reuso e de abstração.

A figura 1.1(b) mostra a seqüência padrão das atividades da MOO para esta estratégia. As primeiras atividades devem ser modificadas a fim de considerar o fato de que os objetos ainda não estão identificados.

A quantidade de procedimentos de propostas que seguem esta estratégia é bem menor em relação à que adota a estratégia orientada por dados. As principais técnicas dentro desta estratégia são as de [GIB 90], [JAC 92] e [RUB 92]. A técnica MOSES (***M**ethodology for **O**bject-Oriented **S**oftware **E**ngineering of **S**ystems*) [HEN 94] usa informação de cenários de interação porém é apenas uma fonte entre outras. A proposta de [WIR 90], que é mais centrada no *design*, utiliza uma abordagem dirigida por responsabilidades (*responsibility-driven*) dentro desta estratégia comportamental.

1.2.3 A Estratégia Dirigida por Processos

Nesta estratégia a modelagem é orientada pela informação sobre as tarefas (processos, responsabilidades ou ainda funções) que devem ser desempenhadas pelo sistema e seus componentes. A modelagem geralmente inicia pelos elementos associados à dimensão funcional dos requisitos e continua com os elementos da dimensão estrutural. Muitas vezes, a dimensão dinâmica não é considerada em benefício da visão funcional que fornece a modelagem de processos.

Essencialmente, esta estratégia utiliza as facilidades dos modelos funcionais (ou de processos) para derivar (ou associar) os objetos e sua estrutura. A forma pela qual os processos se relacionam proporciona elementos para definir a estrutura dos objetos.

As perguntas que devem ser feitas, segundo [YOU 94], são: O que um objeto deve fazer para existir? Quais as suas responsabilidades ou tarefas no contexto do sistema? Quais os requisitos funcionais estabelecidos para o sistema?

Os objetos podem ser identificados nesta estratégia segundo os enfoques: estruturado, de operação, de reuso e de abstração.

Os procedimentos das propostas que seguem esta estratégia estão entre os primeiros que foram publicados. Entre as propostas que utilizam modelos funcionais estão as de [ALA 88], [BAI 89], [BOO 86], [BUL 89], [GIR 90], [JAL 89], [LEE 91], [SCH 90], [SEI 89] e [WAR 89].

1.3 Considerações sobre a MOO

De acordo com a descrição anterior sobre o estado da arte em MOO, podem ser indicadas algumas considerações importantes que apontam principalmente algumas fraquezas neste tipo de modelagem.

Estas considerações são divididas naquelas que tratam da MOO como um todo e naquelas que se referem às dimensões e estratégias deste tipo de modelagem.

1.3.1 A MOO Como um Todo

A essência do paradigma da orientação a objetos, e, mais especificamente, a essência da MOO, é a visão da realidade composta por objetos. Freqüentemente é afirmado na literatura que esta visão é *mais natural* do que uma visão funcional ou comportamental. É razoável perguntar, então (vide [BUS 94a]), se existe realmente uma visão mais natural. Uma afirmação desta natureza é em verdade uma ingenuidade, já que existem muitas indicações em contrário, conforme as perspectivas expostas a seguir [OPD 93]:

- É um fato sustentado há muito tempo na filosofia oriental e, em menor grau, em algumas escolas da filosofia ocidental, que a realidade é antes de tudo mudança, isto é, que uma visão de processos/comportamentos é mais fundamental do que uma de objetos/coisas.
- A visão da realidade surgida da física moderna (vide por exemplo [HEI 85], [CAP 92] e [BOH 92]) sugere que aquela é uma teia de relações dinâmicas no seu nível mais fundamental. As relações dinâmicas (comportamento) incluem os objetos (estáticos).
- Na linguagem natural, mais especificamente, na língua portuguesa, o verbo ("...que indica a ação de fazer uma coisa...", [NAH 62], p.223) é a classe de palavra mais sofisticada existente, com variadas formas de conjugação, modo, tempo e irregularidades. As formas dos substantivos ("...que exprimem determinadamente os seres, pela idéia de sua natureza", [NAH 62], p.194) são em menor número e mais simples (gênero, número e grau). O verbo é a palavra central e usualmente todas as outras palavras da oração se relacionam com ele de alguma forma com o propósito de complementá-lo. Dar ênfase aos nomes/objetos em detrimento de verbos/processos vai contra a natureza da própria linguagem.
- Do ponto de vista da teoria dos sistemas, os sistemas de informação são subsistemas das organizações humanas, e estas são sistemas de mudança contínua *par excellence* (vide por exemplo [CHU 71] e [WEI 75]). Modelar comportamentalmente ou por processos é mais adequado neste contexto.

Segundo Constantine [CON 89], não existem *classes de objetos* no universo físico real. São construções que existem apenas na mente dos observadores; os *objetos* não são mais reais ou naturais do que as *funções*, conclui ele. A Constantine, Loy [LOY 90] acrescenta que esta variação da controvérsia *forma vs. função* não está resol-

vida e provavelmente nunca será resolvida. Assim, não se pode afirmar que as classes de objeto sejam a forma *mais natural* pela qual as pessoas vêm a realidade. Alguns argumentos muito utilizados quanto a esta pretensa "naturalidade" são:

- que os objetos da realidade estão visíveis, prontos para serem "pegos" com facilidade, como indica [MEY 88],
- que os objetos são relativamente fáceis de identificar porque o domínio do problema é facilmente acessível, segundo [MCG 92], ou
- que o mapeamento dos objetos "reais" para o modelo é direto, como explica [KAP 91].

Isto é parcialmente verdade para objetos no sentido físico, porém qualquer outra abstração que pode ser entendida como um objeto é subjetiva e arbitrária, dependendo fortemente do modelador.

A motivação principal para utilizar outras perspectivas para esclarecer o problema da "naturalidade" dos objetos é ir contra a tendência na ciência da computação de forçar a realidade modelada a uma conceituação originada na construção de software. A MOO sustenta-se, entre outros, no conceito de encapsulamento. O encapsulamento propõe que sejam separados, nos objetos, os aspectos externos dos aspectos internos. Os aspectos externos são conhecidos por outros objetos; em compensação, os aspectos internos relativos à implementação são de conhecimento exclusivo do próprio objeto. O encapsulamento não é restrito à orientação a objetos, porém é nela que ganha mais força, porque os objetos combinam internamente a estrutura de dados (atributos) e os processos que a manipulam (métodos). Desta forma, os objetos são auto-contidos e não compartilham nenhum tipo de informação que não a solicitada nas colaborações com outros objetos. A questão (que também foi levantada em [BUS 94a]) é a seguinte: faz sentido encapsular ao nível da modelagem conceitual? Dado que a maior vantagem do encapsulamento revela-se nas fases posteriores de implementação, quais benefícios poderiam se esperar do encapsulamento na fase de modelagem conceitual? Encapsular na modelagem conceitual não representaria introduzir um viés de implementação?

É comum ser apontada como principal vantagem, no sentido de utilizar a MOO, a uniformidade conceitual dos sistemas de objetos, que pode e deveria ser mantida nas seguintes fases do ciclo de vida orientado a objetos, como foi indicado em [BUS

94a]. O *gap semântico*[9] se veria diminuído: assim, por exemplo, os objetos identificados no domínio do problema seriam mapeados diretamente nos objetos especificados no domínio da solução durante o *design*. O mesmo aconteceria com estes objetos ao serem mapeados nos objetos codificados na implementação. Isto é conhecido como *continuidade estrutural* (*structural continuity*), segundo [DEC 92], nos sucessivos mapeamentos entre a MOO, o *design* orientado a objetos e sua implementação também orientada a objetos. Pode afirmar-se, então, que a MOO seria vantajosa se tanto o *design* como a implementação fossem também conduzidos sob o mesmo paradigma. Contudo, o benefício da continuidade estrutural não deveria ter o custo de colocar a MOO mais perto do *design* [OPD 93]. A modelagem conceitual, na engenharia de requisitos, deve ser orientada ao problema (realidade) e não à solução concreta (implementação orientada a objetos).

A confusão existente entre o que é a análise orientada a objetos, da qual a MOO faz parte, e o que é o *design* orientado a objeto, ou, em outras palavras, a pouca claridade nos limites entre estas duas fases é freqüentemente indicada na literatura (vide por exemplo [HEN 90], [KOR 90], [MON 92] e [SHL 93]). O princípio de que a MOO é declarativa em relação ao *design*, isto é, de que descreve *o que* o sistema deve fazer em contraste com o *design* que estabelece *o como* isto deve ser feito, não é tão claro nas técnicas propostas na literatura. Na prática, os modelos orientados a objetos da análise são apenas menos detalhados do que aqueles do *design*; assim, a declaratividade não é atingida [OPD 93]. A MOO torna-se, então, uma espécie de *design* [HOY 93], porque uma vez que os requisitos são entendidos, estes são organizados em um modelo que serve como estrutura interna do sistema a ser projetado [JAC 95b]. Algumas organizações e metodologistas têm decidido nomear mais adequadamente estas fases. Por exemplo, a Ellemtel (uma companhia de pesquisa e desenvolvimento da Ericsson) usa os termos *ideal design* para o que se conhece como MOO e *real design* para o que normalmente é chamado de *design* orientado a objeto [JAC 95b]. McGregor & Sykes [MCG 92] sugerem análises de domínio e de aplicação para depois continuar com um *high-level design*, mas estes três passos contêm muitas das atividades próprias da MOO.

Concluindo, a orientação de objetos deve ser considerada como uma decisão de *design* e implementação e não como uma decisão de modelagem conceitual ou

---

[9] Como *gap semântico* se define a "distância conceitual" entre o domínio do problema no mundo real e o domínio da solução na implementação computacional [BUS 93].

de análise de requisitos. A maturidade das técnicas de MOO existentes[10] não permite realizar modelos de qualidade sem introduzir os conceitos da implementação orientada a objetos.

Um dos caminhos para resolver esta questão é criar mecanismos que permitam postergar tanto quanto possível o encapsulamento. Em outras palavras, modelar o domínio do problema fora do paradigma da orientação a objetos, com suficiente flexibilidade para poder representar a complexidade da realidade. Ao mesmo tempo, poder derivar, com relativa facilidade, modelos orientados a objetos se a implementação desejar ser realizada sob este paradigma. Desta forma, a modelagem é concluída antes de introduzir-se o encapsulamento, ponto em que inicia-se o *design* orientado a objetos.

Outro problema indicado em [BUS 94a] diz respeito ao particionamento do sistema sob modelagem e/ou à agregação das classes componentes do sistema modelado. Pela própria natureza plana da estrutura estática e da interação dos comportamentos dos objetos, a construção dos modelos tende a seguir as estratégias *bottom-up* e/ou *middle-out*. Com estas considerações, devem existir critérios objetivos e práticos que permitam particionar sistemas ou agregar classes, principalmente tratando-se de sistemas grandes e complexos, nos quais uma estratégia *top-down* no processo de modelagem é altamente desejável. As primeiras questões que surgem são: 1) o particionamento inicial deve ser na perspectiva da organização, da qual o sistema de informação faz parte (sistemas e subsistemas), ou na perspectiva do software orientado a objetos (*clusters*, classes e objetos)? e 2) são incompatíveis estas visões? Neste sentido, pode-se questionar se deve ser realizada uma decomposição *antes* da identificação das classes (estratégia *top-down*) ou uma composição *após* a identificação destas (estratégia *bottom-up*). Ou em outras palavras, se o sistema é particionado em subsistemas ou se as classes são agregadas em *clusters* de algum tipo.

1.3.2 As Dimensões e as Estratégias da MOO

Historicamente, as aplicações de processamento de dados de negócios têm apresentado os *dados* como característica "dominante". Uma modelagem segundo a estratégia dirigida por dados é a mais adequada nestes casos. Independentemente da

---

[10] Vide uma comparação conceitual de algumas técnicas de análise orientada a objetos em [BUS 93] e uma aplicação comparativa em [BUS 94].

consideração do domínio do problema a ser modelado, esta é, largamente, a estratégia mais usada pelas técnicas de MOO. É um caminho mais explorado porque evoluiu a partir da modelagem semântica de dados (vide [POT 88]). A dimensão estrutural dos objetos é, também, a principal base para os mecanismos de particionamento/agregação, muitas vezes aplicados para toda a MOO, desconsiderando o aspecto dinâmico do sistema de objetos.

No caso de aplicações de tempo-real ou de sistemas distribuídos, a dimensão dominante é a dinâmica, sugerindo, com isto, a estratégia dirigida por comportamento para a MOO. As propostas que se inserem nesta perspectiva são poucas e mais recentes, indicando assim, um processo de modelagem menos explorado e com maior potencial de desenvolvimento. A dimensão dinâmica do comportamento não é utilizada para conduzir particionamentos e/ou agregações do sistema de objetos. Os construtores disponíveis estão divididos naqueles necessários para hierarquizar o comportamento de um tipo de objeto ou naqueles apenas usados para gerar visões de alto nível de colaborações típicas entre objetos.

De acordo com [FIC 92], os domínios de problemas que contêm processos globais que afetam vários tipos de objetos (e envolvem a execução seqüencial e/ou paralela de numerosos passos intermediários entre sua iniciação e finalização) indicam a dimensão funcional como dominante. Exemplos deste tipo de problema constituem os processos de encomendas para a manufatura, a compensação diária de contas bancárias, e os tradutores de linguagens. Neste tipo de problema, uma estratégia dirigida por processos é de maior utilidade.

Contudo, as técnicas para a MOO negligenciam a modelagem funcional argumentando que o conceito de um processo de transformação global, não subordinado a qualquer objeto individual, vai contra o espírito da orientação a objetos, como indicam [FIC 92]. Alguns autores, tais como [BOO 89] e [DEC 91], sugerem, inclusive, não utilizar nenhum tipo de modelo funcional para não introduzir uma orientação funcional na MOO. Isto relaciona-se principalmente com a inconveniência de utilizar uma decomposição funcional clássica nos passos iniciais da MOO, no sentido da análise estruturada tradicional [DEM 78].

Esta discussão é parcialmente resolvida introduzindo o conceito de visão funcional de um sistema de objetos, isto é, a construção de um modelo de processos en-

capsulados nos objetos que represente seqüenciação, execução condicional e idéias relacionadas com processos globais de transformação do sistema. Esta modelagem é denominada *end-to-end*, segundo [FIC 92], e já tinha sido sugerida em outros contextos por [BAI 89] e [HEN 91]. Esta visão só deveria ser construída após uma MOO conduzida pelas estratégias dirigidas por dados ou por comportamento. A técnica MOSES [HEN 94] recomenda o desenvolvimento de um modelo de estrutura de serviço para esta visão funcional.

As três estratégias já mencionadas têm aparecido e se desenvolvido historicamente conforme a figura 1.2.

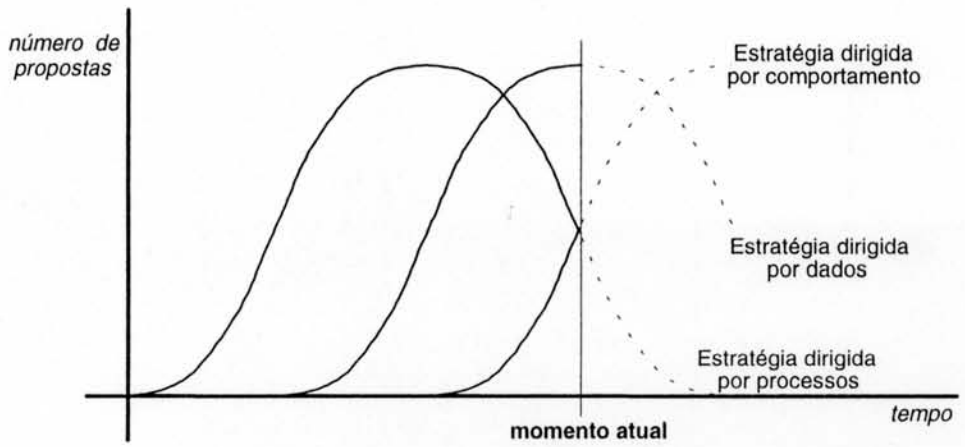


FIGURA 1.2 - Evolução histórica das estratégias da MOO.

A estratégia dirigida por processos foi a primeira a surgir (a partir da segunda metade dos anos 80). As técnicas propostas naquela época tentaram aproveitar os modelos funcionais da análise estruturada de forma a conseguir uma transição a partir de um "terreno conhecido" para os conceitos da orientação a objetos. Porém, como foi indicado acima, existe uma clara ortogonalidade entre os conceitos de classificação, encapsulamento e herança da orientação a objetos e a decomposição funcional. De acordo com figura 1.2, esta estratégia tende a desaparecer e hoje já se pode afirmar que está superada.

Após a estratégia anterior, surge a estratégia dirigida por dados para a MOO (a partir de fins da década dos 80). A maioria das propostas de análise orientada a objetos opta por esta forma. O viés principal desta estratégia é a dominância da identificação das classes e seus relacionamentos estáticos e a organização destas classes em hierarquias de herança. Agindo desta forma, as técnicas capturam primeiro a estática e logo a dinâmica do sistema. Pela subordinação à estrutura estática, a dimensão dinâmica recebe geralmente uma menor ênfase que a dimensão estrutural. Os modelos dinâmicos mostram uma grande quantidade de mensagens/eventos/comunicações sobre a estrutura dos objetos [OPD 93].

Surgem alguns problemas práticos ao realizar as seguintes atividades próprias desta estratégia:

- distribuir apropriadamente as responsabilidades/comportamentos aos diferentes tipos de objetos respeitando a estrutura estática previamente definida de acordo com os requisitos para o sistema;
- verificar se efetivamente o comportamento individual dos objetos e sua colaboração satisfazem adequadamente os requisitos estabelecidos para o comportamento global do sistema e
- iterar sucessivamente entre os dois passos anteriores até conseguir o modelo satisfatório.

A realização destas atividades tornam a MOO fortemente dependente do modelador. Sua experiência é indispensável para obter modelos de maior qualidade e com maior produtividade.

O princípio de que o comportamento do sistema de objetos depende da estrutura do mesmo está sendo questionado. Ou, em outras palavras, a suposição de que a dimensão estrutural é mais fundamental que a dimensão dinâmica é um equívoco que começa a ser notado. O aspecto comportamental do sistema é tão ou mais importante do que o aspecto estático. A maioria das críticas atribuídas à MOO devem-se, em grande medida, ao uso generalizado desta estratégia que, no momento atual, está no ponto mais alto do seu desenvolvimento e divulgação, como é mostrado na figura 1.2. Tanto é assim, que é comum encontrar na literatura termos tais como "modelo de objetos", "modelo orientado a objetos" ou ainda "modelo da análise orientada a objetos" referindo-se, na verdade, apenas a um modelo estrutural orientado a objetos.

O surgimento da estratégia dirigida por comportamento é relativamente mais recente (a partir do início da anos 90) e menos divulgado. Notadamente, o modelo de casos de uso (*use case model*) de Jacobson [JAC 92] é a maior (não a primeira) contribuição desta estratégia. Este modelo representa uma maneira específica de usar o sistema: através da "execução" de alguma parte de sua funcionalidade. Cada caso de uso descreve um cenário completo de eventos iniciados por um ator (ou papel de usuário) e especifica a interação que acontece entre o ator e o sistema. A partir deste modelo são derivados os objetos constituintes do sistema. O modelo de casos de uso é feito para capturar e entender os requisitos do novo sistema antes de construí-lo.

Usar esta estratégia para a MOO resolve os problemas inerentes à estratégia dirigida por dados, quais sejam:

- a distribuição das responsabilidades/comportamentos é derivada do cenários de interação do sistema, diminuindo a importância das decisões do modelador;
- a satisfação dos requisitos estabelecidos para o sistema é garantida na construção dos cenários de interação e
- evita-se excessivas iterações, porque os objetos são derivados a partir de um modelo dinâmico já definido como satisfatório.

Conseqüentemente, o processo de MOO torna-se mais eficaz e eficiente.

O princípio de que a estrutura do sistema de objetos é determinada pelo comportamento esperado do sistema está ganhando maior aceitação. Muitos metodologistas estão sugerindo usar este tipo de modelo de interação global antes de desenvolver a estrutura do sistema de objetos, como é indicado em [HIL 95] e [JAC 95a]. A curva que representa o desenvolvimento desta estratégia está em forte crescimento, conforme mostrada na figura 1.2.

Adicionalmente, esta estratégia está em maior concordância com as perspectivas filosófica, física, lingüística e sistêmica da realidade indicadas na seção 1.3.1.

De acordo com as tendências das curvas destas últimas estratégias, uma extrapolação (mostrada com linhas pontilhadas na figura) levaria a um aumento significativo na adoção da estratégia dirigida por comportamento em substituição à dirigida por dados.

1.4 RESUMO

O presente capítulo apresentou uma descrição do panorama atual do que se conhece como MOO. Para isto, foram descritas as três dimensões ou aspectos ortogonais da modelagem: estrutural, dinâmica e funcional. Também foram descritas brevemente as 6 atividades mínimas da MOO que tratam da identificação e associação dos objetos e atributos, da descrição do comportamento e colaboração entre os objetos, da organização das classes e hierarquias de herança e do particionamento/agregação das classes por níveis de abstração. A seguir, discutiu-se as três estratégias para conduzir a MOO: dirigida por dados, dirigida por comportamento e dirigida por processos.

Posteriormente foi feita uma análise crítica em relação à essência da MOO: a visão da realidade composta por objetos e o encapsulamento no nível conceitual. Chegou-se à conclusão de que a orientação a objetos é uma decisão de *design* e implementação. O encapsulamento deveria ser introduzido o mais tardiamente possível, como um primeiro passo na fase *design*, após concluída a análise e modelagem de sistemas.

Finalmente, indicou-se que dentre as três estratégias possíveis para a MOO, aquela dirigida por processos já está superada. A estratégia dirigida por dados estaria dominando em termos de quantidade de propostas e divulgação. Contudo, a estratégia dirigida por comportamento está em franco crescimento, augurando um futuro promissor, graças às vantagens que oferece, a partir de diferentes perspectivas, em relação à estratégia dirigida por dados.

2 VISÃO GERAL DA PROPOSTA DE MODELAGEM

Em função do estado da arte na MOO e dos problemas conceituais e práticos que esta apresenta, é possível esboçar uma proposta de modelagem de sistemas, cujo processo permita modelar os sistemas na perspectiva dinâmica e também permita postergar o encapsulamento da orientação a objetos. Assim, a proposta é definida, dentro da estratégia descrita, como dirigida por comportamento.

O objetivo deste capítulo é fornecer uma visão geral da proposta de modelagem a ser descrita em detalhe nos próximos capítulos. Esta visão geral inclui algumas bases teóricas da proposta, procedimentos e modelos resultantes.

Adicionalmente, são analisadas as atividades definidas para a MOO dentro do processo da proposta de modelagem.

O capítulo foi dividido como segue: a seção 2.1 apresenta uma síntese das principais idéias usadas para a concepção e desenvolvimento da proposta, extraídas de disciplinas distintas das de ciência de computação; a seção 2.2 fornece uma visão geral da proposta, descrevendo os três grandes passos, suas atividades constituintes e os documentos de entrada e saída para cada uma delas; finalmente, a seção 2.3 analisa a técnica de modelagem proposta na perspectiva da MOO.

2.1 Uma Breve Síntese das Bases da Proposta

A proposta de modelagem pode ser dividida conceitualmente em três assuntos:

1. **Enfoque sistêmico**: para situar, contextualizar e conceber o sistema de informação sob modelagem.
2. **Modelagem dinâmica**: para descrever o comportamento e a estrutura interna do sistema.
3. **Modelagem estrutural orientada a objetos**: para derivar o modelo estrutural de objetos.

Os dois primeiros assuntos merecem uma breve síntese de suas principais idéias, que foram aplicadas na concepção e desenvolvimento da proposta. Estas idéias estão fortemente relacionadas entre si e foram extraídas da ciência dos sistemas (vide [CHU 71], [WEI 75], [BER 80] e [CAP 92]), da epistemologia e da ontologia (vide [HES 74], [MAH 84], [BOH 92] e [VAR 90]). As idéias são:

- **A realidade é essencialmente dinâmica**: A realidade é concebida como uma teia de relações dinâmicas, que pode ser melhor representada como uma rede interconectada de eventos, da qual o observador/modelador também faz parte.
- **Qualquer hierarquização é arbitrária**: Qualquer hierarquização dentro desta teia de relações é puramente arbitrária. As partes (ou subsistemas) dentro de um todo (um sistema) são padrões subjetivos de percepção, determinados pelas relações entre estas supostas partes.
- **As abstrações são criadas pelo observador**: Dado um sistema cujas fronteiras foram arbitrariamente fixadas pelo observador/modelador, o supra-sistema do qual o sistema faz parte e os subsistemas que compõem este sistema são apenas abstrações para lidar com a complexidade intrínseca da teia de relações dinâmicas. Estas abstrações são indispensáveis para abordar qualquer processo de modelagem e dependem fundamentalmente do observador/modelador.
- **O sistema como uma totalidade**: As propriedades das partes só podem ser entendidas a partir da dinâmica do todo. O comportamento e a função de um subsistema estão definidos entendendo-se o subsistema como parte do sistema. Recursivamente, o comportamento e a função do sistema estão definidos entendendo-se o sistema como parte de um supra-sistema.
- **O sistema possui um duplo papel**: Qualquer sistema a ser modelado terá, necessariamente, um duplo papel: compor o supra-sistema e manter sua individualidade como uma composição de subsistemas. Isto quer dizer que é igualmente importante definir a composição de um sistema, como sua interação com outros sistemas que definem seu papel dentro do supra-sistema.

Estas idéias possuem algumas conseqüências específicas para a modelagem de sistemas, e particularmente para a MOO:

- **Mudança da ênfase nos objetos para as relações dinâmicas**: Os objetos podem ser deduzidos a partir das relações dinâmicas em uma rede de eventos. Os objetos são padrões de comportamento nesta rede.

- **Mudança da ênfase na estrutura para os processos**: A estrutura dos objetos pode ser derivada a partir de uma rede de processos subjacente. As associações estáticas entre os objetos são padrões de interação entre os processos.
- **As propriedades dos objetos são determinadas pelas relações**: Os atributos e operações dos objetos estão determinados unicamente pelas suas associações com outros objetos.

Conforme o anterior, é possível construir modelos dinâmicos conceituais de sistemas não orientados a objetos, nos quais posteriormente são identificados objetos (padrões de comportamento) e sua estrutura (padrões de interação) em função dos comportamentos e interações. Desta forma, o encapsulamento é postergado dentro do processo de modelagem e *design*, assumindo claramente uma estratégia dirigida por comportamento para a MOO.

Encapsular tardiamente permite, nestes moldes, aumentar a qualidade, riqueza, eficácia e poder de expressão do modelo sem forçar os conceitos da orientação a objetos dentro da modelagem conceitual inicial. A modelagem inicial torna-se assim orientada ao problema e não orientada à solução (sendo a orientação a objetos uma alternativa possível de implementação).

Também é possível afirmar que os modelos estruturais de objetos podem ser derivados, ou melhor, correspondem a uma visão dos modelos dinâmicos. Assim, a proposta de técnica de modelagem conforme a estratégia dirigida por comportamento ganha maior generalidade. Esta idéia não é completamente nova. Em [HEU 93] é mostrada a representação de entidades e relacionamentos estáticos entre entidades (segundo um modelo ER estendido), utilizando redes de Petri [HEU 90] de alto nível.

O sistema de informação sob modelagem é, então, um sistema que reage planejadamente ao ambiente e que, portanto, desempenha uma função ou papel definido dentro de um supra-sistema. A organização do sistema é feita centrando-se nas propriedades dinâmicas do mesmo, isto é, focalizando o comportamento externo ao sistema e, posteriormente, derivando unidades internas (subsistemas) que respondem especificamente aos estímulos provenientes dos outros sistemas.

O encapsulamento é introduzido finalmente como primeira atividade do *design* orientado a objetos, analisando o comportamento exibido pelos subsistemas. Nesta análise, são detectados padrões de comportamentos que se tornam objetos. As interações entre estes padrões sugerem associações estáticas entre os objetos. As propriedades dos objetos e das associações surgem da natureza dos padrões de comportamento e das interações.

2.2 Visão Geral da Proposta

A proposta de técnica de modelagem consiste em um processo para conduzir a construção de modelos de maneira a satisfazer os seguintes requisitos:

- Usar um enfoque sistêmico para contextualizar e definir inicialmente o sistema sob modelagem.
- Permitir construir um modelo do sistema considerando as propriedades dinâmicas do mesmo em sua relação com outros sistemas.
- Evitar a introdução do encapsulamento da orientação a objetos no início da modelagem.
- Permitir obter, finalmente, uma especificação do sistema sob modelagem nos termos da MOO.

A ênfase na descrição da proposta está no processo ou procedimento adotado, que faz desta proposta uma técnica de modelagem conceitual sob uma estratégia dirigida por comportamento.

O processo da técnica proposta pode ser dividido em três grandes passos, cada um dos quais apresenta diversas atividades, modelos e notações. Estes passos são os seguintes:

1. **Modelagem inicial do sistema**: Neste passo, são definidos o contexto em que opera o sistema e suas relações com outros sistemas, na forma de estímulos e respostas. Adicionalmente, é descrito o comportamento do sistema, particionando-o em unidades denominadas processos. Neste passo é usado o enfoque sistêmico e os conceitos da modelagem dinâmica.

2. **Construção de ciclos de vida**: Neste passo, os processos (ou partes deles) são integrados em modelos maiores, de forma a configurar ciclos de vida para diferentes tipos de dados. Neste passo são usados os conceitos da modelagem dinâmica.
3. **Derivação do modelo estrutural de objetos**: Neste passo, é derivado um modelo estático orientado a objetos a partir dos modelos dinâmicos construídos nos passos anteriores. Neste passo são usados os conceitos da modelagem estrutural orientada a objetos.

A seqüência destes passos é lógica, e não reflete necessariamente uma ordem cronológica aplicável na prática, já que podem existir muitas iterações que permitem modificar os resultados obtidos em passos anteriores.

A figura 2.1 mostra as entradas e saídas para estes passos. Os dois primeiros passos refletem a fase de modelagem conceitual da presente proposta. O terceiro e último passo é o início da fase de *design*. As setas descontínuas à esquerda, nesta figura e nas seguintes, indicam a possibilidade de iterações entre estes passos.

2.2.1 A Modelagem Inicial do Sistema

O ponto de partida para todo o processo de modelagem utilizado pela técnica proposta é a modelagem inicial do sistema. A entrada para este passo é o próprio domínio do problema, ou universo do discurso.

Também é possível que a entrada para este passo encontre-se estruturada como uma descrição em linguagem natural dos requisitos que o sistema deve satisfazer. A modelagem inicial pode ainda ser realizada conjuntamente com a elicitação dos requisitos do sistema sob modelagem. As formas de obter a entrada para este passo não fazem parte da proposta.

A modelagem inicial do sistema permite estruturar os conceitos do universo do discurso na forma de um sistema visto de uma perspectiva dinâmica. Em outras palavras, a modelagem inicial permite uma primeira aproximação do sistema conside-

rando seu aspecto comportamental, isto é, como ele reage a estímulos provenientes do exterior na forma de respostas para o mesmo.

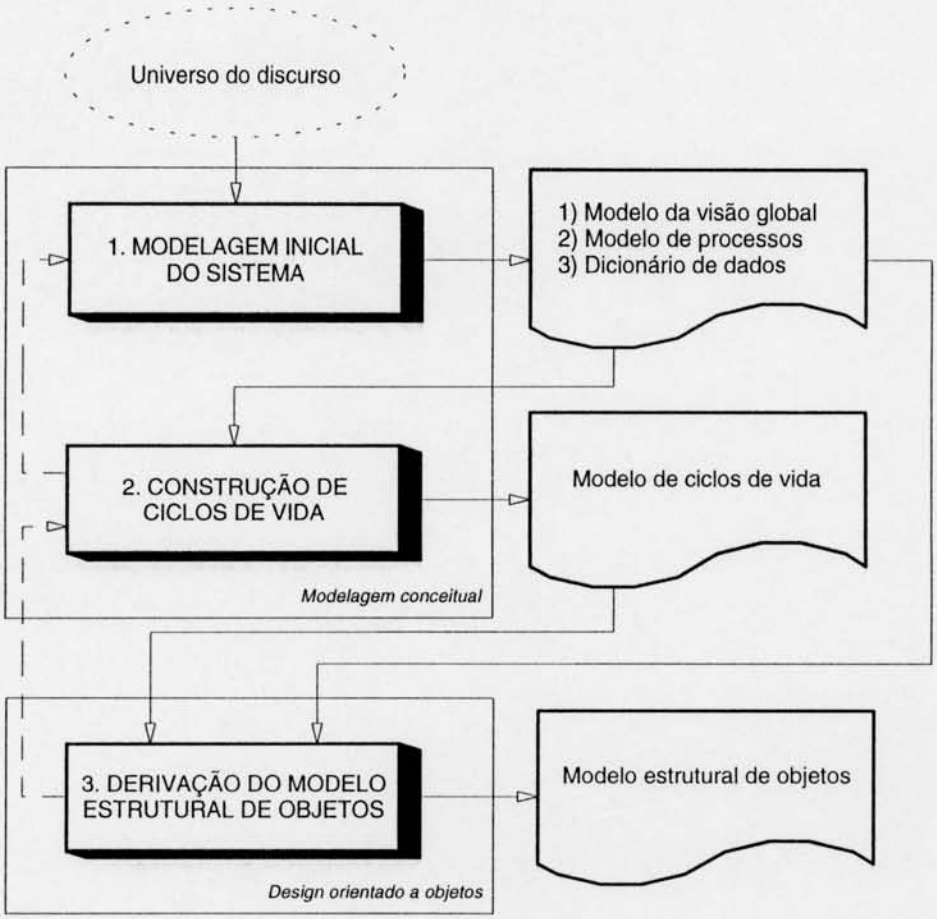


FIGURA 2.1 - Os grandes passos da técnica de modelagem e suas respectivas entradas e saídas.

As saídas da modelagem inicial do sistema são:

1. **Modelo da visão global do sistema**: Este modelo está constituído por dois diagramas:
   - **Diagrama de contexto do sistema**: É um diagrama que contextualiza o sistema sob modelagem, situando-o dentro de um supra-sistema que, por

sua vez, contém outros sistemas ou agentes externos com os quais o sistema sob modelagem se relaciona.

- **Diagrama da visão global**: É um diagrama que descreve o sistema como um conjunto de processos concorrentes, que recebem estímulos e geram respostas de/para os denominados agentes externos.

2. **Modelo de processos**: É um conjunto de diagramas, cada um dos quais descreve o comportamento de um dos processos do diagrama da visão global. Os diagramas do modelo de processos são construídos segundo o formalismo proposto, denominado *High-Level Statecharts* (*HLS*).

3. **Dicionário de dados**: É um conjunto de definições de composição e de domínios para todas as variáveis que são usadas nos diagramas da visão global e de processos. É adotada a notação do dicionário de dados da análise estruturada moderna.

Para obter estas saídas, este passo divide-se nas seguintes atividades:

1. **Construção do modelo da visão global**: Nesta atividade, é contextualizado o sistema, identificando qual é o supra-sistema que o contém, qual o papel que o sistema sob modelagem desempenha nele, quais os outros sistemas ou agentes externos com os quais ele se relaciona. São realizadas também uma primeira aproximação às unidades componentes do sistema (processos concorrentes) e a sua correlação com os estímulos e as respostas que o sistema recebe e emite para cada agente externo. Neste nível de abstração, os processos constituintes do sistema são caixas pretas. Todos os dados envolvidos no sistema são representados como variáveis cujos domínios são definidos.
2. **Construção do modelo de processos**: Nesta atividade, é decomposto cada um dos processos componentes do sistema, isto é, as caixas pretas são "abertas" considerando-se seu aspecto comportamental. Cada processo é explodido em uma rede hierarquizada de estados e transições, representada em um diagrama HLS. Caso novas variáveis apareçam (para necessidades de cálculos por exemplo), estas devem ter seus respectivos domínios definidos.

A figura 2.2 mostra as entradas e saídas destas atividades.

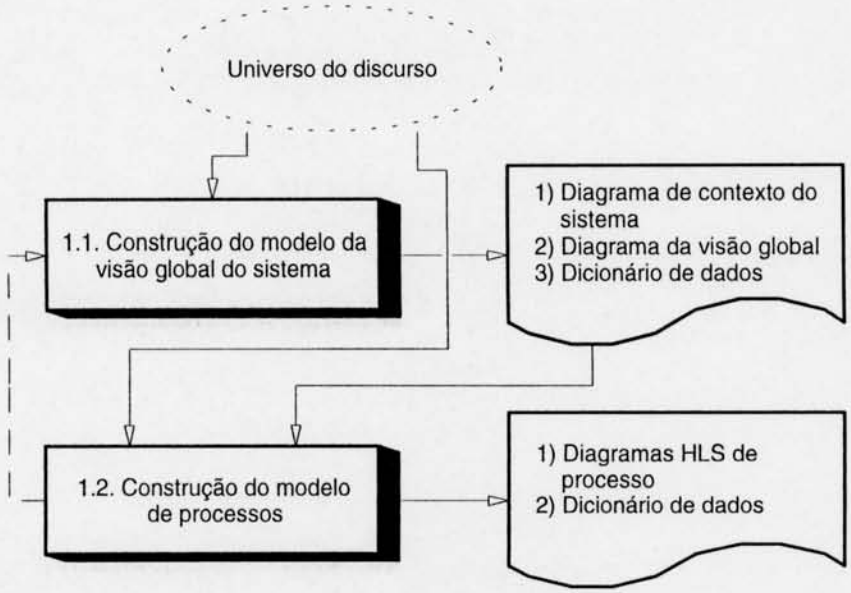


FIGURA 2.2 - As atividades da modelagem inicial do sistema e suas respectivas entradas e saídas.

2.2.2 A Construção de Ciclos de Vida

O segundo passo do processo de modelagem é a construção de ciclos de vida. Este passo usa como entrada os três documentos gerados no passo anterior:

- Modelo da visão global
- Modelo de processos
- Dicionário de dados

A construção de ciclos de vida é um processo de encadeamento de processos (ou partes de processos) que tratem do mesmo tipo de dado (representado por uma variável), de forma a configurar diagramas HLS maiores que descrevem toda a "história" ou o ciclo de vida para aquele tipo de dado. Este é um processo de desintegração de componentes organizados nos diagramas HLS de processos e posterior integração em diagramas HLS de ciclos de vida.

A saída da construção de ciclos de vida é um conjunto de diagramas HLS de ciclos de vida denominado modelo de ciclos de vida.

Para obter este modelo como resultado, este passo é dividido nas seguintes atividades:

1. **Identificação de estados e variáveis em comum**: Esta atividade agrupa os processos (ou partes de processos) em conjuntos que tratem do mesmo tipo de dado. Podem existir problemas de sinônimos (mesmos dados com nomes diferentes) ou homônimos (dados diferentes com o mesmo nome) em relação aos nomes das variáveis e estados. Estes problemas devem ser identificados e devidamente resolvidos. O resultado desta atividade são conjuntos de diagramas (ou partes de diagramas) HLS de processos que tratam dos mesmos tipos de dados.
2. **Conexão de pós e pré-estados**: Nesta atividade, é realizado um ordenamento (não necessariamente seqüencial) dos estados de diferentes processos (ou partes de processos) que tratam de cada tipo de dado, conectando os pós-estados (estados finais) de uns com os pré-estados (estados iniciais) de outros, representando os estados repetidos uma única vez. Obtém-se assim toda a "história" ou ciclo de vida dos distintos tipos de dados. Esta "história" de cada dado é denominada diagrama HLS preliminar de ciclo de vida.
3. **Verificação da consistência no uso de nomes de estados e variáveis**: Os diagramas HLS preliminares de ciclo de vida devem ser submetidos a uma verificação de consistência nos nomes dos estados e variáveis que usam e/ou referenciam. Qualquer inconsistência deve ser corrigida. Os diagramas HLS preliminares de ciclo de vida já verificados e corrigidos tornam-se os diagramas HLS definitivos de ciclo de vida ou, simplesmente, diagramas HLS de ciclo de vida.

A figura 2.3 mostra as entradas e saídas para estas atividades.

2.2.3 A Derivação do Modelo Estrutural de Objetos

O terceiro e último passo do processo de modelagem é a derivação do modelo estrutural de objetos. É neste passo que é introduzido o encapsulamento da orientação a objetos.

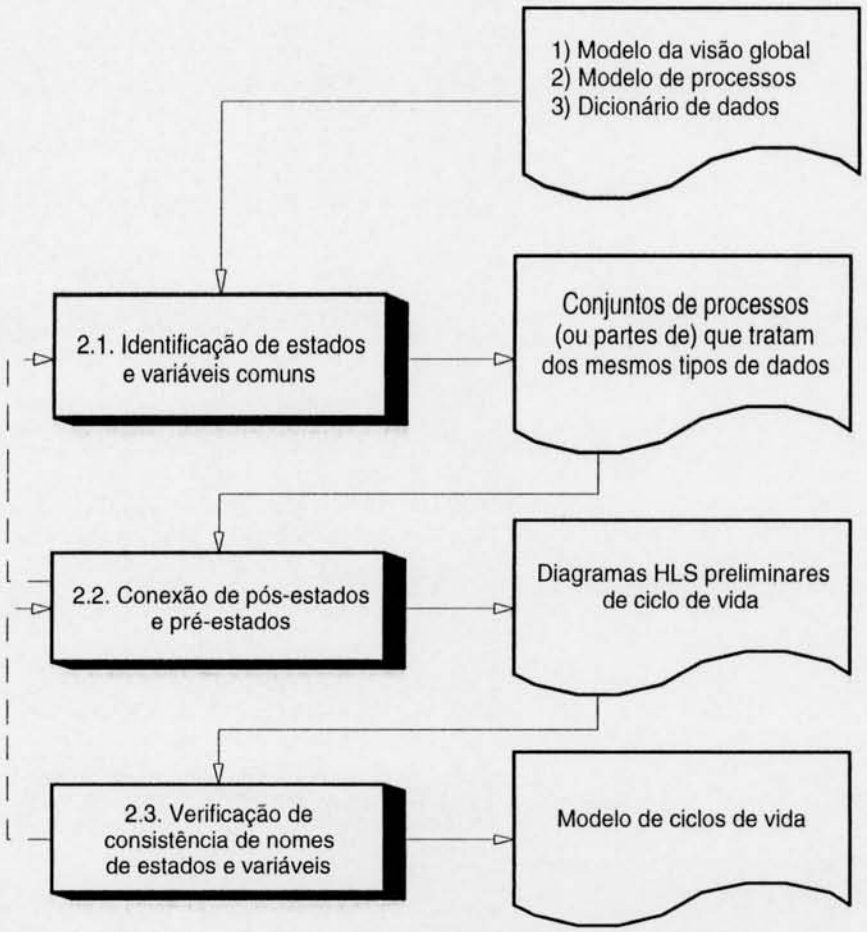


FIGURA 2.3 - As atividades da construção de ciclos de vida e suas respectivas entradas e saídas.

Este passo usa como entrada todos os documentos gerados nos passos anteriores:

- Modelo da visão global
- Modelo de processos
- Dicionário de dados
- Modelo de ciclos de vida

Na derivação do modelo estrutural de objetos, é feita uma identificação, nos modelos dinâmicos construídos nos passos anteriores, de todos os elementos que estruturam um modelo estático orientado a objetos. Estes elementos são as classes e os

objetos, as hierarquias de herança para as classes, as associações estáticas entre os objetos, os atributos e as operações.

O modelo estrutural de objetos é obtido ao realizar as seguintes atividades dentro deste passo:

1. **Identificação de objetos candidatos**: Os objetos candidatos são identificados a partir dos tipos de dados que possuem diagramas HLS de ciclo de vida, dos dados de entrada para o sistema (que aparecem nos estímulos) e de ações específicas realizadas no interior de cada processo.
2. **Derivação de hierarquias de herança**: As subclasses, dentro de uma hierarquia de herança, são identificadas por um processo de especialização a partir de comportamentos alternativos nos ciclos de vida. A ramificação do comportamento em um diagrama HLS de ciclo de vida é mapeada em uma hierarquia de herança.
3. **Determinação de associações estáticas**: As associações estáticas e os objetos participantes nelas são determinados pelos estímulos que o sistema recebe e pelas propriedades que estes apresentam. Uma análise da forma como os estímulos são tratados nos processos permite definir a cardinalidade das associações.
4. **Definição de atributos**: As descrições de dados componentes, incluídas no dicionário de dados, permitem definir atributos das classes. As variáveis usadas em cálculos dentro dos processos também podem ser definidas como atributos.
5. **Definição de operações**: Todas as ações que aparecem em cada um dos diagramas HLS de ciclo de vida servem para definir as operações (assinaturas dos métodos dos objetos).

A figura 2.4 mostra as entradas e saídas para estas atividades. Nesta figura, para simplificar, foram omitidas as setas descontínuas que representam as iterações entre as diferentes atividades.

2.3 Proposta em Relação à MOO

Para poder verificar se a presente proposta de técnica de modelagem é efetivamente MOO, é necessário analisar se as atividades mínimas, definidas no capítulo

anterior como parte deste tipo de modelagem, são realizadas ao longo do processo da proposta.

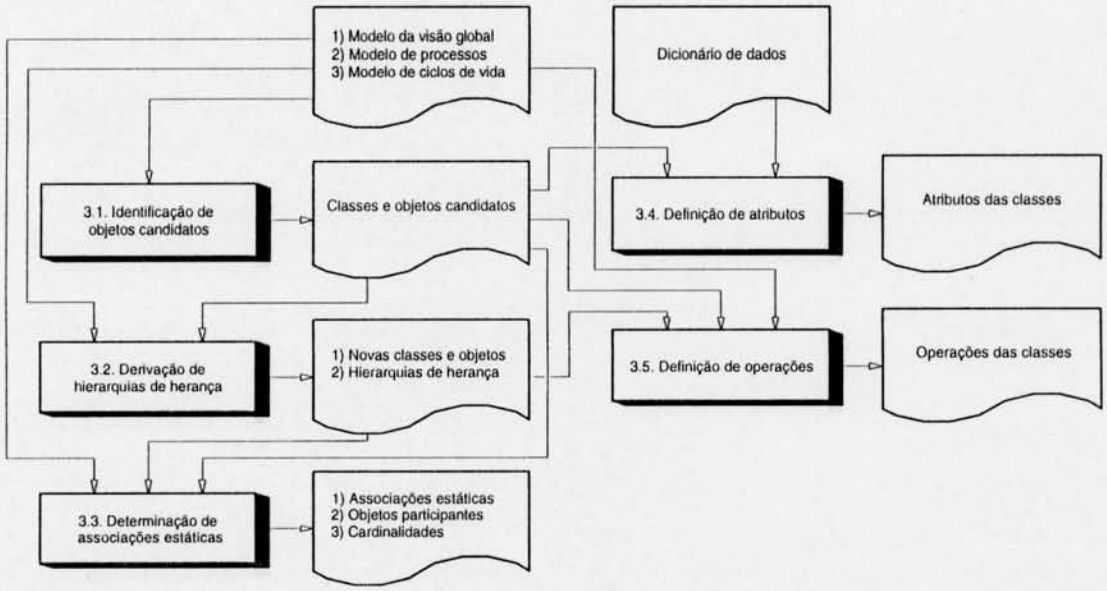


FIGURA 2.4 - As atividades da derivação do modelo estrutural de objetos e suas respectivas entradas e saídas.

As atividades em questão são as seguintes:

- identificar os objetos, os atributos e as classes,
- associar estaticamente os objetos,
- descrever o comportamento dos objetos,
- definir a colaboração do comportamento dos objetos,
- organizar as classes em hierarquias de herança e
- agregar e/ou particionar as classes por níveis de abstração.

As próximas seções descrevem onde e como cada uma destas atividades é desempenhada dentro do processo da técnica de modelagem proposta.

2.3.1 A Identificação dos Objetos, Classes e Atributos

Na técnica de modelagem proposta, a identificação das classes e objetos é realizada de acordo com três enfoques. Estes enfoques aparecem em ordem decrescente de utilização:

- **Enfoque de estado**: É o enfoque mais usado na técnica proposta. A partir dos estados que apresentam os diagramas HLS de processos, são deduzidos objetos que assumem estes estados.
- **Enfoque de abstração**: Os tipos de dados, que são representados por variáveis nos diferentes modelos dinâmicos, são outra fonte de objetos. Estas variáveis aparecem como parâmetros nos estímulos que o sistema recebe.
- **Enfoque de operação**: Este é o enfoque menos utilizado. Algumas ações específicas que aparecem nos modelos dinâmicos definem, apenas pelo fato de existirem, objetos aos quais pertencem como operações.

Identificar objetos, na verdade, quer dizer definir tipos de objetos. Neste sentido, a classificação fica implícita na identificação dos objetos.

Em relação aos atributos, estes são identificados com base nas definições dadas para os dados compostos e em algumas variáveis utilizadas para fins de cálculo nos modelos. Estas definições estão contidas no dicionário de dados.

2.3.2 A Associação Estática dos Objetos

A associação estática dos objetos é definida em função dos estímulos que o sistema recebe. As associações são especificadas indicando o nome da mesma, os objetos participantes e a cardinalidade de cada um.

As associações, como definido na técnica proposta, não apresentam direcionabilidade, padronização de tipos de associações, qualificadores, restrições ou definição de associações como tipos de coleções.

2.3.3 A Descrição do Comportamento dos Objetos

O comportamento do sistema é, primeiramente, modelado na forma de unidades denominadas processos. A partir do comportamento destes processos, são derivados os ciclos de vida que, por sua vez, definem os objetos do sistema.

Desta forma, os próprios ciclos de vida que serviram para identificar os objetos descrevem o seu comportamento. Contudo, nem todos os objetos identificados para o modelo estrutural possuem diagramas HLS de ciclo de vida. A ausência de descrições de comportamento para estes objetos é explicada pela própria organização do sistema (a partir dos estímulos), que não exige deles um comportamento mais complexo que deva ser descrito. Assim, pode-se afirmar que, dado que estes objetos apresentam um comportamento trivial, não é necessário descrevê-lo.

As principais características dos modelos que servem como descrição de comportamento (os diagramas HLS de ciclo de vida) são:

- Os objetos não apresentam concorrência interna.
- Apresentam eventos parametrizados e condicionados.
- Não apresentam verificação de efeitos colaterais das ocorrências de eventos em outras situações que não as definidas.
- São usados identificadores dos objetos.
- Não são modelados *threads* de comportamentos de exceção.
- A complexidade é hierarquizada com os conceitos de superestado/subestado.

2.3.4 A Definição da Colaboração do Comportamento dos Objetos

Na técnica de modelagem proposta, as colaborações não são definidas explicitamente. Elas se encontram implícitas por trás das associações estáticas e nos eventos e ações que aparecem nos modelos dinâmicos do sistema.

Seria necessária a construção de um modelo específico para este propósito, que usasse a informação já existente para representar as colaborações entre os objetos.

Apesar das colaborações serem implícitas, é possível definir algumas de suas características com base na informação contida nos modelos dinâmicos:

- As colaborações são da forma produtor-consumidor, isto é, baseadas no conceito de evento.
- Não é feita a distinção de comunicações apenas entre classes, entre classe e instância ou apenas entre instâncias.
- As mensagens (eventos) podem ser geradas por múltiplos objetos origens e/ou recebidas por múltiplos objetos destinos.
- As mensagens (eventos) podem ser entendidas como qualificadas pela presença de condições (*guards*).
- Não existe qualquer indicação de seqüenciação das mensagens (eventos), a não ser as próprias seqüências derivadas da estrutura dos estados e transições.
- Não são consideradas mensagens (eventos) de exceção, apenas de conformidade.
- São incluídos os identificadores dos objetos nas mensagens (eventos).
- Os agentes externos podem aparecer nos modelos de colaboração.
- Não existem formas de hierarquização da complexidade da colaboração.

2.3.5 A Organização das Classes em Hierarquias de Herança

As hierarquias de herança das classes são derivadas a partir do comportamento que estas apresentam. Considerando a forma desta derivação, o processo de construção destas hierarquias é de especialização (as subclasses são definidas a partir de uma superclasse).

As propriedades herdadas dentro desta hierarquia são atributos, operações, associações estáticas e comportamento. Apenas a colaboração, por não estar definida explicitamente, não é considerada. As propriedades herdadas, tal como definidas nos modelos, não podem ser modificadas ao passar de um nível da hierarquia para outro.

Pela forma em que as hierarquias de herança são derivadas, não é possível encontrar situações de herança múltipla.

A relação entre as subclasses que herdam de uma superclasse é sempre de exclusão. Eventualmente, se os comportamentos alternativos ocorrem bem no início do ciclo de vida, é possível ter também cobertura, com o que se definiria o particionamento da superclasse.

2.3.6 A Agregação/Particionamento das Classes por Níveis de Abstração

Mecanismos de agregação e/ou particionamento das classes por níveis de abstração não são considerados pela técnica proposta.

Em [BUS 94a] e [BUS 94b] é apresentada uma proposta de particionamento/agregação estrutural para a AOO que pode ser incluída como uma extensão aplicável ao modelo estrutural resultante. Esta proposta apresenta originalmente a combinação de estratégias *top-down* e *bottom-up*. Contudo, apenas esta última estratégia é aplicável para a técnica de modelagem. Nesta proposta de agregação estrutural, é feita uma adaptação dos mecanismos de *clustering*, desenvolvidos para a modelagem semântica de dados, notadamente para algumas extensões do modelo entidade-relacionamento [TEO 89].

A proposta de agregação estrutural baseia-se no conceito de *cluster*, que é uma abstração que permite esconder alguns detalhes definidos em um nível inferior. Mais especificamente, um *cluster* é um agregado de classes que apresenta atributos (os nomes das classes agregadas e relacionadas no nível inferior). Os *clusters* podem relacionar-se com outros *clusters* ou com classes através de associações comuns ou associações agregadas (duas ou mais associações comuns paralelas). A figura 2.5 apresenta as representações para classes, *clusters* e associações agregadas, baseadas na notação OMT [RUM 91] (vide também anexo "Resumo da Notação OMT para o Modelo de Objetos").

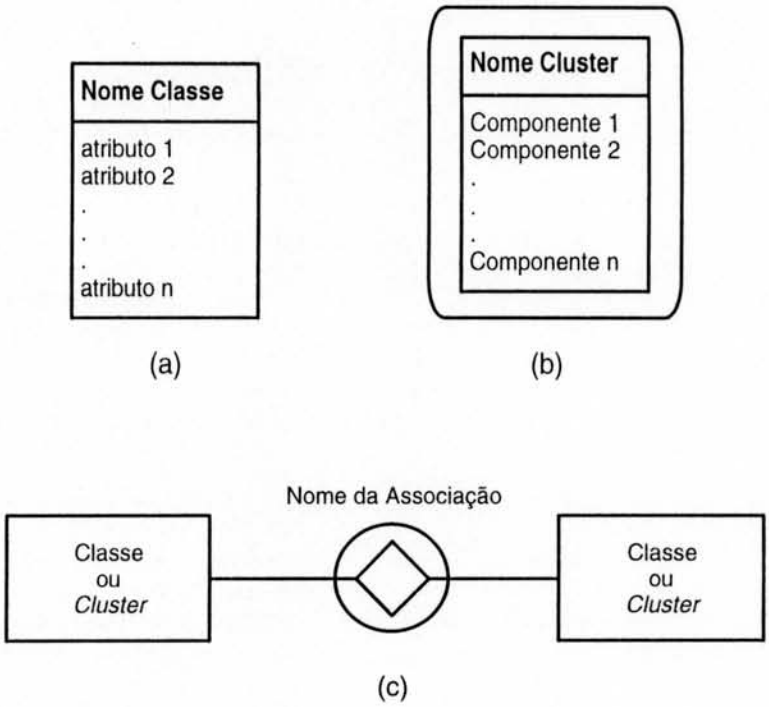


FIGURA 2.5 - Representações de: (a) uma classe comum [RUM 91], (b) um *cluster* e (c) uma associação agregada.

Os critérios de agregação são os seguintes:

- **Dominância**: Quando a maioria das associações em que uma classe participa é da forma um para muitos (1:n), esta classe é dominante e define o *cluster*.
- **Abstração**: Diversas hierarquias de abstração permitem definir a classe do topo da mesma como um *cluster*. As abstrações são: generalização/especialização, composição, agregação e participação.
- **Restrição**: A existência de alguma regra ou restrição que, implicitamente, relaciona duas ou mais classes. A restrição define o respectivo *cluster*.
- **Associação simples**: A existência de associações simples (unárias e binárias) de diversa cardinalidade (1:1, 1:n, n:m) permite realizar a abstração. A associação simples define o respectivo *cluster*.
- **Associação complexa**: A existência de associações complexas (ternárias ou superiores) permite abstrair. A associação complexa define o respectivo *cluster*.

Para quando existirem conflitos entre os critérios, isto é, quando, dado um determinado nível de abstração, for possível criar *clusters* de acordo com mais de um dos critérios acima, é definida uma prioridade entre os critérios, baseada no conceito de coesão do *cluster* ("força" interna que relaciona os componentes do mesmo). O espectro de coesões, apresentados de maior a menor, é o seguinte:

- **Nível 1 - Dominância**: É a mais alta coesão, devido à dependência funcional (ou ainda de existência) dos componentes dominados em relação ao dominante.
- **Nível 2 - Abstração**: É a segunda maior coesão, devido ao relacionamento hierárquico estabelecido entre os componentes na generalização/especialização, composição, agregação e participação.
- **Nível 3 - Restrição**: Existe uma coesão moderada entre os componentes relacionados através de relacionamentos restritos.
- **Nível 4 - Associação simples**: Existe um subespectro dentro deste nível, que vai da maior coesão nas associações unárias, passando pelas associações binárias 1:1, associações binárias 1:n e, finalmente, com a menor coesão, as associações binárias n:m.
- **Nível 5 - Associação complexa**: É a segunda menor coesão, devido à ausência de dominância de qualquer dos componentes que participam no relacionamento ternário ou superior.
- **Nível 6 - Sem associação**: A coesão é apenas decorrente do fato de que os componentes participam de um mesmo *cluster*, sem apresentar nenhuma associação entre eles.

A aplicação dos critérios é recursiva, isto é, a partir do nível de menor abstração, em que aparecem as classes e as associações do modelo estrutural, podem-se ir agregando, definir *clusters* de classes e depois *clusters* de *clusters*, até atingir o nível mais abstrato, em que todo o sistema é representado por um único *cluster*.

2.3.7 A Conclusão

Ao analisar o processo de modelagem proposto à luz das atividades da MOO, chega-se à conclusão de que a técnica proposta necessita de algumas extensões para que se possa referi-la como uma técnica de MOO. Estas extensões são:

- Descrições do comportamento para alguns tipos de objetos.
- Modelo de colaboração entre os objetos a partir da informação contida nas associações estáticas, nos eventos e nas ações.
- Mecanismo de agregação para representar o modelo estrutural de forma hierarquizada.

2.4 Resumo

O presente capítulo apresentou uma introdução à técnica de modelagem proposta neste trabalho. Primeiramente, foram descritas brevemente as bases da proposta, extraídas da ciência dos sistemas, da epistemologia e da ontologia.

A seguir, foram descritos os três grandes passos que constituem o processo de modelagem: a modelagem inicial do sistema, a construção dos ciclos de vida e a derivação do modelo estrutural de objetos. Na modelagem inicial, é definido o contexto do sistema e suas relações estímulo-resposta com outros sistemas. O sistema é decomposto em um conjunto de processos concorrentes, cada um dos quais tem seu comportamento modelado detalhadamente, usando o formalismo proposto HLS (*High-Level Statecharts*). A construção de ciclos de vida permite construir diagramas HLS de ciclo de vida para diferentes tipos de dados, cujos comportamentos encontravam-se distribuídos em distintos diagramas HLS de processos. Finalmente, a derivação do modelo estrutural de objetos é feita com informação contida nos modelos dinâmicos. São identificados e definidos classes e objetos, as hierarquias de herança, as associações estáticas entre os objetos, os atributos e as operações.

A proposta de modelagem também foi analisada na perspectiva da MOO. As seis atividades mínimas que a MOO apresenta não foram todas identificadas no processo de modelagem proposto, concluindo-se que a técnica precisa de algumas extensões para tornar-se uma MOO completa.

3 *HIGH-LEVEL STATECHARTS* (HLS)

High-Level Statecharts (*HLS*) é uma proposta de modelagem conceitual de sistemas de informação na perspectiva dinâmica destes sistemas. O presente capítulo mostra a sintaxe e a semântica dos HLS de maneira informal e apoiada em exemplos.

A notação base utilizada para representar os sistemas nesta perspectiva é uma extensão do formalismo de D. Harel [HAR 87] conhecido como *statecharts*, que por sua vez é uma extensão do formalismo convencional das máquinas de estados finitos e diagramas de transição de estados. As extensões propostas neste trabalho visam fornecer um maior poder de expressão e abstração aos *statecharts* de Harel no sentido de permitir modelar sistemas de informação tradicionais. Para tanto são acrescentadas anotações nos *statecharts* que serão descritas no decorrer deste capítulo.

Como exemplo de aplicação, para este capítulo e os restantes, será utilizado o problema padrão de preparação de congressos da IFIP (*International Federation for Information Processing*) proposto pelo *Working Group* 8.1 da IFIP e descrito em [OLL 82] como base para um congresso de estudo comparativo de metodologias de projeto de sistemas de informação. O caso da IFIP é um problema amplamente divulgado na literatura[11] e muito usado até hoje. Os exemplos mais complexos mantêm o texto em língua inglesa para evitar ambigüidades na tradução dos termos utilizados na sua definição. O anexo "Definição do Problema da IFIP" apresenta o texto original para este problema e o anexo "Modelo Completo do Problema da IFIP" apresenta todos os diagramas construídos para este problema.

Este capítulo está organizado como segue: a seção 3.1 mostra porque os *statecharts* tradicionais não são adequados para a modelagem de sistemas de informação; a seção 3.2 mostra as notações dos *statecharts* que são mantidas nos HLS; finalmente, a seção 3.3 descreve as extensões específicas propostas em termos de variáveis, estados e transições.

---

[11] Em [IFI 82] é possível encontrar este mesmo problema modelado com diversas técnicas.

3.1 Os HLS vs. os *Statecharts*

Os *statecharts* foram concebidos, segundo Harel [HAR 87], para "... especificar e projetar sistemas complexos de eventos discretos, tais como sistemas de tempo-real multi-computadores, protocolos de comunicação e unidades de controle digital". Este tipo de sistema é freqüentemente denominado reativo.

Os *statecharts* servem bem ao propósito de modelar o comportamento de um único elemento, como por exemplo um sistema como um todo, ou, ainda, para descrever o comportamento padrão para uma instância de uma classe qualquer. Quando for necessário representar *simultaneamente* um conjunto de instâncias, ou seja, quando os diferentes elementos apresentam comportamentos concorrentes ou exclusivos, e estes precisam interagir entre si ou com outros elementos que também possuam comportamento concorrente ou exclusivo, os *statecharts* não são mais apropriados. O próprio Harel sugere o que denomina diagramas com "estados parametrizados" para estes casos. Estes estados parametrizados podem ser da forma *or*-parametrizado (estados exclusivos) ou *and*-parametrizado (estados concorrentes). Em ambos os casos, os diferentes estados apresentam estrutura interna idêntica e são representados por um único estado com um "parâmetro". Harel sugere inclusive algumas representações pouco elaboradas para ambas as situações (vide figuras 3.1 a 3.3 para uma comparação).

Com base nessa idéia, os HLS são uma extensão dos *statecharts* para modelar sistemas de informação tradicionais, em que é comum encontrar tratamento de conjuntos de elementos. Os estados, nos HLS, apresentam assim variáveis associadas (pertencentes a domínios específicos) que servem à maneira dos "parâmetros" dos estados sugeridos por Harel. Os estados concorrentes e exclusivos apresentam notações diferentes que serão descritas nas próximas seções.

Para exemplificar esta diferença entre HLS e *statecharts*, na figura 3.1 é apresentado o modelo de um problema adaptado de [HAR 87]. O problema trata sobre o *display* de dígitos em um relógio digital que reage à pressão de botões. Supõe-se quatro botões: *set*, *up*, *down* e *next*, que permitem, respectivamente, ativar o *display* do dígito, incrementar o dígito em 1, decrementar o dígito em 1 e abandonar o *display* do dígito. Durante as ações dos botões *up* e *down*, o *display* do dígito deve estar piscando (*flashing*). Quando for abandonado o *display*, este deve ficar fixo.

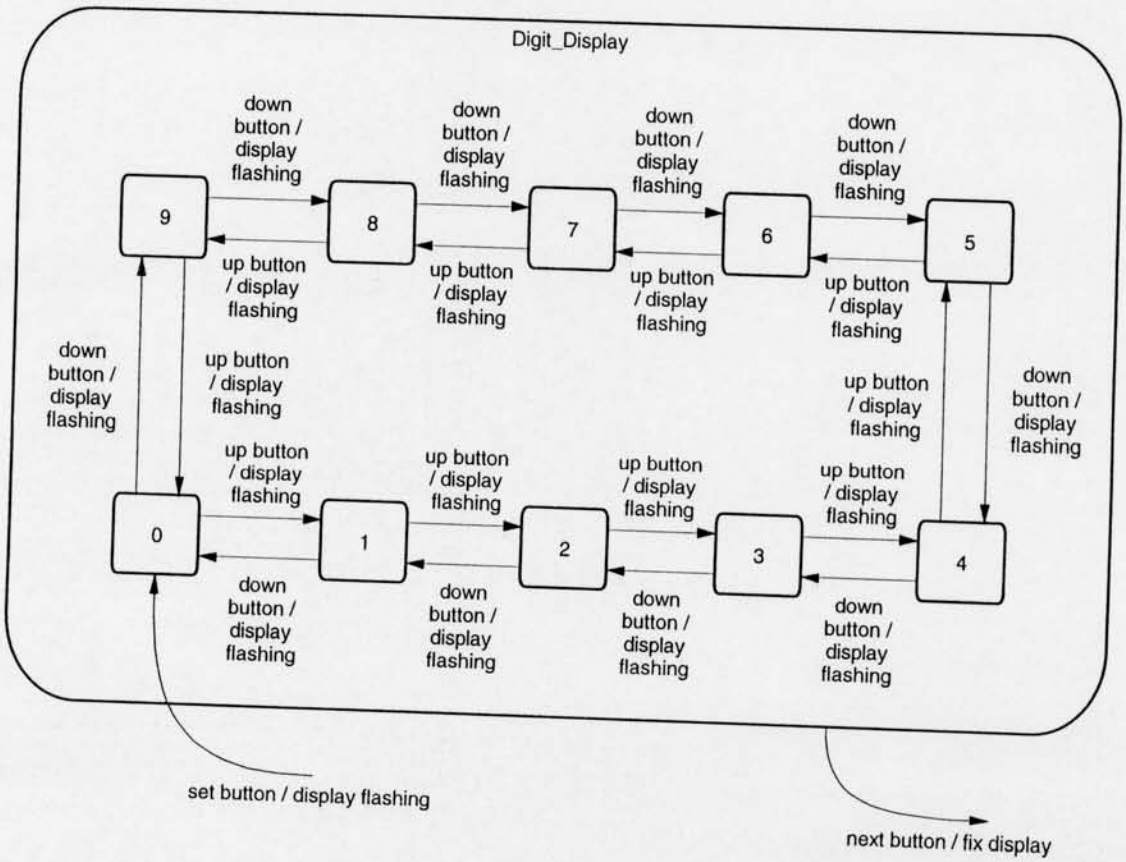


FIGURA 3.1- *Statechart* para o problema do *display* de dígitos em um relógio digital (adaptado de [HAR 87]).

Usando os *statecharts*, os estados e superestados são representados com caixas de cantos arredondados. As transições de estado são representadas pelas setas, nas quais são anotados, antes e depois da barra, respectivamente, o evento que dispara a transição e a ação que é realizada durante a transição. Assim, por exemplo, estando no estado 7, a transição que leva para o estado 6 é disparada se ocorre o evento down button, realizando-se, simultaneamente, a ação display flashing. A transição do evento set button, na parte inferior esquerda da figura, conduz ao estado 0, e a transição do evento next button, na parte inferior direita da figura, permite abandonar o diagrama independentemente do estado em que se encontre.

A figura 3.2 apresenta o mesmo problema modelado de acordo com as extensões sugeridas por Harel [HAR 87]. É definido o domínio para a variável i, que parametriza o estado Digit. As transições genéricas são especificadas nos losangos.

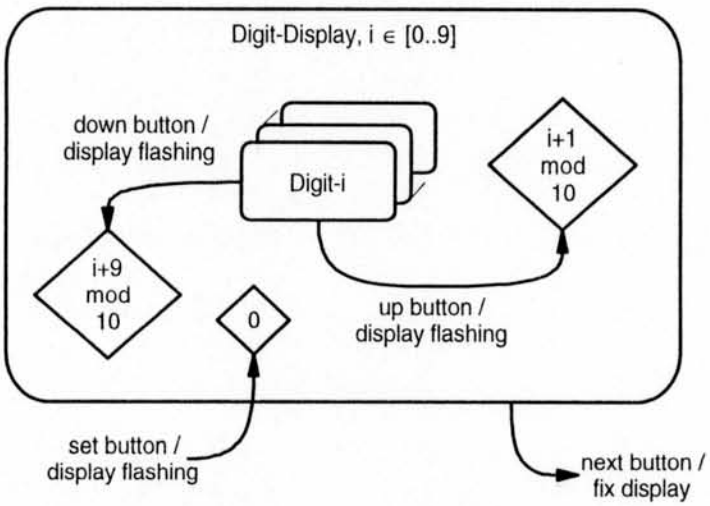


FIGURA 3.2 - *Statechart* estendido conforme Harel [HAR 87], para o problema do *display* de dígitos em um relógio digital.

A figura 3.3 mostra a mesma situação da figura 3.1, representada, agora, com HLS, o que permite comparar com a extensão de Harel da figura 3.2. Neste diagrama HLS, o conjunto de estados exclusivos é representado por um único estado genérico exclusivo Digit(i) (o fato de ser um estado exclusivo é indicado pelos colchetes [ ]), onde a variável i é a que "parametriza" este estado. As transições são determinadas por meio de *funções de mapeamento*. Estas funções são uma extensão introduzida para os HLS, e permitem especificar genérica ou especificamente os estados destinos das transições em que aparecem através do novo valor de sua respectiva variável associada. No diagrama, as funções de mapeamento são anotadas depois dos eventos e separadas por dois pontos (:). Um exemplo de função de mapeamento genérica é i+1 mod 10, e, de específica, 0.

Adicionalmente, o domínio da variável i, usada como parâmetro ou referência para o diagrama, deve ser definido em um dicionário de dados (no caso da extensão de Harel, é definido no próprio superestado), por exemplo, da seguinte forma:

$$i = 0..9$$

Este pequeno exemplo dá uma idéia do poder de expressão que é conseguido com os HLS. Atinge-se com eles um nível mais alto de abstração para facilitar a modelagem dos sistemas de informação; daí o nome adotado de *high-level statecharts*.

Todos estes conceitos e suas respectivas notações, que estendem os *statecharts* e que constituem os HLS, serão descritos em detalhe nas próximas seções.

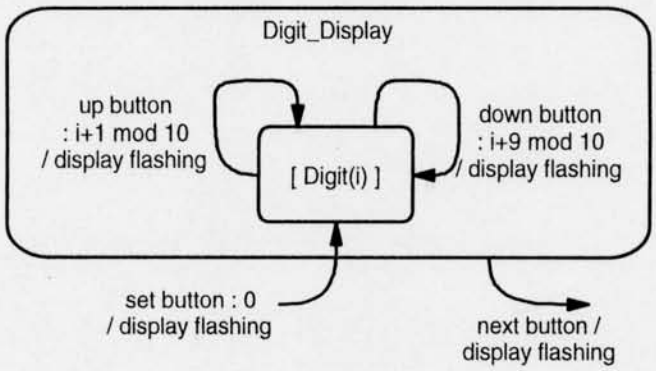


FIGURA 3.3 - HLS para o problema do *display* de dígitos em um relógio digital.

3.2 As Notações Mantidas dos *Statecharts*

Muitas das notações dos *statecharts* são mantidas nos HLS. Elas são as seguintes [HAR 87]:

- Superestados, estados e subestados como caixas arredondadas. Os superestados Y, U e V e os estados A, B, C, D e E da figura 3.4 são alguns exemplos.
- Transições como setas entre estados. As transições entre os estados A e B da figura 3.4 são alguns exemplos.
- Eventos anotados junto às transições que disparam. Exemplos na figura 3.4 são d, e, f, g e h.
- Transições com origens, destinos ou eventos comuns anotadas com um ponto de união. A transição com origem comum no estado C e estados destinos D e E da figura 3.4 é um exemplo.
- Estados exclusivos representados simplesmente dentro de um superestado. Os estados A e B são exclusivos dentro do superestado U (ou U = A XOR B) da figura 3.4. O mesmo acontece com os estados C, D e E em relação ao superestado V (V = C XOR D XOR E).

- Estados concorrentes representados em forma separada por linhas descontínuas dentro de um superestado. Os estados U e V, na figura 3.4, são concorrentes dentro do superestado Y (ou Y = U AND V).
- As condições continuam a aparecer, mas sofrem uma ligeira modificação na representação. Nos *statecharts*, são anotadas entre parênteses ( ), e nos HLS são anotadas entre colchetes[12] [ ]. Na figura 3.4, p e q são exemplos de condições .
- O caso particular de condição da forma in Estado, que verifica se o Estado está ativo ou não. A especificação do Estado pode requerer, para evitar ambigüidades, antepor os nomes dos superestados aninhados e separados por um ponto. A condição in V.C da figura 3.4 é um exemplo.
- Ações realizadas durante as transições e anotadas após o evento separadas por uma barra ("/"). Alguns exemplos de ações da figura 3.4 são a e b.
- Transições condicionadas, indicadas com um círculo com a letra C. Um exemplo, na figura 3.4, é o da transição associada ao evento h, com as condições q e NOT q.

Algumas características dos *statecharts* não são recomendadas nos HLS. Algumas têm algum tipo de conflito com as extensões descritas mais adiante, enquanto outras são muito raras nos domínios dos sistemas de informação. Estes características são:

- estados e transições *defaults* de ingresso aos *statecharts*;
- *resets* e escopos de *resets* que força o modelo ou partes do mesmo a retornar aos seus estados *default*;
- transições *históricas* (*enter-by-history*) e todas as notações associadas, que representam o ingresso no último estado visitado na ativação imediatamente anterior;
- transições de *seleção* (*selection*) onde o estado a ser ativado depende do valor de um evento genérico;
- *delays* e *time-outs* concebidos expressamente para os sistemas de tempo-real e

[12] Esta notação foi adotada para não confundi-la com os parâmetros dos eventos. Vide seção 3.3.3.1 para mais detalhes.

- atividades não associadas a transições, que são realizadas pelo sistema e que podem demandar algum tempo para serem executadas.

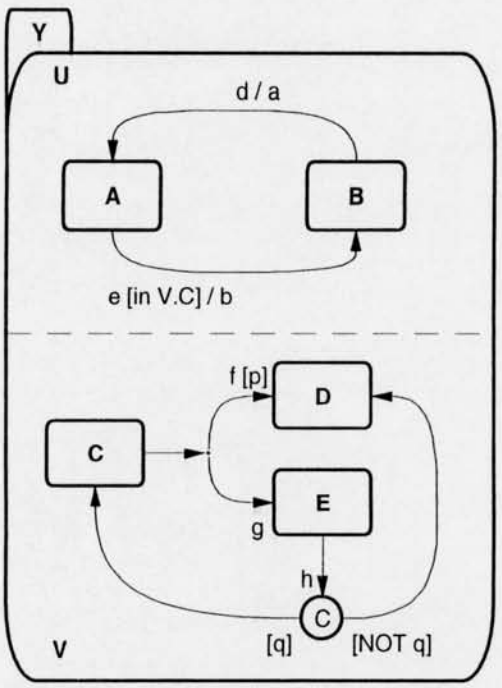


FIGURA 3.4 - Exemplo genérico que mostra todas as notações dos *statecharts* válidas também para os HLS.

3.3 EXTENSÕES PROPOSTAS AOS *STATECHARTS*

Os HLS são uma extensão dos *statecharts* com o fim de facilitar a modelagem de sistemas de informação tradicionais. A principal extensão é a possibilidade de representar conjuntos de comportamentos em forma simultânea, sejam estes concorrentes ou exclusivos, usando variáveis associadas aos estados.

A presente seção mostra, detalhadamente, as extensões introduzidas nos *statecharts*. As extensões foram agrupadas em: aquelas relativas às variáveis que representam tipos de dados, aquelas relativas aos estados e aquelas relativas às transições entre os estados.

3.3.1 As Variáveis nos HLS

As variáveis usadas nos HLS representam tipos de dados. Um tipo de dado é um tipo de objeto[13], entidade, conceito ou item de conhecimento pertencente ao sistema, cujo comportamento está sendo modelado com os HLS. Por exemplo, alguns tipos de dados podem ser os artigos submetidos ao congresso e os autores destes artigos.

Todas as variáveis são tipificadas e pertencem a domínios específicos definidos em um dicionário de dados. As variáveis podem ser categorizadas em:

- **Identificadores únicos**: são variáveis escalares e seus valores no domínio devem ser simples (não compostos) e, na medida do possível, enumeráveis. Este tipo de variável deve servir como identificador único para variáveis compostas (vide mais adiante). Por exemplo, artigo pode assumir os valores[14] a1, a2 e a3, que devem servir como identificadores únicos para os artigos submetidos ao congresso. Os nomes destas variáveis são constituídos por frases com palavras em minúscula e separadas pelo símbolo de sublinhado ("_"). Por exemplo, artigo, artigo_selecionado e autor.
- **Escalares**: são aquelas variáveis de estrutura simples (não composta), mas que não necessariamente são identificadores únicos. A convenção para os nomes é a mesma dos identificadores únicos. Exemplo: título_artigo e nome_autor.
- **Compostas**: são aquelas variáveis de estrutura composta, isto é, que possuem dados componentes. Este tipo de variável não pode ser um identificador único, mas pode ter um dado componente específico como tal. Por convenção, os nomes das variáveis compostas usam o prefixo dados_de em português ou o sufixo _data em inglês para diferenciá-los dos identificadores únicos. Exemplos: dados_de_artigo e paper_data.
- **Listas**: são aquelas que representam uma lista ordenada de variáveis do mesmo tipo (identificadores únicos, variáveis escalares ou compostas). Por convenção, as listas aparecem sempre com a variável componente entre chaves { }, usando a notação de repetição do dicionário de dados da análise estruturada [YOU 90]. Exemplos: {artigo} e {dados_de_autor}.

---

[13] No sentido de objeto da modelagem e não no sentido da orientação a objetos.

[14] A grande maioria das vezes não é necessário definir os valores do domínio das variáveis, simplesmente porque não são conhecidos no momento da modelagem.

Por exemplo, é diferente a variável author da variável {author} e da variável author_data. No primeiro caso, author indica apenas uma variável escalar ou, ainda, um identificador único para um autor dado; no segundo caso, trata-se de uma lista de autores que são anotados como uma repetição da variável author; no último caso, pode corresponder à seguinte descrição que deve existir em um dicionário de dados (usando a notação da análise estruturada moderna [YOU 90]):

author_data = @author + name + institution + address

onde o símbolo @ indica que o dado componente author é um identificador único da variável composta author_data e name, institution e address são variáveis escalares componentes.

3.3.2 Extensões Relativas aos Estados

No caso dos HLS, os estados são entendidos como estágios ou passos dentro da história de uma variável[15], isto é, cada estado é uma situação específica ou um valor[16] da enumeração do domínio de estados em que uma variável pode estar e determina o comportamento desta variável. Se o comportamento muda, é porque houve uma mudança de estado.

Alguns dos estados de um artigo podem ser Submetido e Distribuído. Um artigo no estado Submetido comporta-se diferentemente de um outro artigo no estado Distribuído. Por exemplo, o primeiro não pode ser aceito no congresso, mas pode ser distribuído para avaliação; o segundo não pode participar em uma sessão de apresentação, mas pode ser avaliado.

Por convenção, nos HLS os nomes dos estados são frases constituídas por palavras que começam com maiúsculas e são separadas pelo símbolo de sublinhado ("_"). Por exemplo, Estado_Do_Artigo.

---

[15] Os termos dado, tipo de dado e variável são usados indistintamente, salvo indicação em contrário.

[16] Um estado não deve ser interpretado como um atributo ou campo no qual serão armazenados os valores do estado. Muitas vezes os valores dos estados dependem de um ou mais atributos próprios ou ainda externos. Todas as alternativas anteriores são possíveis implementações do conceito de estado aqui apresentado.

3.3.2.1 Os Estados Atômicos como Estados Passivos

Os estados atômicos, isto é, aqueles que definidos no último nível da hierarquia de estados nos HLS, são interpretados como *passivos* [DEC 93]. Durante a permanência em um estado passivo não ocorre nenhum tipo de atividade, ação ou processo. Nada muda quando um sistema encontra-se em um estado passivo, à exceção do tempo. Geralmente, os nomes mais adequados para os estados passivos atômicos assumem as formas verbais no particípio passado, como os exemplos de estado de artigo citados acima: Submetido e Distribuído.

A categoria complementar à dos estados passivos é a dos estados *ativos*. Um estado é dito ativo se, durante a permanência do sistema neste estado, ele se encontra realizando algum tipo de processo. É difícil definir exatamente quando se entra ou sai de um estado ativo, por estar este vinculado à execução de uma atividade. Por isto, apesar de acrescentar um maior poder de expressão ao formalismo, não são usados nos HLS como estados atômicos. Geralmente, estes tipos de estados são melhor representados por formas verbais no gerúndio, como por exemplo *calculando* ou *distribuindo*.

Contudo, aqueles superestados (portanto estados não atômicos) que apresentem estados atômicos relacionados por meio de transições com ações associadas, podem ser interpretados como estados ativos, porque durante a permanência neles podem ocorrer ações internas. Os nomes destes superestados, por não serem passivos nem atômicos, podem assumir qualquer combinação de palavras apropriada para a representação.

Em síntese, todos os estados atômicos nos HLS são passivos e os superestados (abstrações) construídos em base a estes estados passivos podem ser ativos. No caso de um estado, que pode ter sido considerado inicialmente como atômico, precise alguma decomposição adicional, ele pode deixar de ser passivo para tornar-se ativo neste processo de decomposição.

Se eventualmente for preciso representar a realização de alguma ação durante a permanência em um estado, isto pode ser emulado, para o caso de ações que possam ser interrompidas, através de um estado passivo e uma auto-transição (transição com o mesmo estado origem e destino), como é mostrado na figura 3.5. No exemplo deseja-se que a ação a seja realizada durante a permanência no estado passivo F, bastando

para isto introduzir a auto-transição que aparece na figura. O evento e ou a condição [c] podem inclusive ser omitidos, mas não ambos simultaneamente[17].

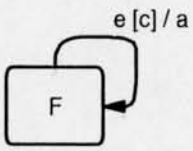


FIGURA 3.5 - Emulação de um estado ativo [DEC 93].

3.3.2.2 Os Estados com Variáveis de Referência

Uma das principais extensões introduzidas nos *statecharts* é o uso de variáveis de referência como "parâmetros" em alguns estados, no sentido de Harel [HAR 87]. Uma variável de referência é um identificador único associado a um estado (superestado), de tal forma que a estrutura que este estado (superestado) apresenta é válida para todos os valores que a dita variável pode assumir. Desta forma, o estado com variável de referência mostra o comportamento de uma forma genérica para esta mesma variável de referência.

Observe-se que um estado que tenha uma variável de referência associada define um conjunto de estados concorrentes, onde cada um dos estados representa o comportamento específico para um dos valores da variável de referência.

A variável de referência aparece sempre anotada entre parênteses ( ) após o nome do estado que está sendo "parametrizado", isto é, Nome_Do_Estado(variável_de_referência). Por exemplo, para o caso indicado mais acima, o artigo que possui os estados Submetido e Distribuído, um superestado possível é Estado_Do_Artigo que inclua ambos os estados de forma exclusiva. A variável de referência é justamente artigo, como é mostrado na figura 3.6 (por simplicidade, nesta figura foram omitidos todos os elementos associados à transição).

---

[17] No caso de que a ação deve estar sempre ocorrendo, o evento pode ser omitido e a condição fixada para true.

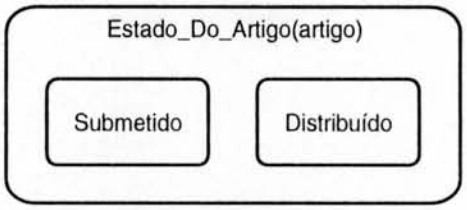


FIGURA 3.6 - Exemplo de uso de uma variável de referência em um HLS.

O exemplo da figura é interpretado como segue: o superestado Estado_Do_Artigo possui os estados exclusivos Submetido e Distribuído. Este superestado tem associada a variável de referência artigo, isto quer dizer que o diagrama HLS da figura é válido para qualquer dos elementos do domínio definido para esta variável. Independentemente dos valores específicos do seu domínio que possa assumir a variável, todos eles possuem o mesmo comportamento expresso genericamente no diagrama.

Subentende-se que, dado que o superestado tem associada uma variável de referência, esta mesma variável de referência está associada a todos os subestados nos sucessivos níveis de aninhamento. Assim, não é necessário incluir uma mesma variável de referência em todos os subestados aninhados de um superestado que já a possua. No caso do exemplo da figura 3.6, a variável artigo está associada tanto ao superestado Estado_Do_Artigo como aos estados Submetido e Distribuído, pelo fato de serem estes subestados do primeiro.

A equivalência entre *statecharts* e HLS para estes tipos de situações é apresentada no exemplo genérico da figura 3.7. Esta figura mostra em: (a) o modelo usando os *statecharts* comuns; (b) a notação para o superestado com variável de referência nos HLS. No caso do HLS, o Superestado(variável) dos HLS da parte (b) da figura representa um conjunto de estados concorrentes. O diagrama HLS mostrado é genérico e válido para todos os elementos do conjunto. A equivalência em *statecharts* pode ser vista na parte (a) da mesma figura. A rigor, não existe representação para o caso mostrado na figura; cada estado concorrente deveria ser representado, mas, como se desconhece o número total deles, deixa-se expresso na forma genérica mostrada na figura, forma genérica esta que, na realidade, não existe nos *statecharts*.

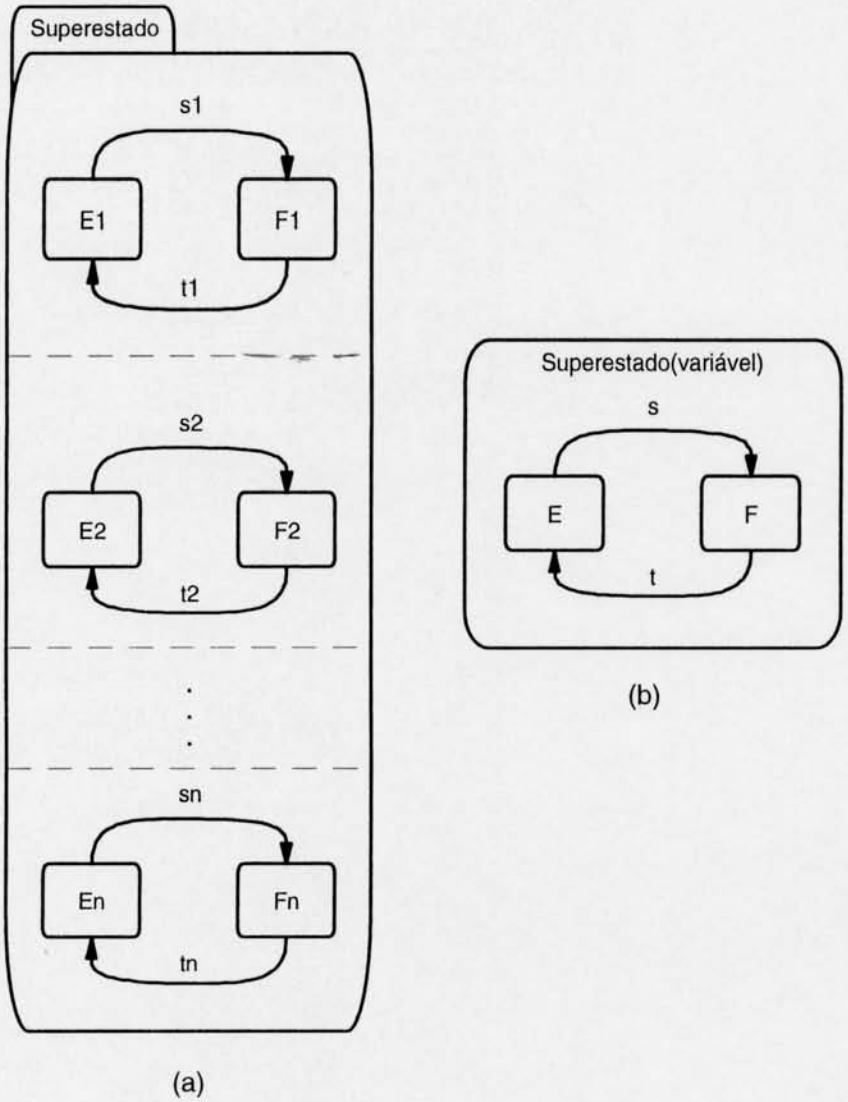


FIGURA 3.7 - Exemplo genérico de um superestado com subestados concorrentes representado com: (a) *statecharts* comuns; (b) diagrama HLS.

É importante atentar para a seguinte questão: no momento de modelar, verificar e/ou simular o sistema, apesar de ser o diagrama HLS genérico para todos os valores da variável de referência, pode ser necessário instanciar estas variáveis de tal maneira que os valores da variável que possam aparecer nas transições (mais especificamente, nos eventos, funções de mapeamento, condições e ações) casem com os valores da variável de referência envolvida. Esta relação de valores será tratada na seção 3.3.3.1.

3.3.2.3 Os Superestados como Conjuntos de Estados Exclusivos

Um superestado como um conjunto de estados exclusivos é uma estrutura em que todos os elementos possuem comportamento idêntico e são representados genérica e simultaneamente em um diagrama HLS. A figura 3.8 mostra as equivalências entre o superestado com subestados exclusivos usando *statecharts* comuns (caso (a)) e com diagramas HLS (caso (b)). No caso do HLS, os subestados exclusivos são indicados como [E(variável)], onde o colchete[18] [ ] representa a exclusividade (XOR). No *statechart* da parte (a), aparecem as transições com os eventos t1, t2, ..., tn-1 entre os estados exclusivos E1, E2, ..., En que têm seu equivalente na transição t : variável+1 mod n, onde t é evento genérico da transição e variável+1 mod n é a função de mapeamento que especifica outro estado exclusivo destino (através do valor resultante da variável) dentro do conjunto representado por [E(variável)]. O domínio de variável (= 0..n) deve também estar definido.

Como exemplos específicos destes tipos de superestados vide as figuras 3.1 e 3.3.

Note-se que a representação de exclusividade nos HLS altera a notação das transições entre os estados exclusivos dos *statecharts*, sendo sempre necessário definir uma função de mapeamento entre estados.

É claro que esta estrutura pode se apresentar em qualquer combinação possível com os estados concorrentes mostrados previamente. Assim, por exemplo, conjuntos de estados concorrentes e exclusivos podem existir em paralelo, ou, ainda, um estado pertencente a um conjunto concorrente pode ter uma estrutura exclusiva, ou vice-versa.

---

[18] Esta notação foi também tomada do operador de seleção usado no dicionário de dados da análise estruturada moderna.

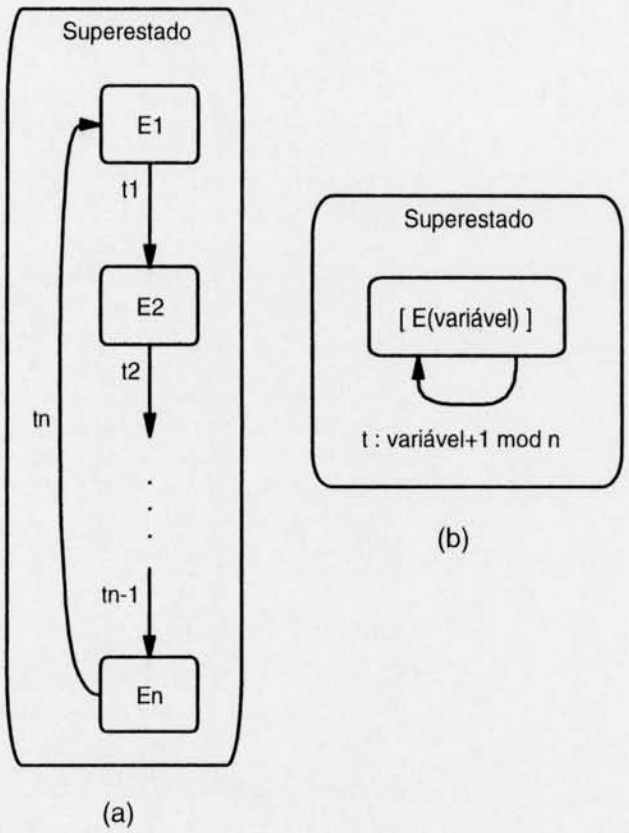


FIGURA 3.8 - Exemplo específico de um superestado com subestados exclusivos representado com: (a) *statecharts* comuns; (b) diagrama HLS.

3.3.3 Extensões Relativas às Transições

Toda transição de estado em um HLS está composta pelos seguintes elementos (alguns deles podem ser omitidos):

1. Estados de origem e destino, que são indicados pelo sentido da seta que representa a transição.
2. Evento que dispara a transição e variáveis associadas como parâmetros.
3. Função de mapeamento, que indica o valor da variável do estado destino da transição.
4. Condição que determina o disparo da transição em função do seu valor_verdade.
5. Ação que é executada junto ao disparo da transição.

Estes elementos são anotados nas transições de acordo com a seguinte convenção[19]:

evento : função_de_mapeamento [condição] / ação

Conforme o caso, cada um destes elementos pode ser omitido em uma transição, sendo que pelo menos o evento ou a condição deve sempre estar presente.

Quando os elementos são aplicáveis a um conjunto de transições concorrentes, estes são anotados entre chaves[20] { }:

{ evento : função_de_mapeamento [condição] / ação }

Esta situação é descrita na próxima seção.

3.3.3.1 O Evento e seus Parâmetros

Um evento é um fato relevante para o sistema, pois provoca algum tipo de reação nele. O evento pode ter sido originado no exterior ou no interior do mesmo. O evento está associado a uma ou mais transições, de tal forma que, quando o evento ocorre, a(s) transição(ões) pode(m) disparar (dependendo ainda das condições). O evento é, portanto, o principal elemento que desencadeia as mudanças de estado e, desta forma, determina o comportamento do sistema.

É importante destacar que é possível que o evento seja omitido em uma transição. Quando isto ocorre, o disparo da transição depende apenas da condição; portanto, esta sempre deve existir se o evento for omitido.

Os eventos são anotados como o primeiro elemento junto às setas que representam as transições. Seu nome é uma frase constituída por palavras em minúscula, separadas pelo símbolo de sublinhado ("_"). As frases devem começar com um verbo que pode ou não apresentar um objeto. Por exemplo, seriam nomes de eventos submeter, enviar_convite e gerar_lista_final_de_participantes. Esta convenção de nomes tem apenas o propósito de uniformizar os eventos e podem ser utilizadas outras convenções

---

[19] Existe uma notação adicional para esta forma geral e ocorre quando são representadas ações associadas a conjuntos de transições. Vide seção 3.3.3.4 para mais detalhes.

[20] Esta notação também é tomada do operador de repetição utilizado no dicionário de dados da análise estruturada moderna.

que venham a manter a mesma homogeneidade desejada. Por exemplo, pode-se optar por nomes na forma nominal tais como submissão, envio_convite e geração_lista_final_de_participantes[21].

Muitas vezes pode ser necessário que um ou mais dados acompanhem o evento para serem tratados no modelo ou para se relacionarem especificamente com a variável de referência do superestado. Estes dados associados são os parâmetros dos eventos e são representados por variáveis de qualquer tipo.

As variáveis usadas como parâmetros são anotadas entre parênteses ( ) e separadas por vírgula (","). Exemplos de eventos com parâmetros são: selecionar_moderador(pessoa), programar(sessão, {artigo}) e submeter(dados_de_artigo, {dados_de_autor}).

Um caso particular de parâmetro é o chamado parâmetro nulo. Os parâmetros nulos são usados quando um evento complexo, com dois ou mais parâmetros, é dividido ou decomposto em eventos mais simples, cada um tratando de um dos parâmetros em subestados concorrentes dentro de um diagrama HLS. Neste caso, o parâmetro que não está sendo tratado em um subestado dado, isto é, que não está sendo usado na função de mapeamento, na condição, ou na ação, é anotado como null. A utilização deste tipo de parâmetro será descrita em detalhe no próximo capítulo.

Como alguns eventos apresentam parâmetros, é necessário manter uma relação de consistência, na hora de modelar, com os estados dos HLS que possuem variáveis de referência. Como estes estados somente são definidos para os valores do domínio de suas variáveis de referência, devem ser justamente estas as que devem aparecer, nos eventos, como parâmetros.

Por exemplo, a figura 3.9 mostra o diagrama HLS Paper_Selection no qual existe um superestado denominado Paper_Status(paper). Este estado possui a variável de referência paper e portanto é definido para os valores desta variável. Assim, o evento select(paper, decision) tem como primeiro parâmetro justamente a variável paper. Neste sentido, este diagrama é consistente, porque tanto o superestado quanto o

---

[21] Neste caso, os nomes dos estímulos (vistos no próximo capítulo) também deveriam assumir esta convenção (vide seção 4.2.3) e os nomes dos processos (vistos também no próximo capítulo) devem ser mudados para a forma verbal no infinitivo (vide seção 4.2.3).

evento que provoca a transição entre seus estados referem-se à mesma variável paper. Para verificar ou simular este modelo, podem-se instanciar os parâmetros do evento. Como o superestado é válido para todos os valores do domínio de paper, o evento é consistente com ele e pode disparar a transição respectiva. A situação representada é a da seleção dos artigos após a revisão, para sua futura inclusão no programa do congresso.

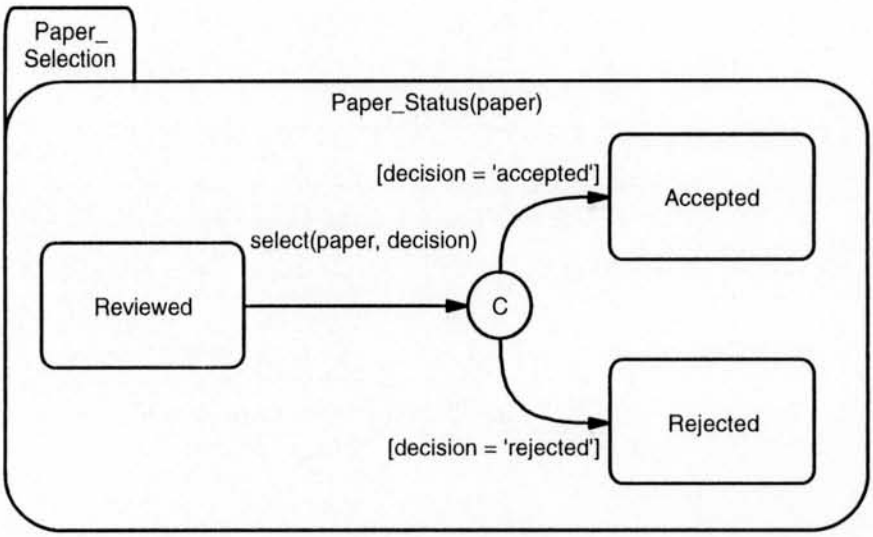


FIGURA 3.9 - Exemplo específico que mostra a relação entre o parâmetro do evento e a variável de referência do superestado.

No exemplo desta figura, ao instanciar o evento select(paper, decision) do diagrama HLS, pode-se observar que ele só aparece naquela transição pertencente ao subestado cuja variável de referência tem o mesmo valor que a do parâmetro. Esta situação é representada graficamente na figura 3.10, na qual a variável paper do evento assume o valor p1 e, portanto, dispara a transição apenas naquele subestado que representa o paper p1.

Note-se que o superestado Paper_Selection que aparece na figura 3.9 pode parecer redundante no sentido dos *statecharts* tradicionais, porém deve existir nos HLS onde a situação mostrada é de um conjunto de estados concorrentes Paper_Status cujo superestado é justamente Paper_Selection como é representado na figura 3.10.

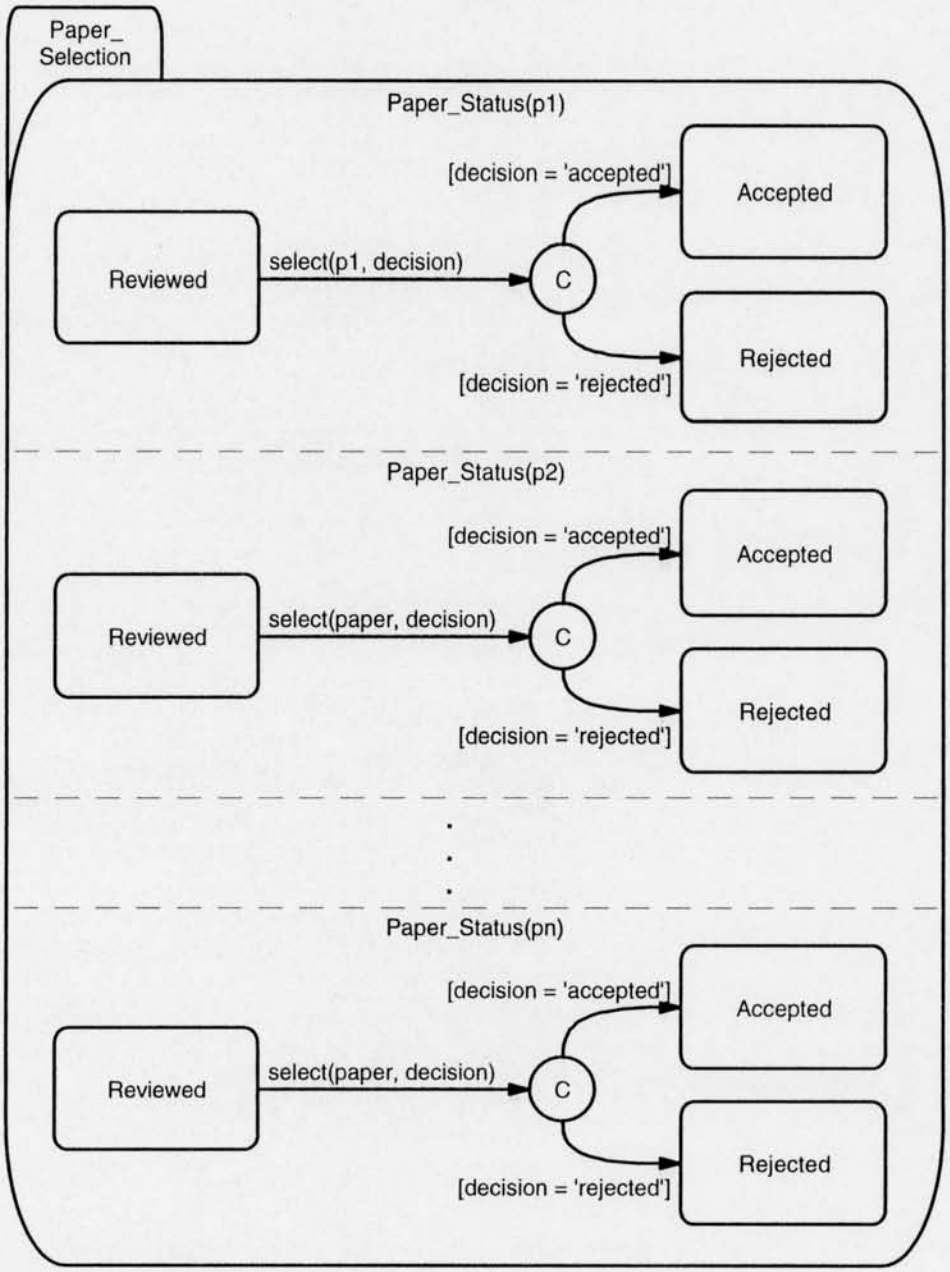


FIGURA 3.10 - Exemplo específico de um parâmetro instanciado do evento e seu efeito sobre o conjunto de estados concorrentes.

Porém, também é possível que um evento ocorra concorrentemente em várias transições, isto é, que cada subestado receba um evento com um valor específico do parâmetro, de tal forma que todo o conjunto de valores seja tratado concorrentemente. Ao ocorrer um conjunto de eventos concorrentes, com valorações diferentes do seu parâmetro, dispara-se um conjunto de transições concorrentes. Como foi indicado na se-

ção anterior, a notação adotada para esta forma de concorrência é a de incluir todos os elementos das transições concorrentes entre chaves { }.

Esta é a situação da figura 3.11, na qual o evento com parâmetro dispara concorrentemente várias transições. Neste caso, existe simultaneidade e o evento select_chairman(person) existe em todos os estados concorrentes (por isto está entre chaves { }); na verdade não é um só evento e sim um conjunto deles[22]. O evento em questão está "repetido" em cada subestado concorrente. Assim, no exemplo da figura, é representada a seleção de um conjunto de moderadores dentre as pessoas que aceitaram o convite.

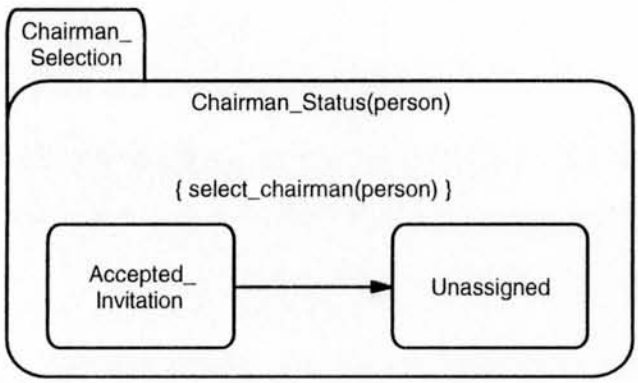


FIGURA 3.11 - Exemplo específico que mostra um conjunto de eventos e transições ocorrendo concorrentemente sobre um superestado.

A figura 3.12 mostra, graficamente, um exemplo de instanciação do parâmetro do conjunto de eventos e transições concorrentes representados na figura 3.11. Supõe-se que o evento ocorre repetidamente, cada repetição com um valor diferente do seu parâmetro dentro do conjunto p1, p2, ..., pn. Assim, o efeito conseguido é o disparo de um conjunto de transições concorrentes, cada uma dentro de um subestado correspondente ao valor do parâmetro do evento da transição.

---

[22] Em um nível maior de abstração, trata-se de um estímulo que o sistema recebe do exterior do mesmo e que é decomposto neste conjunto de eventos. Esta visão mais global e sua relação com os HLS é tratada no próximo capítulo.

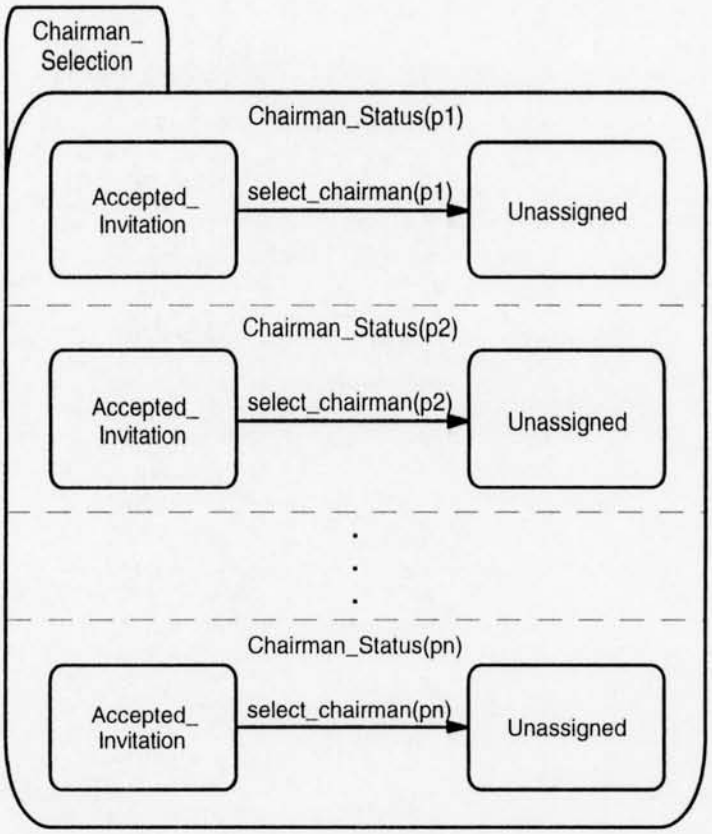


FIGURA 3.12 - Exemplo específico de parâmetros instanciados de um conjunto de eventos e transições concorrentes e seu efeito sobre o conjunto de estados concorrentes.

Em ambos os exemplos, das figuras 3.9 e 3.11, a variável de referência do superestado em que ocorre o evento é igual a um dos parâmetro do mesmo. Como toda variável de referência é um identificador único, segue-se que o parâmetro em questão também o é.

Quando o parâmetro é um dado composto, esta relação é menos óbvia. O nome do parâmetro pode não coincidir com o nome da variável de referência. Isto requer uma consulta ao dicionário de dados para verificar se na definição do dado composto existe algum identificador único que corresponda à variável de referência. Esta é a situação mostrada na figura 3.13, na qual são definidos os papéis de avaliadores entre as pessoas que aceitaram o convite para o congresso. No dicionário de dados existe a seguinte definição necessária para garantir a consistência:

referee_data = @person + maximum

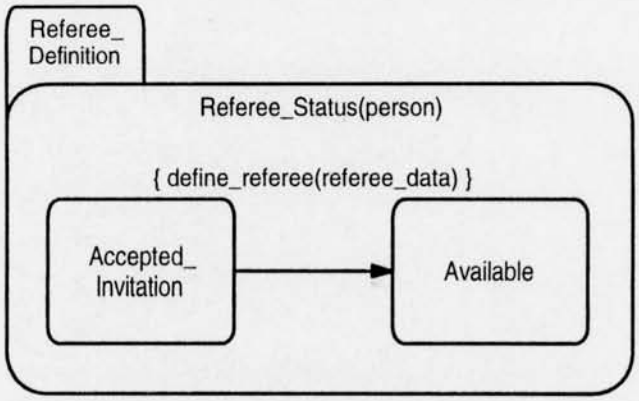


FIGURA 3.13 - Exemplo específico que mostra a relação entre o parâmetro composto do evento e a variável de referência do superestado.

3.3.3.2 A Função de Mapeamento

A função de mapeamento é uma função f_m que, dado qualquer valor do domínio Dx de uma variável de referência x associada a um estado, retorna um valor pertencente ao mesmo domínio Dx. De outra forma:

$$f_m: x \rightarrow y$$
$$Dx \rightarrow Dx$$

Não há risco de ambigüidade neste mapeamento, porque a seta da própria transição, à qual a função de mapeamento está associada, indica o estado de origem e o estado destino. Desta forma, a combinação de uma transição e sua função de mapeamento determina especificamente os estados destinos da transição como um todo. Esta função sempre é necessária quando o estado destino da transição, à qual pertence a função, não se encontra explícito no diagrama HLS, ou, mais precisamente, quando o estado não está representado para o valor da variável de referência do estado destino.

A especificação da função pode ser feita por extensão, isto é, especificando genericamente a transformação do valor de entrada no valor de saída, ou pode ser feita por intensão, indicando explicitamente o valor específico resultante.

A função de mapeamento é representada como o segundo elemento associado à transição e é precedida por dois pontos (":"), independentemente da existência ou não do evento. Não existe convenção de notação para especificar as funções de

mapeamento, ficando o formato totalmente livre. Se a função for muito complexa para ser representada no diagrama HLS, pode até ser especificada no dicionário de dados.

Os exemplos de funções de mapeamento da figura 3.3 têm um mesmo estado de origem e destino e, portanto, um único domínio para as variáveis de referência. Estas funções estão especificadas de acordo com as duas formas indicadas. Para a transição do evento up button, basta calcular o módulo 10 da expressão i+1 para obter genericamente o seguinte estado do dígito, conforme o modelo do *statechart* equivalente. De maneira análoga, a função de mapeamento é genérica para a transição do evento down button, assumindo a forma i+9 mod 10. No caso da transição set button, a função é dada pelo valor específico 0 da variável de referência associada ao estado destino.

3.3.3.3 A Condição

A condição nada mais é do que um *guard* da transição, isto é, uma expressão lógica associada cujo valor_verdade determina ou não o disparo da transição uma vez que o evento tenha ocorrido (se ele estiver presente na transição). Se o seu valor_verdade for verdadeiro, a transição é disparada, se for falso, a transição não é disparada.

A condição pode ser omitida em uma transição sempre que esta possua um evento associado. Nesta situação, o disparo da transição depende exclusivamente da ocorrência do evento.

A expressão da condição é anotada como o terceiro elemento associado à transição e é sempre colocada entre colchetes [ ], independentemente da existência ou não do evento e/ou da função de mapeamento. Podem ser usados todos os operadores relacionais necessários, tais como $<$, $>$, $\leq$, $\geq$, $=$ e $\neq$. Podem ser usadas todas as variáveis (de qualquer tipo), bem como todos os dados componentes disponíveis no diagrama HLS e definidos no dicionário de dados. Valores literais não numéricos devem ser anotados entre ' ' para não serem confundidos com as variáveis. Para aumentar a legibilidade da expressão da condição, é recomendado anotar os operadores lógicos em maiúscula, tais como NOT, AND, OR e XOR.

Como exemplo do uso de condições vide a figura 3.9. A condição aparece em função dos valores de um dos parâmetros.

Existem dois casos particulares de condições que merecem destaque:

- **in Estado**: Esta condição, que é a mesma dos *statecharts*, permite verificar se o Estado encontra-se ativo no momento da ocorrência do evento da transição onde a condição foi definida. Se o Estado estiver aninhado dentro de um ou mais superestados, os nomes destes devem ser antepostos, separados por um ponto, para evitar ambigüidades, assumindo o formato in Superestado.Estado.Subestado.
- **into Estado**: Esta condição, que é uma extensão dos HLS, permite verificar se o Estado é destino de uma transição que é disparada em forma sincronizada com a transição onde se encontra a condição. Esta sincronização é possível se o evento de todas as transições envolvidas é o mesmo[23]. A mesma convenção para evitar ambigüidade é válida para os estados aninhados.

Geralmente estes tipos de condições de verificação de estados ocorrem quando o diagrama HLS apresenta componentes concorrentes com transições que devem ocorrer sincronizadamente para manter uma certa consistência no sistema. Por exemplo, a figura 3.14 mostra um diagrama HLS no qual o evento assign(chairman, session) dispara duas transições. A situação modelada é a seleção de moderador para cada sessão programada no congresso. Em uma primeira aproximação, poderia parecer não necessária a inclusão das condições, porque as transições já possuem o mesmo evento e ambas só seriam disparadas se o evento ocorresse. Entretanto, como ambas as transições encontram-se em subestados com diferentes variáveis de referência associadas, torna-se indispensável verificar, usando as condições da forma in Estado, se os parâmetros do evento (session e chairman) casam com os valores das respectivas variáveis de referência.

Como exemplo de condição da forma into Estado vide a figura 3.23(b). A condição [into Report_Status({report}).Created] serve para verificar que efetivamente os elementos da lista {report} estão por ser criados[24].

---

[23] Esta restrição pode ser relaxada da seguinte forma: a sincronização ocorre se os eventos das transições estão associados por pertencerem à decomposição de um mesmo estímulo externo do sistema. Esta relação estímulo-evento é descrita no próximo capítulo.

[24] Para mais detalhes sobre a criação de estado vide a seção 3.3.3.5.

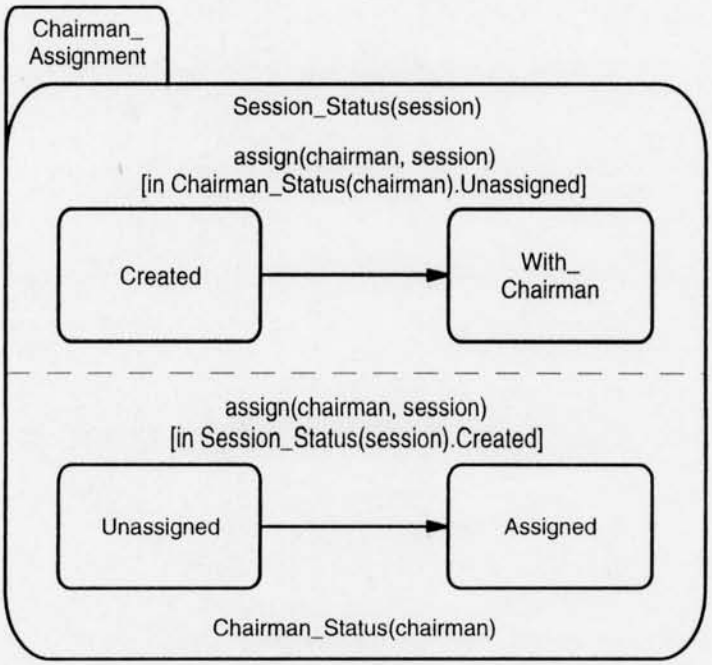


FIGURA 3.14 - Exemplo específico de sincronização consistente de transições usando condições.

Por fim, cabe fazer uma observação em relação às condições da forma in e into Estado e às variáveis de referência. Muitas vezes a variável de referência que aparece em alguma das expressões destes tipos de condições pode não ser exatamente a mesma que está sendo utilizada no superestado onde, efetivamente, o estado se encontra. Neste caso, é suficiente que os domínios das variáveis sejam compatíveis entre si para que esta verificação seja definida. Assim, os valores de variável_1 em uma condição do tipo in ou into Estado(variável_1) podem ser assumidos pela variável de referência do Estado(variável_2). Este é o caso mostrado na figura 3.15, entre as variáveis author e person.

3.3.3.4 A Ação

Uma ação, no contexto restrito de uma transição em um diagrama HLS, é uma operação realizada no momento em que a transição é disparada[25]. Como a modela-

[25] Esta opção de alocar as ações nas transições e não nos estados tipifica o modelo Mealy da máquina de estados finitos [DAV 93], que é subjacente aos HLS e *statecharts*.

gem com os HLS é conceitual, não é relevante o tempo que levam estas ações para executar e podem ser consideradas como instantâneas.

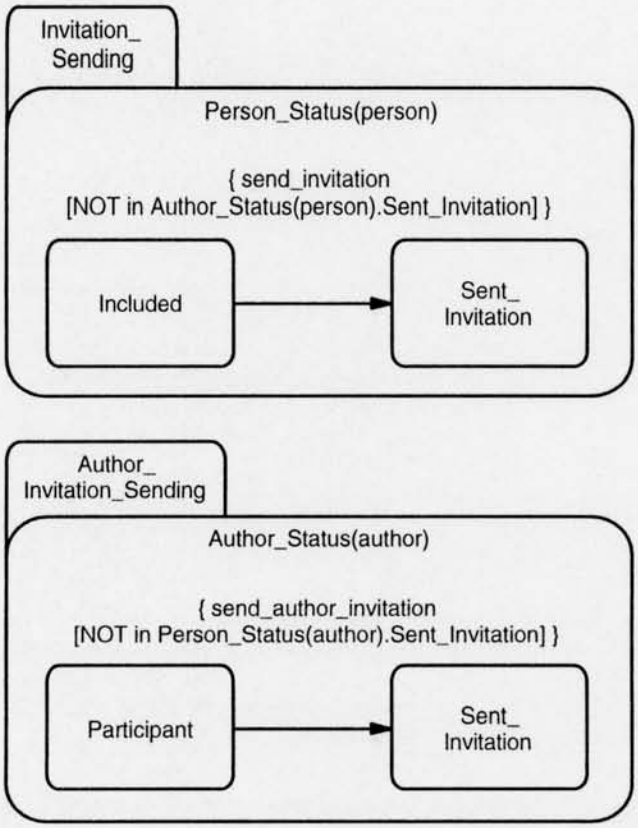


FIGURA 3.15 - Exemplo específico de condições da forma in Estado com variáveis de referência diferentes.

As ações são anotadas como o quarto elemento associado à transição e são precedidas por uma barra ("/"). Se houver mais de uma ação a ser realizada, elas são colocadas seqüencialmente separadas por ponto e vírgula (";"). A ordem em que são anotadas as ações em uma transição não implica qualquer tipo de seqüência de execução[26].

[26] No caso de ser necessário definir uma seqüência, pode se antepor às ações, separado com um ponto, uma numeração seqüencial. Exemplo: 1.ação 1; 2.ação 2; ...; n.ação n.

Da mesma forma que nos eventos, as ações também podem apresentar uma ou mais variáveis como parâmetros. Geralmente, estes parâmetros são resultados de algum tratamento realizado no modelo ou de variáveis definidas para relacionar-se com variáveis de referência de superestados. Os parâmetros são igualmente anotados entre parênteses ( ) e separados por vírgula (",").

As ações podem ser divididas em três categorias:

- **Ações que tornam-se eventos para outras transições**: Nesta categoria, as ações servem como geradoras de eventos que, por sua vez, podem disparar outras transições dentro do modelo. Por esta mesma razão prática, quando a ação torna-se um evento, não existe diferença entre a notação de uma ação e de um evento. Este encadeamento de eventos e ações é muito útil, por exemplo, para modelar consultas. A figura 3.16 mostra uma consulta que encadeia ações e eventos. Trata-se de conhecer os dados dos artigos a serem apresentados em uma sessão dada[27]. O evento inicial get_session's_papers(session) tem associada a ação get_papers({paper}) que torna-se o evento para cada auto-transição concorrente do estado Programmed do artigo. Com isto, é representado o conjunto de artigos da sessão dada. A ação final session's_papers(paper_data) fornece os dados do artigo programado na sessão.
- **Ações de conformidade**: Estas ações apenas indicam que a transição ou tratamento realizado no sistema foi feito à completa satisfação[28]. Para simplificar, deve ser usado o mesmo nome do evento da transição à qual pertence a ação, de maneira que, do ponto de vista externo ao sistema, ambos estejam relacionados. A este nome acrescentar-se-ia o sufixo _ok, para o caso de conformidade simples (vide figura 3.17, que representa o registro das cartas de intenção recebidas em resposta à chamada de trabalhos), ou se usaria a constante 'ok', como parâmetro da ação de conformidade composta, quando houvesse mais de um parâmetro no evento. Estas constantes guardam uma relação posicional com os parâmetros do evento. Assim, por exemplo, o evento assign(chairman, session) pode ter como ação de conformidade composta assign('ok', 'ok'), onde o primeiro 'ok' é a confirmação da transição satisfatória no que a chairman se refere, e o segundo 'ok' é a confirmação em relação ao parâmetro session do evento. Como muitas vezes

---

[27] Nos modelos desenvolvidos na técnica proposta, a informação sobre como o diagrama começa e termina é apresentada em outro diagrama mais abstrato.

[28] Em um nível maior de abstração, estas ações compõem uma resposta que o sistema emite para o exterior do mesmo. Esta visão global será tratada no próximo capítulo.

os parâmetros são tratados em estados concorrentes, a geração da ação pode ser dividida, cada ação contribuindo com um 'ok' para o parâmetro tratado especificamente na porção concorrente do diagrama HLS. Para evitar problemas com a relação posicional, introduz-se o parâmetro null onde o corresponda[29]. A figura 3.18 completa a figura 3.13 com ações divididas.

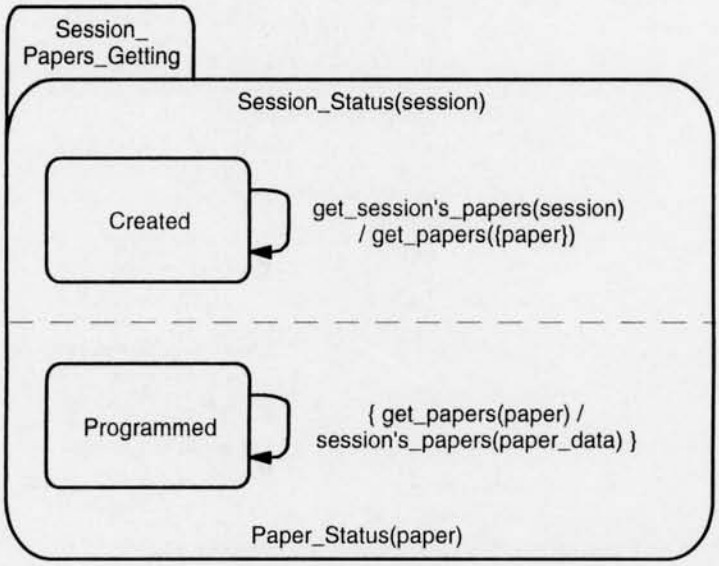


FIGURA 3.16 - Exemplo específico de encadeamento de ações e eventos em uma consulta.

- **Ações como operações específicas**: Nesta categoria, as ações simplesmente realizam operações sobre variáveis componentes tais como cálculos ou modificações de valores em geral. Deste tipo, são também, as ações usadas para inicializar e finalizar estados, as quais serão tratadas na próxima seção. O exemplo da figura 3.19 mostra a ação increment(revisions), que é uma operação específica no contexto da distribuição de artigos entre os avaliadores. Também neste exemplo existem ações de conformidade divididas. A situação modelada é a da distribuição dos artigos para os avaliadores, para os quais foi definido um número máximo de artigos para avaliar.

[29] As ações, ao compor respostas mais abstratas, perdem os parâmetros nulos por sobreposição, assim por exemplo, a resposta obtida seria assign('ok', 'ok') = assign('ok', null) + assign(null, 'ok'). Tudo isto é tratado em detalhe no próximo capítulo.

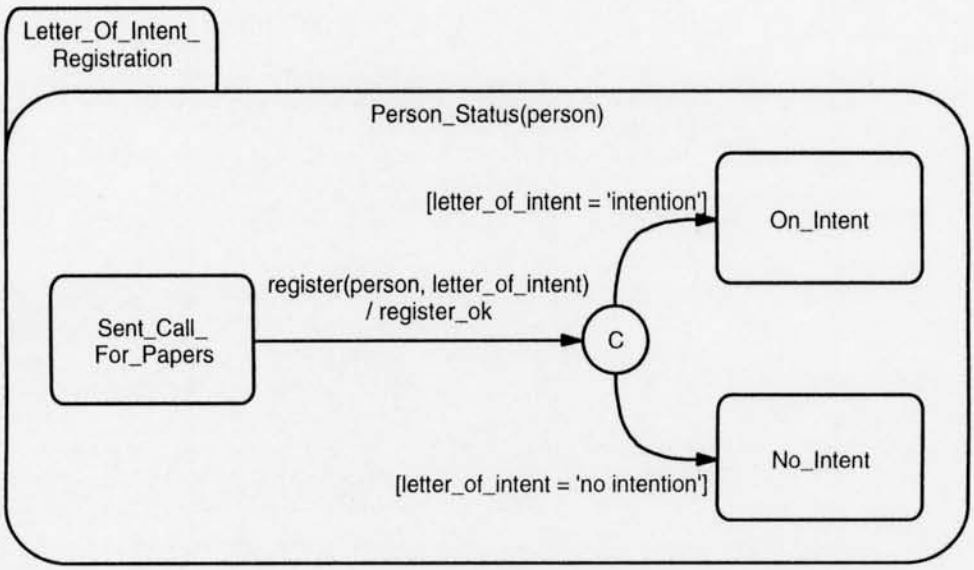


FIGURA 3.17 - Exemplo específico de ação de conformidade simples.

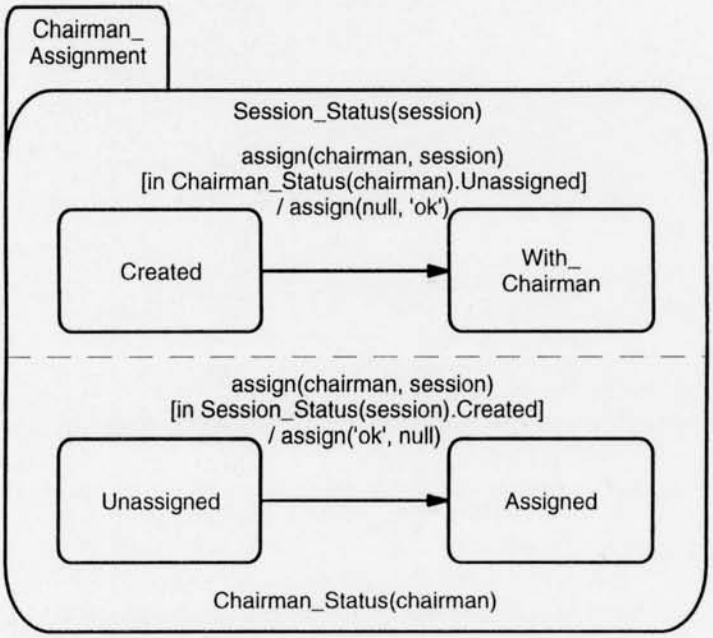


FIGURA 3.18 - Exemplo específico de ações de conformidade divididas.

Nem sempre as ações estão associadas a uma única transição. Existem situações em que a ação pode ser realizada após a ocorrência de um conjunto de eventos,

em diferentes transições. Estas transições podem apresentar também funções de mapeamento, condições e até ações.

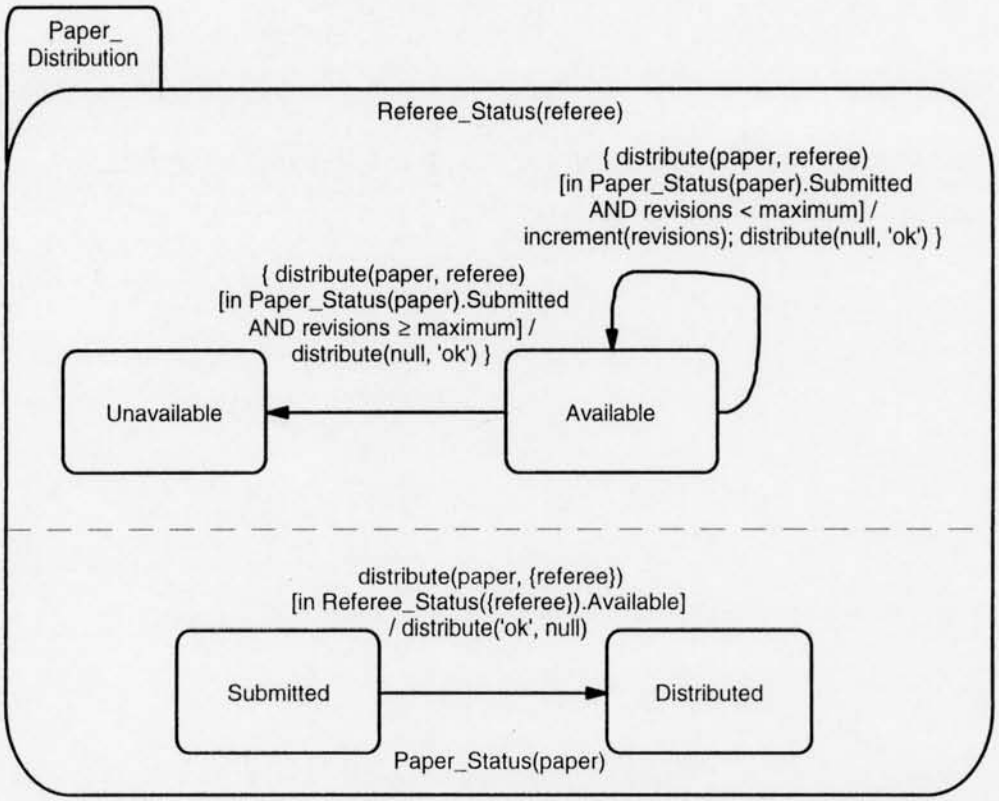


FIGURA 3.19 - Exemplo específico de ações como operações específicas.

Assim, podem ser estabelecidas duas alternativas da relação transição-ação:

- **1 transição ↔ 1 ou mais ações**: É a situação mais comum apresentada em todos os exemplos anteriores desta seção. A convenção de notação já foi apresentada:

  evento : função_de_mapeamento [condição] / ação; ação; ...

- **Conjunto de transições ↔ 1 ou mais ações**: A ação é realizada após a ocorrência de cada um dos eventos nas respectivas transições, incluindo o evento, a função de mapeamento, a condição e a ação dentro deste conjunto. A ação não pertencente ao conjunto é realizada apenas após serem disparadas todas as transições. Esta situação é anotada introduzindo o símbolo asterisco ("*") antes desta

ação. Da mesma forma que no caso anterior, se houver mais de uma ação associada, estas devem ser separadas por ponto e vírgula (";"). A convenção assume a seguinte forma:

{ evento : função_de_mapeamento [condição] / ação; ação; ... } * ação; ação; ...

O evento, a condição, ou ambos devem estar presentes. A função de mapeamento e a primeira ação são optativas.

A figura 3.20 mostra genericamente esta última situação. Cada transição tem seus próprios elementos de eventos, função de mapeamento, condição e ação associados. Após todas as mudanças de estado provocadas pelas respectivas transições, é realizada a ação representada pela * ação maior à direita da figura.

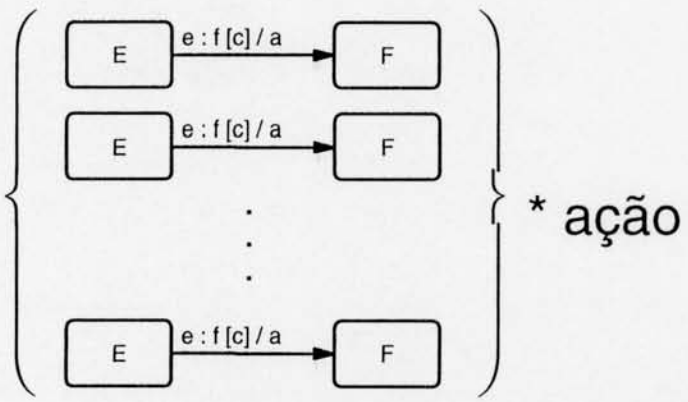
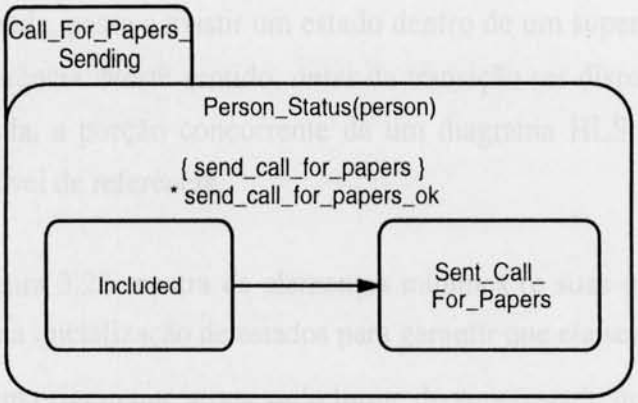


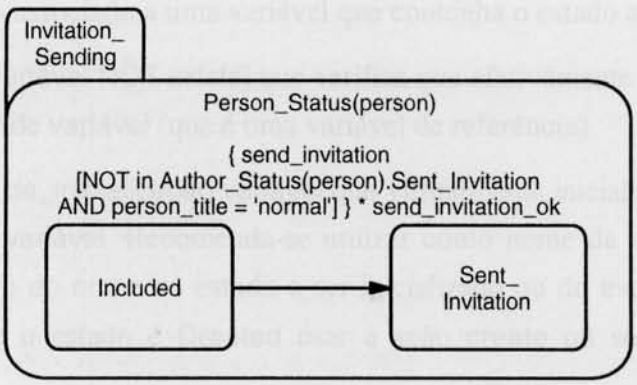
FIGURA 3.20 - Esquema gráfico de uma ação associada a um conjunto de transições.
(Notações: E, F = estados, e = evento, f = função de mapeamento,
c = condição e a = ação).

A figura 3.21 mostra dois exemplos de situações onde a ação está associada a um conjunto de transições. A situação da figura 3.21(a) é mais simples porque existe apenas um evento sem parâmetros em cada transição. A ação de conformidade simples send_call_for_papers_ok é realizada após a ocorrência de todos os eventos onde as pessoas encontram-se no estado Included. A figura 3.21(b) mostra o envio dos convites para as pessoas incluídas. Neste exemplo, as transições apresentam evento e condição. Uma vez que é importante não enviar convites duplicados para as pessoas, é

necessário verificar se a pessoa não é um convidado prioritário ou se a mesma pessoa já não recebeu convite como autor.



(a)



(b)

FIGURA 3.21 - Exemplos específicos de ação de conformidade simples associada a um conjunto de transições com: (a) evento; (b) evento e condição.

3.3.3.5 Os Casos Particulares de Transições

Existem dois casos de transições que merecem especial atenção: transição de inicialização de estados e transição de finalização de estados.

Uma transição de inicialização de estados é uma transição sem estado de origem que possui uma condição que verifica a não existência do estado destino a ser

inicializado (através da variável de referência), e possui, também, uma ação que inicializa este estado no momento do disparo da transição.

A transição de inicialização de estado provoca uma situação tal que, depois que ela é disparada, passa a existir um estado dentro de um superestado associado a uma variável de referência. Neste sentido, *antes* da transição ser disparada, o estado e o superestado, ou ainda, a porção concorrente de um diagrama HLS *não existiam* para aquele valor da variável de referência.

A figura 3.22 mostra os elementos mínimos (e suas representações) que devem existir em uma inicialização de estados para garantir que ela seja consistente:

- Transição sem origem que atravesse o limite do superestado que contém o estado a ser inicializado.
- Superestado associado a uma variável que contenha o estado a ser inicializado.
- Condição [variável NOT exists] que verifica que efetivamente não exista o estado para o valor de variável (que é uma variável de referência).
- Ação ação_de_inicialização variável que virtualmente inicializa o Estado_Inicializado para variável. Recomenda-se utilizar como nome da ação um termo que seja derivado do nome do estado a ser inicializado ou do evento associado. Por exemplo, se o estado é Created usar a ação create ou se for Included usar include.
- Mesma variável de referência usada na condição, na ação e no superestado para garantir consistência na inicialização.

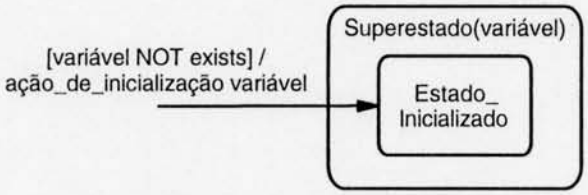


FIGURA 3.22 - Representação dos elementos mínimos para a inicialização consistente de estados.

É claro que a transição de inicialização de estado pode apresentar evento, outras expressões de condição e mais ações. Entretanto, os elementos indicados acima são mínimos porque sempre devem estar presentes neste tipo de transição.

Alguns exemplos específicos de inicialização de estados são mostrados na figura 3.23. A figura 3.23(a) representa a situação da preparação de uma lista de pessoas a serem convidadas ao congresso. O estado Included é inicializado para cada pessoa na lista. A figura 3.23(b) modela a situação da coleta dos relatórios dos artigos emitidos pelos avaliadores. É criado um ou mais report's para cada paper do evento.

Por simetria de propriedades é possível definir o efeito contrário à inicialização de estados: a finalização de estados.

Uma finalização de estados é uma transição sem estado de destino, que possui uma condição que verifica a existência do estado de origem a ser finalizado (através da variável de referência), e uma ação que permite finalizar este estado no momento do disparo da transição.

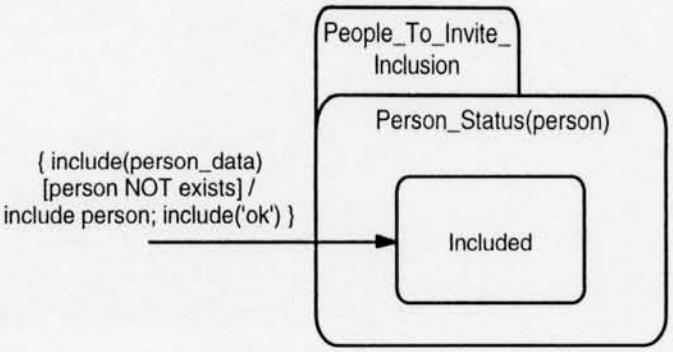
A transição de finalização de estado provoca uma situação tal que, depois de ser disparada, o estado dentro de um superestado associado a uma variável de referência deixa de existir. Neste sentido, *antes* da transição ser disparada, o estado (ou porção concorrente de um diagrama HLS) e o superestado (ou todo o modelo) *existiam* normalmente para aquele valor da variável de referência.

A figura 3.24 mostra os elementos mínimos (e suas representações) que devem existir em uma finalização de estado. Obviamente, o evento, outras expressões de condição e mais ações também podem aparecer. Os elementos mínimos para a consistência da finalização são:

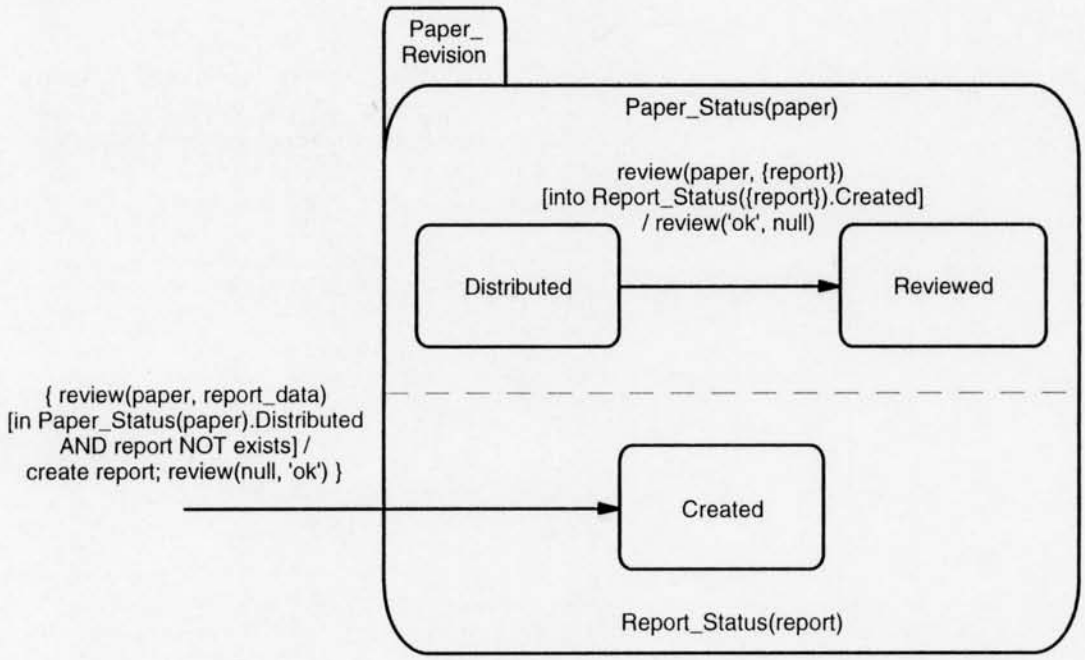
- Transição sem destino que atravesse o limite do superestado que contém o estado a ser finalizado.
- Superestado associado a uma variável que contenha o estado a ser finalizado.
- Condição [variável exists] que verifica que efetivamente exista o estado para o valor de variável (que é uma variável de referência).
- Ação ação_de_finalização variável que virtualmente finaliza o Estado_Finalizado para variável. Recomenda-se usar como nome da ação um termo

vinculado ao nome do estado ou ao nome do evento; entretanto, como nem sempre isto é possível, pode-se optar pela ação padronizada eliminate variável.

- Mesma variável de referência usada na condição, na ação e no superestado para garantir consistência na finalização.



(a)



(b)

FIGURA 3.23 - Exemplos específicos de inicialização de estados.

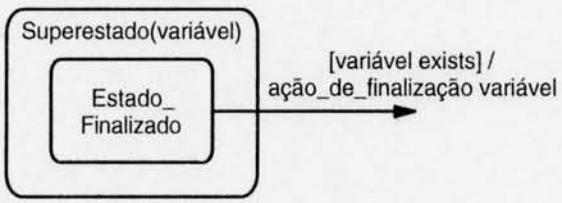


FIGURA 3.24 - Representação dos elementos mínimos para a finalização consistente de estados.

Um exemplo específico simples de finalização de estado é mostrado na figura 3.25. A situação representada é a da eliminação de uma pessoa da lista de convidados ao congresso.

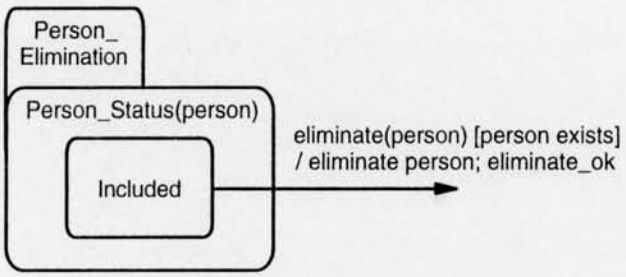


FIGURA 3.25 - Exemplo específico de finalização de estado.

3.4 RESUMO

O presente capítulo apresentou uma descrição detalhada da sintaxe e da semântica dos *High-Level Statecharts* (HLS): uma proposta de extensão dos *statecharts* de Harel [HAR 87]. Esta extensão foi desenvolvida para permitir a modelagem de sistemas de informação tradicionais, dadas as limitações naturais dos *statecharts* para modelar conjuntos.

Foram descritas as representações dos *statecharts* que também são suportadas nos HLS, bem como, também, as exclusões de alguns elementos que não são considerados.

Os HLS consistem em diagramas que mostram redes de estados e transições hierarquizadas à maneira dos *statecharts*. Os estados são considerados como passivos e possíveis de ter associadas variáveis de referência. Estas variáveis permitem representações compactas de conjuntos de estados concorrentes em uma forma genérica; adicionalmente, permitem definir superestados com estrutura de conjuntos de estados exclusivos.

As transições, nos HLS, têm associados quatro elementos: evento, função de mapeamento, condição e ação. O evento, que representa um fato relevante que dispara uma ou mais transições, pode apresentar uma ou mais variáveis associadas como parâmetros. A função de mapeamento transforma valores de variáveis de referência que indicam o estado destino da transição. A condição permite realizar verificações que determinam, em última instância, o disparo da transição em função do seu valor_verdade. A ação, que também pode apresentar variáveis associadas como parâmetros, pode representar um outro evento que é gerado durante a transição, ou um evento específico de conformidade da transição realizada satisfatoriamente, ou ainda ser uma operação específica dentro do diagrama. Adicionalmente, uma ação pode estar associada a um conjunto de transições concorrentes.

Finalmente, são descritos os casos especiais de transições que permitem inicializar e finalizar estados nos diagramas HLS.

4 MODELAGEM INICIAL DE SISTEMAS

O processo de modelagem inicial de sistemas, que inclui o uso dos HLS descritos no capítulo anterior, trata de como obter uma representação de um sistema na perspectiva dinâmica na forma final de diagramas HLS. O objetivo deste capítulo é justamente descrever este passo dentro do processo de modelagem.

Como exemplo básico de aplicação é usado o problema padrão de preparação de congressos da IFIP descrito em [OLL 82]. Os exemplos mantêm o texto em língua inglesa. Tanto a definição do problema da IFIP como todos os modelos desenvolvidos neste capítulo encontram-se nos anexos "Definição do Problema da IFIP" e "Modelo Completo do Problema da IFIP", respectivamente.

O capítulo foi organizado como segue: a seção 4.1 apresenta o porquê de uma visão global na técnica de modelagem; a seção 4.2 descreve como construir o modelo da visão global incluindo o contexto do sistema e definindo o mesmo como uma lista de tuplas com estímulos e respostas; a seção 4.3 descreve a construção do modelo de processos e a relação deste com o modelo da visão global; finalmente, a seção 4.4 apresenta a estrutura geral do modelo completo usando HLS.

4.1 A Necessidade de uma Visão Global

Construir um modelo específico de um universo de discurso ou domínio de problema sem contar com uma estratégia e uma notação que facilite o refinamento sucessivo pode ser uma tarefa muito difícil, dado que estes domínios geralmente apresentam uma grande complexidade. Principalmente, se é considerado o fato de que o processo de modelagem é também um processo de aprendizado, em que é buscado um aumento da compreensão do domínio por parte do modelador.

É por isto que aos diagramas HLS, descritos no capítulo anterior, é acrescentado um segundo modelo mais abstrato, que visa a uma descrição do sistema em um contexto de estímulos e respostas.

Assim, para modelar sistemas com esta técnica de modelagem é preciso construir o seguinte:

1. Modelo da visão global do sistema.
2. Modelo de processos.
3. Dicionário de dados.

4.2 Construção do Modelo da Visão Global do Sistema

O modelo da visão global do sistema é constituído por dois diagramas:

1. Diagrama de contexto do sistema.
2. Diagrama da visão global.

Para poder modelar adequadamente o sistema de informação é necessário, primeiramente, contextualizar o sistema.

4.2.1 O Conceito de Sistema de Informação

O conceito de sistema de informação é chave para a compreensão da abordagem inicial na modelagem dinâmica dos domínios de problema. Um sistema de informação é um subsistema dentro de um sistema mais amplo, qual seja, o supra-sistema, que "reage" fornecendo respostas adequadas a certos estímulos relevantes que o ambiente gera. É através destes estímulos e respostas que a informação flui, entrando e saindo do sistema. A figura 4.1 mostra o contexto inicial para um sistema visto nesta perspectiva.

Na figura 4.1, os agentes externos ao sistema sob modelagem (A1, A2, A3, A4 e A5), tais como outros sistemas de informação, tipos de usuários, unidades da organização, etc. geram estímulos que o sistema detecta e aos quais reage com respostas apropriadas a cada caso. Assim, por exemplo, na figura, o agente externo A3 gera o estímulo de A3 e, por sua vez, o SISTEMA DE INFORMAÇÃO SOB MODELAGEM reage emitindo a resposta a A3. Todos estes sistemas fazem parte do **Supra-sistema**.

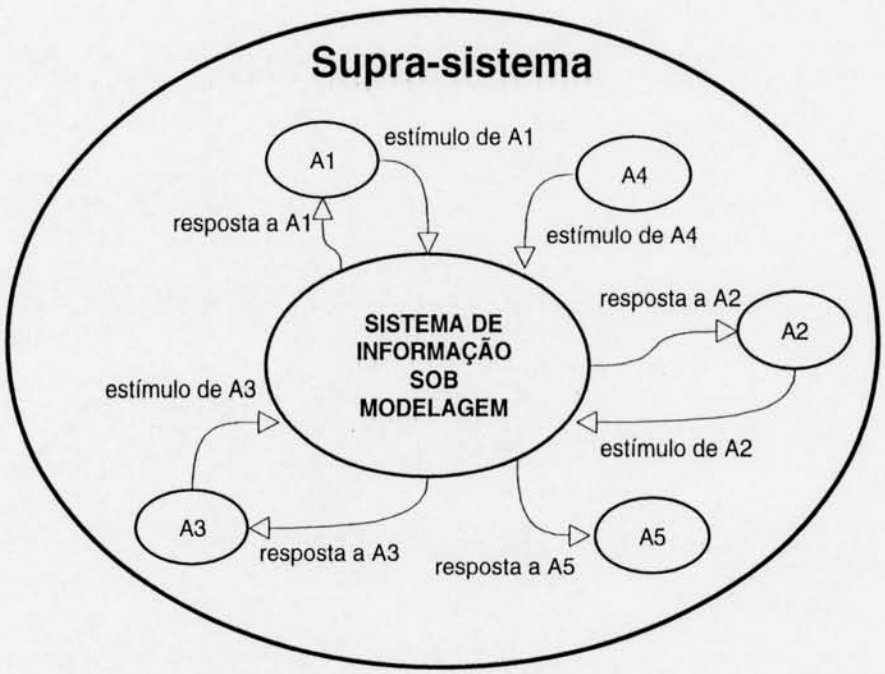


FIGURA 4.1 - Contexto de um sistema de informação sob modelagem.

4.2.2 O Contexto do Sistema

Como primeiro passo no desenvolvimento do modelo da visão global do sistema, deve ser usado um enfoque sistêmico [CHU 71], [WEI 75] e [BER 80] para a identificação do supra-sistema do qual o próprio sistema e os outros sistemas relacionados (os agentes externos) fazem parte. Isto necessariamente passa pela definição das fronteiras do sistema sob modelagem. A fronteira ou limite do sistema determina os estímulos (entradas) e respostas (saídas) do mesmo. Estes estímulos e respostas relacionam o sistema com outros sistemas ou agentes externos e fornecem o contexto para o funcionamento do sistema sob modelagem. A identificação do supra-sistema e dos agentes externos é necessária por dois motivos principais:

- O supra-sistema fornece o contexto ou "pano de fundo" no qual o sistema em questão desenvolve suas atividades. Dentro deste contexto, o sistema sob modelagem deve desempenhar uma função ou papel específico. Esta determinação ajuda no conhecimento da razão de ser do sistema e na compreensão dos requisitos que se espera que satisfaça.

- O agentes externos ou sistemas relacionados permitem determinar os limites do sistema sob modelagem, isto é, quais elementos vão fazer parte do mesmo e quais vão ficar fora dele. Esta determinação fornece os estímulos e as respostas que o sistema recebe e gera. Conhecer cada agente externo ajuda na definição dos estímulos e respostas entre o sistema e este agente.

Um aprofundamento na conceituação necessária para definir os elementos aqui descritos foge ao propósito deste trabalho.

Para o caso do problema da IFIP, o supra-sistema é a IFIP CONFERENCE e os sistemas relacionados ou agentes externos ao IFIP CONFERENCE SYSTEM são o Program Committee e o Organising Committee. O papel que cabe ao IFIP CONFERENCE SYSTEM é o de fornecer suporte para os comitês envolvidos na preparação do congresso de trabalho da IFIP [OLL 82]. Graficamente, isto fica representado na figura 4.2.

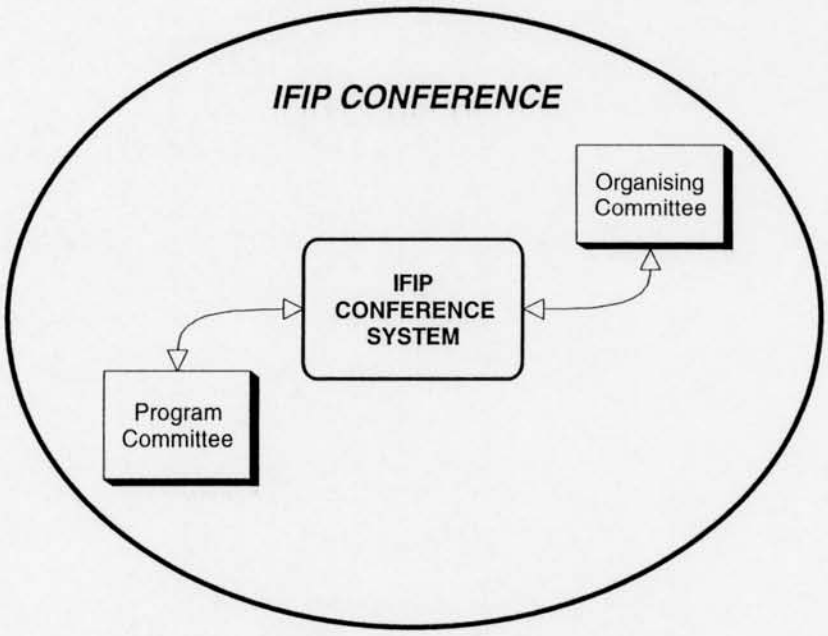


FIGURA 4.2 - O contexto do sistema para o problema da IFIP.

Nesta figura, o supra-sistema *IFIP CONFERENCE* é a elipse maior que contém o sistema e os agentes externos. O sistema IFIP CONFERENCE SYSTEM é representado pela caixa de cantos arredondados e os agentes externos Program Committee e Organising Committee pelas caixas sombreadas. As setas duplas apenas indicam a existência de relações do tipo estímulo-resposta entre o sistema e os agentes externos.

4.2.3 O Sistema Como uma Lista de Tuplas

Uma vez estabelecidos os agentes externos e os limites do sistema sob modelagem, deve-se definir uma lista dos tipos de estímulos relevantes que cada um destes agentes externos gera para o sistema em questão, conforme os requisitos estabelecidos para o mesmo. O agente externo que gera um estímulo é denominado, do ponto de vista do sistema, *emissor*.

Cada tipo de estímulo[30] desta lista é uma mensagem ou comunicação externa que o emissor gera e o sistema detecta. Estas comunicações referem-se a fatos relevantes que devem ser conhecidos pelo sistema. Os estímulos também podem ser interpretados como comandos específicos aos quais o sistema deve responder.

O sistema possui um *processo* para cada tipo de estímulo que recebe de um emissor. Um processo é um tratamento específico que o sistema realiza em função do estímulo recebido. O estímulo é o elemento que desencadeia a execução do processo. Como resultado da execução do processo é gerada uma única resposta[31] do sistema para um agente externo. O agente externo que recebe a resposta é denominado, do ponto de vista do sistema, *receptor*.

No caso de que um estímulo, ao ser tratado pelo processo, requerer mais de uma resposta que podem ou não serem emitidas para diferentes receptores, sugere-se dividir duplicando o estímulo, isto é, definir o mesmo estímulo agindo sobre diferentes processos, respostas e, eventualmente, receptores. Esta forma de decomposição favorece a noção de concorrência no sistema.

---

[30] O termo estímulo é usado também para referir-se ao tipo de estímulo, salvo indicação em contrário.
[31] Na verdade é um tipo de resposta, porém ambos os termos são tratados indistintamente, salvo indicação em contrário.

A resposta é conceitualmente equivalente ao estímulo. Assim, uma resposta é uma comunicação externa que o sistema emite para um receptor. O conteúdo desta comunicação pode ser a mensagem de que o processo foi executado à inteira satisfação ou pode ser algum fato relevante para o receptor. A equivalência entre estímulo e resposta pode ser verificada através do seguinte exercício de modelagem: o objeto de modelagem é "deslocado" para algum de seus agentes externos, de tal forma que o novo sistema sob modelagem é um agente externo e o novo agente externo é o antigo sistema sob modelagem; como o ponto de vista do sistema é mudado, os estímulos tornam-se agora "respostas" e as respostas tornam-se "estímulos", sem vinculação temporal direta entre eles, porque, na verdade, as respostas não dão origem aos estímulos. Esta modificação de ponto de vista não deveria afetar os conteúdos dos estímulos e das respostas já definidos e sim apenas a relação temporal existente entre eles.

A figura 4.3 mostra um exemplo genérico para este exercício:

- o agente externo AE2 de (a) torna-se o SISTEMA SOB MODELAGEM em (b),
- o SISTEMA SOB MODELAGEM de (a) torna-se o agente externo AE4 em (b),
- os estímulos e3 e e2 de (a) tornam-se "respostas" em (b) e
- as respostas r2 e r4 de (a) tornam-se "estímulos" em (b).

Ambas as situações mostradas na figura 4.3(a) e 4.3(b)[32] são válidas, de acordo com as definições de sistema, estímulo, emissor, resposta e receptor.

Cada processo que constitui o sistema pode ser entendido, a partir de um ponto de vista funcional, como uma transformação de um estímulo em uma resposta. Isto fica mais evidente ao se considerar que tanto o estímulo como a resposta podem apresentar dados associados como parâmetros[33].

No modelo da visão global, os processos são considerados como caixas pretas, isto é, o detalhe da forma pela qual o processo converte o estímulo em resposta é interno ao processo e por completo desconhecido fora dele. Os emissores e receptores não sabem nada a respeito do como operam os processos do sistema, apenas conhecem os estímulos e respostas.

---

[32] É claro que se esta situação for realmente modelada, não interessam as comunicações entre agentes externos que aparecem na figura, nem os agentes externos que não se relacionam com o sistema.
[33] Isto é tratado em detalhe na seção 4.2.5.

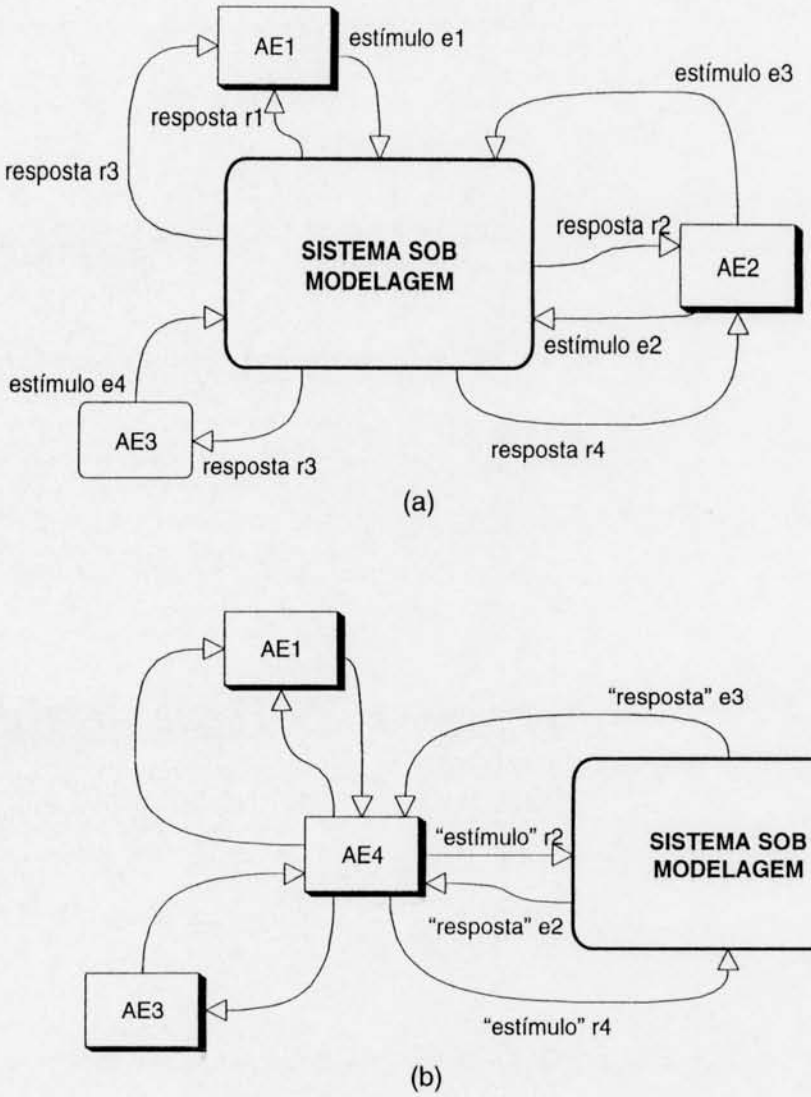


FIGURA 4.3 - Exemplo genérico de "deslocamento" de modelagem: (a) O SISTEMA SOB MODELAGEM e sua relação com os agentes externos AE1, AE2 e AE3; (b) O SISTEMA SOB MODELAGEM deslocado para o agente externo AE2.

Cada estímulo gerado por um emissor tem associado um, e somente um, processo do sistema, e cada processo do sistema gera uma resposta para um receptor. É possível definir o sistema sob modelagem em termos de uma lista de tuplas da forma (emissor, estímulo, processo, resposta, receptor).

Estas tuplas são representadas graficamente (vide figura 4.4) como segue:

1. **Emissor**: Caixa sombreada com o nome do agente externo, em formato livre, dentro da caixa.

2. **Estímulo**: Seta de linha descontínua que se origina no emissor e cruza o limite para dentro do processo. O nome do estímulo, que é anotado sobre a linha, é uma frase constituída por palavras em minúscula separadas pelo símbolo de sublinhado ("_"). A frase deve começar com um verbo seguido ou não por um objeto. Exemplos de nomes de estímulos: review, send_call_for_papers e generate_finallist_of_attendees. Esta convenção de nomes tem apenas o propósito de uniformizar os estímulos e podem ser utilizadas outras convenções que venham a manter a mesma homogeneidade desejada. Por exemplo, pode-se optar por nomes na forma nominal tais como paper_revision, call_for_papers_sending e finallist_of_attendees_generation[34].

3. **Processo**: Caixa de cantos arredondados com o nome do processo dentro dela. O nome do processo, que é derivado do nome do estímulo, é uma frase nominal constituída por palavras que começam com maiúscula separadas pelo símbolo de sublinhado ("_"). Por exemplo, os nomes dos processos para os estímulos acima seriam Paper_Revision, Call_For_Papers_Sending e Finallist_Of_Attendees_Generation. Esta convenção de nomes é proposta apenas para uniformizar os nomes dos processos. Outras convenções tais como nomes na forma verbal do infinitivo também podem ser utilizadas sempre que seja mantida uma certa homogeneidade. Exemplo de nomes de processos seriam: Review_Paper, Send_Call_For_Papers e Generate_Finallist_Of_Attendees[35].

4. **Resposta**: Seta de linha descontínua que se origina dentro do processo cruzando seu limite até o receptor. O nome do estímulo, que também é anotado sobre a linha, é uma frase constituída por palavras em minúscula separadas pelo símbolo de sublinhado ("_"). A frase é derivada diretamente do nome do estímulo, mantendo-se, assim, uma correlação entre o estímulo, o processo e a resposta. No caso mais geral, pode ser o mesmo nome do estímulo, como por exemplo review; ou pode ser acrescentado o sufixo _ok se a resposta for de conformidade simples

---

[34] Neste caso, os nomes dos eventos (vistos no capítulo anterior) também deveriam assumir esta convenção (vide seção 3.3.3.1) e os nomes dos processos (vistos mais adiente) devem ser mudados para a forma verbal no infinitivo.

[35] Neste caso, os nomes dos estímulos (vistos anteriormente) e dos eventos (vistos no capítulo anterior) devem ser mudados para a forma nominal (vide seção 3.3.3.1).

à execução do processo, como por exemplo send_call_for_papers_ok; ou ainda pode ser o objeto do nome do estímulo se a resposta é acompanhada por dados de saída do sistema[36], como por exemplo finallist_of_attendees.

5. **Receptor**: Tem a mesma representação do emissor, isto é, uma caixa sombreada com o nome do agente externo (em formato livre) dentro dela.

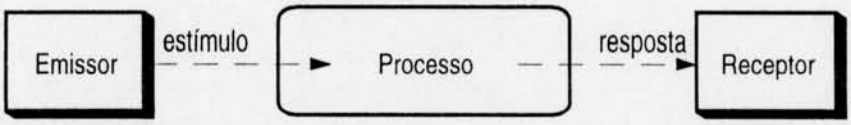


FIGURA 4.4 - Representação de um processo do sistema e sua relação com um estímulo do emissor e uma resposta para o receptor.

Como existe uma lista de tuplas para o sistema, cada tupla deve ser representada no modelo. Isto leva a uma consideração importante: *no modelo da visão global todos os processos são entendidos como concorrentes*. Isto quer dizer que, no nível de abstração em que se está trabalhando no modelo da visão global, e considerando que os processos são caixas pretas, é perfeitamente válido ter todos os processos executando em paralelo. Muitos destes processos podem ser seqüencializados em função dos dados específicos dos estímulos[37], porém, como neste nível de abstração isto ainda não é conhecido, não há meios de, nem é conveniente, estabelecer tais seqüências.

A representação da lista de tuplas concorrentes assume a forma mostrada na figura 4.5. Cada processo é separado dos outros processos por uma linha descontínua para mostrar a concorrência. O sistema é uma lista de tuplas, e seu nome é anotado na parte superior. Esta notação de concorrência é inspirada nos *statecharts* [HAR 87].

Esta figura omite os parâmetros que os estímulos e respostas podem apresentar (vide figura 4.7 para o diagrama completo). Os emissores são posicionados à esquerda do sistema e podem ser representados uma única vez. Os receptores são posicionados à direita do sistema e também podem ser representados uma única vez. É re-

---

[36] Neste caso a resposta apresenta parâmetros. Vide a seção 4.2.5.
[37] Neste caso os estímulos apresentam parâmetros. Vide a seção 4.2.5.

comendável não fundir emissores e receptores em um único agente externo para evitar problemas de cruzamentos de linhas.

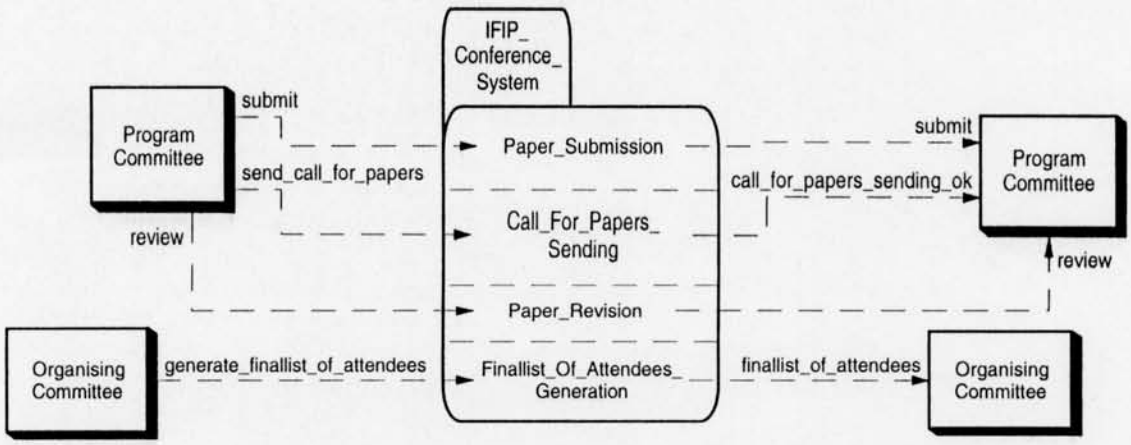


FIGURA 4.5 - Exemplo específico de diagrama da visão global incompleto para o problema da IFIP.

No diagrama da figura 4.5 é possível visualizar o porque das setas dos estímulos e das respostas cruzarem o limite do processo: para destacar a associação que deve existir entre o estímulo e a resposta com seu processo. Desta forma, se estabelece uma conexão gráfica do estímulo e da resposta com o "interior" do processo.

4.2.4 A Noção de Processo

Um processo do sistema foi definido como um tratamento específico encarregado de transformar o estímulo recebido de um emissor em uma resposta para um receptor. O detalhe deste tratamento não é conhecido no momento e, portanto, o processo é concebido como uma caixa preta.

Esta noção de processo é adaptada da noção de *feature* de [DAV 93]. Os sistemas de informação tradicionais modelados na perspectiva dinâmica podem ser entendidos como "ricos em estímulos" (*stimulus rich*), o que significa que os usuários "vêem" o sistema em termos de um conjunto de estímulos e "pensam" o sistema organizado como um conjunto de unidades, onde cada unidade descreve todas as respostas requeri-

das do sistema a um simples estímulo. Cada unidade do sistema é, então, um *feature* ou processo do sistema a partir do ponto de vista do usuário do mesmo. Nesta perspectiva, os usuários podem fazer o seguinte tipo de perguntas em relação ao sistema:

Como é registrado um artigo submetido?

Como são programados os artigos de uma sessão de apresentação?

Como é registrada a carta de intenção de uma pessoa?

Esta abordagem dos sistemas pode ser categorizada como orientada por *features* (*feature-oriented*) para diferenciá-la de uma abordagem orientada por estímulos (*stimulus-oriented*), em que as perguntas que os usuários poderiam formular são do tipo:

O que acontece ao ingressar os dados de um conjunto de pessoas?

O que acontece ao indicar o envio de um convite a um autor?

O que acontece ao ingressar um conjunto de dados de avaliadores?

Como é mais difícil responder às perguntas neste último caso, seria necessário formular outras, tais como:

O que se está tentando fazer?

Deseja-se incluir uma lista de pessoas a serem convidadas ou para receber chamadas de trabalho?

Quer-se definir os avaliadores de um artigo?

Em outras palavras, o importante para a abordagem orientada por features é *O que o usuário/agente externo/ambiente quer fazer?* ou *O que o usuário/agente externo/ambiente quer que o sistema faça?* em vez de *O que acontece quando um estímulo específico chega?* da abordagem orientada por estímulos.

4.2.5 Variáveis como Parâmetros

De maneira análoga aos parâmetros dos eventos e ações nos diagramas HLS descritos no capítulo anterior, os estímulos e respostas também podem apresentar variáveis associadas denominados parâmetros.

Cabe destacar que o conceito de estímulo no modelo da visão global não é exatamente o mesmo que o de evento nos diagramas HLS. Um estímulo pode ser de-

composto em um conjunto de eventos. O mesmo acontece com o conceito de resposta e o de ação dos HLS. Neste caso, existe uma relação de composição de ações em uma resposta. Ambas as relações são descritas em detalhe na seção 4.3.1.

4.2.5.1 Os Parâmetros nos Estímulos e Respostas

De maneira semelhante aos eventos dentro dos diagramas HLS, é comum que, junto à ocorrência de um estímulo, fluam dados que devem ser tratados pelo processo. Estes dados são representados por variáveis e entendidos como parâmetros do estímulo[38]. São anotados da mesma maneira que os parâmetros dos eventos nos HLS: entre parênteses ( ) e separados por vírgula (","). Exemplos de estímulos com parâmetros da figura 4.5 são submit(paper_data, {author_data}) e review(paper, {report_data}).

As respostas também podem apresentar uma ou mais variáveis associadas como parâmetros. Geralmente, as variáveis que aparecem como parâmetros nas respostas são resultados de algum tratamento específico realizado no processo. A notação dos parâmetros nas respostas é a mesma utilizada nos estímulos.

Um caso particular de resposta é aquele em que ela está expressando que o processo, do qual é resultado, foi realizado à inteira satisfação. São as respostas de conformidade composta. O caso de conformidade simples não apresenta parâmetros e é indicado com o sufixo _ok no nome da resposta.

A resposta de conformidade composta é representada usando como parâmetro a constante 'ok' onde for preciso. Estas constantes guardam uma relação posicional com os parâmetros do estímulo. Por exemplo, o estímulo submit(paper_data, {author_data}) pode ter como resposta de conformidade composta submit('ok', {'ok'}), onde o primeiro parâmetro 'ok' indica a conformidade parcial em relação a paper_data (primeiro parâmetro do estímulo), e a lista {'ok'} indica a conformidade relativa à lista {author_data} (segundo parâmetro do estímulo).

---

[38] Para abreviar expressões do tipo "variável composta usada como parâmetro" é usada a expressão equivalente "parâmetro composto".

4.2.5.2 A Definição dos Parâmetros

Como se verá, os parâmetros nos estímulos e respostas cumprem um papel importante nos modelos da visão global e de processos: são eles os objetos de modelagem. Por isto é importante considerar a atividade de definição dos parâmetros.

No caso dos estímulos, pode ocorrer que a chegada de dados a serem processados é que determina a existência do próprio estímulo. Em outros casos, os dados são requeridos para continuar a execução de um novo processo a partir da execução de um processo que recebeu um estímulo anterior. Os dados gerados como resultado da execução do processo são representados como parâmetros da resposta.

Não existe uma regra geral em relação à correspondência na quantidade ou no tipo dos parâmetros do estímulo e da resposta. O estímulo pode não apresentar parâmetros e a resposta sim, e vice-versa; ou tanto o estímulo quanto a resposta podem apresentar parâmetros; ou, ainda, nenhum dos dois pode ter associado dados de nenhum tipo.

A identificação dos dados de entrada (nos estímulos) e de saída (nas respostas) implica um processo prévio de *classificação* no universo do discurso ou domínio do problema. Ao considerar os conceitos deste domínio, é necessária uma abstração de propriedades, de forma tal que permita categorizar e agrupar elementos em classes específicas. Estas classes são os dados representados por variáveis e incluídas, como parâmetros, nos estímulos e respostas.

Uma classificação adequada dos dados do domínio do problema é muito importante para o processo de modelagem descrito neste trabalho, principalmente por duas razões:

- Quanto menores os problemas de consistência e completude entre as variáveis dos estímulos e respostas, mais simples será o passo seguinte na técnica de modelagem. Neste passo devem ser integrados os diferentes processos em ciclos de vida referentes aos mesmos dados. Como os processos podem também ser entendidos como visões (*views*) dos usuários do sistema, o problema de integração pode tornar-se complexo. É necessário, portanto, manter um alto grau de consis-

tência nos nomes e variáveis através da modelagem dos diversos processos, de tal forma que a quantidade de sinônimos e homônimos seja minimizada[39].

- Estes dados serão os principais candidatos a classes de objetos no último passo da técnica de modelagem[40]. Uma correta classificação dos dados permitirá um mapeamento simplificado dos modelos da visão global do sistema e de processos para o modelo de objetos resultante da modelagem aqui descrita.

Contudo, os mesmos tipos de estímulos e respostas já fornecem uma idéia aproximada dos dados de entrada e de saída que são necessários. Desta forma, a classificação torna-se transparente e por isto não é definida como uma atividade separada.

Por exemplo, a identificação dos dados representados pelas variáveis paper_data e author_data como parâmetros do estímulo da figura 4.6 (considerando sua definição no dicionário de dados) é derivada diretamente da existência do estímulo submit, porque para que um artigo seja submetido é necessário conhecer seus dados (tais como seu título) e os dados do(s) autor(es) do mesmo (tais como seu(s) nome(s) e endereço(s)).

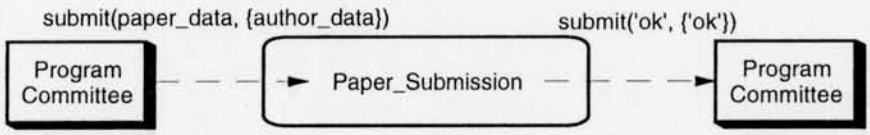


FIGURA 4.6 - Exemplo específico de parâmetros no estímulo de uma tupla.

Inicialmente, pode ser desconsiderado algum dado ou ainda ser considerados dados irrelevantes para o sistema. O próprio desenvolvimento dos modelos da visão global e de processos permitirá corrigir estas situações em forma iterativa. Posteriormente, no passo seguinte de construção de ciclos de vida[41], serão analisados os problemas de integração dos diferentes processos com base em sinônimos e homônimos que podem aparecer entre as variáveis usadas como parâmetros nos estímulos e respostas.

---

[39] Vide a seção 5.1 no próximo capítulo.
[40] Vide a seção 6.1 no capítulo subseqüente.
[41] O capítulo 5 está completamente dedicado a este tema.

4.2.6 As Categorizações de Estímulos

A seguinte categorização dos estímulos, de acordo com diversos critérios, pode servir como guia para a definição dos mesmos no modelo da visão global. Estas categorias não pretendem ser exaustivas, e sim servir apenas de orientação para a modelagem.

Usando o critério dos fluxos de dados que acompanham os estímulos na forma de parâmetros, estes podem ser categorizados em:

- *Estímulos com dados principais*: Estes estímulos sempre apresentam um ou mais dados como parâmetros. É a entrada destes dados que justifica a existência do estímulo. Os dados requerem a execução de um processo de tratamento independente. Um estímulo para registrar o título de um artigo recebido é um exemplo de um estímulo com dados principais.
- *Estímulo com dados secundários*: Também estes estímulos sempre apresentam dados como parâmetros, porém com uma diferença em relação ao estímulo com dados principais, qual seja: os dados são fornecidos devido a uma solicitação explícita do sistema através de uma resposta gerada por outro processo. Desta forma, existe um encadeamento na execução dos processos, e o tratamento do estímulo com dados secundários depende da execução de outro processo. A partir do processo que registrou o título do artigo no exemplo acima, um estímulo para registrar o(s) autor(es) do artigo já registrado é um exemplo de um estímulo com dados secundários.
- *Estímulo de transição*: Estes estímulos não possuem parâmetros de nenhum tipo e produzem transições de estado dentro do sistema. Em outras palavras, estes estímulos modificam o comportamento do sistema sem fornecer dados. Por exemplo, o estímulo que indica que sejam enviadas as chamadas de trabalho para as pessoas incluídas na memória do sistema é um estímulo de transição.

O critério de tempo, isto é, o momento ou a regularidade em que os estímulos ocorrem, permite reconhecer as seguintes categorias:

- *Estímulo temporal*: Estes estímulos estão associados a uma medida de tempo, isto é, à ocorrência do transcurso de intervalos específicos de tempo. Podem ou

não apresentar parâmetros. Por exemplo, o estímulo de geração de um relatório mensal de vendas é um caso de estímulo temporal.

- *Estímulo de controle*: Estes estímulos podem ocorrer em qualquer instante do tempo, sem relação com qualquer medida de tempo. Podem ou não apresentar parâmetros. Por exemplo, o estímulo para gerar a lista de convidados confirmados até o momento é um estímulo de controle.

Finalmente, usando o critério da memória do sistema, isto é, o modo como ela se vê afetada pela ocorrência do estímulo, podem ser identificadas as seguintes categorias:

- *Estímulo construtivo*: São os estímulos que "constroem" o sistema, isto é, contêm informação para estruturar dados no interior do sistema. Estes dados devem fazer parte da memória do sistema. Este tipo de estímulo modifica o comportamento do sistema porque produz pelo menos uma não auto-transição no seu interior. Por exemplo, o estímulo que permite registrar a submissão de um artigo com seus dados pertence a esta categoria.
- *Estímulo de consulta*: Um estímulo de consulta é aquele que apenas *consulta* dados mantidos na memória do sistema, isto é, solicita a recuperação de informações mantidas no sistema. Este tipo de estímulo não modifica o comportamento do sistema, porque só provoca auto-transições no interior do sistema. Por exemplo, o estímulo para gerar a lista de convidados confirmados até o momento pertence a esta categoria.

4.2.7 O Modelo da Visão Global para o Problema da IFIP

O modelo da visão global consiste em dois componentes:

1. Um diagrama que mostra o contexto do sistema sob modelagem, incluindo o supra-sistema e os agentes externos relacionados.
2. Um diagrama da visão global que mostra todos os processos concorrentes do sistema conectados por meio de estímulos e respostas a diferentes agentes externos.

Adicionalmente, é construído um dicionário de dados que será completado posteriormente. Este dicionário de dados inclui definições para todos os parâmetros

que aparecem nos estímulos e respostas (com exceção das constantes) no diagrama da visão global.

O diagrama de contexto do sistema para o problema da IFIP é mostrado na figura 4.2.

O diagrama da visão global com todos os processos concorrentes do sistema de congressos da IFIP assume a forma mostrada na figura 4.7. Nesta figura são mostrados dezessete processos concorrentes e seus emissores, estímulos, respostas e receptores. O superestado de todos estes processos é o IFIP_Conference_System. A numeração dada aos processos serve apenas para facilitar a referência no modelo de processos, e não expressa nenhum tipo de seqüência.

Ambos os diagramas do modelo da visão global, junto com o dicionário de dados e os restantes modelos resultantes da aplicação da presente técnica de modelagem ao problema da IFIP, podem ser encontrados no anexo "Modelo Completo do Problema da IFIP".

A tabela 4.1 mostra as categorias às quais os estímulos da figura 4.7 pertencem. Esta tabela é mostrada apenas para facilitar a compreensão dos estímulos modelados e não faz parte da técnica de modelagem aqui apresentada.

O dicionário de dados mínimo para todos os dados envolvidos nos tipos de estímulos e respostas que aparecem no diagrama da visão global do sistema é o da figura 4.8. A notação utilizada é a do dicionário de dados da análise estruturada moderna [YOU 90].

4.3 Construção do Modelo de Processos

A decomposição do modelo da visão global do sistema é a atividade que segue à construção da visão global do sistema, vista na seção anterior. Seguindo esta seqüência, adota-se, claramente, uma estratégia *top-down* de modelagem. Contudo, as iterações entre estes dois passos são recomendadas com o intuito de garantir consistência e completude entre o modelo da visão global e o modelo de processos.

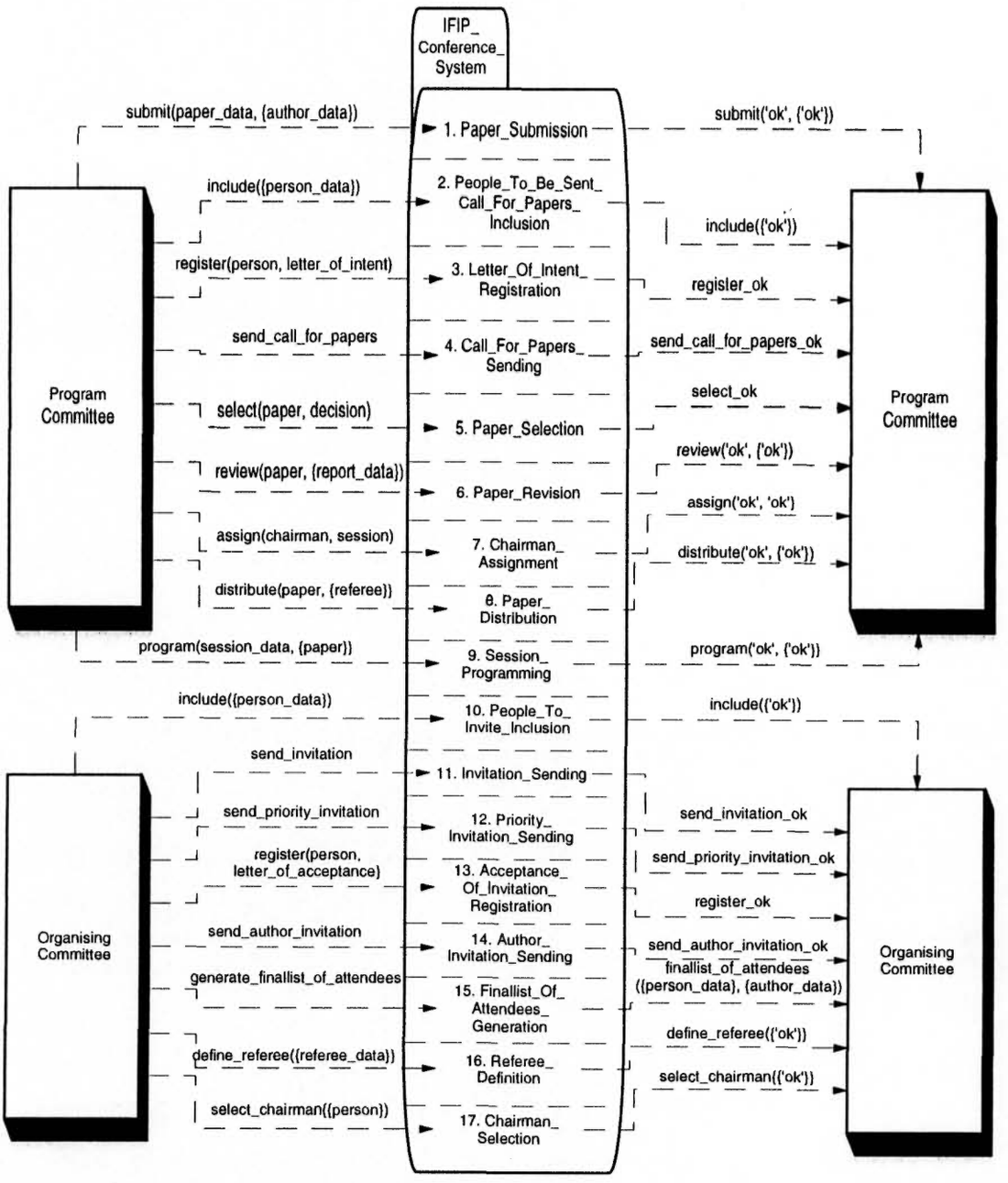


FIGURA 4.7 - O diagrama da visão global do IFIP_Conference_System.

O principal objetivo deste passo é "abrir" as caixas pretas conhecidas como processos no modelo da visão global. Cada processo deve ter o seu comportamento descrito em detalhe, usando HLS para mostrar como é tratado o estímulo para obter a resposta.

TABELA 4.1 - Categorias para os estímulos mostrados no diagrama da visão global da figura 4.7.

	FLUXO DE DADOS			TEMPO		MEMÓRIA DO SISTEMA	
ESTÍMULO	COM DADOS PRINCIPAIS	COM DADOS SECUNDÁRIOS	DE TRANSIÇÃO	TEMPORAL	DE CONTROLE	CONSTRUTIVO	DE CONSULTA
assign(chairman, session)	√				√	√	
define_referee({referee_data})	√				√	√	
distribute(paper, {referee})	√				√	√	
generate_finallist_of_attendees					√		√
include({person_data}) (processo 2)	√				√	√	
include({person_data}) (processo 10)	√				√	√	
program(session_data, {paper})	√				√	√	
register(person, letter_of_acceptance)	√				√	√	
register(person, letter_of_intent)	√				√	√	
review(paper, {report_data})	√				√	√	
select(paper, decision)	√				√	√	
select_chairman({person})	√				√	√	
send_author_invitation			√		√		
send_call_for_papers			√		√		
send_invitation			√		√		
send_priority_invitation			√		√		
submit(paper_data, {author_data})	√				√	√	

```
DATA DICTIONARY

author_data = @author + name + institution + address
decision = [ 'accepted' | 'rejected' ]
letter_of_acceptance = [ 'yes' | 'no' ]
letter_of_intent = [ 'intention' | 'no intention' ]
paper_data = @paper + title + subject
person_data = @person + name + institution + person_title
person_title = [ 'normal' | 'national representative' | 'working group member' ]
report_data = @report + referee + concept
session_data = @session + room + time
referee_data = @person + maximum
```

FIGURA 4.8 - Dicionário de dados mínimo para a figura 4.7.

Assim, o modelo de processos é um conjunto de diagramas HLS separados, cada um dos quais representando a estrutura e o comportamento de um dos processos e sua relação com o estímulo e a resposta mostrados no diagrama da visão global.

Neste nível de abstração, deixam de ser considerados explicitamente os emissores e receptores do sistema sob modelagem. Isto acontece, porque eles já foram devidamente identificados, representados e relacionados por meio de estímulos e respostas com o sistema em questão no modelo da visão global. A forma da tupla que define o sistema na visão global é reduzida agora para (estímulo, processo, resposta).

Pela própria natureza desta atividade, as decisões do modelador pesarão fortemente no desenvolvimento dos modelos dos processos e terão um forte impacto nos passos e atividades seguintes até a conclusão no modelo estrutural orientado a objetos.

É importante destacar que o dicionário de dados desenvolvido durante a construção do modelo da visão global deve ser completado com os eventuais novos dados que possam surgir, assim como devem ser corrigidas todas as inconsistências que possam existir entre as definições realizadas no modelo da visão global e os detalhamentos realizados no modelo de processos.

Para mostrar o resultado da construção do modelo de processos, a figura 4.9(b) mostra um exemplo de um diagrama HLS de um processo cuja tupla está representada na figura 4.9(a). A construção do diagrama HLS é feita a partir do estímulo e da resposta, assim como da semântica do próprio processo. A situação representada é a submissão de um artigo que possui um ou mais autores. Este(s) autor(es) pode(m):

1. Ter enviado previamente uma carta de intenção para participar do congresso, o que é representado pelo estado On_Intent e transição para o estado Participant (o autor participa efetivamente contribuindo com um artigo).
2. Não ter enviado uma carta de intenção (não deve ser verificado o estado) e é sua primeira contribuição com um artigo. Isto é representado pela inicialização do estado Participant para este autor.
3. Depois de ter acontecido qualquer uma das duas situações anteriores, o autor participa com mais de um artigo. Isto é representado pela emulação de um estado ativo com o estado Participant e uma auto-transição.

Em relação aos modelos apresentados na figura 4.9, cabe indicar algumas observações:

- O processo assume uma estrutura concorrente, sendo que cada subestado concorrente trata de um dos parâmetros, isto é, um trata do(s) autor(es) e o outro

trata do artigo. Ambos os subestados apresentam variáveis de referência que também são componentes das variáveis compostas usadas como parâmetros do estímulo (author_data e paper_data).

- O subestado Author_Status(author) requer, neste processo, os estados iniciais alternativos On_Intent e Participant (que também pode ser inicializado). O subestado Paper_Status(paper) apresenta um único estado inicializado Submitted.
- O estímulo submit(paper_data, {author_data}), que aparece na visão global, é decomposto no conjunto de eventos concorrentes { submit(paper, author_data) } e no evento submit(paper_data, {author}), todos internos ao processo.
- As transições são todas baseadas em eventos, isto é, todas apresentam um evento que determina (junto com a condição) o seu disparo.
- Ambos subestados concorrentes apresentam estados finais: Participant em Author_Status(author) e Submitted em Paper_Status(paper).
- As transições apresentam condições. Algumas servem para verificar a inicialização de estados (por exemplo, author NOT exists e paper NOT exists), outras, para garantir consistência no interior do processo ao sincronizar transições (condições da forma into Estado). Observe-se que também é possível usar uma variável tipo lista em uma condição, como por exemplo em into Author_Status({author}).Participant, que representa a verificação do conjunto de estados destinos para as outras transições sincronizadas.
- As transições também apresentam ações. Algumas servem especificamente para a inicialização de estados (por exemplo, participate author e submit paper), outras dão conformidade parcial do tratamento do processo. As ações de conformidade { submit(null, 'ok') } e submit('ok', null) compõem a resposta submit('ok', 'ok') que aparece na visão global.

Todos estes elementos dos diagramas da visão global e HLS de processos são descritos em detalhe nas próximas seções.

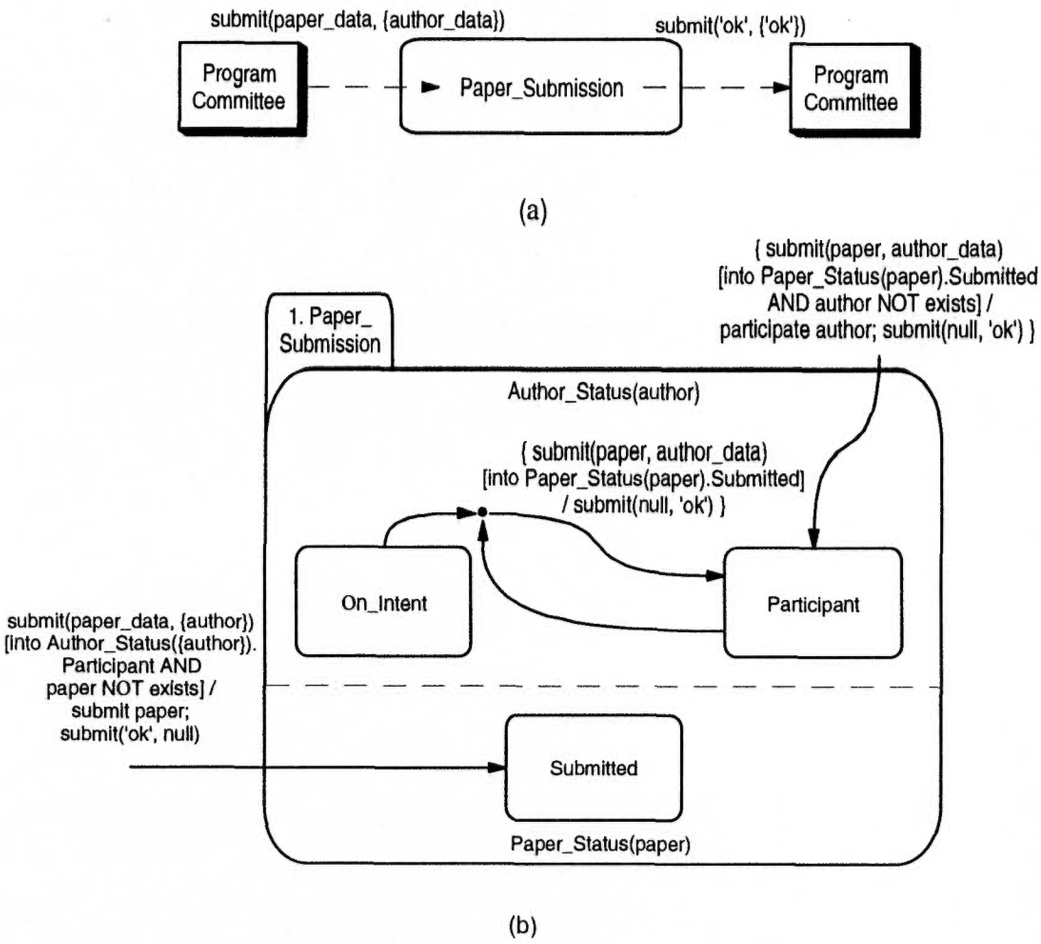


FIGURA 4.9 - Exemplos específicos de: (a) uma tupla do diagrama da visão global; (b) o diagrama HLS do processo respectivo.

4.3.1 Modelagem dos Processos

O diagrama da visão global mostra cada processo dentro de uma tupla (emissor, estímulo, processo, resposta, receptor). É esta a informação disponível para começar a modelagem de um processo qualquer com HLS.

Para modelar um processo usando HLS, é necessário responder às seguintes perguntas:

- Qual a estrutura geral que apresenta o processo? Dado que, por definição, um processo é um superestado, a questão pode ser redefinida para: qual a relação entre os subestados do processo?

- Considerando o estímulo, qual a variável de referência para o processo ou subestados constituintes?
- Quais estados definem inicialmente o processo? É necessário um estado inicial? É necessário inicializar algum estado?
- Como o estímulo recebido pelo processo relaciona-se com os eventos internos?
- Quais as transições entre os estados definidos?
- Quais estados concluem o processo? São necessários estados finais? É necessário finalizar algum estado?
- Quais as funções de mapeamento, condições e ações associadas às transições?
- De que forma as ações relacionam-se com a resposta gerada pelo processo?

É claro que a ordem dada às perguntas acima pode ser modificada, porém esta seqüência permite uma primeira aproximação à organização dos elementos a serem considerados na modelagem do comportamento do processo.

Duas estratégias gerais podem ser citadas no modo de construir o modelo de comportamento de um processo usando HLS:

- Tentar identificar todos os estados possíveis que o processo deve apresentar de forma a poder transformar o estímulo em uma resposta. A seguir, deve-se estabelecer as transições entre os estados dando seqüência aos eventos.
- Começar com algum estado inicial ou um estado inicializado e ir definindo as transições com os eventos associados na seqüência para definir o estado seguinte. Este procedimento deve ser repetido para cada novo estado identificado até resultar em estados finais ou em finalização de estados.

Independentemente da estratégia que possa ser adotada em cada caso, as seguintes seções fornecem elementos que permitem responder às perguntas levantadas acima.

4.3.1.1 A Estrutura Geral do Processo

Um diagrama HLS é representado por um superestado. Este é o ponto de partida da modelagem de um processo: é um superestado cuja estrutura deve ser definida.

Não é por acaso que a representação adotada para os processos dentro do diagrama da visão global "lembra" a de um estado. Mais ainda, os processos do conjunto de tuplas são representados como concorrentes também no sentido dos HLS. Cada processo é decomposto em estruturas hierárquicas de estados e transições que podem operar em paralelo.

As estruturas gerais que o processo pode apresentar classificam-se em:

1. **Simples**: As estruturas simples apresentam um estado com subestados não genéricos, isto é, sem assumir uma forma de conjunto representado genericamente. As estruturas simples podem dividir-se ainda em:
    - **Únicas**: Apresentam um único subestado.
    - **Concorrentes**: Os subestados são concorrentes.
    - **Exclusivas**: Os subestados são exclusivos.
2. **Compostas**: As estruturas compostas apresentam um conjunto de subestados expressos simultaneamente. As estruturas compostas podem ser divididas em:
    - **Concorrentes**: Os subestados do conjunto são concorrentes.
    - **Exclusivas**: Os subestados do conjunto são exclusivos.

Esta classificação de estruturas é recursiva. Assim por exemplo, um processo pode apresentar uma estrutura simples concorrente com dois subestados. Por sua vez, um dos subestados pode assumir uma estrutura composta concorrente e o outro pode assumir uma estrutura simples exclusiva. O caso da figura 4.9(b) apresenta um processo com estrutura simples concorrente. O subestado Author_Status(author) possui uma estrutura composta concorrente, e cada subestado desta última estrutura possui uma estrutura simples exclusiva. O subestado Paper_Status(paper) apresenta uma estrutura composta concorrente, e cada subestado desta estrutura apresenta uma estrutura simples única.

A determinação de qual destas estruturas é a apropriada para um processo dado depende, fundamentalmente, da forma que assume o estímulo, da semântica do processo e, em menor grau, do conteúdo da resposta emitida.

Em relação ao estímulo, e como regra geral, deve ser sempre considerada a possibilidade de se tratá-lo separadamente e em paralelo:

- Quando houver dois ou mais parâmetros no estímulo, cada dado que é parâmetro, independente do seu tipo, é tratado em um subestado concorrente. Alguns exemplos de estruturas simples concorrentes derivadas a partir dos parâmetros do estímulo são mostrados na tabela 4.2 (as variáveis de referência são omitidas por simplicidade).

TABELA 4.2 - Exemplos específicos de processos com estrutura simples concorrente derivada do estímulo com vários parâmetros.

ESTÍMULO	1º SUBESTADO CONCORRENTE	2º SUBESTADO CONCORRENTE
assign(chairman, session)	Chairman_Status	Session_Status
distribute(paper, {referee})	Paper_Status	Referee_Status
program(session_data, {paper})	Session_Status	Paper_Status
review(paper, {report_data})	Paper_Status	Report_Status
submit(paper_data, {author_data})	Paper_Status	Author_Status

- Quando o tipo de algum dos parâmetros for uma lista, deve ser usada uma estrutura composta concorrente com eventos "repetidos" para aquele parâmetro. Cada elemento da lista é tratado individual e simultaneamente em cada subestado do conjunto concorrente[42]. Outros tipos de parâmetros também indicam estruturas compostas concorrentes, porém sem precisar de eventos "repetidos" ou concorrentes da forma que as listas precisam. Exemplos de estruturas compostas concorrentes derivadas a partir de um parâmetro na forma de lista no estímulo são mostrados na tabela 4.3 (as variáveis de referência são omitidas por simplicidade).

[42] Esta decomposição é descrita em detalhe na seção 4.3.1.4.

TABELA 4.3 - Exemplos específicos de processos com estrutura composta concorrente derivada do parâmetro lista do estímulo.

ESTÍMULO	SUPERESTADO COM SUBESTADOS CONCORRENTES
define_referee({referee_data})	Referee_Status
distribute(paper, {referee})	Referee_Status
include({person_data}) (processo 2)	Person_Status
include({person_data}) (processo 10)	Person_Status
program(session_data, {paper})	Paper_Status
review(paper, {report_data})	Report_Status
select_chairman({person})	Chairman_Status
submit(paper_data, {author_data})	Author_Status

- Quando houver variáveis compostas como parâmetros, os dados componentes devem ser tratados concorrentemente se possível. Por exemplo, o estímulo submit(paper_data, {author_data}) poderia ter sido definido apenas como submit(paper_data), com as seguintes definições no dicionário de dados:

  paper_data = @paper + title + subject + { author_data }
  author_data = @author + name + institution + address

  Os dados componentes relativos ao artigo e ao(s) autor(es) podem ser tratados concorrentemente, sugerindo assim uma estrutura simples concorrente. No entanto, para maior clareza e simplicidade, é preferível a forma de parâmetros separados à forma de parâmetros de variáveis compostas.

A determinação da forma geral da estrutura do processo a partir da forma do estímulo é, na realidade, uma relação de mão dupla, isto é, também é possível redefinir a forma do estímulo a partir de uma estrutura geral adotada para o processo. Isto pode ocorrer particularmente quando existem dados componentes em um parâmetro de variável composta que podem e devem ser tratados concorrentemente. Neste caso, o estímulo pode ser modificado de modo a apresentar não um parâmetro composto e sim dois ou mais parâmetros.

Contudo, existem algumas exceções às regras acima:

- Quando, apesar do estímulo apresentar dois ou mais parâmetros, o processo não assume a estrutura simples concorrente. Este é o caso de que um ou mais dos parâmetros não requer um tratamento específico, isto é, um dos parâmetros serve como informação apenas para o tratamento de outro parâmetro. Exemplos de es-

truturas simples não concorrentes e estímulos com dois ou mais parâmetros são mostrados na tabela 4.4 (as variáveis de referência são omitidas por simplicidade).

TABELA 4.4 - Exemplos específicos de exceções de processos com estrutura simples não concorrente.

ESTÍMULO	SUBESTADOS NÃO CONCORRENTES	CAUSA DA EXCEÇÃO
register(person, letter_of_acceptance)	Person_Status e Author_Status são concorrentes*	*letter_of_acceptance* indica se *person* ou *author* aceitou ou não o convite
register(person, letter_of_intent)	Person_Status	*letter_of_intent* indica se *person* pretende participar do congresso
select(paper, decision)	Paper_Status	*decision* indica se *paper* foi ou não aceito

* Esta concorrência não é devida ao número de parâmetros do estímulo.

- Quando, apesar do estímulo apresentar um ou nenhum parâmetro efetivamente tratado, o processo assume a estrutura simples concorrente. Este é o caso em que um mesmo parâmetro requer tratamento concorrente em subestados diferentes e/ou a própria semântica do processo implica em tratamentos concorrentes. Os exemplos de estruturas simples concorrentes não derivadas a partir de dois ou mais parâmetros do estímulo são mostrados na tabela 4.5 (as variáveis de referência são omitidas por simplicidade).

TABELA 4.5 - Exemplos específicos de exceções de processos com estrutura simples concorrente não derivada do número de parâmetros do estímulo.

ESTÍMULO	1º SUBESTADO CONCORRENTE	2º SUBESTADO CONCORRENTE	CAUSA DA EXCEÇÃO
generate_finallist_of_attendees	Person_Status	Author_Status	tanto *person* como *author* devem ser incluídos na lista de participantes
register(person, letter_of_acceptance)	Person_Status	Author_Status	*person* é tratado como *person* e como *author*

- Quando, apesar do estímulo não apresentar parâmetros do tipo lista, o processo assume a estrutura composta concorrente com eventos "repetidos". Este é o caso

em que a semântica do processo requer um tratamento sobre um conjunto de estados concorrentes que satisfazem alguma condição não expressa diretamente pelo parâmetro. Exemplos de estruturas compostas concorrentes não derivadas a partir de um parâmetro na forma de lista no estímulo são mostrados na tabela 4.6 (as variáveis de referência são omitidas por simplicidade).

TABELA 4.6 - Exemplos específicos de exceções de processos com estrutura composta concorrente não derivada de listas nos parâmetros do estímulo.

ESTÍMULO	SUPERESTADO COM SUBESTADOS CONCORRENTES	CAUSA DA EXCEÇÃO
send_author_invitation	Author_Status	*author_invitation* é enviado a *author* que vai participar do congresso
send_call_for_papers	Person_Status	*call_for_papers* é enviado a *person* que tenha sido incluída previamente
send_invitation	Person_Status	*invitation* é enviado a *person* que tenha sido incluída previamente
send_priority_invitation	Person_Status	*priority_invitation* é enviado a *person* que tenha sido incluída previamente

Não existe a possibilidade do estímulo apresentar parâmetros do tipo lista, e o processo não assumir a estrutura composta concorrente, porque, por definição, os elementos de uma lista são sempre tratados concorrentemente.

4.3.1.2 As Variáveis de Referência e os Nomes de Estados

Para estabelecer a variável de referência de cada subestado concorrente ou exclusivo, usa-se o parâmetro (se ele for um identificador único) ou algum dado componente do mesmo (consultando o dicionário de dados neste último caso), que serve de identificador único para o parâmetro composto.

A regra geral é a seguinte: *todo identificador único presente no estímulo na forma de um parâmetro define a variável de referência do subestado onde é tratado.* Nos casos em que o estímulo não apresente parâmetros, pode ser utilizado o identificador único para o dado que está implicitamente sendo tratado. A relação definida acima também é de mão dupla: a definição de uma variável de referência pode conduzir a uma redefinição do identificador único no estímulo.

Para simplificar os passos seguintes da técnica de modelagem descrita neste trabalho, é recomendável padronizar os nomes dos superestados, particularmente aqueles que apresentam variáveis de referência associadas. Para isto pode ser usado o mesmo nome da variável de referência mais o sufixo _Status. A tabela 4.7 mostra todos os estímulos, os superestados e subestados que definem a estrutura do processo com as respectivas variáveis de referência.

TABELA 4.7 - Exemplos específicos de superestados, subestados e variáveis de referência derivados dos estímulos.

ESTÍMULO	SUPERESTADO E SUBESTADOS COM VARIÁVEIS DE REFERÊNCIA
assign(chairman, session)	Session_Status(session) AND Chairman_Status(chairman)
define_referee({referee_data})	Referee_Status(referee)
distribute(paper, {referee})	Paper_Status(paper) AND Referee_Status(referee)
generate_finallist_of_attendees	Person_Status(person) AND Author_Status(author)
include({person_data}) (processo 2)	Person_Status(person)
include({person_data}) (processo 10)	Person_Status(person)
program(session_data, {paper})	Session_Status(session) AND Paper_Status(paper)
register(person, letter_of_acceptance)	Person_Status(person) AND Author_Status(author)
register(person, letter_of_intent)	Person_Status(person)
review(paper, {report_data})	Paper_Status(paper) AND Report_Status(report)
select(paper, decision)	Paper_Status(paper)
select_chairman({person})	Chairman_Status(chairman)
send_author_invitation	Author_Status(author)
send_call_for_papers	Person_Status(person)
send_invitation	Person_Status(person)
send_priority_invitation	Person_Status(person)
submit(paper_data, {author_data})	Author_Status(author) AND Paper_Status(paper)

O maior superestado em um diagrama HLS, isto é, o superestado que representa o processo como um todo, sempre possui um nome: é o que foi dado ao processo na tupla correspondente. Este nome, junto com o número de referência, é anotado em cada diagrama HLS, no topo da hierarquia de estados, ainda que já exista um superestado com nome. Vide, por exemplo, os superestados e os nomes dos processos da figura 4.10.

4.3.1.3 O Início do Processo

Existem duas formas a partir das quais o processo inicia: um estado é inicializado ou existe um *pré-estado*.

A inicialização de estados é caracterizada pela necessidade de incluir novos dados no sistema. Isto é, uma nova instância de uma determinada variável deve ser criada em função do valor dado por um dos parâmetros do evento.

No problema da IFIP existem vários exemplos de inicialização de estados como início de processo. A figura 4.10 mostra dois destes exemplos[43]. O diagrama HLS da figura 4.10(a) representa o processo de criar uma lista de pessoas para as quais deve ser enviada uma chamada de trabalhos (call_for_papers). O diagrama HLS da figura 4.10(b) mostra o processo de criar uma lista de pessoas a serem convidadas para o congresso.

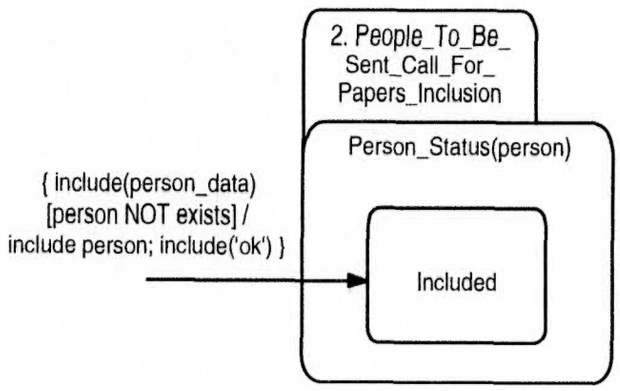
O caso da figura 4.10 é de estruturas compostas. Em estruturas simples concorrentes também é possível identificar inicialização de estados em algum dos subestados do diagrama HLS. A figura 4.11 mostra diagramas HLS incompletos, em que acontece inicialização de estados no problema da IFIP. No caso do processo Paper_Revision está se criando um conjunto de relatórios. No caso do processo Session_Programming, está se criando uma sessão de apresentação.

Um *pré-estado* é um estado inicial em (ou parte de) um processo que é necessário para a execução do mesmo, isto é, o processo deve encontrar-se neste estado antes da recepção do estímulo. Todo (ou parte de) o tratamento realizado pelo processo origina-se no pré-estado. Em outras palavras, é a condição, expressa em termos de um estado pré-existente, que deve ser satisfeita no momento da recepção do estímulo para que o processo possa ser executado.

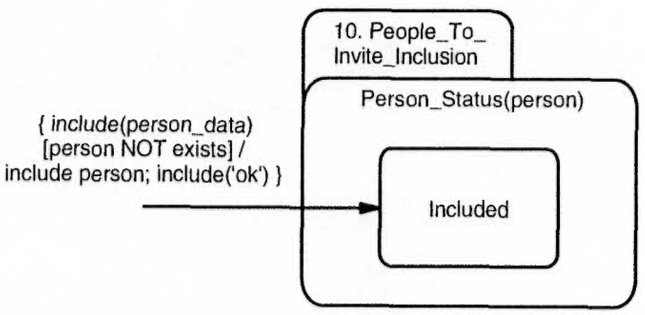
A figura 4.12 mostra dois exemplos de pré-estados em diagramas HLS no problema da IFIP. No caso do processo Letter_Of_Intent_Registration, é necessário que, previamente, tenha sido enviada à pessoa a chamada de trabalhos para poder registrar sua carta de intenção para participar no congresso. No caso do processo Pa-

[43] Acompanha estas figuras e outros diagramas HLS completos o texto em inglês do requisito específico que é modelado pelo diagrama respectivo.

per_Selection é necessário que o artigo tenha sido previamente revisado para poder ser selecionado.



(a) Requisito modelado: "*Preparing a list to whom the call for papers is to be sent*" [OLL 82].



(b) Requisito modelado: "*Preparing a list of people to invite to the conference*" [OLL 82].

FIGURA 4.10 - Exemplos específicos de inicialização de estados na modelagem de processos usando HLS.

Outro exemplo é o de pré-estados concorrentes, como, por exemplo, o mostrado na figura 4.13. No processo da figura, é necessário que tanto a sessão tenha sido criada, quanto que o moderador esteja disponível para poder alocar o mesmo em uma sessão.

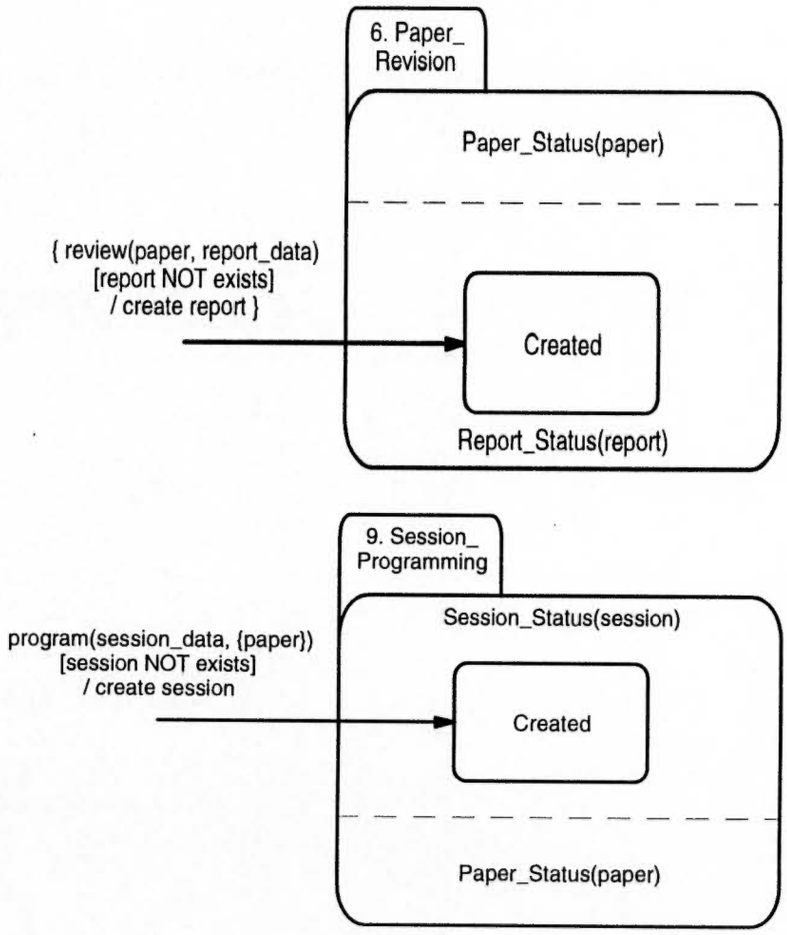


FIGURA 4.11 - Exemplos específicos de inicialização de estados em diagramas HLS incompletos.

4.3.1.4 A Relação entre o Estímulo e os Eventos

Como já foi indicado anteriormente, o estímulo é uma comunicação externa que o sistema recebe de um emissor. Este estímulo é tratado por um processo do sistema. Como o processo é decomposto em estados usando HLS, as transições entre eles requerem eventos que devem ser derivados do estímulo recebido pelo processo. Esta relação é de decomposição, isto é, na perspectiva do mais abstrato para o mais detalhado: *o estímulo é decomposto em um ou mais eventos relacionados*; ou na perspectiva inversa: *o evento é um componente de um estímulo.*

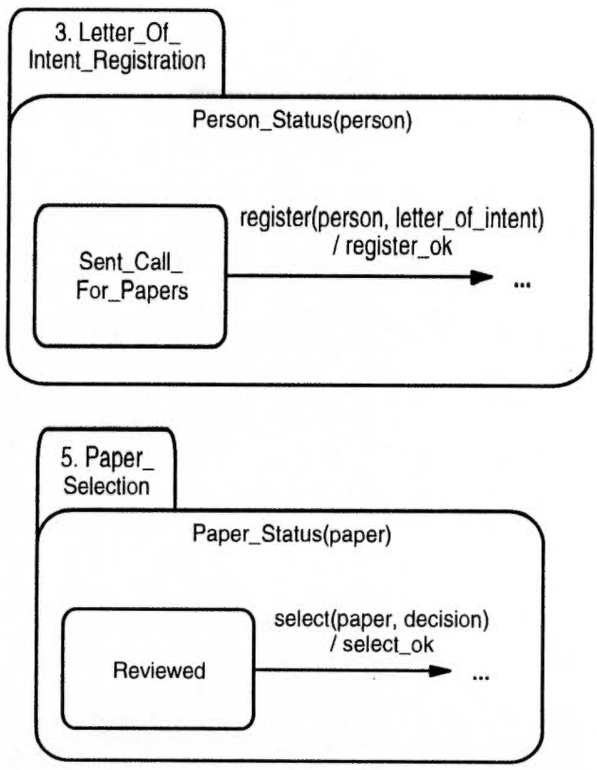


FIGURA 4.12 - Exemplos específicos de pré-estados em diagramas HLS incompletos.

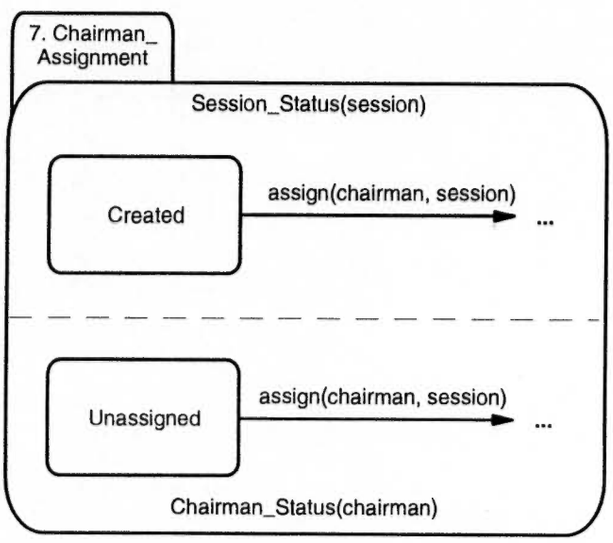


FIGURA 4.13 - Exemplo específico de pré-estados concorrentes em um diagrama HLS incompleto.

Nem todos os eventos são decompostos a partir de um estímulo. Aqueles eventos gerados a partir de ações internas ao processo constituem a exceção. Estes eventos e as ações que os geram são descritos na seção 4.3.1.9.

Uma das diferenças entre o estímulo e os eventos é que o estímulo é *externo* e relaciona o sistema com os emissores, e os eventos são *internos* aos processos, relacionando seus estados por meio das transições. A partir de outro ponto de vista, os estímulos e os eventos pertencem a níveis de abstração diferentes: o estímulo é um elemento do modelo da visão global do sistema, os eventos são elementos do modelo de processos, que é um detalhamento do modelo da visão global.

Esta relação também é de mão dupla. A decomposição de um estímulo em eventos pode levar a uma redefinição do estímulo.

Em termos de notação, não existe nenhuma diferença entre estímulo e evento. Os nomes de ambos são formados da mesma maneira e ambos podem apresentar parâmetros. A única diferença é que o nome do estímulo é anotado sobre uma seta de linha descontínua, e o do evento, sobre uma seta de linha contínua. Esta diferença ocorre porque, no último caso, existe o conceito de transição, e no primeiro, não.

A decomposição do estímulo em eventos é um dos elementos centrais na relação entre os modelos da visão global e de processos[44]. O mapeamento do estímulo em eventos é feito simultaneamente, com a explosão do processo em um conjunto de estados hierarquizados em um diagrama HLS.

A decomposição de um estímulo ocorre de diversas formas, dependendo tanto da semântica do processo, quanto da forma como se apresentam os parâmetros do estímulo e a estrutura geral assumida pelo processo.

Em termos de notação, a indicação de que o evento é uma decomposição de um determinado estímulo é a seguinte: os eventos conservam o mesmo nome do estímulo, apenas variando na forma e no tipo dos parâmetros.

---

[44] O outro é a composição de ações em uma resposta. Vide seção 4.3.1.10.

A situação mais simples possível de decomposição é quando o evento é exatamente o estímulo, isto é, o estímulo e o evento são equivalentes (e portanto possuem também o mesmo nome). A rigor, não existe decomposição, e o mesmo estímulo é considerado como um evento no interior do processo. Em termos gerais, a equivalência entre estímulo e evento pode ocorrer quando não houver parâmetros do tipo lista no estímulo do processo, porque isto significa que o estímulo deve ser decomposto em um conjunto de eventos concorrentes. Mais especificamente, podem ser identificados os seguintes casos de equivalência entre estímulo e evento:

- Quando o subestado de um processo com estrutura composta concorrente assume uma estrutura simples única, e o evento não se encontra "repetido". No problema da IFIP não existe um exemplo de um processo com estas características. No entanto, este seria o caso se fossem simplificados os requisitos para os diagramas da figura 4.10: em vez de tratar de listas de pessoas, a inclusão pode ser individual em cada estímulo, como é mostrado na figura 4.14. Em ambos os casos, o estímulo seria include(person_data), exatamente o mesmo evento que inicializa os estados nos processos.
- Quando o subestado do processo (de estrutura composta concorrente) assume uma estrutura simples exclusiva. A figura 4.12 mostra dois exemplos deste caso.
- Quando o processo assume uma estrutura simples concorrente, nenhum dos parâmetros é uma lista, e todos os parâmetros são necessários em cada subestado concorrente. Como exemplo deste caso, pode ser indicado o diagrama da figura 4.13.
- Quando o estímulo não apresenta parâmetros. Como no exemplo da figura 4.15, em que o estímulo generate_finallist_of_attendees é interpretado como dois conjuntos de eventos concorrentes, todos equivalentes entre si e com o estímulo. Este diagrama representa a geração de uma lista de todos os participantes, sejam autores ou não, que aceitaram o convite.

As formas de decomposição de estímulos em eventos podem ser categorizadas em:

- Estímulo apresenta dois ou mais parâmetros e cada um deles é tratado em um subestado concorrente. Nesta forma de decomposição, existe um evento específico para cada subestado que trata de um dos parâmetros do estímulo. Assim, existirá um evento componente por cada parâmetro existente no estímulo. A forma geral

desta decomposição é mostrada na figura 4.16. Para cada subestado existe um evento com o parâmetro que está sendo tratado em evidência, os restantes parâmetros são substituídos pelo parâmetro null (sempre que nenhum dos outros parâmetros esteja sendo usado em alguma função de mapeamento, condição ou ação no subestado de tratamento de um parâmetro). Como isto quase não ocorre nos diagramas HLS, é possível modificar os eventos do diagrama do exemplo da figura 4.13 para obter o esquema de decomposição da figura 4.17, assumindo que o parâmetro session não é usado onde chairman está sendo tratado e vice versa.

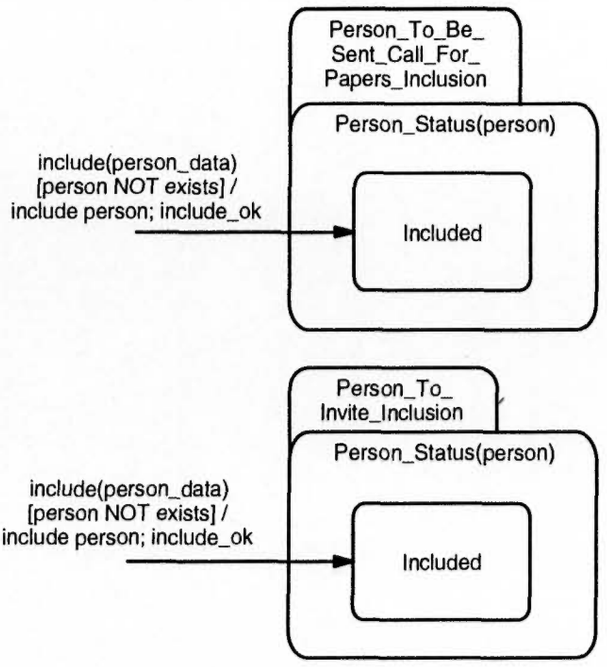
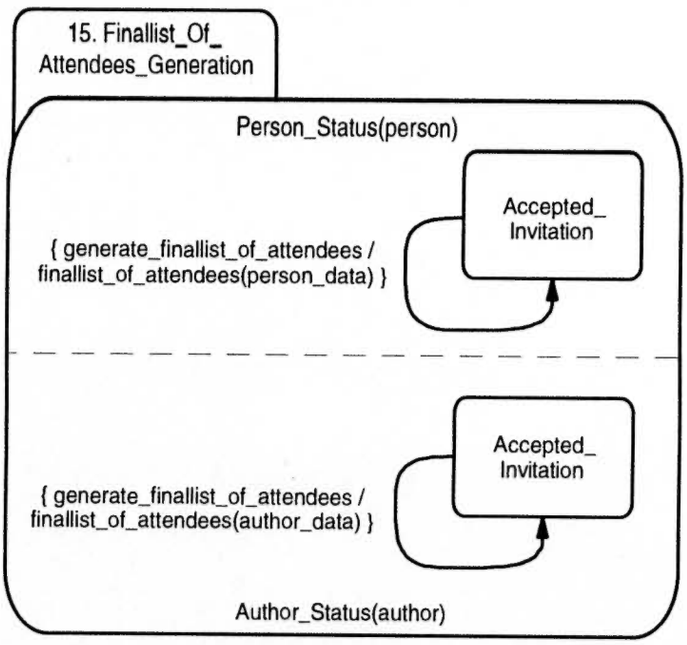


FIGURA 4.14 - Exemplo específico hipotético de processos com subestado de estrutura simples única e equivalência entre estimulo e evento.

- Estímulo apresenta um parâmetro na forma de lista. A decomposição ocorre de tal maneira que cada um dos elementos da lista torna-se parâmetro em um evento. O resultado desta decomposição é um conjunto de eventos concorrentes que reflete os elementos da lista do parâmetro e a estrutura composta concorrente do processo, como é mostrado na figura 4.18. Exemplos específicos desta forma de

decomposição são os da figura 4.10. O estímulo de cada um destes processos é include({person_data}), e o conjunto de eventos concorrentes é representado por { include(person_data) }. O esquema de decomposição para ambos os casos é mostrado na figura 4.19.



Requisito modelado: "*Generating finallist of attendees.*" [OLL 82].

FIGURA 4.15 - Exemplo específico de equivalência entre um estímulo sem parâmetros e os eventos.

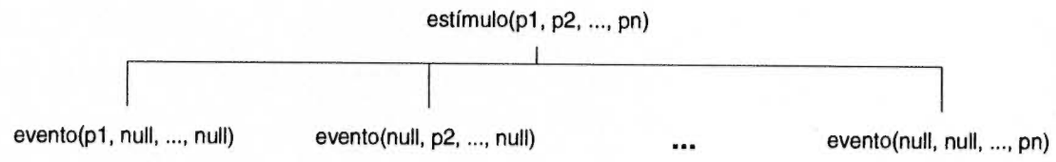


FIGURA 4.16 - Esquema de decomposição geral de um estímulo com vários parâmetros.

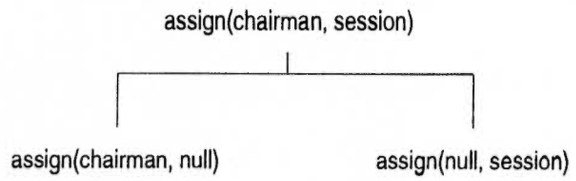


FIGURA 4.17 - Esquema de um exemplo específico hipotético de decomposição de um estímulo com dois parâmetros.

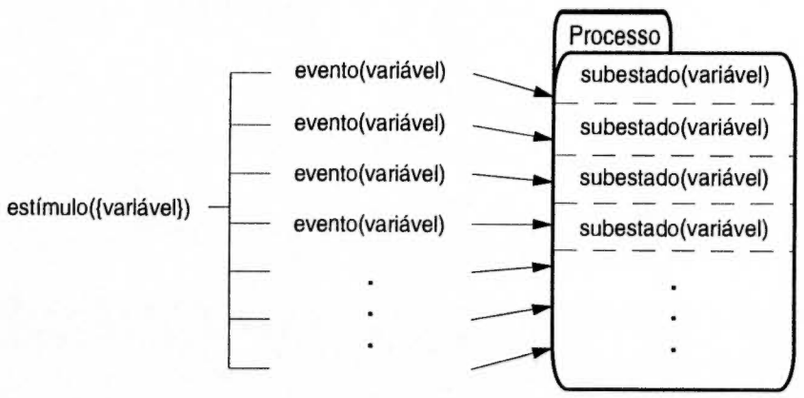


FIGURA 4.18 - Esquema de decomposição de um estímulo com parâmetro de tipo lista em um conjunto de eventos concorrentes associados à estrutura composta concorrente do processo.

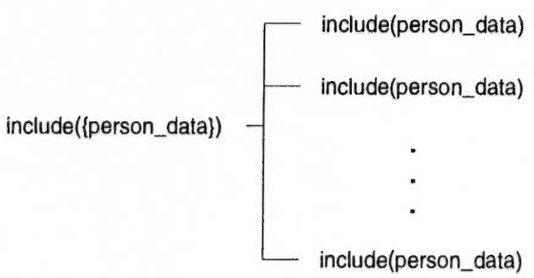


FIGURA 4.19 - Esquema de decomposição específica dos estímulos da figura 4.10.

Todas as formas de decomposição acima descritas (incluindo a de equivalência) podem ser combinadas. Por exemplo, o caso do estímulo submit(paper_data,

{author_data}) da figura 4.9(a) apresenta dois parâmetros tratados concorrentemente, sendo um deles uma lista. O esquema desta decomposição ideal é mostrado na figura 4.20(a). Um esquema mais comum é o da figura 4.20(b), que mostra os eventos do diagrama HLS da figura 4.9(b). A diferença se deve ao fato de que, no segundo caso, são considerados os parâmetros que estão sendo usados em outros subestados que não o próprio. Observe-se que nas figuras 4.9(b) e 4.20(b) não é necessário todo o parâmetro composto (author_data ou paper_data) no evento pertencente ao tratamento do outro parâmetro, bastando apenas o identificador único do dado (author e paper, respectivamente).

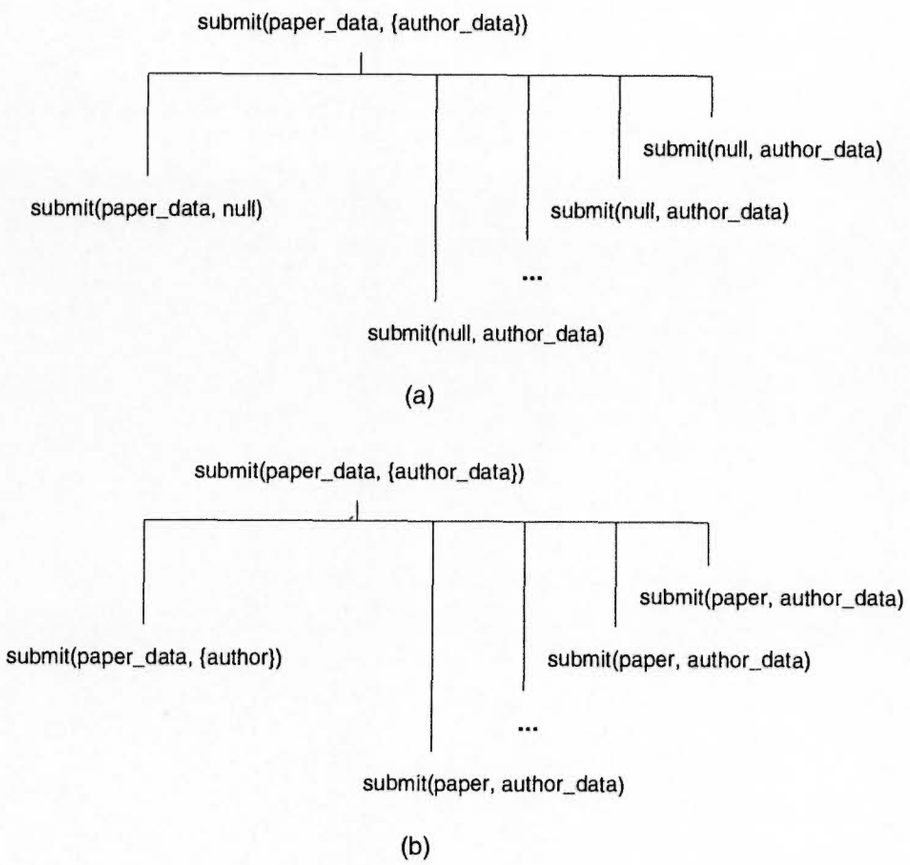


FIGURA 4.20 - Esquemas de decomposição específica de um estímulo: (a) ideal; (b) mais comum.

Ainda é possível outra combinação em que exista um evento equivalente ao estímulo e, também, eventos componentes. Isto ocorre pela necessidade de estarem todos os parâmetros do estímulo em um dos subestados concorrentes (o que torna o evento equivalente ao estímulo), e apenas de alguns deles em outro subestado. Por exemplo, a figura 4.21 mostra o esquema de decomposição para um caso destes.

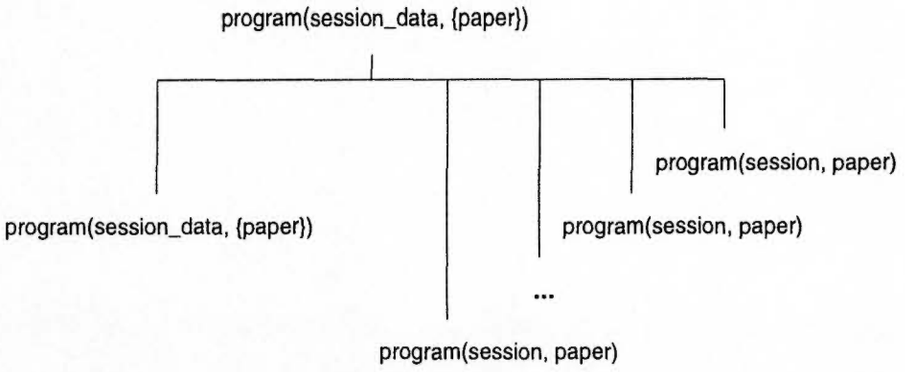


FIGURA 4.21 - Esquema de decomposição específica de um estímulo com eventos equivalentes.

4.3.1.5 As Transições entre Subestados

Em termos gerais, uma transição entre dois estados deve ser introduzida quando for necessária uma mudança de comportamento no processo. Isto significa que a transição representa uma "memorização" que o processo deve realizar.

Esta concepção da transição considera apenas o efeito da ocorrência de um evento para o disparo da mesma. No entanto, nos HLS, a transição supõe além disso, uma condição e uma ação. Assim, a transição deve ser introduzida se alguma condição deve ser verificada, ou se uma ação específica dentro do processo deve ser realizada.

É possível ter qualquer combinação destes elementos em uma transição que pretenda ser introduzida entre dois estados, sempre que exista um evento, ou uma condição, ou ambos.

Todas as transições no problema da IFIP são baseadas principalmente em eventos, isto é, não existem transições que dependam unicamente de uma condição. Vide por exemplo as transições dos diagramas HLS das figuras 4.9 à 4.13, todas elas têm um evento como elemento central para disparar a transição.

4.3.1.6 A Conclusão do Processo

Existem duas formas em que um processo pode concluir: um estado é finalizado ou resta um *pós-estado*.

A finalização de estados é caracterizada pela necessidade de eliminar dados existentes no sistema. Isto é, uma instância de uma determinada variável deve ser excluída em função do valor dado por uns dos parâmetros do evento.

No problema da IFIP não existem exemplos de finalização de estados como conclusão de um processo.

Um *pós-estado* é um estado final em um processo (ou em parte de um processo) que é deixado ativo após a execução do mesmo, isto é, o processo deve concluir neste estado. Todo (ou parte de) o tratamento realizado pelo processo termina no pós-estado. Em outras palavras, é a condição, expressa em termos de um estado resultante, que deve ser satisfeita no momento da conclusão do tratamento realizado pelo processo.

A figura 4.22 mostra o diagrama da figura 4.13, agora com pós-estados. No processo desta figura, tanto a sessão quanto o moderador terminam o processo nos pós-estados With_Chairman e Assigned, respectivamente.

Podem ocorrer situações onde o pós-estado seja o mesmo estado inicializado. Na figura 4.10, os estados Included são, ao mesmo tempo, estados inicializados e pós-estados.

Finalmente, um estado pode ser um pré e um pós-estado simultaneamente. Isto acontece na presença de auto-transições, como as mostradas na figura 4.15.

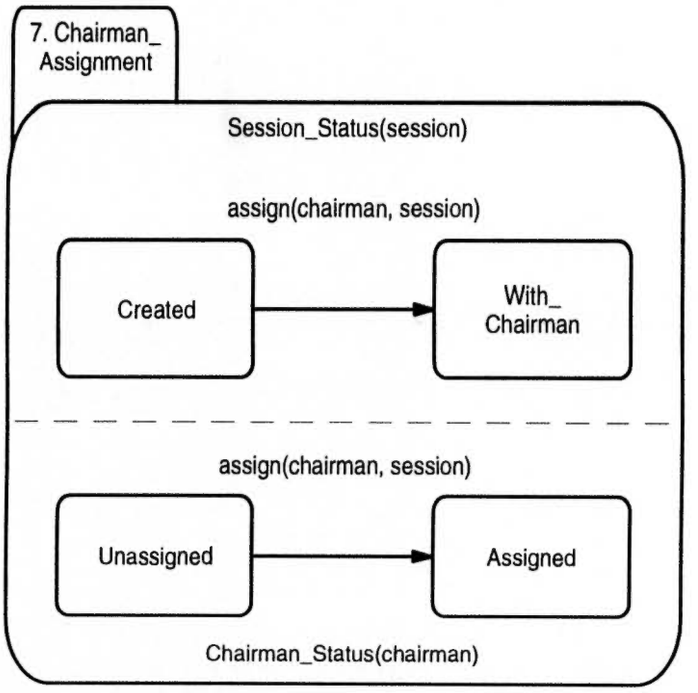


FIGURA 4.22 - Exemplo específico de pós-estados concorrentes em um diagrama HLS incompleto.

4.3.1.7 As Funções de Mapeamento

As funções de mapeamento nos HLS são usadas para descrever o estado destino de uma transição através de um novo valor de uma variável de referência. Estas funções de mapeamento devem ser especificadas, geralmente, no caso de superestados de estrutura composta exclusiva.

Se existe a necessidade de funções de mapeamento muito complexas, ou que não possam ser especificadas genericamente, podem incluir-se definições ou tabelas de mapeamento no dicionário de dados de maneira a não sobrecarregar as transições nos diagramas HLS dos processos. A tabela 4.8 mostra um exemplo hipotético de uma função de mapeamento de estados para a variável de referência i, descrita na forma de tabela. Os estados variam em função desta variável numérica, que assume os valores do 0 ao 9.

TABELA 4.8 - Exemplo hipotético de uma tabela de mapeamentos para a variável numérica i.

i	Resultado
0	1
1	5
2	5
3	5
4	7
5	2
6	4
7	9
8	7
9	3

No caso do problema da IFIP, não existe nenhum estado com estrutura composta exclusiva[16], portanto não foi preciso definir funções de mapeamento em nenhuma transição de nenhum processo do sistema.

4.3.1.8 As Condições

As condições nos HLS são usadas principalmente para manter uma consistência interna nos processos e, às vezes, externa entre processos, garantindo assim uma consistência para o sistema como um todo.

De acordo com o critério de consistência, as condições podem ser divididas naquelas de consistência intraprocesso (interna ao processo) e naquelas de consistência interprocessos (externa entre processos). Em geral, estas consistências manifestam-se por meio de verificações da forma in Estado e into Estado.

O propósito específico de uma condição de consistência intraprocesso é o de garantir o sincronismo entre transições de subestados concorrentes dentro de um processo com estrutura simples concorrente. Todas estas transições envolvidas devem possuir eventos componentes do estímulo do processo, para que o sincronismo esteja definido.

[16] Na realidade, é difícil encontrar este tipo de situações nos domínios de sistemas de informação.

figura, o evento das duas transições que devem ser sincronizadas é o mesmo, o que explica a simplicidade do caso. Em cada subestado é verificado se o estado de origem para a transição no outro subestado concorrente é o apropriado. Para que uma sessão possa ter um moderador alocado, é preciso que o mesmo esteja disponível. Em contrapartida, para que um moderador possa ser alocado a uma sessão, é preciso que a mesma tenha sido criada. Esta mútua verificação garante a consistência intraprocesso desejada.

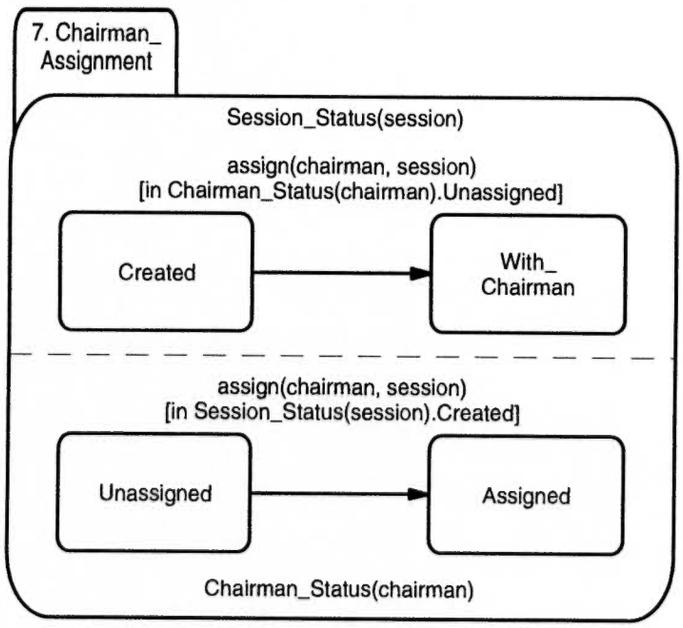


FIGURA 4.23 - Exemplo específico de consistência intraprocesso simples em um diagrama HLS incompleto.

Uma situação mais complexa de consistência intraprocesso acontece quando os eventos que sincronizam as transições são diferentes entre si, apesar de serem todos componentes do estímulo do processo. A figura 4.24 mostra este tipo de consistência intraprocesso. Nesta figura, os eventos review(paper, {report}) e { review(paper, report_data) } são componentes do estímulo review(paper, {report_data}). Para que um artigo possa estar revisado, é preciso que o conjunto de relatórios de avaliação do mesmo seja, concorrentemente, criado. Em contrapartida, para poder criar os relatórios de avaliação de um artigo, é necessário que o artigo tenha sido previamente distribuído.

Uma das condições usa, então, uma variável tipo lista para verificar o conjunto de estados destinos para as restantes transições sincronizadas.

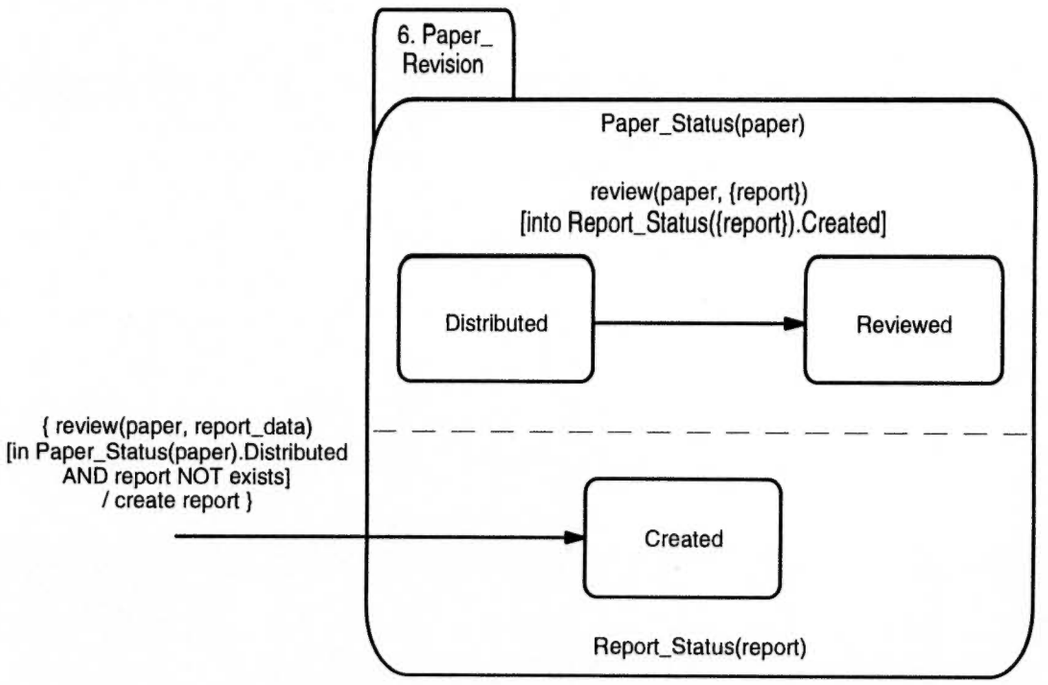


FIGURA 4.24 - Exemplo específico de consistência intraprocesso mais complexa em um diagrama HLS incompleto.

As condições de consistência interprocessos têm como propósito garantir algum tipo de associação entre dados representados em diferentes processos. Este tipo de consistência depende exclusivamente da semântica dos processos envolvidos.

Em termos gerais, a consistência interprocessos é mantida em um processo em relação a outro(s), graças a expressões nas condições de um processo na forma in Estado, onde o Estado pertence ao(s) outro(s) processo(s). Neste(s) outro(s) processo(s), pode(m) aparecer (mas não necessariamente sempre é assim) condições de consistência simétrica que verificam algum estado no primeiro processo. Contudo, também é possível encontrar situações em que este tipo de consistência pode ser feita através de expressões, usando dados componentes ou parâmetros.

Um exemplo de consistência interprocessos é mostrado na figura 4.25. Os três processos mostrados apresentam condições que verificam:

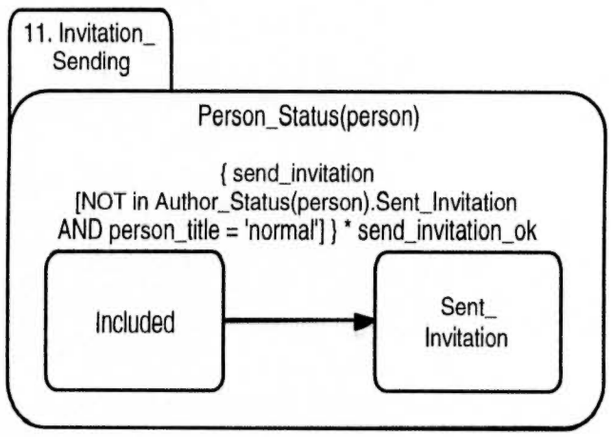
1. Se não foi enviado um convite de autor a um convidado normal, através da condição NOT in Author_Status(person).Sent_Invitation do processo Invitation_Sending.
2. Se não foi enviado um convite de autor a um convidado prioritário, através da condição NOT in Author_Status(person).Sent_Invitation do processo Priority_Invitation_Sending.
3. Se não foi enviado um convite de convidado normal a um convidado prioritário, e vice versa, através das expressões da forma person_title = '...' nas condições dos processos Invitation_Sending e Priority_Invitation_Sending.
4. Se não foi enviado um convite de convidado normal ou de convidado prioritário a um autor, através da expressão NOT in Person_Status(author).Sent_Invitation AND NOT in Person_Status(author).Sent_Priority_Invitation da condição do processo Author_Invitation_Sending.

Outros tipos de expressões usam parâmetros ou dados componentes de parâmetros para verificar condições específicas para os processos. Algumas expressões dos processos da figura 4.25 usam o dado componente person_title em suas condições, porém como uma verificação de consistência interprocessos. A figura 4.26 mostra a combinação de condições específicas (revisions < maximum e revisions ≥ maximum) de um processo com condições de consistência intraprocesso. A situação modelada é a da distribuição de artigos entre os avaliadores, para os quais foi definido um número máximo de artigos para avaliar.

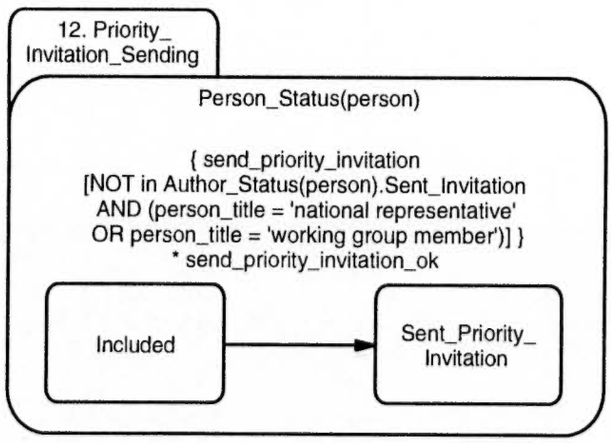
Finalmente, a figura 4.27 que completa parte da figura 4.12, mostra o uso de um parâmetro em uma condição. A situação representada é a da seleção dos artigos após a revisão, para sua futura inclusão no programa do congresso.

4.3.1.9 As Ações Não Vinculadas à Resposta

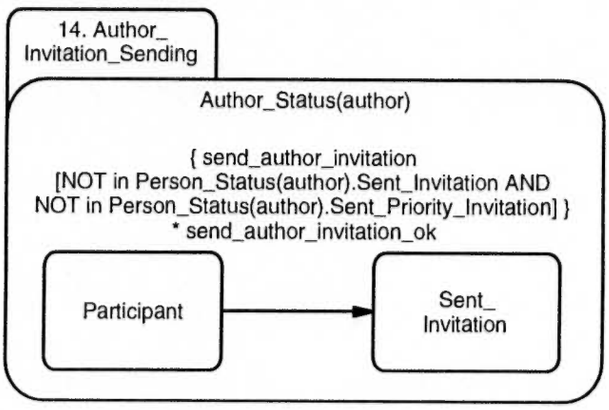
As ações não vinculadas à resposta do processo podem ser divididas em duas categorias: as ações que são operações específicas dentro do processo e as ações que tornam-se eventos internos ao processo.



(a) Requisito modelado: "*Preparing a list of people to invite to the conference. Avoiding sending duplicate invitations to any individual.* " [OLL 82]



(b) Requisito modelado: "*Issuing priority invitations to National Representatives, Working Group members and members of associated working groups. Avoiding sending duplicate invitations to any individual.* " [OLL 82]



(c) Requisito modelado: "*Ensuring all authors of each selected/rejected papers receive an invitation.* " [OLL 82]

FIGURA 4.25 - Exemplos específicos de condições de consistência interprocessos em diagramas HLS.

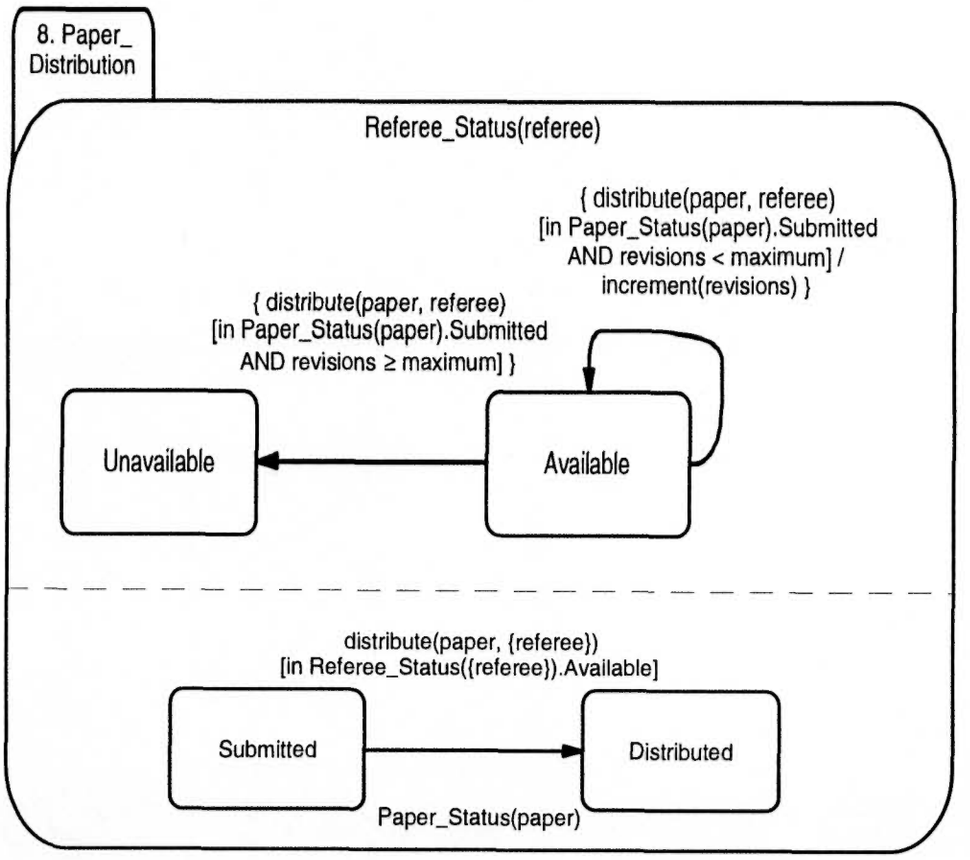
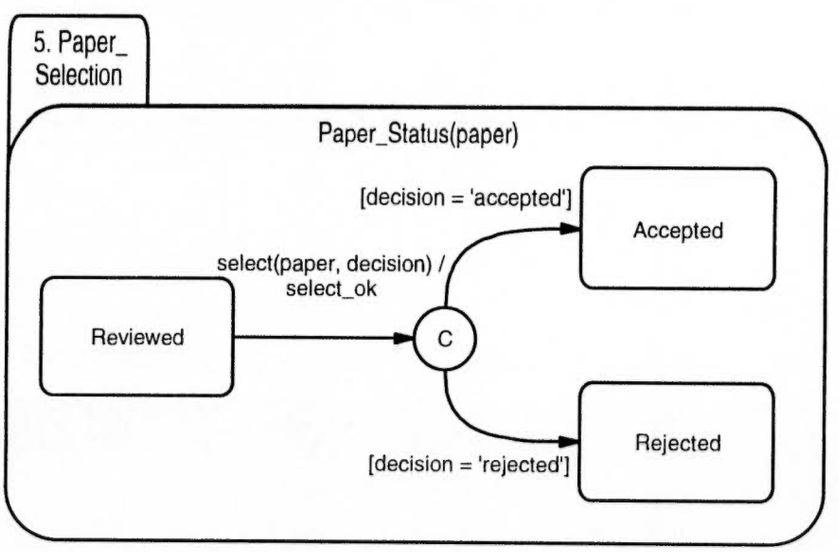


FIGURA 4.26 - Exemplo específico de expressões específicas e expressões de consistência intraprocesso em um diagrama HLS incompleto.

Algumas ações servem apenas para realização de cálculos, modificações de valores de variáveis, inicialização de estados e abandono de estados. Estas ações não são compostas em respostas e são internas ao processo em que são executadas. Exemplos deste tipo de ação podem ser vistos nos diagramas HLS dos processos da figura 4.10, ambos os processos apresentam a ação include person para inicializar estados. A figura 4.26 também apresenta uma ação deste tipo. Neste caso, é realizado um cálculo com a ação increment(revisions).

Outras ações têm como único propósito o de provocar o disparo de uma ou mais transições por meio da geração de um evento. Neste caso, tanto a ação, como o evento são equivalentes em termos de notação e são diferenciados apenas por pertencerem a transições distintas. Este tipo de ações e eventos ocorre mais freqüentemente em processos de consulta, nos quais são encadeadas várias ações e eventos para relacionar os dados ao interior do processo. No problema da IFIP, não existem processos que usem

este tipo de ação em suas transições. Um exemplo hipotético para esta situação é mostrado na figura 4.35, em que, por exemplo, as ações get_paper's_author({paper}) e get_authors({author}) tornam-se eventos para as outras transições. A suposta consulta modelada busca saber quais são os autores dos artigos programados em uma sessão dada.



Requisito modelado: "*...selecting the papers for inclusion in the program.*" [OLL 82].

FIGURA 4.27 - Exemplo específico de uma condição que usa um parâmetro em um diagrama HLS.

4.3.1.10 A Relação entre as Ações e a Resposta

Conforme foi indicado anteriormente, uma resposta é uma comunicação externa que o sistema gera para um receptor. Esta resposta é o resultado do tratamento realizado por um processo do sistema. Como o processo é um componente do sistema, as ações realizadas naquele devem configurar a resposta gerada pelo processo dentro do sistema. Esta relação é de composição, isto é, na perspectiva do mais detalhado para o mais abstrato: *as ações do processo compõem a resposta do mesmo*; ou na perspectiva inversa: *a resposta é uma composição de ações*. São estas ações as realizadas nas transições ao interior do processo.

É importante destacar que nem todas as ações do processo compõem a resposta do mesmo. As ações descritas na seção anterior constituem exceções por servirem a necessidades internas do tratamento do processo.

Da mesma forma que a relação entre estímulo e eventos, a diferença entre a resposta e as ações é que a resposta é *externa* e relaciona o sistema com os receptores, e as ações são *internas* aos processos. A partir de outro ponto de vista, as respostas e as ações pertencem a níveis de abstração diferentes: a resposta é um elemento do modelo da visão global do sistema, e as ações são elementos do modelo de processos.

Esta também é uma relação de mão dupla. Ao compor a resposta a partir das ações, pode ser necessário redefinir a resposta.

Também não existe nenhuma diferença entre a resposta e a ação em termos de notação. Os nomes de ambas são conformados da mesma maneira e ambas podem apresentar parâmetros. Apenas o nome da resposta é anotado sobre uma seta de linha descontínua, e o da ação, sobre uma seta de linha contínua (uma transição).

A composição da resposta a partir de ações, junto com a decomposição do estímulo em eventos, são os elementos centrais na relação entre os modelos da visão global e de processos.

Utilizando a mesma convenção de notação existente entre o estímulo e seus eventos componentes, a indicação de que uma resposta é uma composição de um determinado conjunto de ações é feita conservando o mesmo nome das ações na resposta, podendo variar apenas os parâmetros. Como esta regra também é válida para os estímulos e eventos, na grande maioria dos casos tanto o estímulo e os eventos, quanto as ações e a resposta, terão todos o mesmo nome.

A situação de composição mais simples possível ocorre quando a resposta é exatamente a ação, isto é, quando a ação e a resposta são equivalentes (e portanto possuem também o mesmo nome). Mais especificamente, podem ser identificados os seguintes casos:

- Quando o subestado de um processo com estrutura composta concorrente assume uma estrutura simples única, e o evento não se encontra "repetido". Como no problema da IFIP não existe uma situação deste tipo, os exemplos hipotéticos da

figura 4.14 mostram uma equivalência entre a ação de conformidade simples include_ok e a resposta do processo.

- Quando o subestado do processo (de estrutura composta concorrente) assume uma estrutura simples exclusiva. A figura 4.27 mostra um exemplo deste caso.
- Quando a ação está associada a um conjunto de transições concorrentes. A figura 4.25 apresenta três diagramas HLS com este tipo de ação.

As formas de composição de ações podem ser categorizadas em:

- Duas ou mais ações em transições de subestados concorrentes. Nesta forma de composição, é realizada uma ação para cada subestado em que um dos parâmetros do estímulo é tratado. A forma desta composição é mostrada na figura 4.28. Para cada subestado existe uma ação associada ao parâmetro que está sendo tratado; os restantes parâmetros são substituídos pelo parâmetro nulo (null). Estes parâmetros não aparecem na resposta porque são perdidos por sobreposição de outros parâmetros não nulos na mesma posição. Os parâmetros de cada ação podem ser dados relativos à saída do processo ou podem expressar apenas conformidade com a constante 'ok'. A figura 4.29 mostra o diagrama HLS completo da figura 4.23, em que se pode observar ambas as ações em subestados concorrentes configurando a resposta, como é mostrada no esquema da figura 4.30.
- Ações em cada subestado de uma estrutura composta concorrente. A composição ocorre de tal maneira que cada uma das ações contribui com um elemento na lista da resposta. Exemplos específicos desta forma de decomposição são os da figura 4.10. O esquema de composição para ambos os casos é mostrado na figura 4.31.

As formas de composição acima descritas também podem ser combinadas. Este é o caso mostrado nos diagramas da figura 4.9. A figura 4.32 apresenta o esquema de composição das ações deste diagrama.

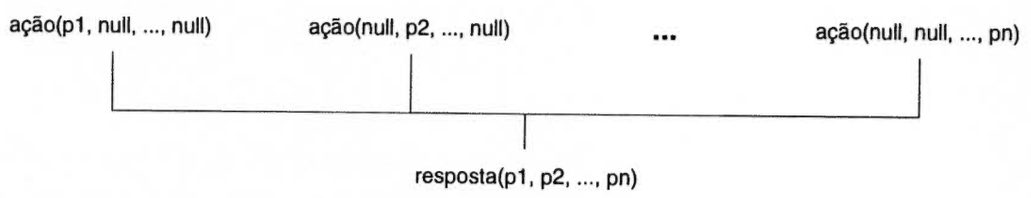
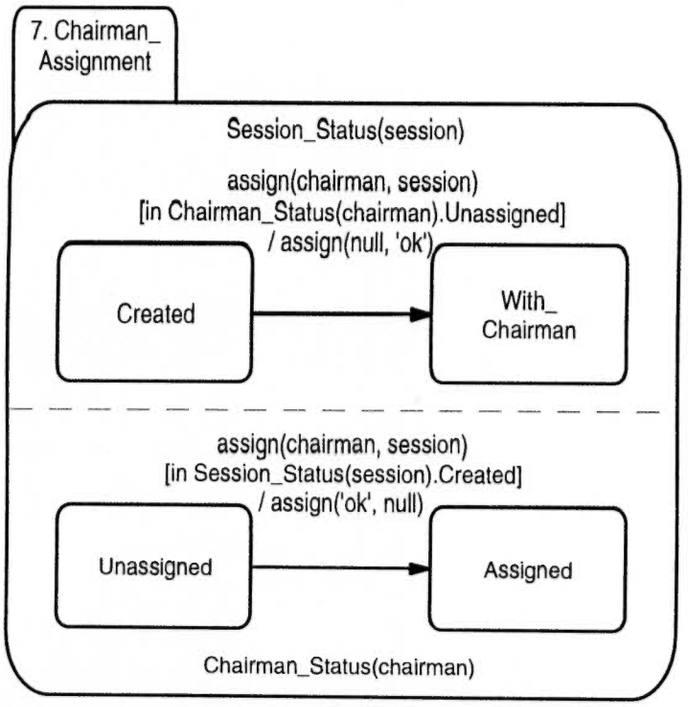


FIGURA 4.28 - Esquema de composição de um conjunto de ações em uma resposta.



Requisito modelado: "*...selecting chairman for each session.*" [OLL 82].

FIGURA 4.29 - Exemplo específico de uma composição de ações em um diagrama HLS.

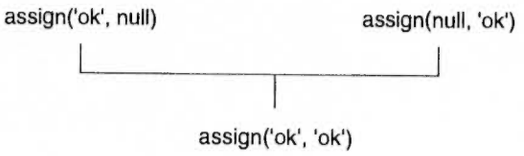


FIGURA 4.30 - Esquema de composição específica de ações em uma resposta para a figura 4.29.

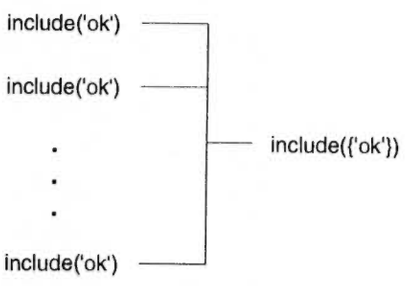


FIGURA 4.31 - Esquema de composição específica de ações da figura 4.10.

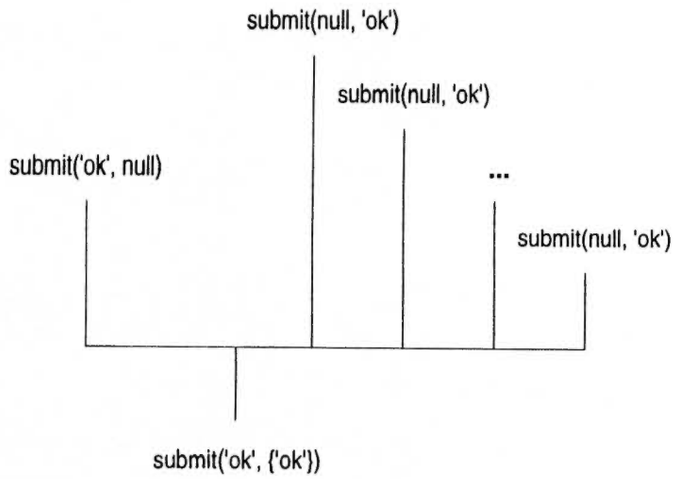


FIGURA 4.32 - Esquema de composição específica combinada de ações da figura 4.9(b).

Ainda é possível outra combinação: dois ou mais conjuntos de ações concorrentes. Este caso é exemplificado na figura 4.15. O esquema desta composição aparece na figura 4.33. Em todos os exemplos anteriores, tanto as ações, como a resposta eram de conformidade composta. O exemplo da figura 4.33 é o único, dentro do problema da IFIP, que possui variáveis como parâmetros nas ações e na resposta.

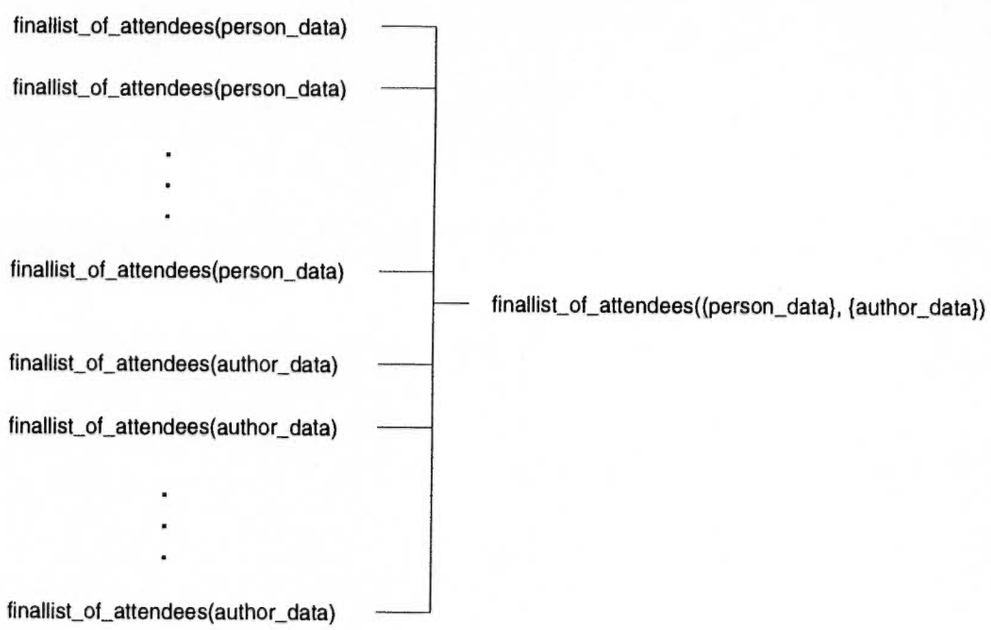


FIGURA 4.33 - Esquema de composição específica de dois conjuntos de ações concorrentes em uma resposta.

4.3.2 Outras Considerações

Além dos elementos descritos nas seções anteriores, que permitem construir o modelo de processos, existem algumas considerações globais que merecem destaque. Estas considerações são a respeito dos relacionamentos entre dados componentes em um processo e da correlação que é estabelecida entre estímulo, eventos, ações e resposta em um processo.

4.3.2.1 Os Relacionamentos entre Dados Componentes

No caso que existam dados componentes associados a uma variável de referência, isto é, quando o estímulo, que implica um ou mais eventos de inicialização de estado, apresenta um ou mais parâmetros compostos cujos dados devem ser incluídos na memória do processo e como apenas a variável de referência é "visível", se subentende que todos os outros dados que fluem junto com o identificador único, que é a mesma variável de referência, estão associados a ela. Este relacionamento não é representado explicitamente nos diagramas HLS e sim no dicionário de dados, onde o dado composto é definido.

Este é o caso mostrado nos exemplos da figura 4.10, na qual o parâmetro composto person_data do evento include(person_data) inclui o identificador único person mais os dados componentes name, institution e person_title. Todos estes dados mantêm-se associados à variável de referência person no momento de serem incluídos no processo.

Este relacionamento também pode ser consultado, isto é, quando a resposta apresenta parâmetros compostos ou não, que referenciam os dados componentes associados implicitamente dentro do processo. Neste caso, é possível gerar a resposta graças ao fato de estar o relacionamento definido no dicionário de dados.

Por exemplo, é possível consultar os mesmos dados incluídos na figura 4.10 através do processo com o estímulo de consulta mostrado no subestado Person_Status(person) da figura 4.15. Neste caso, o parâmetro composto person_data da

ação finallist_of_attendees(person_data) apresenta todos os dados componentes associados à variável de referência person que foram previamente incluídos.

4.3.2.2 As Correlações entre o Estímulo, o(s) Evento(s), a(s) Ação(ões) e a Resposta

Todo o processo de transformação de um estímulo em uma resposta, dentro do sistema, pode ser entendido como um fluxo de entrada que inicia com o estímulo. Este, ao entrar no processo, explode em um conjunto de eventos que se distribuem no interior do processo em diferentes transições, cada uma das quais pode realizar uma ação. O conjunto destas ações é condensado em uma resposta que é o fluxo de saída do processo e do sistema. A figura 4.34 sintetiza as diferenças de abstração e as relações de decomposição e composição que ocorrem neste fluir de informação pelo sistema.

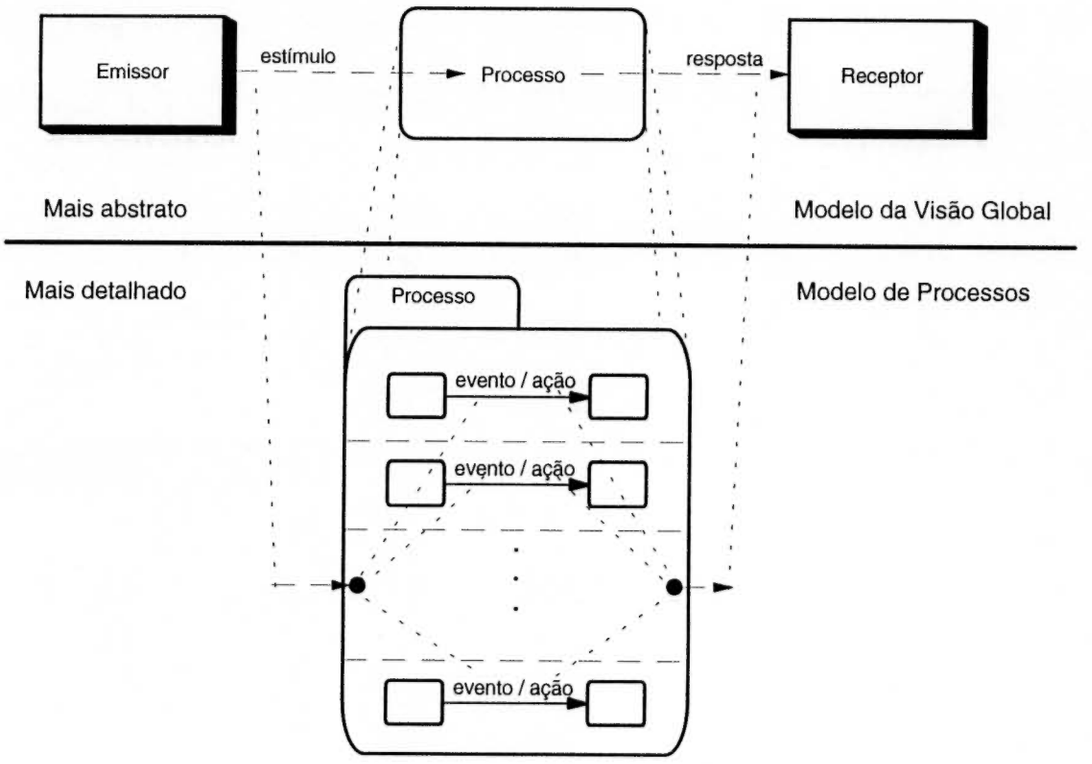


FIGURA 4.34 - Síntese esquemática das relações e diferenças entre estímulo, eventos, ações e resposta.

A transformação descrita acima genericamente, pode assumir diversas formas na prática. A tabela 4.9 resume as principais correlações entre o estímulo (sua forma genérica e a decomposição em eventos), a estrutura que assume o superestado do processo, as ações associadas e a resposta. As entradas da tabela podem ser facilmente combinadas e estendidas para um maior número de parâmetros.

TABELA 4.9 - Algumas correlações entre estímulo, eventos, estados, ações (de conformidade) e resposta (de conformidade) concorrentes nos HLS.

	ESTÍMULO		PROCESSO	RESPOSTA	
Nº	Forma Genérica	Decomposição em Eventos*	Estrutura	Composição de Ações	Forma Genérica
1	estímulo	1 ou mais eventos	1 ou mais estados	1 ou mais ações	resposta({p}) ou resposta_ok
2	estímulo(p)	evento(p)	simples única	ação_ok	resposta_ok
3	estímulo(p1, p2)	evento(p1, null), evento(null, p2)	simples concorrente	ação(p1, null), ação(null, p2)	resposta(p1, p2)
4	estímulo({p})	{ evento(p) }	composta concorrente	1 ou mais ações	resposta({p}) ou resposta_ok
5	estímulo(p1, {p2})	evento(p1, null), { evento(null, p2) }	simples concorrente com um subestado composto concorrente	ação(p1, null), { ação(null, p2) }	resposta(p1, {p2})
6	estímulo(p1, {p2})	evento(p1, null), { evento(null, p2) }	simples concorrente com um subestado composto concorrente	2 ações	resposta(p1, p2)
7	estímulo(p1, {p2})	evento(p1, null), { evento(null, p2) }	simples concorrente com um subestado composto concorrente	ação_ok	resposta_ok
8	estímulo({p1}, {p2})	{ evento(p1, null) }, { evento(null, p2) }	simples concorrente com 2 subestados compostos concorrentes	{ ação(p1, null) }, { ação(null, p2) }	resposta({p1}, {p2})
9	estímulo({p1}, {p2})	{ evento(p1, null) }, { evento(null, p2) }	simples concorrente com 2 subestados compostos concorrentes	1 ou mais ações e 1 ação	resposta({p1}, p2)
10	estímulo({p1}, {p2})	{ evento(p1, null) }, { evento(null, p2) }	simples concorrente com 2 subestados compostos concorrentes	1 ação e 1 ou mais ações	resposta(p1, {p2})
11	estímulo({p1}, {p2})	{ evento(p1, null) }, { evento(null, p2) }	simples concorrente com 2 subestados compostos concorrentes	2 ações	resposta(p1, p2)
12	estímulo({p1}, {p2})	{ evento(p1, null) }, { evento(null, p2) }	simples concorrente com 2 subestados compostos concorrentes	ação_ok	resposta_ok

p, p1, p2 são parâmetros escalares

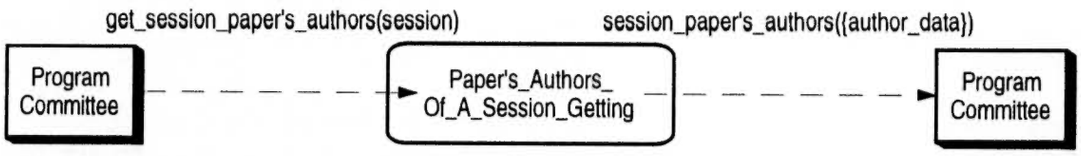
* Supõe-se que as variáveis nos parâmetros dos eventos não são referenciadas nas funções de mapeamento, nas condições e nem nas ações das transições.

4.3.3 A Modelagem de Consultas com HLS

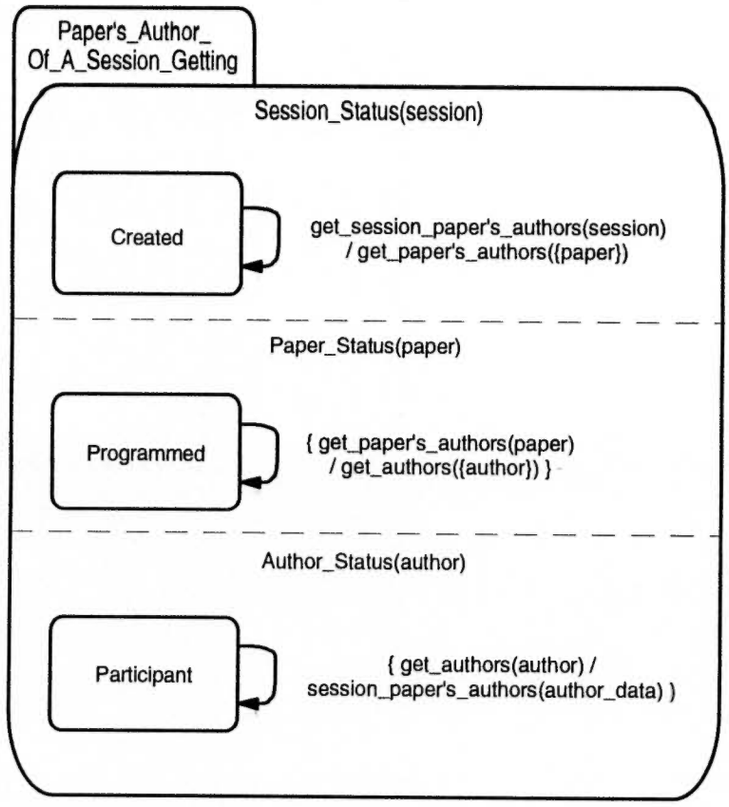
Um caso particular de processo é aquele que responde a um estímulo de consulta. As consultas em relação a dados armazenados são muito comuns em sistemas de informação. As consultas, pela sua própria natureza, não modificam dados, apenas produzem respostas mais complexas, que são geradas geralmente por auto-transições.

O problema da IFIP usado ao longo deste capítulo apresenta um único requisito de consulta sobre o sistema. Esta consulta é mostrada na figura 4.15. Nela é modelada a necessidade de gerar uma lista final com todos os participantes do congresso. Tanto participantes autores como não autores figuram nesta lista. O evento generate_finallist_of_attendees não possui parâmetros, e sua resposta é composta por ações associadas a auto-transições nos estados de Accepted_Invitation, tanto para pessoas, quanto para autores. Os dados de cada pessoa ou autor são acrescentados concorrentemente na lista, resultando assim em dois conjuntos de dados gerados como resposta.

Não existem transições entre estados diferentes, e o sistema não se vê afetado pela recepção do estímulo da figura 4.15 (no sentido de modificar seu comportamento futuro após ter sido estimulado por uma consulta). Este é o caso típico de modelagem de consultas. Consultas envolvendo mais variáveis apenas encadeiam eventos em estados com auto-transições. Por exemplo, uma consulta que não faz parte da definição do problema da IFIP poderia ser: quais os autores dos artigos programados em uma sessão dada? Para responder a esta consulta (vide figura 4.35), os eventos são encadeados da seguinte maneira: o estímulo get_session_paper's_authors(session) seria ele próprio o primeiro evento da cadeia, a ação get_paper's_author({paper}) seria o evento para cada auto-transição do estado Programmed do artigo (conjunto de estados concorrentes que representa os artigos da sessão) com a ação get_authors({author}), que por sua vez seria o evento para cada auto-transição do estado Participant do autor (também um conjunto de estados concorrentes), que, finalmente, teria a ação session_paper's_author(author_data). A composição do conjunto destas últimas ações concorrentes seria a resposta session_paper's_author({author_data}) desejada.



(a)



(b)

FIGURA 4.35 - Exemplos específicos hipotéticos de: (a) a visão global da tupla; (b) diagrama HLS do processo de consulta com várias variáveis.

4.4 A Estrutura do Modelo Inicial do Problema da IFIP

A representação completa do sistema obtida após a modelagem inicial consiste em:

1. **Modelo da visão global do sistema**:
   - **Diagrama de contexto do sistema**: Que mostra o sistema sob modelagem, o supra-sistema do qual faz parte e os agentes externos a ele relacionados.
   - **Diagrama da visão global**: Que mostra o sistema como uma lista de processos concorrentes interagindo com agentes externos através de estímulos e respostas.

2. **Modelo de processos**: Conjunto de diagramas HLS que descrevem detalhadamente o comportamento de cada processo como uma transformação dos estímulos em respostas.

3. **Dicionário de dados**: Define a composição e domínios para todos os dados que aparecem nos diagramas da visão global e dos processos.

A figura 4.36 mostra a estrutura geral do modelo para o problema da IFIP desenvolvido ao longo deste capítulo. No anexo "Modelo Completo do Problema da IFIP" podem ser encontrados todos os diagramas do problema da IFIP.

4.5 Resumo

O presente capítulo descreveu o processo de modelagem inicial do sistema.

Este processo inicia com a construção da visão global do sistema. Para isto, é necessário identificar o supra-sistema do qual o sistema sob modelagem faz parte e os sistemas relacionados ou agentes externos. A seguir, se define de uma lista de estímulos aos quais o sistema sob modelagem está submetido em sua relação com os agentes externos. Cada estímulo desta lista é processado por um processo específico que gera uma resposta. Implícito na definição dos parâmetros dos estímulos existe um processo de

abstração por classificação que determina os tipos de dados a serem processados pelo sistema. A decomposição da visão global é realizada de tal forma que para cada processo mostrado no modelo da visão global, é construído um diagrama HLS de processo, que descreve detalhadamente seu comportamento. Especial ênfase foi dada às estruturas gerais que o processo pode assumir, às variáveis de referência nos superestados, às relações de decomposição entre estímulo e eventos e às relações de composição entre ações e resposta. Finalmente, foi tratado o caso particular das consultas nos sistemas de informação.

Todo o processo de modelagem foi exemplificado com o problema de preparação de congressos da IFIP.

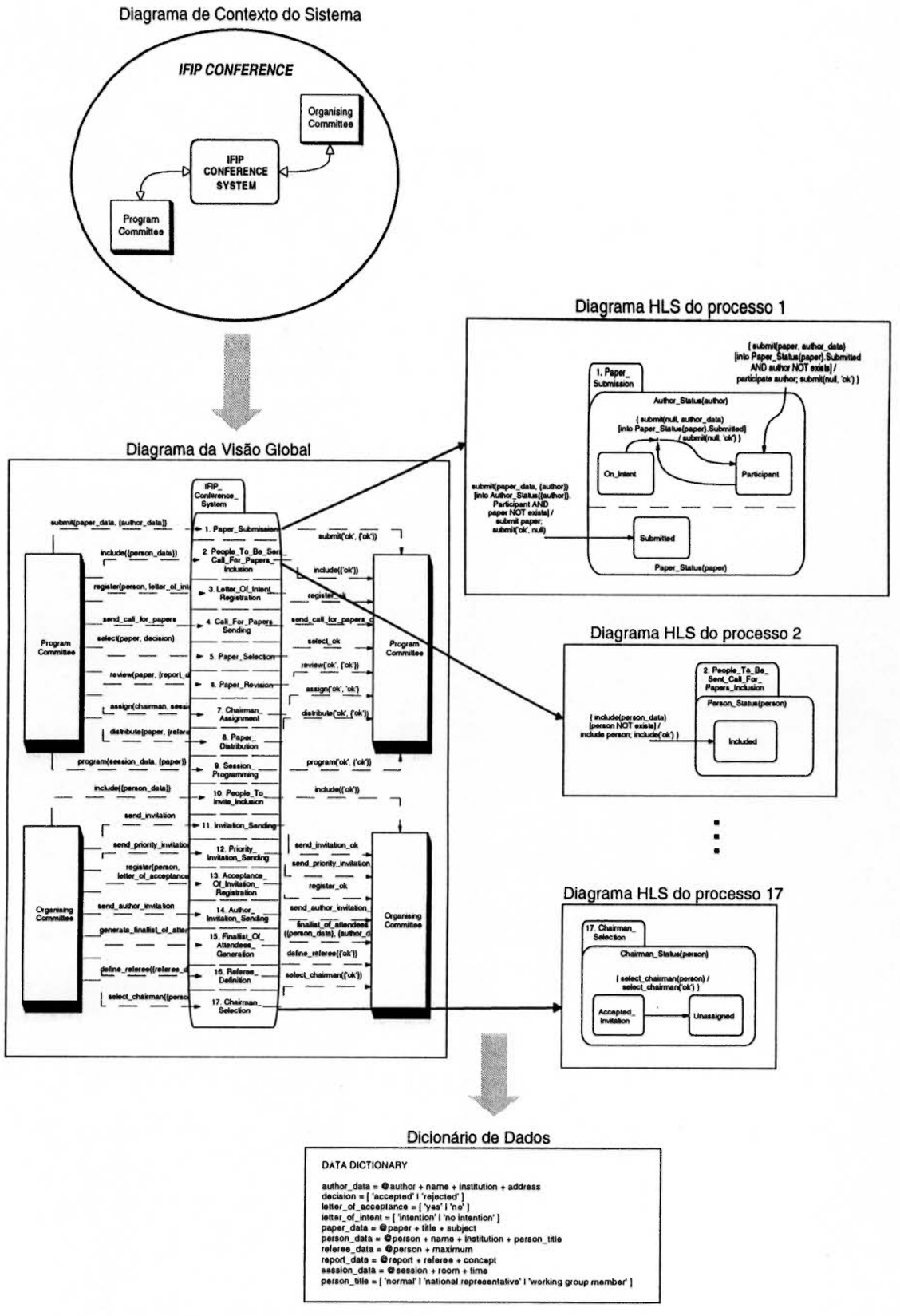


FIGURA 4.36 - Estrutura geral do modelo para o problema da IFIP.

5 CONSTRUÇÃO DE CICLOS DE VIDA

Uma vez concluída a modelagem do sistema na perspectiva dinâmica, que inclui o diagrama de contexto do sistema, o diagrama da visão global, o dicionário de dados e os diagramas HLS dos processos do sistema, segue-se o passo denominado construção de ciclos de vida.

A construção de ciclos de vida é o processo de *encadeamento* de pós-estados e pré-estados de diferentes processos, mas com dados (variáveis de referência) em comum para configurar diagramas de ciclo de vida dos dados do sistema. Em outras palavras, processos que tratam dos mesmos tipos de dados são integrados em modelos HLS maiores. O resultado deste processo é um conjunto de diagramas HLS, em que cada diagrama mostra os distintos estados atingidos pelos dados definidos através da variável de referência do mesmo. O conjunto destes diagramas HLS de ciclo de vida é denominado modelo de ciclos de vida do sistema.

Por ser este passo um processo de mapeamento, é relativamente mais simples do que a atividade anterior de modelagem, porém existem certas verificações de consistência de nomes de superestados, subestados e variáveis de referência que devem ser consideradas pelo modelador e que serão descritas em detalhe neste capítulo.

A construção de ciclos de vida está dividida em três atividades:

- **Identificação de estados e variáveis comuns**: Que agrupa diagramas HLS (ou partes de diagramas) de diferentes processos que usem a mesma variável de referência.
- **Conexão de pós e pré-estados de diferentes processos**: Que integra os diagramas HLS (ou partes de diagramas) em diagramas HLS preliminares de ciclo de vida de cada variável de referência, através da representação única de pós-estados e pré-estados pertencentes a diferentes processos.
- **Verificação de consistência no uso de nomes de estados e variáveis**: Verifica e corrige possíveis inconsistências no uso de nomes de estados e variáveis em cada diagrama HLS preliminar de ciclo de vida para gerar os diagramas HLS definitivos de ciclo de vida.

O capítulo está organizado como segue: a seção 5.1 descreve a identificação de estados e variáveis comuns e os problemas de integração que podem ser encontrados; a seção 5.2 mostra a forma de conectar diferentes processos (ou partes de processos) que tratem dos mesmos dados, de acordo com os pós e pré-estados que apresentem, configurando os diagramas HLS preliminares de ciclo de vida; finalmente, a seção 5.3 apresenta a verificação de consistência que deve ser realizada nos nomes de estados e variáiveis destes diagramas para obter os diagramas HLS definitivos de ciclo de vida.

5.1 Identificação de Superestados, Subestados e Variáveis de Referência Comuns

Ao modelar previamente o sistema com a técnica apresentada, obteve-se, no nível mais detalhado, um conjunto de diagramas HLS que representam o comportamento específico de cada processo face a um estímulo e sua resposta correspondente. Desta maneira, o sistema é indiretamente particionado em peças de comportamento que, explicitamente, não apresentam relações.

Dois tipos de relação implícita entre os processos merecem destaque: a homonímia e a sinonímia de superestados, subestados e variáveis de referência em geral. Muitos processos (superestados) e seus subestados concorrentes apresentam os mesmos nomes ou as mesmas variáveis de referência em contextos de estímulo-resposta diferentes. Também é possível encontrar superestados ou subestados que possuem nomes distintos ou usam variáveis com nomes diferentes, que se referem a um mesmo dado ou ainda haver superestados ou subestados com nomes e variáveis com mesmo nome, porém referindo-se a dados ou conceitos diferentes.

O resultado desta atividade de identificação de estados e variáveis comuns é um conjunto de grupos de superestados (diagramas HLS de processos) ou subestados (partes concorrentes de processos) que tratam dos mesmos dados expressos pelas variáveis de referência (que podem ou não apresentar algum tipo de conflito de nomes).

Este problema pode ser abordado como um problema de integração de visões (*view integration*) em banco de dados, como exposto em [BAT 86] e [BAT 92]. Neste contexto, o conflito de nomes das variáveis de referência, indicado acima, pode originar-se de duas fontes: *sinônimos* e *homônimos*.

Existe uma *similitude de conceitos* (*concept similarity*) quando dados ou conceitos com diferentes nomes possuem propriedades em comum. Os nomes envolvidos nesta similitude conceitual seriam sinônimos. Quando conceitos com o mesmo nome possuem propriedades diferentes, trata-se de uma *desigualdade de conceitos* (*concept mismatch*) expressa por meio dos homônimos.

As propriedades dos conceitos ou dados, no caso dos diagramas HLS, referem-se a:

- papéis específicos que desempenham os dados representados pelos nomes (variáveis),
- contextos temporais, isto é, ciclos de vida aos quais os dados pertencem e,
- em geral, semântica do estímulo, eventos, ações e resposta do processo.

Os quatro casos possíveis são sintetizados na tabela 5.1, na qual também indicam-se quais as ações gerais a realizar em cada caso.

TABELA 5.1 - Situações possíveis na identificação de estados e variáveis comuns.

NOMES / CONCEITOS	O MESMO	DIFERENTES
SIMILITUDE	Incluir ambos no mesmo conjunto de superestados.	Caso de sinônimos. Pode ser necessário renomear. Incluir ambos no mesmo conjunto de superestados.
DESIGUALDADE	Caso de homônimos. Pode ser necessário renomear. Incluir cada dado em conjuntos de superestados diferentes.	Incluir cada dado em conjuntos de superestados diferentes.

Todos aqueles diagramas (ou partes de diagramas) HLS de processos que se julgam tratar-se dos mesmos dados, independentemente dos superestados, subestados ou variáveis de referência possuirem ou não o mesmo nome, devem ser agrupados em conjuntos de superestados de tal forma que cada um destes conjuntos venha a configurar um só diagrama HLS de ciclo de vida. Constitui uma ajuda muito importante para esta atividade a utilização, na modelagem inicial do sistema, de nomes e variáveis significativos e relativamente padronizados. Por exemplo, o uso de nomes tais como paper_data (que corresponde a um dado composto) ou Author_Status(author) (superestado/subesta-

do e variável de referência, respectivamente) já dão uma indicação para poder agrupar em diagramas relativos às variáveis paper e author, respectivamente.

Finalmente, cada caso indicará a necessidade ou não de renomear algum dos conceitos envolvidos no conflito (sinônimos ou homônimos) para eliminar a ambigüidade. Em alguns casos, isto não é preciso porque, ao serem tratados depois, em um único diagrama HLS de ciclo de vida, podem ser introduzidas variáveis de referência sinônimas. Porém, a renomeação pode ser indispensável quando existe mais de um modelador, cada um dos quais construiu diagramas HLS de processos que, eventualmente, terão que ser integrados em um mesmo diagrama HLS de ciclo de vida.

Nas próximas seções são descritas as ações tomadas no agrupamento de diferentes processos (ou partes de processos) para o problema da IFIP.

5.1.1 Os Conceitos e Nomes Comuns

Esta é a situação mais simples: os mesmos dados ou conceitos possuem o mesmo nome. Tomando como exemplo os processos modelados no capítulo anterior, pode-se identificar o seguinte:

- O processo[46] Paper_Revision da figura 5.1, com o subestado Report_Status(report), é único, isto é, não existe outro processo e/ou subestado que trate da variável de referência report.

- Os subestados Session_Status(session) dos processos Session_Programming e Chairman_Assignment são comuns (ambos mostrados na figura 5.2). Tanto os nomes dos subestados quanto as variáveis de referência representadas são os mesmos.

- Todos os superestados e subestados Paper_Status(paper) dos seguintes processos podem ser agrupados para fazer parte do mesmo diagrama de ciclo de vida:

    1. Paper_Selection da figura 5.3,

[46] Todos os diagramas HLS de processos encontram-se no anexo "Modelo Completo do Probkema da IFIP".

2. Paper_Submission da figura 5.3,
3. Paper_Revision da figura 5.1,
4. Paper_Distribution da figura 5.3, e
5. Session_Programming da figura 5.2.

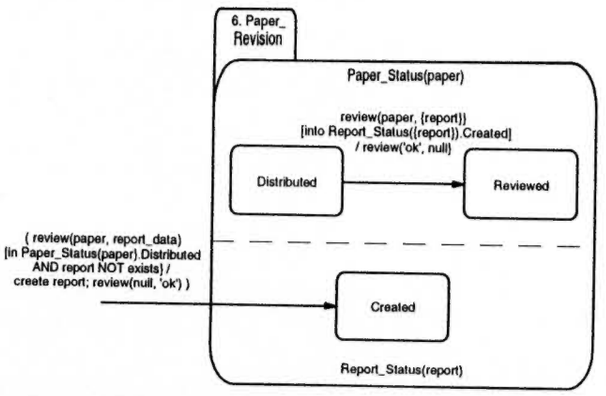


FIGURA 5.1 - Diagrama HLS do processo Paper_Revision.

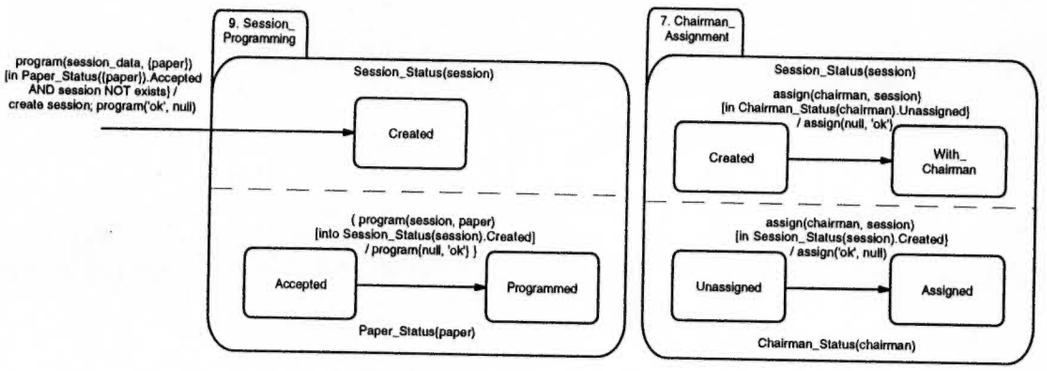


FIGURA 5.2 - Diagramas HLS dos processos Session_Programming e Chairman_Assignment.

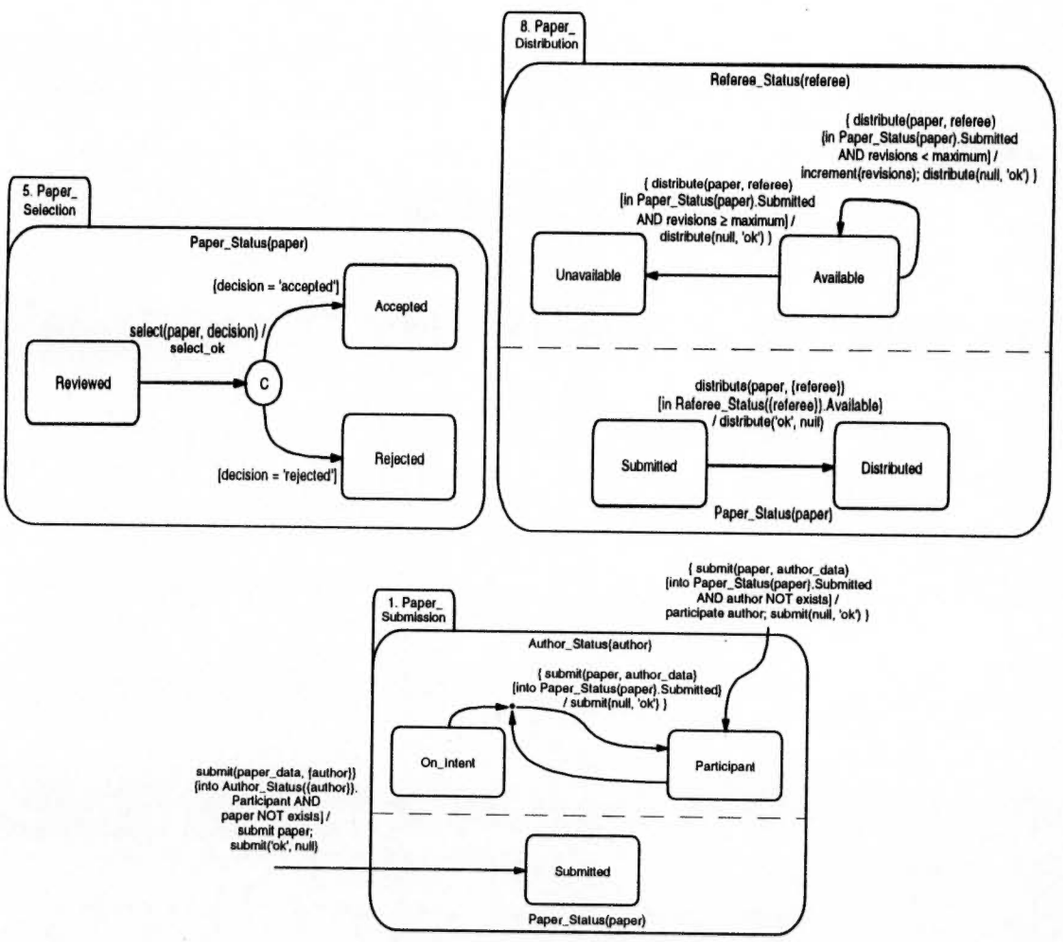


FIGURA 5.3 - Diagramas HLS dos processos Paper_Selection, Paper_Distribution e Paper_Submission.

- Todos os superestados e subestados Author_Status(author) dos seguintes processos podem ser incluídos em um mesmo conjunto:
  1. Paper_Submission da figura 5.3,
  2. Author_Invitation_Sending da figura 5.4 e
  3. Finallist_Of_Attendees_Generation da figura 5.4.

- Todos os superestados e subestados Person_Status(person) dos seguintes processos também possuem mesmo nome e mesma variável de referência:
  1. People_To_Invite_Inclusion da figura 5.5,
  2. Acceptance_Of_Invitation_Registration da figura 5.5,

3. Invitation_Sending da figura 5.5,
4. Priority_Invitation_Sending da figura 5.5 e
5. Finallist_Of_Attendees_Generation da figura 5.4.

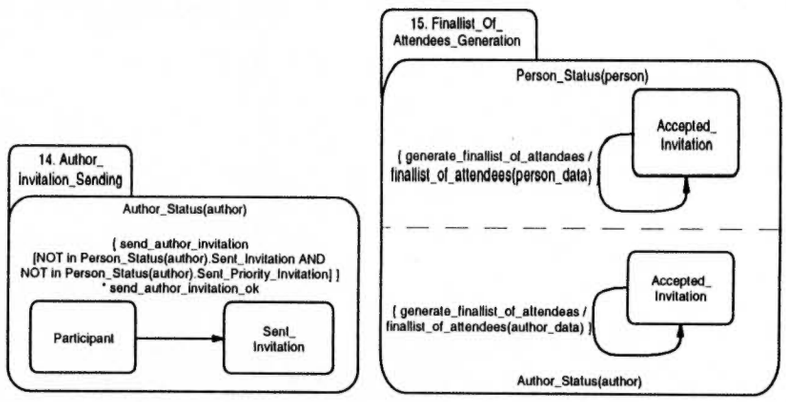


FIGURA 5.4 - Diagramas HLS dos processos Author_Invitation_Sending e Finallist_Of_Attendees_Generation.

5.1.2 Os Homônimos

Esta situação é mais delicada. No caso do problema da IFIP, inicialmente, pode-se considerar que os superestados/subestados Person_Status(person) de todos os processos mostrados na figura 5.5 poderiam se agrupar junto dos superestados com o mesmo nome de todos os processos da figura 5.6.

Contudo, ao analisar mais cuidadosamente as propriedades destes processos, observa-se que os três superestados da figura 5.6, apesar de possuírem os mesmos nomes que os da figura 5.5, em verdade, representam uma desigualdade de conceitos. Este é caso típico de homônimos. Para o caso da figura 5.5, a variável person, e seu superestado associado, refere-se a pessoas convidadas em geral, excluindo assim o caso particular dos autores dos artigos submetidos ao congresso. Esta última é justamente a situação mostrada nos processos da figura 5.6, em que a variável person faz referência apenas aos autores potenciais de artigos e não a qualquer convidado. Assim sendo, estes dois conjuntos de processos devem ser tratados em diagramas HLS de ciclo de vida separados.

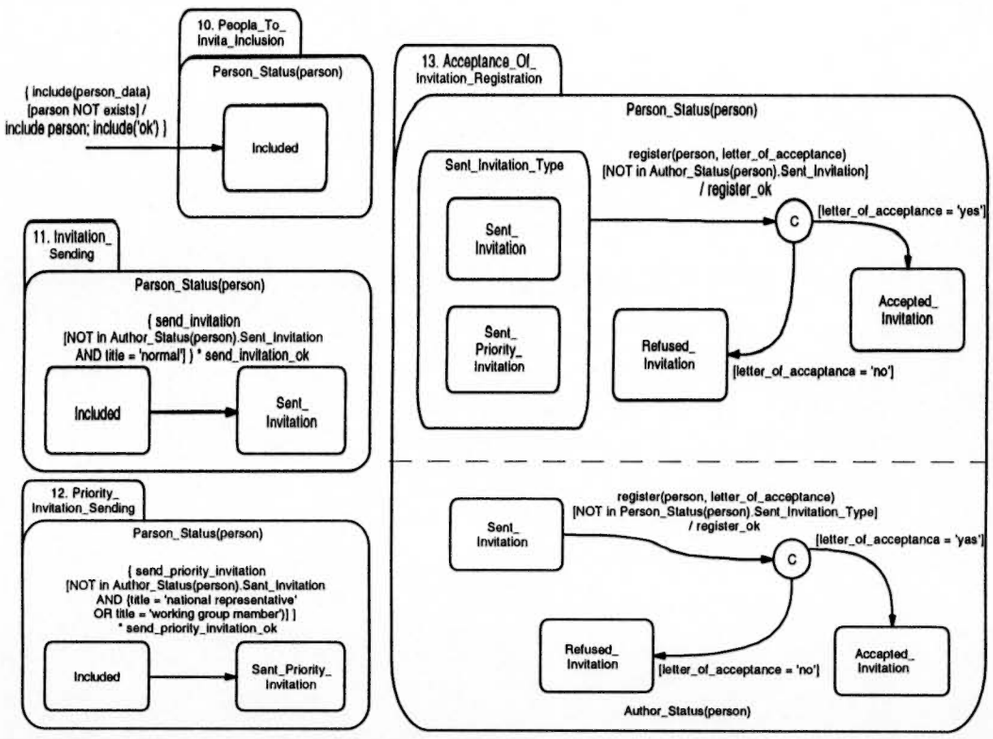


FIGURA 5.5 - Diagramas HLS dos processos People_To_Invite_Inclusion, Invitation_Sending, Priority_Invitation_Sending e Acceptance_Of_Invitation_Registration.

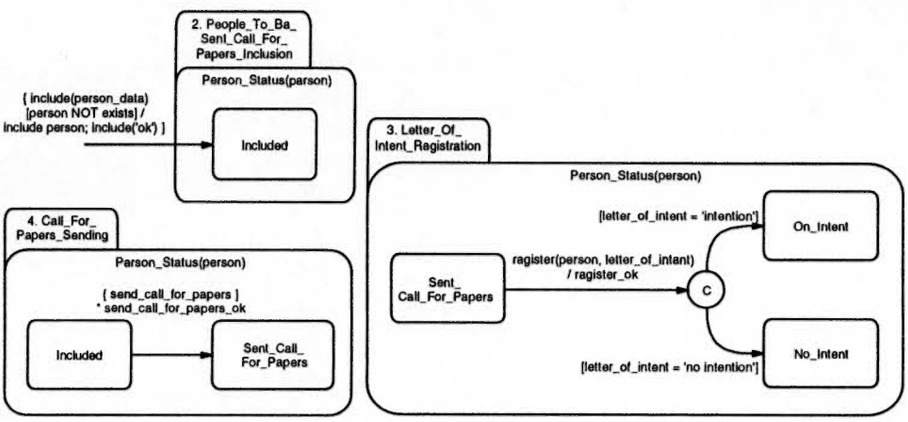


FIGURA 5.6 - Diagramas HLS dos processos People_To_Be_Sent_Call_For_Papers_Inclusion, Call_For_Papers_Sending e Letter_Of_Intent_Registration.

É possível renomear a variável de person para author, porém não é indispensável assim fazê-lo, dado que ambos os nomes são definidos nos seus respectivos contextos de estímulo-resposta. Esta ambigüidade será resolvida mais adiante, introduzindo uma variável de referência sinônima para o diagrama HLS de ciclo de vida.

5.1.3 Os Sinônimos

Esta situação também é encontrada no problema da IFIP. Conforme foi explicado na seção anterior, os processos da figura 5.6 referem-se a dados que desempenham papéis diferentes em distintos momentos: pessoas como autores potenciais, e estes mesmos autores potenciais que tornam-se autores efetivos ao submeterem artigos ao congresso. Com isto, pode-se afirmar que os estados Author_Status(author) dos processos da figura 5.4 podem ser tratados em conjunto com os superestados Person_Status(person) dos processos da figura 5.6. Em ambos os casos, apesar dos nomes dos estados e variáveis serem diferentes, estão representando os mesmos dados em papéis diferentes. Esta situação tipifica o caso dos sinônimos. Também não é necessário renomear as variáveis de referência pela mesma razão indicada no caso dos homônimos.

Também haveria que se acrescentar o caso particular do subestado Author_Status(person) do processo Acceptance_Of_Invitation_Registration da figura 5.5. Neste exemplo, o superestado possui o mesmo nome relativo ao autor, porém com a variável de referência person. Isto é assim porque neste processo representa-se a situação do registro de uma aceitação a um convite que pode corresponder a um convidado geral ou a um autor específico. Esta diferença não está estabelecida no estímulo, que usa apenas a variável genérica person, válida (em termos de domínios) para ambas as situações.

Em síntese, e em relação aos autores, tanto os superestados/subestados dos processos das figuras 5.3 e 5.5, quanto o subestado do processo da figura 5.5 podem ser tratados em conjunto em um só diagrama HLS de ciclo de vida.

Outra situação semelhante é a que ocorre em relação aos avaliadores (*referee*) e aos moderadores (*chairman*). Ambos os papéis são desempenhados por pessoas convidadas, assim, apesar de usar nomes de estados e variáveis tais como:

1. Referee_Status(person) no processo Referee_Definition da figura 5.7,
2. Referee_Status(referee) no processo Paper_Distribution da figura 5.3,
3. Chairman_Status(person) no processo Chairman_Selection da figura 5.7, e
4. Chairman_Status(chairman) no processo Chairman_Assignment da figura 5.2.

Todos eles representam pessoas (person) convidadas desempenhando papéis específicos. Isto é mais evidente ainda ao observar os processos da figura 5.7 que, justamente, definem estes papéis entre as pessoas que aceitaram o convite para o congresso.

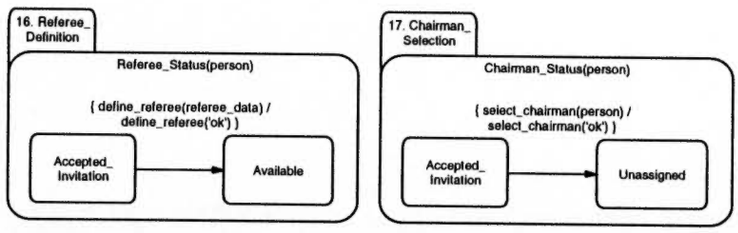


FIGURA 5.7 - Diagramas HLS dos processos Referee_Definition e Chairman_Selection.

Como resultado da categorização realizada neste passo, é obtido um ou mais conjuntos de superestados/subestados pertencentes a diferentes processos. Cada conjunto refere-se aos mesmos dados representados na forma de transições de estados que precisam ser ordenadas adequadamente em ciclos de vida. Este ordenamento é a seguinte atividade a ser descrita.

5.2 A Conexão de Pós-Estados e Pré-Estados

Uma vez concluída a categorização dos superestados/subestados de acordo com os dados representados, é necessário agrupar e ordenar todos estes superestados/subestados para poder configurar toda a *história* de cada dado tratado pelo sistema. Esta história é denominada ciclo de vida, isto é, um conjunto de estágios (estados) sucessivos, e não necessariamente seqüenciais, pelos quais os dados referidos por aqueles superestados/subestados podem passar em algum instante do tempo.

Os superestados/subestados que pertencem a uma mesma categoria de dados devem ser examinados em conjunto buscando:

- Estados inicializados que correspondem a estados iniciais do ciclo de vida a ser construído.
- Pós-estados em alguns superestados/subestados que podem ter o mesmo nome de pré-estados em outros superestados/subestados, cuidando que representem efetivamente o mesmo estado, senão se estaria face a uma nova situação de homônimos.
- Pós-estados e pré-estados, em diferentes superestados/subestados, que podem ser equivalentes apesar de possuírem nomes distintos. Isto ocorre quando se dão nomes diferentes a um mesmo estado em superestados ou subestados diferentes. Esta situação é o caso dos sinônimos.
- Pós-estados em alguns superestados que podem ser entendidos como estados finais em relação aos dados representados.
- Estados finalizados que correspondem a estados finais do ciclo de vida a ser construído.

Uma análise de conflito de nomes entre estados (ou subestados) também pode ser necessária aqui. O procedimento a aplicar é muito semelhante ao usado no caso dos superestados, subestados e variáveis de referência. Contudo, as situações anômalas são mais fáceis de detectar ao construir os diagramas HLS de ciclos de vida. Geralmente, estas situações podem se apresentar como estados isolados e/ou alternativos ou transições sem significado entre alguns estados.

Pode ajudar na resolução destes conflitos ou, mais ainda, na identificação de possíveis pós-estados e pré-estados equivalentes a consideração dos estímulos com dados principais e dados secundários. O processo que trata um estímulo com dados secundários deve ser conectado formando uma seqüência após o processo que processa o estímulo com dados principais, dada a própria definição destes dois tipos de estímulo.

Tendo em consideração os estados caracterizados acima, deve ser configurado um diagrama HLS preliminar que conecta os estados inicializados, os pós e pré-estados equivalentes, os pós-estados finais e os estados finalizados. Estes diagramas são chamados preliminares porque ainda falta serem submetidos a uma correção, produto da

verificação de consistência de nomes de superestados, subestados e variáveis de referência. Esta verificação será descrita na próxima seção.

É importante indicar uma regra a aplicar neste diagrama HLS preliminar, na definição da estrutura do superestado que o representará. Pode ocorrer que os superestados a serem conectados sejam de estrutura simples ou composta e concorrente ou exclusiva. Independentemente de qual for a estrutura apresentada por estes superestados, *o superestado do diagrama HLS preliminar de ciclo de vida sempre possuirá a estrutura composta concorrente*. Isto é assim porque a perspectiva utilizada no diagrama HLS de ciclo de vida é mudada em relação aos processos da modelagem inicial. Os diagramas HLS dos processos construídos expressam o tratamento de um estímulo específico para gerar uma resposta também específica. Nos diagramas HLS de ciclo de vida são mostrados todos os eventos possíveis (derivados de vários estímulos) que provocam transições entre os estados e geram ações (que compõem várias respostas). Desta forma, a estrutura dos superestados destes diagramas deve ser o mais genérica possível a fim de poder representar adequadamente esta multiplicidade de eventos (estímulos) e ações (respostas), que antes encontrava-se distribuída nos superestados/subestados dos diferentes processos. A forma mais geral para o superestado dos diagramas HLS de ciclo de vida é a fornecida pelo conjunto de estados concorrentes. Isto não gera nenhum tipo de conflito com os outros tipos de estruturas dos superestados a conectar. A estrutura simples é refletida agora como uma repetição em cada elemento no conjunto de subestados concorrentes. O mesmo acontece com o conjunto de subestados exclusivos.

A única convenção adicional às dos diagramas HLS é a dos nomes dos diagramas de ciclo de vida. O superestado que representa o ciclo vida como um todo é composto pelo nome da variável de referência do mesmo, seguido da frase _Life_Cycle. Assim, por exemplo, o nome do superestado do diagrama HLS de ciclo de vida da variável paper é Paper_Life_Cycle.

Para exemplificar a identificação e conexão dos estados caracterizados e a estrutura concorrente do superestado, o caso mais simples, no problema da IFIP, é o estado inicializado denominado Created no subestado Report_Status(report) do processo Paper_Revision da figura 5.1. Este é o único processo, em todo o sistema, que trata da variável report. O diagrama HLS preliminar de ciclo de vida para o dado representado pela variável report é mostrado na figura 5.8. É acrescentado o nome Report_Life_Cycle para este novo superestado.

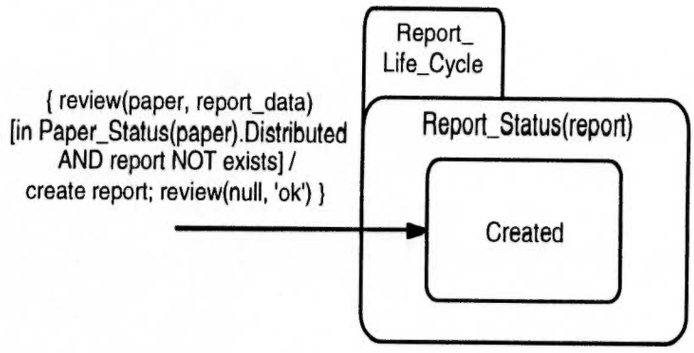


FIGURA 5.8 - Diagrama HLS preliminar para o ciclo de vida da variável report.

A figura representa então todos os estados possíveis (um no caso) em que a variável em questão pode estar, de acordo com os processos que o sistema deve desempenhar. O diagrama HLS de ciclo de vida é construído isolando os superestados/subestados, relacionados pelos mesmos dados, dos restantes superestados/subestados nos processos dos quais faz parte. No exemplo mostrado, que é a situação mais simples possível, o subestado Report_Status(report) é separado daquele relativo à variável paper do processo da figura 5.1. Deve-se notar que o estado inicializado é também o estado final neste diagrama, isto é, não ocorre mais nenhuma transição uma vez que o report é inicializado.

O caso que segue em complexidade no problema da IFIP é o conjunto de dois subestados relacionados pela mesma variável de referência session. Neste conjunto, também existe um estado inicializado denominado Created no subestado Session_Status (session) do processo Session_Programming da figura 5.2, mas também existe o caso de um pós-estado (o mesmo estado Created) coincidente com o pré-estado Created do subestado Session_Status(session) do processo Chairman_Assignment da mesma figura 5.2. Como ambos os estados são um só (apenas aparecem separados por estarem em processos diferentes), devem ser representados uma única vez, conectando as transições e formando, neste caso, uma seqüência de estados. Na figura 5.9 é mostrado este processo de integração. O pós-estado original With_Chairman torna-se, neste diagrama, um estado final para session.

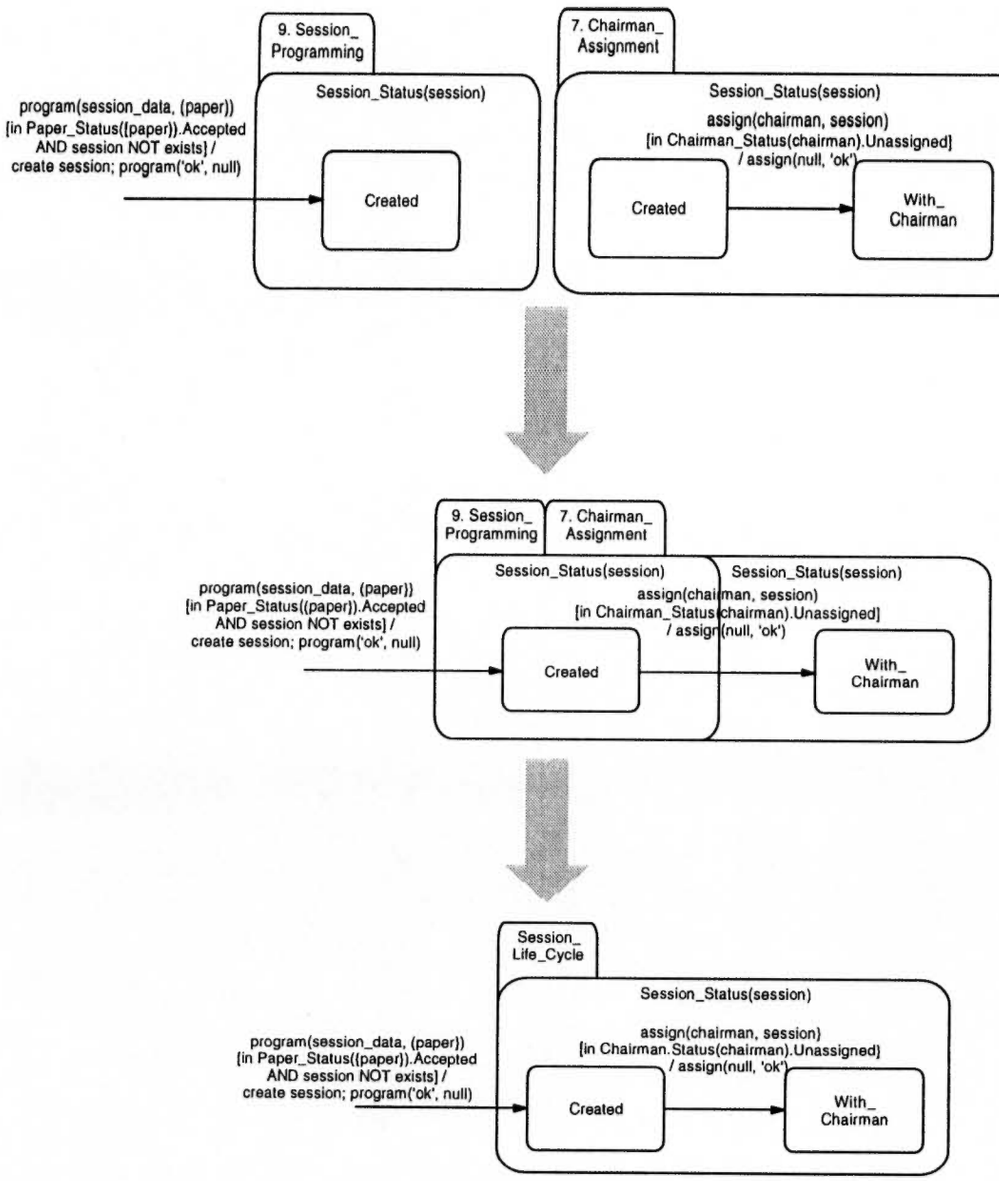


FIGURA 5.9 - Integração dos subestados dos processos no diagrama HLS preliminar para o ciclo de vida da variável session.

Uma situação semelhante à da variável de referência session é a que ocorre em relação ao conjunto de cinco superestados/subestados da variável paper. Este conjunto apresenta um estado inicializado Submitted do superestado Paper_Status(paper) do processo Paper_Submission da figura 5.3 e os pós-estados e pré-estados coincidentes mostrados na tabela 5.2.

TABELA 5.2 - Pós-estados e pré-estados coincidentes nos superestados/subestados da variável paper.

PÓS-ESTADO	PROCESSO	FIGURA	PRÉ-ESTADO	PROCESSO	FIGURA
Accepted	Paper_Selection	5.2	Accepted	Session_Programming	5.1
Distributed	Paper_Distribution	5.2	Distributed	Paper_Revision	5.2
Reviewed	Paper_Revision	5.2	Reviewed	Paper_Selection	5.2
Submitted	Paper_Submission	5.2	Submitted	Paper_Distribution	5.2

Os estados Programmed, no processo Session_Programming, e Rejected, no processo Paper_Selection, são os estados finais para o diagrama de ciclo de vida preliminar da figura 5.10.

Os exemplos anteriores tratavam conjuntos de superestados, subestados e variáveis de referência com mesmos nomes e referindo-se aos mesmos dados. Esta situação muda em relação ao conjunto de sete superestados e subestados relativos à variável author. Como foi explicado na seção 5.1.3, este é o caso de sinônimos, o que dificulta um pouco a conexão dos superestados e subestados envolvidos. Neste conjunto existem dois estados inicializados que são os seguintes:

1. Included do superestado Person_Status(person) do processo People_To_Be_Sent_Call_For_Papers_Inclusion da figura 5.6 e
2. Participant do superestado Author_Status(author) do processo Paper_Submission da figura 5.3.

Ambos os estados inicializados são estados iniciais para o diagrama HLS preliminar de ciclo de vida. A tabela 5.3 mostra os pós-estados e pré-estados coincidentes, apesar de pertencerem a superestados e subestados de nomes e variáveis diferentes.

Observando-se os pós-estados dos diferentes superestados a serem conectados, encontram-se três estados finais para o diagrama HLS de ciclo de vida: No_Intent do processo Letter_Of_Intent_Registration, Refused_Invitation do processo Acceptance_Of_Invitation_Registration e Accepted_Invitation do processo Finallist_Of_Attendees_Generation. A figura 5.11 mostra o diagrama HLS preliminar resultante da conecção de todos estes superestados e subestados. Dado que tanto a variável author, quanto a variável person representam autores efetivos e potenciais (vide seção 5.1.3), o nome adotado para o estado genérico do diagrama HLS preliminar é Author_Status (author). É possível introduzir aqui as variáveis de referências sinônimas author e person

como referências para o estado genérico do diagrama preliminar, porém é preferível postergar esta introdução para a seguinte atividade que trata de verificações de consistência nos nomes de estados e variáveis.

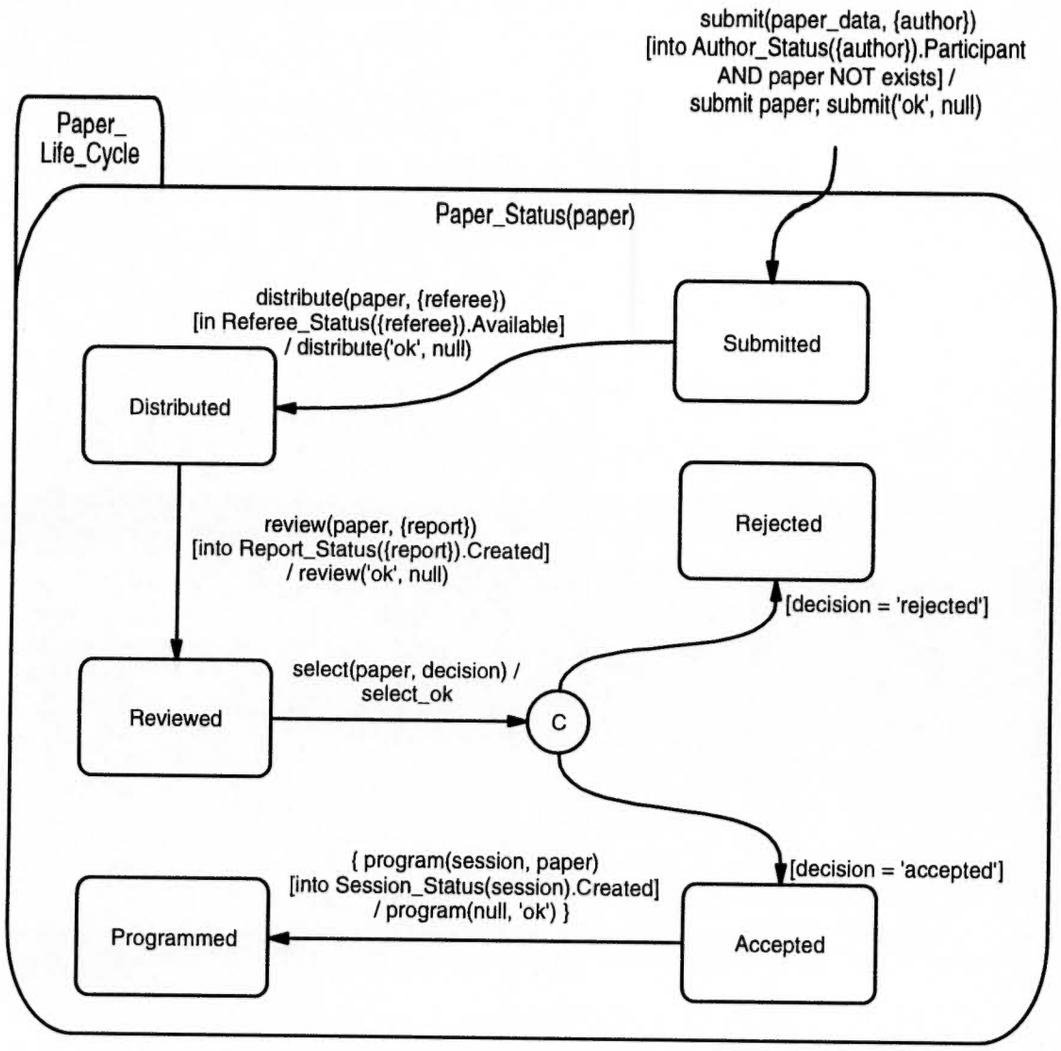


FIGURA 5.10 - Diagrama HLS preliminar para o ciclo de vida da variável paper.

Finalmente, o último conjunto de nove superestados/subestados a conectar é aquele associado à variável person. Aqui existe o estado inicializado Included do superestado Person_Status(person) do processo People_To_Invite_Inclusion da figura 5.5. Na tabela 5.4 são mostrados os pós-estados e os pré-estados coincidentes para o caso das variáveis person, referee e chairman.

TABELA 5.3 - Pós-estados e pré-estados coincidentes nos superestados/subestados das variáveis author e person.

PÓS-ESTADO	SUPER/SUB ESTADO	PROCESSO	FIGURA	PRÉ-ESTADO	SUPER/SUB-ESTADO	PROCESSO	FIGURA
Accepted_ Invitation	Author_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4	Accepted_ Invitation	Author_ Status (author)	Finallist_Of_ Attendees_Generation	5.3
Included	Person_ Status(person)	People_To_Be_ Sent_Call_For_ Papers_Inclusion	5.5	Included	Person_ Status(person)	Call_For_Papers_Sending	5.5
On_Intent	Person_ Status(person)	Letter_Of_Intent _Registration	5.5	On_Intent	Author_ Status (author)	Paper_ Submission	5.2
Participant	Author_ Status (author)	Paper_ Submission	5.2	Participant	Author_ Status (author)	Author_Invitation_Sending	5.3
Sent_Call_ For_Papers	Person_ Status(person)	Call_For_Papers_Sending	5.5	Sent_Call_ For_Papers	Person_ Status(person)	Letter_Of_Intent _Registration	5.5
Sent_Invitation	Author_ Status (author)	Author_Invitation_Sending	5.3	Sent_Invitation	Author_ Status (person)	Acceptance_Of_ Invitation_ Registration	5.4

Os três estados finais para este caso são Refused_Invitation, Assigned e Unavailable dos processos Acceptance_Of_Invitation_Registration, Chairman_Assignment e Paper_Distribution, respectivamente. O diagrama HLS preliminar é o da figura 5.12.

5.3 A VERIFICAÇÃO DE CONSISTÊNCIA NOS NOMES DE ESTADOS E VARIÁVEIS

Logo após ter configurado todos os diagramas HLS preliminares de ciclos de vida, estes devem ser submetidos a uma verificação com o intuito de detectar possíveis inconsistências no uso das variáveis nos estados, eventos, funções de mapeamento, condições e ações, assim como também nos nomes dos superestados e subestados referenciados entre os diferentes diagramas. Muitos deles podem ter sido renomeados, o que pode introduzir algumas inconsistências indesejáveis entre os modelos.

Não existe um procedimento padrão para realizar esta verificação. Deve ser examinado cuidadosamente cada diagrama, verificando-se, em especial, os estados de outros diagramas referenciados nas condições e as variáveis nos superestados, subestados, eventos, funções de mapeamento, condições e ações.

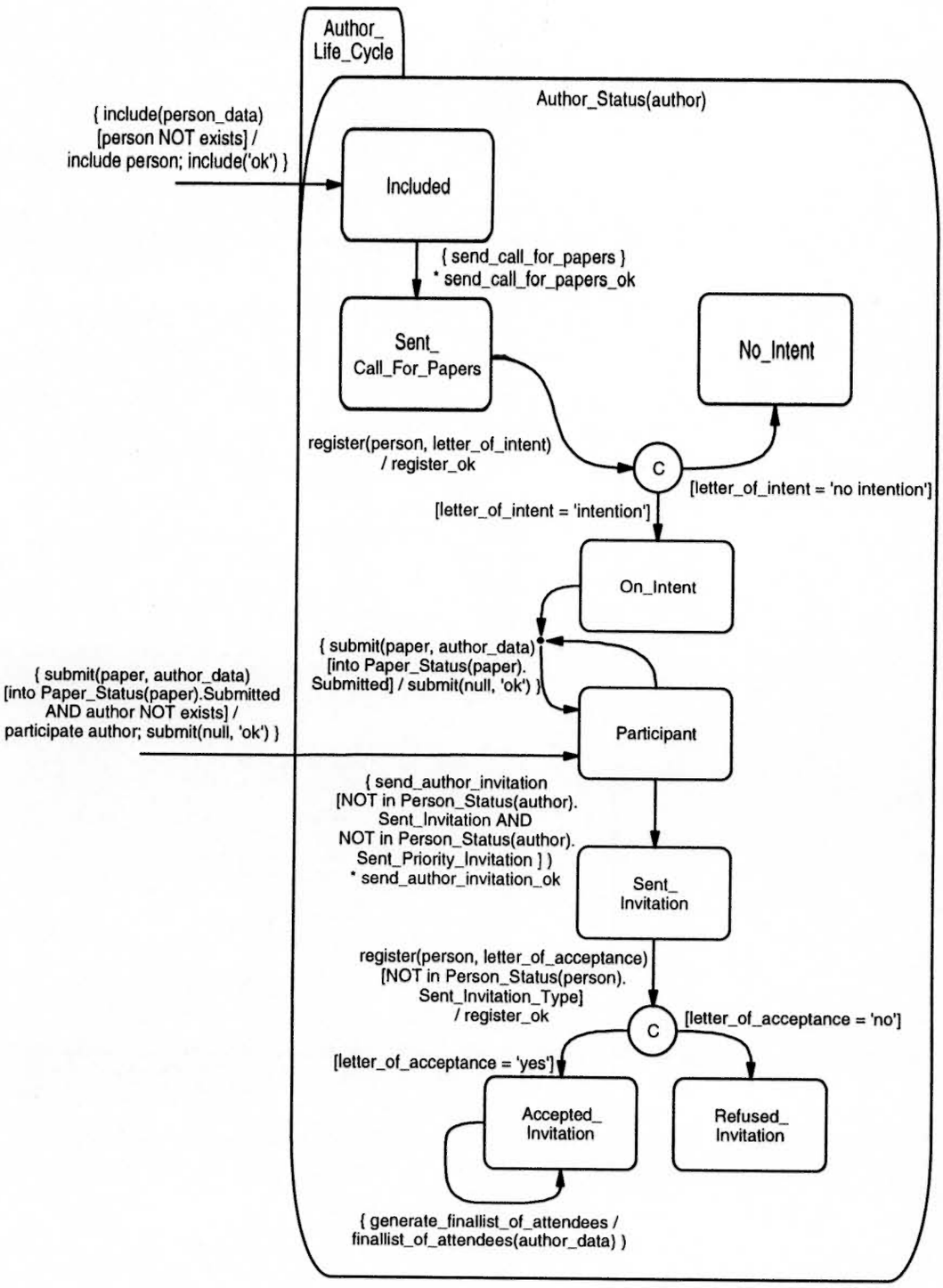


FIGURA 5.11 - Diagrama HLS preliminar para o ciclo de vida das variáveis author e person.

TABELA 5.4 - Pós-estados e pré-estados coincidentes nos superestados/subestados das variáveis person, referee e chairman.

PÓS-ESTADO	SUPER/SUB ESTADO	PROCESSO	FIGURA	PRÉ-ESTADO	SUPER/SUB ESTADO	PROCESSO	FIGURA
Accepted_ Invitation	Person_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4	Accepted_ Invitation	Chairman_ Status(person)	Chairman_ Selection	5.6
Accepted_ Invitation	Person_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4	Accepted_ Invitation	Referee_ Status(person)	Referee_ Definition	5.6
Accepted_ Invitation	Person_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4	Accepted_ Invitation	Person_ Status(person)	Finallist_Of_ Attendees_Generation	5.3
Available	Referee_ Status(person)	Referee_ Definition	5.6	Available	Referee_ Status(referee)	Paper_ Distribution	5.2
Included	Person_ Status(person)	People_To_ Invite_Inclusion	5.4	Included	Person_ Status(person)	Invitation_ Sending	5.4
Included	Person_ Status(person)	People_To_ Invite_Inclusion	5.4	Included	Person_ Status(person)	Priority_Invitation_Sending	5.4
Sent_Invitation	Person_ Status(person)	Invitation_ Sending	5.4	Sent_Invitation_ Type.Sent_ Invitation	Person_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4
Sent_Priority_Invitation	Person_ Status(person)	Priority_ Invitation_ Sending	5.4	Sent_Invitation_ Type.Sent_ Priority_ Invitation	Person_ Status(person)	Acceptance_Of_ Invitation_ Registration	5.4
Unassigned	Chairman_ Status(person)	Chairman_ Selection	5.6	Unassigned	Chairman_ Status(chairman)	Chairman_ Assignment	5.1

Para o caso em que diferentes variáveis de referência sejam utilizadas no mesmo diagrama (isto pode ocorrer ao integrar processos com variáveis sinônimas), isto é, quando existam dentro do diagrama eventos com parâmetros diferentes, mas que se referem aos mesmos dados, a seguinte notação[47] para as variáveis de referência anotadas junto ao nome do superestado é incluída:

Nome_Do_Superestado([variável_1 | variável_2 | ... | variável_n])

onde variável_1, variável_2, ..., variável_n são as variáveis de referência sinônimas associadas ao superestado Nome_Do_Superestado. Isto quer dizer que todas as variáveis e seus respectivos domínios são válidos para o diagrama HLS de ciclo de vida com este superestado.

[47] Esta notação é adaptada do dicionário de dados da análise estruturada moderna. A representação (X | Y) significa que os elementos X e Y são alternativos, que é exatamente o que se deseja expressar aqui.

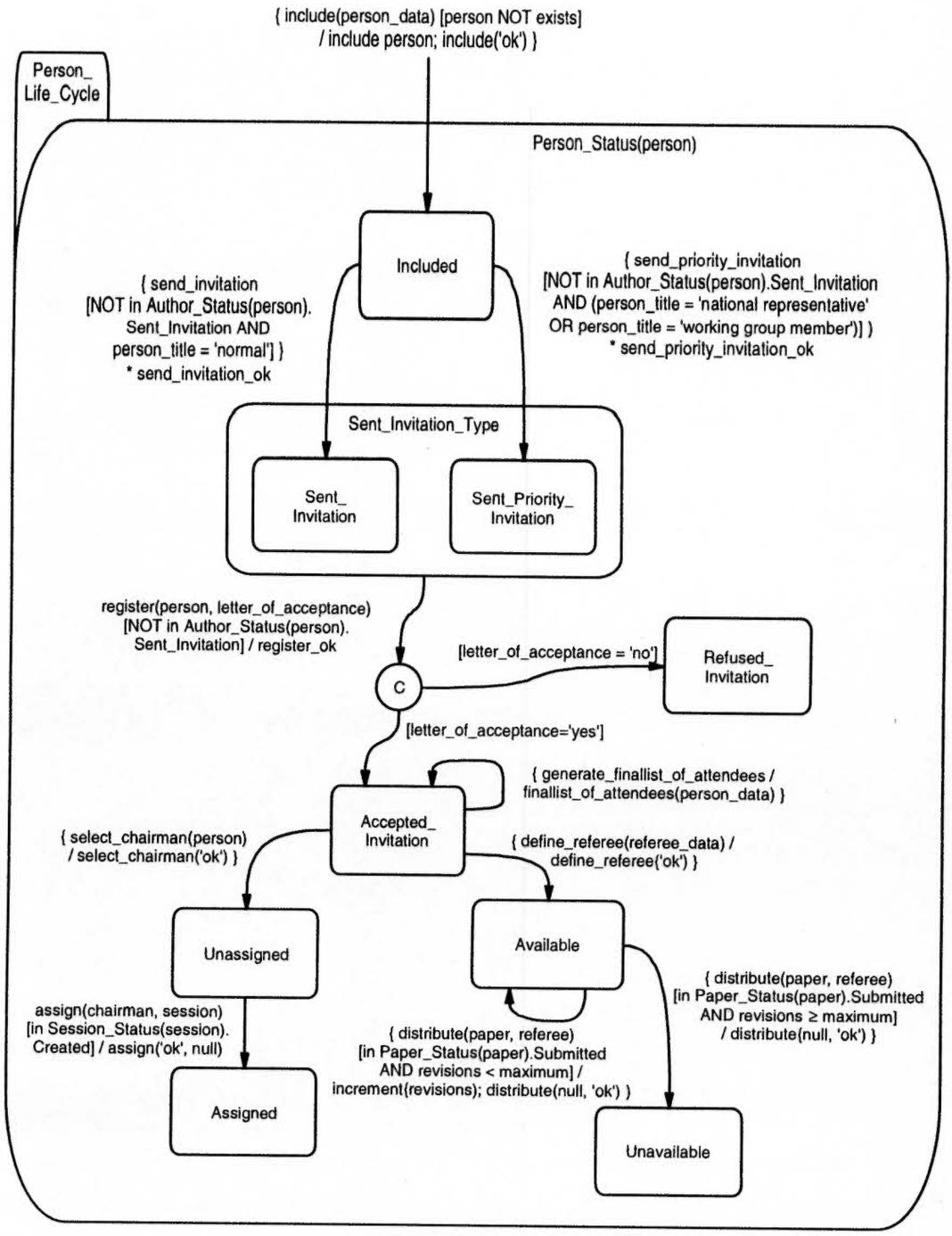


FIGURA 5.12 - Diagrama HLS preliminar para o ciclo de vida das variáveis person, referee e chairman.

Os diagramas HLS preliminares de ciclo de vida tornam-se, após esta verificação e correção, diagramas HLS definitivos de ciclo de vida, ou, simplesmente, dia-

gramas HLS de ciclo de vida. O conjunto destes diagramas é denominado modelo de ciclos de vida do sistema.

Para os diagramas preliminares do caso da IFIP, as verificações e correções resultaram em:

1. Diagrama HLS preliminar de ciclo de vida da variável report (figura 5.8): Não existe nenhum tipo de inconsistência, logo o diagrama HLS definitivo de ciclo de vida é o mesmo diagrama HLS preliminar da figura 5.8.

2. Diagrama HLS preliminar de ciclo de vida da variável session (figura 5.9): Não existe mais o nome de superestado Chairman_Status(chairman) que foi perdido ao ser integrado no novo superestado Person_Status(person), conforme exposto na tabela 5.4 e no diagrama da figura 5.12. A variável chairman na condição é válida em termos de valoração em relação à variável person, isto é, qualquer valor que apresente a chairman dentro do diagrama HLS de ciclo de vida de session é interpretado corretamente no diagrama HLS de ciclo de vida de person, particularmente, para testar o estado envolvido na condição. O diagrama HLS definitivo de ciclo de vida para a variável session é o mostrado na figura 5.13.

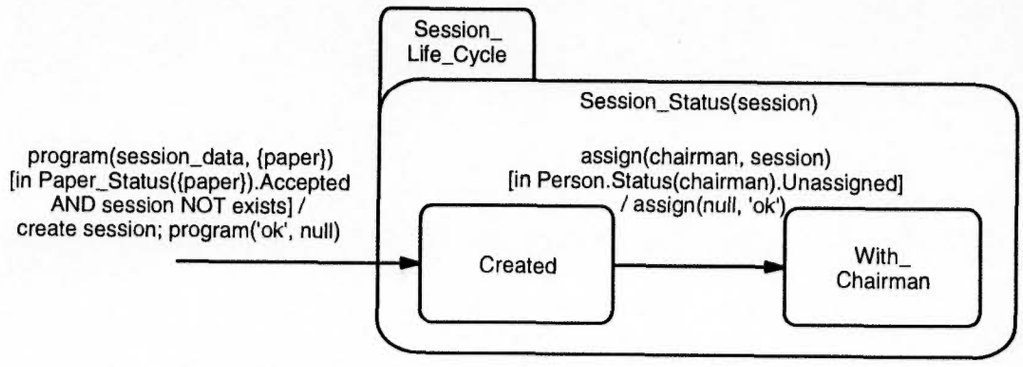


FIGURA 5.13 - Diagrama HLS definitivo para o ciclo de vida da variável session.

3. Diagrama HLS preliminar de ciclo de vida da variável paper (figura 5.10): Também não existe mais o nome de superestado Referee_Status({referee}), que foi substituído pelo nome Person_Status(person) no processo de integração. A variável referee também é valida para o outro diagrama quando seu valor é usado

na avaliação de condições, como neste caso. A figura 5.14 mostra então o diagrama HLS definitivo para paper.

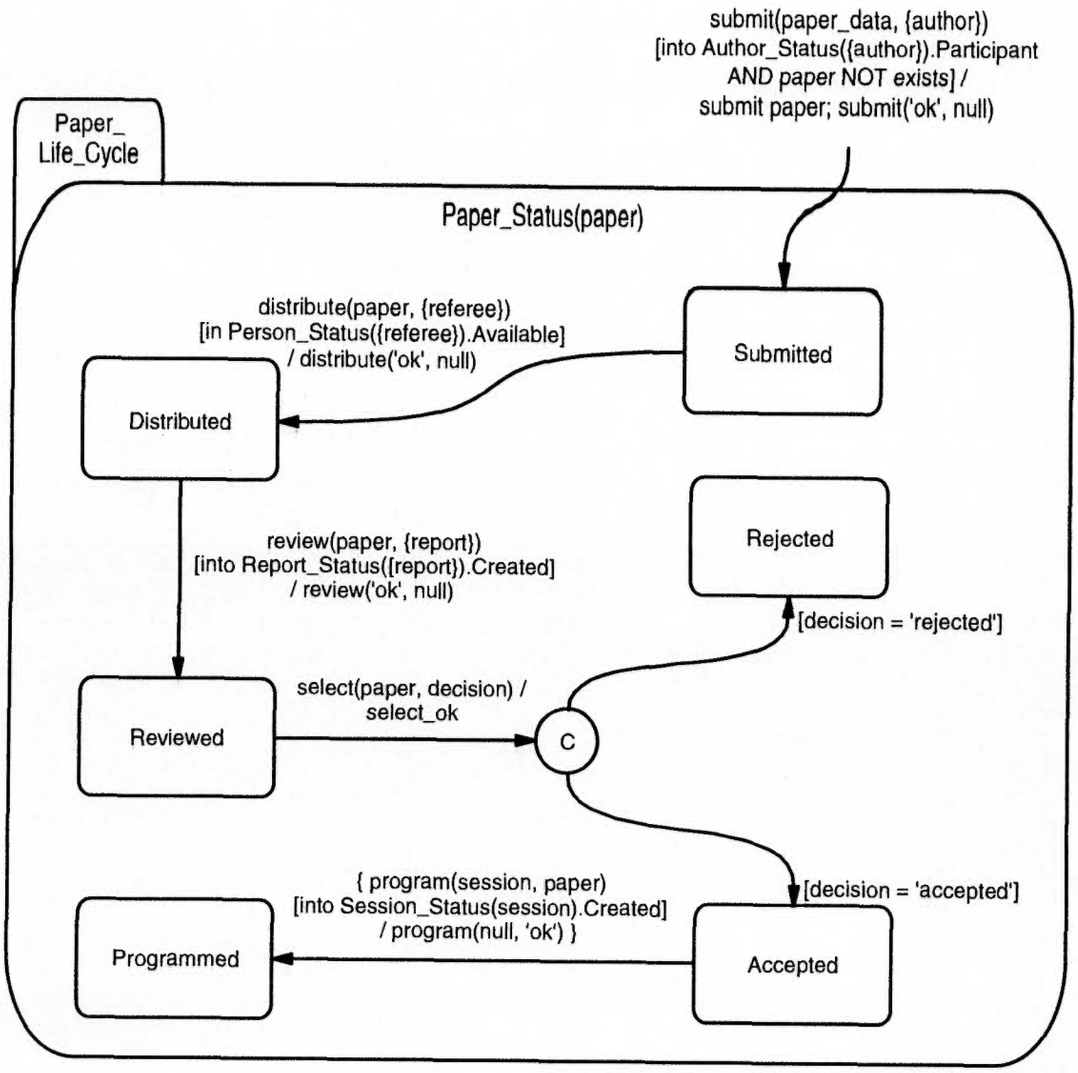


FIGURA 5.14 - Diagrama HLS definitivo para o ciclo de vida da variável paper.

4. Diagrama HLS preliminar do ciclo de vida das variáveis author e person (figura 5.11): Neste diagrama, a condição [NOT in Person_Status(author).Sent_Invitation AND NOT in Person_Status(author).Sent_Priority_Invitation], na transição entre os estados Participant e Sent_Invitation, pode ser simplificada para [NOT in Person_Status(author).Sent_Invitation_Type], dado que ambos os estados testa-

dos na primeira condição compõem um novo superestado Sent_Invitation_Type na segunda condição. Este último superestado não existia no momento de modelar o processo que deu origem a esta transição. Além desta, outra modificação surge como conseqüência de que este diagrama trata de duas variáveis ao mesmo tempo (ele conectou processos de person e de author), como pode ser observado nos eventos que possuem como parâmetro person e também author. O diagrama definitivo fica como o da figura 5.15.

5. Diagrama HLS preliminar do ciclo de vida para as variáveis person, referee e chairman: Neste diagrama, também existem variáveis sinônimas referenciadas pelos eventos de suas transições. Assim, o superestado fica na forma mostrada na figura 5.16.

5.4 Resumo

O presente capítulo apresentou um procedimento para construir diagramas HLS de ciclo de vida a partir dos diagramas HLS dos processos. Este é um processo de integração dos processos do sistema.

Este processo de construção de ciclos de vida inicia com a resolução de conflitos de sinônimos e homônimos de variáveis e estados representados inicialmente. Como resultado, obtêm-se conjuntos de diagramas HLS de processos que tratam dos mesmos dados. A seguir, os pré-estados e pós-estados dos diferentes processos dentro de um mesmo conjunto são ordenados e integrados, formando seqüências de estados para cada tipo de dado. Os estados comuns entre os processos são representados uma única vez, formando, com isto, diagramas HLS maiores, denominados diagramas HLS preliminares de ciclo de vida.

Submetendo-se os diagramas HLS preliminares de ciclo de vida a verificações de consistência nos nomes de estados e variáveis, chega-se aos diagramas definitivos. A correção pode introduzir variáveis de referência sinônimas nos diagramas.

O exemplo usado para mostrar esta construção de ciclos de vida foi o problema de preparação de congressos da IFIP.

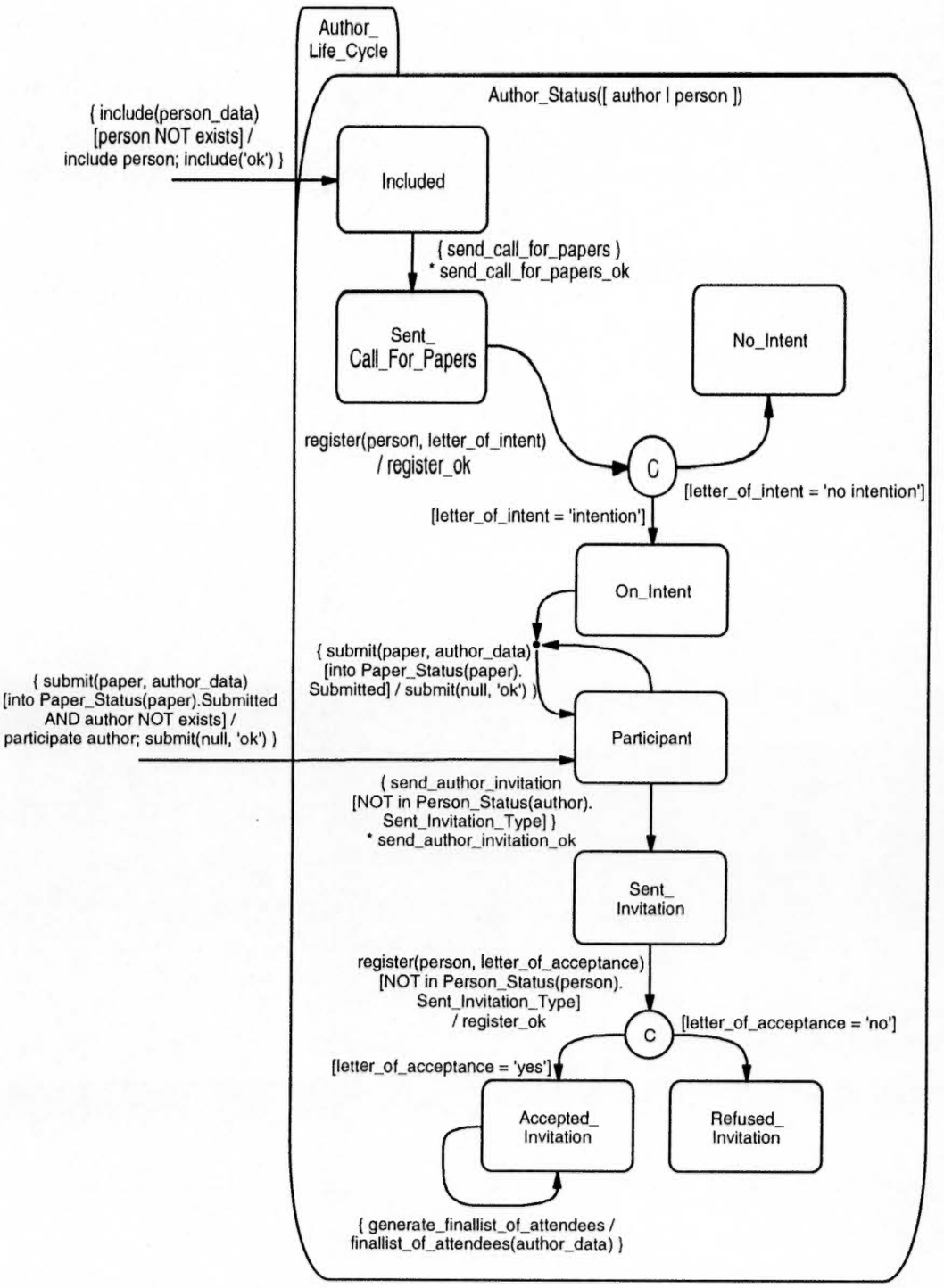


FIGURA 5.15 - Diagrama HLS definitivo para o ciclo de vida das variáveis person e author.

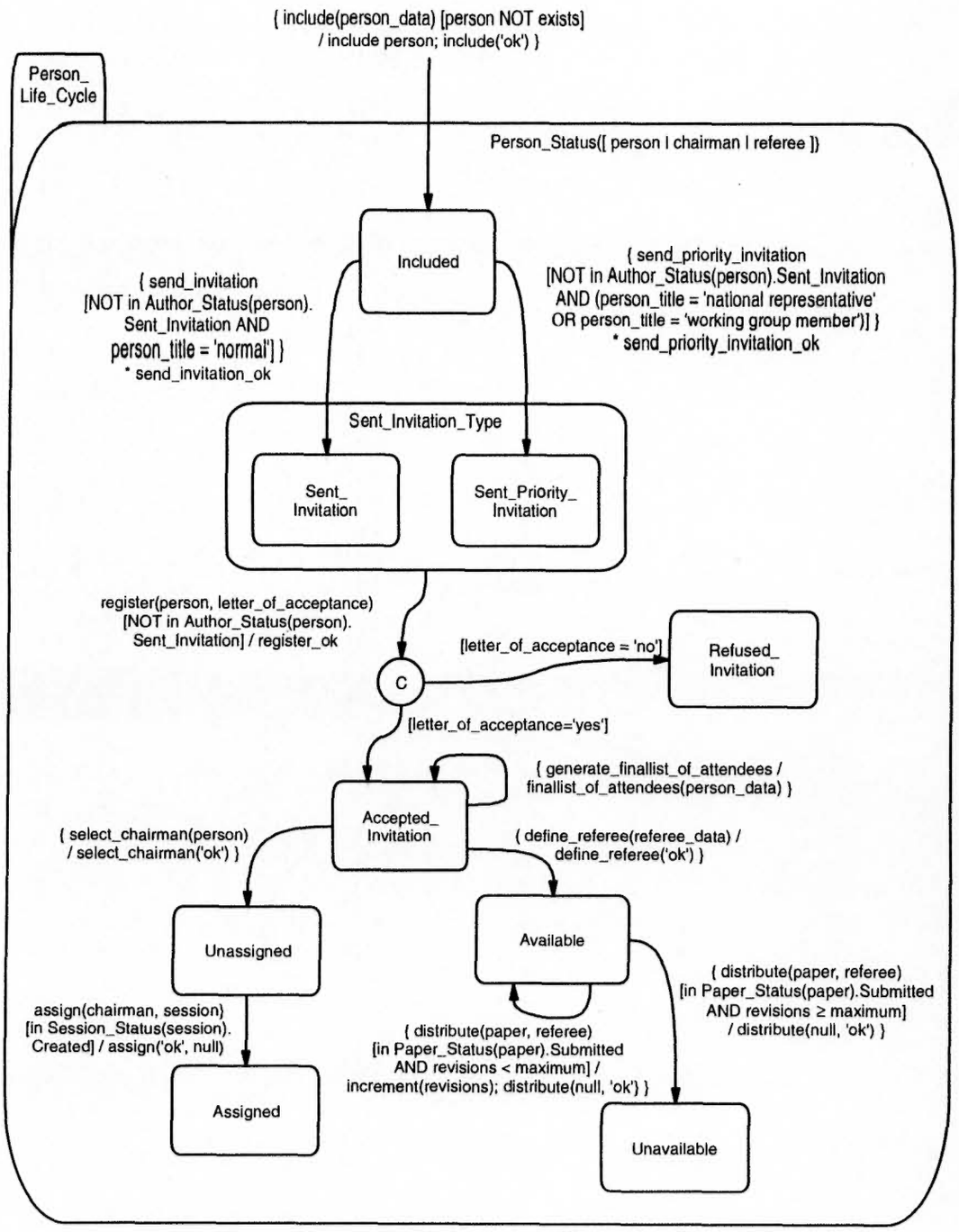


FIGURA 5.16 - Diagrama HLS definitivo para o ciclo de vida das variáveis person, referee e chairman.

6 DERIVAÇÃO DO MODELO ESTRUTURAL DE OBJETOS

A *derivação do modelo estrutural de objetos* é o passo para obter a representação de uma estrutura orientada a objetos. Esta representação mostra as diferentes classes de objetos[48], as relações entre elas por meio de hierarquias de herança, as associações entre as instâncias ou objetos destas classes e os atributos e operações próprios de cada uma delas.

A característica mais importante deste passo é que corresponde exatamente a uma derivação: toda a informação utilizada para construir este modelo estrutural encontra-se contida nas representações dos passos anteriores. Em outras palavras, tanto a modelagem inicial em termos da visão global do sistema e a representação de todos os processos do mesmo, quanto o dicionário de dados e o modelo de ciclos de vida para estes dados contêm todos os elementos necessários para obter uma representação da dimensão estrutural dos objetos.

O modelo estrutural obtido é o "mínimo" que satisfaz o comportamento modelado anteriormente. Este modelo não é definitivo e deve ser adequadamente estendido utilizando os conceitos da modelagem estrutural de objetos. Assim, ainda é possível, entre outras modificações, fazer o seguinte:

- acrescentar outros tipos de objetos ainda não considerados;
- redefinir como atributos de um objeto, outros tipos de objetos diretamente associados;
- redefinir alguns atributos como tipos de objetos associados se requerirem ser tratados especificamente ou associados com outros novos tipos de objetos;
- incluir novas associações estáticas que não as derivadas a partir dos estímulos;
- incluir novas hierarquias de herança ou estender as existentes através de processos de generalização, fatoração ou, ainda, de especialização e
- acrescentar novas operações aos objetos identificados.

[48] Neste capítulo, os termos classe, objeto e tipo de objeto serão tratados indistintamente, salvo indicação em contrário.

Em geral, a derivação é bem estruturada de acordo com um conjunto de regras que serão detalhadas no decorrer do presente capítulo. Neste sentido, não existem decisões arbitrárias a serem tomadas pelo modelador durante este processo.

Como notação para esta representação será usada a proposta por Rumbaugh et al. [RUM 91] para o modelo de objetos da técnica de análise OMT (*Object Modeling Technique*), por ser uma das mais divulgadas. Porém, a derivação descrita é completamente independente desta notação, e pode ser utilizada qualquer outra representação, sempre que possua o suficiente poder de expressão para os conceitos aqui considerados. No anexo "Resumo da Notação OMT para o Modelo de Objetos" encontram-se todas as representações usadas neste capítulo.

As seguintes seções descrevem como obter os diferentes elementos necessários ao modelo estrutural dos objetos a partir dos modelos desenvolvidos nos passos anteriores. A ordem das seções indica uma seqüência *default* para a derivação proposta. A seção 6.1 descreve as diversas fontes de objetos candidatos para o modelo estrutural. A seção 6.2 mostra como derivar hierarquias de herança a partir dos diagramas HLS de ciclo de vida. A seção 6.3 apresenta a forma de determinar as associações, os objetos participantes e as respectivas cardinalidades. A seção 6.4 indica as fontes de atributos para as classes. E a seção 6.5 descreve como identificar as operações dos objetos.

6.1 Objetos Candidatos

A primeira atividade a realizar é procurar quais os candidatos a objetos que podem ser identificados nos modelos já desenvolvidos. As três fontes de *objetos candidatos* são:

- as variáveis de referência dos diagramas HLS de ciclo de vida,
- as variáveis usadas como parâmetros nos eventos, e
- as ações definidas para um conjunto de transições.

As duas primeiras fontes são chamadas diretas porque existe uma variável já classificada que será interpretada como objeto. A última é chamada indireta pela inexistência de algum dado que possa ser entendido como tipo de objeto.

6.1.1 As Variáveis de Referência nos Ciclos de Vida

Como fonte principal direta de objetos candidatos, podem ser indicados os diferentes diagramas HLS de ciclos de vida. A regra de derivação pode ser enunciada da seguinte maneira: *considerar cada diagrama HLS de ciclo de vida como representando o comportamento de um potencial tipo de objeto.* Desta forma, pode ser mapeado diretamente cada dado, representado pela variável de referência do diagrama HLS definitivo de ciclo de vida, em um tipo de objeto[49].

No capítulo anterior, obtiveram-se cinco diagramas HLS definitivos de ciclo de vida das variáveis: report, session, paper, author/person e person/referee/chairman. Para estes últimos dois casos será tomada arbitrariamente a primeira variável como representativa[50], assim, os cinco primeiros objetos candidatos são Report, Session, Paper, Author e Person. A figura 6.1 mostra estes objetos.

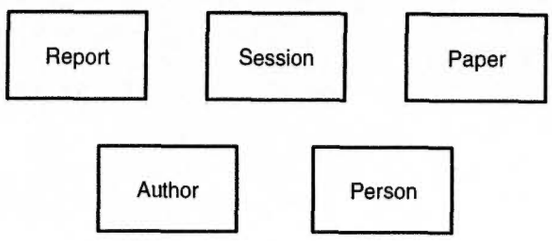


FIGURA 6.1 - Objetos candidatos obtidos a partir das variáveis de referência dos diagramas HLS de ciclo de vida.

6.1.2 Os Parâmetros nos Eventos

Outra fonte direta para obter objetos candidatos constitui os parâmetros nos eventos de algumas transições. Na maioria dos casos, os parâmetros dos eventos referem-se a algum dado que eventualmente possui um ciclo de vida já definido, contudo,

---

[49] Este enfoque de identificação é denominado *enfoque por estado* segundo [DEC 93].

[50] Isto não tem maior relevância como será mostrado na seção 6.2. Se o nome da variável escolhida não for adequado, no momento de derivar as hierarquias de herança, isto se tornará evidente e será facilmente corrigido.

naqueles casos em que isto não ocorra, estes parâmetros podem ser considerados como potenciais objetos. A regra é a seguinte: *todos aqueles dados que aparecem como parâmetros nos eventos do sistema, e que não são variáveis de referência nos diagramas HLS definitivos de ciclo de vida, devem ser considerados como tipos de objetos candidatos.*

Observando os eventos nos diagramas HLS de ciclo de vida do capítulo anterior, chega-se aos seguintes dados, que aparecem como parâmetros e não fazem parte de nenhum diagrama HLS de ciclo de vida como variável de referência:

1. O parâmetro decision do evento select(paper, decision) do diagrama HLS definitivo para o ciclo de vida da variável paper da figura 5.14.
2. O parâmetro letter_of_intent do evento register(person, letter_of_intent) do diagrama HLS definitivo para o ciclo de vida da variável author/person da figura 5.15.
3. O parâmetro letter_of_acceptance do evento register(person, letter_of_acceptance) dos diagramas HLS definitivos para os ciclos de vida das variáveis author/person e person/referee/chairman das figuras 5.15 e 5.16.

Assim, os objetos candidatos obtidos a partir dos parâmetros dos eventos são: Decision, Letter of Intent e Letter of Acceptance. A figura 6.2 mostra todos os objetos candidatos identificados até aqui.

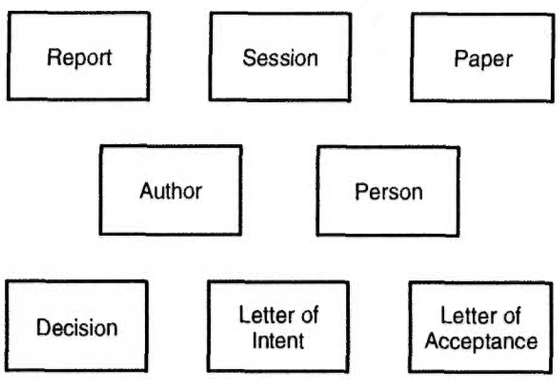


FIGURA 6.2 - Objetos candidatos obtidos das variáveis de referência e dos parâmetros dos eventos nos diagramas HLS de ciclo de vida.

6.1.3 As Ações Associadas a Conjuntos de Transições

A última fonte de objetos é devida a existência de algumas ações (ou operações na orientação a objetos) que não podem ser atribuídas a nenhum dos objetos candidatos antes identificados. Estas ações são aquelas realizadas para todo um conjunto de transições.

Como se verá na seção 6.5, as ações podem ser facilmente atribuídas aos objetos quando associadas a um evento específico. O mesmo não ocorre em se tratando de conjuntos de transições concorrentes. Neste caso, torna-se necessário definir objetos que possuam como operações as ditas ações. É por isto que esta identificação é chamada *indireta*[51].

Este tipo de ação pode ser identificado nos diagramas HLS definitivos de ciclos de vida por estar associado a eventos (e eventualmente a funções de mapeamento, condições e/ou ações) anotados após o símbolo asterisco ("*"). São quatro as ações que podem ser enquadradas neste tipo:

1. A ação send_call_for_papers_ok associada ao evento send_call_for_papers do diagrama HLS definitivo para o ciclo de vida das variáveis author/person da figura 5.15.
2. A ação send_author_invitation_ok associada ao evento e à condição send_author_invitation [NOT in Person_Status(author).Sent_Invitation_Type] do diagrama HLS definitivo para o ciclo de vida das variáveis author/person da mesma figura 5.15.
3. A ação send_invitation_ok associada ao evento e à condição send_invitation [NOT in Author_Status(person).Sent_Invitation AND person_title = 'normal'] do diagrama HLS definitivo para o ciclo de vida das variáveis person/referee/chairman da figura 5.16.
4. A ação send_priority_invitation_ok associada ao evento e à condição send_invitation [NOT in Author_Status(person).Sent_Invitation AND (person_title = 'national representative' OR person_title = 'working group member')] do diagra-

[51] De acordo com [DEC 93] este seria um *enfoque por operação* para a identificação de objetos.

ma HLS definitivo para o ciclo de vida das variáveis person/referee/chairman da mesma figura 5.16.

Os objetos candidatos obtidos desta fonte são: Call for Paper, Author Invitation, Invitation e Priority Invitation. A figura 6.3 mostra então todos os objetos candidatos derivados a partir dos diagramas HLS.

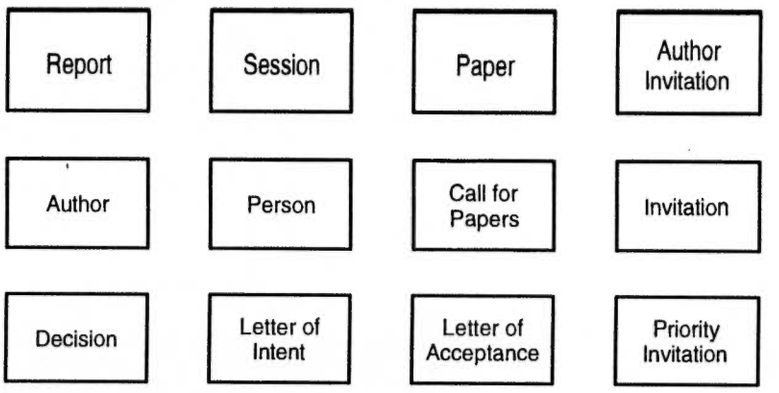


FIGURA 6.3 - Objetos candidatos do problema da IFIP derivados do modelo HLS.

6.2 As Hierarquias de Herança

O comportamento dos objetos é a fonte de informação para derivar hierarquias de herança. Como apenas os tipos de objetos candidatos identificados a partir das variáveis de referência possuem alguma descrição de comportamento, são justamente eles os que podem participar das hierarquias que serão construídas.

O princípio básico que orienta a derivação destas hierarquias é o do comportamento alternativo ou os denominados *subciclos de vida*.

Uma classe candidata a subclasse em uma hierarquia de herança pode ser associada, dentro do modelo de comportamento de uma classe superior (superclasse), a um comportamento alternativo, representado por um subciclo dentro do ciclo de vida da superclasse. O requisito que este comportamento alternativo deve satisfazer é que uma vez que um determinado objeto (instância) opta por este comportamento através de al-

guma transição específica, não existe nenhuma possibilidade de retorno ao comportamento principal ou de conexão a outros comportamentos alternativos. Isto define comportamentos mutuamente exclusivos que constituem por si próprios ciclos de vida para as subclasses candidatas, por isto são denominados subciclos de vida.

Os subciclos de vida podem ser facilmente identificados nos diagramas HLS dos ciclos de vida de um objeto candidato, por apresentarem transições que dividem o *caminho* em duas ou mais alternativas independentes e não conexas entre si, dependendo do valor_verdade da condição. Outra forma que determina a existência de subciclos é quando, a partir de um estado dado, pode ocorrer um evento condicionado que provoque transições para estados diferentes que constituam alternativas exclusivas de comportamento. Ou, ainda, transições com eventos e estados origens comuns representadas por pontos de união. Todas estas formas são apenas variações de notação para transições alternativas. Algumas das formas descritas são mostradas graficamente na figura 6.4, em que as anotações nas transições foram omitidas por simplicidade.

Nesta figura, ambas as situações apresentam apenas dois caminhos alternativos exclusivos e independentes, indicados pelos conjuntos de estados { Estado 1.1, Estado 1.2, ..., Estado 1.n } e { Estado 2.1, Estado 2.2, ..., Estado 2.m }. Estes comportamentos constituem subciclos e podem ser associados a subclasses da classe cujo comportamento é modelado pelo Superestado mostrado na figura. A figura 6.5 mostra a associação dos ciclos e subciclos do diagrama da figura 6.4(b) com a superclasse e subclasses na hierarquia de herança, obtidas a partir desta figura.

Este processo de identificação de subciclos dentro de ciclos de vida é recursivo, isto é, pode ser aplicado novamente, tomando cada subciclo como o ciclo de vida mais amplo inicial com o propósito de descobrir novos subciclos que determinariam novas subclasses dentro da mesma hierarquia de herança. Este procedimento deve ser aplicado sucessivamente até não haver mais subciclos no último ciclo examinado. Ao concluir, obtém-se uma hierarquia que pode apresentar vários níveis, dependendo do número de subciclos encontrados em cada repetição. Quanto mais ramificado for o comportamento exibido pelo tipo de objeto descrito no ciclo de vida inicial, tanto mais profunda será a hierarquia de herança derivada. Se o ciclo de vida inicial de uma classe é seqüencial e linear, não existirá uma hierarquia de herança para aquela classe.

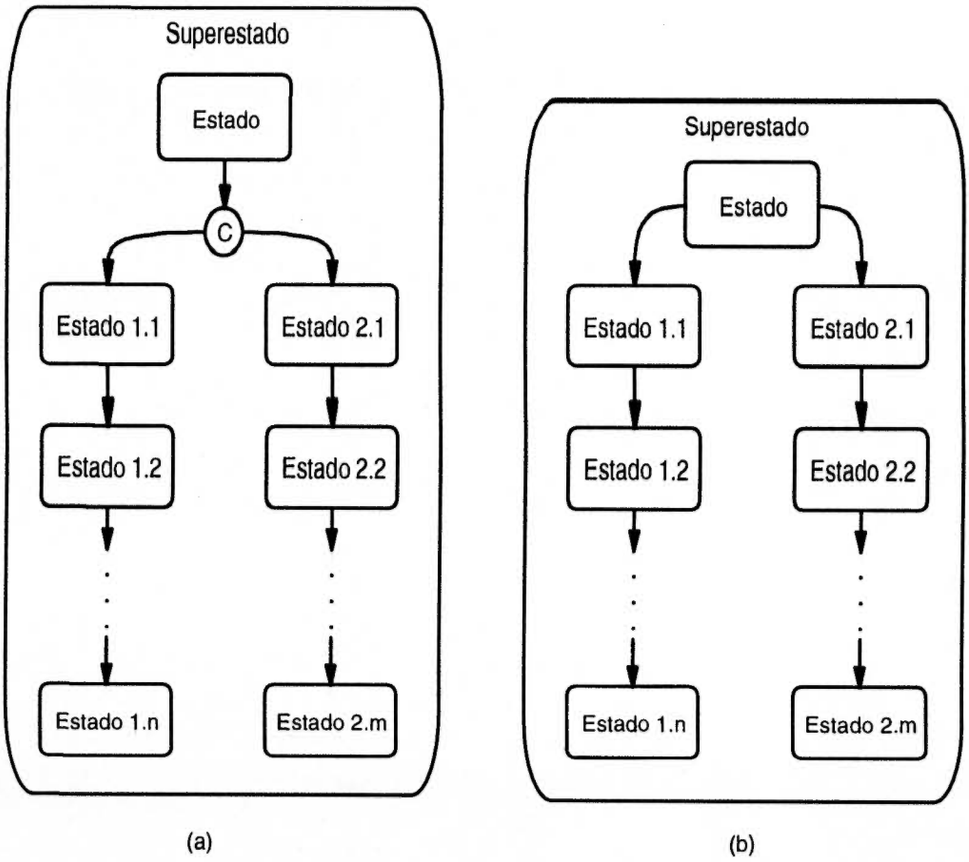


FIGURA 6.4 - Identificação de subciclos de vida: (a) através de uma condição sobre um evento; (b) através de eventos diferentes sobre um estado.

É necessário nomear adequadamente todas as classes envolvidas na hierarquia de herança, desde a superclasse no topo até a última subclasse em profundidade. Os nomes devem ser representativos e consistentes entre si. Caso existam ciclos de vida com variáveis sinônimas, este é o momento de verificar se foi acertada a escolha do nome para o objeto candidato derivado. É muito provável que alguns objetos dentro da hierarquia adotem os nomes das outras variáveis sinônimas.

Para o caso do problema da IFIP e, mais especificamente, para os diagramas HLS de ciclo de vida mostrados no capítulo anterior, as hierarquias de herança são:

- **Diagrama HLS para os ciclos de vida dos objetos candidatos report e session**. Não apresentam subciclos, portanto não existem hierarquias de herança deriváveis nestes diagramas.

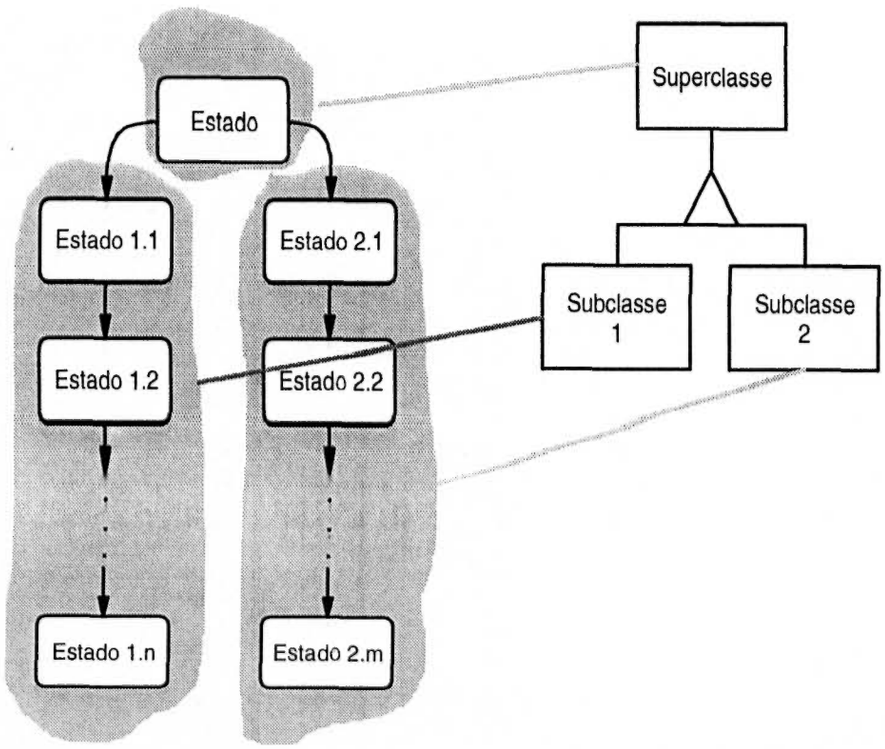


FIGURA 6.5 - Hierarquia de herança hipotética com subclasses exclusivas obtida a partir da figura 6.4(b).

- **Diagrama HLS para o ciclo de vida do objeto candidato paper**. Existem dois subciclos a partir da condição: um subciclo inclui unicamente o estado Rejected, e o outro subciclo inclui os estados Accepted e Programmed. A derivação da hierarquia de herança para paper é mostrada na figura 6.6. As áreas sombreadas mostram os subciclos associados às classes dentro da hierarquia de herança.

- **Diagrama HLS para o ciclo de vida do objeto candidato author**. É possível identificar dois níveis na hierarquia de herança a derivar:

  1. Os subciclos que se iniciam na primeira condição (do evento register(person, letter_of_intent)), incluindo um subciclo o estado único No_Intent e incluindo o outro subciclo os estados On_Intent, Participant, Sent_Invitation, Accepted_Invitation e Refused_Invitation.

  2. Os subciclos que começam na segunda condição (do evento register(person, letter_of_acceptance)). O primeiro subciclo inclui o estado Accepted_Invitation, o segundo inclui o estado Refused_Invitation.

A figura 6.7 mostra as associações entre os subciclos e as classes nos diferentes níveis da hierarquia de herança para author.

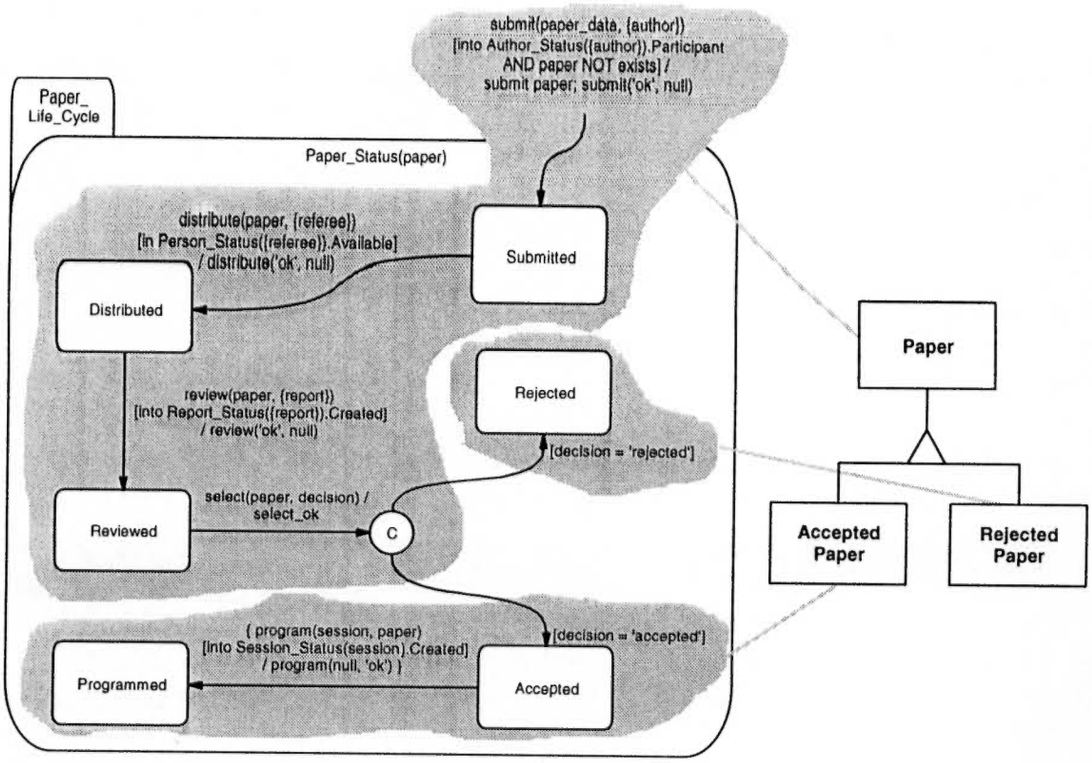


FIGURA 6.6 - Exemplo específico de hierarquia de herança derivada do diagrama HLS de ciclo de vida do objeto candidato paper.

- **Diagrama HLS para o ciclo de vida do objeto candidato person**. Existem também dois níveis na hierarquia derivada:

    1. Os subciclos que se iniciam na condição, o primeiro subciclo incluindo apenas o estado Refused_Invitation, e o segundo subciclo, os estados Accepted_Invitation, Unassigned, Assigned, Available e Unavailable.

    2. Os subciclos dentro do segundo subciclo anterior, isto é, aqueles subciclos que se iniciam nas transições a partir do estado Accepted_Invitation, incluindo os estados Unassigned e Assigned em um primeiro subciclo e os estados Available e Unavailable para o segundo subciclo.

A hierarquia de herança resultante é mostrada na figura 6.8.

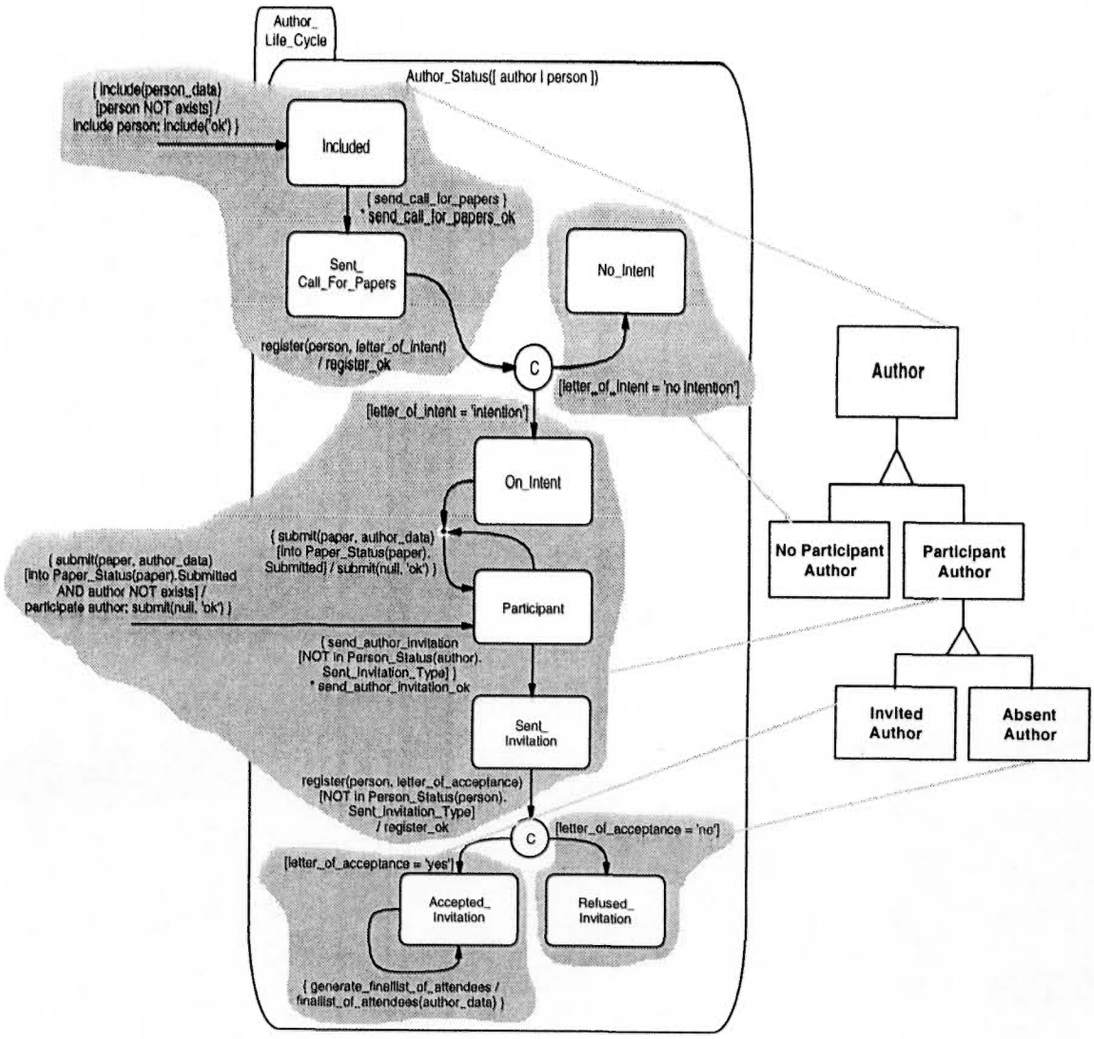


FIGURA 6.7 - Exemplo específico de hierarquia de herança derivada do diagrama de ciclo de vida do objeto candidato author.

6.3 ASSOCIAÇÕES

Tendo os objetos candidatos identificados e as hierarquias de herança definidas para algumas classes, já é possível estabelecer associações estáticas entre instâncias. Estas associações devem ser identificadas considerando seus objetos participantes e devem ter as cardinalidades definidas.

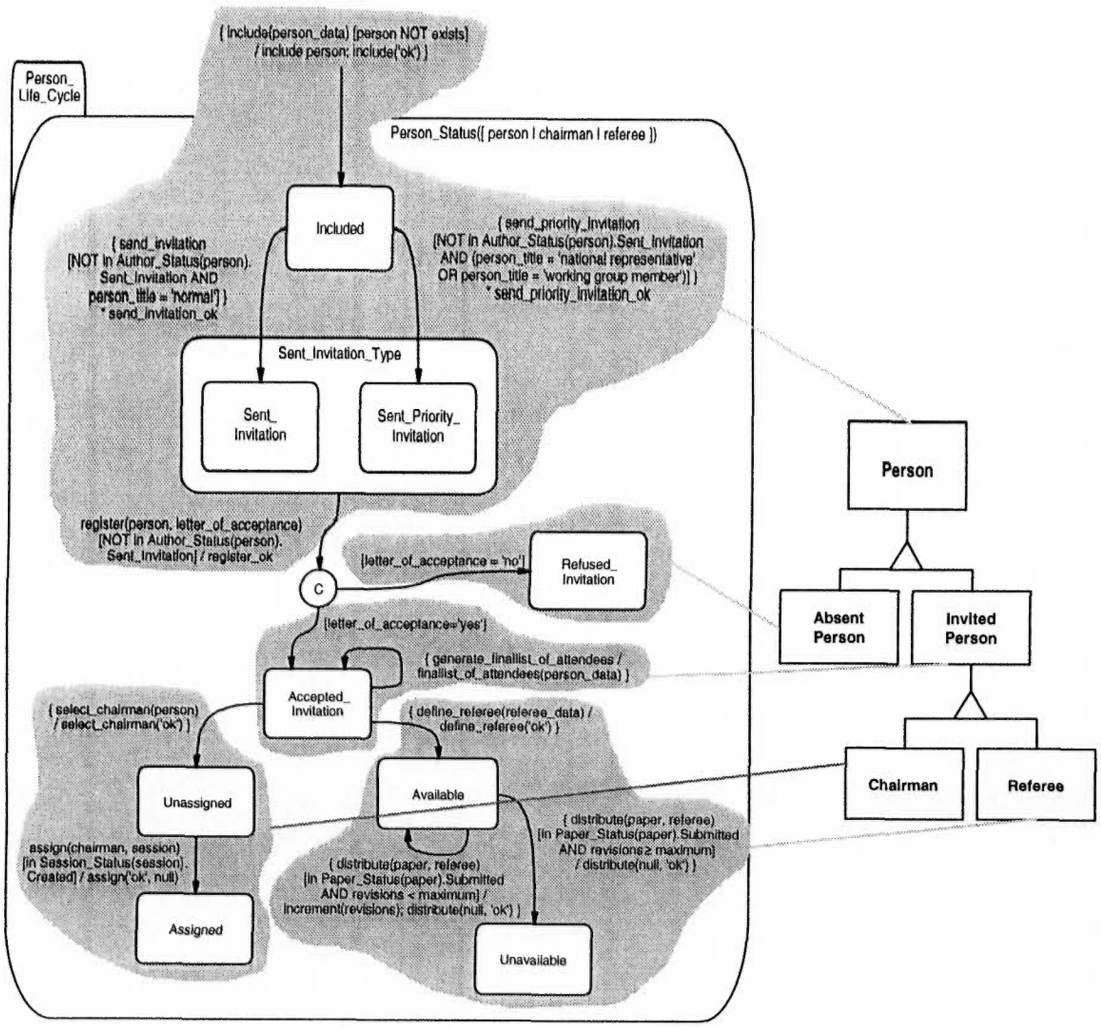


FIGURA 6.8 - Exemplo específico de hierarquia de herança derivada do diagrama de ciclo de vida do objeto candidato person.

6.3.1 A Determinação das Associações e dos Objetos Participantes

As associações de um modelo estrutural de objetos podem ser derivadas de um diagrama HLS a partir dos estímulos que são recebidos dentro dos distintos ciclos de vida dos objetos candidatos.

Como foi descrito nos capítulos anteriores, os eventos que aparecem nos diagramas HLS de ciclos de vida são componentes de um estímulo para o sistema, que é tratado especificamente em um processo. Os diagramas HLS, que representam estes processos do sistema, geralmente mostram concorrência através do tratamento em paralelo dos eventos decompostos a partir do estímulo. Este paralelismo tornou-se menos eviden-

te ao construir os diagramas de ciclos de vida. Esta informação é importante no momento de definir associações entre os objetos candidatos.

O procedimento indicado a seguir mostra como identificar associações e como determinar os objetos envolvidos nestas associações. Não existe restrição em relação a quantos tipos de objetos podem participar de uma associação. O mais comum são as associações binárias, contudo as associações ternárias ou de grau superior também podem ser identificadas[52].

Especificamente, são usados os seguintes diagramas para determinar as associações e objetos participantes:

- diagrama da visão gobal do sistema,
- diagramas HLS dos processos,
- diagramas HLS de ciclo de vida e, eventualmente,
- dicionário de dados.

As regras para a determinação de associações candidatas e a posterior identificação dos objetos participantes estão esquematizadas em duas tabelas de decisão. A tabela 6.1 mostra as regras para poder determinar se um estímulo dado pode ser entendido como uma associação candidata. Desta tabela foram eliminadas as regras cuja combinação de condições não está definida. As notações utilizadas na tabela são as seguintes:

1. *Tipo de estímulo*: Se o estímulo é de consulta ou construtivo.
2. Tratamento concorrente: Se o estímulo é tratado concorrentemente (S = sim) ou não (N = não) ao interior do processo.
3. Nº de parâmetros: Quantidade de parâmetros presentes no estímulo (0 = nenhum, 1 = exatamente 1, 1+ = mais de 1).
4. Tipo do parâmetro: O tipo do parâmetro pode ser lista, composto, ou não composto (escalar ou identificador único).
5. Ação associada a um conjunto de transições: Se existe alguma ação que compõe a resposta para o estímulo, que se encontra associada à ocorrência de um conjunto de transições (S = sim) ou não (N = não).

---

[52] Geralmente estas últimas associações são derivadas de estímulos que apresentam parâmetros compostos e/ou mais de dois parâmetros.

A ação especifica se o estímulo em questão constitui (S = sim) ou não (N = não) uma associação candidata.

TABELA 6.1 - Tabela de decisão para a determinação de uma associação candidata.

CONDIÇÕES	1	2	3	4	5	6	7	8	9	10
1. Tipo de estímulo	consulta		construtivo	construtivo	construtivo	construtivo	construtivo	construtivo	construtivo	construtivo
2. Tratamento concorrente		N	S	S	S	S		S	S	
3. Nº de parâmetros			0	0	1	1	1	1+	1+	1+
4. Tipo do parâmetro					lista	composto	não composto	lista	composto	não composto
5. Ação é associada a um conjunto de transições			S	N						
AÇÃO										
O estímulo constitui uma *associação candidata*	**N**	**N**	**S**	**N**	**N**	**S**	**N**	**S**	**S**	**S**

A tabela 6.2 define as ações para identificar os objetos participantes de uma associação candidata. Não foram incluídas as regras que definem situações que não podem ocorrer ou aquelas nas quais não existem associações candidatas. As regras especificam com S (= sim) e N (= não) os valores de verdade das condições envolvidas. As ações aplicáveis são indicadas com o símbolo (√).

TABELA 6.2 - Tabela de decisão para a identificação dos objetos participantes em uma associação candidata.

CONDIÇÕES	1	2	3	4	5	6	7	8	9
1. Presença de parâmetro(s)	**S**	**S**	**S**	**S**	**S**	**S**	**N**	**N**	**N**
2. Existe hierarquia de herança	**S**	**S**	**N**	**N**	**N**	**N**	**S**	**N**	**N**
3. Parâmetro com comportamento modelado	**S**	**S**	**S**	**S**	**N**	**N**	**N**	**N**	**N**
4. Ação associada a um conjunto de transições	**S**	**N**	**S**	**N**	**S**	**N**	**S**	**S**	**S**
AÇÕES									
1. Objeto participante é derivado do parâmetro do estímulo.					√	√			
2. Objeto participante é derivado da variável de referência do ciclo de vida.	√	√	√	√			√		
3. Objeto participante é derivado da ação associada ao conjunto de transições.	√		√		√		√	√	√
4. Fixar exatamente qual o subciclo para associar com objeto na hierarquia de herança.	√	√					√		

As tabelas são usadas da seguinte forma:

1. Para cada estímulo presente no diagrama da visão global do sistema, verificar se constitui uma associação segundo a tabela de decisão 6.1.
2. Se o estímulo é uma associação candidata, identificar cada um dos tipos de objetos participantes (2 no mínimo), seja considerando a ausência de parâmetros, seja considerando parâmetros/dados compostos tratados concorrentemente, segundo a tabela de decisão 6.2.

Como convenção para os nomes das associações é adotada a seguinte: dar nome à associação a partir do nome do estímulo da qual se deriva. Como os nomes dos estímulos são formas verbais, as associações assumem formas nominais simples. Assim, por exemplo, os nomes das associações derivadas dos estímulos submit e send_call_for_papers seriam submission e sending, respectivamente.

Ao examinar os estímulos modelados na visão global do sistema do problema da IFIP (vide figura 4.7) e ao aplicar estas regras ao caso do problema da IFIP chega-se ao seguinte:

1. O estímulo submit(paper_data, {author_data}) define a associação submission (pelas regras 8 e 10 da tabela 6.1). Os eventos em que se decompõe este estímulo são: submit(paper_data, {author}), que está associado à transição para o estado Submitted, que pertence ao ciclo do objeto Paper (a superclasse na hierarquia de herança); e { submit(paper, author_data) }, que está associado às transições para o estado Participant, que pertence ao subciclo do objeto Participant Author na hierarquia de herança de Author (vide figuras 6.6 e 6.7). Ambos os objetos participantes foram identificados pela aplicação da regra 2 da tabela 6.2.
2. Os estímulos include({person_data}), select_chairman({person}), define_referee({referee_data}) e generate_finallist_of_attendees não constituem associações candidatas, conforme a aplicação da regra 5 da tabela 6.1 para os três primeiros estímulos e pela aplicação da regra 1 da mesma tabela para o último estímulo.
3. Os estímulos send_call_for_papers, send_invitation, send_priority_invitation, send_author_invitation constituem 4 associações candidatas, todas denominadas sending, conforme a aplicação da regra 3 da tabela 6.1. Os objetos participantes (identificados pela aplicação da regra 7 da tabela 6.2) são: Author & Call for Pa-

pers, Person & Invitation, Person & Priority Invitation, e Participant Author & Author Invitation, respectivamente.

4. Os estímulos register(person, letter_of_intent), select(paper, decision) e register(person, letter_of_acceptance) constituem as associações candidatas registration, selection e registration, respectivamente, conforme a aplicação da regra 10 da tabela 6.1. Os objetos participantes (identificados pela aplicação das regras 2 e 6 da tabela 6.2) são: Author & Letter of Intent, Paper & Decision para as duas primeiras associações, e Author Participant & Letter of Acceptance, Person & Letter of Acceptance para a terceira associação.
5. Os estímulos review(paper, {report_data}) e program(session_data, {paper}) constituem as associações candidatas revision e programming, respectivamente, pelas regras 8 e 10 da tabela 6.1. Os objetos participantes identificados pelas regras 2 e 4 da tabela 6.2 são: Paper & Report, Session & Paper Accepted, respectivamente.
6. O estímulo assign(chairman, session) constitui a associação candidata assignment, conforme a regra 10 da tabela 6.1. Nesta associação participam os objetos Session e Chairman, de acordo com as regras 2 e 4 da tabela 6.2.
7. O estímulo distribute(paper, referee) constitui a associação candidata distribution, de acordo com as regras 8 e 10 da tabela 6.1. Os objetos participantes desta associação são Paper e Referee, segundo a regra 2 da tabela 6.2.

O modelo que representa todos os objetos identificados e as associações derivadas é o da figura 6.9.

As associações mostradas nesta figura não apresentam ainda nenhum tipo de cardinalidade, isto é, nenhuma indicação em relação a restrições de quantas instâncias de cada objeto podem participar em cada associação. As indicações a este respeito são apresentadas na próxima seção.

6.3.2 A Cardinalidade

A cardinalidade das associações candidatas pode também ser integralmente obtida a partir de informações contidas nos modelos construídos inicialmente.

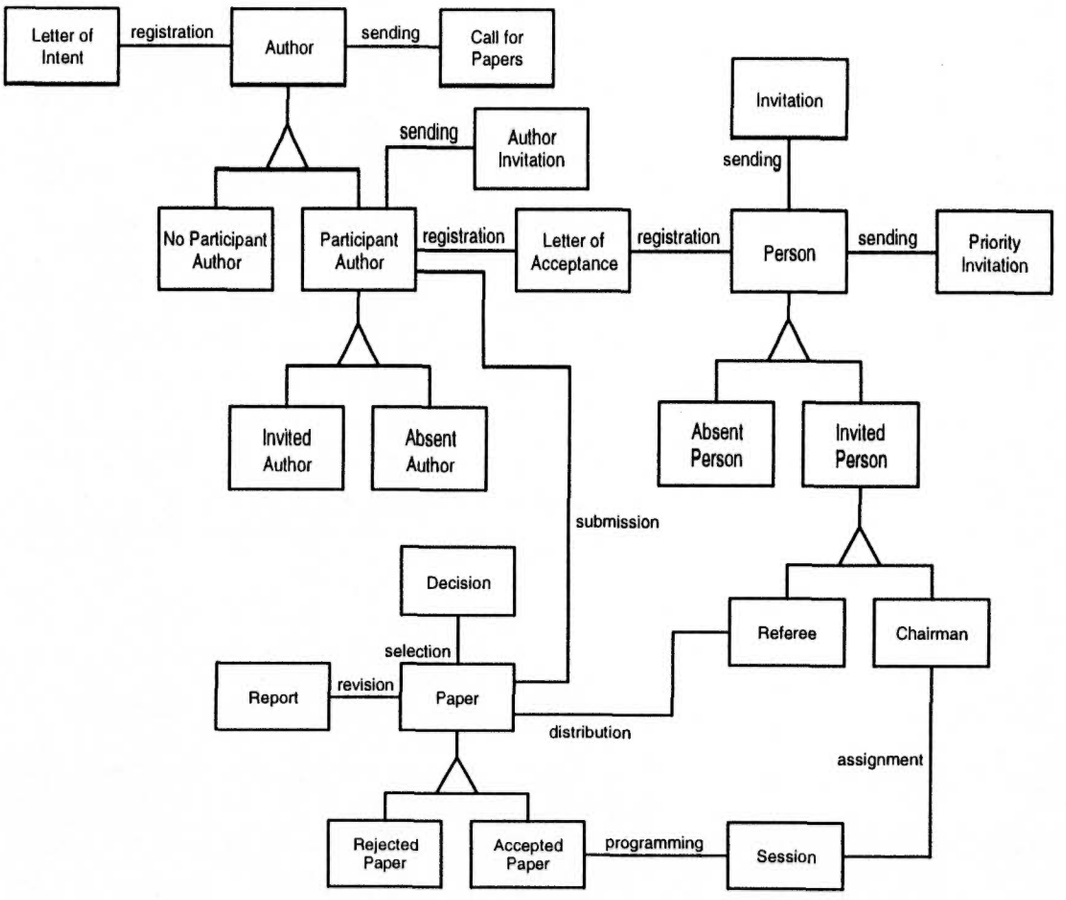


FIGURA 6.9 - Objetos e associações candidatas do problema da IFIP, conforme a derivação.

As ações associadas a conjuntos de transições, as variáveis de referência nos ciclos de vida e os parâmetros dos eventos constituem as fontes de objetos candidatos, conforme a seção 6.1. Cada associação conecta dois ou mais tipos de objetos destas diversas fontes, relacionando suas instâncias de acordo com certas restrições de cardinalidade que definem um número mínimo e um número máximo de participantes de cada tipo. Deve ser examinado separadamente cada tipo de objeto envolvido na associação, considerando sua fonte, para determinar sua cardinalidade específica em termos destes valores mínimos e máximos.

A determinação de cardinalidade é estabelecida apenas para associações binárias. Não são consideradas aqui as associações ternárias ou que envolvam, ainda, mais tipos de objetos, porque a maioria das notações de cardinalidades é ambígua para estes casos (vide por exemplo [RUM 91]).

Os lineamentos gerais para a determinação de cardinalidade das instâncias com informação contida nos diagramas HLS dos ciclos de vida, para o caso de associações binárias que envolvam tipos de objetos obtidos a partir de ações (identificação indireta), são os seguintes:

- Quando o conjunto de transições, que tem associada a ação, ocorre com um ou mais pré-estados, indicando uma opcionalidade de associação das instâncias, devido à existência prévia de instâncias do objeto, que podem estar no pré-estado (ou ainda em estados anteriores) sem terem recebido o evento e sem participarem, então, da associação.
- Quando a ação em questão ocorre em um conjunto de auto-transições, indicando uma associação de múltiplas instâncias, porque a auto-transição representa uma repetição.
- Quando existe uma inicialização de estados[53] com um conjunto de transições concorrentes e uma ação associada a este conjunto, significando uma associação com uma dependência de existência, ou seja, que uma instância sempre é criada por meio da ocorrência do conjunto de eventos e que não poderiam existir instâncias prévias.

A tabela 6.3 apresenta, detalhadamente, as regras para a determinação das cardinalidades para os objetos obtidos, indiretamente, a partir de ações. Nesta tabela, as regras especificam com S (= sim) e N (= não) os valores de verdade das condições consideradas. As ações aplicáveis são indicadas por (√). As cardinalidades são indicadas entre parênteses, onde min indica o número mínimo de instâncias participantes e max, o número máximo delas em uma associação.

Para o caso dos tipos de objetos obtidos de outras fontes, existem outras regras para a determinação da cardinalidade a partir dos diagramas HLS de ciclo de vida. Em termos gerais, pode-se afirmar o seguinte em relação aos objetos que provêm de variáveis de referência de ciclos de vida e de parâmetros dos eventos (identificação direta):

- Os lineamentos estabelecidos para os objetos derivados de ações também são válidos.

---

[53] Isto mostra também que não é necessário considerar a finalização de estados para a determinação de cardinalidades.

- Se o tipo do parâmetro do outro objeto associado é não composto (escalar ou identificador único), então, a participação da instância é única.
- Caso contrário, isto é, se o parâmetro é um lista[54], então, participam várias instâncias na associação.

TABELA 6.3 - Tabela de decisão para a determinação da cardinalidade em associações binárias, considerando os objetos identificados a partir de ações relativas a conjuntos de transições.

CONDIÇÕES	1	2	3	4	5	6	7	8
1. Evento em uma transição de um pré-estado	S	S	S	S	N	N	N	N
2. Evento em uma transição de inicialização de estado	S	S	N	N	S	S	N	N
3. Evento em uma auto-transição	S	N	S	N	S	N	S	N
AÇÕES								
1. Instância com participação opcional (min = 0)	√	√	√	√			√	
2. Instância com dependência de existência (min = 1)					√	√		√
3. Instância com participação única (max = 1)		√		√		√		
4. Instância com participação múltipla (max = n)	√		√		√		√	√

A tabela 6.4 apresenta as regras para estes objetos. Esta tabela apresenta as regras em forma horizontal, porém a notação é exatamente a mesma das outras tabelas de decisão.

Para determinar exatamente quantas são as instâncias de cada tipo de objeto que participam em uma determinada associação binária, deve ser seguido o seguinte procedimento, usando os diagramas HLS dos ciclos de vida e o modelo que inclui objetos e associações sem cardinalidades:

1. Examinar cada transição, no conjunto, que apresente ações associadas. Existe um objeto candidato derivado a partir desta ação. Submeter o evento em questão à tabela de decisão 6.3: primeiramente, a partir do ponto de vista do objeto derivado do ciclo de vida, é possível obter a cardinalidade anotada junto ao objeto da ação; a seguir, a partir do ponto de vista do objeto da ação, pode-se anotar a

[54] Eventualmente, se o conjunto tem, por sua vez, sua cardinalidade limitada por um mínimo e um máximo (vide anexo "Notação dos HLS"), são exatamente estes os valores que devem ser considerados para a cardinalidade da associação entre as instâncias de objetos.

cardinalidade do objeto do ciclo de vida. Repetir para todas as transições deste tipo.

2. Examinar as transições que apresentam eventos com parâmetros, mesmo que eles já tenham sido examinados como listas para o caso anterior. Submeter o evento destas transições à tabela de decisão 6.4, tratando separadamente cada parâmetro para determinar e anotar a cardinalidade do outro objeto envolvido. Repetir até concluir com as transições deste tipo.

TABELA 6.4 - Tabela de decisão para a determinação da cardinalidade em associações binárias, considerando os objetos identificados a partir de variáveis de referência em ciclos de vida e de parâmetros nos eventos.

	CONDIÇÕES				AÇÕES			
	1. Evento em uma transição de um pré-estado				1. Instância com participação opcional (min = 0)			
		2. Evento em uma transição de inicialização de estado				2. Instância com dependência de existência (min = 1)		
			3. Evento em uma auto-transição				3. Instância com participação única (max = 1)	
				4. Tipo do parâmetro				4. Instância com participação múltipla (max = n)
1	Sim	Sim	Sim	Não composto	√			√
2	Sim	Sim	Sim	Lista	√			
3	Sim	Sim	Não	Não composto	√		√	
4	Sim	Sim	Não	Lista	√			√
5	Sim	Não	Sim	Não composto	√			√
6	Sim	Não	Sim	Lista	√			√
7	Sim	Não	Não	Não composto	√		√	
8	Sim	Não	Não	Lista	√			√
9	Não	Sim	Sim	Não composto		√		√
10	Não	Sim	Sim	Lista		√		√
11	Não	Sim	Não	Não composto		√	√	
12	Não	Sim	Não	Lista		√		√
13	Não	Não	Sim	Não composto	√		√	
14	Não	Não	Sim	Lista	√			√
15	Não	Não	Não	Não composto		√	√	
16	Não	Não	Não	Lista		√		√

No caso do problema da IFIP, as cardinalidades derivadas para as associações mostradas na figura 6.9 são as seguintes:

1. Na associação registration entre os objetos Letter of Intent e Author, como o evento register(person, letter_of_intent), visto da perspectiva do objeto Author (variável de referência do ciclo de vida neste caso), ocorre no pré-estado Sent_Call_For_Papers e apresenta o parâmetro não composto letter_of_intent, a regra 7 da tabela de decisão 6.4 indica min = 0 e max = 1 para o objeto Letter of Intent. O mesmo evento, visto da perspectiva do objeto Letter of Intent (parâmetro do evento no mesmo ciclo de vida de Author), como o parâmetro person é não composto, mas não apresenta pré-estado, inicialização de estados nem auto-transição; pela regra 15 da tabela de decisão 6.4, a cardinalidade é min = max = 1 para o objeto Author. Casos semelhantes ocorrem nas associações selection (com os objetos Decision e Paper), registration (com os objetos Letter of Acceptance e Participant Author) e registration (com os objetos Letter of Acceptance e Person).

2. Na associação sending entre os objetos Author e Call for Papers, como o evento send_call_for_papers, visto na perspectiva do objeto Author (variável de referência do ciclo de vida) apresenta uma ação associada ao conjunto destes eventos, a tabela de decisão à aplicar é a 6.3. Como apenas existe um pré-estado Included, a regra 4 indica cardinalidade min = 0 e max = 1 para o objeto Call for Papers. Na perspectiva deste último objeto, que não apresenta nenhum tipo de modelagem de comportamento (sem pré-estados, ou inicialização de estados, ou auto-transições), a regra 8 indica cardinalidade min = 1 e max = n para o objeto Author. Casos semelhantes acontecem com as associações sending (com os objetos Person e Invitation), sending (com os objetos Person e Priority Invitation) e sending (com os objetos Participant Author e Author Invitation).

3. A associação revision entre os objetos Report e Paper possui as cardinalidades min = 0 e max = n para o objeto Report (conforme a regra 8 da tabela de decisão 6.4) e min = max = 1 para o objeto Paper (conforme a regra 15 da mesma tabela).

4. A associação submission entre os objetos Participant Author e Paper possui cardinalidades min = 0 e max = n para o objeto Paper (conforme a regra 1 da tabela 6.4) e min = 1 e max = n para o objeto Participant Author (conforme a regra 12 da mesma tabela).

5. A associação distribution entre os objetos Referee e Paper possui cardinalidades min = 0 e max = n para o objeto Paper (conforme a regra 5 da tabela 6.4) e min = 0 e max = n para o objeto Referee (conforme a regra 6 da mesma tabela).

6. A associação programming entre os objetos Accepted Paper e Session possui cardinalidades min = 0 e max = 1 para o objeto Session (conforme a regra 7 da tabela 6.4) e min = 1 e max = n para o objeto Accepted Paper (conforme a regra 12 da mesma tabela).

7. A associação assignment entre os objetos Session e Chairman possui cardinalidades min = 0 e max = 1 para ambos os objetos Session e Chairman (conforme a regra 7 da tabela 6.4).

A figura 6.10 mostra as cardinalidades acima indicadas para as associações entre os objetos mostrados na figura 6.9.

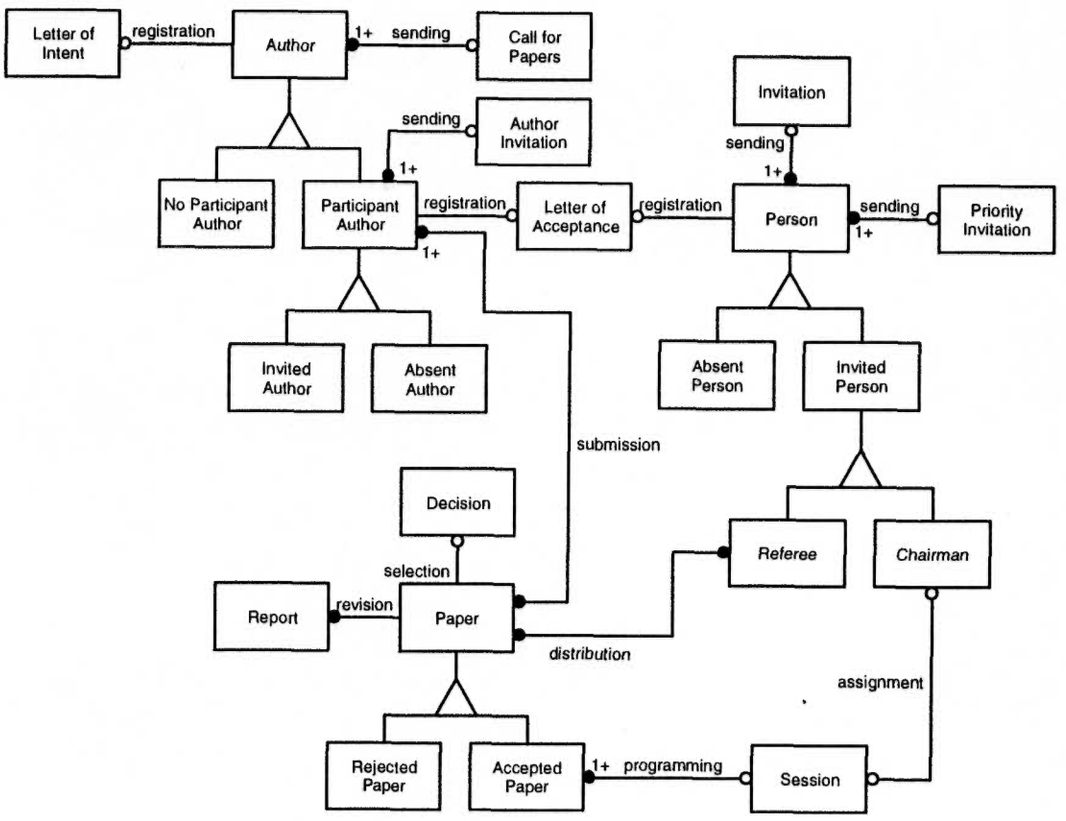


FIGURA 6.10 - Objetos e associações com cardinalidades para o problema da IFIP.

6.4 ATRIBUTOS

Os atributos constituem propriedades estáticas dos objetos. Para identificar quais os atributos que possuem cada um dos objetos do modelo, devem ser examinadas duas fontes de informação: o dicionário de dados construído durante a modelagem inicial do sistema e as variáveis secundárias que são usadas nos diagramas HLS de ciclo de vida.

6.4.1 O Dicionário de Dados

O dicionário de dados, construído no início da modelagem e completado com a modelagem dos processos, fornece informação simples e direta para identificar os atributos dos objetos. Todos os parâmetros compostos (aqueles com o sufixo _data no nome) possuem uma definição de sua composição como, por exemplo, as mostradas na figura 4.8. Geralmente, estes parâmetros tornam-se objetos candidatos por serem variáveis de referência em ciclos de vida ou parâmetros nos eventos que definem associações. Assim, a composição dos dados do dicionário de dados define os atributos para aqueles dados que aparecem como objetos no modelo estrutural. Apenas não são incluídos o dado componente usado como identificador único (anotado com o símbolo @) e aqueles dados que eventualmente são tratados concorrentemente nos processos.

Para os objetos do modelo estrutural da IFIP da figura 6.10, o dicionário de dados (vide figura 4.8) fornece o seguinte:

1. Objeto Author possui os atributos name, institution e address conforme a definição de author_data mais o atributo person_title da definição de Person (ambas as variáveis são válidas no ciclo de vida de Author, por isto é incluída a definição de Person).
2. Objeto Report possui os atributos referee e concept conforme a definição de report_data.
3. Objeto Paper possui os atributos title e subject conforme a definição de paper_data.
4. Objeto Person possui os atributos name, institution e person_title de acordo com a definição de person_data.

5. Objeto Session possui os atributos room e time conforme a definição de session_data.

6. Objeto Referee possui o atributo maximum conforme a definição de referee_data.

6.4.2 As Variáveis Secundárias

As variáveis secundárias nos ciclos de vida são aquelas variáveis que não são de referência e que não aparecem explicitamente como componentes de dados no dicionário de dados, porém são usadas nas condições ou nas ações das transições entre estados do ciclo de vida. Este tipo de variável pode ser entendido como atributo dos objetos candidatos derivados das variáveis de referência daqueles ciclos de vida ou, mais especificamente, dos subciclos onde são usadas, correspondendo assim à atributos de subclasses dentro da hierarquia de herança.

No problema da IFIP, existe um uso de variáveis secundárias dentro do ciclo de vida das variáveis person/referee/chairman (vide figura 5.16). Nas condições e ações associadas ao evento distribute(paper, referee), aparecem as variáveis maximum e revisions. A variável maximum já está incluída no dicionário de dados, mas a variável revisions não. Esta variável pode ser atribuída ao objeto Person (variável de referência do ciclo de vida), porém é mais apropriado considerá-la como atributo do objeto Referee, de acordo com o subciclo particular em que é usada.

A figura 6.11 apresenta os atributos de alguns objetos.

6.5 Operações

As operações próprias dos objetos podem ser facilmente identificadas nos modelos obtidos até aqui. As fontes para esta identificação são duas: as ações nos ciclos de vida e as ações associadas a conjuntos de transições.

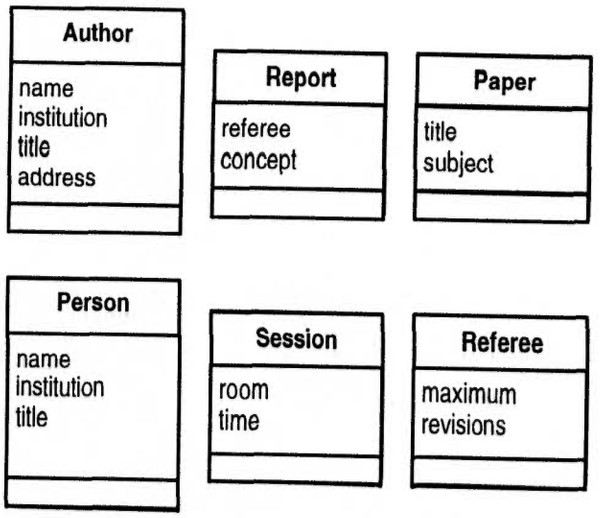


FIGURA 6.11 - Atributos identificados para alguns objetos do problema da IFIP.

As operações são apenas as assinaturas dos métodos que terão que ser implementados posteriormente, por isto não incluem nenhum tipo de descrição algorítmica.

6.5.1 As Ações nos Ciclos de Vida

De maneira semelhante às variáveis secundárias, entendidas como atributos dos objetos, as operações são todas as ações que acontecem no interior dos ciclos de vida dos objetos. Cada ação associada a um evento específico, e definida em uma transição, é uma operação do objeto respectivo. A classe ou subclasse, dentro de uma hierarquia de herança, à qual pertence a ação, depende do ciclo ou subciclo onde é executada a mesma.

Não devem ser consideradas como operações do objeto aquelas ações que se encontram associadas a conjuntos de transições. Estas ações deram origem a outros objetos diferentes aos dos ciclos de vida, e portanto tais ações serão operações daqueles objetos. Esta situação é descrita na próxima seção.

No exemplo do problema da IFIP, podem ser identificadas as operações mostradas na tabela 6.5. Os objetos que não aparecem nesta tabela não possuem ações de acordo com este critério.

TABELA 6.5 - Operações de alguns objetos como ações nos ciclos de vida.

OBJETO	OPERAÇÕES
Accepted Paper	program(null, 'ok')
Author	include person include('ok') register_ok
Chairman	assign('ok', null)
Invited Author	finallist_of_attendees(author_data)
Invited Person	finallist_of_attendees(person_data) define_referee('ok') select_chairman('ok')
Paper	submit paper submit('ok', null) distribute('ok', null) review('ok', null) select_ok
Participant Author	participate author submit(null, 'ok') register_ok
Person	include person include('ok') register_ok
Referee	distribute(null, 'ok') increment(revisions)
Report	create report review(null, 'ok')
Session	create session program('ok', null) assign(null, 'ok')

6.5.2 As Ações Associadas a Conjuntos de Transições

A outra fonte de operações é a das ações que determinam a existência de objetos por estarem associadas a conjuntos de transições (vide seção 6.1.3). Cada uma destas ações permitiu identificar indiretamente um tipo de objeto que possui como operação a dita ação.

As ações que tornam-se operações segundo este critério são as mostradas na tabela 6.6.

TABELA 6.6 - Operações de objetos como ações associadas a conjuntos de transições.

OBJETO	OPERAÇÕES
Author Invitation	send_author_Invitation_ok
Call For Papers	send_call_for_papers_ok
Invitation	send_invitation_ok
Priority Invitation	send_priority_invitation_ok

Finalmente, a figura 6.12 mostra o modelo de objetos completo, incluindo objetos, herança, associações, cardinalidades, atributos e operações, todos derivados a partir dos modelos HLS desenvolvidos previamente.

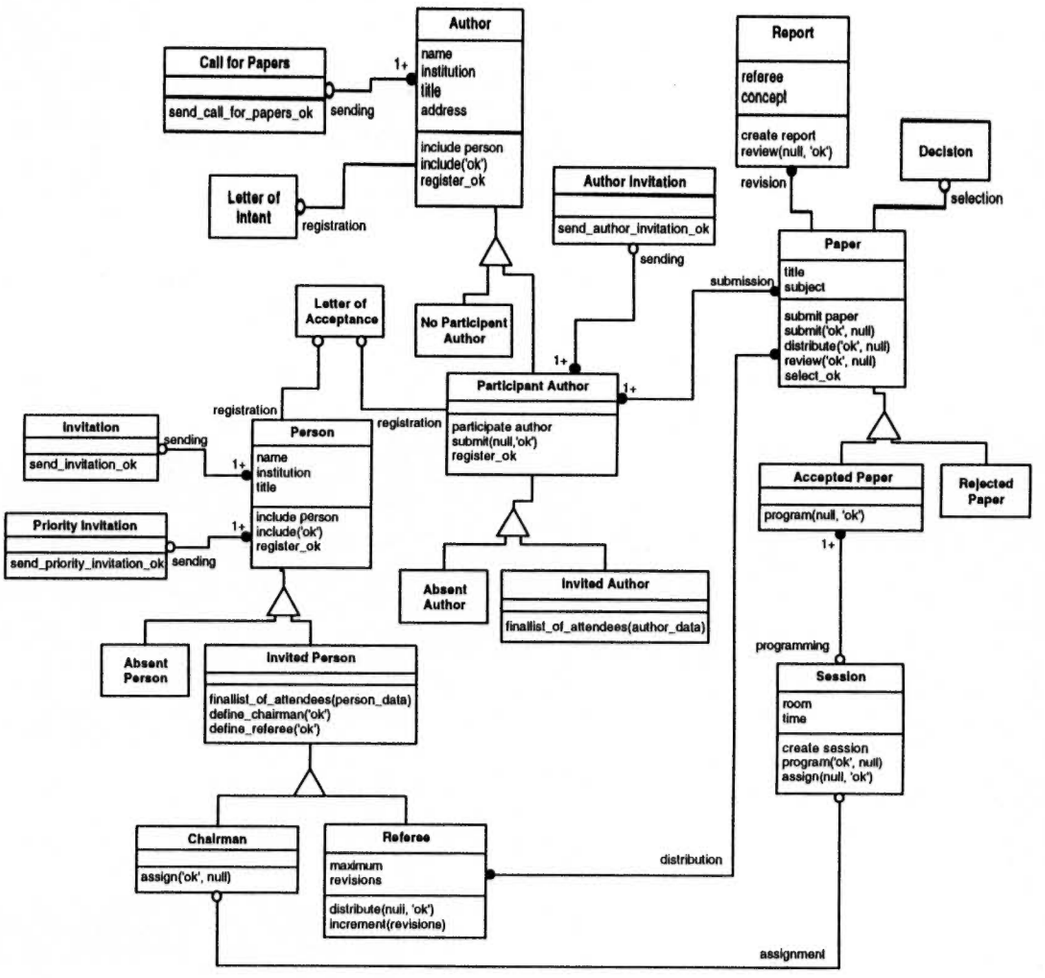


FIGURA 6.12 - Modelo estrutural de objetos do problema da IFIP derivado dos modelos HLS.

6.6 RESUMO

O presente capítulo apresentou o procedimento para derivar um modelo estrutural de objetos a partir dos diagramas da visão global, de processos e de ciclo de

vida e do dicionário de dados. Todos os elementos necessários para o modelo estrutural de objetos são extraídos exclusivamente das fontes indicadas, sem precisar de decisões do modelador.

O processo de derivação começa com a identificação dos objetos candidatos que podem ser obtidos das variáveis de referência dos ciclos de vida, dos dados usados como parâmetros no eventos e das ações associadas a conjuntos de transições. A seguir, através do conceito de subciclos de vida, são derivadas hierarquias de herança baseadas em comportamento para algumas classes.

Com os objetos identificados e as hierarquias de herança definidas, é possível determinar as associações (a partir dos estímulos) e os objetos envolvidos nelas. Foram fornecidas tabelas de decisão para saber quais estímulos constituem associações candidatas e quais os tipos de objetos que participam em cada uma delas. Foi também apresentada uma tabela de decisão para a determinação da cardinalidade em associações binárias.

Finalmente, os atributos dos objetos são obtidos a partir das definições do dicionário de dados e das variavéis secundárias usadas em alguns ciclos de vida. As operações são derivadas das ações mostradas nos diagramas HLS de ciclo de vida.

Com o intuito de dar continuidade, o procedimento é exemplificado com o mesmo problema de preparação de congressos da IFIP mostrado nos capítulos anteriores.

7 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Após a descrição detalhada de todo o processo de modelagem nos capítulos anteriores, usando a técnica proposta sob a estratégia dirigida por comportamento, podem ser apresentadas alguma conclusões e considerações finais. Para isto são indicados os objetivos atingidos, as contribuições, as comparações com outras técnicas e os trabalhos futuros.

Este capítulo final foi dividido como segue: a seção 7.1 mostra o cumprimento dos objetivos levantados na introdução para o desenvolvimento do presente trabalho; a seção 7.2 apresenta as contribuições gerais e as particularidades do presente trabalho; a seção 7.3 apresenta algumas considerações gerais em relação ao processo descrito neste trabalho; a seção 7.4 mostra outras técnicas que seguem a estratégia dirigida por comportamento, e descreve as principais semelhanças e diferenças com a técnica proposta e a seção 7.5 apresenta os lineamentos para futuros desenvolvimentos que podem ser realizados a partir do trabalho até aqui apresentado.

7.1 O CUMPRIMENTO DOS OBJETIVOS

Pode-se concluir que o principal objetivo deste trabalho foi plenamente atingido. Foi detalhado todo um processo de modelagem conceitual de sistemas de informação de um ponto de vista dinâmico/comportamental, e um primeiro passo dentro do *design* orientado a objetos com a derivação de um modelo estrutural orientado a objetos.

Quanto aos objetivos secundários, estes também foram satisfeitos:

1. O estado da arte em MOO foi descrito e analisado criticamente, detectando-se alguns problemas. Os principais problemas indicados foram:
   - o de argumentar a favor da orientação a objetos, que seria uma visão mais "natural" da realidade,
   - o encapsulamento no início da modelagem favorece a continuidade estrutural com o custo de colocar a MOO mais perto do *design* e

- a ênfase majoritária na estratégia dirigida por dados na MOO, o que traz como conseqüência uma sobrevaloração do aspecto estático em relação ao aspecto dinâmico dos sistemas.

2. A proposta resolve alguns destes problemas, porque:
   - os objetos deixam de ser *a forma* de ver a realidade, mudando-se da visão orientada a objetos para uma perspectiva dinâmica/comportamental,
   - o encapsulamento é realizado como último passo do processo de modelagem ao derivar o modelo estrutural de objetos e
   - a estratégia dirigida por comportamento é usada para conduzir a técnica de modelagem proposta, com isto, a estrutura do sistema de objetos é derivada do comportamento que este apresenta.

3. Foi criada uma síntese, especificamente no modelo da visão global, de uma visão sistêmica, que permite contextualizar e conceber o sistema sob modelagem (no diagrama de contexto do sistema) e de uma visão dinâmica, que permite definir as relações do sistema sob modelagem com outros sistemas na forma de estímulos e respostas (no diagrama da visão global).

4. Foi desenvolvida uma extensão denominada *High-Level Statecharts* (HLS) a partir do formalismo *statecharts*, que fornece maior poder de expressão, necessário para a modelagem conceitual de sistemas de informação na perspectiva dinâmica/comportamental. Notações específicas para cada extensão foram desenvolvidas.

5. Conforme o processo de modelagem conceitual proposto, o encapsulamento é introduzido como um primeiro passo do *design*, e apenas com o propósito de obter um modelo estrutural orientado a objetos. Esta derivação não é necessária se a implementação não segue o paradigma da orientação a objetos.

6. Com as extensões propostas no formalismo HLS (particularmente no que se refere a variáveis, eventos, ações e conjuntos de estados concorrentes) viabiliza-se uma derivação relativamente simples e direta de um modelo estrutural de objetos.

7.2 As Contribuições e as Particularidades da Proposta

Em termos gerais, o presente trabalho contribui nos seguintes assuntos:

- **Definição do processo de modelagem conceitual**: O trabalho contribui principalmente na definição do *processo* da MOO, isto é, a forma em que é conduzida a modelagem sob a estratégia dirigida por comportamento. Contrariamente à maioria das técnicas descritas na literatura, que privilegiam os aspectos notacionais e sintáticos dos modelos.
- **Mudança na ênfase do estático para o dinâmico**: O trabalho contribui na viabilização da mudança da ênfase nos objetos, no aspecto estático, na estrutura dos objetos e na estratégia dirigida por dados da MOO; esta mudança acontece para o comportamento, o aspecto dinâmico, as relações dinâmicas e a estratégia dirigida por comportamento da MOO.
- **Amadurecimento da MOO sob a estratégia dirigida por comportamento**: O trabalho fornece uma proposta específica, seguindo a estratégia dirigida por comportamento, ao propor um processo de modelagem conceitual completo de sistemas do ponto de vista dinâmico/comportamental e um primeiro passo de *design* com um modelo estrutural orientado a objetos.
- **Demonstração de que o comportamento pode preceder a estrutura dos objetos**: O trabalho demonstra empiricamente a idéia de que o comportamento exibido por um sistema pode preceder a estrutura dos objetos que podem representá-lo. É esta idéia que está por trás da estratégia dirigida por comportamento.
- **Identificação de padrões de comportamento e interação**: O trabalho contribui para a comprovação de que é possível identificar os objetos como padrões de comportamento (modelo de ciclos de vida), e as associações estáticas entre os objetos como padrões de interação (estímulos e eventos nos modelos dinâmicos). Em outras palavras, que os modelos estáticos de objetos são uma visão (até certo ponto arbitrária) dos modelos dinâmicos do sistema.
- **Determinação das propriedades dos objetos a partir das relações**: O trabalho contribui mostrando que os atributos dos objetos identificados são determinados pelas associações estáticas nas quais ele participa. Dado que estas associações derivam-se dos estímulos e eventos, muitos atributos são identificados mais genericamente como tipos de objetos associados. Conseqüentemente, o modelo estrutural de objetos para um sistema qualquer, obtido com a técnica proposta, será

sempre mais "plano", isto é, com menos atributos e mais classes do que um modelo obtido usando uma estratégia dirigida por dados, em que os atributos sejam representados exclusivamente como tais.

- **Introdução da concorrência na modelagem conceitual**: O trabalho contribui para a modelagem *conceitual* de sistemas no sentido de favorecer o paralelismo interno nos modelos, evitando a imposição de seqüênciações desnecessárias de ações ou operações do sistema. Isto é possível graças ao alto grau de concorrência introduzido na modelagem com HLS.

Em termos mais específicos, o presente trabalho possui as seguintes particularidades:

- **Enfoque de estado**: Usa particularmente o enfoque de estado para identificar objetos, contrariamente à maioria das técnicas encontradas na literatura, que usam enfoques de abstração para esta mesma identificação.
- **Estados atômicos passivos**: Aumenta a precisão da modelagem dinâmica ao reconhecer apenas estados atômicos passivos nos diagramas HLS.
- **Objetos sem concorrência interna**: Define objetos de comportamento mais simples ao supor que estes não apresentam concorrência interna. A forma em que é realizada a integração dos diagramas HLS (ou partes concorrentes dos mesmos) do modelo de processos no modelo de ciclos de vida impede esta concorrência interna.
- **Estímulo tratado concorrentemente**: Introduz o conceito de *tratamento concorrente do estímulo* ao decompor o estímulo, em função do número e dos tipos de parâmetros que apresente, em conjuntos de eventos que são tratados concorrentemente no modelo de processos. Desta forma, não é imposta nenhuma seqüência para o tratamento do estímulo pelo sistema e, especificamente, pelo processo respectivo.
- **Resposta gerada concorrentemente**: Introduz também o conceito de *geração concorrente da resposta* ao compor a resposta com base em um conjunto de ações que são executadas concorrentemente no modelo de processos. As ações são distribuídas como operações em diferentes objetos, contribuindo parcial e concorrentemente à geração da resposta do sistema. Desta forma, é evitada qualquer seqüência desnecessária para a geração desta resposta.

- **Herança definida por comportamento**: Utiliza o comportamento ou, mais especificamente, os comportamentos alternativos e independentes que as classes podem assumir como base para a especialização das superclasses e para a definição das hierarquias de herança, contrariamente à maioria das propostas de MOO na literatura, que usam principalmente a herança como um mecanismo de "economia" conceitual de atributos e/ou operações[55].
- **Estímulo como fonte semântica de associações**: Considera as variáveis usadas como parâmetros nos estímulos (e também presentes nos eventos componentes dos mesmos) como a principal fonte de associação dos objetos. Dois ou mais parâmetros que fluem juntos em um mesmo estímulo definem uma associação estática entre os objetos identificados a partir deles.
- **Identificadores dos objetos**: Incluindo-se os identificadores para os tipos de dados compostos (que se correlacionam com as variáveis de referência dos processos) chega-se, naturalmente, à conclusão de que estes mesmos identificadores e variáveis de referência tornam-se os identificadores dos objetos, contrariamente a algumas propostas de técnicas de MOO na literatura, em que a inclusão de identificadores para os objetos parece mais uma imposição notacional do que uma clara necessidade semântica.

7.3 Considerações Gerais

Algumas considerações gerais podem ser mencionadas em relação ao presente trabalho:

- **A proposta de modelagem e *design* é uma técnica incompleta de MOO**: A rigor, a proposta de modelagem não é uma técnica de MOO completa. Contudo, as extensões requeridas são menores, dado que a informação necessária para incluí-las já está contida nos modelos resultantes. Estas extensões são: algumas descrições adicionais de comportamento de tipos de objetos, um modelo de colaborações entre os objetos e a aplicação do mecanismo de agregação para o modelo estrutural de objetos.

---

[55] Diferente é a situação no *design* ou na programação orientados a objetos, em que a herança é a base para o reuso.

- **As consultas não influem na estrutura dos objetos**: É interessante destacar que todos os estímulos de consulta e, conseqüentemente, os processos que desempenham estas consultas não contribuem em nada para a derivação do modelo estrutural de objetos, isto é, não acrescentam nenhuma informação estática ao sistema na forma de associações deriváveis ou redundantes. Como se está modelando conceitualmente o sistema, a estrutura de objetos não deveria ver-se afetada por considerações de implementação de consultas.
- **O refinamento do modelo estrutural de objetos**: O modelo estrutural de objetos finalmente derivado não é definitivo, ele pode e deve ser refinado utilizando-se os conceitos próprios da modelagem estrutural de objetos, particularmente para identificar composições que, na forma atual da técnica proposta, não são destacadas, e para definir certas classes como atributos de outras.
- **As condições na fase de *design*:** Ao avançar para os seguintes passos na fase de *design* dentro do paradigma da orientação a objetos, é necessário definir os métodos para as operações indicadas no modelo estrutural. Estas operações foram derivadas de ações que eram realizadas nas transições dos processos. Nestas mesmas transições também existem algumas condições, as quais podem servir para definir pré-condições para os respectivos métodos. Adicionalmente, as condições de consistência intraprocesso e interprocessos podem requerer mensagens adicionais entre os objetos.

7.4 Outras Técnicas Dirigidas por Comportamento

Como foi indicado no capítulo 1, existem poucas técnicas de MOO conduzidas sob a estratégia dirigida por comportamento. Duas são as principais:

- *Object Behavior Analysis* (OBA) de Rubin & Goldberg [RUB 92] e
- a técnica de análise da metodologia *Object-Oriented Software Engineering* (OOSE) de Jacobson et al. [JAC 92].

Ambas as técnicas são brevemente descritas a seguir e comparadas com a proposta de modelagem.

7.4.1 Técnica OBA

A técnica OBA [RUB 92] tem como idéia central que o comportamento esperado do sistema permite identificar os objetos que iniciam e participam deste comportamento.

7.4.1.1 O Processo de Modelagem

A técnica OBA sugere cinco passos interativos:

1. **Definir o contexto da análise**: Trata com metas e objetivos, recursos, áreas de atividades e plano preliminar da análise.
2. **Entender o problema**: Consiste na definição de cenários de interação para derivar *scripts* de comportamentos. Adicionalmente, são desenvolvidos glossários de partes e serviços. Os atributos são derivados e também definidos em um glossário.
3. **Definir os objetos**: São definidos como objetos as partes envolvidas nos *scripts*, sempre que a parte seja um participante interno do sistema, e não esteja simplesmente iniciando um comportamento. Estes objetos são descritos nos *Cartões de Modelagem de Objeto*.
4. **Classificar os objetos e identificar os relacionamentos**: São definidos os relacionamentos contratuais dos objetos através dos serviços fornecidos/contratados. Os objetos são organizados hierarquicamente de acordo com os conceitos de abstração (generalização), especialização e fatorização.
5. **Modelar a dinâmica do sistema**: São definidos os estados de cada objeto em um glossário. São modelados os estados e os eventos que disparam as transições em diagramas de ciclo de vida. Adicionalmente, as atividades dos *scripts* devem ser seqüencializadas/paralelizadas utilizando alguma notação dinâmica adequada.

7.4.1.2 Os Modelos Resultantes

Os modelos resultantes da aplicação da técnica OBA são:

- **Cartões de modelagem de objetos**: Cartões que descrevem, entre outros, os seguintes itens dos objetos: nome, herança, lista de atributos (nome e *script* em que é usado), lista de serviços fornecidos (nome e *script* em que é usado) e lista de serviços contratados (nome, objeto ao qual pertence o serviço e *script* em que é usado).
- **Diagrama de ciclo de vida de objetos**: *Statecharts* tradicionais que descrevem o comportamento padrão de uma instância de cada tipo de objeto.
- **Diagrama de relacionamentos contratuais**: Diagrama que descreve os relacionamentos entre objetos na forma de contratos do tipo *fornece informação* e *requer ação*.
- **Diagrama de relacionamentos organizacionais**: Diagrama que mostra as hierarquias de herança. Cada tipo de objeto tem seus serviços indicados.
- **Diagrama de seqüenciação de operações**: Diagramas que representam as atividades, descritas nos *scripts*, em forma seqüencial/paralela. Não existe um modelo definido para tal propósito, podem ser usados *statecharts* [HAR 87] ou redes de Petri [HEU 90].
- **Glossário de atributos**: Glossário que descreve os atributos em termos do nome, definição, contrato da parte que utiliza o atributo, serviços acessador e atualizador do atributo, tipo de valoração (mono ou multivalorado), domínio e a determinação ou não pelo atributo dos estados para a parte.
- **Glossário de estados**: Glossário que descreve o nome dos estados, a definição em função de atributos, uma descrição e a referência aos *scripts*.
- **Glossário de nomes de partes**: Glossário que descreve as partes (iniciadores e/ou participantes de cada comportamento descrito nos *scripts*) em termos do nome, definição, *script* em que aparece a parte e papel desempenhado (iniciador, participante ou iniciador/participante).
- **Glossário de serviços**: Glossário que descreve os serviços em termos do nome, definição, parte e *script* em que aparece serviço.
- ***Scripts* do sistema**: Tabelas de descrição dos comportamentos esperados do sistema. Cada *script* possui um cabeçalho que inclui, entre outros, o nome, as pré-

condições para a aplicação do *script* e as pós-condições resultantes da conclusão do *script*. Os comportamentos são descritos com um iniciador (que inicia a ação a ser descrita), a ação realizada, o participante (que interage com o iniciador, recebendo a ação) e o serviço (que é realizado pelo participante).

7.4.1.3 O Comparativo

As principais equivalências entre a técnica OBA e a proposta de modelagem são mostradas na tabela 7.1.

TABELA 7.1 - Tabela comparativa de conceitos da técnica proposta em relação à técnica OBA.

CONCEITO	EQUIVALENTE NA TÉCNICA OBA	OBSERVAÇÕES
ação	serviço	
agente externo	parte apenas com o papel de iniciador	
associações estáticas		deriváveis dos relacionamentos contratuais
atributos	atributos	
concorrência nos processos	seqüenciação/paralelização de atividades	representação com *statecharts* ou redes de Petri
dicionário de dados	glossários de atributos e cartões de modelagem de objetos	
estímulo	ação	modelada no *script*
evento	implícito no passo das pós-condições de um *script* para as pré-condições de outro *script*	corresponde mais precisamente à transição
composição da resposta		
decomposição do estímulo		
hierarquia de herança	derivadas em função de atributos e serviços	
identificação de objetos	participantes que desempenham algum serviço contratado	
identificador único		
integração de ciclos de vida	integração de (ou partes de) *scripts* em ciclos de vida dos objetos	
limites do sistema	implícito ao definir os iniciadores que não são participantes	
parâmetros	incluídos nas ações	
processo		
resposta		
supra-sistema		

Em síntese, as principais semelhanças entre ambas as técnicas são:

- **Construção de ciclos de vida**: Ambas as técnicas constroem os ciclos de vida a partir dos componentes contidos no modelo de comportamento inicial do sistema.

- **Descrição dos atributos**: Ambas as técnicas usam dicionário de dados ou glossários e cartões de modelagem para descrever os atributos.

As principais diferenças dizem respeito a:

- **O modelo fundamental**: Na técnica OBA, são os *scripts* de comportamento que tratam com muito detalhe cada ação iniciada desde o exterior do sistema. Os *scripts* devem posteriormente ser particionados de acordo a mudanças de estado que ocorram nos objetos participantes. A seqüência/concorrência na execução destas partições deve ser definida posteriormente. Os HLS apresentam a vantagem de tratarem todos os elementos (estímulos, estados e respostas) já em forma seqüencial/concorrente.
- **O nível de abstração na descrição do comportamento do sistema**: Na técnica OBA, a descrição do comportamento do sistema é tratada no mesmo nível de abstração dos serviços dos objetos potenciais. Na técnica proposta, a abstração é maior ao considerar os processos como receptores dos estímulos. Esta nível de abstração parece mais apropriado face a modelagem conceitual.
- **A identificação dos objetos**: Na técnica OBA, a identificação dos objetos é feita através dos participantes que desempenham serviços. Na técnica proposta, é feita a partir das variáveis tratadas pelo sistema diminuindo com isto a influência do modelador, que na técnica OBA deve definir inicialmente o objeto que interage com o iniciador de uma ação.
- **A interação com os agentes externos**: A técnica OBA centra-se em interações unidirecionais entre o iniciador (agente externo potencial) e o sistema. Na técnica proposta, esta interação é bidirecional, isto é, vai do estímulo do emissor até a resposta para o receptor fornecendo maior riqueza de interação do sistema com o ambiente.

Em [BUS 94e] é apresentada a aplicação da técnica OBA ao problema da IFIP. A figura 7.1 apresenta o exemplo específico de um *script* para a submissão de um artigo dentro deste problema.

Nome do *Script*	Paper.Submission		
Autor	GBR		
Pré-condição	true		
Pós-condição	paper_exists(P), author_exists(A_1, A_2, ..., A_n)		
Referência	Core Activity Area - Program Administration		
Iniciador	**Ação**	**Participante**	**Serviço**
Program Committee	include paper P of authors A_1, A_2, ..., A_n	Paper	inclusion
Paper	verify existence of A_1	Author	verification
Paper	include A_1 if not exists	Author	inclusion
Paper	verify existence of A_2	Author	verification
Paper	include A_2 if not exists	Author	inclusion
...	...	...	...
Paper	verify existence of A_n	Author	verification
Paper	include A_n if not exists	Author	inclusion

FIGURA 7.1 - Exemplo específico de um *script* para a submissão de um artigo no problema da IFIP, conforme a técnica OBA.

7.4.2 Técnica de Análise na Metodologia OOSE

A metodologia OOSE [JAC 92] (uma simplificação da metodologia *Objectory*) cobre as fases de análise, *design* e implementação. Trata de modelos de requisitos, de análise, de *design*, de implementação e de teste.

A técnica de análise, dentro desta metodologia, baseia-se na idéia de que o modelo do sistema pode ser construído em torno das interações dos diferentes tipos de usuários com o sistema.

7.4.2.1 O Processo de Modelagem

A técnica de análise da metodologia OOSE sugere os seguintes passos:

1. **Identificação de atores**: Definição dos tipos de usuários do sistema ou papéis que os mesmos podem desempenhar em sua interação com o sistema. Os atores são externos ao sistema, fixando assim os limites do mesmo.
2. **Construção dos casos de uso**: Para cada ator identificado é construído um caso de uso do sistema que descreve um aspecto da funcionalidade do sistema iniciada pelo ator. É um modelo que tipifica o "diálogo" entre o ator e o sistema.

3. **Descrição das interfaces**: Se as interfaces no modelo de casos de uso é muito complexa, pode ser necessário descrevê-las com o propósito de validá-las com os usuários.
4. **Modelagem dos objetos do domínio do problema**: Construção de uma visão lógica do sistema usando os objetos do domínio do problema. Os objetos podem ser identificados a partir do modelo de casos de usos ou diretamente de uma descrição textual disponível dos requisitos do sistema.
5. **Refinamento do modelo de requisitos**: Revisão do modelo de requisitos para melhorar o reuso e prepará-lo para o passo seguinte. O principal refinamento é a geração de casos de uso *abstratos*. Partes comuns dos casos de uso *concretos* (gerados no passo 2) são fatorizados nestes casos de uso abstratos.
6. **Construção do modelo de análise**: Consiste na distribuição do comportamento descrito no modelo de casos de uso entre os objetos e suas associações estáticas. Estes objetos podem ser de interface (traduzem as entradas dos atores em eventos para o sistema e as operações em saídas do mesmo), entidades (objetos do domínio do problema aos quais são acrescentados atributos) e de controle (recebem comportamento que não é de interface, nem é do domínio do problema).

7.4.2.2 Os Modelos Resultantes

Os modelos resultantes da aplicação da técnica de análise da metodologia OOSE são dois:

1. **Modelo de requisitos**: Consiste em:
   - **Descrições de interfaces**: Descrições detalhadas das interfaces necessárias no modelo de casos de uso.
   - **Modelo de casos de uso (*use case model*)**: Modelo que mostra o uso do sistema por atores (papéis que os usuários podem desempenhar). Cada caso de uso descreve um fluxo completo de eventos, que é iniciado pelo ator, da perspectiva do usuário do sistema.
   - **Modelo do domínio do problema**: Modelo que descreve os objetos do sistema e suas associações de herança derivadas do domínio do problema.

2. **Modelo de análise**: Modelo derivado do modelo de casos de uso que descreve as associações de conhecimento (*acquaintance associations*) entre os três tipos de objetos (de interface, entidades e de controle).

7.4.2.3 O Comparativo

As principais equivalências entre a técnica de análise da metodologia OOSE e a proposta de modelagem são mostradas na tabela 7.2.

TABELA 7.2 - Tabela comparativa de conceitos da técnica proposta em relação à técnica de análise da metodologia OOSE.

CONCEITO	EQUIVALENTE NA TÉCNICA DE ANÁLISE DA METODOLOGIA OOSE	OBSERVAÇÕES
ação	operação	
agente externo	*ator*	mais orientado a papéis de usuários
associações estáticas	associações de conhecimento	
atributos	atributos	
concorrência nos processos	implícito no modelo de casos de uso	não é particularmente privilegiada
dicionário de dados	implícito no modelo de análise	definições descritas como associações
estímulo	*input* do sistema	
evento	evento	
composição da resposta	feita por objetos de interface	não descrita na fase de análise
decomposição do estímulo	feita por objetos de interface	não descrita na fase de análise
hierarquia de herança	baseada em atributos e operações	
identificação de objetos	de interface (interfaces de comunicação entre o usuário e o sistema), entidades (informação mantida no sistema) e de controle (para cada caso de uso)	todos derivados do modelo de casos de uso
identificador único		
integração de ciclos de vida		
limites do sistema	realizada conjuntamente com a definição dos atores	
parâmetros	implícito nos *inputs* e *outputs*	
processo	implícito no caso de uso	
resposta	*output* do sistema	
supra-sistema		

Em síntese, as principais semelhanças são as seguintes:

- **O modelo estrutural de objetos**: Ambas as técnicas descrevem os objetos e as associações estáticas derivadas dos modelos de interação de maneira semelhante.

- **Os níveis de abstração nas comunicações**: Ambas as técnicas fazem uma distinção entre as comunicações externas (*input* e estímulo, *output* e resposta) e as internas (eventos) ao sistema.
- **A definição dos limites do sistema**: Ambas as técnicas definem agentes externos ou atores, tendo como um dos propósitos o de delimitar o sistema como a linha imaginária entre estes agentes ou atores e o interior do sistema.

As principais diferenças são encontradas em:

- **O modelo fundamental**: Na técnica de análise da metodologia OOSE, o modelo fundamental é o de casos de uso, que centra-se na comunicação usuário-sistema de maneira informal. O modelo HLS é mais formal, rico em elementos de modelagem e poderoso em termos de expressividade do que o *use case model*.
- **O desenvolvimento do modelo estrutural**: Na técnica de análise de OOSE, o modelo é desenvolvido como uma combinação da modelagem de conceitos do domínio do problema e da derivação de objetos usados no modelo de casos de uso. Na técnica proposta neste trabalho, a estrutura é integralmente derivada dos modelos HLS dominuindo assim a influência de modelador na identificação dos objetos.
- **Categorização dos objetos**: Na técnica de análise OOSE, os objetos do modelo da análise são categorizados em objetos de interface, entidade e de controle. Esta distinção não é feita na técnica proposta, porque se julga ser esta uma atividade de *design*.

A figura 7.2 mostra um exemplo de aplicação do modelo de casos de uso da técnica de análise OOSE ao problema da IFIP. Trata-se da submissão de um artigo dentro deste problema.

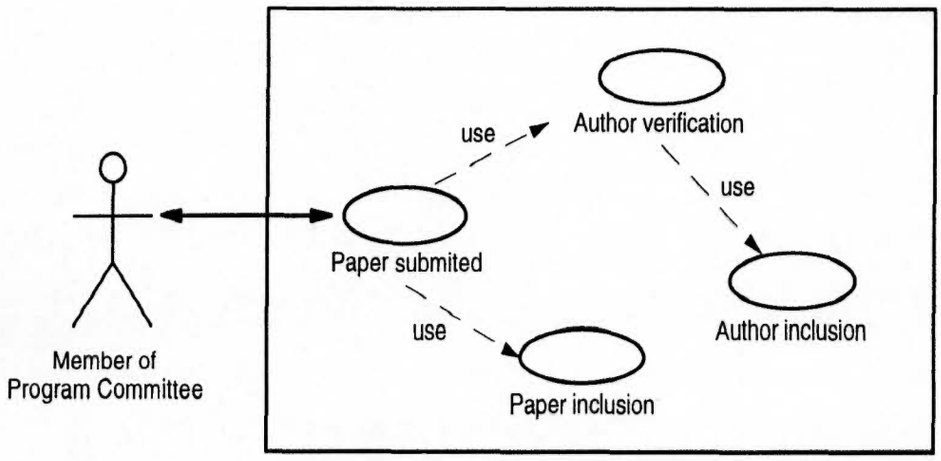


FIGURA 7.2 - Exemplo específico de um modelo de caso de uso para a submissão de um artigo no problema da IFIP, conforme a técnica de análise da metodologia OOSE.

7.5 Os Trabalhos Futuros

A partir do presente trabalho, é possível visualizar vários lineamentos para futuros desenvolvimentos, entre eles cabe destacar os seguintes:

1. **Reutilização de especificações**: Avaliar a possibilidade de considerar a tripla (estímulo, processo, resposta) como unidade de reuso a nível de especificação. Como a lista de tipos de estímulos é configurada a partir dos requisitos de comportamento do sistema sob modelagem, esta tripla, usando alguma representação mais abstrata, poderia se tornar um construtor de reuso no nível da modelagem conceitual. Estes construtores deveriam ser devidamente instanciados conforme o estímulo que se esteja modelando e posteriormente especializados a fim de satisfazer os requisitos específicos do problema.

2. **Formalização dos HLS**: Especificar formalmente todas as extensões sugeridas para os *statecharts* e que definem os denominados *High-Level Statecharts* (HLS). Neste trabalho, as extensões foram apresentadas de maneira informal e fortemente baseadas em exemplos para facilitar a compreensão.

3. **Identificação de associações padronizadas**: Estudar as formas que assumiriam os padrões de interações (estímulos, eventos e ações) nos modelos dinâmicos

para identificar associações padronizadas, tais como composição, agregação e participação (*membership*).

4. **Experimentação com a técnica de modelagem**: Aplicar o processo de modelagem a problemas de diferentes domínios, idealmente com distintos modeladores, para ganhar informação experimental que permita avaliar mais profundamente a técnica e indique modificações e extensões de caráter prático. O ideal seria realizar esta experimentação em conjunto com o desenvolvimento das ferramentas indicadas mais acima.

5. **Comparar modelos obtidos sob diferentes estratégias**: Analisar diferenças nos modelos estruturais derivados usando a técnica proposta sob a estratégia dirigida por comportamento, em relação aos modelos desenvolvidos usando técnicas de análise dirigidas por dados. A natureza das diferenças pode indicar o grau de influência do modelador na qualidade dos modelos obtidos por estas estratégias diferentes. Por exemplo, a figura 7.3 mostra um modelo de objetos desenvolvido para o mesmo problema[56] [BUS 94d] (aqui em português), usando a técnica OMT [RUM 91]. As diferenças mais notáveis com o modelo estrutural da figura 6.12 ocorrem nas hierarquias de herança.

6. **Extensões para a técnica de modelagem**: Tornar a proposta de modelagem uma técnica completa de MOO conforme as atividades definidas para ela. Algumas extensões sugeridas são:

    - incluir descrições de comportamento para aqueles tipos de objetos que não as apresentam (todos os que não foram identificados a partir das variáveis de referência do modelo de ciclos de vida);
    - incluir modelo de colaborações entre os objetos a partir da informação implícita nas associações estáticas, estímulos, eventos e ações;
    - aplicar o mecanismo de agregação, brevemente descrito no capítulo 2, para hierarquizar os objetos do modelo estrutural por níveis de abstração;

[56] Em [BUS 94f] e [BUS 94h] são mostrados outros modelos estruturais de objetos desenvolvidos com técnicas dirigidas por dados. Em [BUS 94g] é apresentado um modelo estrutural desenvolvido com uma técnica dirigida por processos. O caso modelado para todas estas aplicações é o mesmo problema da IFIP.

- flexibilizar as notações utilizadas nos diferentes modelos, visando uma maior praticidade no seu uso e
- incluir uma atividade para reorganizar o dicionário de dados para torná-lo um *dicionário de classes*, refletindo ou complementando os componentes mostrados no modelo estrutural. Particularmente necessário é a inclusão das variáveis secundárias que não aparecem no dicionário de dados inicial, mas que foram introduzidas nos modelos HLS.

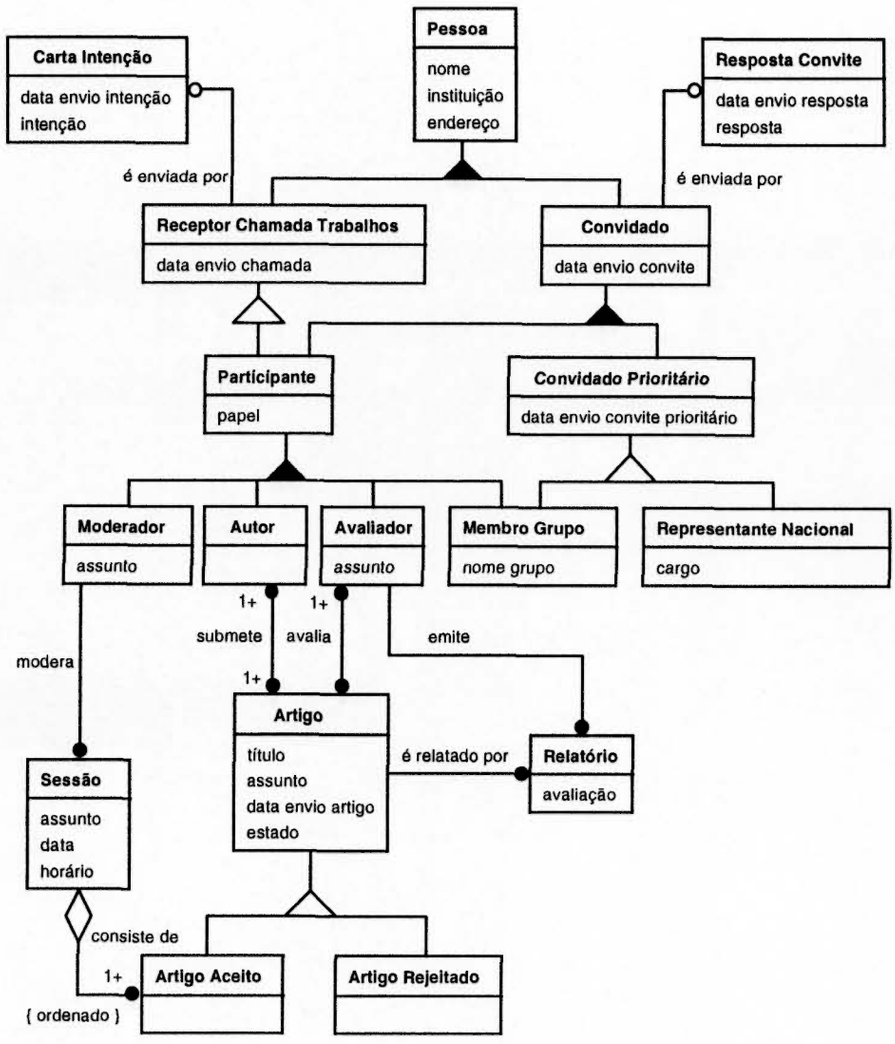


FIGURA 7.3 - Modelo estrutural de objetos para o problema da IFIP, desenvolvido segundo a técnica OMT.

7. **Ferramentas de suporte**: As ferramentas de suporte necessárias para criar um ambiente automatizado de suporte à modelagem conceitual e ao *design* orientado a objetos, seriam as seguintes:

   - **Editor de diagramas HLS**: Um editor desta natureza poderia servir de suporte tanto para modelar os processos, como para modelar os ciclos de vida. Com pequenas extensões poderia até permitir editar os diagramas de contexto do sistema e da visão global do sistema.
   - **Simulador de HLS**: Este simulador permitiria a verificação dos modelos ao instanciar as variáveis e executar o modelo.
   - **Ferramenta de suporte à integração do modelo de processos no modelo de ciclos de vida**: Como existe o problema de sinônimos e homônimos para variáveis e estados nos diagramas, esta ferramenta de integração deveria fornecer informação sobre as possíveis similitudes e desigualdades de conceitos para o modelador decidir caso a caso a ações aplicáveis.
   - **Gerador de modelos estruturais de objetos**: Para construir este gerador, seria necessário definir as heurísticas para identificar e mapear adequadamente todos os elementos considerados no modelo estrutural.
   - **Gerador de modelos de colaboração entre os objetos**: Aproveitando a informação já contida nos modelos dinâmicos e usando heurísticas que devem ainda ser desenvolvidas, seria possível obter automatizadamente as colaborações entre os objetos do sistema.
   - **Gerenciador do repositório**: Este repositório seria útil para manter, no dicionário de dados, todas as variáveis definidas no sistema. Também poderia servir para definir funções de mapeamento.

7.6 Resumo

O presente capítulo apresentou as conclusões e algumas considerações finais em relação à proposta de modelagem descrita neste trabalho. Para isto foi verificado que os objetivos levantados no início do trabalho foram atingidos, principalmente no que se refere à modelagem dinâmica do sistema e postergação do encapsulamento.

Foram descritas as contribuições do presente trabalho. Entre as contribuições gerais estão o amadurecimento da MOO sob a estratégia dirigida por comportamento, a mudança da ênfase da estrutura para o comportamento e a derivação deste comportamento a partir da estrutura. Entre as particularidades, cabe destacar o enfoque de estado usado para identificar os objetos, o tratamento e geração concorrente de estímulos e respostas, respectivamente, e a derivação das hierarquias de herança a partir do comportamento dos objetos.

Algumas técnicas de MOO dirigidas por comportamento foram descritas e comparadas com a proposta. Em síntese, a técnica OBA e a técnica de análise OOSE mostram mais diferenças do que semelhanças, apesar de conduzirem o processo de modelagem com a mesma idéia central (de que o comportamento é anterior à estrutura) e a mesma estratégia.

Os trabalhos futuros mais relevantes são o desenvolvimento de ferramentas para um ambiente de modelagem com a técnica proposta, a formalização dos HLS, a necessidade de uma maior experimentação com a técnica, a comparação de modelos estruturais gerados com estratégias diferentes e incluir as extensões necessárias para tornar a técnica uma MOO completa.

ANEXO 1 DEFINIÇÃO DO PROBLEMA DA IFIP

O seguinte é o texto original em inglês da definição do problema da IFIP como aparece em [OLL 82]:

"**IFIP**

COMPARATIVE REVIEW OF INFORMATION SYSTEMS DESIGN METHODOLOGIES

Problem Definition

1. Background

An IFIP Working Conference is an international conference intended to bring together experts from all IFIP countries to discuss some technical topic of specific interest to one or more IFIP Working Groups. The usual procedure, and that to be considered for the present purposes, is an invited conference which is not open to everyone. For such conferences it is something of a problem to ensure that members of the involved IFIP Working Group(s) and Technical Committee(s) are invited even if they do not come. Furthermore, it is important to ensure that sufficient people attend the conference so that the financial break-even point is reached without exceeding the maximum dictated by the facilities available.

IFIP Policy on Working Conferences suggest the appointment of a Program Committee to deal with the technical content of the conference and an Organising Committee to handle financial matters, local arrangements, and invitations and/or publicity. These committees clearly need to work together closely and have a need for common information and to keep their recorded information consistent and up to date.

2. Information system to be designed

The information system which is to be designed is that necessary to support the activities of both a Program Committee and an Organising Committee involved in arranging an IFIP Working Conference. The involvement of the two committees is seen as analogous to two organisational entities within a corporate structure using some common information.

The following activities of the committees should be supported.

Program Committee:

1. Preparing a list to whom the call of papers is to be sent.
2. Registering the letters of intent received in response to the call.
3. Registering the contributed papers on receipt.
4. Distributing the papers among those undertaking the refereeing.
5. Collecting the referees' reports and selecting the papers for inclusion in the program.
6. Grouping selected papers into sessions for presentation and selecting chairman for each session.

Organising Committee:

1. Preparing a list of people to invite to the conference.
2. Issuing priority invitations to National Representatives, Working Group members and members of associated working groups.
3. Ensuring all authors of each selected paper receive an invitation.
4. Ensuring authors of rejected papers receive an invitation.
5. Avoiding sending duplicate invitations to any individual.
6. Registering acceptance of invitations.
7. Generating finallist of attendees.

3. Boundaries of system

It should be noted that budgeting and financial aspects of the Organising Committee's work, meeting plans of both committees, hotel accommodation for attendees and the matter of preparing camera ready copy of the proceedings have been omitted from this exercise, although a submission may include some or all of these extra aspects if the authors feel so motivated."

ANEXO 2 NOTAÇÃO DOS HLS

A base é a notação dos *statecharts*. As seguintes são as notações para as extensões propostas:

1. **Variáveis**: Tipos disponíveis:
   - **identificadores únicos e variáveis escalares**: nomes constituídos por frases com palavras em minúscula e separadas pelo símbolo de sublinhado ("_"). Exemplo: nome_da_variável.
   - **variáveis compostas**: nomes constituídos por frases com palavras em minúscula e separadas pelo símbolo de sublinhado ("_") mais o sufixo _data. Exemplo: nome_data.
   - **listas**: lista de variáveis componentes do mesmo tipo (identificadores únicos, variáveis escalares ou compostas) anotadas entre chaves { }. Exemplo: {variável}.

2. **Estados**: Nomes constituídos por frases com palavras que começam com maiúscula e separadas pelo símbolo de sublinhado ("_"). Exemplo: Nome_Do_Estado.

   Notações adicionais para:
   - **estados atômicos (sem variáveis de referência)**: nomes na forma verbal do particípio passado. Exemplos: Incluído, Avaliado.
   - **superestados com variáveis de referência associadas**: a variável de referência (um identificador único) é anotada entre parênteses ( ). Exemplo: Superestado(variável).
   - **superestados com variáveis de referência sinônimas**: as variáveis sinônimas são anotadas entre colchetes [ ] e separadas pelo símbolo de barra vertical ("|"). Exemplo: Superestado([variável_1 | variável_2 | ... | variável_n]).
   - **conjuntos de estados exclusivos**: anotados entre colchetes [ ]. Exemplo: [Estado(variável)].

3. **Estímulo**: Frase com palavras em minúscula e separadas pelo símbolo de sublinhado ("_"). Usa-se preferentemente formas verbais no infinitivo, seguidas ou não por objetos. Exemplos: nome_do_estímulo.

   Os estímulos podem apresentar variáveis como parâmetros anotados entre parênteses ( ) e separados por vírgula (","). Exemplo: nome_do_estímulo(parâmetro_1, parâmetro_2, ..., parâmetro_n).

4. **Respostas**: Frase com palavras em minúscula e separadas pelo símbolo de sublinhado ("_"). Usa-se preferentemente formas verbais no infinitivo, seguidas ou não por objetos. Exemplo: nome_da_resposta.

   As respostas podem apresentar variáveis como parâmetros anotados entre parênteses ( ) e separados por vírgula (","). Exemplo: nome_da_resposta(parâmetro_1, parâmetro_2, ..., parâmetro_n).

   Casos particulares:

   - **resposta de conformidade simples**: acrescenta o sufixo _ok. Exemplo: nome_da_resposta_ok.
   - **resposta de conformidade composta**: usa a constante 'ok' ou a lista de constantes {'ok'} em pelo menos dois parâmetros.

5. **Processo**: Nomes constituídos por frases com palavras que começam com maiúscula e separadas pelo símbolo de sublinhado ("_"). Usa-se preferentemente formas nominais. Exemplo: Nome_Do_Processo.

6. **Transições**: Os elementos que são anotados junto às transições têm o seguinte formato:

   - **forma padrão**: todos os elementos são optativos com exceção do evento ou a condição

     evento: função_de_mapeamento [condição] / ação

   - **forma "repetida"**: todos os elementos se repetem em estados concorrentes

     { evento: função_de_mapeamento [condição] / ação }

- **ação associada a um conjunto de transições**: ação é realizada após a ocorrência de todos os eventos

  { evento: função_de_mapeamento [condição] / ação } * ação

  Casos particulares de transições:

- **inicialização de estado**: inclui condição e ação específicas:

  [variável NOT exists] / ação_de_inicialização variável

- **finalização de estado**: inclui condição e ação específicas:

  [variável exists] / ação_de_finalização variável

ou, alternativamente:

[variável exists] / eliminate variável

7. **Eventos**: Frase com palavras em minúscula e separadas pelo símbolo de sublinhado ("_"). Usa-se preferentemente formas verbais no infinitivo, seguidas ou não por objetos. Exemplos: nome_do_evento.

   Os eventos podem apresentar variáveis como parâmetros anotados entre parênteses ( ) e separados por vírgula (","). Exemplo: nome_do_evento(parâmetro_1, parâmetro_2, ..., parâmetro_n).

   Caso particular: parâmetro nulo anotado com a palavra null.

8. **Funções de Mapeamento**: Formato livre, com definição genérica (por extensão) ou específica (por intensão). Se apresentarem muita complexidade podem ser especificadas fora dos diagramas HLS, usando algum tipo de referência.

9. **Condições**: Expressão lógica. Formatos disponíveis e casos particulares:

   - **verificação de estado ativo**:

     in Superestado.Estado.Subestado...

   - **verificação de estado destino em uma transição sincronizada**:

     into Superestado.Estado.Subestado...

   - **operadores lógicos em maiúscula**: NOT, AND, OR, XOR, ...

- **operadores relacionais**: >, <, ≤, ≥, ≠, =, ...
- **valores literais não numéricos**: entre ' '. Exemplo: 'literal'

10. **Ações**: Frase com palavras em minúscula. Usa-se preferentemente formas verbais no infinitivo, seguidas ou não por objetos. Exemplo: nome da ação.

    As ações podem apresentar variáveis como parâmetros anotados entre parênteses ( ) e separados por vírgula (","). Exemplo: ação(parâmetro_1, parâmetro_2, ..., parâmetro_n).

    Caso exista mais de uma ação, elas podem ser escritas seqüencialmente (sem indicar, contudo, nenhuma ordem específica de execução) separadas por ponto e vírgula (";"). Se for preciso introduzir uma seqüência de ações, estas devem possuir anteposto, separado por um ponto, uma numeração. Exemplo: 1.ação 1; 2.ação 2; ...; n.ação n.

    Casos particulares:

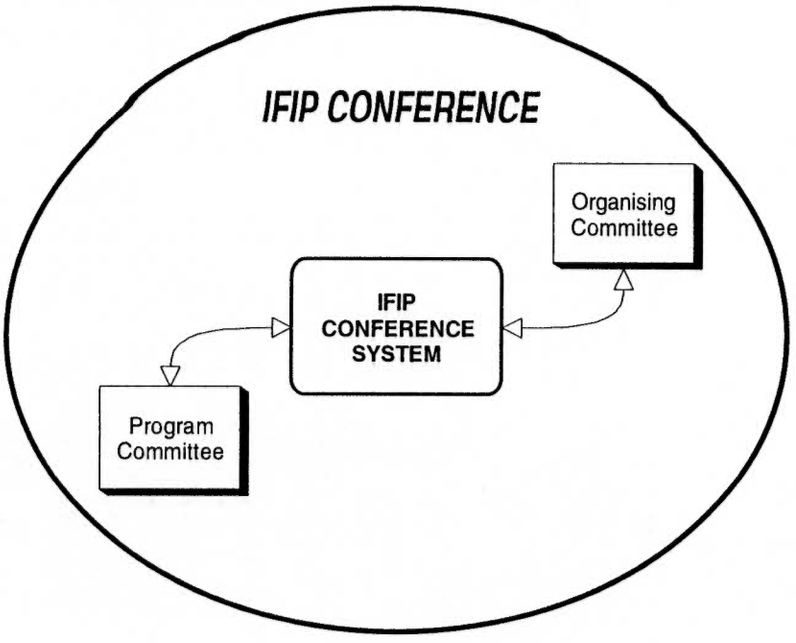
    - **ação de conformidade simples**: acrescenta o sufixo _ok. Exemplo: nome_da_ação_ok.
    - **uso do parâmetro nulo**: anotado com a palavra null.
    - **ação de conformidade composta**: usa a constante 'ok' como parâmetro.

Observação: Para todas aquelas notações que usam alguma forma de repetição entre chaves { }, é permitida a inclusão de limites numéricos mínimo e máximo que indicam o mínimo e o máximo de vezes que o elemento está repetido. Estes limites mínimo e máximo são anotados antes e após as chaves, respectivamente. Exemplo: 1{ elemento }3

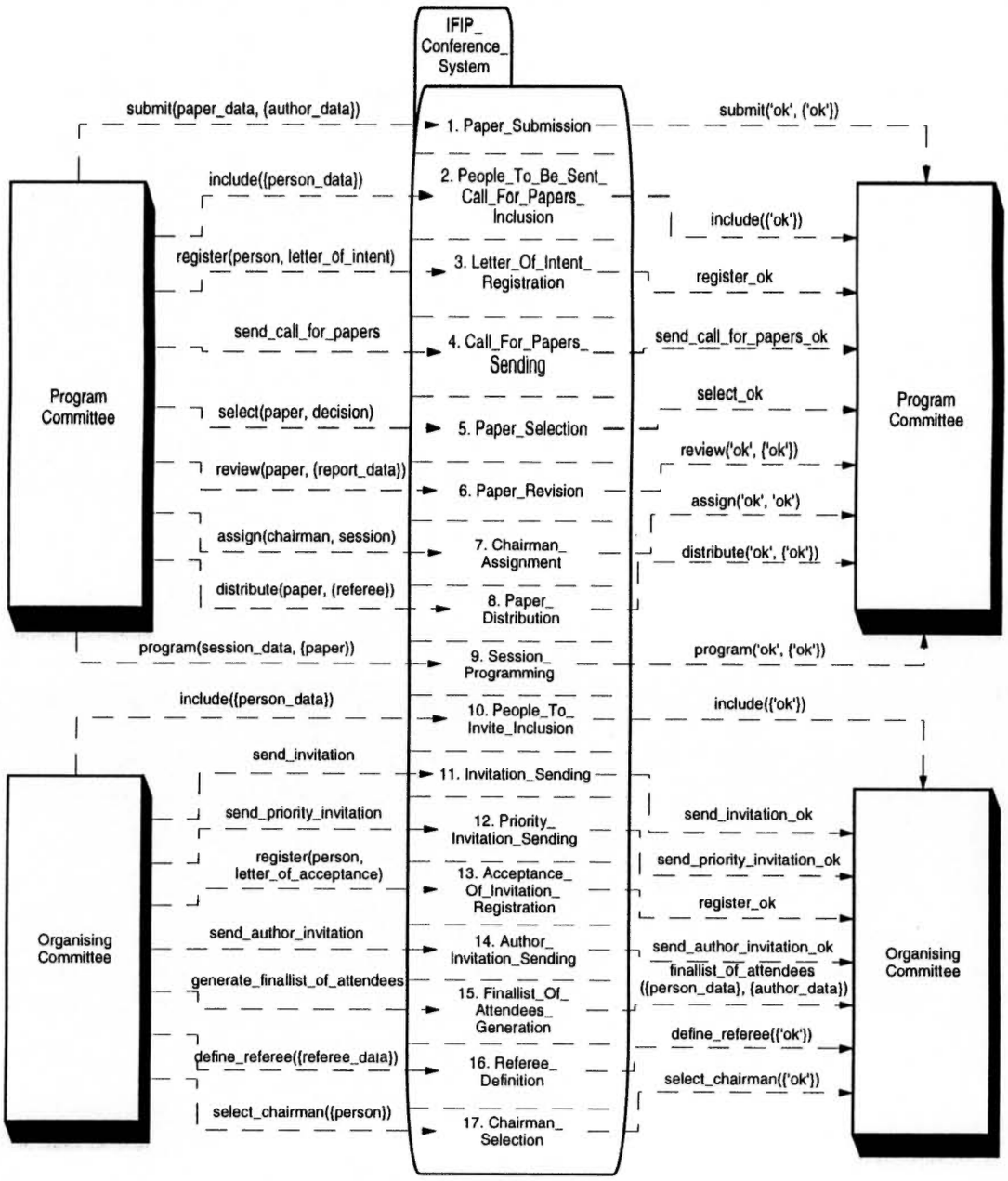
ANEXO 3 MODELO COMPLETO DO PROBLEMA DA IFIP

1 Modelo da Visão Global

1.1 Diagrama de Contexto do Sistema

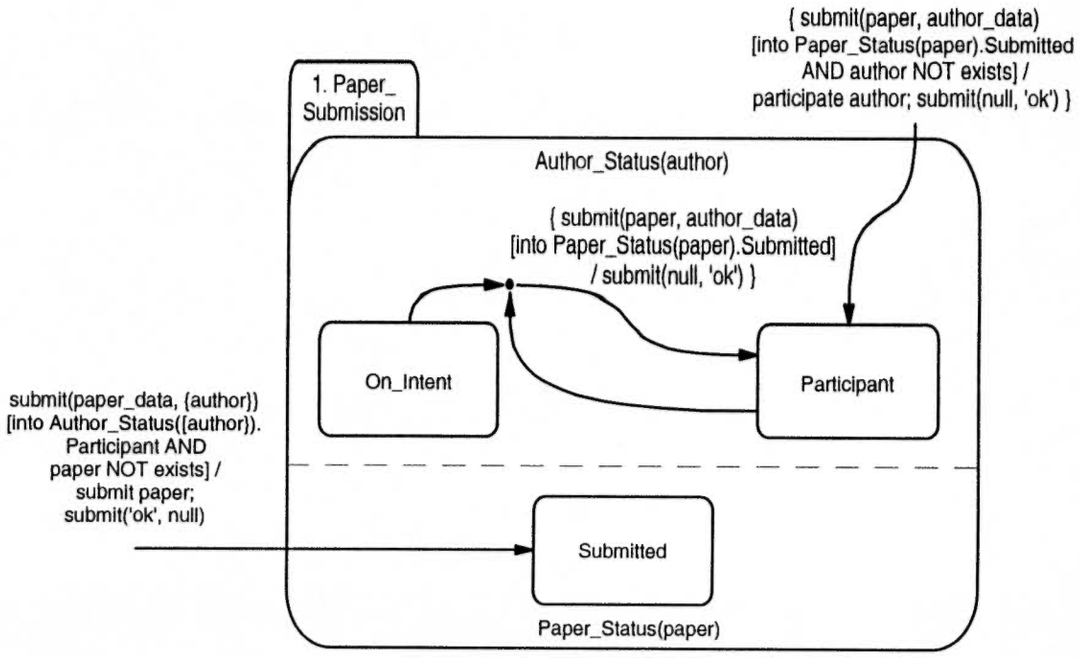


1.2 Diagrama da Visão Global do Sistema



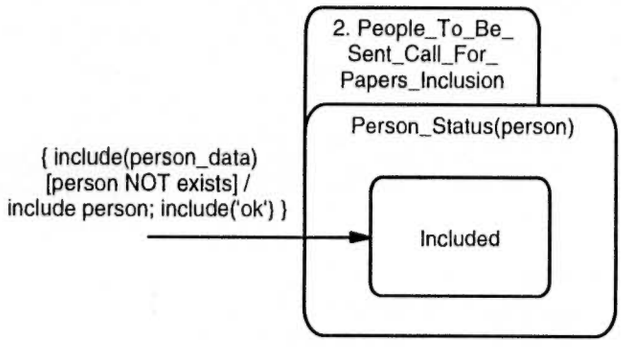
2 Modelo de Processos

2.1 Diagrama HLS do processo "Paper_Submission"



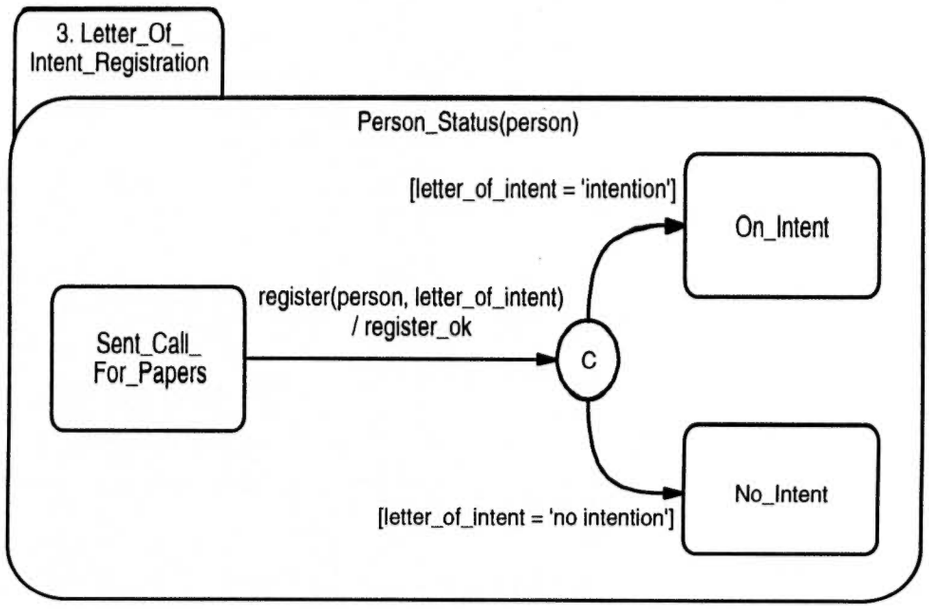
Requisito modelado: "*Registering the contributed papers on receipt.*" [OLL 82].

2.2 Diagrama HLS do processo "People_To_Be_Sent_Call_For_Papers_Inclusion"



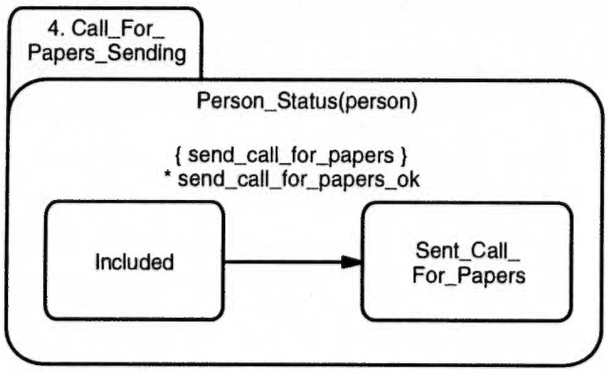
Requisito modelado: " *Preparing a list to whom the call for papers is to be sent.* " [OLL 82].

2.3 Diagrama HLS do processo "Letter_Of_Intent_Registration"



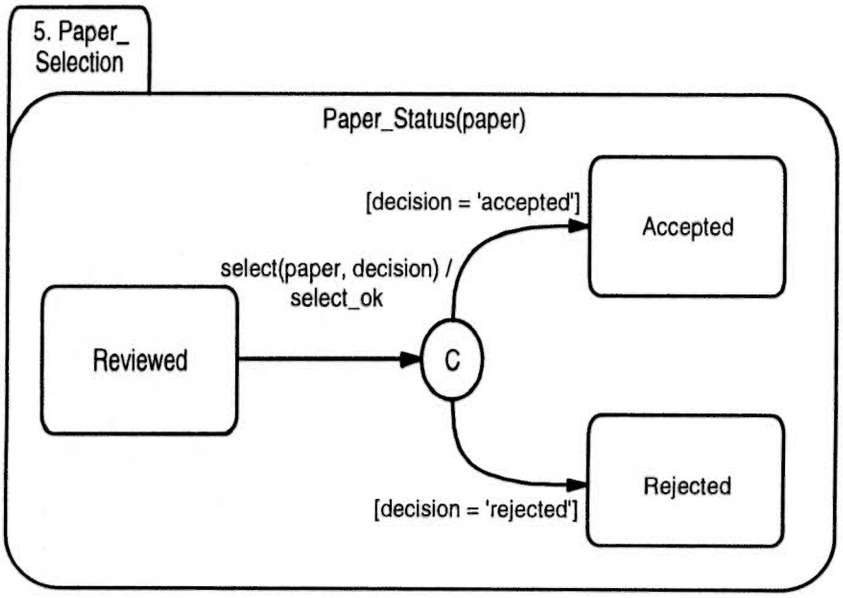
Requisito modelado: " *Registering the letters of intent received in response to the call.* " [OLL 82].

2.4 Diagrama HLS do processo "Call_For_Papers_Sending"



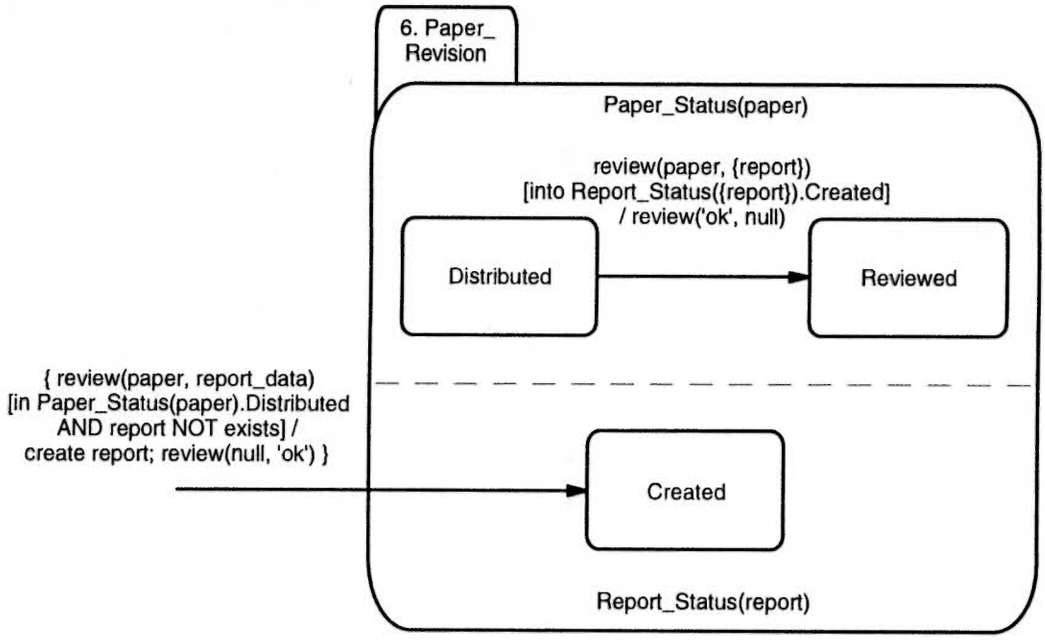
Requisito modelado: " *Preparing a list to whom the call for papers is to be sent.* " [OLL 82].

2.5 Diagrama HLS do processo "Paper_Selection"



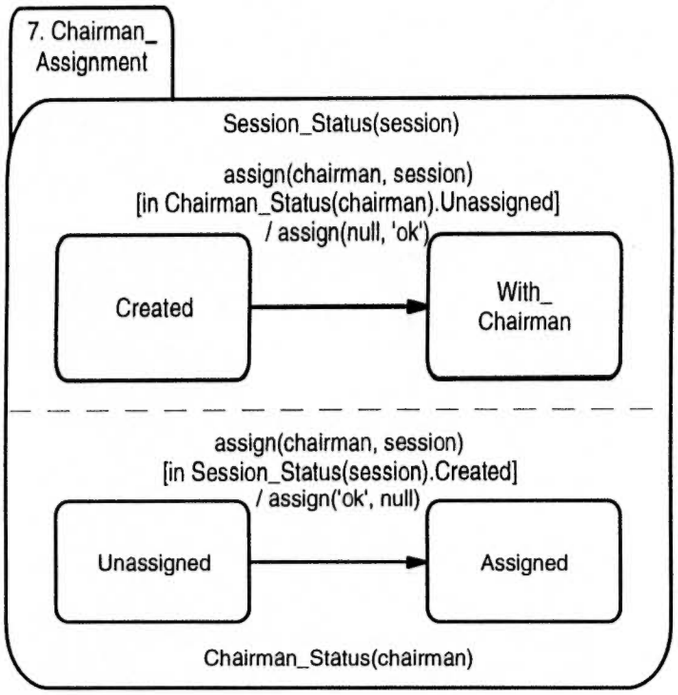
Requisito modelado: "*...selecting the papers for inclusion in the program.*" [OLL 82].

2.6 Diagrama HLS do processo "Paper_Revision"



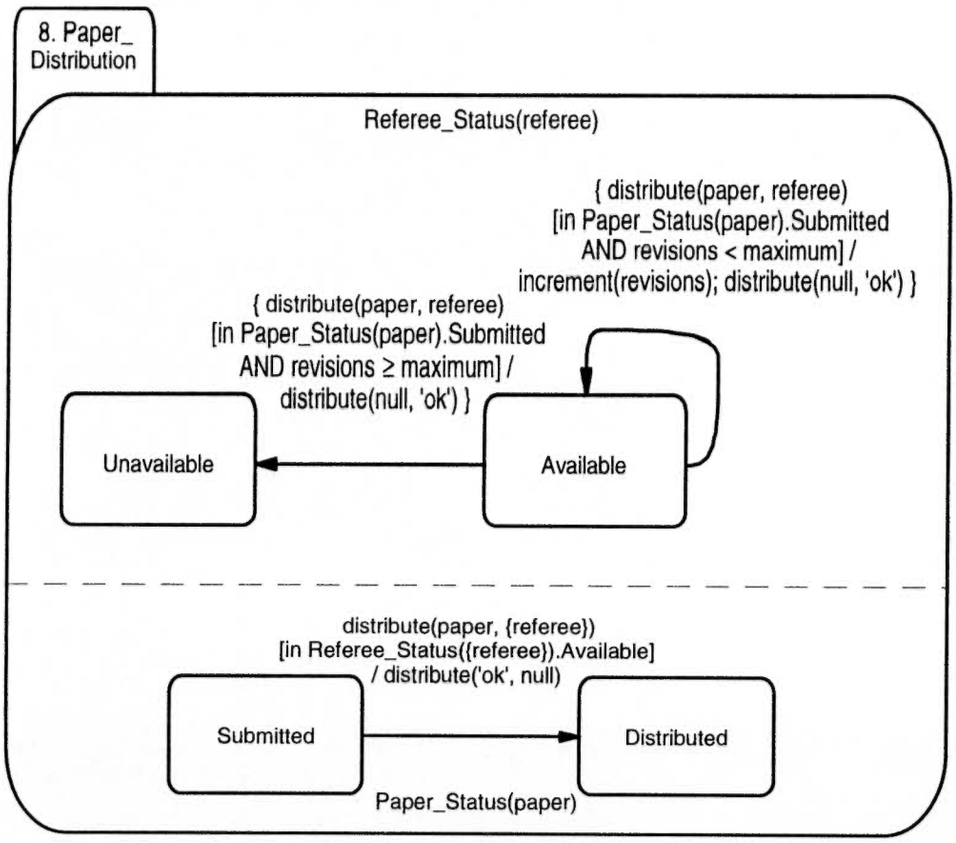
Requisito modelado: "*Collecting the referees' reports ...*" [OLL 82].

2.7 Diagrama HLS do processo "Chairman_Assignment"



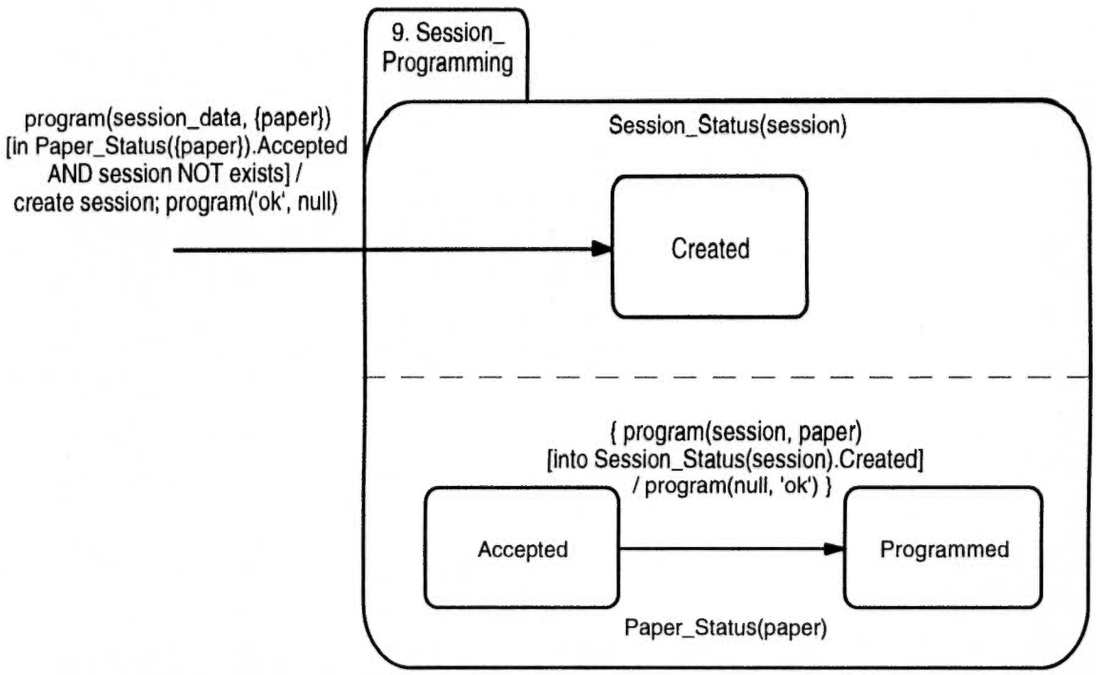
Requisito modelado: "*...and selecting chairman for each session*." [OLL 82].

2.8 Diagrama HLS do processo "Paper_Distribution"



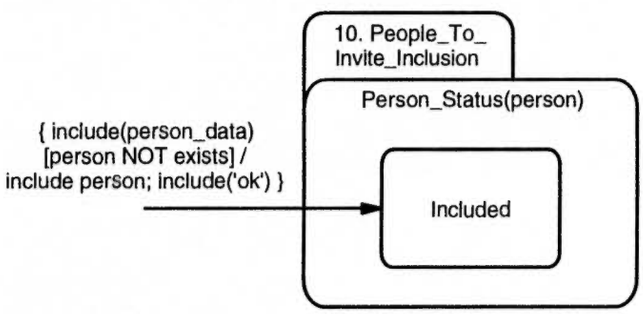
Requisito modelado: "*Distributing the papers among those undertaking the refereeing*." [OLL 82].

2.9 Diagrama HLS do processo "Session_Programming"



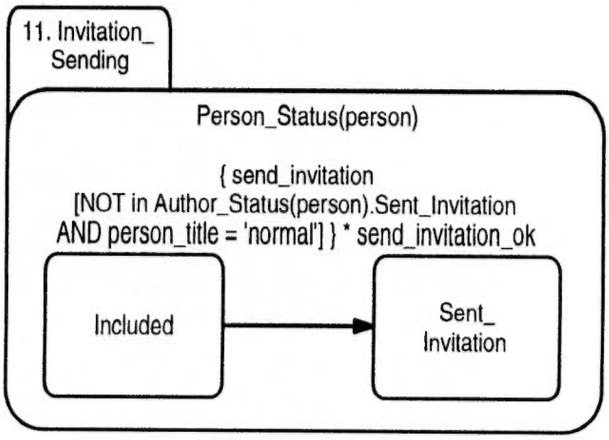
Requisito modelado: "*Grouping selected papers into session for presentation..*" [OLL 82].

2.10 Diagrama HLS do processo "People_To_Invite_Inclusion"



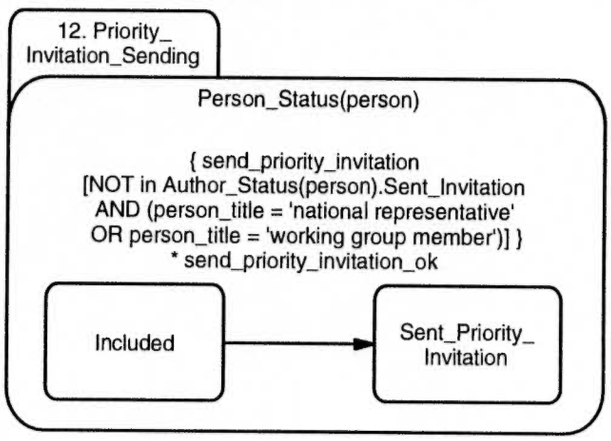
Requisito modelado: " *Preparing a list of people to invite to the conference.* " [OLL 82].

2.11 Diagrama HLS do processo "Invitation_Sending"



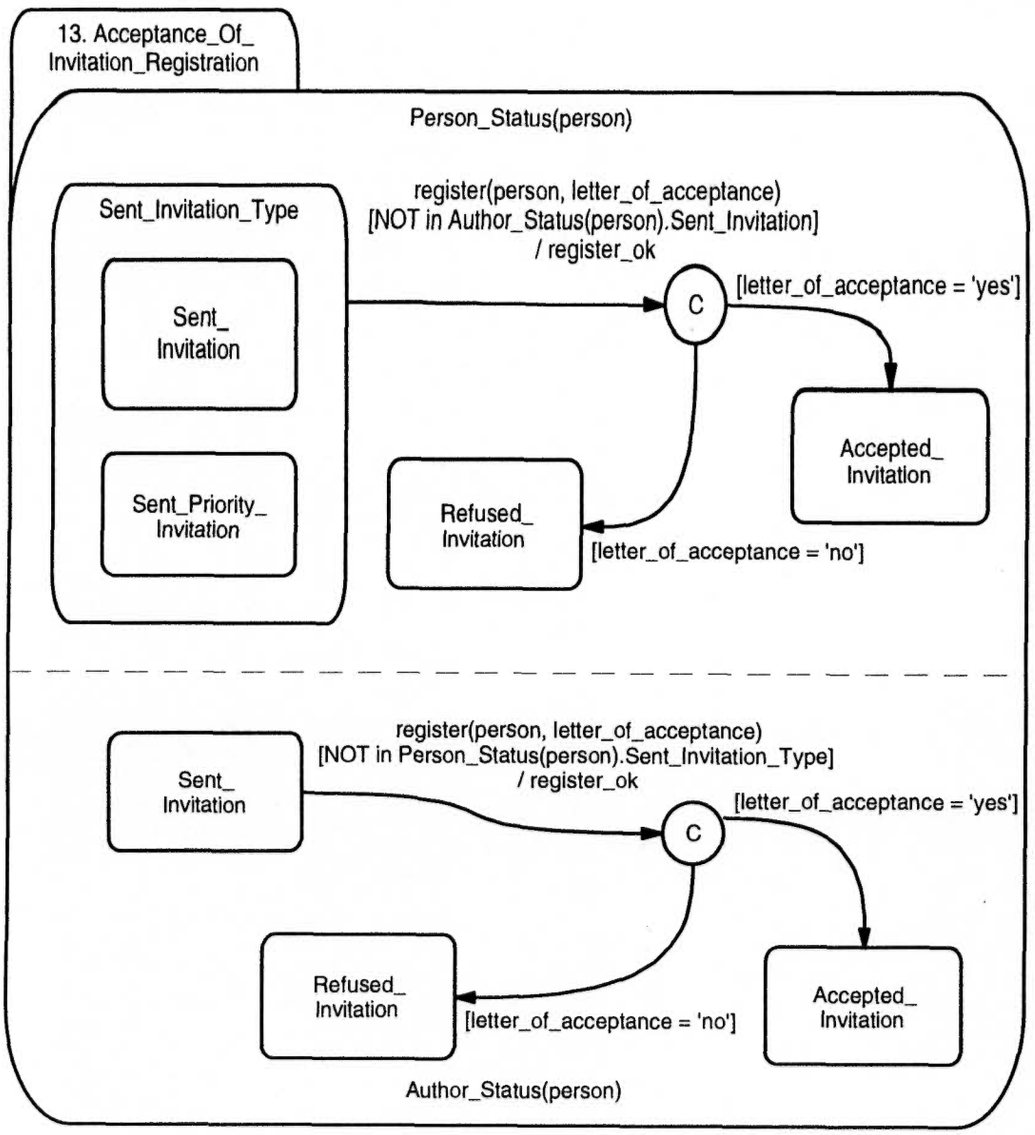
Requisito modelado: " *Preparing a list of people to invite to the conference. Avoiding sending duplicate invitations to any individual.* " [OLL 82].

2.12 Diagrama HLS do processo "Priority_Invitation_Sending"



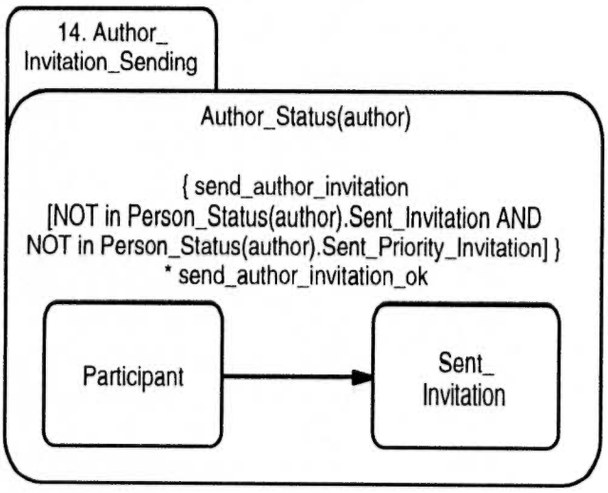
Requisito modelado: "*Issuing priority invitations to National Representatives, Working Group members and members of associated working groups. Avoiding sending duplicate invitations to any individual*".[OLL 82].

2.13 Diagrama HLS do processo "Acceptance_Of_Invitation_Registration"



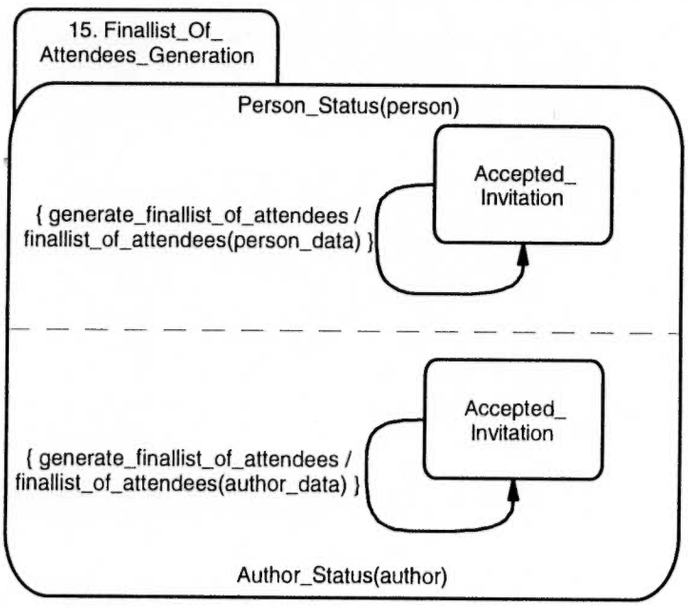
Requisito modelado: " *Registering acceptance of invitations.*" [OLL 82].

2.14 Diagrama HLS do processo "Author_Invitation_Sending"



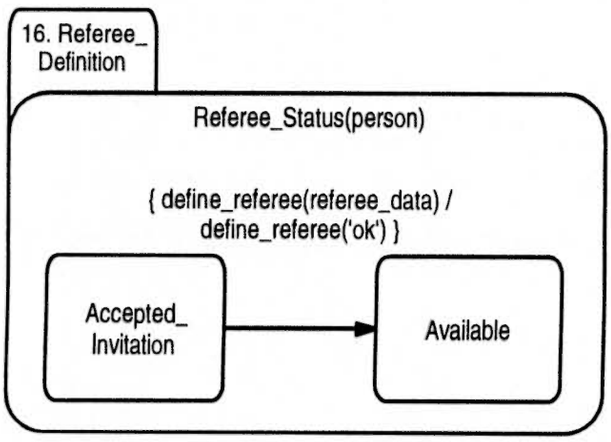
Requisito modelado: "*Ensuring all authors of each selected/rejected papers receive an invitation*." [OLL 82].

2.15 Diagrama HLS do processo "Finallist_Of_Attendees_Generation"



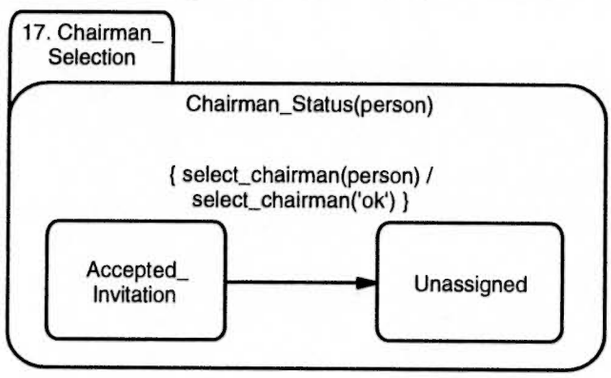
Requisito modelado: "*Generating finallist of attendees*." [OLL 82].

2.16 Diagrama HLS do processo "Referee_Definition"



Requisito modelado: " *...those undertaking the refereeing.* " [OLL 82].

2.17 Diagrama HLS do processo "Chairman_Selection"



Requisito modelado: " *...selecting chairman...* " [OLL 82].

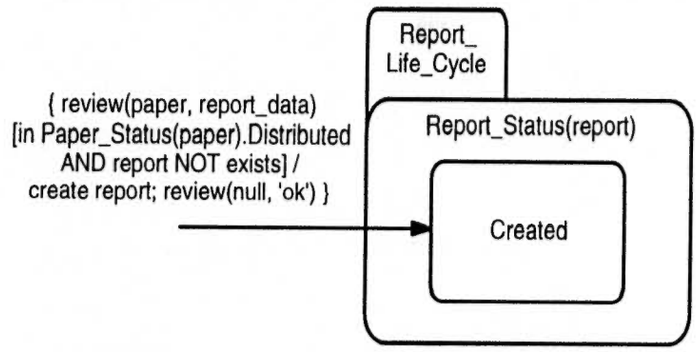
3 Dicionário de Dados

DATA DICTIONARY

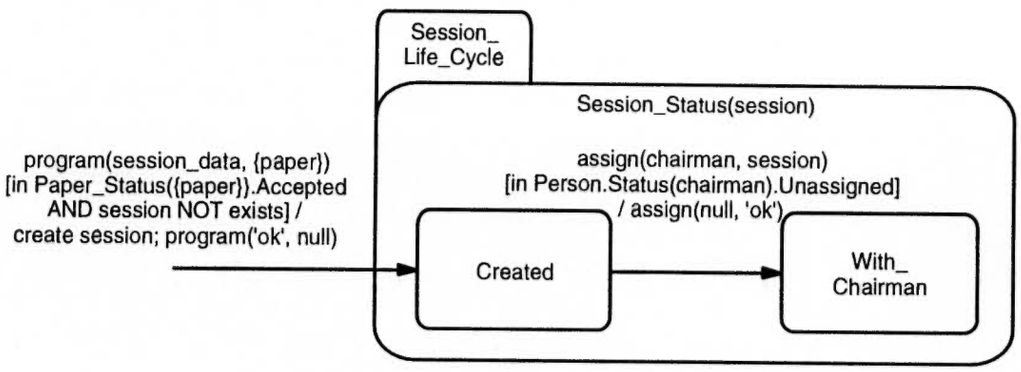
author_data = @author + name + institution + address
decision = ['accepted' | 'rejected']
letter_of_acceptance = ['yes' | 'no']
letter_of_intent = ['intention' | 'no intention']
paper_data = @paper + title + subject
person_data = @person + name + institution + person_title
referee_data = @person + maximum
report_data = @report + referee + concept
session_data = @session + room + time
person_title = ['normal' | 'national representative' | 'working group member']

4 Diagramas HLS de Ciclo de Vida

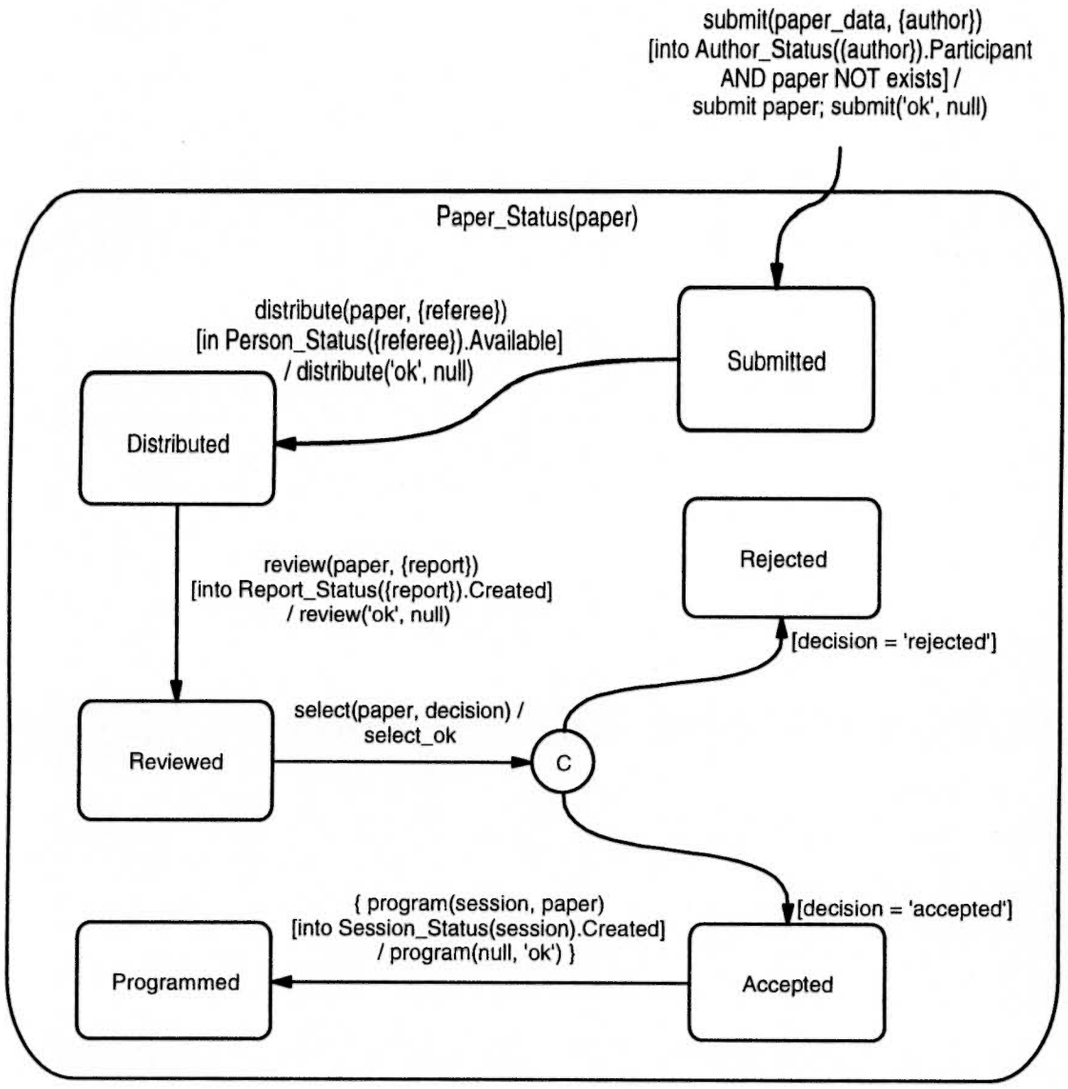
4.1 Diagrama HLS de ciclo vida da variável report



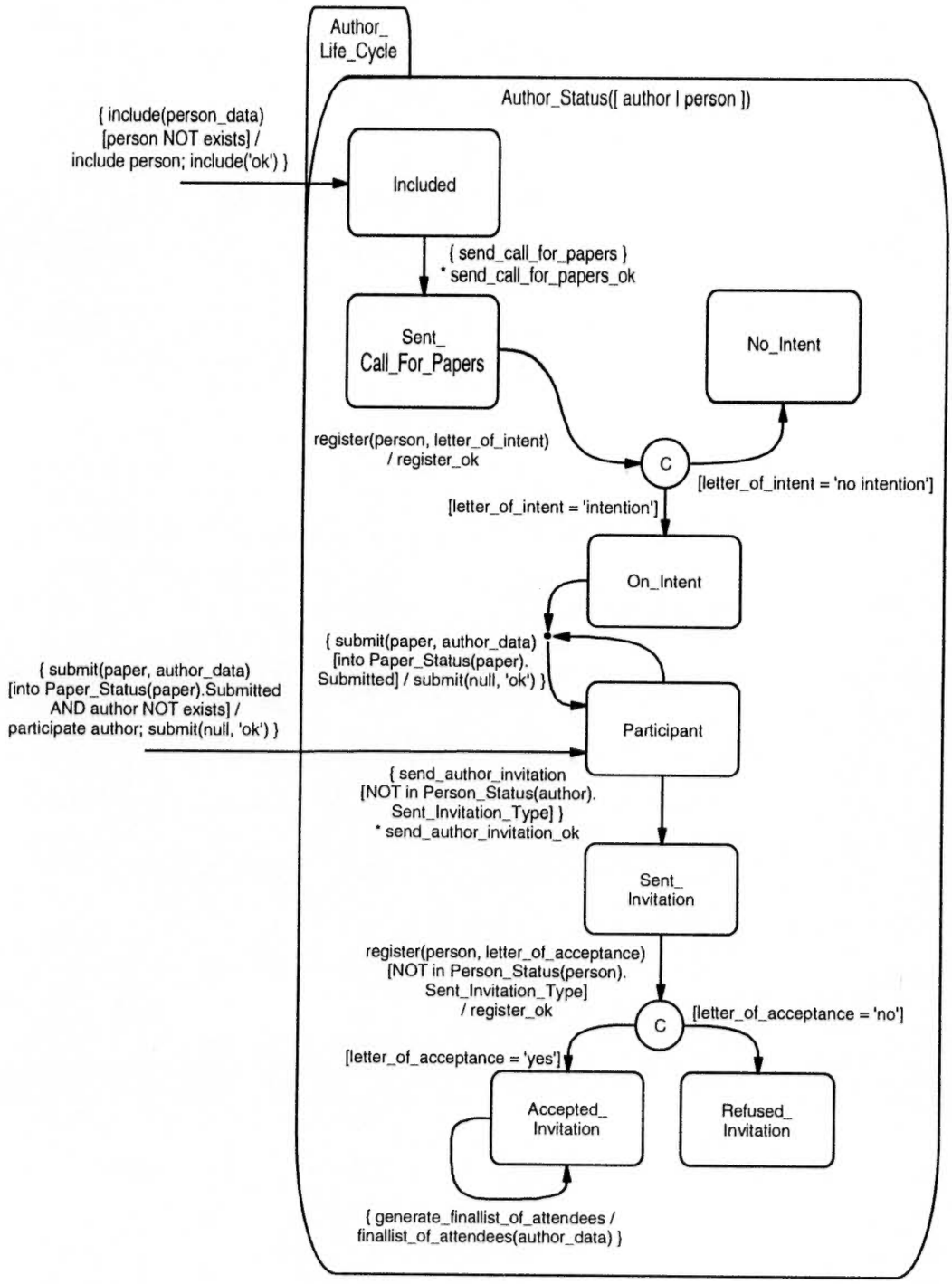
4.2 Diagrama HLS de ciclo vida da variável session



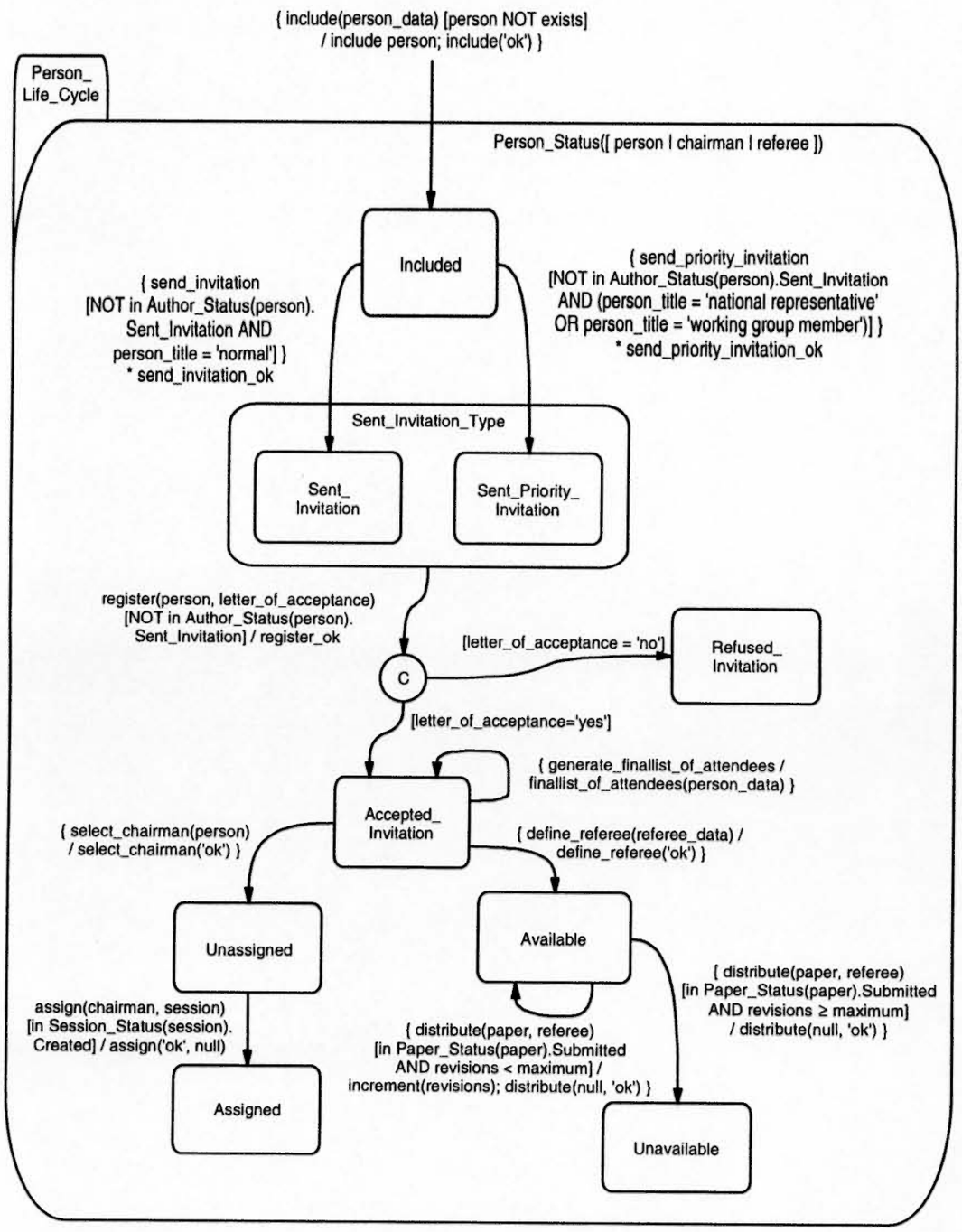
4.3 Diagrama HLS de ciclo vida da variável paper



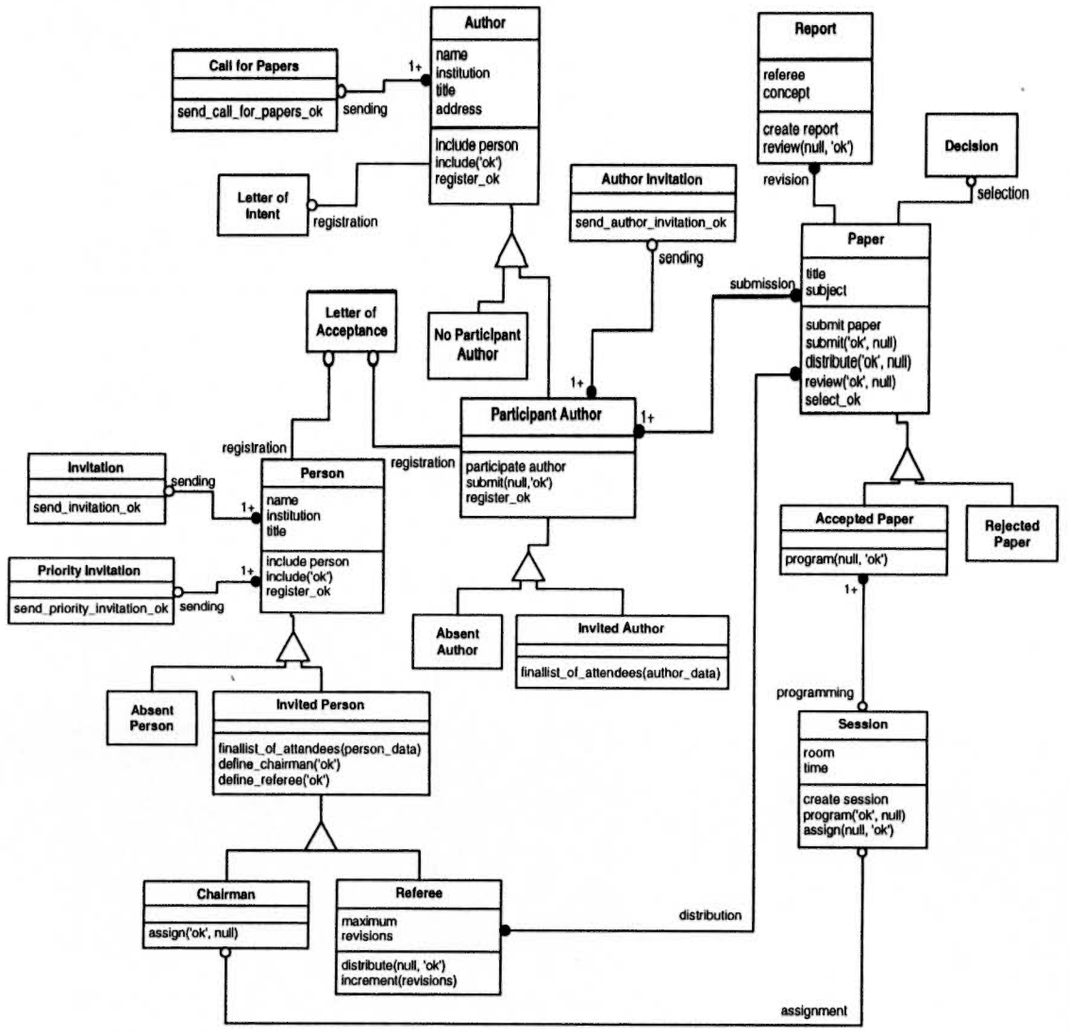
4.4 Diagrama HLS de ciclo vida da variável author/person



4.5 Diagrama HLS de ciclo vida da variável person/referee/chairman



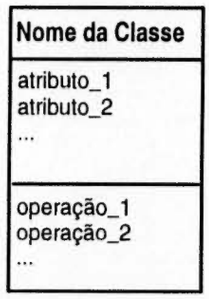
5 Modelo Estrutural Orientado a Objetos



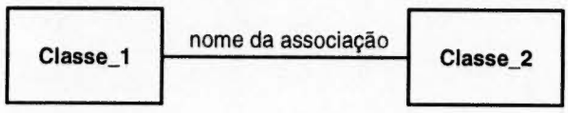
ANEXO 4 RESUMO DA NOTAÇÃO OMT PARA O MODELO DE OBJETOS

A notação utilizada para o modelo estrutural de objetos é a proposta por Rumbaugh et al. [RUM 91] na técnica OMT (*Object Modeling Technique*). As representações usadas são descritas a seguir.

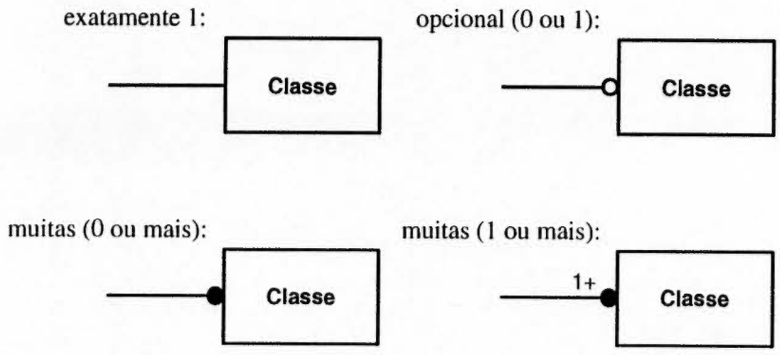
Classe:



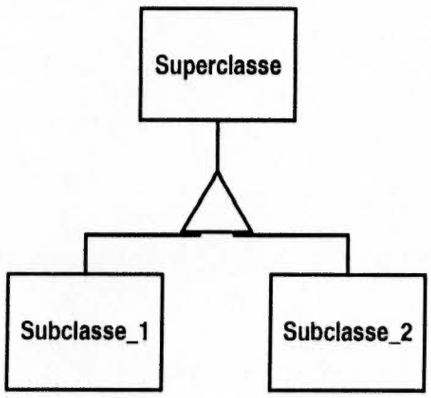
Associação entre classes:



Cardinalidade das associações:



Hierarquia de herança (sem sobreposição):



GLOSSÁRIO

- A -

ação: uma atividade realizada durante uma transição, que pode ser uma operação específica dentro do processo, um evento para outra transição ou pode compor a resposta do processo.

ação de conformidade: ação cuja única função é indicar que a transição ou o processo foi realizado com total satisfação. As ações de conformidade podem ser simples ou compostas.

ação de conformidade composta: ação de conformidade que apresenta um ou mais parâmetros constantes 'ok'. A ordem posicional destes parâmetros é importante, porque indica que tratamento para o parâmetro do estímulo que está nessa posição foi realizado em conformidade.

ação de conformidade simples: ação de conformidade que não apresenta parâmetros, apenas é acrescentado o sufixo _ok ao nome da ação para indicar que todo o processo foi realizado em conformidade.

agente externo: qualquer outro sistema (usuário, papel de usuário, dispositivo, unidade organizacional, etc.) que venha a interagir com o sistema modelado, seja emitindo um ou mais estímulos para o mesmo, seja recebendo uma ou mais respostas emitidas pelo sistema ou ainda ambas as coisas. O agente externo pode fazer parte também do supra-sistema do sistema sob modelagem.

associação candidata: associação entre instâncias de tipos de objetos, que é identificada a partir do tipo do estímulo, do tratamento concorrente dentro do processo, do número de parâmetros do estímulo e, eventualmente, de ações associadas a conjuntos de eventos concorrentes.

auto-transição: transição que tem como origem e destino o mesmo estado.

- C -

cluster: abstração na forma de um agregado de classes que esconde detalhes de associações e atributos de um nível dado de abstração. Os atributos de um *cluster* correspondem às classes associadas no nível imediatamente inferior.

clustering: processo que permite definir *clusters* sobre um conjunto de classes associadas. Este processo é recursivo permitindo realizar *clustering* também sobre *clusters*.

condição: expressão lógica cujo valor_verdade determina o disparo da transição quando o evento desta transição ocorre. Se o seu valor_verdade for verdadeiro, a transição ocorre, se for falso, a transição não ocorre.

condição de consistência: condição que verifica a consistência nos dados e nos estados de processo e do sistema. As condições de consistência podem ser intraprocesso ou interprocessos.

condição de consistência interprocessos: condição de consistência que relaciona dois ou mais processos diferentes do sistema.

condição de consistência intraprocesso: condição de consistência que relaciona dois ou mais subestados concorrentes pertencentes a um mesmo processo do sistema.

- D -

desigualdade de conceitos: ocorre quando dados ou conceitos (representados por variáveis de referência) com o mesmo nome possuem propriedades (papéis, contextos temporais, semântica) diferentes. Esta desigualdade de conceitos se expressa por meio de homônimos.

diagrama de contexto do sistema: diagrama que mostra o sistema em um contexto composto por um supra-sistema que contém o sistema sob modelagem e os agentes externos relacionados com o sistema por estímulos e respostas.

diagrama de visão global: diagrama que mostra o sistema sob modelagem como um conjunto de tuplas (emissor, estímulo, processo, resposta, receptor) representadas em forma concorrente. Cada processo é concebido como uma caixa preta, e seu comportamento é descrito nos diagramas HLS de processo.

diagrama HLS de ciclo de vida: aquele que representa a "história" de um tipo de dado expresso pela variável de referência através de sucessivos estados. O mesmo que **diagrama HLS definitivo de ciclo de vida**.

diagrama HLS de processo: diagrama de um processo específico do sistema que mostra sua estrutura em termos de estados hierarquizados, transições, eventos, funções de mapeamento, ações e variáveis de referência. Existe um diagrama HLS de processo por cada processo mostrado no diagrama da visão global.

diagrama HLS definitivo de ciclo de vida: diagrama HLS preliminar de ciclo de vida no qual foi verificada a consistência nos nomes dos estados e variáveis de referência utilizadas.

diagrama HLS preliminar de ciclo de vida: diagrama resultante da integração de diagramas HLS de processo com estados e variáveis comuns. Esta integração é realizada conectando-se pós-estados e pré-estados dos diferentes diagramas. Este diagrama deve ainda ser submetido a uma verificação para tornar-se um diagrama HLS definitivo de ciclo de vida.

- E -

emissor: agente externo que desempenha o papel específico de somente emitir estímulos para o sistema.

estado ativo: refere-se a uma situação na qual o estado está envolvido em alguma atividade durante a permanência no estado. Este tipo de estados não é usado como estado atômico nos HLS. Não deve ser confundido com a ativação de estados, isto é, quando um estado pode ou não *estar* ativo em um momento dado. O estado ativo com ações que podem ser interrompidas pode ser emulado com um estado passivo e uma auto-transição.

estado atômico: estado que aparece no último nível da hierarquia de estados, indicando com isto que não é mais decomponível e constitui a base fundamental para todos os estados abstratos (superestados) construídos sobre ele.

estado passivo: estado atômico, nos HLS, durante o qual não ocorre nenhuma atividade, ação ou operação. Durante a permanência neste tipo de estado, nada muda, com exceção do tempo. Os nomes destes estados são formas verbais geralmente no particípio passado.

estados concorrentes: dois ou mais estados são designados concorrentes se podem estar simultaneamente ativos.

estados exclusivos: dois ou mais estados são designados exclusivos se um, e somente um dentre eles, pode estar ativo por vez.

estímulo: qualquer tipo de comunicação que algum emissor envia para o sistema. Os estímulos podem ser categorizados como construtivos e de consulta; com dados principais, com dados secundários e de transição; e de controle e temporais.

estímulo com dados principais: estímulo que apresenta um ou mais dados (na forma de parâmetros do mesmo) cuja entrada no sistema é a que justifica a existência do estímulo. Estes dados devem ser tratados por um processo específico do sistema.

estímulo com dados secundários: estímulo que apresenta um ou mais dados (na forma de parâmetros do mesmo). Este estímulo ocorre sob explícita solicitação do sistema através de uma resposta gerada por um outro processo, isto é, este estímulo depende da ocorrência de um outro estímulo (com dados principais ou secundários) que foi anteriormente tratado por um processo e cuja resposta pede por aquele.

estímulo construtivo: aquele que "constrói" o sistema, no sentido de conter informação para estruturar os dados no interior do mesmo ao fornecer dados de entrada que farão parte de sua memória. Este tipo de estímulo modifica o comportamento do sistema porque produz transições de estado no seu interior.

estímulo de consulta: estímulo que apenas solicita dados registrados na memória do sistema, ou, em outra palavras, é um requisito de recuperação de informação mantida pelo sistema. Este tipo de estímulo não produz mudanças de comportamento no sistema porque não provoca transições de estados que não sejam auto-transições.

estímulo de controle: estímulo que pode ocorrer em qualquer instante do tempo, sem uma periodicidade estabelecida. Geralmente, é um estímulo que solicita alguma informação ou comunica algum fato eventual que deva ser registrado. Pode ou não apresentar parâmetros.

estímulo de transição: estímulo que não possui parâmetros de nenhum tipo e que apenas produz transições de estado dentro do sistema. Estes estímulos modificam o comportamento do sistema sem fornecer dados.

estímulo temporal: estímulo que está associado a alguma medida de tempo. Indica o passar do tempo, como um relógio ou calendário do sistema.

estratégia *bottom-up*: forma de construção de modelos de qualquer natureza através de um processo de agregação sucessiva, em que cada nível de agregação acrescenta maior abstração ao modelo, escondendo detalhes que eram considerados no nível de agregação imediatamente inferior. Nesta estratégia, o sentido de construção do modelo vai dos níveis mais detalhados (*bottom*) aos mais abstratos (*up*) em uma hierarquia de abstração.

estratégia *middle-out*: forma de construção de modelos de qualquer natureza através de um processo "linear" sem refinamento nem agregação de nenhuma espécie, isto é, a construção do modelo não segue o sentido vertical (vide as estratégias *top-down* e *bottom-up*) entre os níveis de uma hierarquia de abstração, senão o sentido horizontal, acrescentando e relacionando novos elementos do mesmo nível de abstração e detalhe. Nesta estratégia, a construção do modelo é feita por extensão e não por hierarquização.

estratégia *top-down*: forma de construção de modelos de qualquer natureza através de um processo de refinamento sucessivo, em que cada nível de refinamento acrescenta novos detalhes ao modelo que não eram considerados no nível de refina-

mento imediatamente superior. Nesta estratégia, o sentido de construção do modelo vai dos níveis mais abstratos (*top*) aos mais detalhados (*down*) em uma hierarquia de abstração.

estrutura composta: estrutura de estados que pode assumir um processo que apresenta um conjunto de subestados expressos simultaneamente. As estruturas compostas podem ser concorrentes ou exclusivas.

estrutura composta concorrente: estrutura composta de um conjunto de estados concorrentes.

estrutura composta exclusiva: estrutura composta de um conjunto de estados exclusivos.

estrutura simples: estrutura de estados que pode assumir um processo que apresenta estados não genéricos. As estruturas simples podem ser únicas, concorrentes ou exclusivas.

estrutura simples concorrente: estrutura simples que apresenta dois ou mais subestados concorrentes.

estrutura simples exclusiva: estrutura simples que apresenta dois ou mais subestados exclusivos.

estrutura simples única: estrutura simples que apresenta um único subestado.

evento: fato relevante para um processo específico que dispara uma transição (dependendo também de uma condição se esta existe). O evento pode ser um estímulo de algum emissor ou pode ser um seu componente parcial.

- F -

finalização de estado: uma transição que não deixa nenhum estado ativo no modelo, de tal maneira que, depois de ocorrida a transição, não mais existe o estado e o diagrama para o valor específico da variável de referência do superestado.

fonte direta de objetos candidatos: são aquelas fontes que permitem identificar diretamente os objetos candidatos no modelo HLS. As fontes diretas são as variáveis de referência dos diagramas HLS de ciclo de vida e os dados usados como parâmetros nos eventos que não são variáveis de referência.

fonte indireta de objetos candidatos: são aquelas fontes que, devido a sua existência, obrigam à definição de objetos candidatos associados no modelo HLS. A principal fonte indireta são as ações definidas para um conjunto de eventos concorrentes. Estas ações devem ser alocadas junto a algum objeto, o que determina a definição de um objeto que as contenha.

função de mapeamento: função que define genérica (por extensão) ou especificamente (por intensão) o estado ao qual conduz a transição, da qual faz parte, através de um novo valor resultante de uma variável de referência. A função de mapeamento sempre é necessária se o estado destino da transição não aparece no diagrama que a contém.

- G -

guard: o mesmo que **condição**.

- H -

High-Level Statecharts: proposta de extensão dos *statecharts* de Harel [HAR 87], desenvolvida com o propósito de permitir modelar adequadamente sistemas de informação. As principais adições são a de tratamento de conjuntos através de estados "parametrizados" e a definição de estruturas de conjuntos de estados concorrentes e exclusivos. A notação dos *statecharts* é mantida em sua maior parte (vide anexo "Notação dos HLS").

homônimos: dois ou mais conceitos (variáveis de referência) que possuem um mesmo nome. Expressa uma desigualdade de conceitos.

- I -

identificador: o mesmo que **identificador único**.

identificador único: variável que serve para identificar unicamente um determinado dado composto. O identificador único é representado no dicionário de dados antepondo o símbolo @ e nunca possui o sufixo _data.

inicialização de estado: transição que ativa um estado no modelo, de tal forma que, antes de ocorrida a transição, não existia o estado para o valor específico da variável de referência do superestado.

- M -

modelo de ciclos de vida: conjunto de diagramas HLS definitivos de ciclo de vida do sistema.

modelo de processos: conjunto de diagramas HLS de todos os processos que aparecem no diagrama da visão global.

modelo da visão global do sistema: modelo composto pelo diagrama de contexto do sistema e pelo diagrama da visão global do mesmo.

modelo HLS: conjunto de diagramas de contexto do sistema, da visão global, HLS dos processos e HLS dos ciclos de vida.

- O -

objeto candidato: tipo de objeto identificado a partir das fontes diretas ou indiretas. Este objeto pode participar de associações, fazer parte do modelo estrutural de objetos e, eventualmente, compor hierarquias de herança.

- P -

parâmetro: dados que acompanham ou eventualmente justificam o estímulo e que servem de entrada para o processo que trata particularmente este estímulo. Os parâmetros também aparecem nos eventos como uma derivação dos parâmetros do estímulo. No caso dos parâmetros das respostas, estes correspondem a dados de saída do processo e, eventualmente, justificam a resposta ou apenas a acompanham. Os parâmetros das ações desempenham a mesma função que os das respostas.

parâmetro nulo: parâmetro que apenas indica que não existe nenhum dado na posição do parâmetro. É representado pela palavra null.

pós-estado: estado final ou estado resultante da execução de um determinado processo, isto é, é o estado que fica ativo quando a resposta é finalmente emitida pelo processo.

pré-estado: estado inicial, ou estado necessário para a execução de um determinado processo, isto é, é o estado que deve estar ativo quando o estímulo é recebido pelo processo.

processo: componente do sistema encarregado do tratamento de um estímulo específico. Como conseqüência da execução interna deste tratamento, resulta a emissão de uma resposta também específica.

- R -

receptor: agente externo que desempenha o papel específico de somente receber respostas emitidas pelo sistema.

resposta: qualquer tipo de comunicação que o sistema emite para um receptor. As respostas podem conter parâmetros ou, simplesmente, indicar que o tratamento do estímulo pelo processo foi realizado satisfatoriamente tanto para todo estímulo (neste caso usa-se o sufixo ok na resposta), quanto para cada parâmetro do estímulo (neste caso aparece posicionalmente a constante 'ok').

resposta de conformidade: resposta cuja única função é indicar que o processo foi realizado com total satisfação. As respostas de conformidade podem ser simples ou compostas.

resposta de conformidade composta: resposta de conformidade que apresenta um ou mais parâmetros constantes 'ok'. A ordem posicional destes parâmetros é importante, porque indica que tratamento para o parâmetro do estímulo que está nessa posição foi realizado em conformidade.

resposta de conformidade simples: resposta de conformidade que não apresenta parâmetros, apenas é acrescentado o sufixo '_ok' ao nome da resposta para indicar que todo o processo foi realizado em conformidade.

- S -

similitude de conceitos: ocorre quando dados ou conceitos (representados por variáveis de referência) com diferentes nomes apresentam propriedades (papéis, contextos temporais, semântica) comuns. Esta similitude de conceitos expressa-se por meio de sinônimos.

sinônimos: dois ou mais nomes que representam um mesmo conceito (variável de referência). Expressa uma similitude de conceitos.

subciclo de vida: alternativas de comportamento (conjunto de estados e transições) dentro de um ciclo de vida que são exclusivas e independentes, isto é, uma vez que optou-se por uma transição que leva para o subciclo, não é possível sair dele. Os subciclos de vida são usados como indicadores para definir hierarquias de herança baseadas em comportamento.

supra-sistema: sistema maior que inclui o sistema sob modelagem, assim como outros sistemas tais como os agentes externos (porém nem todo agente externo pertence ao supra-sistema).

- V -

variável de identificação: o mesmo que **identificador único**.

variável de referência: identificador usado como referência em um superestado, de tal forma que os estados e transições que ele contém são válidos para todos os valores do domínio dessa variável de referência. O superestado representa genericamente o comportamento para aquela variável de referência.

variável livre: variável de referência de um superestado cujos valores não dependem de nenhum valor recebido como parâmetro e sim da necessidade de tratamento interno do processo do qual faz parte.

variável secundária: variável que é usada para fins de cálculo em forma interna a um processo. Esta variável não pode ser de referência.

variáveis sinônimas: duas ou mais variáveis de referência são ditas sinônimas se podem ser referenciadas indistintamente em um diagrama HLS de ciclo de vida, isto é, estímulos que apresentem parâmetros com qualquer destas variáveis de referência são aceitos nestes diagramas.

BIBLIOGRAFIA

[ABB 83] ABBOTT, Russell. Program Design by Informal English Descriptions. **Communications of the ACM**, New York, v.26, n.11, p.882-894, Nov. 1983.

[ALA 88] ALABISO, B. Transformation of Data Flow Analysis Models to Object-Oriented Design. **ACM SIGPLAN Notices**, New York, v.23, n.11, p.335-353, Nov. 1988. Trabalho apresentado na Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '88, 3., 1988, San Diego.

[BAI 89] BAILIN, Sidney. An Object-Oriented Requirements Specification Method. **Communications of the ACM**, New York, v.32, n.5, p.608-623, May 1989.

[BAT 86] BATINI, Carlo; LENZERINI, Mario; NAVATHE, Shamkant. A Comparative Analysis of Methodologies for Database Schema Integration. **Computing Surveys**, New York, v.18, n.4, p.323-364, Dec. 1986.

[BAT 92] BATINI, Carlo; CERI, Stefano; NAVATHE, Shamkant. **Conceptual Database Design:** An Entity-Relationship Approach. Redwood City: Benjamin/Cummings, 1992. 470p.

[BER 80] BERTALANFFY, Ludwig Von. **Teoria Geral dos Sistemas**. Rio de Janeiro: Vozes, 1980. 351p.

[BOH 92] BOHM, David. **A Totalidade e a Ordem Implicada**. São Paulo: Cultrix, 1992. 292p.

[BOO 83] BOOCH, Grady. Object-Oriented Design. In: FREEMAN, P.; WASSERMAN, A. (Eds.). **Tutorial on Software Design Techniques**. Los Alamitos: IEEE Computer Society Press, 1983. p.420-436.

[BOO 86] BOOCH, Grady. Object-Oriented Development. **IEEE Transactions on Software Engineering**, New York, v.12, n.2, p.211-221, Feb. 1986.

[BOO 89] BOOCH, Grady. What Is and What Isn't Object-Oriented Design? **American Programmer**, New York, v.2, n.7/8, p.3-7, Aug. 1989.

[BOO 91] BOOCH, Grady. **Object Oriented Design with Applications**. Redwood City: Benjamin/Cummings, 1991. 580p.

[BUL 89] BULMAN, David. An Object-Based Development Model. **Computer Language**, New York, v.6, n.8, p.49-59, Aug. 1989.

[BUS 93] BUSTOS, Guillermo. **Estudo Comparativo de Técnicas de Análise Orientada a Objetos**. Porto Alegre: CPGCC-UFRGS, 1993. 93p. (TI-317).

[BUS 94] BUSTOS, Guillermo. **Aplicação Comparativa de Técnicas de Análise Orientada a Objetos**. Porto Alegre: CPGCC-UFRGS, 1994. 68p. (TI-416).

[BUS 94a] BUSTOS, Guillermo. **Uma Proposta de Particionamento/Agregação Estrutural Para a Análise Orientada a Objetos**. Porto Alegre: CPGCC-UFRGS, 1994. 75p. (TI-412).

[BUS 94b] BUSTOS, Guillermo; HEUSER, Carlos. Particionamento e Agregação Estruturais na Análise Orientada a Objetos. In: CONFERENCIA LATINOAMERICANA DE INFORMÁTICA (CLEI), 20., 1994, Estado de México. **Memoria...** Ciudad de México: [s.n], 1994. p.759-770.

[BUS 94c] BUSTOS, Guillermo; HEUSER, Carlos. Comparando o Processo de Modelagem de Técnicas de Análise Orientada a Objetos. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), 8., 1994, Curitiba. **Anais...** Curitiba:[s.n], 1994. p.177-193.

[BUS 94d] BUSTOS, Guillermo. **Aplicação da Técnica de Análise *Object Modeling Technique* (OMT) ao Problema da IFIP**. Porto Alegre: CPGCC-UFRGS, 1994. 29p. (RP-232).

[BUS 94e] BUSTOS, Guillermo. **Aplicação da Técnica *Object Behavior Analysis* (OBA) ao Problema da IFIP**. Porto Alegre: CPGCC-UFRGS, 1994. 56p. (RP-234).

[BUS 94f] BUSTOS, Guillermo. **Aplicação da Técnica *Object-Oriented Analysis* de Coad & Yourdon ao Problema da IFIP**. Porto Alegre: CPGCC-UFRGS, 1994. 26p. (RP-230).

[BUS 94g] BUSTOS, Guillermo. **Aplicação da Técnica *Object-Oriented Requirements Specification Method* ao Problema da IFIP**. Porto Alegre: CPGCC-UFRGS, 1994. 22p. (RP-231).

[BUS 94h] BUSTOS, Guillermo. **Aplicação de uma Técnica Reversa de Análise Orientada a Objetos ao Problema da IFIP**. Porto Alegre: CPGCC-UFRGS, 1994. 26p. (RP-233).

[BUS 95] BUSTOS, Guillermo; HEUSER, Carlos. Modelagem Dinâmica Orientada a Objetos: Uma Análise Comparativa de Técnicas. In: Simpósio Brasileiro de Engenharia de Software (SBES), 9., 1995, Recife. **Anais...** Recife: [s.n], 1995. p.191-207.

[CAP 92] CAPRA, Fritjof. **O Ponto de Mutação**. São Paulo: Cultrix, 1992. 447p.

[CHE 76] CHEN, Peter. The Entity-Relationship Model: Toward a Unified View of Data. **ACM Transactions on Database Systems**, New York, v.1, n.1, p.9-36, Mar. 1976.

[CHU 71] CHURCHAM, C. West. **Introdução à Teoria dos Sistemas**. Petrópolis: Vozes, 1971. 309p.

[COA 92] COAD, Peter; YOURDON, Edward. **Análise Baseada em Objetos**. Rio de Janeiro: Campus, 1992. 225p.

[COL 94] COLEMAN, Derek et al. **Object-Oriented Development:** The Fusion Method. Englewood Cliffs: Prentice-Hall, 1994. 314p.

[CON 89] CONSTANTINE, Larry. Object-Oriented and Structured Methods: Toward Integration. **American Programmer**, New York, v.2, n.7/8, Aug. 1989.

[DAV 93] DAVIS, Alan. **Software Requirements:** Objects, Functions, & States. Englewood Cliffs: Prentice-Hall, 1993. 521p.

[DEC 91] DE CHAMPEAUX, Dennis. Object-Oriented Analysis and Top-Down Software Development. In: AMERICA, P. (Ed.). EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING (ECOOP '91), 1991, Geneva. **Proceedings...** Geneva: Springer-Verlag, 1991. p.360-376. (Lecture Notes in Computer Science, n.512).

[DEC 92] DE CHAMPEAUX, Dennis; LEA, Douglas; FAURE, Penelope. The Process of Object-Oriented Design. **ACM SIGPLAN Notices**, New York, v.27, n.10, p.45-62, Oct. 1992. Trabalho apresentado na Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '92, 7., 1992, Vancouver.

[DEC 93] DE CHAMPEAUX, Dennis; LEA, Douglas; FAURE, Penelope Faure. **Object-Oriented System Development**. Reading: Addison-Wesley, 1993. 532p.

[DEM 78] DE MARCO, Tom. **Structured Analysis and System Specification**. New York: Yourdon Press, 1978. 352p.

[EMB 92] EMBLEY, David; KURTZ, Barry; WOODFIELD, Scott. **Object-Oriented System Analysis:** A Model-Driven Approach. Englewood Cliffs: Prentice-Hall, 1992. 302p.

[EVB 86] EVB SOFTWARE ENGINEERING. **Object-Oriented Design Handbook**. Rockville: EVB Software Engineering Inc., 1986.

[FIC 92] FICHMAN, Robert; KEMERER, Chris. Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique. **IEEE Computer**, Los Alamitos, v.25, n.10, p.22-39, Oct. 1992.

[FIR 93] FIRESMITH, Donald. **Object-Oriented Requirements Analysis and Logical Design:** A Software Engineering Approach. New York: Wiley, 1993.

[GIB 90] GIBSON, Elizabeth. Objects - Born and Bred. **Byte**, Peterborough, v.15, n.10, p.245-254, Oct. 1990.

[GIR 90] GIRARDI, María; PRICE, Roberto. O Paradigma de Desenvolvimento por Objetos. **Revista de Informática Teórica e Aplicada**, Porto Alegre, v.1, n.2, p.69-98, maio 1990.

[HAR 87] HAREL, David. Statecharts: A Visual Formalism for Complex Systems. **Science of Computer Programming**, Amsterdam, v.8, n.3, p.231-274, June 1987.

[HEI 85] HEISENBERG, Werner. **La Imagen de la Naturaleza en la Física Actual**. Madrid: Orbis, 1985. 153p.

[HEN 90] HENDERSON-SELLERS, Brian; EDWARDS, Julian. The Object-Oriented Systems Life Cycle. **Communications of the ACM**, New York, v.33, n.9, p.142-159, Sept. 1990.

[HEN 91] HENDERSON-SELLERS, Brian; CONSTANTINE, Larry. Object-Oriented Development and Functional Decomposition. **Journal of Object-Oriented Programming**, New York, v.5, n.1, p.11-17, Jan. 1991.

[HEN 94] HENDERSON-SELLERS, Brian; EDWARDS, Julian. **BOOKTWO of Object-Oriented Knowledge:** The Working Object. Sydney: Prentice-Hall, 1994. 594p.

[HES 74] HESSEN, Johan. **Teoría del Conocimiento**. Buenos Aires: Losada, 1974. 57p.

[HEU 90] HEUSER, Carlos. **Modelagem Conceitual de Sistemas:** Redes de Petri. Buenos Aires: Kapelusz, 1990. 150p.

[HEU 93] HEUSER, Carlos A.; PERES, Eduardo; RICHTER, Gernot. Towards a Complete Conceptual Model: Petri Nets and Entity-Relationship Diagrams. **Information Systems**, Oxford, v.18, n.5, p.275-298, 1993.

[HIL 95] HILLS, Nicholas; TORNIER, Pierre. An Object-Oriented Approach to "Architecting" Open Client/Server Applications. **Object Magazine**, New York, v.4, n.9, p.51-56, Feb. 1995.

[HOY 93] HOYDAHLSVIK, G.; SINDRE, G. On the Purpose of Object-Oriented Analysis. **ACM SIGPLAN Notices**, New York, v.28, n.10, Oct. 1993. Trabalho apresentado na Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '93, 8., 1993, Washington.

[IFI 82] IFIP WG 8.1 WORKING CONFERENCE ON COMPARATIVE REVIEW OF INFORMATION SYSTEMS DESIGN METHODOLOGIES, 1982, Noordwijkerhout. **Proceedings...** Amsterdam: North-Holland, 1982.

[JAC 92] JACOBSON, Ivar et al. **Object-Oriented Software Engineering:** A Use Case Driven Approach. Wokingham: Addison-Wesley, 1992. 528p.

[JAC 95a] JACOBSON, Ivar; CHRISTERSON, Magnus. A Growing Consensus on Use Cases. **Journal of Object-Oriented Programming**, New York, v.8, n.1, p.15-19, Jan. 1995.

[JAC 95b] JACOBSON, Ivar; CHRISTERSON, Magnus. Modeling With Use Cases: A Confused World of OOA and OOD. **Journal of Object-Oriented Programming**, New York, v.8, n.5, p.15-20, Sept. 1995.

[JAL 89] JALOTE, Pankaj. Functional Refinement and Nested Objects for Object-Oriented Design. **IEEE Transactions on Software Engineering**, New York, v.15, n.3, p.264-270, Mar. 1989.

[KAP 91] KAPPEL, Gerti; SCHREFL, Michael. Using an Object-Oriented Diagram Technique for the Design of Information Systems. In: SOL, H.; HEE, K. Van (Eds.). **Dynamic Modelling of Information Systems**. Amsterdam: North-Holland, 1991. p.121-164.

[KOR 90] KORSON, Timothy; McGREGOR, John. Understanding Object-Oriented: A Unifying Paradigm. **Communications of the ACM**, New York, v.33, n.9, p.40-60, Sept. 1990.

[LEE 91] LEE, Sangbum; CARVER, Doris. Object-Oriented Analysis and Specification: A Knowledge Base Approach. **Journal of Object-Oriented Programming**, New York, v.4, n.1, p.35-43, Jan. 1991.

[LOY 90] LOY, Patrick. A Comparison of Object-Oriented and Structured Development Methods. In: THAYER, Richard; DORFMAN, Merlin (Eds.). **System and Software Requirements Engineering**, Los Alamitos: IEEE Computer Society Press, 1990.

[MAN 89] MANFREDI, F.; ORLANDI, G.; TORTORICI, P. An Object-Oriented Approach to the System Analysis. In: EUROPEAN SOFTWARE ENGINEERING CONFERENCE (ESEC '89), 2., Berlin. **Proceedings...** Berlin: Springer-Verlag, 1989. p.395-410 (Lectures Notes in Computer Science, n.387).

[MAR 94] MARTIN, James; ODELL, James. **Princípios de Análise e Projeto Baseados em Objetos**. Rio de Janeiro: Campus, 1994. 486p.

[MAT 88] MATTOSO, Adriana; BLUM, Hélcio. Proposta de Desenvolvimento de Software com Orientação a Objetos. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES), 2., 1988, Canela. **Anais...** Canela: CPGCC-UFRGS, 1988, p.7-16.

[MAH 84] MATURANA, Humberto; VARELA, Francisco. **El Arbol del Conocimiento**. Santiago: Universitaria, 1984. 172p.

[MCG 92] McGREGOR, John; SYKES, David. **Object-Oriented Software Development**. New York: Van Nostrand Reinhold, 1992. 352p.

[MEY 88] MEYER, Bertrand. **Object-Oriented Software Construction**. Hertfordshire: Prentice-Hall, 1988. 534p.

[MON 92] MONARCHI, David; PUHR, Gretchen. A Research Typology for Object-Oriented Analysis and Design. **Communications of the ACM**, New York, v.35 n.9, p.35-47, Sept. 1992.

[NAH 62] NAHUYS, Rosa; FONSECA, Lygia. **Lições de Português**. 2. ed. Buenos Aires: Kapelusz, 1962. 388p.

[NAV 89] NAVATHE, Shamkant; PILLALAMARRI, Mohan. OOER: Toward Making the E-R Approach Object-Oriented. In: BATINI, C. (Ed.). **Entity-Relationship Approach**. Amsterdam: North-Holland, 1989. p.185-206.

[NER 92] NERSON, Jean-Marc. Applying Object-Oriented Analysis and Design. **Communications of the ACM**, New York, v.35, n.9, p.63-74, Sept. 1992.

[OLL 82] OLLE, T. William. Comparative Review of Information Systems Design Methodologies: Problem Definition. In: IFIP WG 8.1 WORKING CONFERENCE ON COMPARATIVE REVIEW OF INFORMATION SYSTEMS DESIGN METHODOLOGIES, 1982, Noordwijkerhout. **Proceedings...** Amsterdam: North-Holland, 1982. p.8-9.

[OPD 93] OPDAHL, Andreas L.; SINDRE, Guttorm. Concepts for Real-World Modelling. In: COLETTE, R.; BODART, F.; CAUVET, C. (Eds.). INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING (CAiSE '93), 5., Paris. **Proceedings...** Berlin: Springer-Verlag, 1993. p.309-327.

[POT 88] POTTER, Walter; TRUEBLOOD, Robert. Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling. **IEEE Computer**, Los Alamitos, v.21, n.6, p.53-63, June 1988.

[PRE 87] PRESSMAN, Roger. **Software Engineering: A Practitioner's Approach**. 2nd ed. New York: McGraw-Hill, 1987. Section 4.4: Object-Oriented Analysis.

[REE 92] REENSKAUG, Trygve et al. OORASS: Seamless Support for the Creation and Maintenance of Object-Oriented Systems. **Journal of Object-Oriented Programming**, New York, v.5, n.6, p.27-41, Oct. 1992.

[RUB 92] RUBIN, Kenneth; GOLDBERG, Adele. Object Behavior Analysis. **Communications of the ACM**, New York, v.35, n.9, p.48-62, Sept. 1992.

[RUM 91] RUMBAUGH, James et al. **Object-Oriented Modeling and Design**. Englewood Cliffs: Prentice-Hall, 1991. 500p.

[SCH 90] SCHIEL, Ulrich; MISTRIK, Ivan. Using Object-Oriented Analysis and Design for Integrated Systems. **Arbeitspapiere der GMD**, [S.l.], v.449, June 1990.

[SEI 89] SEIDEWITZ, Edward. General Object-Oriented Software Development: Background and Experience. **The Journal of Systems and Software**, [S.l.], v.9, n.2, p.95-108, Feb. 1989.

[SHL 90] SHLAER, Sally; MELLOR, Stephen. **Análise de Sistemas Orientada para Objetos**. São Paulo: McGraw-Hill, 1990. 178p.

[SHL 92] SHLAER, Sally; MELLOR, Stephen. **Object Life Cycles:** Modeling the World in States. Englewood Cliffs: Prentice-Hall, 1992. 251p.

[SHL 93] SHLAER, Sally; MELLOR, Stephen. A Deeper Look... at the Transition from Analysis to Design. **Journal of Object-Oriented Programming**, New York, v.6, n.1, p.16-19, Jan. 1993.

[TEO 89] TEOREY, Toby et al. ER Model Clustering as an Aid for User Communication and Documentation in Database Design. **Communications of the ACM**, New York, v.32, n.8, p.975-987, Aug. 1989.

[VAR 90] VARELA, Francisco. **Conocer.** Las Ciencias Cognitivas: Tendencias y Perspectivas. Cartografía de las Ideas Actuales. Barcelona: Gedisa, 1990. 120p.

[WAR 89] WARD, Paul. How to Integrate Object Orientation with Structured Analysis and Design. **IEEE Software**, Los Alamitos, v.6, n.3, Mar. 1989.

[WEI 75] WEINBERG, Gerald. **An Introduction to General Systems Thinking**. New York: Wiley, 1975. 279p.

[WIR 90] WIRFS-BROCK, Rebecca; WILKERSON, Brian; WIENER, Lauren. **Designing Object-Oriented Software:** A Responsibility-Driven Approach. Englewood Cliffs: Prentice-Hall, 1990.

[YOU 90] YOURDON, Edward. **Análise Estruturada Moderna**. Rio de Janeiro: Campus, 1990. 836p.

[YOU 94] YOURDON, Edward. **Object-Oriented Systems Design**. Englewood Cliffs: Prentice-Hall, 1994. 400p.

Informática UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uma Proposta de Modelagem Conceitual de Sistemas Dirigida por Comportamento

por

Guillermo Alejandro Bustos Reinoso

Defesa de Tese apresentada aos Senhores:

Prof. Dr. Paulo César Masiero (ICMSC/USP)

Prof. Dr. Bruno Maffeo (PUCRJ)

Prof. Dr. Roberto Tom Price

Profa. Dra. Nina Edelweiss

Vista e permitida a impressão.
Porto Alegre, 03 / 05 / 96.

Prof. Dr. Carlos Alberto Heuser,
Orientador.

Prof. Flávio Rech Wagner
Coordenador do Curso de Pós-Graduação
em Ciência da Computação - CPGCC
Instituto de Informática - UFRGS