MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-7229-TDI/683

ESPECIFICAÇÃO DOS ASPECTOS DINÂMICOS DOS SISTEMAS

Miriam Célia Bergue Alves

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Tatuo Nakanishi, aprovada em 28 de maio de 1999.

INPE
São José dos Campos
1999

681. 3. 06

ALVES, M.C.B.
Especificação dos aspectos dinâmicos dos sistemas /M.C.B.Alves. – São José dos Campos: 1999.
277p. (INPE-7229-TDI/683).

1.Especificação. 2.Requisitos. 3.Aspectos dinâmicos. 4.Animação. 5.Objeto orientado. I.Título

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Doutor** em **Computação Aplicada.**

Dr. Airam Jônatas Preto — Presidente

Dr. Tatuo Nakanishi — Orientador

Dr. Solon Venâncio de Carvalho — Membro da Banca

Dr. João Bosco Schumann Cunha — Membro da Banca

Drª Selma Shin Shimizu Melnikoff — Membro da Banca Convidada

Dr. Adilson Marques da Cunha — Membro da Banca Convidado

Candidato (a): Miriam Célia Bergue Alves

São José dos Campos, 28 de maio de 1999.

“Em todo ser vivo, aquilo que designamos como partes constituintes forma um todo inseparável, que só pode ser estudado em conjunto, pois a parte não permite reconhecer o todo, nem o conjunto deve ser reconhecido como partes...”

Goethe

Escritor alemão do século XIX

Ao meu marido Fábio, meu grande incentivador e companheiro...

Às minhas queridas filhas Catarina e Luíza, que sempre me deram a alegria e o sorriso para continuar...

AGRADECIMENTOS

Ao Prof. Dr. Tatuo Nakanishi pela orientação, pela dedicação e, principalmente pela amizade.

Aos colegas do Instituto de Estudos Avançados (IEAv-CTA), pelo apoio, incentivo e compreensão.

Aos meus pais, pelo carinho e apoio durante todos os anos da minha vida.

À Universidade de Mogi das Cruzes (UMC) pelo auxílio concedido para a realização deste trabalho dentro do Programa de Apoio à Qualificação Docente.

RESUMO

Com o crescente aumento da complexidade e porte dos sistemas, é importante que os aspectos dinâmicos sejam observados com cuidado na fase inicial do desenvolvimento desses sistemas. Este trabalho faz um levantamento de elementos que contribuem na especificação e representação mais efetiva dos aspectos dinâmicos dos sistemas, a partir do estudo de alguns casos. Baseado no resultado desta análise e na atual carência de técnicas que plena e adequadamente representem estes aspectos, propõe-se uma nova abordagem para a sua representação, com a criação de modelos de animação. Estes modelos são criados graficamente utilizando um simbolismo simples e a animação é obtida com o emprego de técnicas de simulação discreta de sistemas, dentro de um ambiente de fácil uso e que possibilita a interação do modelador do sistema durante o processo de animação. Com isso, uma visão da dinâmica do sistema é obtida, melhorando a compreensão e, consequentemente, a especificação dos aspectos dinâmicos. Por meio de sucessivos refinamentos do modelo, é possível, ainda, que o modelador obtenha a especificação de alguns dos objetos de software componentes do sistema a ser implementado.

DYNAMIC ASPECTS SPECIFICATION OF THE SYSTEMS

ABSTRACT

A consequence of the continuous growing of the systems size and complexity is the highlighting of the importance of the software requirements specification. Dynamic aspects must be early and carefully considered in this phase. This work suggests a set of key elements that would be considered in the dynamic modeling of the systems in order to improve their specification. The practical importance of these key elements are verified and analyzed based on some case studies. Considering the limitations of the existing models and techniques for these elements representation, a new approach using animation models is proposed to create the initial model of the systems. These models are graphically created and animated by using discrete simulation techniques. An user friendly environment is proposed to support the process of animation model creation and to allow interactivity during the model animation. This approach improves both the comprehension and dynamic aspects specification of the systems by providing one more complete view of the system's dynamic. With successive model refinements, it is also possible to identify some of the software objects to be implemented.

SUMÁRIO

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

CAPÍTULO 1

INTRODUÇÃO

As áreas de aplicações computacionais estão se expandindo rapidamente devido à constante melhoria do desempenho do hardware. Aplicações não factíveis de se implementar antes, encontram-se agora viabilizadas em computadores de alto desempenho. Isto resulta numa crescente demanda por aplicações computacionais grandes e complexas.

Considerando que as aplicações computacionais envolvem sistemas constituídos de componentes de hardware e software, e que estes componentes também podem ser vistos como constituídos de outros componentes que interagem entre si para cumprir um objetivo, a importância da fase de especificação e análise dos requisitos, principalmente para o processo de desenvolvimento do software, tem sido muito enfatizada nos últimos anos (Felder e Morzenti, 1994; Heimdahl e Leveson, 1995; Kang e Ko, 1995a; Ozcan e Siddiqi, 1995; Urban e Joo, 1995; Ebert, 1997; Kang at al, 1998).

O desenvolvimento de grandes projetos de sistemas caracteriza-se pela utilização de múltiplas tecnologias e objetivos, exigindo um detalhado planejamento antes do trabalho de implementação. A produção de software de alta qualidade é demorada e dispendiosa, representando uma tarefa crítica para as indústrias, fazendo com que várias empresas que desenvolvem software, classifiquem o seu processo de desenvolvimento de acordo com o Modelo de Maturidade do Instituto de Engenharia de Software (Humphrey, 1989; SEI, 1991).

Dado a complexidade, o porte e a natureza específica dos sistemas atuais, principalmente aqueles de tempo real, onde falhas do projeto podem causar danos

irreparáveis, os aspectos dinâmicos devem ser observados com mais cuidado na fase inicial da modelagem.

A modelagem do sistema, antes de iniciar o trabalho de desenvolvimento do software, ajuda a evitar erros e aumenta a produtividade no desenvolvimento de projetos de médio a grande porte. Na realidade, especificações pobremente estruturadas, incorretas e incompletas têm produzido problemas organizacionais e econômicos significativos, comprometendo todas as demais fases do desenvolvimento do software. Isto justifica a grande atenção devotada pela comunidade de pesquisa ao estudo de linguagens de especificação e métodos de modelagem mais adequados (Felder e Morzenti, 1994; Heimdahl e Leveson, 1995; Péralti e Decotigne, 1995; Buhr, 1996; Kang at al, 1998).

O objetivo primário de uma metodologia de desenvolvimento de software é facilitar a criação, comunicação, verificação e acompanhamento durante as atividades de análise dos requisitos, projeto e implementação de um produto. Para ser verdadeiramente efetiva, uma moderna metodologia deve explorar também a evolução tecnológica para, automaticamente, produzir implementações a partir dos projetos, gerar casos de testes derivados das especificações de requisitos e analisar projetos e bibliotecas de componentes de software reutilizáveis.

Na análise desses sistemas devem ser levados em conta aspectos temporais e funcionais, assim como os dados, para que se possa especificar e analisar os requisitos com precisão. O manuseio apropriado destes aspectos temporais e funcionais envolve: um método de análise que deve fornecer um conjunto de primitivas de modelagem para representá-los; uma base formal para a verificação da consistência; uma linguagem de especificação orientada ao usuário; e a possibilidade de uma abordagem tanto *top-down* como *bottom-up* para simplificar o trabalho de produção das especificações.

A complexidade de uma especificação de sistema pode ser reduzida iniciando o processo de especificação a partir de uma especificação altamente abstrata, e posteriormente refinando-a gradualmente, de uma maneira *top-down*, repetindo a mesma seqüência de atividades.

Uma análise prática, completa e correta, e uma validação e verificação nos estágios iniciais do desenvolvimento de software são tarefas muito importantes. A validação refere-se ao processo de esclarecer e confirmar as necessidades do usuário, enquanto a verificação é um processo de verificar se certas propriedades do sistema especificadas pelo usuário serão satisfeitas (Kang et al, 1998).

Uma validação e uma verificação não satisfatórias podem causar interações de projeto custosas, quando uma revisão de requisitos é feita. Davis (1993) relata que erros de requisitos encontrados durante a implementação podem custar de cinco a dez vezes o preço de corrigí-los durante a fase de especificação de requisitos.

Ainda hoje, as técnicas e metodologias existentes para a modelagem de sistemas continuam evoluindo, buscando resolver problemas ainda não solucionados e tentando facilitar o processo de criação de modelos abstratos da realidade.

As alterações feitas nas tradicionais técnicas de análise e projeto estruturado realizadas por Ward e Mellor (1985) e Hatley e Pirbhai (1987), já buscavam incorporar novas características, para que os modelos construídos pudessem retratar melhor os sistemas de tempo real.

O paradigma de orientação a objetos não é novo. Durante os anos 80, a orientação a objetos começou a ter um impacto em partes do ciclo de vida do software, fora da atividade de codificação. Em 1980, Grady Booch introduziu o conceito de projeto orientado a objeto (Object Oriented Design - OOD) como um processo do ciclo de vida que definiu os inter-relacionamentos entre os componentes de software que compunham a aplicação (Booch, 1994). Outros autores (Wirfs-Brock at al, 1990; Rumbaugh, 1991), também externaram suas visões sobre o projeto de sistemas orientado a objetos.

No final dos anos 80, a necessidade de se especificar os requisitos de um produto de software de uma maneira orientada a objetos tornou-se evidente. A tentativa de se utilizar métodos tradicionais, tal como a análise estruturada, na fase de análise de um projeto orientado a objetos, não foi satisfatória. Isto provocou o estabelecimento da Análise Orientada a Objetos (Object Oriented Analysis – OOA) (Coad e Yourdon, 1991) e da Análise de Requisitos Orientada a Objetos (Object Oriented Requirements

Analysis - OORA) de Shlaer e Mellor (1990). Posteriormente, surgiram outras metodologias orientadas a objeto (Jacobson et al 1992; Selic 1992, 1993, 1994; Coad et al, 1996).

Mais recentemente, o desenvolvimento de uma técnica inovadora de modelagem chamada Mapas *use case* (Buhr e Casselman, 1996), na área de orientação a objetos, considera a importância de representar os aspectos dinâmicos e concorrentes dos sistemas. A técnica, além de outras características, utiliza-se da especificação de cenários de uso, os *use cases* (Jacobson et al, 1992).

Atualmente existem discussões sobre análise de domínio orientada a objeto, testes de software orientados a objeto (Jorgensen e Erickson, 1994; Poston, 1994) e métricas para complexidade de software orientado a objeto (Henderson-Sellers, 1996). Há também literatura sobre o gerenciamento do desenvolvimento de software orientado a objeto (Booch, 1995; Goldberg e Rubin, 1995; Cockburn, 1998; Taylor, 1998).

Em síntese, pode ser observado que o paradigma de orientação a objeto surgiu no final dos anos 60, começou a ter uma definição significativa nos anos 70 e se estabeleceu em meados dos anos 80. Nos anos 90 o entendimento, a prática e o uso da orientação a objetos aumentou, tornando-se o principal ponto de referência para o desenvolvimento de software.

Uma das vantagens preconizadas por diversos autores para a adoção do paradigma de orientação a objeto, é que ele se aproxima mais do mundo real quando comparado aos paradigmas clássicos, exigindo menos abstração na construção dos modelos. Isto significa, a princípio, que se pode desenvolver sistemas numa linguagem próxima à natural, e vários métodos orientados a objetos são baseados neste argumento. Como exemplos, podem ser citados a Análise de Requisitos Orientada a Objetos Hierárquica (*Hierarchical Object Oriented Requirement Analysis* - HOORA) e o Projeto Orientado a Objetos Hierárquico (*Hierarchical Object Oriented Design* - HOOD), utilizados pela Agência Espacial Européia (Gennaro, 1995), e o método de Informação Kristen e Serviços de Software (*Kristen Information and Software Services* - KISS), proposto por Gerald Kristen (Wienand, 1997).

A tecnologia de orientação a objetos atualmente está bastante amadurecida, e oferece mais confiança aos desenvolvedores. O crescente aumento da necessidade de utilização de sistemas distribuídos e de tempo real, justificado principalmente pela Internet, juntamente com o paradigma de desenvolvimento de sistemas baseados em componentes, deu ainda mais força a esta tecnologia.

A definição de padrões para *middleware* (camada de software intermediária entre os aplicativos e os serviços em um ambiente de processamento distribuído) como a Arquitetura de Gerenciamento e Comunicação entre Objetos (*Common Object Request Broker Architecture* – CORBA) (Orfali et al, 1996) é uma nítida evidência de que se caminha no sentido do desenvolvimento de componentes interoperáveis, localizados em lugares físicos distintos, alojados em sistemas operacionais distintos e escritos em linguagens de programação diferentes. Neste contexto, entende-se por componentes de software pedaços de código auto contidos e inteligentes.

No que diz respeito à representação da dinâmica dos sistemas, tanto as técnicas estruturadas como as orientadas a objeto criam modelos predominantemente estáticos. Modelos estáticos podem ser vistos como a representação de instantâneos do sistema, que mesmo tomados em vários momentos, ainda não fornecem a visão dinâmica do todo. Modelos animados são fortes candidatos a representar melhor aspectos dinâmicos de um sistema.

A ênfase no uso de modelos de animação na maioria dos casos (Gaskell e Phillips, 1994) está na modelagem dos componentes de software. Entretanto a compreensão dos aspectos dinâmicos do sistema envolve também os componentes de hardware.

Neste contexto, o objetivo deste trabalho é propor uma nova abordagem para a representação dos modelos dinâmicos dos sistemas num sentido mais amplo, visando melhorar a especificação e a compreensão dos aspectos dinâmicos envolvidos. Para cumprir este propósito, inicialmente são levantados alguns elementos considerados necessários para uma especificação e representação mais efetiva dos aspectos dinâmicos, fazendo a análise de alguns casos.

Com base no resultado desta análise e considerando a atual carência de técnicas que representem adequadamente, e de modo completo estes aspectos (Alves, 1998), uma nova técnica para a especificação e representação dos aspectos dinâmicos é proposta, resultando na criação dos modelos de animação do sistema.

Para dar suporte à criação destes modelos de animação, um ambiente de modelagem é caracterizado para criá-los graficamente, utilizando um simbolismo simples. A animação é obtida com o emprego de técnicas baseadas em simulação discreta de sistemas. O resultado é uma visão da dinâmica do sistema como um todo, melhorando a sua compreensão e facilitando a especificação dos seus aspectos dinâmicos.

O modelo de animação pode ser decomposto em vários níveis, tornando possível, por meio de sucessivos refinamentos, a identificação de alguns dos objetos de software componentes do sistema a ser implementado, resultando num ponto de partida para que uma análise orientada a objetos mais detalhada possa ser realizada e um projeto orientado a objetos possa ser posteriormente conduzido.

A motivação para este trabalho leva em conta também o desafio fundamental no moderno desenvolvimento de software, ou seja, gerar um produto de alta qualidade a despeito dos crescentes níveis de sofisticação e complexidade dos sistemas, principalmente os de tempo real e os distribuídos, comuns nas telecomunicações, indústria aeroespacial, indústria de manufatura e sistemas de controle de vôo e de defesa.

Enquanto o tamanho e a quantidade de sistemas de software produzidos têm aumentado dramaticamente nas duas últimas décadas, a habilidade para reduzir a taxa de erros não tem progredido de maneira correspondente. A taxa de erros em projetos de software tem sido muito maior do que a taxa de erros em projetos produzidos nos campos tradicionais da engenharia (Joyce, 1987; Fitzgerald, 1990; Neumann, 1990; Leveson e Turner, 1993; Lions, 1996).

Para minimizar este problema, o desenvolvimento de software de grande porte e complexo deveria ser acompanhado por abordagens mais sistemáticas. Há um grande número de tais abordagens, e todas elas com o propósito de produzir bons sistemas. A

definição de um bom sistema varia em certos aspectos, de acordo com as diferentes aplicações. Em alguns, o desempenho é mais importante, em outros a interface amigável com o usuário. A maneira como um sistema encontra-se estruturado, distribuído ou centralizado, também pode influenciar nesta definição.

Diferentes métodos de desenvolvimento têm sido propostos com o objetivo de projetar sistemas de grande porte e complexos. Tradicionalmente, o processo de desenvolvimento começa com a especificação dos requisitos que, em geral, constituí-se de uma descrição textual do sistema. Normalmente esta descrição realiza-se pelo cliente, ou em conjunto com os desenvolvedores, a partir da especificação dos requisitos, de uma análise e de uma descrição lógica. Alternativamente, estas atividades podem ser conduzidas juntamente com a especificação dos requisitos.

O trabalho de análise deve ser longo o suficiente para que o sistema seja compreendido como um todo, mas não tão longo para considerar detalhes da atividade de projeto na atividade de análise.

No domínio de sistemas de software de tempo real, os requisitos devem ser cumpridos de forma cada vez mais rígida. O tempo de resposta, isto é, a resposta a eventos externos dentro de um intervalo de tempo pré-definido, representa somente um dos muitos desafios significativos do desenvolvimento (Gullekson, 1995a). Dentro do conjunto de requisitos a satisfazer, pode-se destacar aqueles ligados à reação que o sistema tem que ter diante da ocorrência de uma série de eventos ou estímulos, cuja ordem e o momento da ocorrência são freqüentemente imprevisíveis. Além disso, estes sistemas normalmente são grandes e distribuídos e a fase de projeto deve considerar a possibilidade de falhas aleatórias dos componentes e interrupção das comunicações.

O paralelismo na ocorrência de diferentes eventos é outro aspecto importante. O tempo de resposta, a reação aos eventos e estímulos e a distribuição necessitam de múltiplas ramificações de controle. Comunicação e sincronização entre estas ramificações tornam-se problemas maiores da fase de projeto.

Outro requisito normalmente presente em sistemas de tempo real é a sua variação de estrutura, que freqüentemente precisam mudar dinamicamente para se adaptarem às

mudanças no ambiente. Sistemas de telefonia, onde usuários podem ser adicionados e o número de chamadas em progresso varia, são apenas alguns exemplos.

Além dos desafios inerentes aos sistemas de tempo real, existem os tradicionais do desenvolvimento de software. Os requisitos freqüentemente incompletos inicialmente podem esconder inconsistências. As implementações do produto consistem do código fonte descrevendo o sistema, enquanto a documentação descreve o projeto de alto nível intencionado. A equipe de desenvolvimento deveria manter estas duas representações do sistema consistentes e a ligação delas com os requisitos. Contudo, normalmente não há ligações formais entre requisitos, projetos e implementações, devido a problemas de mapeamento entre os modelos produzidos durante as atividades do processo de desenvolvimento, tornando os documentos de projeto difíceis de serem validados de forma confiável.

Problemas de mapeamento são geralmente causados por dois tipos de erros: os de interpretação e os de tradução (Pountain, 1996). Erros de interpretação ocorrem quando os implementadores não entendem a semântica do projeto de nível mais alto. Este fato pode ocorrer com mais freqüência se os modelos usados para a especificação não têm uma semântica formal precisa. Erros de tradução ocorrem mesmo se a semântica do projeto é entendida pelos implementadores, mas a natureza informal da transformação pode resultar em implementações incorretas. Quanto mais complexas as semânticas de alto nível, maior a chance de que a tradução para o código esteja incorreta.

Há descontinuidades semânticas entre as abstrações de alto nível usadas para descrever e entender sistemas de software complexos e as abstrações de baixo nível, oferecidas pelo hardware, linguagens de programação e sistemas operacionais.

Sistemas de telecomunicações, por exemplo, têm uma longa vida de serviços e taxas altas de atualização. Isto, inevitavelmente, significa que a arquitetura do sistema fica comprometida com a tradução manual. A única descrição precisa do sistema torna-se o código fonte que continuamente muda. Neste caso, muito do valor da análise e do projeto podem ficar perdidos (Gullekson, 1995a).

Algumas ferramentas de Engenharia de Software Assistida por Computador (*Computer Aided Software Engineering* - CASE) mantêm ligações bidirecionais entre o código fonte e o modelo, isto é, mudanças que são feitas no código fonte são refletidas no modelo e vice-versa. Esta característica é freqüentemente chamada de *round-trip engineering* (Pountain, 1996).

Os desafios apresentados acima, justificam todo o esforço da comunidade científica da área de Engenharia de Software na busca de técnicas melhores que produzam sistemas com mais qualidade, mais confiáveis e em prazos adequados.

Nesta busca deve ser considerada a importância de se produzir melhores especificações de requisitos e modelos iniciais dos sistemas que, diante do contexto colocado acima, devem levar em conta seus aspectos dinâmicos.

A situação atual das especificações de requisitos em relação à representação dos aspectos dinâmicos, encontra-se mais voltada para descrições textuais, mais informais e suscetíveis a erros de interpretação e mapeamento, ou ao uso de técnicas mais formais, como redes de Petri (Peterson , 1981), cenários de uso (Jacobson, 1992; Coad at al, 1996), diagramas de transição de estado (Harel, 1988) e diagramas de seqüência de mensagens (Rumbaugh, 1991; UML, 1997), exigindo de quem as utiliza um conhecimento mais específico para aplicá-las. Em ambos os casos, as técnicas empregadas não fornecem a visão global do funcionamento do sistema.

Uma técnica que considere a representação do modelo dinâmico do sistema e não exija de quem a utilize conhecimentos extensos para aplicá-la, pode trazer melhorias para o processo de especificação dos requisitos. Com o modelo dinâmico deseja-se especificar e visualizar o transcorrer dos acontecimentos, ao longo do tempo, durante o funcionamento do sistema. O modelo de animação proposto neste trabalho decorre da representação da dinâmica e associa-se aos eventos, às ações e transformações que os componentes do sistema sofrem ou realizam ao longo do tempo, sob certas condições ou circunstâncias. Este comportamento deve ser visível no modelo de animação pelos outros componentes, que participam e colaboram entre si.

Considerando que a técnica para criação deste modelo de animação é proposta para ser utilizada na fase de especificação de requisitos, é importante deixar claro quais são os participantes deste processo. Há no mínimo duas partes envolvidas: representada pelos usuários, preocupados com o comportamento externo do sistema, e representada pelos analistas que preocupados com o seu comportamento interno (Kang e Ko, 1995b). No escopo deste trabalho utiliza-se o termo modelador para os participantes deste processo, representando analistas, especialistas no domínio do problema e usuários do sistema a ser projetado.

O trabalho estrutura-se em partes conforme descritas a seguir.

O Capítulo 1 consiste desta introdução, onde foram apresentados a motivação e o objetivo. Com a intenção de justificar a importância dos aspectos dinâmicos na modelagem dos sistemas, alguns dos principais desafios, envolvendo o desenvolvimento dos sistemas atuais, também foram apresentados.

O Capítulo 2 traz um levantamento de alguns elementos julgados chaves, que devem estar representados no modelo dinâmico de um sistema. Nele, por meio do estudo de alguns casos, são analisados o emprego e a importância destes elementos.

Com base na análise dos resultados apresentados no Capítulo 2, é proposta no Capítulo 3 a criação de especificações gráficas animadas, ou de modelos de animação, para a representação dos aspectos dinâmicos dos sistemas, utilizando técnicas de simulação e animação. Neste Capítulo realiza-se também uma exposição dos principais mecanismos empregados.

O Capítulo 4 apresenta as características de um ambiente para a criação dos modelos de animação de um sistema, utilizando técnicas de animação e de simulação discreta orientada a eventos. É proposto um simbolismo para a representação gráfica deste modelo. Este ambiente considera os mecanismos de simulação necessários para a animação do modelo gráfico criado, de forma transparente para o modelador, não implicando que ele tenha que ter extensos conhecimentos das áreas de simulação e animação envolvidas. Também é apresentado o ciclo de vida do modelo de animação.

Levando em conta o ambiente caracterizado no Capítulo 4, o Capítulo 5 apresenta a especificação e a análise de uma estrutura orientada a objetos para a implementação deste ambiente.

O Capítulo 6 apresenta um exemplo da criação do modelo de animação para um dos casos apresentados no Capítulo 2, o Sistema de Gerência de Tráfego de Trens. O modelo é refinado até que se consiga derivar alguns objetos de software do sistema.

Fechando o texto, no Capítulo 7 são retomados os principais aspectos do trabalho, e identificadas suas contribuições e possibilidades de extensões para trabalhos futuros.

CAPÍTULO 2

ASPECTOS DINÂMICOS DOS SISTEMAS

2.1 DINÂMICA DOS SISTEMAS

A habilidade para representar o comportamento e a dinâmica é importante para uma compreensão mais abrangente e de alto nível na fase inicial de modelagem, na fase de projeto, na evolução e reengenharia de todos os tipos de sistemas, desde programas orientados até objetos a sistemas computacionais paralelos e distribuídos.

Atualmente os sistemas tendem a ser cada vez mais abertos permitindo, por exemplo, que um mesmo sistema de software possa ser executado em diferentes plataformas de hardware, tenha acesso a diferentes bases de dados e ainda seja distribuído (Orfali et al, 1996).

Entender como o sistema funciona e mostrar o funcionamento deste para terceiros, envolve a visualização da sua dinâmica e do seu comportamento, uma tarefa não trivial. São necessários modelos e técnicas melhores do que as atuais para lidarem com esses aspectos.

Toda metodologia de projeto inclui um conjunto de conceitos fundamentais de modelagem, tais como máquinas de estados finitos e processos concorrentes, utilizado para construir as especificações do sistema. Isto constitui uma linguagem de modelagem que, como qualquer outra linguagem, representa o vocabulário básico usado durante a análise e projeto. Essa linguagem, não somente define “como” os objetos devem ser descritos, mas também “o que” pode ser dito sobre eles, ou seja, suas propriedades.

Os sistemas de tempo real possuem um conjunto muito específico de requisitos de modelagem, que são primariamente uma conseqüência da natureza assíncrona, concorrente e distribuída do mundo real no qual estão embutidos. Devido a estas

propriedades serem tão específicas, as metodologias mais gerais não as suportam adequadamente. Isto restringe a habilidade das equipes de desenvolvimento na formulação e comunicação de suas idéias e na melhor compreensão do domínio do problema.

A predominância do assincronismo, distribuição e concorrência em sistemas de tempo real é refletida nos sistemas operacionais e bibliotecas de suporte à aplicação, utilizados nas implementações. Alguma forma de processos concorrentes e facilidades de comunicação entre os processos são fornecidas em praticamente todo sistema operacional de tempo real. Em contraste encontram-se as abstrações de nível mais alto, tais como máquinas de estados finitos, usadas em modelos produzidos pelas atividades de análise e projeto. Se não há ligação entre os conceitos de modelagem de nível mais alto e conceitos de implementação, os desenvolvedores podem fazer transformações *ad hoc*, mais propensas a erros.

Um dos atributos dominantes e que preocupa nos sistemas atuais, é a sua distribuição física. Uma aplicação neste domínio tipicamente opera sobre uma configuração de hardware, consistindo de um conjunto de locais de processamento distintos, que interagem sobre um meio de comunicação compartilhado. Isto significa que uma aplicação típica é inerentemente concorrente e deve lidar com dificuldades de projeto, como falhas parciais locais e falhas de comunicação não confiável (Selic at al, 1992; Gullekson, 1995b). Em adição, esses sistemas envolvem funcionalidade complexa, consistindo de milhares de funções diversas e freqüentemente requerendo equipes de desenvolvimento numerosas. Essas funções são dirigidas por eventos e o momento em que os eventos externos afetarão o sistema, em geral, não pode ser previsto antecipadamente, como ocorre, por exemplo, em sistemas de telecomunicações e na indústria aeroespacial.

Estas questões trazem dificuldades ao processo de modelagem dos sistemas e desafiam as atuais metodologias a fornecerem técnicas que os representem de forma mais completa.

2.2 COMPOSIÇÃO DOS SISTEMAS

Os sistemas podem ser vistos como constituídos de **entidades componentes** que colaboram e interagem para atingir um objetivo comum. Estas entidades componentes, por sua vez, podem ser constituídas de outras entidades componentes, criando-se assim vários níveis de decomposição.

No contexto deste trabalho, entende-se por entidades componentes: as **entidades concretas** ou **objetos físicos** (tais como: documentos; produtos; robôs; veículos; trens; empilhadeiras; sensores; atuadores e computadores); os **seres humanos** (tais como: usuários; clientes e operários); e as **entidades abstratas** (tais como: dados; sinais; eventos; programas de computador e instruções). O termo **entidade** também poderá ser empregado no texto para designar as entidades componentes de um sistema.

A modelagem da colaboração e da interação das entidades componentes envolve, entre outros aspectos, a modelagem da dinâmica e do comportamento dos sistemas a que pertencem.

A **dinâmica** de um sistema refere-se; ao conjunto de eventos (eventos externos e geração de eventos internos), que podem ocorrer em seqüência ou em paralelo; ao fluxo de entidades, à criação e destruição de entidades; e às transformações ocorridas com elas. Com o modelo dinâmico deseja-se especificar e visualizar o transcorrer dos acontecimentos, ao longo do tempo, durante o funcionamento do sistema, trazendo desta forma melhorias para a especificação dos requisitos.

O **comportamento** do sistema decorre da representação da dinâmica e está associado a ações e transformações que as entidades componentes sofrem ou realizam, sob certas condições ou circunstâncias. Este comportamento deve ser visível no modelo dinâmico pelas outras entidades, que participam e colaboram entre si para o correto funcionamento de todo o sistema.

O comportamento de uma entidade, por sua vez, pode ser entendido por meio da representação da sua dinâmica, tornando possível a criação de diferentes níveis de visualização da dinâmica do sistema: a dinâmica global e a dinâmica da entidade. Para que as seqüências de eventos e as transformações possam ser representadas de uma

maneira mais realística, aspectos temporais devem também ser incorporados na representação dos modelos.

Os sistemas podem mudar de forma enquanto estão sendo executados, ou seja, eles podem mudar sua estrutura dinamicamente durante sua execução. Quando estes sistemas são modelados com diagramas estáticos, a dinâmica só pode ser visualizada por meio de uma seqüência de instantâneos do sistema, tirados em diferentes momentos (Buhr, 1996). Estes instantâneos teriam que ser montados para que a dinâmica pudesse ser visualizada. Mesmo assim perde-se os passos intermediários, relacionados com o comportamento transiente do sistema.

Existem propostas para que a dinâmica dos sistemas de software seja representada, no início da modelagem, por meio da descrição de cenários de uso, complementados por diagramas de interação (Jacobson et al, 1992). Estes diagramas são genericamente chamados de diagramas de seqüência de mensagens entre os objetos ou componentes. Para a criação do modelo inicial, esta combinação de cenários de uso e diagramas de seqüências de mensagens já incorpora detalhes que, às vezes, ainda não são conhecidos. A abstração de tais detalhes é importante para que a visualização de um quadro maior do funcionamento do sistema torne-se possível.

Representar a dinâmica dos sistemas, utilizando-se apenas de diagramas estáticos compromete a sua compreensão como um todo e dificulta também a escolha apropriada de sua arquitetura, provocando o adiamento destas decisões para fases posteriores do processo de desenvolvimento do sistema.

2.3 LIMITAÇÕES DOS MODELOS E TÉCNICAS ATUAIS

Como já mencionado na Seção anterior, os sistemas podem ser vistos como constituídos de entidades componentes que colaboram entre si para atingir a funcionalidade requerida. As entidades componentes de software podem ser vistas como componentes autônomos, auto gerenciáveis e colaboradores, e deveriam fornecer aos usuários e desenvolvedores o mesmo nível de interoperabilidade disponível para os clientes e fabricantes de partes eletrônicas ou circuitos integrados adaptados.

Fracamente acoplados e altamente configuráveis, os componentes devem ser capazes de acomodar mudanças contínuas nos sistemas, aumentando de maneira considerável a produtividade. A tecnologia de componentes, em todas as suas formas, promete mudar radicalmente a maneira como os sistemas de software serão desenvolvidos.

Hoje já existe um consenso razoável na comunidade científica sobre os conceitos da orientação a objetos e sua grande importância no desenvolvimento de software (Nicol et al, 1993). Considerando suas características já bastante conhecidas, a orientação a objetos vem ao encontro das necessidades da modelagem de sistemas baseados em componentes.

Além disso, esta abordagem tem-se mostrado promissora. Vários projetos de pesquisa, incluindo sistemas de computação distribuída e projetos para Gerenciamento de Objetos Distribuídos, estão investindo esforços para padronização nessa área.

Padrões como o CORBA, do Grupo de Gerenciamento de Objetos (*Object Management Group*) (Orfali et al, 1996; Otte et al, 1996), o Ambiente de Gerenciamento Distribuído (*Distributed Management Environment*) da Fundação de Software Aberto (*Open Software Foundation* – OSF) (Orfali et al, 1996) e a Padronização para Processamento Distribuído Aberto (*Open Distributed Processing Standardization)* (Orfali et al, 1996) estão surgindo, refletindo a crescente popularidade de padrões envolvendo abordagens de orientação a objetos em sistemas distribuídos. Diversas aplicações já vêm sendo construídas utilizando-se destes padrões.

Componentes de software são hoje construídos visando principalmente a modularidade e reutilização, aumentando assim a produtividade e evitando que códigos redundantes sejam escritos. A ênfase nas interfaces e módulos tem levado muitos especialistas a concordarem que, modelar um sistema distribuído como uma coleção distribuída de objetos interagindo, é apropriado para integrar recursos de processamento distribuído em ambientes de computação distribuída e ambientes de tempo real, como em telecomunicações por exemplo (Nicol at al, 1993).

Algumas limitações podem ser apontadas em relação às técnicas que dão suporte à criação dos modelos destes sistemas. Os diagramas de classes e objetos, essencialmente

diagramas estáticos, são os primeiros diagramas recomendados para serem construídos, de acordo com as principais metodologias e notações orientadas a objeto (Shlaer e Mellor, 1990; Coad e Yourdon, 1991; Rumbaugh, 1991; Booch, 1994; Martin, 1995; Coad at al, 1996; UML, 1997; Alves, 1998), não fornecem a visão do funcionamento global, pois o comportamento e a dinâmica do sistema ficam encapsulados nas operações dos objetos e conexões de mensagens entre eles..

Os diagramas de seqüências de mensagens ou diagramas de interação entre objetos, que são recomendados por várias metodologias (Embley, 1992; Jacobson et al, 1992; Booch, 1994; Coleman et al, 1994; Gennaro, 1995; Martin, 1995; UML, 1997), enfocam aspectos específicos de um determinado cenário de interação.

Os diagramas de transição de estados, também recomendados por várias metodologias e notações (DeMarco, 1978; Gane e Sarson, 1979; Ward e Melor, 1985; Hatley e Pirbhai, 1987; Shlaer e Mellor, 1990; Yourdon, 1990; Coad e Yourdon, 1991; Rumbaugh, 1991; Embley, 1992; Jacobson et al, 1992; Meyer, 1992; Booch, 1994; Gennaro, 1995; Martin, 1995; UML, 1997), mostram os possíveis estados de um determinado objeto ou do próprio sistema durante o seu funcionamento. Nenhum dos diagramas mostra toda a funcionalidade do sistema, que é dada pela execução quase sempre concorrente desses cenários, normalmente envolvendo vários objetos.

A análise estruturada convencional é uma ferramenta para ajudar a lidar com a complexidade funcional. Ela não modela a dinâmica do sistema e ajuda muito pouco com os dados. O modelo de entidades e relacionamentos incorporado ao método é uma ferramenta para lidar com a complexidade dos dados armazenados.

Os diagramas de contexto e os Diagramas de Fluxo de Dados (DFD) são propostos como os primeiros diagramas a serem construídos de acordo com a maioria das metodologias estruturadas (DeMarco, 1978; Gane e Sarson, 1979; Yourdon, 1990), mas estes não proporcionam a visão conjunta dos processos (funções, operações e ações) e dados, dando enfoque maior nas transformações executadas pelo sistema. Seqüência e controle das ações ao longo do tempo não são levados em conta nestes diagramas.

Existem extensões significativas do diagrama de fluxo de dados clássico para atender as necessidades da modelagem de sistemas de tempo real (Ward e Mellor, 1985; Hatley e Pirbhai, 1987). As extensões ao DFD relativas aos estados e controle formam uma ferramenta para tentar lidar com a complexidade dinâmica dos sistemas. Estas extensões nos diagramas acrescentam fluxos de controle e processos de controle. Estes fluxos e processos podem ser representados num mesmo diagrama (Ward e Mellor, 1985), junto com os outros processos e fluxos, ou ainda podem ser criados diagramas separados (Hatley e Pirbhai, 1987). A representação da seqüência e do tempo não é considerada pela técnica, prejudicando a visão global do funcionamento do sistema.

Diagramas de eventos (Rumbaugh, 1991; Coleman at al, 1994), que podem ser utilizados tanto em abordagens estruturadas como orientadas a objeto, retratam um cenário particular de uso do sistema e são construídos a partir do conhecimento de alguns detalhes, nem sempre disponíveis no início da modelagem.

Os casos de uso ou *use cases* (Jacobson et al, 1992) são especialmente adequados para aplicações de negócios, onde os usuários devem estabelecer a notação usada para modelar seu trabalho prático. Um avanço foi dado com os *use cases*, mas continua sendo uma técnica limitada. Seus próprios criadores propõem a sua utilização em conjunto com outras técnicas, como por exemplo diagramas de interação e diagramas de fluxo de eventos, que os complementam com mais detalhes. Novamente aqui, a visão da dinâmica do sistema como um todo não é fornecida.

Os cenários (Rumbaugh, 1991; Booch, 1994; Coleman et al, 1994; Coad et al 1996), que podem ser vistos como a execução particular de um *use case*, mostram uma seqüência de eventos que ocorrem durante uma transação do sistema, e suas descrições textuais devem ser complementadas por diagramas de interação e fluxo de eventos para uma maior clareza. Contudo, a dinâmica de um sistema como um todo quase sempre é dada pela execução paralela de vários destes cenários.

Statecharts e *objectcharts* (Harel, 1988; Coleman et al, 1992; Selic, 1993), assim como os diagramas de transição de estados, são boas ferramentas para a modelagem do comportamento e da dinâmica de cada um dos objetos do sistema. A novidade em relação aos diagramas de transição de estados é que com os *statecharts* é possível

modelar paralelismo e concorrência. Já com os *objectcharts* pode-se ainda incorporar informações a respeito dos atributos do objeto sendo modelado. Existem possibilidades de se animar os *statecharts*, o que já acrescenta um ganho considerável em termos de visualização (Harel e Gery, 1997; Rolina, 1998). O resultado prático disto é uma melhoria na capacidade de modelagem da dinâmica dos objetos, mas não na modelagem do sistema como um todo.

A abordagem dirigida por responsabilidade dos mapas *use case*, proposta por Buhr (1994, 1995, 1996), é a técnica que mais atende a necessidade já discutida neste trabalho. Ela fornece uma notação visual para os *use cases* e também uma maneira de estendê-los para uma abstração de nível mais alto (Achite, 1997). Entender mapas *use case* não depende de familiaridade e conhecimento dos *use case*. Eles por si só são uma nova abstração, fornecendo uma estrutura comportamental e dinâmica baseada em responsabilidades do sistema, possibilitando também a tomada de decisões arquiteturais em um nível alto de abstração. Embora bastante abrangente neste sentido, a técnica ainda não representa seqüências, aspectos temporais (“quando”) e as transformações realizadas pelo sistema.

A rede de Petri (Peterson, 1981) é adotada por alguns devido à sua facilidade de combinação de redes em relação aos diagramas de transição de estados (diagrama de estados finitos), facilidade de modelagem do comportamento dinâmico integrado dos objetos, melhor visualização do sistema e similaridade entre os conceito de objetos e os conceitos de processos que se comunicam de maneira concorrente dentro de um sistema.

As redes de Petri podem ser utilizadas para a modelagem do comportamento dinâmico integrado dos objetos de um sistema, onde estes objetos podem ser representados por um grupo de processos cooperantes, que se comunicam por meio de um mecanismo de troca de mensagens. Elas são mais gerais do que os diagramas de transição de estados, e apresentam melhor capacidade de representação e possibilidade de análise de sistemas complexos e concorrentes.

Existem técnicas de análise de redes de Petri que podem ser aplicadas ao sistema modelado para verificar a ausência de *deadlock*, a conservação de recursos e o seu

comportamento cíclico. Porém, esta análise da rede exige um bom conhecimento do assunto por parte do analista. Mesmo que o analista detenha o conhecimento, ainda resta o fato de que na representação de redes de Petri não leva em consideração outros aspectos importantes da modelagem, como transformações, fluxo de dados e tempo, dificultando a visão do sistema como um todo.

Considerando o exposto acima, a seqüência e o controle das ações ao longo do tempo só poderia ser representada se o tempo (relativo ou absoluto) fosse considerado no processo de modelagem. Seria também possível avaliar gargalos de tempo de resposta do sistema quando os estímulos e os eventos mais importantes ocorressem, guiando na escolha de alternativas de projeto e na identificação de problemas que podem comprometer o desempenho geral do sistema, tais como: concorrência por recursos; execução paralela de processos; e o uso de técnicas orientadas a objetos (herança e criação dinâmica de objetos).

O uso de uma técnica mais formal para a especificação do primeiro modelo do sistema é o que alguns argumentam como sendo a cura para os problemas de software, principalmente no que diz respeito às especificações ambíguas dos requisitos, onde uma verificação formal poderia ser conduzida. Contudo, um certo grau de informalidade faz parte da essência do início do processo de desenvolvimento, quando a mente humana trabalha com requisitos estabelecidos oralmente ou em prosa para tratar abordagens candidatas que os satisfaçam (Amyot, 1993; Bordeleau e Amyot, 1993; Bordeleau et al, 1994).

Os métodos formais precisam ser integrados ao processo de desenvolvimento. Uma das principais razões para a falha da integração dos métodos formais nesse processo, está no fato de que a maioria dos métodos propostos baseiam-se na hipótese de que os desenvolvedores irão mudar completamente suas metodologias de projeto para metodologias formais (Bordeleau et al, 1994; Santos, 1996).

O aspecto informal do processo inicial de modelagem de um sistema faz parte da característica criativa deste estágio. Partes do sistema que são bem conhecidas já poderiam, no início, passar por um processo formal de especificação. Porém, podem existir partes do sistema que ainda são desconhecidas e precisem de processos

experimentais e criativos, até que se chegue a um consenso sobre seu funcionamento adequado.

À medida que se melhora a compreensão do funcionamento do sistema e mais detalhes são acrescentados na modelagem de suas partes, a construção de especificações mais formais torna-se viável e estas, sem dúvida nenhuma, diminuiriam os erros normalmente ocorridos durante a tradução dos modelos de projeto para os modelos de implementação (código). O ideal seria a existência de ferramentas automatizadas que traduzissem os diagramas e os modelos produzidos inicialmente para uma linguagem formal, de tal forma que essa tradução fosse feita automaticamente pelo computador (Bordeleau et al, 1994).

Hrischuk et al (1995) concebeu um modelo para avaliar o desempenho dos sistemas usando protótipos orientados a objeto, combinando abordagens empíricas e preventivas. Operações completas são executadas no protótipo e elas são registradas mantendo os relacionamentos causais. Com isso, previsões mais precisas sobre desempenho são fornecidas conforme o protótipo vai sendo refinado, até a implementação. Contudo o protótipo precisa, pelo menos, conhecer a seqüência de comandos executados pelas entidades concorrentes, quase nunca disponível no início do projeto. Além disso, maximizar o desempenho pode não ser o objetivo principal de um projeto.

Apesar da importância do modelo dinâmico do sistema como um todo na fase inicial da modelagem, não é essencial a sua construção como o primeiro modelo (processo *top-down*). É importante deixar claro que, em certos casos, o modelo dinâmico do sistema pode ser construído num momento posterior, durante o desenvolvimento. O desenvolvedor pode querer inicialmente trabalhar de uma maneira *bottom-up*, devido aos seus conhecimentos e experiência naquele domínio de problema.

Às vezes, a própria especificação dos requisitos está incompleta e imprecisa, e trabalhar primeiro com partes conhecidas do sistema pode ser a solução para melhorar a especificação dos requisitos e caminhar no sentido da compreensão do funcionamento global do sistema. Com isso, partes desconhecidas ou nebulosas do sistema podem ser melhor trabalhadas com o cliente, que nem sempre sabe especificar claramente o que quer.

Isto, contudo, não descarta a necessidade de que um modelo dinâmico do sistema seja construído num momento posterior, possibilitando a visualização e melhor especificação do funcionamento das partes em conjunto com o todo. É com este modelo que gargalos de tempo de resposta, aspectos de concorrência, sincronismo, e assincronismo podem tornar-se evidentes, permitindo que decisões arquiteturais possam ser tomadas de acordo com critérios que venham de encontro às imposições de projeto, tais como desempenho, quantidade de memória RAM (Memória de Acesso Randômico), dispositivos físicos de armazenamento, distribuição física dos equipamentos e reutilização de software e hardware.

2.4 COMPOSIÇÃO DESEJÁVEL DO MODELO DINÂMICO DOS SISTEMAS

A dinâmica de um sistema mostra a sua evolução ao longo do tempo e é geralmente descrita por meio da dinâmica de seus componentes. Esta descrição é feita utilizando-se técnicas variadas, disponíveis na literatura e recomendadas nas metodologias de desenvolvimento de software (Peterson, 1981; Ward, 1986; Jacobson et al, 1992; Selic, 1993; Buhr, 1996; Ambler, 1997).

Porém, a dinâmica de um sistema não pode ser representada somente por meio da modelagem da dinâmica de seus componentes, mas também deve ser considerada a interação destes componentes ao longo do tempo, sujeitos a várias condições ou circunstâncias.

A composição inicial da representação dinâmica de um sistema deveria oferecer a possibilidade da visão conjunta da dinâmica, da estrutura estática e da funcionalidade, num único modelo. Refinamentos posteriores poderiam ser feitos à medida que mais detalhes fossem levantados e considerados durante a modelagem, obtendo-se uma maior compreensão do sistema e tornando possível a posterior confecção de novos modelos que retratassem melhor a fase de projeto.

Buscando a compreensão da dinâmica global dos sistemas, foram levantados alguns elementos considerados elementos-chave para a representação do modelo dinâmico. Este levantamento não teve a pretensão de fechar a questão sobre um conjunto mínimo e

suficiente de elementos-chave que atendam às necessidades colocadas, mas sim acrescentar algo consistente que pudesse contribuir para a melhoria da compreensão.

São eles:

EC1-Seqüências de acontecimentos ou eventos;

EC2-Quando os eventos ou acontecimentos ocorrem no tempo;

EC3-Representação das entidades componentes;

EC4-Fluxos de dados ou entidades (periódico ou aperiódico);

EC5-Transformações ocorridas no decorrer do tempo que afetem a seqüência, que gerem novos componentes e entidades, que os destruam ou que os transformem;

EC6-Paralelismo na ocorrência de dois ou mais conjuntos de eventos ou acontecimentos e paralelismo entre transformações;

EC7-Sincronismo entre dois ou mais conjuntos de eventos ou acontecimentos e sincronismo entre transformações;

EC8-Concorrência por recursos;

EC9-Estabelecimento de pré e pós-condições; e

EC10-Estabelecimento de prioridades.

Para facilitar a representação destes elementos no texto, para cada um deles foi adotado uma palavra mnemônica para a sua identificação, conforme listado a seguir.

EC1-Seqüência

EC2- Eventos no tempo;

EC3-Entidades componentes;

EC4-Fluxos de entidades;

EC5-Transformação;

EC6-Paralelismo;

EC7-Sincronismo;

EC8-Concorrência;

EC9-Pré e pós-condições;

EC10-Prioridade.

Alguns destes elementos-chave não possuem, neste contexto, o significado padrão dado ao termo e serão melhor explicados nas seções seguintes.

2.4.1 EC1-SEQÜÊNCIAS DE ACONTECIMENTOS OU EVENTOS

Como já mencionado anteriormente, os sistemas podem ser vistos como constituídos de **entidades componentes** que colaboram e interagem para atingir um objetivo comum. Estas entidades, por sua vez, podem ser constituídas de outras entidades, resultando na criação de vários níveis de detalhes do sistema.

A representação de uma seqüência de acontecimentos ou eventos do sistema envolve a representação de uma ordem parcial, dado que numa mesma seqüência, pode-se ter eventos e acontecimentos ocorrendo não somente de maneira seqüencial, mas também em paralelo. Várias seqüências de acontecimentos ou eventos podem estar ocorrendo simultaneamente no sistema.

Um **evento** é definido, neste contexto, como algo que afeta o sistema e que está sendo considerado como atômico e instantâneo em relação à escala de tempo utilizada na modelagem. Um evento pode causar uma mudança de estado, disparar outros eventos, ou provocar o início e o término de uma atividade.

Um evento de entrada é enviado por um agente ao sistema; um evento de saída é enviado pelo sistema a um agente. Quando um sistema recebe um evento ele pode causar uma mudança de estado e a saída de outros eventos.

Segundo Rumbaugh (1991), um evento é algo que acontece num instante do tempo, tal como pressionar o botão de um mouse ou a partida de um vôo. Um evento não tem duração. Naturalmente nada é instantâneo e um evento é simplesmente uma ocorrência que é rápida demais comparada com a escala de tempo de uma dada abstração. Os

eventos poderiam ser representados no eixo do tempo como ocorrências pontuais, conforme mostra a Figura 2.1.

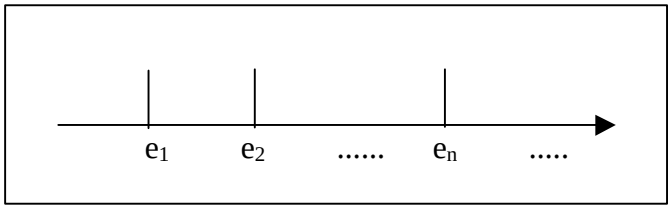

Fig. 2.1 - Eventos no eixo do tempo.

Alguns eventos, como o recebimento e envio de sinais, podem transportar dados, que são considerados seus atributos.

Um evento pode preceder ou suceder logicamente outro, ou dois eventos podem não estar relacionados. Os eventos são paralelos quando um não pode influenciar o outro, ou seja, um não tem efeito sobre o outro.

Um evento poderá ainda acarretar ou provocar um ou mais outros eventos. Estes eventos, por sua vez, poderão provocar novos eventos e assim por diante, formando um encadeamento de eventos.

O encadeamento de eventos é constituído por todas as seqüências de eventos provocadas por um dado evento no transcorrer do tempo. Na modelagem de um sistema não se pode estabelecer uma ordenação entre os eventos paralelos porque eles podem ocorrer em qualquer ordem (Rumbaugh, 1991). Um modelo realístico de um sistema de tempo real e distribuído deve incluir eventos concorrentes e paralelos, pois no mundo real os objetos existem desta forma.

Quanto aos acontecimentos, eles têm na realidade uma duração que pode ser bem pequena (como nanosegundos ou menos) ou grande (como dias, semanas ou meses). Os acontecimentos podem envolver vários eventos e transformações, mas o que importa para a modelagem é sua duração comparada à do evento.

Um acontecimento pode ser considerado um evento se, para o objetivo do sistema, a sua duração puder ser modelada como sendo muito pequena, ou seja, instantânea e atômica. Por exemplo, o recebimento de um pedido de compra através de telefone poderá levar

minutos para se realizar, mas para um sistema Gerenciador de Negócios da empresa, este acontecimento poderá ser considerado um evento. Por outro lado, para um sistema que processa o recebimento de pedidos, talvez isso não seja conveniente. Para este sistema, o início e o término do recebimento poderão ser tomados como eventos distintos.

Os acontecimentos poderiam ser representados no eixo do tempo apresentando alguma duração, não sendo necessariamente importante determinar precisamente qual ou quais eventos originaram o acontecimento ou marcaram seu término. A Figura 2.2 mostra uma possível representação para os acontecimentos.

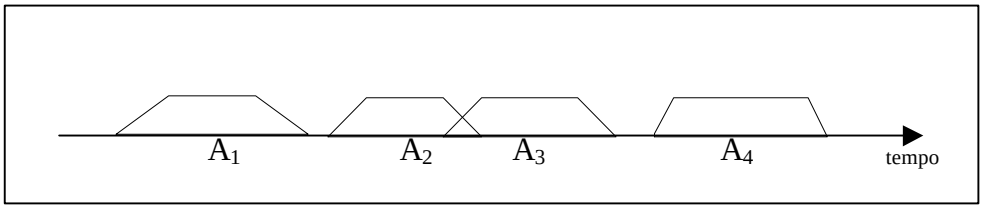

Fig. 2.2 - Acontecimentos no eixo do tempo.

2.4.2 EC2-QUANDO OS EVENTOS OU ACONTECIMENTOS OCORREM NO TEMPO

O segundo elemento-chave levantado envolve a representação de eventos ou acontecimentos no tempo. A representação do tempo pode estar desassociada da representação de seqüências. Deve ser possível representar eventos ou acontecimentos periódicos e aleatórios e também eventos ou acontecimentos temporizados em relação a um outro evento ou acontecimento, ou em relação a uma escala de tempo definida.

2.4.3 EC3-REPRESENTAÇÃO DE ENTIDADES COMPONENTES

O terceiro elemento-chave levantado refere-se à representação das entidades componentes do sistema, esboçadas num alto nível de abstração, formando um pano de fundo sobre o qual a dinâmica irá se desenvolver. Esta representação pode ser vista como a forma estrutural de alto nível de um sistema, acima dos níveis de detalhes internos de seus componentes.

A representação envolve respostas a questões de alto nível, tais como: o número e o tipo de componentes que deveriam estar presentes no sistema; se os componentes deveriam ser agrupados em unidades maiores, em níveis ou em subsistemas; e que tipo de estruturas deveriam conectar os componentes em grupos para manusear responsabilidades de alto nível (tais como: *pipelines;* anéis e redes).

Além disto, deve-se considerar também: quais são as estruturas adicionais necessárias para monitorar as falhas e recuperação destas falhas; se as estruturas dos componentes deveriam ser fixas ou dinâmicas; como as responsabilidades devem ser alocadas entre os componentes da estrutura; que estruturas oferecem melhor desempenho ao longo de caminhos críticos ou são mais robustas na presença de falhas; quais estruturas dão melhor flexibilidade, reutilização e extensibilidade; e qual é a distribuição física dos elementos mais importantes e suas ligações.

2.4.4 EC4-FLUXOS DE DADOS OU ENTIDADES

O quarto elemento-chave levantado, os fluxos de dados ou entidades, mostra entidades ou valores de dados que fluem de uma origem para um destino. A origem e o destino são as entidades componentes que constituem o sistema. Elas podem representar valores intermediários que às vezes não têm significado no mundo real, mas que são importantes para a modelagem do sistema.

2.4.5 EC5-TRANSFORMAÇÕES

Este elemento-chave trata das transformações ocorridas no decorrer do tempo. Estas transformações são atividades executadas pelas entidades, e podem envolver a criação e destruição de outras entidades componentes e dados, processamento automático (por exemplo, a execução de um programa) e processamento manual (por exemplo, a montagem de uma peça).

Os processos computacionais são casos específicos de transformações que fazem o mapeamento de entradas em saídas, por meio de processamentos resultantes de algum comportamento da entidade.

As transformações podem alterar uma determinada seqüência de acontecimentos ou eventos.

2.4.6 EC6-PARALELISMO

O sexto elemento-chave, o paralelismo, envolve a ocorrência simultânea de dois ou mais eventos ou acontecimentos ou ainda a realização simultânea de várias transformações. Isto pode resultar na concorrência por recursos e na necessidade de sincronização entre os eventos, acontecimentos e transformações.

A concorrência e o sincronismo, apesar de estarem fortemente relacionados ao paralelismo, são colocados também como elementos-chave a serem considerados a parte. Isto porque, dependendo do nível de abstração do modelo do sistema sendo trabalhado, o paralelismo pode já não mais aparecer representado explicitamente, mas a concorrência e o sincronismo sim.

Num sistema onde ocorre o paralelismo, o controle pode residir paralelamente em vários componentes e entidades independentes, cada um com uma atividade separada (Rumbaugh, 1991). Uma atividade pode esperar por uma entrada, mas as outras continuam a sua execução. O paralelismo adiciona dificuldades na especificação do funcionamento correto de um sistema.

Em termos de implementação, isto significa que certos tipos de problemas necessitarão de um sistema automático que possa manusear muitos eventos simultâneos. Outros problemas podem envolver tanta computação que a capacidade de um único processador é excedida. Em cada um destes casos pode ser considerado o uso de um conjunto de computadores distribuídos ou o uso de vários processadores capazes de multitarefa. Desta forma, num sistema paralelo, têm-se várias linhas de controle (*thread of control*), algumas transitórias e outras que persistem durante todo o tempo de vida do sistema.

2.4.7 EC7-SINCRONISMO

O sétimo elemento-chave levantado, o sincronismo, ocorre entre dois ou mais conjuntos de eventos e acontecimentos ou entre transformações. A espera por algum evento ou a espera pela utilização em conjunto de um recurso comum, são exemplos de pontos de sincronização que devem ser representados.

O sincronismo de múltiplos conjuntos de eventos ou acontecimentos e o sincronismo entre transformações é também importante devido a questões de concorrência que envolvem a exclusão mútua.

2.4.8 EC8-CONCORRÊNCIA

O oitavo elemento-chave levantado, a concorrência por recursos, pode ocorrer entre dois ou mais conjuntos de entidades, eventos ou acontecimentos, disputando o uso de um recurso do sistema, onde este recurso pode ser um dispositivo físico, um recurso humano ou até uma transformação que deva ser realizada.

A concorrência é também importante devido a questões que envolvem a exclusão mútua. É inadequado permitir que várias entidades, eventos ou acontecimentos atuem simultaneamente sobre o mesmo recurso, porque eles podem interferir no seu estado, causando inconsistências.

2.4.9 EC9-ESTABELECIMENTO DE PRÉ E PÓS-CONDIÇÕES

As pré e pós-condições dizem respeito à satisfação de alguma regra, restrição, ou lógica, que podem ocorrer, respectivamente: antes e depois de uma transformação, da criação e destruição de uma entidade; ou ainda da geração de um evento, pertencente ao modelo dinâmico.

A satisfação da precondição deve ser verificada antes do disparo de uma transformação, da criação ou destruição de uma entidade do modelo e da geração de um evento. A satisfação da pós-condição deve ser garantida após a execução de uma transformação, a criação ou destruição de uma entidade do modelo e a geração de eventos.

2.4.10 EC10-ESTABELECIMENTO DE PRIORIDADES

O estabelecimento de prioridades diz respeito à implementação da ordem apropriada de execução das várias transformações e processos do sistema, garantindo seu correto funcionamento. Freqüentemente, estas informações somente aparecem quando um modelo mais detalhado do sistema é feito, incorporando mais informações sobre seu funcionamento. Porém, quando prioridades são definidas logo no início da modelagem, devem ser consideradas já na definição do modelo inicial.

2.5 ESTUDO DE CASOS

Com a construção de modelos que levem em conta todos estes elementos-chave, acredita-se que a dinâmica e, consequentemente, o comportamento do sistema poderão ser representados de maneira melhor e mais global na modelagem inicial.

Com o objetivo de verificar e reforçar a necessidade dos elementos-chave para a representação do modelo dinâmico descritos anteriormente, foram analisados alguns casos apresentados nas seções seguintes, e obtidas algumas conclusões bastante relevantes.

A idéia principal por trás do estudo dos vários casos foi procurar especificações de sistemas que se apresentassem interessantes e expressivas para uma análise, abrangendo diferentes áreas.

É importante ressaltar que descrições de sistemas, principalmente de grandes sistemas, normalmente nunca são completas, são vagas e muitas vezes contraditórias, suscitando dúvidas quanto ao que estão querendo descrever. Isto ocorreu com as descrições utilizadas nos casos estudados. O gerenciamento da incerteza durante o desenvolvimento é sempre uma preocupação e assim é importante que o desenvolvimento de tais sistemas possa evoluir no tempo de uma maneira incremental e interativa.

Além de as descrições serem incompletas e vagas, existe também o interesse do modelador que pode estar olhando o sistema sobre o enfoque de eventos, dados ou funções, ou ainda uma combinação destes três. É comum que exista então, especificações e modelos iniciais de sistemas que reflitam mais a funcionalidade do sistema, outros que evidenciam mais a modelagem dos dados e ainda outros onde os eventos são os aspectos mais modelados.

Os sistemas em estudo foram analisados com o objetivo geral de se levantar, num primeiro momento, porções do sistema onde a representação da dinâmica e do tempo é necessária.

O termo porção do sistema foi adotado por representar uma idéia mais flexível do que a noção dada com o conceito de partes do sistema. Por parte do sistema subentende-se um

elemento de um todo já dividido, onde a união das partes comporão o todo. Já a porção pode ser considerada como uma parcela ou quantidade limitada do sistema, não significando necessariamente que o sistema já esteja completamente dividido em partes. Neste contexto pode-se, inclusive, considerar que algumas porções do sistema eventualmente se sobreponham.

Também foram levantados nesta análise as possíveis entidades candidatas a fazerem parte da composição do modelo dinâmico.

A análise não foi exaustiva no levantamento de todas as porções do sistema. Apenas procurou-se colocar algumas porções principais que servissem de base para uma verificação e confirmação dos elementos-chave desejáveis na composição dos modelos dinâmicos, levantados na Seção 2.4.

Para que esta análise se tornasse o menos subjetiva possível e válida, considerou-se vários níveis de granulosidade na identificação das porções dinâmicas e comportamentais dos sistemas estudados, onde os níveis de menor granulosidade eqüivalem a refinamentos sucessivos dos níveis anteriores de maior granulosidade.

O primeiro estudo de caso, a Fábrica de Artefatos, representa um domínio de aplicação orientado a manipulação de dados (sistema de informação), mas também possui especificações de controle ligadas ao setor produtivo da fábrica. Em seguida é apresentado o Sistema Integrado de Controle de Satélites, cujo enfoque principal é o recebimento de dados de telemetria de satélite, interpretação de telemetrias e envio de telecomandos para um satélite por meio da Estação Terrena, em tempo real. Finalmente, o terceiro estudo de caso trata de um Sistema de Gerência de Tráfego de Trens, que controla as rotas e localizações de trens num sistema ferroviário, utilizando centros regionais de operação, dispositivos de hardware conectados aos trilhos e um sistema computadorizado a bordo dos trens.

Todos os sistemas analisados têm como premissa uma arquitetura distribuída, uma tendência cada vez mais presente nos sistemas de software e hardware atuais.

2.5.1 PRIMEIRO ESTUDO DE CASO - FÁBRICA DE ARTEFATOS

2.5.1.1 DESCRIÇÃO DO SISTEMA

A descrição deste estudo de caso foi baseada em (Sant'Anna, 1993).

Uma fábrica de pequeno porte produz artefatos a partir de chapas metálicas. Ela possui três setores básicos: Setor de Vendas, Setor Produtivo e Setor de Compras, conforme mostrado na Figura 2.3.

No Setor de Vendas, os vendedores recebem pedidos dos clientes, verificam as características de crédito desses clientes, e confirmam o pedido caso não haja nenhum problema. Ao confirmar o pedido, o vendedor estabelece um prazo para o atendimento do pedido e emite uma solicitação de produção para o setor produtivo.

Um pedido, de um dado cliente, só poderá ser recebido por um único vendedor, e este será o encarregado de entregar o produto ao cliente, assim que terminar a sua fabricação.

O Setor Produtivo possui um conjunto de Unidades de Fabricação, cada uma delas com capacidade de fabricar artefatos metálicos de forma quase totalmente automatizada. Para seu trabalho, a unidade precisa ser carregada com os parâmetros adequados, extraídos de uma ordem de produção e receber o comando de início de operação.

Uma unidade de fabricação é composta de uma máquina operatriz e de um robô. O robô controla a máquina operatriz com base nos valores de parâmetros que lhes são fornecidos por um operador através de um terminal de vídeo.

O robô trata da alimentação da máquina operatriz com chapas necessárias para a fabricação de artefatos e controla a quantidade dos produtos, aceitando ou rejeitando o artefato fabricado.

Além disso, para cada ordem de produção, o robô mantém informações a respeito de sua operação e a respeito da fabricação, tais como: os seus parâmetros colocados para atender a ordem de produção; instantes de início, de interrupções e de término da operação; e número de artefatos fabricados e rejeitados.

A alocação da unidade de fabricação para atender uma determinada ordem de produção é feita pelo encarregado da produção. Este funcionário é também responsável pela transformação de solicitações em ordens de produção.

O transporte das chapas do almoxarifado para a área de materiais da unidade é realizado por uma empilhadeira.

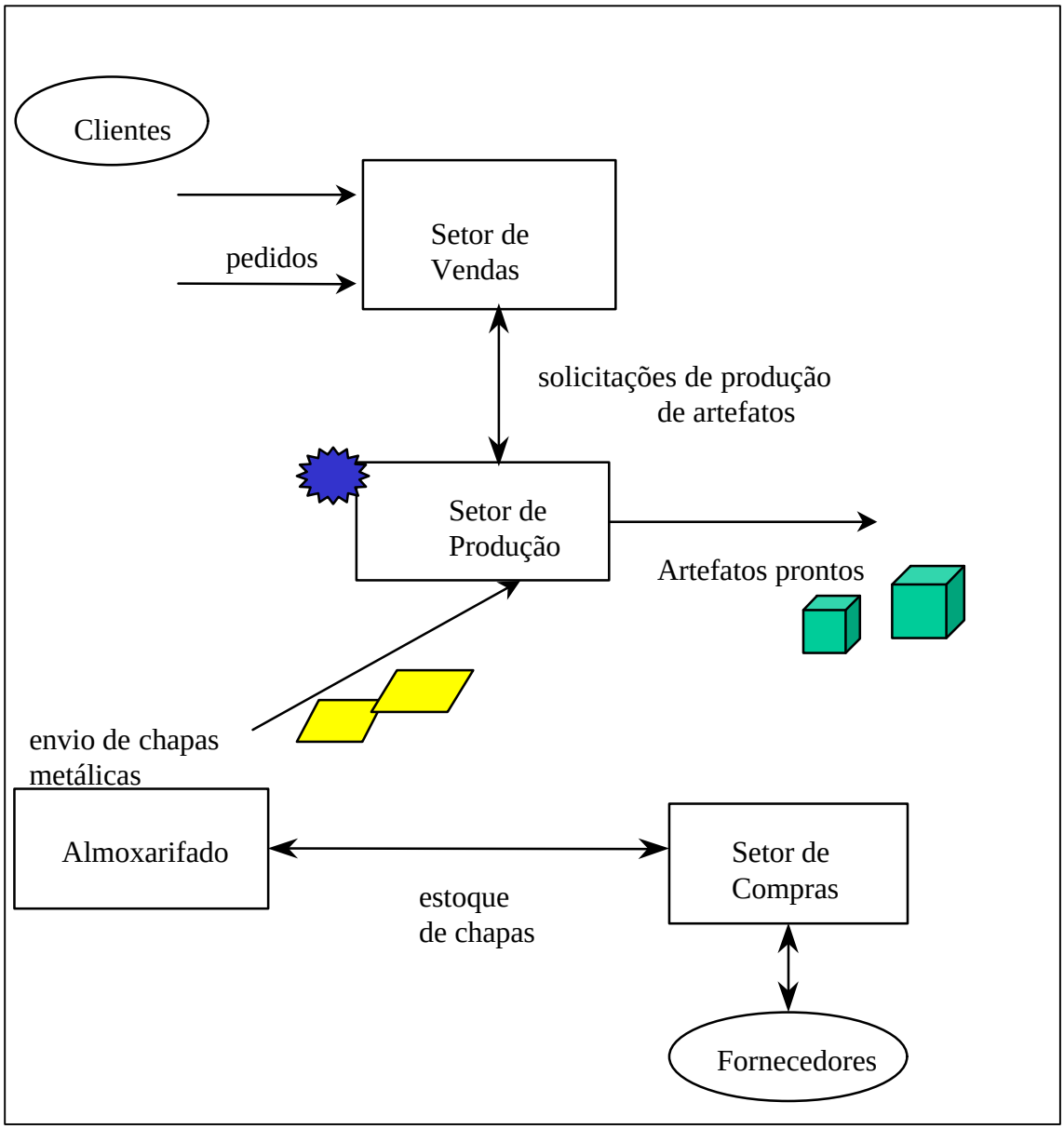

Fig. 2.3 - Esquema geral da Fábrica de Artefatos.

O Setor de Compras mantém o estoque de chapas do almoxarifado seguindo uma política de estoque mínimo. Empregados desse setor consultam os fornecedores quanto

ao tipo de chapas que eles fornecem, preços e prazos de entrega. Após a análise das opções, escolhem a melhor e emitem o pedido de compras.

Um sistema de gerenciamento da informação deverá ser desenvolvido com a finalidade de melhorar os trabalhos dos setores de Vendas e de Compras, melhorar o monitoramento das diversas Unidades de Produção, facilitar o trabalho de elaboração de ordens de produção e manter o gerente informado sobre todos os setores da fábrica, reduzindo os atuais desperdícios.

Para os vendedores, ele deverá fornecer as informações necessárias para as atividades de aceitação de pedidos e auxiliar na elaboração das solicitações de produção. O encarregado do Setor de Produção deverá ter à sua disposição as informações que o auxilie no acompanhamento da produção propriamente dita e na alocação das Unidades de Fabricação. Para isso, o sistema deverá ser conectado ao robô de cada uma das Unidades de Fabricação para facilitar a carga dos parâmetros e coleta dos dados monitorados.

O Setor de Compras deverá ter à sua disposição as informações atualizadas dos níveis de estoque de chapas no almoxarifado, os valores dos estoques mínimos dos diversos tipos de chapas e as informações sobre os fornecedores. Além disso, o Setor deverá ter facilidades para a emissão dos pedidos de compra.

O gerente deseja ter acesso às informações sobre:

- Cadastro de clientes e a satisfação desses clientes;
- Pedidos e artefatos pedidos;
- Produção, unidades alocadas e dados sobre produção e rejeição;
- Entradas e saídas de operação dos robôs e taxas de falhas das Unidades de Fabricação;
- Características técnicas dos artefatos e preços;
- Características técnicas das chapas e quantidades em estoque; e
- Tipos de chapas dos fornecedores cadastrados, preços e prazos de entrega.

2.5.1.2 IDENTIFICAÇÃO DA DINÂMICA E DO COMPORTAMENTO

Nesta Seção são identificadas as principais porções do funcionamento do sistema onde a representação da dinâmica e do comportamento seriam necessárias. Neste primeiro estudo de caso a modelagem foi feita variando-se os níveis de detalhes utilizados, para que um estudo comparativo entre os vários níveis pudesse ser feito em relação à utilização dos elementos-chave da representação, levantados na Seção 2.4.

Pela descrição, os três setores do sistema necessitam automatizar o trabalho, tanto para um melhor controle da fábrica ser feito pela gerência, como para melhorar a eficiência e o trabalho dos três setores: vendas, produção e compras. Pode-se observar que o Setor de Produção já está parcialmente automatizado com o uso de Máquinas Operatrizes e robôs. Porém a Ordem de Produção gerada pelo encarregado da produção ainda é feita manualmente. Partindo-se do pressuposto que a parte já automatizada esteja funcionando bem, o objetivo agora é modelar as partes dos setores cujo trabalho precisa ser mais eficiente e agilizado.

Com as informações fornecidas na descrição do sistema, foram levantadas porções dinâmicas e comportamentais considerando três níveis diferentes de abstração dos detalhes. Estes níveis são apresentados a seguir em ordem decrescente de granulosidade, referenciados como nível 1, nível 2 e nível 3, respectivamente. O nível 2 foi construído acrescentando-se mais informações às porções do nível 1, não havendo para isto nenhum critério específico. Algumas funções não foram consideradas no nível 1 propositadamente, sendo somente consideradas no nível 2. Já o nível 3 detalha cada uma das porções levantadas no nível 2.

As tabelas que acompanham este levantamento têm como objetivo relacionar as porções dinâmicas e comportamentais do sistema da Fábrica de Artefatos e a representação destas porções em termos dos elementos-chave candidatos a representarem o modelo dinâmico.

Nível 1:

1) Chegada de pedidos de clientes ao Setor de Vendas, solicitando a compra de artefatos. Aceitação ou não dos pedidos, e em caso de aceitação, confirmação para os clientes da data de entrega dos mesmos;

2) Para cada novo pedido recebido, o Setor de Vendas deve confeccionar uma solicitação de produção e encaminhá-la para o Setor de Produção;

3) Setor de Produção gera as ordens de produção necessárias para os robôs e monitora o processo de produção;

4) O Almoxarifado envia chapas para o Setor de Produção conforme solicitação deste;

5) Setor de Produção avisa o Setor de Vendas do término da produção de uma determinada solicitação; e

6) Setor de Compras faz consultas periódicas ao Almoxarifado para informar-se dos níveis atuais de estoque, tomando providências para mantê-lo no mínimo exigido.

A Tabela 2.1 mostra as porções levantadas no nível 1 e a sua relação com os elementos-chave.

A porção 1 envolve seqüência na chegada e confirmações de pedidos para os clientes. O instante em que os eventos ocorrem não importa neste momento. Fica claro a existência de um Setor de Vendas, uma entidade componente. Não existe transformações aparentes, mas existe paralelismo na chegada de pedidos. Para a aceitação do pedido, existe a precondição de que o cliente tenha crédito, e como pós-condição, a confirmação do pedido deve ser enviada para o cliente. Quanto ao estabelecimento de prioridades, podem existir clientes preferenciais e uma prioridade maior pode ser atribuída no atendimento do seu pedido; ou ainda podem existir pedidos que devam ser atendidos mais rapidamente do que outros.

A análise da porção 2 é análoga à da porção 1, porém há transformações de pedidos recebidos em solicitações de produção, e não há pré e pós-condições.

TABELA 2.1 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 1

	Porções Levantadas					
Elementos-chave	**1**	**2**	**3**	**4**	**5**	**6**
EC1-Seqüência	x	x	x	x	-	x
EC2-Eventos no tempo	-	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x
EC5-Transformação	-	x	x	-	-	-
EC6-Paralelismo	x	x	x	-	-	-
EC7-Sincronismo	-	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-	-
EC9-Pré e pós-condições	x	-	-	-	-	-
EC10-Prioridade	x	-	-	-	-	-

Na porção 3, o Setor de Produção executa as atividades de geração de ordens de produção e monitoramento do processo de produção paralelamente e não seqüencialmente. A geração de ordens de produção envolve transformações e fluxo de dados. O monitoramento do processo produtivo também pode ser visto neste momento como um processo automático, sendo representado por uma transformação.

Na porção 4, as chapas são enviadas do Almoxarifado para o Setor de Produção, envolvendo somente fluxo de dados e entidades e a representação do almoxarifado. Raciocínio análogo é feito para a porção 5.

A porção 6 envolve a seqüência consulta e tomada de providências para manter o estoque dentro dos níveis mínimos aceitáveis. Também estão envolvidas entidades componentes (Almoxarifado e Setor de Compras) e fluxos de dados envolvidos nas consultas.

Nível 2:

1) Recebimento, pelos vendedores do Setor de Vendas, de pedidos de clientes, seguido de uma consulta de crédito, confecção e confirmação do pedido para o cliente com a data prevista de entrega ou a não confirmação do pedido, privilegiando os clientes preferenciais;

2) Emissões de solicitações de produção pelos vendedores para o encarregado da produção, que fica no Setor de Produção; estas solicitações podem envolver prioridades no atendimento (cliente preferencial, por exemplo);

3) Transformação das solicitações de produção em ordens de produção pelo encarregado da produção;

4) Alocação de unidades de fabricação pelo encarregado da produção para atender ordens de produção, enviando parâmetros através de um terminal de vídeo ligado aos robôs;

5) Almoxarifado envia chapas para o Setor de Produção, conforme solicitação deste;

6) Controle pelo robô da alimentação da máquina operatriz com chapas necessárias para a fabricação de artefatos e controle da produção, aceitando e rejeitando o artefato fabricado;

7) Envio dos parâmetros da produção pelo robô para o encarregado da produção (ligação física);

8) Aviso do encarregado da produção para o vendedor do Setor de Vendas que o(s) produto(s) de um ou mais pedidos feitos por ele estão prontos (monitoração das unidades de fabricação);

9) Se o estoque estiver abaixo do mínimo requerido, os empregados do Setor de Compras consultam os fornecedores quanto: ao tipo de chapas, preços e prazos de entrega; analisam as opções de compra; escolhem a melhor; e emitem o pedido de compra de chapas; e

10) Acesso pela gerência aos dados de interesse armazenados nos Setores de Venda, Produção e Compra.

A Tabela 2.2 a seguir mostra a relação entre as porções levantadas e os elementos-chave.

TABELA 2.2 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 2

	Porções Levantadas									
Elementos-chave	1	2	3	4	5	6	7	8	9	10
EC1-Seqüência	x	x	-	x	x	x	-	-	x	-
EC2-Eventos no tempo	-	-	-	-	-	-	-	-	-	-
EC3-Entidades componentes	x	x	-	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x	x	x
EC5-Transformação	x	x	x	-	-	x	-	-	-	-
EC6-Paralelismo	x	x	-	-	-	x	-	-	-	x
EC7-Sincronismo	-	-	-	x	-	x	-	-	-	-
EC8-Concorrência	x	-	x	-	-	-	-	-	-	-
EC9-Pré e pós-condições	x	-	-	x	-	x	x	-	x	-
EC10-Prioridade	x	x	x	-	-	-	-	-	-	-

Na porção 1 existe uma seqüência (receber, consultar, aceitar ou não) a ser representada e há concorrência pelo recurso vendedor (precondição: cliente ter crédito; pós-condição: envio para o cliente da confirmação do pedido).

Na porção 2, existe uma seqüência de envio das solicitações de produção, que pode envolver prioridades.

A porção 3 (transformações das solicitações de produção) não envolve seqüência, por tratar-se de um processamento localizado, e envolve concorrência por um recurso

(encarregado da produção), fluxo de solicitações e estabelecimento de prioridades para as ordens de acordo com as solicitações.

A porção 5 envolve a seqüência solicitação e envio de chapas do almoxarifado para o Setor de Produção.

As porções 7 e 8 referem-se ao envio de mensagens transportando dados, sem uma seqüência pré-definida. Na porção 7 existe a precondição de que o encarregado da produção deve estar conectado ao robô para receber os parâmetros. Já a porção 9 envolve uma seqüência de acontecimentos e a precondição de que o estoque deve estar abaixo do mínimo.

A alocação das unidades de fabricação e envio de parâmetros do encarregado de produção para estas unidades, representados na porção 4, devem ser feitos de maneira sincronizada.

A porção 6, controle pelo robô da máquina operatriz e da produção de artefatos é uma porção que envolve a produção de artefatos (transformações). Envolve, também, uma ordem parcial, paralelismo e sincronismo, além da pós-condição de inspecionar os artefatos produzidos.

A porção 10 representa o acesso aleatório da gerência aos dados (fluxo de dados) armazenados nos diversos setores da fábrica (entidades componentes e paralelismo).

Nível 3 (detalhamento da porção 1 do nível 2 - Recebimento de pedidos de clientes):

1) Contato do cliente com um vendedor da empresa, pessoalmente, via telefone ou fax, para fazer a solicitação de um pedido de compra de artefatos;

2) Dado um pedido de compra de artefatos de um cliente, o vendedor responsável pelo pedido faz contato com o setor de crédito para verificar a credibilidade deste cliente;

3) Caso haja problema com o crédito de um determinado cliente, o vendedor responsável faz contato com o cliente avisando-o que o pedido foi rejeitado por falta de crédito e o contato é encerrado.

4) Caso não haja problema com o crédito do cliente, o vendedor pode estabelecer uma prioridade para o pedido. Em seguida o vendedor consulta o encarregado da produção para o estabelecimento de um prazo de entrega do pedido. O encarregado da produção, baseado na quantidade de artefatos pedidos e na prioridade do pedido, avalia a disponibilidade das unidades de fabricação e o estoque de chapas no almoxarifado, retornando ao vendedor uma possível data de entrega para o pedido; e

5) vendedor, ao receber do Setor de Produção a data de entrega prevista de um determinado pedido, faz contato com o cliente para confirmar o pedido e passar a referida data. O contato é encerrado.

A Tabela 2.3 ilustra a relação entre os elementos-chave e o detalhamento da porção 1 do nível 2.

TABELA 2.3 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 1 DO NÍVEL 2)

	Porções Levantadas				
Elementos-chave	1	2	3	4	5
EC1-Seqüência	-	-	-	x	x
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x
EC5-Transformação	x	-	-	x	-
EC6-Paralelismo	x	-	-	-	-
EC7-Sincronismo	x	-	x	-	x
EC8-Concorrência	x	-	-	x	-
EC9-Pré e pós-condições	-	-	x	-	x
EC10-Prioridade	-	-	-	x	-

A representação de entidades componentes e fluxo de dados ou entidades aparecem em todas as porções levantadas. Nelas sempre são mencionados setores da fábrica (Vendas, Produção e Compras), representando entidades componentes do sistema.

As transformações realizadas são a confecção de pedidos, identificada na porção 1, e o processamento de informações que o sistema deve fazer para prever para o encarregado da produção a possível data de entrega de um pedido de artefatos, identificado na porção 4.

Na porção 1 podem ocorrer contatos simultâneos de clientes, estabelecidos em sincronia com os vendedores. Isto pode gerar, além do paralelismo, concorrência por recursos (vendedores).

As porções 2 e 3 são análogas, com exceção do sincronismo que ocorre entre clientes e vendedores na porção 3 e a precondição que determina problemas de crédito com o cliente.

A porção 4 envolve a seqüência consulta do vendedor, análise das informações e retorno da data de entrega, o estabelecimento de prioridades nos pedidos, processamento de informações necessárias para o estabelecimento de uma data de entrega (transformação) e concorrência pelo recurso encarregado da produção.

A porção 5 pode ser vista como o envio de uma mensagem para o cliente, de maneira síncrona (por telefone), transportando dados a respeito de seu pedido, e o estabelecimento da pós-condição que determina o envio da confirmação do pedido para o cliente.

Nível 3 (detalhamento da porção 2 do nível 2 – Emissões de solicitações de produção):

1) Os vendedores com as informações contidas em um pedido de compras confirmado de um cliente, acionam o sistema para a preparação da solicitação de produção. As informações necessárias são fornecidas pelo vendedor ao sistema (tipo de artefato, quantidade, data de entrega, etc) e uma prioridade pode ser atribuída a ela. A solicitação pronta é armazenada pelo sistema; e

2) As solicitações de produção armazenadas pelo sistema e ainda não enviadas ao encarregado da produção são enviadas automaticamente, fazendo com que o encarregado da produção receba em seu terminal um aviso sonoro.

A Tabela 2.4 ilustra a relação entre os elementos-chave e as novas porções levantadas.

TABELA 2.4 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 2 NÍVEL 2)

	Porções Levantadas	
Elementos-chave	1	2
EC1-Seqüência	x	x
EC2-Eventos no tempo	-	-
EC3-Entidades componentes	x	x
EC4-Fluxos de entidades	x	x
EC5-Transformação	x	-
EC6-Paralelismo	x	-
EC7-Sincronismo	-	-
EC8-Concorrência	-	-
EC9-Pré e pós-condições	-	-
EC10-Prioridade	x	-

A porção 1 envolve a seqüência acionar o sistema, preparar e criar a solicitação de produção (seqüência de eventos e acontecimentos e transformação), fluxo de informações fornecidas pelo vendedor, entidades componentes (Setor de Vendas e sistema computacional), paralelismo (vários vendedores podem estar preparando solicitações), e o estabelecimento de prioridades no atendimento das solicitações.

A porção 2 trata do envio e aviso pelo sistema de solicitações de produção prontas ao encarregado da produção, envolvendo seqüência de eventos (armazenamento das ordens

e posterior envio), arquitetura do sistema (conexão física ou lógica entre os setores de Vendas, Produção e Compras) e fluxo de informação.

Nível 3 (detalhamento da porção 3 do nível 2 – Geração de ordens de produção):

1) Uma vez que as solicitações de produção cheguem ao encarregado da produção, este avalia a disponibilidade das unidades de fabricação, o cronograma das ordens de produção pendentes e o estoque de chapas no almoxarifado; e

2) Baseado nas informações sobre a disponibilidade das unidades de fabricação e o estoque de chapas no almoxarifado, e também no tipo e quantidade de artefatos a serem produzidos, o encarregado da produção prepara e agenda as ordens de produção com a informação da unidade em que elas serão executadas e uma prioridade; estas ordens são armazenadas pelo sistema.

A Tabela 2.5 mostra a relação entre os elementos-chave e as novas porções levantadas.

TABELA 2.5 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 3 NÍVEL 2)

	Porções Levantadas	
Elementos-chave	1	2
EC1-Seqüência	x	x
EC2-Eventos no tempo	-	-
EC3-Entidades componentes	-	-
EC4-Fluxos de entidades	-	x
EC5-Transformação	x	x
EC6-Paralelismo	-	-
EC7-Sincronismo	-	-
EC8-Concorrência	-	-
EC9-Pré e pós-condições	-	x
EC10-Prioridade	-	x

A porção 1 envolve a seqüência de atividades de verificação de vários itens pelo encarregado da produção para coletar informações para a preparação das ordens de produção.

Na porção 2, o encarregado da produção por meio de uma seqüência de atividades, cria as ordens de produção (transformação, fluxo de dados ou entidades), estabelece uma prioridade de execução e agenda estas ordens. Em seguida, o controle é passado para o sistema computacional que fará o armazenamento destas novas informações.

Nível 3 (detalhamento da porção 4 do nível 2 – Alocação de Unidades de Fabricação):

1) Quando uma unidade de fabricação fica desocupada, o encarregado da produção verifica se existe alguma ordem de produção pendente que deva ser alocada para esta unidade. Caso exista, ele conecta-se ao robô responsável pela unidade via terminal; e

2) Uma vez estabelecido o contato com o robô de uma unidade de produção via terminal, o encarregado da produção envia os parâmetros de uma determinada ordem de produção para o robô, obedecendo o cronograma estabelecido.

A Tabela 2.6 ilustra a relação entre os elementos-chave e as novas porções levantadas.

A porção 1 retrata a seqüência de conexão de uma unidade de fabricação disponível com o robô, envolvendo as entidades componentes da conexão. Esta conexão é síncrona e envolve a precondição de que a unidade de fabricação esteja desocupada.

A porção 2 representa a seqüência de envio de parâmetros (fluxo de dados) de uma determinada ordem de produção para o robô, envolvendo entidades componentes e sincronismo entre o robô e o terminal do encarregado da produção, e como precondição o robô deve estar pronto para receber os parâmetros da produção.

TABELA 2.6 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 4 NÍVEL 2)

	Porções Levantadas	
Elementos-chave	1	2
EC1-Seqüência	x	x
EC2-Eventos no tempo	-	-
EC3-Entidades componentes	x	x
EC4-Fluxos de entidades	-	x
EC5-Transformação	-	-
EC6-Paralelismo	-	-
EC7-Sincronismo	x	x
EC8-Concorrência	-	-
EC9-Pré e pós-condições	x	x
EC10-Prioridade	-	-

Nível 3 (detalhamento da porção 5 do nível 2 – Envio de chapas):

1) Antes de uma unidade de fabricação entrar em operação, o encarregado da produção verifica se ela já possui em sua área de armazenamento o número de chapas necessárias para a sua produção;

2) Caso o número de chapas que uma unidade de fabricação possui em sua área de armazenamento não seja suficiente para sua produção, o encarregado da produção solicita a quantidade de chapas necessárias ao almoxarifado; e

3) Diante da chegada de novos pedidos de chapas para as unidades de fabricação, o almoxarifado verifica o estoque e então providencia o transporte das chapas para as referidas unidades. Caso não existam chapas ou transporte disponíveis, os pedidos de chapas entram numa fila de espera.

A Tabela 2.7 mostra a relação das porções identificadas neste detalhamento com os elementos-chave.

As porções 1 e 2 retratam os eventos que disparam a verificação de chapas disponíveis nas unidades de fabricação e a solicitação de chapas para estas unidades envolvendo entidades componentes e fluxo de dados.

A porção 3 envolve seqüência de eventos, dado pelo paralelismo no recebimento de vários pedidos pelo almoxarifado (entidades componentes, fluxo de dados) e pela saída de chapas do almoxarifado. Há ainda a concorrência pelo recurso de transporte das chapas.

TABELA 2.7 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 5 NÍVEL 2)

	Porções Levantadas		
Elementos-chave	1	2	3
EC1-Seqüência	-	-	x
EC2-Eventos no tempo	-	-	-
EC3-Entidades componentes	x	x	x
EC4-Fluxos de entidades	x	x	x
EC5-Transformação	-	-	-
EC6-Paralelismo	-	-	x
EC7-Sincronismo	-	-	-
EC8-Concorrência	-	-	x
EC9-Pré e pós-condições	-	-	-
EC10-Prioridade	-	-	-

Nível 3 (detalhamento da porção 6 do nível 2 – Controle da produção):

1) Assim que uma determinada máquina operatriz termina o processamento de uma chapa para a produção de artefatos e ela necessita de novas chapas, ela envia um sinal para o robô (evento) e o robô insere uma nova chapa na máquina;

2) À medida que os artefatos vão sendo produzidos, eles vão sendo colocados numa fila para inspeção, e os dados de produção são armazenados num registro de parâmetros da produção pelo robô (arquivo log da produção);

3) Se o robô da unidade de fabricação não estiver ocupado alimentando a máquina operatriz com chapas metálicas, ele avalia o primeiro artefato da fila de inspeção e, baseado em critérios de qualidade pré-estabelecidos, aceita o artefato ou o rejeita;

4) Caso o artefato seja aprovado pela inspeção do robô, este recebe o carimbo "aceito" e é colocado na fila de embalagem pelo robô. O robô registra esta informação; e

5) Caso o artefato não seja aprovado pela inspeção, ele é carimbado como rejeitado e é colocado numa fila de artefatos rejeitados. O robô registra esta informação.

A Tabela 2.8 ilustra a relação entre os elementos-chave e as novas porções levantadas.

Todas as porções levantadas fazem parte de uma seqüência de eventos e acontecimentos, com fluxo de dados e entidades (chapas, artefatos e informações de produção).

Na porção 1, o robô deve estar apto para carregar chapas na máquina operatriz (precondição).

A porção 2 envolve a fabricação de artefatos (transformação).

A porção 3 retrata o processo de inspeção dos artefatos produzidos pelo robô, envolvendo uma seqüência de ações do robô e fluxo de artefatos. Este processo de inspeção pode ser interrompido para que o robô possa alimentar a máquina operatriz com chapas metálicas (fluxo de chapas). O robô deve estar desocupado (precondição) para fazer a inspeção dos artefatos.

TABELA 2.8 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 6 NÍVEL 2)

	Porções Levantadas				
Elementos-chave	1	2	3	4	5
EC1-Seqüência	x	x	x	x	x
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x
EC5-Transformação	-	x	-	x	x
EC6-Paralelismo	-	x	-	-	-
EC7-Sincronismo	x	x	x	x	x
EC8-Concorrência	-	-	-	-	-
EC9-Pré e pós-condições	x	-	x	x	x
EC10-Prioridade	-	-	-	-	-

A porção 4 e 5 envolve uma seqüência de ações realizadas pelo robô para a colocação de rótulo no artefato produzido (precondição: aceito ou precondição: rejeitado) após a inspeção. A colocação do rótulo que pode ser vista como uma transformação que altera o estado do artefato, envolvendo além de entidades componentes, o fluxo de artefatos de uma fila para outra.

Em todas as porções levantadas, o robô trabalha em sincronismo com a máquina operatriz.

Nível 3 (detalhamento da porção 7 do nível 2 – Envio dos parâmetros da produção):

1) À medida que os artefatos são produzidos, o robô, após a inspeção, armazena quais deles são aceitos e quais são rejeitados no arquivo de registros dos parâmetros de produção da unidade;

2) Caso ocorra algum problema na produção (quebra da máquina, falta de chapas, etc), estas informações são armazenadas pelo robô no arquivo de registros dos parâmetros de produção da unidade; e

3) O arquivo de registros dos parâmetros de produção da unidade é enviado periodicamente ao encarregado da produção ou por solicitação deste, para fins de monitoração do processo de produção.

A Tabela 2.9 relaciona as porções levantadas neste detalhamento com os elementos-chave.

TABELA 2.9 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 7 NÍVEL 2)

	Porções Levantadas		
Elementos-chave	1	2	3
EC1-Seqüência	x	-	-
EC2-Eventos no tempo	-	-	-
EC3-Entidades componentes	x	x	x
EC4-Fluxos de entidades	x	x	x
EC5-Transformação	x	x	-
EC6-Paralelismo	x	-	-
EC7-Sincronismo	x	-	-
EC8-Concorrência	-	-	-
EC9-Pré e pós-condições	-	x	x
EC10-Prioridades	-	-	-

A porção 1 retrata a seqüência de produção de artefatos pela máquina operatriz e a inspeção pelo robô da produção (fluxo de artefatos, entidades componentes, transformações e sincronismo entre robô e máquina operatriz). Ao mesmo tempo que o

robô armazena os parâmetros da produção localmente (*log* da produção), ele inspeciona os artefatos produzidos.

A porção 2 retrata o armazenamento pelo robô dos parâmetros da produção (*log* da produção) localmente, em casos de falhas (precondição). Isto envolve a representação de entidades componentes, fluxo de informações e transformações. As transformações referem-se ao surgimento de novos valores de parâmetros e seu posterior armazenamento, como também a eventuais procedimentos que devam ocorrer em casos de falhas (precondição: parada de máquina, substituição da máquina quebrada, etc).

A porção 3 retrata o envio de um arquivo de dados (*log* da produção) do robô para o encarregado da produção (precondição: robô e encarregado da produção estarem conectados), envolvendo entidades componentes (conexões físicas) e fluxo de dados.

Nível 3 (detalhamento da porção 8 do nível 2 – Monitoração das Unidades de Fabricação):

1) Ao término da produção de artefatos relacionados a uma determinada ordem de produção (lote de artefatos), o encarregado da produção solicita ao sistema a verificação das solicitações de produção pendentes;

2) Caso uma ou mais solicitações de produção pendentes tornem-se completas a partir da produção do último lote de artefatos, o encarregado da produção estabelece que estas estão completas (não mais pendentes);

3) O sistema que armazena as solicitações de produção analisa as solicitações completadas e mostra para o encarregado da produção quais são os vendedores responsáveis por elas;

4) O encarregado da produção autoriza o sistema a enviar um aviso para o terminal dos vendedores, informando que determinada solicitação feita por ele está pronta para ser entregue ao cliente; e

5) O vendedor, a partir do recebimento de um aviso, faz contato telefônico com o cliente avisando-o que o pedido está pronto para ser retirado e encaminha a cobrança do mesmo ao cliente.

A Tabela 2.10 ilustra a relação entre os elementos-chave e as novas porções levantadas.

A porção 1 retrata a seqüência término da produção de uma ordem e verificação das solicitações pendentes pelo sistema, que envolve a representação da seqüência e das entidades componentes (unidade de produção, sistema computacional e encarregado da produção).

A porção 2 retrata a alteração do estado das ordens de produção armazenadas pelo sistema, envolvendo transformações.

A porção 3 envolve a seqüência de armazenamento e processamento das solicitações de produção, onde entidades componentes, fluxo de solicitações e transformações devem ser representados.

TABELA 2.10 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 8 NÍVEL 2)

	Porções Levantadas				
Elementos-chave	1	2	3	4	5
EC1-Seqüência	x	-	x	-	x
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	-	x	x	x
EC4-Fluxos de entidades	-	-	x	x	x
EC5-Transformação	-	x	x	-	-
EC6-Paralelismo	-	-	-	-	-
EC7-Sincronismo	-	-	-	-	x
EC8-Concorrência	-	-	-	-	-
EC9-Pré e pós-condições	-	-	-	-	-
EC10-Prioridades	-	-	-	-	-

A porção 4 retrata o envio de mensagem com dados do Setor de Produção para o Setor de Vendas (entidades componentes e fluxo de dados). A porção 5 retrata a seqüência aviso do sistema de solicitação pronta, contato telefônico com o cliente e encaminhamento de cobrança para o mesmo, envolvendo entidades componentes, fluxo de dados e sincronismo no contato com o cliente.

Nível 3 (detalhamento da porção 9 do nível 2 – Compra de Chapas):

1) Quando o estoque de chapas no almoxarifado estiver abaixo de um mínimo exigido, o sistema avisa o Setor de Compras;

2) Com o estoque abaixo do mínimo, os empregados do Setor de Compras consultam os fornecedores cadastrados e estes encaminham suas propostas; e

3) Com as propostas dos fornecedores disponíveis, os empregados do setor de compras analisam as opções de tipo de chapa, preço e prazo de entrega, escolhem a melhor opção e emitem o pedido de compra.

A Tabela 2.11 mostra a relação entre as porções identificadas neste nível de detalhamento e os elementos-chave.

A porção 1 retrata a seqüência verificação dos níveis de estoque de chapas pelo almoxarifado e envio de mensagem para o Setor de Compras reportando os níveis de estoque (entidades componentes e fluxo de dados). Precondição: estoque abaixo do mínimo estipulado.

A porção 2 retrata a seqüência contato com o fornecedor de chapas por meio do Setor de Compras e envio de propostas dos fornecedores para o Setor de Compras, envolvendo fluxo de dados, entidades componentes e paralelismo no envio das propostas.

A porção 3 retrata a avaliação das propostas dos fornecedores (precondição: propostas disponíveis) e a emissão do pedido de compras. Isto envolve fluxo de dados e transformações (avaliação das propostas e criação do pedido de compras).

TABELA 2.11 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 9 NÍVEL 2)

	Porções Levantadas		
Elementos-chave	1	2	3
EC1-Seqüência	x	x	x
EC2-Eventos no tempo	-	-	-
EC3-Entidades componentes	x	x	-
EC4-Fluxos de entidades	x	x	x
EC5-Transformação	-	-	x
EC6-Paralelismo	-	x	-
EC7-Sincronismo	-	-	-
EC8-Concorrência	-	-	-
EC9-Pré e pós-condições	x	-	x
EC10-Prioridades	-	-	-

Nível 3 (detalhamento da porção 10 do nível 2 – Consultas da Gerência):

1) Os Setores de Venda, Produção e Compras enviam mensalmente relatórios para a gerência informando o movimento do mês; e

2) A gerência pode, a qualquer momento, consultar a base de dados dos Setores de Vendas, Produção e Compras e realizar sobre estes dados vários processamentos para extrair informações de interesse.

A Tabela 2.12 mostra a relação entre os elementos-chave e as novas porções levantadas.

TABELA 2.12 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DA FÁBRICA DE ARTEFATOS - NÍVEL 3 (PORÇÃO 10 NÍVEL 2)

	Porções Levantadas	
Elementos-chave	1	2
EC1-Seqüência	-	-
EC2-Eventos no tempo	-	-
EC3-Entidades componentes	x	x
EC4-Fluxos de entidades	x	x
EC5-Transformação	-	x
EC6-Paralelismo	x	-
EC7-Sincronismo	-	-
EC8-Concorrência	-	x
EC9-Pré e pós-condições	-	-
EC10-Prioridades	-	-

A porção 1 retrata o envio periódico de relatórios para a gerência, partindo do Setor de Compras, Vendas e Produção. Isto envolve a representação de fluxo de dados e entidades componentes. Além disso, os relatórios podem ser enviados paralelamente pelos setores à gerência.

A porção 2 retrata o acesso da gerência aos dados do Setor de Vendas, Compras e Produção, que pode ocorrer a qualquer momento, dependendo das necessidades da gerência. A gerência pode processar estes dados para extrair informações de interesse (transformações).

2.5.1.3 ENTIDADES CANDIDATAS A SEREM MODELADAS.

A seguir são listadas as entidades componentes do Sistema da Fábrica de Artefatos, identificadas durante o levantamento das porções, candidatas a fazerem parte da representação da dinâmica do sistema.

- Setor de Vendas;
- Setor de Produção;
- Setor de Compras;
- Pedidos de compra de chapas;
- Pedido de compras de artefatos;
- Solicitação de produção e Ordem de produção;
- Almoxarifado;
- Robôs e máquinas operatrizes;
- Encarregado da produção;
- Vendedor;
- Empregado do setor de compras;
- Chapa;
- Artefato; e
- Gerência.

2.5.2 SEGUNDO ESTUDO DE CASO - SISTEMA DE CONTROLE DE SATÉLITES

2.5.2.1 DESCRIÇÃO DO SISTEMA

A descrição deste estudo de caso foi baseada em (INPE, 1989).

O Sistema de Controle de Satélites (SICS) deve ser composto pelo Software Operacional do Satélite do Centro de Controle de Satélites (CCS), denominado

Software de Controle de Satélites (SCS), que fica localizado em São José dos Campos - SP, e pelo Software Operacional de Estação Terrena (CET) que fica localizado, juntamente com outros equipamentos, em Cuiabá - MT e em Alcântara – MA, onde funciona o Centro de Missão e Coleta de Dados, conforme mostra a Figura 2.4. A meta deste sistema é atender as necessidades da Missão Espacial Completa Brasileira (MECB).

Descrição Funcional

O SCS faz parte do software aplicativo do Centro de Controle de Satélites e deve propiciar a realização: das tarefas de tempo real para o controle de satélites; das tarefas de comunicação com as entidades externas ao CCS; e das tarefas de suporte ao processamento *off-line*. Este software também deve efetuar a monitoração e o controle de satélites, e a monitoração e o controle das Estações Terrenas (ET).

O SCS envia à Estação Terrena (ET) informações necessárias para a sua operação e para a análise de desempenho do Sistema de Coleta de Dados. O SCS também recebe dados da ET para a análise dos Subsistemas do Satélite.

O CET deve fazer parte do software aplicativo do computador das Estações Terrenas de Alcântara e Cuiabá, e tem como objetivo dar apoio nas funções de supervisão dos equipamentos das Estações, nas funções de controle de antena e nas funções de *back-up* parcial do Centro de Controle de Satélite, em situações de emergência.

Os principais equipamentos da Estação Terrena relacionados à comunicação com o satélite, excetuando os computadores, são: a **Antena**; o **TCM** (equipamento responsável pelo envio de telecomandos para o satélite); o **TLM** (equipamento responsável pelo recebimento de telemetrias do satélite) e **CMD** (equipamento responsável pelas medidas de distância e calibração).

A Figura 2.4 mostra uma arquitetura simplificada do Sistema de Controle de Satélites (SICS).

Funções principais do SICS:

1) Prover meios para controle de acesso; inicializar ou desativar o subsistema, controlar o acesso dos usuários às funções do sistema por meio de senhas;

2) Prover meios para a transmissão de telecomandos, propiciando o cancelamento de telecomandos a serem transmitidos, a execução da validação e da verificação de comandos por meio de telemetria em tempo real e a gravação dos comandos enviados em um histórico;

3) Prover meios para a visualização dos dados de telemetria em tempo real e a partir de um histórico, assinalando, no caso de visualização de tempo real, os dados de telemetria fora dos limites, identificados por meio do processamento da telemetria, visualizando todos os dados de telemetria fora dos limites de um mesmo quadro, e recuperando, no caso de tempo real, os últimos eventos recebidos de visualização;

4) Gerenciar a antena, permitindo a execução de uma estratégia de aquisição e orastreio do satélite durante a passagem. A estratégia de aquisição selecionada deve ser monitorada e interrompida, caso necessário;

5) Calcular os dados de apontamento da antena para uma dada estratégia, enviando-os para o operador da antena;

6) Enviar o telecomando que liga o transmissor de bordo do satélite;

7) Gerar, em tempo real, os dados para apontamento da antena, necessários no caso de perda de sinal, executando assim a estratégia de Rastreio Assistido;

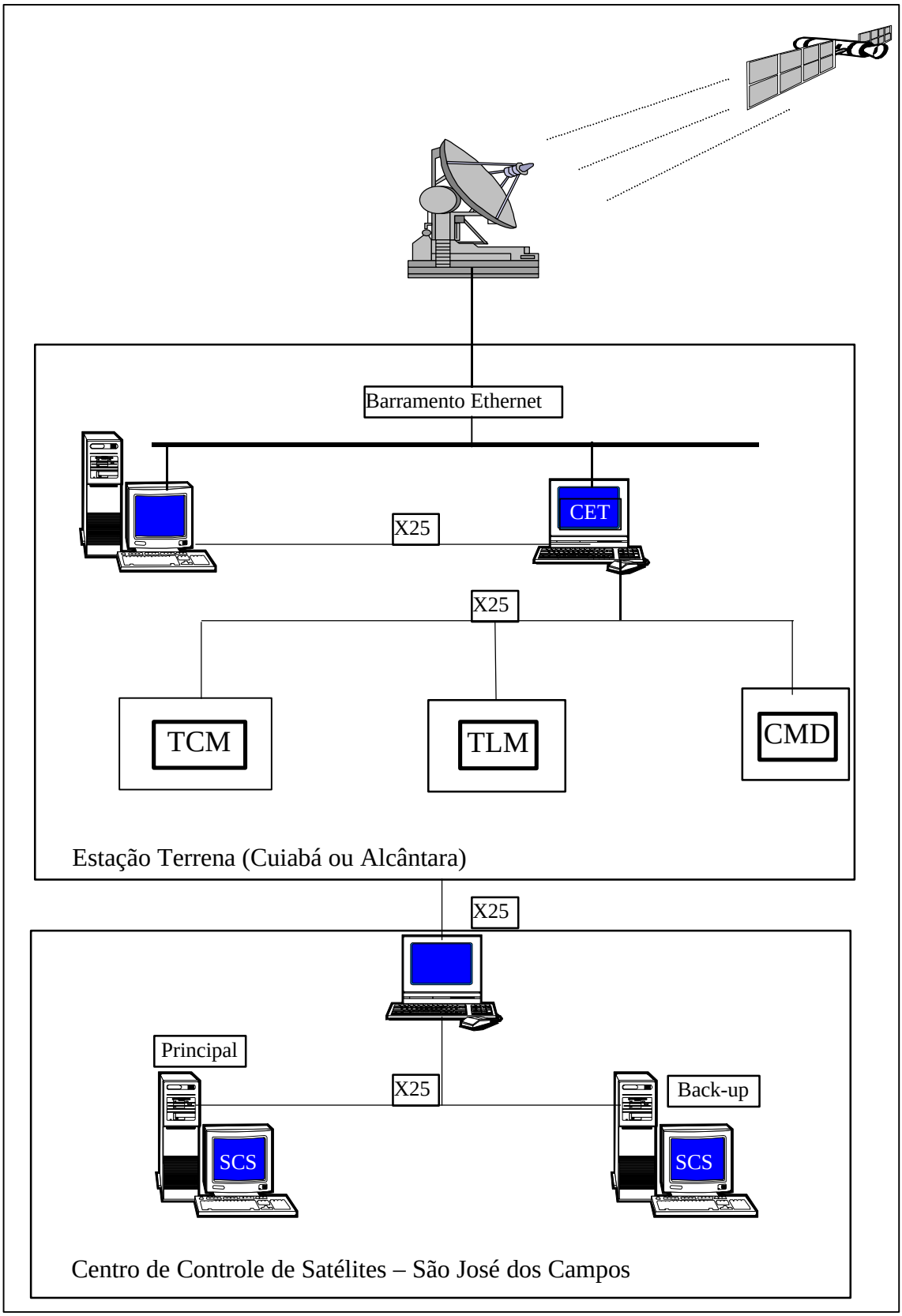

Fig. 2.4 - Arquitetura simplificada do Sistema de Controle de Satélites.

8) Prover os meios para o acompanhamento visual da órbita do satélite em tela mural do CCS;

9) Prover os meios para manutenção, armazenamento e recuperação de dados em arquivos históricos;

10) Prover meios para a visualização, em tempo real, dos dados de supervisão dos equipamentos de uma determinada estação terrena, tanto localmente, como remotamente, no CCS, propiciando a gravação dos dados de supervisão num arquivo histórico de supervisão da estação e reportando os problemas críticos detectados nos equipamentos;

11) Prover os meios para auxiliar na operação de uma Estação Terrena, propiciando a realização das funções auxiliares de operação do Segmento Solo (operações em terra), a comunicação operador/operador, o envio de dados de previsão de passagem do satélite (no CCS), o reconhecimento de alarmes, a visualizaçãoda configuração do Segmento Solo para que o operador tenha conhecimento das conexões ativas, a ativação e desativação dos procedimentos de varredura da antena (na Estação Terrena), o recebimento do estado de operação da antena, a interrupção de um processo de aquisição, o teste de uma determinada estratégia de aquisição. A maioria destas funções deve estar disponível tanto no CCS como nas Estações Terrenas;

12) Prover meios para a solicitação e o armazenamento de medidas de distância do satélite e de medida de calibração do Conjunto de Medidas de Distância (CMD) da Estação Terrena, propiciando o armazenamento das medidas de distância e de calibração que foram consideradas válidas num histórico;

13) Prover meios para inicialização e atualização do relógio dos computadores do CCS e das Estações Terrenas com o horário universal (GMT);

14) Prover meios para a troca de mensagens entre os componentes do SCS e demais hospedeiros da Rede de Comunicação de Dados de Satélites (RECDAS) ou das Estações de Rastreio e Controle Estrangeiras;

15) Prover meios para a transferência de arquivos de dados entre o CCS e as Estações Terrenas e/ou o armazenamento de arquivos de previsão de passagem em disco para o histórico no CCS;

16) Supervisionar o estado dos equipamentos de uma Estação Terrena, propiciando a configuração dos equipamentos da Estação para passagem, calibração e testes; e

17) Prover meios para o armazenamento dos dados de telemetria e o processamento destes dados em tempo real e dos dados de testes, propiciando a provisão de meios para a recuperação de alarmes, para o armazenamento de mensagens de telemetria em tempo real e do computador de bordo válidas, e para a validação dos quadros de telemetria em tempo real e das mensagens de telemetria de testes.

2.5.2.2 IDENTIFICAÇÃO DA DINÂMICA E DO COMPORTAMENTO

O SICS envolve software e hardware situados em duas localidades distintas e separadas fisicamente: o Centro de Controle de Satélites de São José dos Campos e as Estações Terrenas, onde funcionam os Centros de Coleta de Dados da Missão, em Alcântara e Cuiabá, que recebem diretamente do satélite dados de telemetria e enviam telecomandos.

Considerando esta arquitetura distribuída, foram identificados os eventos e processamentos mais importantes, as interações que são disparadas nos sistemas e a identificação das principais entidades (objetos físicos, seres humanos, entidades abstratas) nos dois componentes principais do SICS.

Com as informações fornecidas pela descrição do sistema, foram levantadas porções dinâmicas e comportamentais considerando três níveis diferentes de abstração. Estes níveis são apresentados a seguir em ordem decrescente de granulosidade, referenciados como nível 1, nível 2 e nível 3 respectivamente. O nível 2 foi construído acrescentando-se mais informações às porções do nível 1, não havendo para isto nenhum critério específico. Algumas funções não foram consideradas no nível 1 propositadamente, sendo somente consideradas no nível 2. O nível 3 é resultante de um detalhamento da porção 2 levantada no nível 2 do SCS.

As tabelas que acompanham este levantamento têm como objetivo relacionar as porções dinâmicas e comportamentais do sistema SICS, com os elementos-chave candidatos a representarem o modelo dinâmico, descritos na Seção 2.4.

Para melhor entendimento, estas porções foram levantadas para o Software de Controle de Satélites e para o Software das Estações Terrenas separadamente.

No nível 2, por questões de clareza, existem duas tabelas que representam as porções descritas para o Software de Controle de Satélite (Tabela 2.15 e 2.16) e duas tabelas que representam as porções descritas para o Software da Estação Terrena (Tabelas 2.17 e 2.18).

Nível 1 do Software de Controle de Satélites (CCS-SJC):

1) Envio de telecomandos do Centro de Controle de Satélites para a Estação Terrena;

2) Recebimento, visualização e processamento de telemetria em tempo real pelo Centro de Controle de Satélites enviados pela Estação Terrena;

3) Comunicação operador (SJC) - operador (ET) via RECDAS (Rede de Comunicação de Dados de Satélites) para o envio de previsão de passagem do satélite e informações para a ET;

4) Troca de mensagens entre as Estações de Rastreio e Controle Estrangeiras por solicitação; e

5) Monitoração dos equipamentos da ET.

A Tabela 2.13 mostra a relação entre os elementos-chave e as várias porções levantadas no nível 1 de abstração do Software de Controle de Satélites.

A porção 1 envolve o fluxo de telecomandos entre duas localidades distintas, Centro de Controle e Estação Terrena (entidades componentes), desde que a Estação Terrena esteja apta para receber telecomandos (precondição).

TABELA 2.13 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO SCS - NÍVEL 1

	Porções Levantadas				
Elementos-chave	1	2	3	4	5
EC1-Seqüência	-	x	-	-	-
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x
EC5-Transformação	-	x	-	-	-
EC6-Paralelismo	-	-	-	-	-
EC7-Sincronismo	-	-	x	x	-
EC8-Concorrência	-	-	-	-	-
EC9-Pré e pós-condições	x	x	x	x	-
EC10-Prioridades	-	-	-	-	-

A porção 2 envolve a seqüência recebimento (fluxo de dados), visualização e processamento (transformação) de telemetrias, englobando entidades componentes (Centro de Controle, Estação Terrena e conexões físicas entre as duas localidades), desde que o Centro de Controle esteja apto para receber telemetrias (precondição).

A porção 3 retrata a comunicação síncrona entre os operadores para envio de dados envolvendo fluxo de dados e a arquitetura do sistema. Tratamento análogo é dado à porção 4. Ambas possuem como precondição que as entidades componentes envolvidas estejam conectadas.

A porção 5 retrata o controle periódico dos equipamentos da ET, que é realizado por meio do recebimento de dados dos equipamentos enviados pela ET (fluxo de dados) através das ligações físicas que a conecta com o Centro de Controle (entidades componentes).

Nível 1 do Software Operacional de Estação Terrena (Cuiabá e Alcântara):

1) Posicionamento da antena no instante previsto da passagem do satélite;

2) Gerar em tempo real os dados de apontamento da antena em caso de perda de sinal;

3) Aquisição de telemetria e rastreio do satélite;

4) Recebimento de telecomandos do CCS e envio destes para o satélite durante o rastreio; e

5) Monitoração dos equipamentos da ET.

A Tabela 2.14 ilustra a relação entre os elementos-chave e as porções levantadas para o nível 1 da Estação Terrena.

TABELA 2.14 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO CET- NÍVEL 1

	Porções Levantadas				
Elementos-chave	1	2	3	4	5
EC1-Seqüência	x	-	x	x	-
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x
EC5-Transformação	-	-	-	-	-
EC6-Paralelismo	-	-	-	-	-
EC7-Sincronismo	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-
EC9-Pré e pós-condições	-	x	x	x	-
EC10-Prioridades	-	-	-	-	-

A porção 1 retrata o posicionamento da antena na ET a partir de parâmetros enviados do Centro de Controle. Isto envolve fluxo de dados e as entidades componentes dos sistemas envolvidos.

A porção 2 retrata a geração (transformação) em tempo real dos dados de apontamento da antena em caso de perda de sinal (precondição), envolvendo o fluxo de dados e entidades componentes dos sistemas envolvidos.

A porção 3 envolve a seqüência de aquisição de telemetrias (fluxo de dados) e rastreio do satélite pela ET (precondição: antena rastreando satélite), onde a representação das entidades componentes da estação é importante.

A porção 4 retrata a seqüência de recebimento e envio de telecomandos para o satélite (precondição: antena rastreando o satélite), envolvendo o CCS e os *links* de comunicação com o satélite (entidades componentes) e fluxo de telecomandos.

A porção 5, que trata do monitoramento dos equipamentos da ET, envolve o fluxo periódico de dados de monitoramento da ET para o Centro de Controle.

Nível 2 do Software de Controle de Satélites (CCS-SJC):

1) Inicialização do sistema SICS: envio de mensagens de inicialização para os subsistemas; inicialização do relógio dos computadores (GMT);

2) Transmissão de telecomandos para a Estação Terrena e recebimento de mensagens referentes a telecomandos da Estação Terrena, incluindo o recebimento de alarmes relativos a operação de telecomandos;

3) Recebimento de pedido para o envio de telecomando que liga o transmissor de telemetria do satélite;

4) Conexão com a Estação Terrena para recebimento de telemetria via Rede de Comunicação de Dados de Satélites (RECDAS);

5) Processamento dos quadros de telemetria e visualização destes dados de telemetria em tempo real juntamente com o histórico, para que o operador possa assinalar dados fora dos limites; armazenamento das mensagens de telemetria em

tempo real e do computador de bordo do satélite válidas; enviar eventos de alarme em caso de dados de telemetria fora dos limites;

6) Recebimento de telemetria em tempo real da Estação Terrena para validação e verificação de telecomandos e cancelamento de telecomandos a serem transmitidos; gravação de telecomandos enviados em um histórico;

7) Acompanhamento visual da órbita do satélite em tela mural no CCS;

8) Comunicação operador (SJC)- operador (ET) via RECDAS para o envio de dados de previsão de passagem do satélite e informações para a ET;

9) Receber o status da operação da antena;

10) Interromper um processo de aquisição da antena em andamento;

11) Receber comandos para testes de uma dada estratégia de aquisição;

12) Solicitação de serviço ao CMD (Conjunto de Medidas de Distância) na ET de medidas de calibração do CMD e recebimento das mensagens geradas pelo CMD; enviar alarmes/alertas em casos de medidas de distância e calibração não válidas; armazenamento em um histórico das medidas de distância e calibração válidas;

13) Troca de mensagens entre as Estações de Rastreio e Controle Estrangeiras por solicitação;

14) Solicitar e receber informações sobre o estado dos equipamentos da ET periodicamente (2 em 2 segundos) e armazenar em um histórico; e

15) Desconexão com a Estação Terrena.

A Tabela 2.15 mostra a relação entre os elementos-chave e as oito primeiras porções levantadas no nível 2 do Software de Controle de Satélites.

A porção 1 representa a inicialização de todos os subsistemas e engloba uma seqüência de mensagens enviadas transportando dados e alterando o estado de todos os subsistemas (transformações).

TABELA 2.15 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO SCS/1-NÍVEL 2

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	x	x	-	-	x	x	-	x
EC2-Eventos no tempo	-	-	-	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	x	-	-	-	x	x	-	-
EC6-Paralelismo	-	x	-	-	x	-	-	-
EC7-Sincronismo	-	x	x	x	x	-	-	x
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	-	x	-	-	x	-	-	x
EC10-Prioridades	-	-	-	-	-	-	-	-

A porção 2 envolve uma seqüência de envio de telecomandos para a ET e uma seqüência de recebimento de mensagens da ET, relativas aos telecomandos enviados, e estas seqüências podem ocorrer paralelamente. Sua representação também abrange entidades componentes, fluxo de dados, sincronismo e a precondição de que a Estação Terrena deve estar apta para receber as mensagens.

A porção 3, por tratar-se do envio de uma mensagem transportando dados, envolve além de fluxo de dados, entidades componentes para representar os sistemas envolvidos, suas ligações e sincronismo. O tratamento da porção 4 é análogo ao da porção 3.

A porção 5 retrata seqüências de eventos e transformações que podem ocorrer de maneira paralela: processamento, visualização e armazenamento dos dados de telemetria e envio de alarmes à ET caso necessário (precondição: telemetrias fora dos

limites). Estas seqüências envolvem fluxo de dados, transformações, a representação de entidades componentes e sincronismo.

A porção 6 pode ser vista com uma seqüência de dados enviados para o Centro de Controle, a realização de verificação, validação e armazenamento dos dados (transformações) englobando fluxo de dados e entidades componentes dos subsistemas envolvidos.

A porção 7 envolve somente entidades componentes e fluxo de dados para sua representação. A porção 8 abrange a comunicação síncrona (precondição: operadores sincronizados) entre os operadores do Centro de Controle e da ET, por meio de uma seqüência de envio de dados (fluxo de dados), envolvendo as conexões físicas entre eles (entidades componentes).

A Tabela 2.16 mostra a relação entre os elementos-chave e as sete últimas porções levantadas para o nível 2 do Software de Controle de Satélites.

TABELA 2.16 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO SCS/2 - NÍVEL 2

	Porções Levantadas						
Elementos-chave	9	10	11	12	13	14	15
EC1-Seqüência	-	-	-	x	x	x	-
EC2-Eventos no tempo	-	-	-	-	-	x	-
EC3-Entidades componentes	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x
EC5-Transformação	-	-	-	x	-	-	-
EC6-Paralelismo	-	-	-	-	-	-	-
EC7-Sincronismo	-	-	-	-	x	x	-
EC8-Concorrência	-	-	-	-	-	-	-
EC9-Pré e pós-condições	-	-	-	x	x	-	-
EC10-Prioridades	-	-	-	-	-	-	-

A porção 9 envolve o fluxo de informações a respeito da antena vindos da ET para o Centro de Controle (entidades componentes). As porções 10 , 11 e 15 têm tratamento análogo à 9.

A porção 12 retrata a seqüência recebimento de dados da ET, processamento destes dados (transformações) e possível retorno de mensagens à ET transportando dados (pós-condição: enviar alarmes/alertas de medidas de distância e calibração não válidas). Por tratar-se de comunicação entre duas localidades distintas, a representação de entidades componentes é importante.

A porção 13 envolve seqüências de troca de mensagens (fluxo de dados) entre o Centro de Controle e alguma Estação de Rastreio e Controle Estrangeira (precondição: sincronia entre o Centro de Controle e a Estação Estrangeira). Nesta porção a representação de entidades componentes é importante por tratar-se de uma comunicação com uma entidade externa ao sistema, e esta comunicação ocorrer de maneira síncrona.

A porção 14 retrata a seqüência de solicitação e recebimento de dados da ET de 2 em 2 segundos (momento em que os fluxos de dados ocorrem) de maneira síncrona, e o posterior armazenamento destes dados num histórico (transformação).

Nível 2 do Software Operacional de Estação Terrena (Cuiabá eAlcântara):

1) Estabelecer comunicação operador (ET)- operador(SJC);

2) Recebimento pelo gerenciador da antena de comandos de supervisão para configurar o controlador da antena; envio pelo controlador da antena de comandos para a execução de uma estratégia de rastreio selecionada;

3) Envio pelo gerenciador da antena ao controlador da antena dos dados de apontamento para o posicionamento da antena no instante previsto da passagem do satélite, recebidos do Centro de Controle de Satélites;

4) Interrupção da execução e testes de uma determinada estratégia;

5) Recebimento pelo controlador da antena de comandos de atuação/interrupção do procedimento de varredura do up-link;

6) Ativação e interrupção pelo gerenciador da antena da varredura de freqüência up-link da antena;

7) Envio pelo gerenciador da antena ao operador do status da estratégia de aquisição selecionada e da varredura de freqüência periodicamente. Estes dados são armazenados num histórico;

8) Pedido de envio de telecomando para ligar o transmissor de telemetria do satélite;

9) Aquisição de telemetria e possível interrupção pelo operador de um processo de aquisição em andamento;

10) Gerar em tempo real os dados para apontamento da antena necessários em caso de perda de sinal; e

11) Receber dados de monitoração dos equipamentos da ET coletados, visualização pelo operador em tempo real destes dados e envio de mensagens referentes aos equipamentos para um histórico no CCS; divulgar mensagens de problemas críticos encontrados nos equipamentos da ET, caso necessário.

A Tabela 2.17 mostra a relação entre os elementos-chave e as seis primeiras porções levantadas para o nível 2 da Estação Terrena.

A porção 1, estabelecimento de conexão entre os operadores, envolve uma seqüência de eventos com fluxo de dados, onde a representação dos aspectos da conexão física é importante.

A porção 2 envolve uma seqüência de fluxo de dados e procedimentos de configuração (transformações) que culminam também com a necessidade da representação de entidades componentes. A porção 3 tem tratamento análogo ao da 2, onde o novo posicionamento da antena representa a transformação realizada.

TABELA 2.17 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO CET/1 - NÍVEL 2

	Porções Levantadas					
Elementos-chave	1	2	3	4	5	6
EC1-Seqüência	x	x	x	-	-	-
EC2-Eventos no tempo	-	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x
EC5-Transformação	-	x	x	-	-	-
EC6-Paralelismo	-	-	-	-	-	-
EC7-Sincronismo	-	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-	-
EC9-Pré e pós-condições	-	-	-	-	-	-
EC10-Prioridades	-	-	-	-	-	-

As porções 4, 5 e 6 podem ser vistas como o envio de mensagens contendo dados para o controlador da antena, onde a representação do gerenciador e do controlador da antena são importantes (entidades componentes).

A Tabela 2.18 ilustra a relação entre os elementos-chave e as cinco últimas porções levantadas para o nível 2 da Estação Terrena.

A porção 7 envolve a seqüência envio de dados para o operador da ET e o armazenamento destes, que pode ser representado por fluxo de dados e transformações (processo de armazenagem).

A porção 8 envolve o envio de mensagem com dados para o Centro de Controle, ficando caracterizado o fluxo de dados e a representação de entidades componentes.

TABELA 2.18 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO CET/2 - NÍVEL 2

Elementos-chave	7	8	9	10	11
EC1-Seqüência	x	-	x	x	x
EC2-Eventos no tempo	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x
EC5-Transformação	x	-	-	x	x
EC6-Paralelismo	-	-	-	-	-
EC7-Sincronismo	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-
EC9-Pré e pós-condições	-	-	x	x	x
EC10-Prioridades	-	-	-	-	-

Na porção 9 está retratada a seqüência do processo de aquisição de telemetrias pela ET (precondição: antena rastreando satélite), que engloba fluxo de dados e os subsistemas envolvidos na aquisição.

A porção 10 envolve a seqüência de geração de dados em tempo real (transformações) e fluxo destes dados para o controlador da antena (precondição: perda do sinal do satélite). Os subsistemas envolvidos devem ser representados.

Na porção 11 observa-se uma seqüência de eventos que envolvem fluxo de dados e transformações (pós-condição: divulgar problemas críticos), onde as conexões físicas e lógicas entre o Centro de Controle e a ET devem ser representadas.

Nível 3 do Software de Controle de Satélites (CCS-SJC) (detalhamento da porção 2 do Nível 2 do SCS – Transmissão de telecomandos e recebimento de mensagens):

1) Inserir um telecomando básico ou uma seqüência de telecomandos na pilha de telecomandos a serem enviados para a ET e retirar um comando da pilha;

2) Estabelecer conexão com o equipamento de telecomando (CTC) da ET;

3) Montar, enviar, receber e verificar as mensagens de comando entre o Software de Telecomandos e o equipamento de telecomando (CTC);

4) Autorizar o envio de telecomandos temporizados da pilha;

5) Abortar um telecomando armazenado no Conjunto de Telecomandos da Estação Terrena;

6) Executar a validação e verificação dos telecomandos por meio de telemetria em tempo real;

7) Gravação dos telecomandos enviados num Histórico de Comandos; e

8) Acusar o recebimento de alarme e visualizar os últimos alarmes associados a operação do Software de Telecomandos.

A Tabela 2.19 ilustra a relação entre os elementos-chave e as porções levantadas no detalhamento da porção 2 do nível 2 do Software de Controle de Satélites.

A porção 1 envolve um procedimento de inserção de dados em uma estrutura (transformação e fluxo de dados).

A porção 2 retrata uma seqüência de envio de dados do Centro de Controle para a ET a fim de ser estabelecida uma conexão síncrona. Novamente aqui a representação das conexões físicas e interfaces existentes é importante.

A porção 3 retrata seqüências de envio e recebimento simultâneo de dados e processamentos realizados (transformações) entre o Centro de Controle a ET (precondição: Centro de Controle e Estação Terrena estarem conectados). Por envolver a comunicação entre duas localidades distintas a representação de entidades componentes é importante.

TABELA 2.19 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO SCS - NÍVEL 3 (PORÇÃO 2 DO NÍVEL 2)

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	-	x	x	-	-	x	-	x
EC2-Eventos no tempo	-	-	-	-	-	-	-	
EC3-Entidades componentes	-	x	x	x	x	x	-	
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	x	-	x	x	x	x	x	x
EC6-Paralelismo	-	-	x	-	-	-	-	x
EC7-Sincronismo	-	x	-	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	-	-	x	x	-	x	-	-
EC10-Prioridades	-	-	-	-	-	-	-	-

O tratamento das porções 4 e 5 é o mesmo. Ambas as porções envolvem comunicação entre localidades distintas, fluxo de dados e transformações resultantes nos subsistemas envolvidos.

A porção 4 envolve a conexão do Centro de Controle e a Estação Terrena (precondição).

A porção 6 envolve a seqüência de recebimento de telemetria em tempo real (fluxo de dados) e procedimentos de verificação e validação dos telecomandos baseados nestas telemetrias (transformações), englobando também a representação de entidades componentes. Deve existir a conexão entre o Centro de Controle e a Estação Terrena (precondição).

A porção 7 envolve somente o processamento e o fluxo de informações. Já em 8 existe uma seqüência de eventos (recebimento e visualização) simultâneos que também deve ser considerada.

2.5.2.3 ENTIDADES CANDIDADAS A SEREM MODELADAS

A seguir são listadas as entidades componentes do Sistema de Controle de Satélites, identificadas durante o levantamento das porções, candidatas a fazerem parte da representação da dinâmica do sistema.

- Equipamentos da Estação Terrena: TCM, TLM, CMD e Antena;
- Computador(es) da Estação Terrena; Computador(es) do CCS;
- Controlador da antena;
- Gerenciador da antena (ET);
- Operadores (CCS e ET);
- Histórico;
- Processamento de telemetrias e telemetrias;
- Processamento de telecomandos e telecomandos;
- Alarmes;
- Visualizador de telecomandos(CCS e ET);
- Visualizador de telemetrias (CCS e ET);
- Visualizador da órbita do satélite (CCS);
- Supervisor dos equipamentos da ET;
- Visualizador dos dados de supervisão dos equipamentos da ET (CCS e ET);
- Processador de telecomandos (CCS) e Processador de telemetrias (CCS); e
- Módulo de medidas de distância (CCS).

2.5.3 TERCEIRO ESTUDO DE CASO: SISTEMA DE GERÊNCIA DE TRÁFEGO DE TRENS

2.5.3.1 DESCRIÇÃO DO SISTEMA

A descrição deste sistema é apresentada em (Booch, 1994). O Sistema de Gerência de Tráfego de Trens tem duas funções primárias básicas: o escalonamento e a rota dos trens, e a monitoração dos sistemas dos trens. Funções relacionadas incluem: planejar o tráfego; acompanhar a localização do trem; monitorar o tráfego; evitar colisões; prevenir falhas; e registrar dados para a manutenção.

A Figura 2.5 mostra um esquema simplificado da arquitetura do sistema em termos dos principais subsistemas e sua disposição.

O Sistema de Análise e Relatórios do trem, conforme mostra a Figura 2.5, inclui vários sensores analógicos e discretos para monitorar elementos tais como temperatura do óleo, quantidade de combustível, aceleração, RPM (Rotações Por Minuto) do motor, temperatura da água e força de arranque. Os valores dos sensores são mostrados ao engenheiro do trem através de um monitor a bordo, e aos despachantes e pessoal da manutenção, em qualquer lugar da rede. Condições de alarme e advertência são registradas se certos valores dos sensores ficam fora do intervalo normal de operação. Um registro (*log*) dos valores dos sensores é mantido para suportar a manutenção e a gerência do combustível.

O Sistema de Gerência da Energia, Figura 2.5, avisa o engenheiro do trem, em tempo real, os valores mais eficientes da aceleração e do freio. Entradas para este sistema incluem: o perfil do trilho e a inclinação, os limites de velocidade, os cronogramas, a carga do trem e a potência disponível. Com estes dados o sistema pode determinar a aceleração eficiente e uso dos freios, consistentes com o cronograma desejado e com a segurança. Os valores de aceleração e as intensidades de frenagem sugeridas, o perfil do trilho e a inclinação, a posição do trem e a velocidade, estarão disponíveis no Sistema de Visualização, a bordo do trem.

O Sistema de Visualização, Figura 2.5, fornece a interface homem-máquina para o engenheiro do trem. Informações do Sistema de Análise e Relatórios, do Sistema de

Gerência da Energia e da Unidade de Gerência de Dados encontram-se disponíveis para visualização. Existem maneiras de permitir que o engenheiro selecione diferentes visualizações.

A Unidade de Gerência de Dados, Figura 2.5, serve como um tradutor (recebimento e envio) das comunicações entre os sistemas a bordo e o resto da rede, a qual todos os trens, despachantes e outros usuários encontram-se conectados.

O acompanhamento da localização do trem é realizado por meio de dois dispositivos: um *transponder* de localização e um Sistema de Posicionamento Global (Global Position System - GPS) conforme mostra a Figura 2.5. O Sistema de Análise e Relatórios do trem pode determinar a localização geral do trem, via contagem das revoluções da roda. Esta informação é melhorada pelos *transponders* de localização, colocados a cada quilômetro do trilho e nas junções críticas. Estes *transponders* fornecem a posição para o trem que estiver passando, recebida pela sua Unidade de Gerência de Dados, sendo assim possível determinar a sua localização mais exata. Os trens podem estar equipados com receptores GPS, fornecendo a sua localização com 1 metro de precisão.

Uma Unidade de Interface *Wayside,* Figura 2.5, é colocada em cada dispositivo controlável (tal como uma chave) ou um sensor (um sensor infravermelho para detectar superaquecimento das rodas). Cada Unidade de Interface *Wayside* que existir no trilho pode receber comandos de um Controlador Terminal-Terra local, por exemplo, ligar e desligar um sinal. Dispositivos podem ser substituídos por controles manuais locais. Um Controlador Terminal-Terra serve para enviar e receber informação dos trens que passam e das Unidades de Interface. Controladores Terminal-Terra são colocados ao longo dos trilhos, espaçados de tal forma que todo trem encontre-se sempre em um intervalo de no mínimo um terminal.

Todo Controlador Terminal-Terra repassa sua informação para um Sistema de Controle da Rede. Conexões entre o Sistema de Controle da Rede e cada Controlador Terminal-Terra podem ser realizadas via enlace de microondas, linhas por terra, ou fibras óticas, dependendo do quão remoto cada Controlador encontra-se do Sistema de Controle da

Rede. Este controle monitora a saúde de toda a rede e pode automaticamente rotear a informação para caminhos alternativos, em caso de falhas do equipamento.

O Sistema de Controle da Rede, Figura 2.5, é finalmente conectado a um ou mais Centros de Despacho. Cada Centro de Despacho compreende o Sistema de Controle de Operações da ferrovia (trilhos). No Sistema de Controle de Operações, despachantes podem estabelecer rotas do trem e trilhar o progresso destes trens individuais. Despachantes individuais controlam diferentes territórios.

Cada console de controle do despachante pode ser iniciado para controlar um ou mais territórios. Rotas dos trens incluem instruções para automaticamente trocar os trens de trilho para trilho, com restrições de velocidade, deixar ou pegar vagões e permitir ou negar a entrada do trem numa seção do trilho específica. Despachantes podem anotar a localização dos trilhos em manutenção ao longo das rotas dos trens.

Trens podem ser parados pelo Sistema de Controle de Operações da ferrovia (manualmente pelos despachantes ou automaticamente) quando condições adversas são detectadas (tais como trem sem controle, falha no trilho ou condição de colisão potencial). Despachantes também podem ter acesso a qualquer informação disponível para os engenheiros de trens individuais, como também enviar autorização de movimento, configurar dispositivos *Wayside* e planejar revisões.

A disposição dos trilhos e os equipamentos de interface podem mudar com o tempo. O número de trens e suas rotas podem mudar diariamente. O sistema deve ser projetado para permitir a incorporação de novos sensores, redes e tecnologia de processamento.

2.5.3.2 CONSIDERAÇÕES PARA MODELAGEM

Duas observações podem ser feitas a partir desta descrição de alto nível dos requisitos deste sistema: deve ser permitida a evolução da arquitetura com o tempo e a implementação deve basear-se em padrões existentes para uma maior extensão prática.

Existe uma rede de comunicação de dados distribuída ligando este sistema. Mensagens são passadas por meio de sinais de rádio dos *transponders* para os trens, entre os trens e os Controladores Terminal-Terra e entre os trens e as Unidades de Interface *Wayside*.

Cada Controlador Terminal-Terra liga-se a uma rede local, uma para cada Centro de Despacho. Mensagens também circulam entre eventos do Centro de Despacho e os Controladores Terminal-Terra individuais. A operação segura do sistema inteiro depende de transmissão e recepção a tempo e confiável das mensagens.

O sistema de Controle de Operações do Centro de Despacho deve acompanhar as localizações atuais e rotas planejadas de muitos trens simultaneamente. Esta informação deve ser mantida atualizada e consistente, mesmo na presença de atualizações concorrentes e consultas de toda parte da rede. Isto é basicamente um problema de banco de dados distribuídos.

As interfaces homem-máquina devem considerar os diferentes tipos de usuários do sistema. Os usuários são principalmente engenheiros do trem e despachantes. Todas as formas de interação com o usuário devem ser cuidadosamente projetadas para adequar-se a este grupo específico.

O Sistema de Gerência de Tráfego de Trens, quando operacional, deve envolver dezenas de computadores, incluindo um para cada trem, um para cada Unidade de Interface *Wayside* e dois para cada Centro de Despacho (um primário e um *back-up* para caso de falhas no principal).

O Sistema de Gerência de Tráfego deve interagir com uma variedade de sensores e atuadores. Não importa o dispositivo, o problema de adquirir dados dos sensores e controlar o ambiente deve ser tratado de uma maneira consistente pelo sistema. Os atuadores, por exemplo, têm funções *fail safe* que não foram explicitadas na especificação.

Os tipos de valores retornados dos vários sensores são todos diferentes, mas o processamento dos diferentes dados é o mesmo. É altamente recomendável fazer uma análise de domínio para todos os sensores periódicos e não discretos, de tal forma que se possa inventar um mecanismo de sensor comum para todos os tipos.

Todo trem possui uma localização no trilho e um plano de movimentação ativo, que relata o percurso que o trem deve realizar.

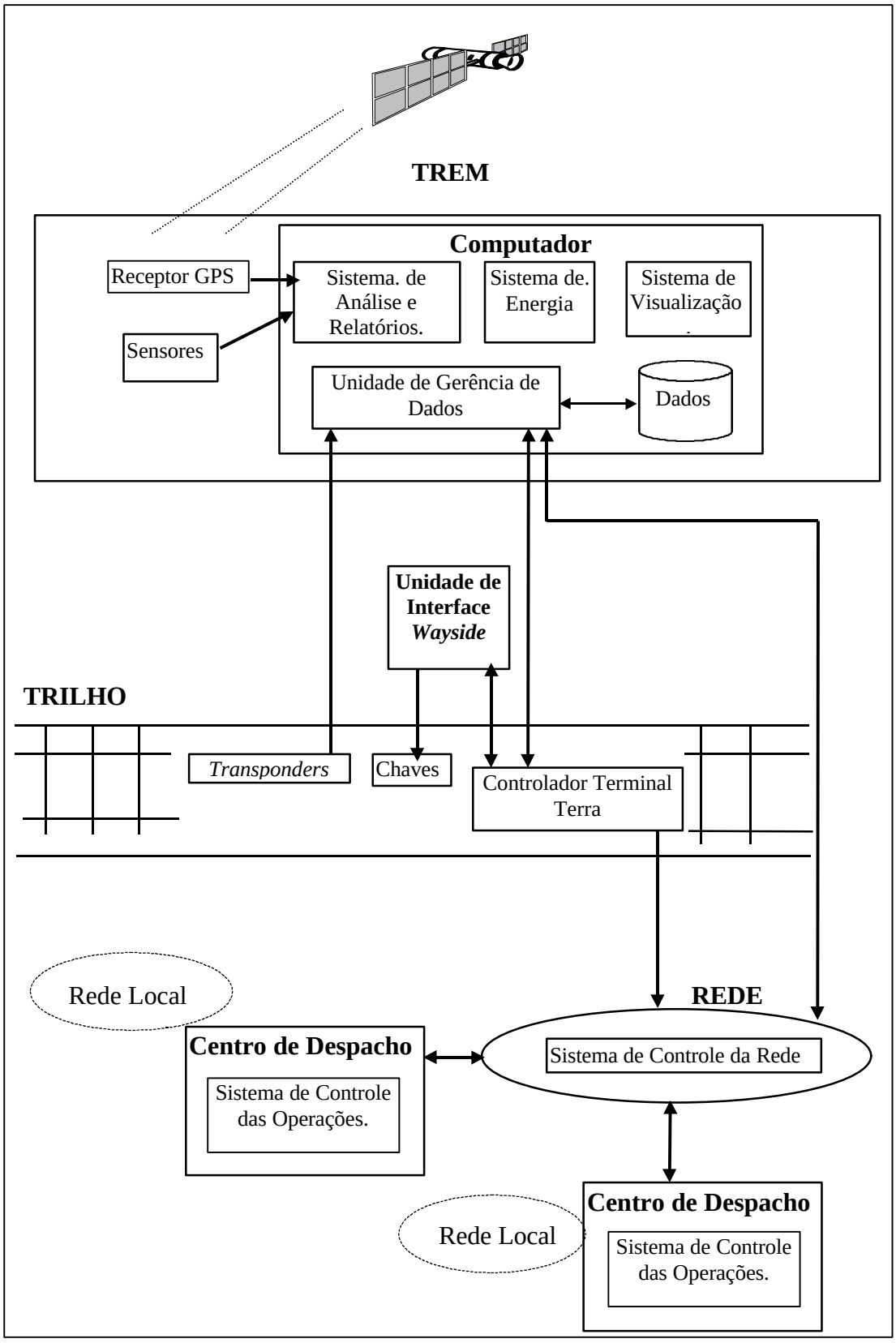

Fig. 2.5 - Arquitetura do sistema de Controle de Tráfego de Trens.

2.5.3.3 IDENTIFICAÇÃO DA DINÂMICA E DO COMPORTAMENTO

Nesta Seção é feita a identificação dos principais aspectos dinâmicos deste sistema para tentar compreender melhor o seu funcionamento, envolvendo os cenários principais e os eventos mais importantes, o momento em que eles ocorrem, as interações que são disparadas no sistema e as principais entidades (objetos físicos, seres humanos e entidades abstratas). A identificação de alguns cenários de uso do sistema e a definição das interfaces são um bom início para um sistema desta magnitude.

É assumido que os arquitetos do sistema já tenham escolhido a sua arquitetura de hardware inicial, conforme mostra a Figura 2.5. Esta escolha não é necessariamente irreversível, mas ela fornece um ponto de partida para localizar os requisitos de software.

Assim como nos casos estudados anteriormente, com as informações fornecidas pela descrição do sistema, foram levantadas porções dinâmicas e comportamentais considerando três níveis diferentes de abstração dos detalhes. Estes níveis são apresentados a seguir em ordem decrescente de granulosidade, referenciados como nível 1, nível 2 e nível 3 respectivamente. O nível 2 foi construído acrescentando-se mais informações às porções do nível 1, não havendo para isto nenhum critério específico. Algumas funções foram propositadamente consideradas somente no nível 2.

No nível 1 considera-se as porções que envolvem eventos tanto no interior como no exterior dos trens, dando maior importância à recepção e envio de mensagens em detrimento ao processamento propriamente dito. Já no nível 2, para maior clareza, foram consideradas as porções no interior do trem e no exterior separadamente e alguns dos processamentos realizados.

O nível 3 é um detalhamento das porções 1 e 2 do sistema do interior dos trens identificadas no nível 2.

Para todos os níveis considerados, foi assumido que as comunicações que envolvem a rede de dados encontram-se funcionando adequadamente.

As tabelas que acompanham este levantamento têm como objetivo relacionar as porções dinâmicas e comportamentais do Sistema de Gerência de Tráfego com os elementos-chave candidatos a representar o modelo dinâmico, descritos na Seção 2.4.

Nível 1:

1) Recebimento de dados pelo trem sobre seu posicionamento através do receptor GPS, para que sua posição correta possa ser determinada com no máximo um metro de erro;

2) Recebimento de dados dos sensores do trem; utilização destes dados para calcular o melhor desempenho do trem, considerando consumo de combustível, aceleração e inclinação do trilho;

3) Recebimento de dados dos transponders pelo trem para ajudar na sua localização;

4) Envio de comandos do trem para o Controlador Terminal-Terra e deste para as Interfaces Wayside;

5) Envio de comandos do Centro de Despacho para o Terminal-Terra e deste para as Interfaces Wayside;

6) Envio de mensagens das Interfaces Wayside para o Controlador Terminal-Terra informando o estado dos equipamentos dos trilhos;

7) Troca de dados entre os Centros de Despacho e os trens por meio da Unidade de Gerência de Dados do trem; e

8) Comunicação entre os Centros de Despacho para a troca de informações sobre os trens e condições dos trilhos.

A Tabela 2.20 ilustra a relação entre os elementos-chave e as porções levantadas no nível 1 para o Sistema de Gerência de Tráfego de Trens.

A porção 1 envolve a seqüência de recebimento de dados do GPS (precondição: receptor GPS funcionando), englobando fluxo de dados e entidades componentes. A

porção 2 envolve as seqüência de recebimento simultâneo de dados dos vários sensores (paralelismo e fluxo de dados) e o processamento destes dados (transformações).

A porção 3 envolve os fluxos de dados vindos dos *transponders* (precondição: *transponder* funcionando) e entidades componentes da conexão *transponder*-trem.

A porção 4 retrata uma seqüência de envio de dados (fluxo de dados) entre localidades distintas, com transformações ocorrendo nos dispositivos de hardware (chaves e sensores).

A porção 5 tem tratamento análogo ao da porção 4.

TABELA 2.20 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA DO SISTEMA DE GERÊNCIA DE TRÁFEGO - NÍVEL 1

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	-	x	-	x	x	-	x	x
EC2-Eventos no tempo	-	-	-	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	-	x	-	-	-	-	-	-
EC6-Paralelismo	-	x	-	-	-	-	-	x
EC7-Sincronismo	-	-	-	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	x	x	x	-	-	-	-	-
EC10-Prioridades	-	-	-	-	-	-	-	-

A porção 6 trata do envio de mensagens com dados para um sistema fisicamente separado, envolvendo além de entidades componentes, o fluxo de dados.

A porção 7 envolve a seqüência de troca de dados entre os Centros de Despacho e os trens, onde fluxo de dados e as entidades componentes devem ser considerados.

A porção 8 é análoga à porção 7, podendo haver comunicação paralela entre os vários Centros de Despacho.

Nível 2 do Sistema de Gerência de Tráfego de Trens- Interior dos trens:

1) Recebimento dos dados dos sensores que estão a bordo do trem pelo Sistema de Análise e Relatórios do trem, análise dos dados, apresentação destes dados ao engenheiro do trem e registro dos valores dos sensores num registro log;

2) Envio de alarmes pelo Sistema de Análise e Relatórios para o engenheiro do trem em caso de valores coletados dos sensores fora dos limites estabelecidos;

3) Envio de dados gerais do trem para os despachantes e pessoal da manutenção através da rede;

4) Recebimento pelo Sistema de Gerência de Energia de dados do perfil do trilho, limites de velocidades e cronogramas, carga do trem e potência disponíveis; determinação da aceleração eficiente e uso de freios, tornando disponível estes valores no Sistema de Visualização a bordo;

5) Cálculo pelo Sistema de Energia dos valores mais eficientes da aceleração e do freio e avisa o engenheiro do trem em tempo real, utilizando o Sistema de Visualização;

6) Recebimento pelo Sistema de Visualização dos dados enviados pelo Sistema de Análise e Relatórios (dados dos sensores analógicos e discretos, condições de alarme e advertência) e Unidade de Gerência de Dados;

7) Recebimento e envio pela Unidade de Gerência de Dados das informações trocadas entre o sistema de bordo e o resto da rede; e

8) Recebimento e análise dos dados pelo Sistema de Análise e Relatórios vindos dos transponders e GPS; envio destas informações para um registro log e para o engenheiro do trem.

A Tabela 2.21 ilustra a relação entre os elementos-chave e as porções levantadas no nível 2 considerando somente os sistemas no interior do trem.

A porção 1 retrata a seqüência de recebimento de dados dos sensores a bordo do trem (precondição: sensores funcionando), processamento (transformações), interpretação e armazenamento destes dados, envolvendo além de entidades componentes do sistema, o fluxo de dados, as transformações, paralelismo no recebimento dos dados dos sensores e sincronismo.

O envio de alarmes para o engenheiro (precondição: limites dos valores dos sensores extrapolados), retratado na porção 2, envolve o Sistema de Análise e Relatórios (entidade componente) como também o fluxo de dados.

A porção 3 trata do envio simultâneo de dados do trem para os despachantes e pessoal da manutenção. A porção 4 retrata a seqüência de recebimento de dados, processamento destes dados e posterior envio dos dados processados para o Sistema de Visualização a bordo do trem.

Na porção 5 o Sistema de Visualização deve estar funcionando (precondição) para o recebimento dos dados (fluxo de dados e entidades componente).

A porção 6 retrata a chegada paralela de dados para o engenheiro do trem vindo de sistemas diferentes.

A porção 7 retrata a seqüência de recebimento e o envio de dados do trem para o resto da rede, envolvendo a representação dos sistemas envolvidos e do fluxo de dados.

A porção 8 envolve a seqüência de recebimento dos dados dos equipamentos *transponder* e GPS (entidades componentes, fluxo de dados e sincronismo), análise e registro destes dados pelo Sistema de Análise e Relatórios (transformação).

TABELA 2.21 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA NO INTERIOR DOS TRENS - NÍVEL 2

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	x	-	-	x	x	-	x	x
EC2-Eventos no tempo	-	-	-	-	-	-	-	
EC3-Entidades componentes	x	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	x	-	-	x	x	-	-	x
EC6-Paralelismo	x	-	x	-	-	x	-	-
EC7-Sincronismo	x	-	-	-	-	-	-	x
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	x	x	-	-	x	-	-	-
EC10-Prioridades	-	-	-	-	-	-	-	-

Nível 2 do Sistema de Gerência de Tráfego de Trens - Exterior dos trens:

1) Recebimento pela Unidade de Interface Wayside de comandos vindos de um Controlador Terminal-Terra local e a conseqüente atuação da chave conectada;

2) Envio e recebimento de informações pelo Controlador Terminal-Terra dos trens que estejam passando e dos Centros de Despacho;

3) Envio e recebimento de informações pelo Controlador Terminal-Terra vindas das Unidades de Interface Wayside;

4) Envio de informações pelo Controlador Terminal-Terra para o Sistema de Controle da Rede;

5) Envio de instruções pelo Centro de Despacho para os trens (cada Centro controla um território). Instruções incluem: mudança de trilho, parada do trem, trilhos em funcionamento.

6) Acesso pelos despachantes à informação disponível no interior dos trens;

7) Envio pelo despachante de autorização de movimento do trem; e

8) Configuração das Interfaces Wayside pelos despachantes.

A Tabela 2.22 ilustra a relação entre os elementos-chave e as porções levantadas no nível 2 para os sistemas no exterior do trem.

TABELA 2.22 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA NO EXTERIOR DOS TRENS - NÍVEL 2

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	x	x	x	-	-	-	-	-
EC2-Eventos no tempo	-	-	-	-	-	-	-	
EC3-Entidades componentes	x	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	x	-	-	-	-	-	-	x
EC6-Paralelismo	-	x	-	-	x	-	-	-
EC7-Sincronismo	x	-	-	-	-	-	-	-
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	x	-	-	-	-	-	-	x
EC10-Prioridades	-	-	-	-	-	-	-	-

A porção 1 envolve a seqüência de recebimento de um comando (fluxo de dados) e posterior atuação (transformação) da interface *Wayside* sobre um dispositivo (precondição: comunicação Interface-Dispositivo funcionando).

A porção 2 retrata o recebimento de dados e o envio de dados (fluxo de dados) dos trilhos para os trens passantes e para os Centros de Despacho (entidade componente). As porções 3, 4, 5, 6 e 7 têm tratamento análogo ao da porção 2, com paralelismo ocorrendo na porção 5 e seqüência de envio e recebimento de dados na porção 3.

A porção 8 envolve o procedimento de configuração (transformação) de uma interface *Wayside* (precondição: comunicação Centro de Controle-Interface estar funcionando), no qual devem ser representados tanto o fluxo de dados como as conexões entre os Centros de Despacho e os dispositivos de interface (representação das entidades componentes).

Nível 3 do Sistema de Gerência de Tráfego de Trens (detalhamento das porções 1 e 2 do interior do trens/nível 2 - Recebimento dos dados dos sensores e envio de alarmes):

1) Os dados analógicos do sensor de temperatura do óleo periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

2) Os dados analógicos do sensor de pressão do óleo periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

3) Os dados analógicos do sensor de quantidade de combustível periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

4) Os dados analógicos do sensor de RPM do motor periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

5) Os dados analógicos do sensor de temperatura da água periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

6) Os dados analógicos do sensor de aceleração periodicamente são lidos e enviados para o Sistema de Análise e Relatórios do trem;

7) O Sistema de Análise e Relatórios converte os dados analógicos recebidos para digitais, registra os valores dos sensores no arquivo de log do trem, os envia para o sistema de visualização para que o engenheiro do trem possa visualizá-los e também os envia através da rede para os despachantes e pessoal da manutenção; e

8) Os dados recebidos dos sensores já convertidos são analisados e caso algum deles esteja fora dos limites de operação pré-estabelecidos, condições de advertência ou de alarme são enviadas para o engenheiro do trem na tela de visualização.

A Tabela 2.23 ilustra a relação entre os elementos-chave e as porções levantadas no detalhamento das porções 1 e 2 dos sistemas no interior do trem.

TABELA 2.23 - ELEMENTOS-CHAVE DO MODELO X DINÂMICA NO INTERIOR DOS TRENS - NÍVEL 3 (PORÇÕES 1 E 2 DO INTERIOR DOS TRENS - NÍVEL 2)

	Porções Levantadas							
Elementos-chave	1	2	3	4	5	6	7	8
EC1-Seqüência	-	-	-	-	-	-	x	x
EC2-Eventos no tempo	-	-	-	-	-	-	-	-
EC3-Entidades componentes	x	x	x	x	x	x	x	x
EC4-Fluxos de entidades	x	x	x	x	x	x	x	x
EC5-Transformação	-	-	-	-	-	-	x	x
EC6-Paralelismo	-	-	-	-	-	-	x	-
EC7-Sincronismo	x	x	x	x	x	x	-	-
EC8-Concorrência	-	-	-	-	-	-	-	-
EC9-Pré e pós-condições	x	x	x	x	x	x	-	x
EC10-Prioridades	-	-	-	-	-	-	-	-

A porção 8 retrata o processamento local feito pelo Sistema de Análise e Relatórios do trem (transformação) e o possível envio de alarmes (fluxo de dados e precondição: limites dos valores dos sensores extrapolados) para o Sistema de Visualização (entidades componentes) para que o engenheiro possa tomar providências.

As porções de 1 a 6 tratam da leitura e do envio síncrono de dados (fluxo de dados) dos vários sensores instalados no trem (precondição: sensores funcionando) para o Sistema de Análise e Relatórios, envolvendo a representação de entidades componentes.

A porção 7 envolve o recebimento de dados (fluxo de dados) pelo Sistema de Análise e Relatórios, que pode ser simultâneo ou não, o processamento destes dados (conversão analógico-digital) e o seu armazenamento local; em seguida, eles são enviados para visualização local e através da rede (entidades componentes).

2.5.3.4 ENTIDADES CANDIDATAS A SEREM MODELADAS

A seguir são listadas as entidades componentes do Sistema de Gerência de Tráfego de Trens, identificadas durante o levantamento das porções, candidatas a fazerem parte da representação da dinâmica do sistema.

- Trens;
- Trilhos;
- Engenheiro do trem;
- Despachante;
- Sensores;
- Transponders;
- Computador de bordo do trem;
- Sistema de Análise e Relatórios;
- Sistema de Energia;
- Sistema de Visualização;
- Unidade de Gerência de Dados;
- Dados Analógicos e Digitais;
- Unidade de Interface Wayside;
- Chaves (Switches);

- Sistema Terminal-Terra;
- Centro de Despacho;
- Sistema de Controle da Rede; e
- Sistema de Controle das Operações.

2.6 ANÁLISE DOS RESULTADOS DO ESTUDO DE CASOS

Para tornar possível a realização de uma análise quantitativa no estudo dos três casos apresentados, utilizando os elementos-chave levantados na Seção 2.4, contabilizou-se o número de vezes que esses elementos-chave foram utilizados em cada nível, colocando-se esses dados gráficos para uma melhor visualização de resultados e posteriores análises.

2.6.1 ANÁLISE QUANTITATIVA

As colunas nos gráficos (eixo Y) representam a freqüência (quantidade de utilização) que o elemento-chave assinalado no eixo x aparece para cada um dos níveis de abstração utilizados (nível l, nível 2 e nível 3). Os diferentes níveis são representados por preenchimentos diferentes nas colunas e a quantidade dos elementos pode depender do tipo de sistema.

O primeiro estudo de caso tornou possível a construção do gráfico comparando os três níveis de abstração utilizados porque permitiu o detalhamento completo do sistema no nível 3. Já nos estudo dos casos seguintes não foi possível fazer esta comparação, porque no nível 3 de cada um está retratado apenas o detalhamento de uma parte do sistema, sendo selecionado para isto somente uma ou duas porções do nível 2. Para estes casos, foram elaborados dois gráficos: um comparando os níveis de abstração 1 e 2 e outro comparando a(s) porção(ões) do nível 2 selecionadas e seu respectivo detalhamento no nível 3.

2.6.1.1 ANÁLISE DO PRIMEIRO ESTUDO DE CASO

Na Figura 2.6 é apresentado o gráfico referente à análise do primeiro estudo de caso, a Fábrica de Artefatos.

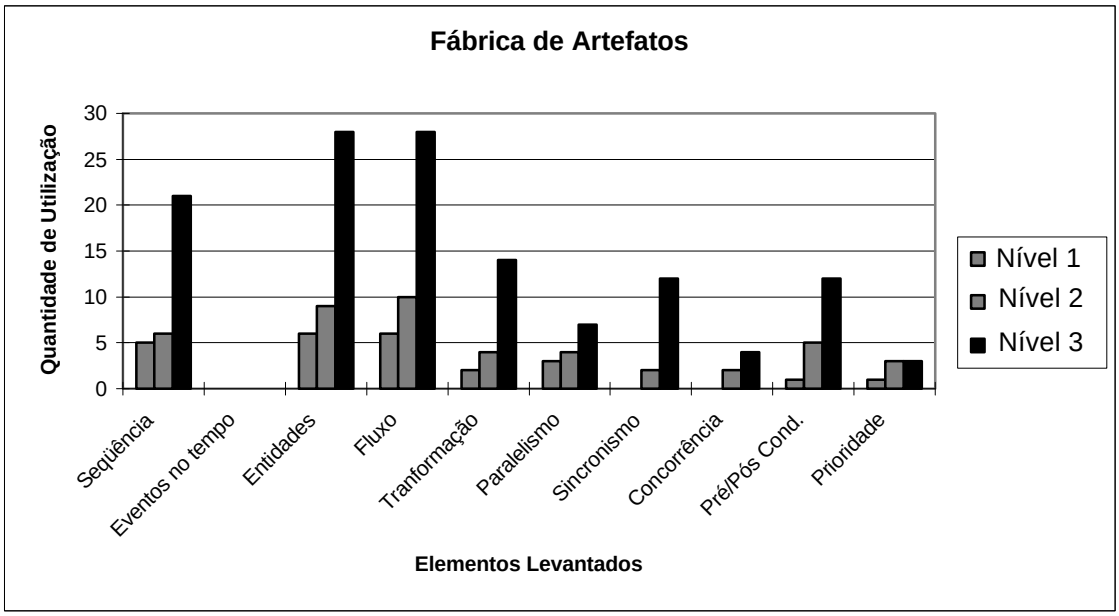

Fig. 2.6 - Análise dos resultados para a Fábrica de Artefatos.

Pode-se observar neste gráfico a presença de seqüências (EC1) em todos os níveis de abstração. Esta presença aumenta conforme mais detalhes são incorporados ao modelo nos níveis seguintes. Resultados análogos foram obtidos para as entidades componentes e fluxo de dados (elementos-chave EC3 e EC4, respectivamente). Quanto as transformações e as pré e pós-condições (EC5 e EC9), também aparecem com maior intensidade nos níveis 2 e 3, como era de se esperar.

A presença de paralelismo, elemento-chave EC6, é menos intensa, conforme a granulosidade de detalhes das porções levantadas aumenta, pois elas tornam-se cada vez mais localizadas. O sincronismo, elemento-chave EC7, aparece a partir do nível 2 e sua presença aumenta bastante no nível 3, conforme as porções se tornam mais detalhadas.

A concorrência, elemento-chave EC8, só aparece no nível 2 e continua discretamente no nível 3, porque as porções são tratadas mais localmente. As pré e pós-condições, elemento-chave EC9, em encontra-se presente nos três níveis, aumentando sua presença nos níveis mais detalhados, à medida que o sistema é melhor especificado. As prioridades, elemento-chave EC10, estabelecidas no nível 1 mantêm-se nos níveis 2 e 3 com a mesma intensidade. O tempo, elemento-chave EC2, não foi necessário para a

representação de nenhuma das porções levantadas, que tomou como base a descrição do sistema.

2.6.1.2 ANÁLISE DO SEGUNDO ESTUDO DE CASO

A análise do segundo estudo de caso é mostrada nos gráficos das Figuras 2.7, 2.8 e 2.9.

O gráfico da Figura 2.7 compara os níveis 1 e 2 do Software de Controle de Satélites. Observa-se que o número de seqüências, elemento-chave EC1, aumenta consideravelmente no nível 2 em relação ao nível 1, e os eventos no tempo, elemento-chave EC2, aparece pela primeira vez, no nível 2. Entidades componentes, elemento-chave EC3, e fluxo de dados e entidades, elemento-chave EC4, aparecem em quase todas as porções do sistema levantadas do nível 2.

As transformações, elemento-chave EC5, e as pré e pós-condições, elemento-chave EC9, aumentam sua presença no nível 2, à medida que mais detalhes são incorporados.

O paralelismo, elemento-chave EC6, só aparece no nível 2, em algumas porções do sistema.

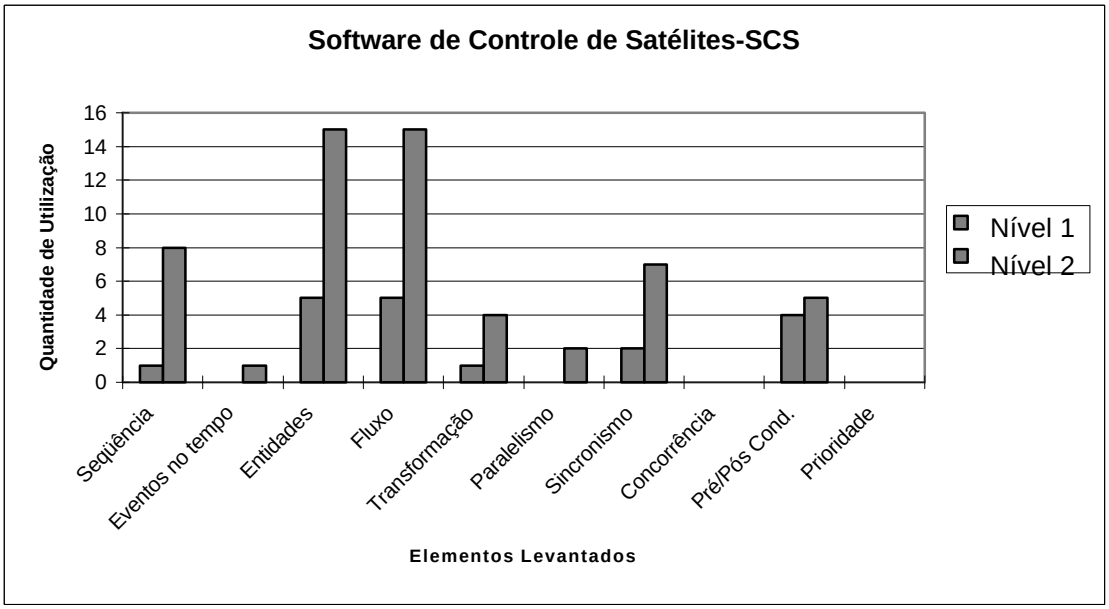

Fig. 2 7 - Análise dos resultados para o Software de Controle de Satélites: nível 1 e nível 2.

A presença de sincronismo, elemento-chave EC7, aumenta nas porções detectadas no nível 2 em relação aquelas do nível 1, à medida que mais detalhes são incorporados ao modelo.

A concorrência, elemento-chave EC8, e o estabelecimento de prioridades, elemento-chave EC9, não estão presentes nas porções levantadas para estes dois níveis.

O gráfico da Figura 2.8 mostra a análise dos níveis 1 e 2 do Sistema da Estação Terrena.

Este gráfico confirma a tendência mostrada nos gráficos anteriores. Seqüências (EC1), entidades componentes (EC3), fluxo de dados e entidades (EC4) e as pré e pós-condições (EC9) aparecem intensamente nos níveis 1 e 2 e as transformações são mais pronunciadas no nível 2.

Os eventos no tempo (EC2), o paralelismo (EC6), o sincronismo (EC7), a concorrência (EC8) e estabelecimento de prioridades (EC10) não estão presentes nas porções do sistema levantadas nos níveis 1 e 2.

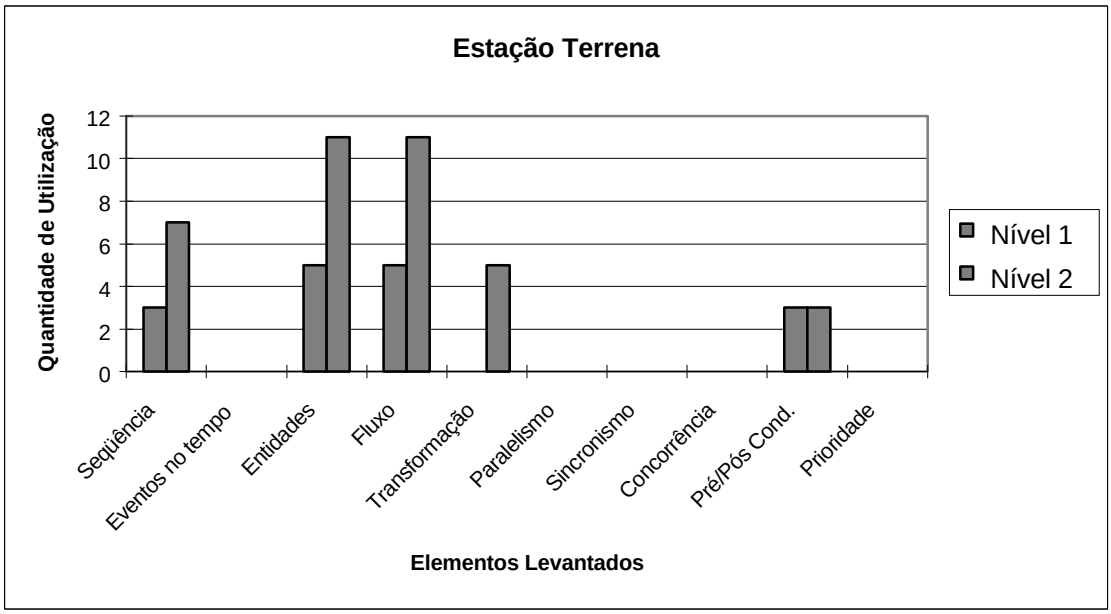

Fig. 2.8 - Análise dos resultados para o Software Operacional da Estação Terrena: nível 1 e nível 2

O gráfico da Figura 2.9 mostra a análise resultante do detalhamento da porção 2 do nível 2 do Software de Controle de Satélites, que corresponde ao nível 3.

Para melhor entendimento e facilidade de consulta, a porção 2 é colocada novamente abaixo:

> Porção 2 - Transmissão de telecomandos para a Estação Terrena e recebimento de mensagens referentes a telecomandos da Estação Terrena, incluindo o recebimento de alarmes relativos a operação de telecomandos.

Observa-se no gráfico da Figura 2.9 a tendência já observada nos outros gráficos. A presença de seqüências (EC1) aumenta no nível 3 bem como a de entidades componentes (EC3) e fluxo de dados e entidades (EC4). As transformações (EC5) só aparecem no nível 3.

O paralelismo (EC6) e as pré e pós-condições (EC9) aparecem no nível 2 e seu aumento é moderado no nível 3 em relação ao nível 2. O sincronismo (EC7) tem a mesma presença nos dois níveis. O tempo (EC2), a concorrência (EC8) e as prioridades (EC10) não estão presentes nas porções levantadas no gráfico da Figura 2.9.

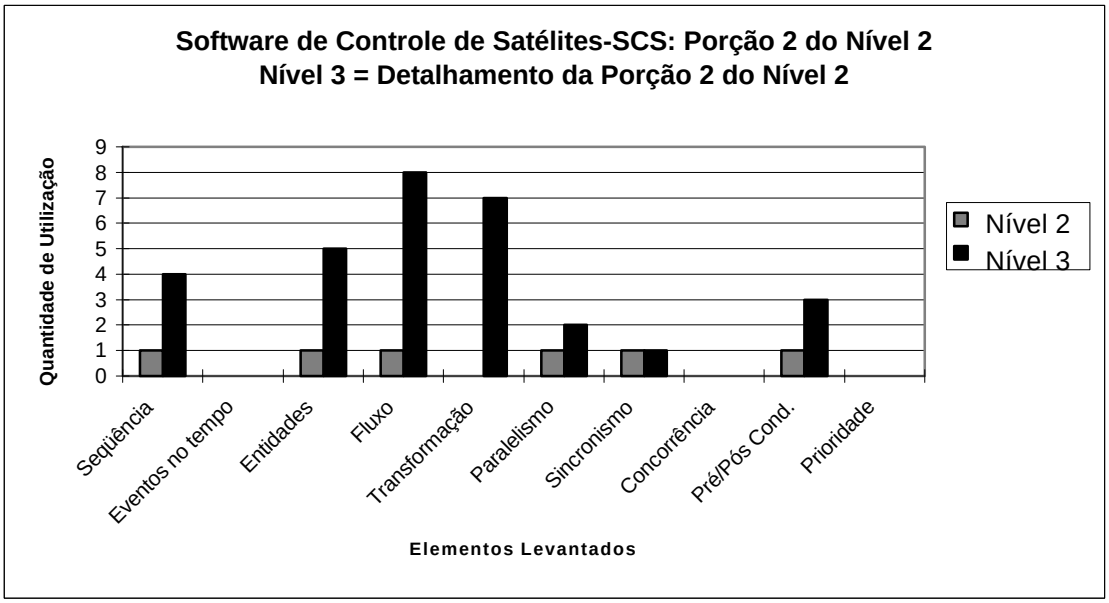

Fig. 2.9 - Análise dos resultados para o Software de Controle de Satélites: porção 2 do nível 2 e nível 3.

2.6.1.3 ANÁLISE DO TERCEIRO ESTUDO DE CASO

A análise do terceiro estudo de caso, o Sistema de Gerência de Tráfego de Trens, é apresentada nos gráficos das Figuras 2.10 e 2.11.

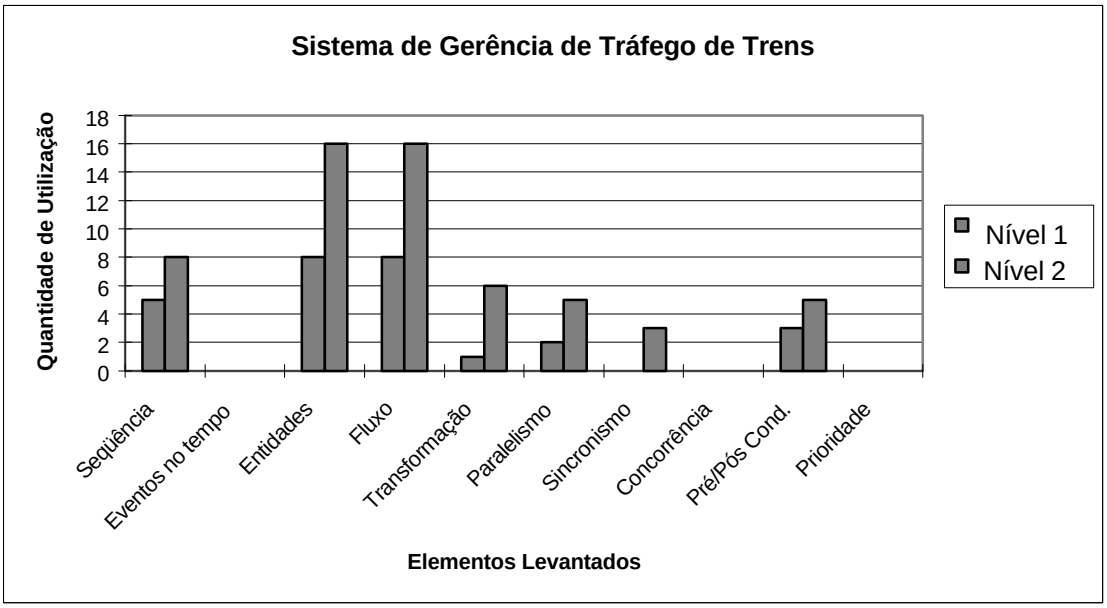

Fig. 2.10 - Análise dos resultados para o sistema de Gerência de Tráfego de Trens: nível 1 e nível 2.

O gráfico da Figura 2.10 mostra os níveis 1 e 2 do Sistema de Gerência de Tráfego de Trens. Pode-se observar a presença de seqüências (EC1) já no nível 1 e no nível 2. Entidades componentes (EC3) e fluxo de dados e entidades (EC4) presentes no nível 1, aumentam de intensidade no nível 2, dobrando sua presença.

A presença de transformações (EC5) e pré e pós-condições (EC9) são maiores no nível 2 em relação ao nível 1, conforme mais detalhes são incorporados ao modelo.

Observa-se um aumento na presença do paralelismo (EC6) do nível 1 para o nível 2, talvez evidenciando a natureza concorrente do sistema e a falta de conhecimento para um detalhamento melhor das porções.

O sincronismo (EC7) aparece somente no nível 2, à medida que mais detalhes são incorporados ao modelo.

O tempo (EC2), a concorrência (EC8) e o estabelecimento de prioridades (EC10) não estão presentes nas porções levantadas no gráfico da Figura 2.10.

A Figura 2.11 mostra a análise dos resultados quando são comparadas as porções 1 e 2 do nível 2 e seu posterior detalhamento, correspondendo ao nível 3.

Para um melhor entendimento e facilidade de consulta, as porções 1 e 2 são transcritas novamente a seguir:

> Porção 1 - Recebimento dos dados dos sensores que estão a bordo de cada trem pelo Sistema de Análise e Relatórios do trem, análise dos dados, apresentação destes dados ao engenheiro do trem e registro dos valores dos sensores num registro *log*;
>
> Porção 2 - Envio de alarmes pelo Sistema de Análise e Relatórios para o engenheiro do trem em caso de valores coletados dos sensores fora dos limites estabelecidos.

Observa-se no gráfico da Figura 2.11 que o número de seqüências (EC1) aumenta muito pouco no nível 3 em relação ao nível 2, talvez devido à própria falta de um maior conhecimento do sistema sendo modelado.

Quanto a representação dos elementos-chave entidades componentes (EC3), do fluxo de dados e entidades (EC4) e de pré e pós-condições (EC9), eles aparecem nos dois níveis, sendo consideravelmente mais intensos no nível 2. O número de transformações (EC5) no nível 2 (uma) é atribuído à porção 1 e o nível 3 retrata esta transformação em duas de suas oito porções levantadas.

O paralelismo (EC6) aparece com igual quantidade nos dois níveis, o que é de se esperar, à medida que mais detalhes do sistema são incorporados na modelagem do nível 3 e as porções ficam mais localizadas.

O sincronismo (EC7), como já era esperado, aparece em um número bem maior de vezes no nível 3, acentuando o caráter mais detalhado deste nível.

O tempo (EC2), a concorrência (EC8) e o estabelecimento de prioridades (EC10) não estão presentes nas porções levantadas no gráfico da Figura 2.11.

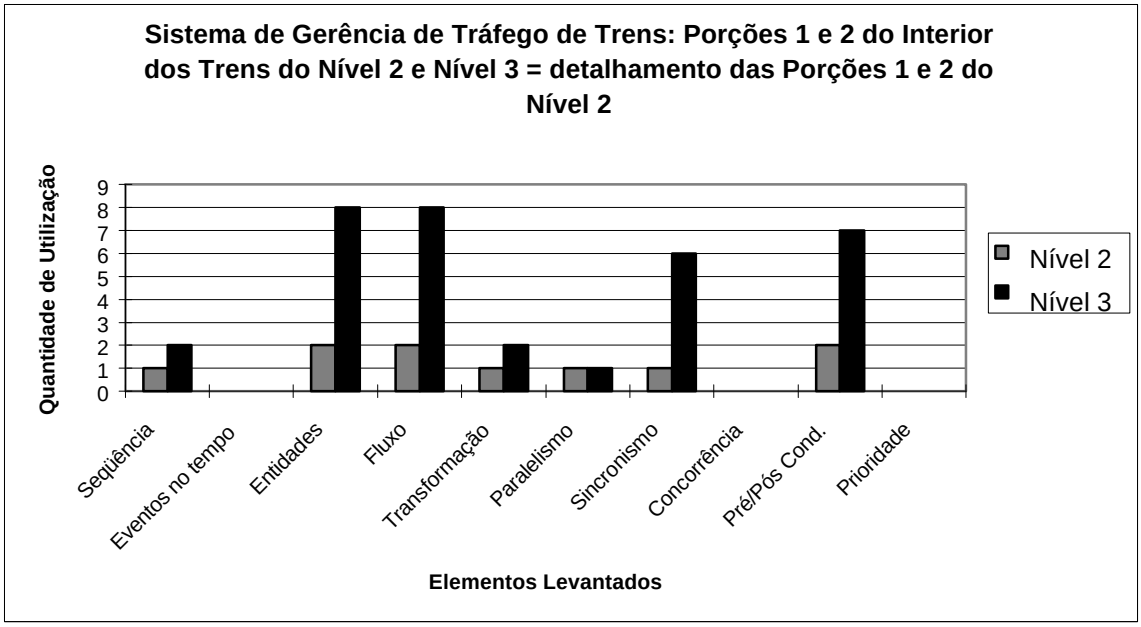

Fig. 2.11 - Análise dos resultados para o sistema de Gerência de Tráfego de Trens: porções 1 e 2 do nível 2 e nível 3.

2.6.2 ANÁLISE FINAL DO ESTUDO DE CASOS

As seqüências (EC1) encontram-se presentes nos três níveis de todos os casos estudados e sua intensidade entre os níveis é moderadamente variada, sendo freqüentemente mais intensa nos níveis mais detalhados dos modelos. Sua importância na modelagem pode ser sem dúvida reconhecida.

A representação dos evento no tempo (EC2) foi somente utilizada em um estudo de caso (Software de Controle de Satélites), porém ela já se fez necessária no nível 1 e continuou também presente no nível 2. Isto retrata a importância da modelagem deste aspecto já na construção dos primeiros modelos de certos sistemas.

Nota-se que o elemento EC2 não aparece nas porções levantadas do Sistema de Gerência de Tráfego de Trens, em nenhum dos níveis considerados, apesar de ser um sistema com características de tempo real. Este exemplo comprova os comentários feitos à respeito das especificações de sistemas no início do Capítulo. Se a especificação cobrisse com mais detalhes a forma de aquisição dos dados ou informações sobre o plano de movimentação dos trens, certamente este elemento teria sido necessário na representação de algumas das porções levantadas neste estudo de caso.

A presença de entidades componentes (EC3) é observada intensamente em todos os níveis. Este fato reforça a idéia já colocada de que a representação dos principais componentes do sistema é importante quando se constrói os primeiros modelos. Sua importância também é sentida à medida que os modelos vão se tornando mais específicos e localizados em um determinado componente.

Quanto ao fluxo de dados ou entidades (EC4), sua presença é unânime em todos os níveis, o que também já era um resultado esperado. Isto constata a preocupação das primeiras metodologias e técnicas estruturadas.

As transformações (EC5), apesar de presentes em todos os níveis de abstração, aparecem mais intensamente nos níveis 2 e 3, à medida que mais detalhes são considerados e as transformações passam a ser melhor modeladas.

O paralelismo (EC6) está presente no nível 1 de quase todos os casos estudados, com exceção do Sistema de Controle de Satélites. À medida que o nível de abstração torna-se mais refinado, o paralelismo passa a ter uma presença mais moderada. Isto constata sua importância no início da modelagem e sua diluição à medida que mais detalhes são incorporados ao modelo.

O sincronismo (EC7), ao contrário do paralelismo, aparece mais acentuadamente nos níveis mais baixos de abstração, porque passa-se a tratar aspectos mais específicos de sincronia entre as entidades componentes envolvidas. Quanto maior o porte do sistema, alguns dos elementos-chave precisam de maior detalhamento para se manifestarem. Por outro lado, quando os detalhes atingiram um certo nível, certos elementos-chave já não aparecem.

A concorrência (EC8) está presente no estudo de caso da Fábrica de Artefatos. Ela já aparece no nível 1 e diminui de intensidade no nível 2, quando passa-se a considerar porções mais localizadas do sistema. Convém lembrar que a concorrência detectada aqui é aquela ligada a partes de maior granulosidade do sistema e não a detalhes mais finos de projeto e implementação como recursos de CPU e acesso concorrente a dados.

Uma conclusão relevante que pode ser colocada neste momento é que o paralelismo e a concorrência são elementos-chave importantes a serem considerados desde os níveis

mais altos de abstração na criação dos modelos dos sistemas, resgatando um aspecto da modelagem que normalmente só é considerado nas fases de projeto detalhado e implementação.

O uso de pré e pós-condições (EC9) é observado em todos os níveis de abstração dos casos estudados, reforçando a importância da sua presença como um dos elementos dos modelos dinâmicos.

O estabelecimento de prioridades (EC10), apesar de presente somente em um estudo de caso, o da Fábrica de Artefatos, tem um papel importante no início da modelagem. Sua necessidade de especificação surgiu logo na criação dos modelos iniciais, e sua representação deve ser considerada sempre que possível. A prioridade continua sendo representada também nos níveis mais baixos de abstração.

Com o exposto acima, observou-se que todos os elementos-chave levantados foram necessários para a representação da dinâmica dos sistemas dos casos analisados, considerando a modelagem num alto nível de abstração e a especificação dos sistemas fornecida. Não é possível garantir que estes elementos-chave sejam suficientes e nem que formem um conjunto mínimo de elementos-chave, porém acredita-se que com os casos estudados e a análise feita, foi possível mostrar a existência de sistemas onde pode-se especificar e representar a dinâmica de maneira mais satisfatória do que tem sido feita até agora.

Por outro lado, os elementos-chave levantados poderão servir como base para a verificação (*check points*)da abrangência e da profundidade dos aspectos dinâmicos considerados em uma dada especificação.

As técnicas hoje disponíveis para a representação da dinâmica dos sistemas, tanto em abordagens estruturadas como orientadas a objeto (Alves, 1998), não oferecem a cobertura desejada em relação a estes elementos-chave desejáveis na composição do modelo dinâmico.

Baseado nestes resultados, o Capítulo seguinte propõe o uso de técnicas de simulação e animação para a construção dos modelos dinâmicos dos sistemas na fase de especificação dos requisitos, constituindo assim o que se pode chamar de especificações

gráficas animadas dos sistemas ou modelos de animação. Com esta nova abordagem, os elementos-chave considerados anteriormente poderão ser representados nos modelos de animação. Estes modelos serão construídos utilizando uma nova técnica de modelagem, que será apresentada no Capítulo 4.

CAPÍTULO 3

ESPECIFICAÇÕES GRÁFICAS PARA A REPRESENTAÇÃO DA DINÂMICA DOS SISTEMAS

3.1 ESPECIFICAÇÃO DE SISTEMAS

Os sistemas atuais freqüentemente exibem um comportamento dinâmico complexo e requisitos de tempo críticos, que são difíceis para os usuários especificarem completamente e de forma consistente. Em adição a isto, as especificações de requisitos dos sistemas feitas pelos usuários tendem a ser informais, vagas e ambíguas.

O uso de linguagens de especificação formal com sintaxe matemática e semântica tem sido proposto na esperança de fornecer maior habilidade aos usuários e analistas para claramente especificar os requisitos de um sistema. Vários métodos matemáticos têm sido desenvolvidos, baseados em especificações formalmente representadas, para verificar propriedades de tempo real importantes, tais como, consistência e segurança (Bicarregui et al, 1994; Heimdahl e Leveson, 1995; Kang e Ko, 1995a).

Porém, estes métodos de verificação formal são freqüentemente usados para analisar somente algumas propriedades, normalmente as mais críticas do sistema. Para a porção restante do sistema não verificada, a simulação pode ser usada para validar requisitos do usuário, visualizando o comportamento do sistema por meio da animação gráfica.

Deve ser observado que as simulações podem somente melhorar o nível de confiança sobre a especificação, mas não podem garantir a correção das especificações. Assim um método combinado de verificação formal e validação baseada em simulação parece ser uma abordagem intermediária para uma engenharia de requisitos efetiva e prática.

3.2 ESPECIFICAÇÕES GRÁFICAS ANIMADAS DE SISTEMAS

Neste trabalho, o termo **modelo de animação** será utilizado para identificar os modelos gráficos de um sistema que utilizam-se da animação para simular a sua dinâmica e seu comportamento, como uma forma de visualização do modelo dinâmico.

O modelo de animação é construído inicialmente, num alto nível de abstração, para representar as especificações de um sistema, dando origem às chamadas **especificações gráficas animadas**. Neste trabalho, os termos modelo de animação e especificações gráficas animadas possuem o mesmo significado e podem ser utilizados alternativamente.

Com a criação do modelo de animação no início da modelagem, a dinâmica do sistema poderá ser melhor compreendida, e decisões arquiteturais e compromissos com desempenho e flexibilidade poderão ser melhor analisados, sem que estas decisões sejam adiadas para fases posteriores do projeto, onde os custos de correção são maiores e as surpresas são indesejáveis.

A idéia de animar modelos gráficos dos sistemas não é nova. Várias pesquisas nesta área encontram-se documentadas (Gaskell e Phillips, 1994). Apesar disto, tais pesquisas estão ainda em sua infância, principalmente em relação aos ambientes CASE e a aplicação prática destas idéias.

Muitos dos argumentos favoráveis à adoção de especificações formais (Fuchs, 1992) também se aplicam às especificações gráficas animadas. Um problema que é colocado em relação à esta última é o de como combinar o tipo de informalidade e flexibilidade, que é natural para a maioria dos usuários, com a rigidez da especificação formal de um sistema (Tate, 1990). Há assim um dilema entre linguagens de especificação formal e as técnicas mais populares, menos formais, dos modelos gráficos.

As especificações gráficas animadas podem fornecer uma maneira rápida de construir protótipos do comportamento funcional de um sistema (Burns, 1986; Lea, 1990). Se as interfaces externas do sistema puderem ser simuladas e ligadas a estas especificações, elas podem exibir o comportamento proposto (Harbert, 1990).

A validação dos requisitos é um dos principais propósitos da prototipação, e as especificações gráficas animadas têm um grande potencial nesta área. Porém, mesmo que as especificações possam ser animadas, elas ainda são fundamentalmente especificações, com um componente adicional que auxilia a sua compreensão (Tate, 1990; Gaskell e Phillips, 1994).

A prototipação (Sommerville, 1989) envolve o desenvolvimento de um programa para experimentos do usuário. Especificações animadas são muito mais diretas do que isto, enquanto o desenvolvimento de um programa protótipo requer seu próprio projeto e atividades de programação, testes e implementação.

Além de permitir que aspectos funcionais de um sistema sejam validados, a animação gráfica de uma especificação pode fornecer também uma plataforma, na qual estudos sobre vários aspectos não funcionais de um sistema, tais como, tempo, desempenho e implementação dos modelos, possam ser realizados (Gaskell e Phillips, 1994).

Um modelo dinâmico do sistema pode ser assim testado com a incorporação de restrições do mundo real, tais como número limitado de processadores, prioridades de processos e tempo finito de execução. Dado um conjunto de restrições, isto ajudaria na investigação da viabilidade da implementação.

As ferramentas CASE, também deveriam oferecer suporte para animação dos vários modelos produzidos durante o ciclo de vida do desenvolvimento de software. As atuais, geralmente fornecem um bom suporte para a criação de diagramas e armazenamento de informações, com o uso de editores gráficos e repositórios de dados.

A capacidade de animação das especificações dos sistemas formarão uma regra predominante no futuro das ferramentas CASE. Apesar de importantes, especificações gráficas animadas formam somente uma fração do potencial de desenvolvimento, se integradas em ambientes CASE. Se estas ferramentas fornecerem suporte para manter um modelo de animação durante todo o ciclo de vida, desde os estágios dos requisitos iniciais até a implementação, é improvável que a divisão entre as várias fases do ciclo de vida tradicional seja mantida.

3.3 A IMPORTÂNCIA DA ANIMAÇÃO DOS MODELOS

Do ponto de vista de um analista e também de um especialista no domínio do problema, a animação é uma ferramenta poderosa para ajudar na compreensão da dinâmica e do comportamento de um sistema. A animação fornece, de uma maneira amigável, um conjunto de informações a respeito da seqüência de eventos gerados pelo modelo. Este traçado visual permite uma abordagem que é muito superior àquela das ferramentas clássicas, porque ela possibilita o acompanhamento paralelo das várias entidades navegando pelo sistema.

A animação é muito útil para entender as várias interações entre as entidades concorrentes. Interações complexas entre várias entidades do modelo são freqüentemente difíceis de compreender, especialmente quando se está limitado ao traçado no papel. Com a animação é possível ainda representar o comportamento transiente do sistema.

A análise da animação pode também ser utilizada com o objetivo de testar hipóteses. A animação pode inspirar alternativas para as soluções implementadas. Testes de regras de gerenciamento complexo, intencionadas para seguir os gargalos e fazer uso de um grande número de entidades do sistema, poderiam ser avaliadas mais facilmente de uma forma visual.

A animação é uma ferramenta de apresentação e comunicação, aumentando a possibilidade de diálogo entre analistas, usuários e especialistas no domínio do problema.

O emprego de técnicas de simulação discreta de sistemas pode ser visto como uma maneira de se criar o modelo de animação para o modelo dinâmico, que representa uma especificação de sistema, e será a técnica adotada neste trabalho. A Seção seguinte faz uma introdução à simulação de sistemas, destacando aspectos de interesse para o objetivo do trabalho.

3.4 SIMULAÇÃO

Shannon (1975) coloca a simulação como o processo de se elaborar um modelo de um sistema real e conduzir experimentos com esse modelo, tendo como propósito a compreensão do comportamento do sistema ou a avaliação de diversas estratégias (dentro dos limites impostos por um critério ou conjunto de critérios) para a operação do sistema.

A definição acima é importante para que se possa entender como a simulação será utilizada no contexto deste trabalho. O modelo de simulação é normalmente construído para um sistema conhecido, ou seja, existente ou já concebido mentalmente, no qual a precisão e a fidelidade do modelo em relação ao sistema real são aspectos importantes considerados para a sua validação.

O comportamento estocástico do modelo de simulação é determinado por meio de um levantamento de dados dos sistemas reais, feito anteriormente, necessários para a construção do modelo que será simulado, diferentemente de sistemas novos, que estão ainda sendo especificados e não possuem nenhum similar no mundo real.

A organização cronológica total do processo de modelagem e simulação é conhecida como ciclo de vida do modelo (Balci, 1985), apresentado na Seção 3.4.2. Nele podem ser visualizadas todas as atividades envolvidas, bem como suas inter-relações.

As seções seguintes esclarecem melhor os conceitos envolvidos no processo de construção de um modelo de simulação e na simulação. O uso da simulação como ferramenta para a especificação de sistemas, sugerido neste trabalho, tem um enfoque diferente do tradicional, conforme será explicado no Capítulo 4.

3.4.1 CONCEITOS BÁSICOS DE SIMULAÇÃO

A simulação é uma das técnicas mais poderosas à disposição de cientistas e administradores. Seu uso pode ajudar a planejar e criar sistemas que possuem muitas variáveis, relações complexas entre seus elementos, incerteza sobre seus dados e opções variadas de estruturação. Há muito tempo que ela vem sendo utilizada para uma melhor avaliação das inúmeras alternativas de solução, que ocorrem em problemas das mais diversas áreas do conhecimento humano (Kienbaum, 1996).

Ao longo dos últimos anos, dois fatos vieram a contribuir de forma importante para a popularização do uso da simulação. O primeiro deles foi o aparecimento de computadores relativamente baratos e poderosos, como os microcomputadores e superminicomputadores. O outro, refere-se ao aparecimento de programas especialmente criados para dar ao gerente, ou decisor, a oportunidade de participar do processo de modelagem de muitos de seus problemas.

O desenvolvimento destes programas só tem sido possível graças a avanços conceituais significativos no tocante à modelagem de sistemas. Os progressos mais recentes obtidos pelas pesquisas sobre metodologias e ferramentas são destinados a auxiliar na elaboração de estudos de simulação.

O termo Modelagem por Simulação (*Simulation Modeling)* tem sido usado para descrever o uso da simulação como uma ferramenta para melhorar a compreensão dos sistemas, ao invés de pensar nela como uma técnica voltada para a solução dos problemas.

Segundo Kreutzer (1986), o potencial da simulação para a exploração computacional das estruturas de símbolos dos modelos é muito mais importante do que sua aplicação tradicional, como ajuda na tomada de decisão. Logo, unir um conjunto de ferramentas de software para auxiliar o modelador no desenvolvimento dos estudos de simulação seria desejável.

Segundo Soares (1990), os modelos para simulação podem ser empregados com quatro objetivos:

- Como ferramenta explanatória para a definição de um sistema ou problema;
- Como ferramenta de análise para a detecção de efeitos críticos;
- Como ferramenta para síntese e avaliação de soluções propostas; e
- Como ferramenta de planejamento para desenvolvimentos futuros.

Para uma melhor compreensão de como ocorre o processo de criação de um modelo de simulação, a Seção seguinte apresenta seu ciclo de vida.

3.4.2 CICLO DE VIDA DE UM MODELO DE SIMULAÇÃO

Um estudo de simulação pode ser entendido como um processo de transformação aplicado à informação disponível sobre um sistema, relativa a um determinado objetivo de estudo (Kienbaum, 1996). Este processo é composto das seguintes etapas:

1) A definição do problema (aquisição da informação);

2) Desenvolvimento do modelo, seguido de sua experimentação (a organização e o processamento da informação); e

3) Apoio à tomada de decisão (apresentação da informação obtida, acrescida de outras novas, sob forma transformada).

Dentro de cada uma destas etapas há vários processos simples, que vão transformando a representação do problema original em formas distintas, embora sucessivas e interligadas.

A cada passo do desenvolvimento do estudo deve haver pelo menos um passo correspondente de verificação da qualidade do progresso realizado, o que se convenciona chamar de Estágios de Aferição de Credibilidade (EAC).

A apresentação global de um estudo de simulação segundo este roteiro pode ser feita num diagrama, que recebe o nome de ciclo de vida do modelo. O ciclo de vida do modelo é composto de dez formas de representação do problema, conforme mostrado na Figura 3.1, baseada em Balci (1985). As formas são representadas por símbolos ovais e as setas tracejadas representam os processos.

O ciclo de vida não pode ser entendido como estritamente seqüencial. A direção seqüencial das setas indica apenas a progressão cronológica das atividades ao longo do desenvolvimento do estudo. O ciclo de vida, no entanto, é iterativo e realimentações entre os passos são esperadas.

As formas de representação que se iniciam na **descrição do sistema e de seus objetivos** e culminam com os **resultados do modelo,** correspondem à **etapa de desenvolvimento do modelo,** e é esta etapa que apresenta interesse para este trabalho. Ela possui alguns processos comuns com o ciclo de vida de desenvolvimento de um sistema de software.

A etapa que antecede a etapa de desenvolvimento é chamada etapa de definição do problema, e a que a sucede é chamada de etapa de apoio à decisão.

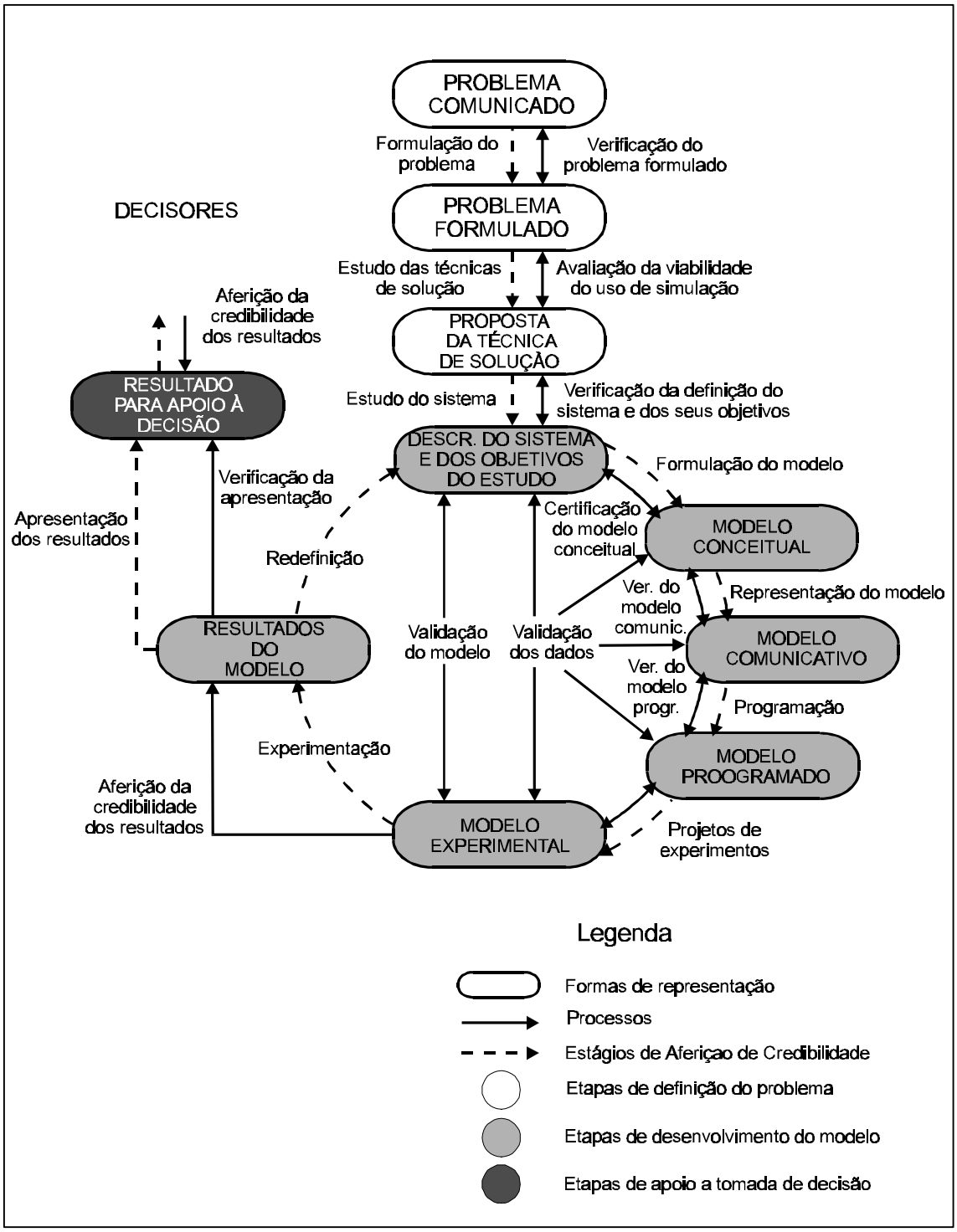

Fig. 3.1 - O ciclo de vida do modelo em um estudo de simulação. Adaptado de Balci, 1985.

A seguir são brevemente descritos os processos que relacionam as formas de representação do problema, representados na Figura 3.1 por setas tracejadas, considerando somente a etapa de desenvolvimento do modelo, cujas formas de representação estão pintadas de cinza. Maiores detalhes sobre os outros processos e formas de representação envolvidos podem ser encontrados em Balci (1985).

- **Formulação do Modelo:** a formulação do modelo é o processo por meio do qual o modelo conceitual do sistema é construído. O modelo conceitual é o modelo que existe na cabeça do modelador (Nance, 1981). O modelo não pode excluir detalhes essenciais do sistema, nem tampouco ele deve incluir detalhes desnecessários. A fronteira do sistema é definida, e separa os elementos que devem ser incluídos no modelo daqueles que devem permanecer fora dele.

- **Representação do Modelo**: é o processo pelo qual o modelo conceitual é transcrito num modelo comunicativo. Um **modelo comunicativo** é uma forma de representação de um modelo que pode ser comunicada para outros seres humanos, podendo ser julgada e comparada em relação ao sistema original e aos objetivos do estudo, por mais de uma pessoa. (Nance, 1981). Um modelo comunicativo pode ser representado de diversas formas diferentes (diagramas de blocos e linguagem natural são algumas das formas de representação). Para um mesmo sistema podem ser elaborados diferentes tipos de modelos comunicativos.

- **Programação**: O processo de programação consiste na transcrição do modelo comunicativo para um modelo programado. Um modelo programado é uma forma de representação de um modelo que admite execução em um computador, para obtenção de resultados (Nance, 1981). Embora o modelo programado possa ser construído utilizando uma linguagem de programação de alto nível, como Fortran, Pascal, C ou o C++, geralmente é preferível a adoção de uma linguagem de simulação.

  Uma linguagem de simulação reduz significativamente o tempo necessário para a elaboração do modelo programado, por colocar à disposição do usuário as seguintes funções pré-programadas : uma estratégia de abordagem (um esqueleto que serve de

base para a programação e controle do programa); um mecanismo de controle do tempo simulado; um gerador de números aleatórios; algumas funções de distribuição estocásticas para a geração de variáveis aleatórias; mecanismos de análise estatística elementares; diagnósticos para facilitar a identificação dos erros de programação.

Exemplos de linguagens de simulação incluem GPSS (Gordon, 1978; Schriber, 1991), QNAP2 (Badel et al, 1991; Hill, 1993), SIMAN (Pegden et al, 1990), SIMSCRIPT e MODSIM (Law e Larmey, 1984), SLAM/TESS (Pritsker, 1986), RESQ (Soares, 1990). Além de poupar tempo de programação, uma linguagem para simulação também ajuda na formulação dos modelos, fornecendo um conjunto de conceitos para sua descrição.

- **Projeto de Experimentos**: consiste em formular uma estratégia que possibilite a coleta da informação desejada sobre o modelo para que o modelador possa fazer inferências válidas sobre ele, incorrendo num custo mínimo (Shannon, 1975). Um modelo experimental é o modelo programado acrescido de uma descrição executável das operações contidas na referida estratégia.

- **Experimentação**: é o processo de experimentar com o modelo com um propósito específico. Alguns propósitos da experimentação são (Shannon, 1975): comparação de diferentes políticas de operação do sistema; avaliação do seu comportamento; análise de sensibilidade; previsão; otimização; determinação de relações funcionais internas do sistema. O processo de experimentação é o que produz os resultados do modelo, que são as saídas obtidas a partir da execução do modelo experimental.

- **Redefinição**: o processo de redefinição consiste em: atualizar o modelo experimental para que ele represente a forma corrente do sistema; alterá-lo para obter um novo conjunto de resultados; modificá-lo sob qualquer outro aspecto, visando sua manutenção; modificá-lo para reutilização em outro contexto ou redefinir um novo sistema a ser modelado, visando uma solução alternativa do problema.

O estudo da simulação atribui importância à fase de concepção e construção do modelo lógico do ciclo de vida do processo de simulação. Desde que estas atividades são as mais criativas e difíceis, o modelador deveria estar apto para produzir modelos melhores e mais rapidamente, adotando esta abordagem. Isto é particularmente verdadeiro se ferramentas adicionais estão disponíveis para as fases do estudo do ciclo, por meio de geradores de programas e avaliação dos resultados, com a aplicação de análises estatísticas e técnicas de inteligência artificial.

O conceito chave do modelo de simulação é o da descrição de estado do sistema. Se um sistema pode ser descrito por um conjunto de variáveis onde uma combinação de valores represente um único estado ou condição do sistema, então a manipulação dos valores das variáveis simula o movimento do sistema de um estado a outro. Este é exatamente o conceito de experimento de simulação que envolve a observação do comportamento dinâmico do sistema através de seu movimento de estado a estado de acordo com regras bem determinadas.

3.4.3 MODELOS DE SIMULAÇÃO

Por trás das diversas metodologias propostas na literatura de simulação, observa-se o interesse em se atingir o ainda distante objetivo de se formalizar o procedimento para a elaboração de modelos de sistemas (modelos conceituais e computacionais).

Duas formas de especificação de sistemas, uma com base na Teoria Geral de Sistemas (Zeigler e De Wael, 1986) e outra usando diagramas de ciclo de atividades (Balci, 1985), ilustram a multiplicidade de abordagens possíveis e as enormes diferenças entre elas.

As definições de sistema e modelos, segundo a Teoria Geral de Sistemas (Zeigler, 1976), estabelecem que **sistema** é uma entidade que consiste de partes em interação que visam alcançar um objetivo comum de forma organizada e **modelos** são representações simplificadas dos sistemas. Os modelos físicos assemelham-se fisicamente aos sistemas que representam, enquanto os modelos abstratos mantêm com estes apenas uma semelhança lógica. O projeto de experimentos é o modelo do sistema na sua forma final, adaptado à condução do experimento.

Além da natureza do sistema, um segundo aspecto é também importante para a determinação das características do modelo: os objetivos do estudo que se pretende realizar. É só após considerar estes objetivos, as características intrínsecas do sistema e a qualidade dos resultados pretendidos, que o analista pode decidir o nível de precisão e de detalhes exigidos para a condução do estudo de simulação. Não há interesse em se produzir um modelo detalhado, se o que se requer são apenas estimativas grosseiras.

As decisões práticas que precisam ser tomadas compreendem dois aspectos principais, que dão origem às formas de classificação da simulação (Kienbaum, 1996):

1) Controle de tempo - instantes de observação do sistema, que é decidido pelo analista e dá origem a duas políticas:

   - A do particionamento em intervalos constantes; e
   - A do próximo evento.

2) Função de transição entre estados - inerentes ao sistema, dando origem a:

   - Predictibilidade:

     Simulação determinística (inteiramente previsível); e

     Simulação estocástica (eventos aleatórios).

   - Forma de ocorrência:

     Simulação discreta (ocorrências discretas no tempo); e

     Simulação contínua (ocorrências contínuas no tempo).

3.4.4 CONTROLE DO TEMPO SIMULADO

A essência de um estudo de simulação é a de modelar as mudanças de estado do sistema em função do tempo. O modelo é visto como algo dinâmico através do tempo simulado, e as ocorrências de eventos precisam se dar na ordem apropriada e com o intervalo de tempo correto entre elas. Com isto o problema que se apresenta ao projetista do modelo é o de representar os componentes do sistema real, que têm funcionamento paralelo, por meio de componentes de um modelo sob a forma de programas.

O intervalo de tempo durante o qual o modelo é simulado é chamado corrida ou rodada de simulação. Este tempo é estipulado pelo modelador e depende do tipo de sistema modelado e do estudo sendo conduzido. O modelador pode determinar várias rodadas de simulação para um mesmo modelo.

O controle do tempo simulado é de extrema importância e desempenha três funções fundamentais: permite o correto seqüenciamento dos eventos que ocorrem no modelo; permite a atualização do estado do modelo; e permite, em alguns casos, controlar a velocidade com que o experimento é realizado.

A Figura 3.2 ilustra as relações entre o tempo simulado e as ocorrências de eventos no modelo para ambos os tipos de formulações. Nesta Figura podem ser vistos os dois mecanismos para controle de tempo que dão origem a dois tipos de políticas para a simulação: incrementos variáveis de tempo (política do próximo evento) e incrementos fixos de tempo (política do passo fixo).

No mecanismo do próximo evento, o processamento dos eventos é feito um por um, seqüencialmente. No mecanismo de incrementos fixos de tempo, o processamento é feito por lotes ou conjuntos de ocorrências.

Os modelos com ocorrências contínuas no tempo geralmente usam o mecanismo de incrementos fixos de tempo. Ele se revela o mais apropriado quando o analista considera que o sistema, que ele está estudando, consiste de um fluxo contínuo de informações ou itens contados de forma agregada, cujo significado individual não é relevante.

Nos modelos de ocorrências discretas no tempo, o analista está interessado no que acontece aos elementos que fluem no sistema. A maioria dos modelos discretos utilizam, portanto, o mecanismo de controle do tipo próximo evento. Pode-se dizer que esta política tem duas vantagens sobre a de incrementos fixos no tempo. A primeira é que o incremento de tempo se ajusta automaticamente aos períodos de alta e baixa atividade do modelo, evitando o exame constante e desnecessário dos seus estados. A outra é que ela mostra claramente os instantes de tempo em que se deram as ocorrências durante a rodada de simulação, que podem ser registrados para posterior referência. Em

contrapartida, muito mais informação precisa ser mantida durante a execução para este tipo de política, e o tempo da rodada flui por saltos.

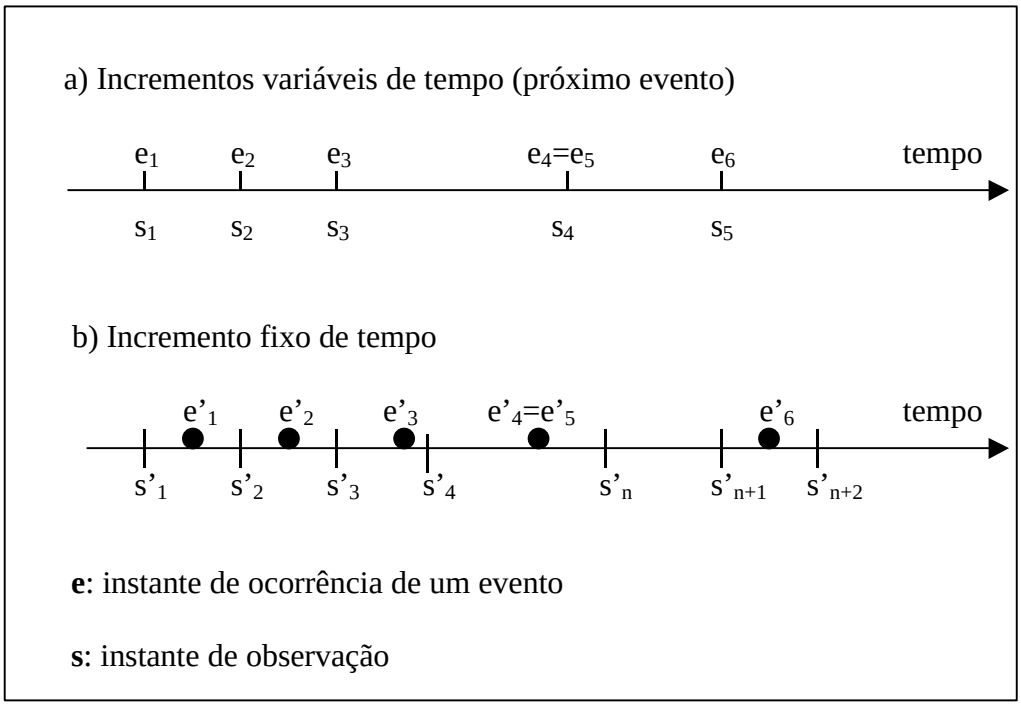

Fig. 3.2 - Progressão do tempo simulado. Adaptado de Shannon (1975).

Alguns tipos de modelos são claramente melhor controlados por um destes mecanismos, enquanto outros admitiriam qualquer um dos mecanismos, alternativamente.

3.4.5 SIMULAÇÃO DETERMINÍSTICA E SIMULAÇÃO ESTOCÁSTICA

Um sistema determinístico é aquele que tem um comportamento inteiramente previsível. Desde que o sistema tenha sido completamente compreendido em seu funcionamento, é possível se predizer precisamente o que nele se passará. Um ciclo de operações de uma máquina automática pode ser visto como um exemplo de um sistema determinístico. Cada ciclo toma exatamente o mesmo tempo para ser realizado, a menos que hajam condições externas que influenciem a sua duração.

Um sistema estocástico é aquele cujo comportamento não pode ser inteiramente previsto, sendo possível apenas se estimar com que probabilidade determinados eventos ocorrem ou qual a duração das tarefas que nele se realizam.

Devido a natureza diferenciada destes dois tipos de sistemas, os modelos que os representam dão origem a dois tipos de denominações diferentes para a simulação (Kienbaum, 1996): Simulação Determinística e Simulação Estocástica.

As variáveis descritivas de um modelo são aquelas capazes de descrever o seu comportamento de forma não ambígua, consistente e completa. Um conjunto de variáveis de estado é um subconjunto das variáveis descritivas do modelo, capaz de determinar de forma unívoca o valor de todas estas variáveis em qualquer instante futuro de tempo, a partir da situação presente e do valor da entrada ao qual o sistema é submetido.

A simulação estocástica é um método de solução que envolve a geração dos valores das diversas variáveis de estado, a partir de suas respectivas distribuições de probabilidade, por meio de números aleatórios ou pseudo-aleatórios.

3.4.6 FORMULAÇÃO DE UM MODELO PARA SIMULAÇÃO DISCRETA

A formulação de um modelo para a simulação discreta pode ser realizado de três formas:

- Pela definição das mudanças nos estados que podem ocorrer em cada tempo de evento;
- Pela descrição das atividades nas quais as entidades do sistema se envolvem; e
- Pela descrição do processo por meio do qual as entidades do sistema fluem.

As relações entre os conceitos de atividade, evento e processo podem ser melhor entendidas pela Figura 3.3 a seguir (Pritsker, 1986).

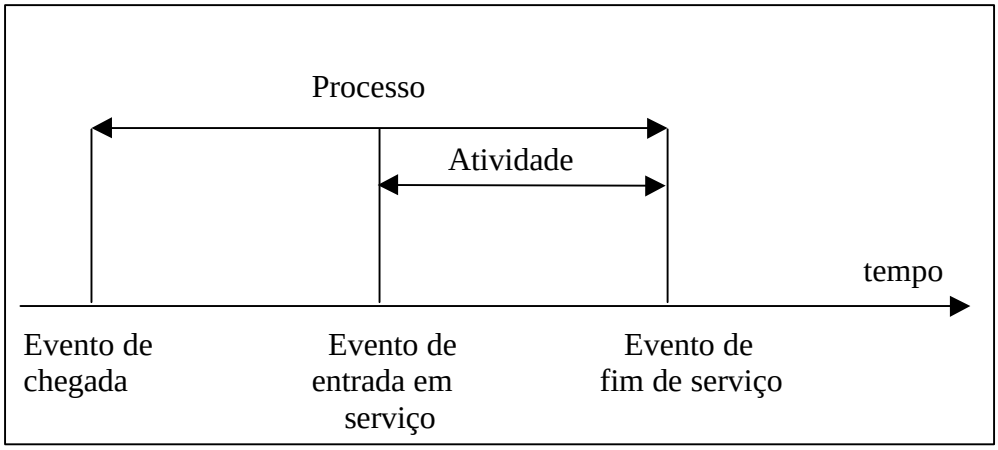

Fig. 3.3 - Relações entre eventos, processos e atividades.

Pode-se dizer que um evento acontece em um ponto isolado do tempo, no qual as decisões devem ser tomadas de forma a iniciar ou terminar uma atividade. Já um processo é uma seqüência ordenada de eventos e pode englobar várias atividades. Estes conceitos levam naturalmente a três alternativas de visão ou enfoque de um modelo para simulação:

- Modelagem orientada a evento;
- Modelagem orientada ao exame da atividade; e
- Modelagem orientada a processo.

O enfoque adotado para este trabalho é o da modelagem orientada a evento, que será melhor explicada na Seção seguinte.

3.4.6.1 MODELAGEM E SIMULAÇÃO ORIENTADA A EVENTO

Na simulação orientada a evento, um sistema é modelado pelas descrições das mudanças que ocorrem em pontos discretos do tempo, chamados de tempos de evento.

A tarefa do modelador é determinar eventos que podem causar as mudanças nos estados do sistema e então desenvolver a lógica associada com cada tipo de evento. A simulação do sistema é produzida pela execução da lógica associada a cada evento, em uma seqüência ordenada no tempo.

O estado de um sistema em um modelo orientado a evento é representado por variáveis, por entidades (e seus atributos) e os estados destas entidades no modelo. O estado do modelo é iniciado pela especificação dos valores das variáveis de estado empregadas na simulação, pela criação das entidades iniciais e pelo escalonamento inicial dos eventos.

O conceito de que os eventos acontecem instantaneamente é crucial. Um evento pode iniciar ou terminar uma atividade, mas as mudanças no estado do sistema só podem ocorrer no tempo de evento.

Quando ocorre um evento, o estado de um sistema pode mudar de quatro modos, ou por uma combinação destes quatro modos (Soares, 1990):

- Pela alteração do valor de uma ou mais variáveis associadas com a simulação;
- Pela alteração do número de entidades presentes no sistema;
- Pela alteração do valor designado a um ou mais atributos da entidade; e
- Pela alteração da relação que existe entre entidades, por meio da manipulação de filas.

A seguir, é apresentado um exemplo de como a ocorrência de eventos pode mudar o estado de um sistema.

Exemplo: Chegada e processamento de um pedido de um cliente, recebido por um atendente:

1) Chegada de um pedido de cliente;
2) Início de processamento do pedido pelo atendente; e
3) Fim de processamento do pedido pelo atendente.

Os três eventos acima provocam, respectivamente, as seguintes mudanças de estados:

1) Ocorre o evento início de processamento;
2) O atendente não estará ocioso, mas envolvido no processamento do pedido durante o tempo estabelecido para isso; e
3) A fila de pedidos será reduzida de 1.

Em qualquer linguagem de simulação orientada a evento, devem existir métodos para a realização de todas estas mudanças. Funções importantes, que devem ser realizadas na simulação orientada a evento, incluem:

- Função primária para o escalonamento dos eventos, colocando-os numa ordem cronológica, e avanço do relógio simulado;
- Mecanismos de manipulação de filas;
- Mecanismos de coleta estatística e suporte estatístico; e
- Mecanismos para geração de amostras aleatórias.

Como a implementação do calendário de eventos e do mecanismo para o processamento dos eventos na ordem cronológica apropriada é uma função comum a todos os modelos de eventos discretos, várias linguagens de simulação foram desenvolvidas fornecendo estas facilidades específicas, assim como outras funções comuns a modelos de evento discreto (Kiviat, 1969; Pritsker, 1974, 1986; Lilegdon e O'Reilly, 1987).

Para sistemas de simulação discreta, a política de controle do tempo simulado normalmente adotada é a política do próximo evento.

3.4.7 PROBLEMAS ESTOCÁSTICOS COM A SIMULAÇÃO

Quando se analisa as medidas de desempenho produzidas pela resolução de um modelo de simulação, elas podem conter erros provenientes de diferentes fontes. Uma fonte particular de erro é a estimação imprecisa dos parâmetros de entrada do modelo. Para a execução do modelo, devem ser fornecidos valores para as variáveis, ou seja, as distribuições de tempo de serviço, distribuições de tempo de chegada, probabilidades de rotas, uso de um recurso passivo, população das filas, disciplina das filas, entre outros. Os valores fornecidos são simples estimativas dos parâmetros reais. Alguns destes parâmetros são críticos na solução do modelo e pequenos erros cometidos em sua estimativa podem redundar em grandes erros nas medidas de desempenho.

Existem também erros inerentes ao modelo. A estrutura do modelo pode estar incorreta, podendo omitir alguns elementos importantes, ou seja, o modelo pode conter alguns

erros lógicos. Esses erros não são tão difíceis de lidar em relação aos erros de estimação de parâmetros mencionados no parágrafo anterior.

Ao se resolver um modelo por simulação, podem também ser introduzidos erros concernentes à variabilidade estatística dos resultados, ao transitório da simulação, à geração de amostras aleatórias, entre outros.

3.4.8 CONDIÇÕES INICIAIS

Implícitas em todo o problema de simulação estão as condições iniciais ou estado de partida da simulação (Soares, 1990). O estado inicial mais simples e mais usado é o estado ocioso ou vazio, no qual a simulação começa com nenhuma entidade no sistema e todos os servidores desocupados. Se este estado é apropriado ou não, depende do sistema a ser modelado e do interesse na determinação do transitório ou do estado permanente de um sistema.

Quando o propósito da modelagem é o comportamento estacionário, freqüentemente pode-se melhorar a estimativa da média, partindo de um outro estado que não seja o ocioso. Já para a análise do transiente, a condição de partida deve espelhar o estado inicial do sistema.

Assim, três regras básicas foram propostas (Pritsker, 1986) para o estabelecimento das condições iniciais do sistema:

- Comece o sistema do estado vazio ou ocioso;
- Comece o sistema no estado de maior probabilidade de ocorrência; ou
- Comece o sistema na média do estado de equilíbrio.

3.4.9 SIMULAÇÃO E ORIENTAÇÃO A OBJETOS

Os especialistas da área de simulação estão cada vez mais interessados em construir ambientes amigáveis que possibilitem aos não especialistas a construção de seus modelos de simulação mais facilmente (Balci at al, 1997). A produção de um ambiente de simulação visto desta maneira constitui o equivalente às ferramentas CASE na área de Engenharia de Software.

O desejo de colocar juntas as técnicas avançadas de Engenharia de Software e as técnicas de simulação, utilizando o paradigma de orientação a objetos, é compartilhado pelas duas comunidades. O modelo de objeto não está limitado somente às técnicas de programação, desde que ele engloba também métodos de análise e projeto de software, o que é de interesse para a criação dos ambientes de simulação. Além disso, as pesquisas em Engenharia de Software têm produzido técnicas de verificação de software que podem ser utilizadas para verificar e validar o software de simulação.

A simulação tem procurado explorar algumas características da orientação a objetos que, segundo Hill (1996), são:

- O uso de todos os conceitos de orientação a objetos;
- A separação dos aspectos estáticos e dinâmicos dos sistemas a serem modelados;
- A promoção da reutilização, não somente do código produzido, mas também da análise e projeto; e
- O destaque dos aspectos dinâmicos dos sistemas, tais como, interações entre os objetos, comportamento interno dos objetos e transações entre os sistemas.

A linguagem Simula (Dahl et al, 1968; Kirkerud, 1989) representa uma das primeiras tentativas de se colocar juntas estas características.

Um sistema de software desenvolvido, usando uma abordagem orientada a objetos, pode ser considerado um sistema de objetos que se comunicam por meio da troca de mensagens, sendo que cada objeto é responsável pelo cumprimento de determinadas tarefas que dependem de seu estado.

A visão de um sistema de software orientado a objetos é similar aquela usada na simulação discreta de eventos para modelar as várias entidades de um sistema real e descrever suas interações durante a simulação. Há, assim, uma importante analogia entre o desenvolvimento de software orientado a objetos e a produção de um modelo de simulação.

Assim como a área de simulação tem usufruído das potencialidades das técnicas de orientação a objetos para criar seus ambientes e modelos de simulação, a Engenharia de

Software pode utilizar-se das potencialidades da simulação, principalmente aquelas concernentes à animação visual dos modelos, aliada também às técnicas de orientação a objetos, para criar modelos de animação que representem a dinâmica dos sistemas de software, como é explorado neste trabalho. Com esta abordagem, deseja-se que um desenvolvimento uniforme possa ser conduzido, desde a concepção dos modelos iniciais, até a especificação dos objetos que serão realmente implementados, mantendo a visão da dinâmica do sistema como um todo.

3.4.10 PACOTES PARA SIMULAÇÃO

É importante mencionar nesta investigação sobre a área de simulação, alguns dos diversos pacotes de modelagem e algumas das diversas linguagens de simulação existentes. O uso generalizado da simulação, como uma ferramenta de análise de sistemas, deu origem a uma série de linguagens de modelagem especificamente projetadas para este fim. Estas diversas linguagens e pacotes de modelagem impõem uma certa estruturação aos modelos e simplificam suas soluções.

A estratégia de se construir e analisar um modelo de simulação, utilizando uma linguagem de alto nível qualquer é viável, mas não é normalmente a melhor estratégia em termos de esforço humano despendido para a resolução de problemas. O uso de um pacote de modelagem apropriado geralmente é a ferramenta mais recomendável (Soares, 1990). Os pacotes podem ser divididos em duas grandes categorias: pacotes de simulação de uso específico e pacotes de simulação de uso geral, conforme descritos nas seções seguintes.

3.4.10.1 PACOTES DE SIMULAÇÃO DE USO ESPECÍFICO

São pacotes voltados à avaliação de desempenho de sistemas particulares. Com o crescimento do uso da simulação, surgiu a necessidade de linguagens orientadas à resolução de problemas específicos e várias linguagens foram projetadas com este objetivo.

Nestes pacotes, a formulação do modelo é construída na própria ferramenta, sendo os parâmetros do modelo especificados por meio de uma linguagem relacionada ao

domínio do sistema modelado. Quando o sistema se adapta à linguagem de modelagem, é mais fácil construir o modelo e analisar o sistema.

Os pacotes desta categoria, SIMUFLEX (Hill, 1996), BEST/1 (Waltham, 1983), MAP/1 (Miner e Rolston, 1986), SAINT (Wortman et al, 1978), SNAP (Polito, 1983), MICROSAINT (Micro Analysis & Design Simulation Software, 1992) entre outros, oferecem aos usuários que são especialistas em modelagem, facilidades para construir modelos, pois o formalismo usado é muito próximo daquele usado na indústria. A limitação principal é que eles somente sabem lidar com um número restrito de problemas que existem no mundo real: aqueles para os quais foram projetados.

3.4.10.2 PACOTES DE SIMULAÇÃO DE USO GERAL

São pacotes de simulação projetados para a modelagem de sistemas de vários tipos. Alguns exemplos desta categoria são GPSS (Gordon, 1978; Schriber, 1991), QNAP2 (Badel et al, 1991; Hill, 1993), SIMAN (Pegden et al, 1990), SIMSCRIPT (Law e Larmey, 1984), MODSIM (Law e Larmey, 1984), SLAM/TESS (Pritsker, 1986) e RESQ (Soares, 1990).

Esta generalidade tem como vantagem o fato de que uma vasta gama de sistemas podem ser estudados por meio do uso destes pacotes. A maior desvantagem é que eles são baseados num grau de formalismo difícil de se aprender rapidamente. Construir um modelo torna-se assim uma tarefa mais difícil. Por outro lado, estes pacotes gerais permitem a modelagem de sistemas de maior complexidade e possuem mais recursos do que um software dedicado.

3.5 CONSIDERAÇÕES FINAIS

O uso dos mecanismos de um software ou linguagem geral de simulação é uma possível maneira de se criar um modelo de animação de um sistema de software. Porém, para isso ser possível, o modelador teria que ter conhecimento dos conceitos de simulação e da linguagem para construir o seu modelo, o que não é uma solução prática e provavelmente inviável, dado os vários domínios de problemas e os vários perfis de conhecimento dos modeladores.

A solução proposta para este problema é que o modelador crie seu modelo de animação, sem se preocupar com os mecanismos de simulação necessários para animar o modelo.

O Capítulo 4 apresenta as características de um ambiente que suporta uma técnica para a criação dos modelos de animação de um sistema, utilizando mecanismos de animação e de simulação discreta orientada a eventos, propondo um simbolismo para a representação gráfica deste modelo. Este ambiente considera os mecanismos de simulação necessários para a animação do modelo gráfico criado, de forma transparente para o modelador, não implicando que ele tenha conhecimentos específicos das áreas de simulação e animação envolvidas.

CAPÍTULO 4

AMBIENTE PARA A CRIAÇÃO DE MODELOS DE ANIMAÇÃO DOS SISTEMAS BASEADOS EM SIMULAÇÃO

4.1 INTRODUÇÃO

O uso de notações visuais está em crescimento constante em projetos de sistemas. Elas oferecem mais flexibilidade aos projetistas e dão maior poder de expressividade aos projetos resultantes, tornando mais fácil representar graficamente descrições textuais.

Em Engenharia de Software, as técnicas existentes procuram fazer um uso intenso de diagramas para a modelagem de sistemas. Descrições textuais podem ser utilizadas para complementar a especificação do modelo, mas elas sozinhas podem ser ambíguas e subjetivas.

Baseado neste fato, e considerando as necessidades de representação da dinâmica e do comportamento dos sistemas levantadas no Capítulo 2, e ainda a atual carência de técnicas que adequada e completamente representem estas necessidades (Alves, 1998), este Capítulo apresenta uma técnica para retratar melhor a dinâmica e o comportamento, por meio da criação de modelos de animação dos sistemas. Estes modelos são baseados em simulação e consideram as entidades componentes do sistema, num alto nível de abstração.

À medida que mais detalhes são adicionados aos modelos de animação durante o processo de modelagem e uma maior compreensão do sistema é obtida, deseja-se, de forma gradual, chegar a especificação das entidades de software e de hardware do sistema.

Com sucessivos refinamentos das entidades de software, deseja-se, de forma gradual, chegar mais próximo da especificação dos objetos de software do sistema a ser implementado.

Pode-se dizer que toda especificação é uma implementação de algum outro nível mais alto da especificação. Isto quer dizer que, especificações e implementações intercalam-se numa hierarquia de níveis de abstração (Ozcam e Siddiqi, 1995).

Uma vez atingida uma etapa de desenvolvimento onde já se tem uma especificação em alto nível do software do sistema , todo o potencial já disponível nas técnicas e nas metodologias orientadas a objeto e nas ferramentas que as suportam poderá ser utilizado com intuito de trabalhar melhor os modelos nas fases de análise, projeto e implementação. Com isso, pretende-se dar um tratamento uniforme a todo o processo de desenvolvimento do sistema, dando continuidade aos modelos criados em fases anteriores e abrindo a possibilidade de uma maior interação.

A exploração da animação, o uso de mecanismos de simulação, o uso de cores e a possibilidade de criar vários níveis de decomposição destes modelos são recursos que serão utilizados para que um resultado efetivo na visualização dos aspectos dinâmicos possa ser gerado.

Foi feita a concepção de um ambiente para dar suporte à técnica de criação dos modelos de animação, de tal forma que seu uso seja fácil e não limite a criatividade do analista, e nem do especialista no domínio do problema, durante o processo inicial de modelagem. As ferramentas disponíveis para a criação do modelo são de fácil utilização, não exigindo um longo tempo de aprendizagem para o seu emprego.

A proposta do ambiente e suas ferramentas de modelagem é estabelecer um compromisso com a clareza através da simplicidade. Parece ser uma tendência natural procurar um conjunto mínimo de conceitos por um lado, e um alto grau de visualização do outro. O primeiro, ajuda no raciocínio sobre a essência dos problemas complexos, enquanto o segundo ajuda na percepção do sistema como um todo. O sistema pode ser muito complexo, mas o modelo a ser criado deve ser simples. Mesmo não podendo

capturar toda a complexidade dos sistemas reais, tais modelos vão oferecer informações bastante úteis para o modelador.

A técnica proposta adota uma notação gráfica e textual simples e independente de qualquer linguagem de programação. Estas características da notação devem torná-la facilmente adaptável a outros métodos de análise e projeto, principalmente aqueles orientados a objetos.

O enfoque dado à simulação para a criação destes modelos durante a especificação de um sistema é explicado na Seção seguinte.

4.2 SIMULAÇÃO E ESPECIFICAÇÃO DE REQUISITOS DOS SISTEMAS

Os modelos de simulação, conforme já discutidos no Capítulo 3, são caracterizados por mais rigidez na modelagem, tendo em vista que o sistema a ser simulado, na maioria das vezes, possui similar no mundo real (Shannon, 1975).

A forma de uso e os objetivos da simulação como ferramenta de modelagem para a criação dos modelos de animação, a partir de especificações de requisitos, é bem diferente da forma como é utilizada tradicionalmente. Estes novos sistemas não possuem similares no mundo real com os quais possam ser comparados. A sua modelagem e criação envolve informações que, na maioria das vezes, não estão disponíveis. Esta característica é fundamental para diferenciar o uso tradicional da simulação do uso pretendido para este trabalho.

Um outro aspecto da criação de um modelo de simulação é a determinação dos objetivos do estudo que se pretende realizar. É só após considerar estes objetivos, as características intrínsecas do sistema e a qualidade dos resultados pretendidos, que o modelador pode decidir o nível de precisão e de detalhes exigidos para a condução do estudo de simulação. Quando um sistema novo começa a ser modelado, normalmente os objetivos, as características intrínsecas do sistema e a qualidade dos resultados ainda não podem ser determinados com clareza.

As informações necessárias para a construção do sistema vão sendo adquiridas conforme uma melhor compreensão do seu funcionamento é obtida, por meio de sua modelagem. A simulação contribui neste processo com o suporte necessário para a

animação dos modelos gráficos construídos, permitindo que o modelador interaja durante a animação e faça alterações. Estas alterações não invalidam o modelo, como pode ocorrer com um modelo de simulação tradicional.

Construir um modelo de animação utilizando simulação, requer que os modeladores adicionem informação relacionada à simulação às especificações existentes. Isto inclui um tempo de computação aproximado para um processo, uma descrição rudimentar da própria computação a ser realizada, e o comportamento estocástico das entradas externas, como taxa de chegada de dados e pontos de parada, com o intuito de experimentar com o sistema durante a animação.

Um modelo de animação deve ser construído no topo das especificações, que evoluem em detalhes com o sistema, com o objetivo de simular a dinâmica envolvida nestas especificações. A especificação de um sistema freqüentemente é feita de uma maneira mais detalhada em alguns aspectos, trabalhando-se em uma parte do sistema mais conhecida, e depois trabalhando-se em outras.

Assim, a evolução da especificação é caracterizada por níveis de refinamento variados em diferentes partes do sistema. Logo, uma especificação evolui com graus variados de detalhes, de especificações abstratas até a implementação. Isto pode ser chamado de simulação durante a especificação e ela deve suportar muito bem as práticas de desenvolvimento de software incremental (Boehm, 1988).

Tem havido vários esforços para implementar a simulação de especificações para validação dos requisitos dos usuários. A seguir são colocados alguns exemplos.

A linguagem Gist (Balzer et al, 1983) foi desenvolvida para validar especificações por meio da execução da especificação. Contudo, a linguagem Gist não fornece uma animação do modelo executado e não é adequada para expressar quando os eventos ocorrem ao longo do tempo.

A Especificação Gráfica Executável (EGS - Executable Graphical Specification) (Gaskell e Phillips, 1994) é um ambiente CASE que usa a notação de DFD de DeMarco (1978) para descrever os modelos funcionais. Nele, o usuário melhora o modelo com elementos funcionais da linguagem código. Isto habilita os modelos a serem animados

de uma maneira interpretativa, permitindo que a funcionalidade do sistema seja observada. Contudo, além do EGS ser projetado para sistemas orientados a dados e executar os modelos somente de uma maneira interativa, a animação de DFDs não é suficiente para entender o comportamento dinâmico de um sistema que envolvem, além de dados e funções, também os eventos.

Outro sistema, o SCHEMASIM (Coomber, 1994), foi desenvolvido para modelar sistemas de tempo real e simular seu comportamento. O SCHEMASIM também usa um diagrama de fluxo de dados, só que estendido para propósitos de especificação de sistemas de tempo real (Ward, 1986; Bruyn, 1987). Ele pode descrever componentes funcionais de um sistema em termos de fluxo de dados e fluxo de controle, mecanismos de controle e restrições de tempo requerido. Ele necessita de informações bastante detalhadas, como fluxo de controle, para a criação do modelo e estas podem não estar disponíveis nas especificações iniciais existentes dos sistemas.

A PAISLey (Zave, 1991) é outra linguagem que usa processos cíclicos assíncronos e programação funcional para especificação de sistemas. Embora ela tenha uma capacidade de simulação das especificações para prever o desempenho do sistema, não possui suporte para simulação interativa e nem um ambiente de modelagem gráfico.

O Ambiente de Execução de Esquemas (*Schema Execution Environment* – SEE) (Coomber, 1997) é um simulador gráfico para um modelo de especificação baseado em objetos, integrado com o Esquema de Transformação (*Transformation Schema)* (Ward, 1986). O Esquema de Transformação estende o diagrama de fluxo de dados e torna possível expressar o comportamento de tempo real, usando diagramas de transição de estados (Harel, 1988). O SEE é adequado para executar um sistema de acordo com um cenário e verificar se as transformações necessárias são ativadas e as saídas desejáveis são geradas. Por outro lado, ele não suporta a simulação muito bem, porque o seu simulador incrementa o tempo em intervalos regulares, não sendo possível especificar entradas estocásticas para o sistema.

O STATEMATE (Harel at al, 1990; Harel e Politi, 1996) é também um ambiente gráfico para especificação, análise, projeto e documentação de sistemas. Utilizando o princípio de decomposição funcional, ele quebra um sistema complexo em partes

hierárquicas mais simples, descrevendo-o em termos de funções de alto nível e suas interfaces. Estas funções de alto nível são quebradas em subfunções mais específicas. Conforme a necessidade de modelagem das funções do sistema, são criados diagramas de transição de estado associados a estas funções, que podem ser animados por meio da simulação. A dinâmica do sistema como um todo não é representada.

O sistema ASADAL/SIM (Kang et al, 1998) permite a construção de um modelo de simulação onde as especificações referem-se a diagramas de seqüência de mensagens, *statecharts* enriquecidos com informações de tempo, diagramas de transição de estado melhorados com requisitos de tempo e diagramas de fluxo de dados, hierarquicamente decompostos em diagramas de processo. Resultados da simulação e da análise são apresentados aos usuários por meio de gráficos, diagramas de fluxo de dados animados e *statecharts* animados.

Nos sistemas citados acima, a grande potencialidade da orientação a objetos na área de simulação e especificação de sistemas, aliada a técnicas de representação gráfica do elementos do modelo, considerando a dinâmica do sistema como um todo, ainda não é bem explorada.

A animação dos modelos dinâmicos dos sistemas, considerando o uso da simulação e da orientação a objetos, pode tornar possível a criação de uma especificação inicial, que vai sendo refinada gradualmente conforme mais informações forem sendo incorporadas ao modelo.

As seções seguintes apresentam as características principais de um ambiente de modelagem para dar suporte à técnica de criação dos modelos de animação, definindo os termos básicos empregados e o simbolismo adotado para o modelo gráfico que será animado.

4.3 DEFINIÇÕES E TERMOS EMPREGADOS

A seguir são colocadas algumas definições e termos que deverão ser conhecidos pelo modelador durante a modelagem. Estes serão os elementos básicos utilizados pela técnica de definição e criação do modelo de animação do sistema dentro do ambiente de modelagem.

- **Entidade Básica** é uma abstração básica que não pode ser decomposta em outras entidades;
- **Entidade Composta** é um conjunto de entidades que podem estar organizadas hierarquicamente ou agrupadas.

As entidades formarão a estrutura inicial do modelo sobre a qual a dinâmica se desenvolverá. Elas deverão manusear as responsabilidades do sistema, num alto nível de abstração.

Sensores analógicos, fornecedores, clientes, pedidos de clientes são exemplos de entidades básicas. Exemplos de entidades compostas incluem sistemas de gerência de dados, sistemas de cadastro, computadores, janelas gráficas para visualização de dados, etc.

Além da classificação das entidades em entidades básicas e entidades compostas, elas são também classificadas em entidades **estáticas** ou **dinâmicas**.

- **Entidades Estáticas** são aquelas que permanecem fixas numa determinada posição dentro da estrutura do modelo a ser animado, durante todo tempo considerado na modelagem. Elas formam a infra-estrutura do modelo.
- **Entidades Dinâmicas** movem-se fisicamente de um ponto a outro, compondo a parte não estática dos modelos de animação.

Exemplos de entidades estáticas incluem computadores, unidades de gerência de dados, unidades de processamento de informação e sensores. Os fluxos de informações ou dados normalmente representam as entidades dinâmicas do modelo de animação. Exemplos incluem mensagens enviadas entre sistemas distintos e dados vindos de entidades externas ao sistema, tais como, pedidos de clientes, dados de sensores, telemetria de satélite, telecomandos enviados para o satélite e ondas de rádio.

Mensagens trocadas entre as entidades componentes do modelo também podem ser modeladas como entidades dinâmicas. Uma mensagem pode ser definida como uma unidade atômica de comunicação entre entidades do modelo utilizadas para o envio de

alguma informação. Exemplos de mensagens incluem envio de telecomandos para satélites e mensagens enviadas através de uma rede de dados.

- **Filas** são locais de espera onde as entidades podem estar aguardando o uso de um determinado recurso, algum tipo de processamento ou a chegada de um evento.

As filas podem estar diretamente ligadas a uma entidade, de forma explícita ou implícita. O uso explícito pressupõe a necessidade de representar um local externo à entidade, que servirá como um tipo de depósito, onde outras entidades ficarão armazenadas por um tempo. A fila implícita pode ocorrer em entidades que representam recursos e que resolvem o problema da concorrência utilizando políticas internas próprias. Neste caso, considera-se que a responsabilidade do tratamento da concorrência passa a ser da entidade possuidora da fila, e são detalhes não representados no nível de especificação em questão.

- **Evento** é definido como algo que acontece no sistema e que está sendo considerado como atômico e instantâneo em relação à escala de tempo utilizada na modelagem. Um evento pode causar uma mudança de estado, disparar outros eventos, ou provocar o início e o término de uma atividade.

Alguns exemplos de eventos incluem a partida de um vôo, o pressionar de um botão do mouse, a ligação de um equipamento eletrônico, o envio de um sinal de alarme, etc.

4.4 SIMBOLISMO ADOTADO PARA A REPRESENTAÇÃO DOS ELEMENTOS DO MODELO

A seguir são apresentados os elementos de um simbolismo padrão básico, sugerido para ser adotado para a representação gráfica do modelo, dentro do ambiente proposto.

Entidade Estática Básica/Composta.

Ao símbolo utilizado para a entidade estática será atribuído a cor transparente. Esta cor será mudada para a cor especificada para cada estado que a entidade passar durante a animação do modelo. O ambiente atribui as cores iniciais, que podem ser posteriormente alteradas pelo modelador.

Entidade Dinâmica Básica/Composta.

Analogamente à entidade estática, a entidade dinâmica também terá a cor preta atribuída inicialmente pelo ambiente. O modelador poderá alterar esta cor quando desejar.

Entidade Dinâmica Básica/Composta com o sentido do movimento, quando tomado um instantâneo do sistema.

Entidade Dinâmica Básica/Composta com mais de um sentido de movimento, quando tomado um instantâneo do sistema.

Ser Humano

Ao elemento Ser Humano será atribuída inicialmente a cor transparente que poderá ser alterada pelo modelador. O elemento Ser Humano pode ser visto tanto como ma entidade estática ou como uma entidade dinâmica, mas devido à sua importância na modelagem, ele foi considerado separadamente.

Fila Vazia

Indica que não há nenhuma entidade dinâmica na fila.

Fila Cheia

Indica que há uma ou mais entidades dinâmicas na fila.

Outros elementos gráficos, como retângulos, setas, elipses, textos e linhas podem ser utilizados para complementar e enriquecer o modelo gráfico criado, mas eles não fazem parte do simbolismo padrão adotado e não serão animados.

Os eventos, apesar de não serem representados graficamente, são elementos que fazem parte do processo de criação do modelo de animação, como será descrito na Seção 4.5.

Variáveis globais, utilizadas para propósitos da animação do modelo, também podem ser definidas pelo modelador através de janelas gráficas do ambiente.

As transformações que uma entidade estática realiza são representadas por meio dos possíveis estados definidos para a entidade.

Além disso, podem existir entidades em sincronia e também pré e pós-condições associadas a elas. Para facilitar a identificação destes elementos do modelo, é sugerido o uso de rótulos, conforme mostra a Figura 4.1.

Para indicar a sincronia, o rótulo deve ser seguido de um número. O número é necessário para se relacionar conjuntos de entidades que estão em sincronia, caso haja mais de um conjunto no modelo, como pode ser visto na Figura 4.2. Neste caso, também fica a cargo do modelador decidir o uso ou não do rótulo.

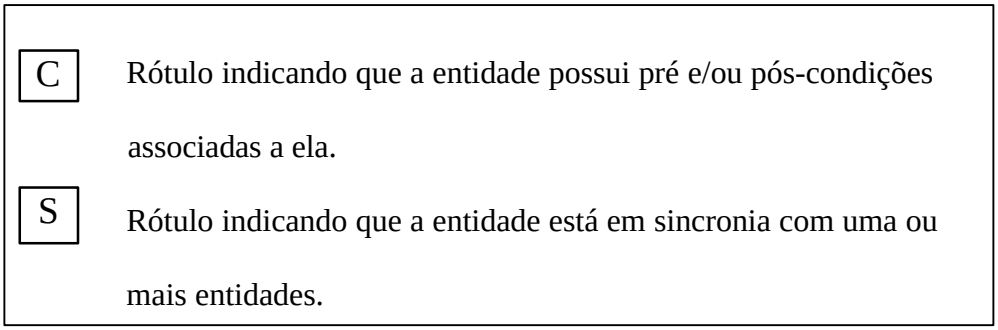

Fig. 4.1- Rótulos utilizados nas entidades.

Alguns exemplos da representação das entidades com o uso destes rótulos, são apresentados na Figura 4.2.

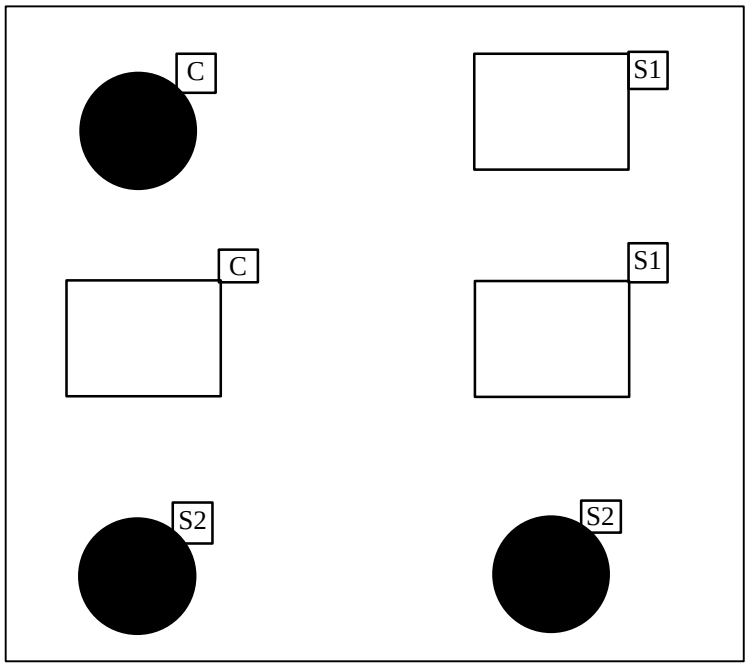

Fig. 4.2 - Exemplos de entidades estáticas e dinâmicas rotuladas. S1 e S2 definem dois conjuntos distintos de entidades que estão em sincronia.

4.5 APRESENTAÇÃO GRÁFICA DOS ELEMENTOS

A apresentação gráfica das entidades no modelo poderá ser alterada pelo modelador, substituindo-a por outras figuras que melhor as representem, de acordo com o tipo do sistema sendo modelado. Isto poderá ser feito dentro do ambiente proposto, desde que a consistência seja mantida.

Foram usadas as cores preto e branco para representar os estados ativo e desocupado, respectivamente, para efeitos de ilustração. Porém as entidades podem ter outros estados, diferentes destes, que necessitem ser modelados e visualizados. Neste caso, outras cores poderão ser utilizadas para modelar estes estados. A atribuição das cores deverá ser feita automaticamente pelo ambiente, contudo será possível, o próprio modelador fazer a escolha das cores.

Se o modelador decidir usar somente as cores preta e branca na sua representação do modelo, todas as entidades estáticas do modelo serão representadas por retângulos pretos e brancos e todas as entidades dinâmicas serão representadas por círculos pretos e brancos, prejudicando a distinção gráfica entre elas, principalmente quando se toma um instantâneo do sistema.

Neste caso, os nomes dados às entidades estáticas permitirão fazer a diferenciação. Às entidades dinâmicas, serão atribuídos números distintos. A notação numérica também ajuda na impressão não colorida em papel de um instantâneo do modelo. Durante a animação do modelo, estes números perderão o poder de expressar a distinção e é aconselhável que sempre se utilize cores diferentes.

Os atributos de uma entidade, descritos na Seção 4.6, serão definidos no próprio ambiente, com a utilização de janelas gráficas apropriadas. Estas informações e as relacionadas com o sincronismo e o uso de pré e pós-condições poderão ser visualizadas a qualquer momento, bastando para isso selecionar com o mouse a representação gráfica do objeto em questão, conforme mostra a Figura 4.3.

Esta notação enriquece a representação de um instantâneo do modelo, diferenciando de uma maneira gráfica bastante simples os elementos participantes. Caberá ao modelador decidir se usará ou não os rótulos identificadores.

Caso se queira saber mais informações a respeito do objeto gráfico apontado, bastará clicar duas vezes com o mouse sobre ele e informações mais detalhadas sobre estados de uma entidade, incluindo a existência de pré e pós-condições e o tempo associado à permanência em cada estado, poderão ser definidas e visualizadas no ambiente por meio de janelas gráficas.

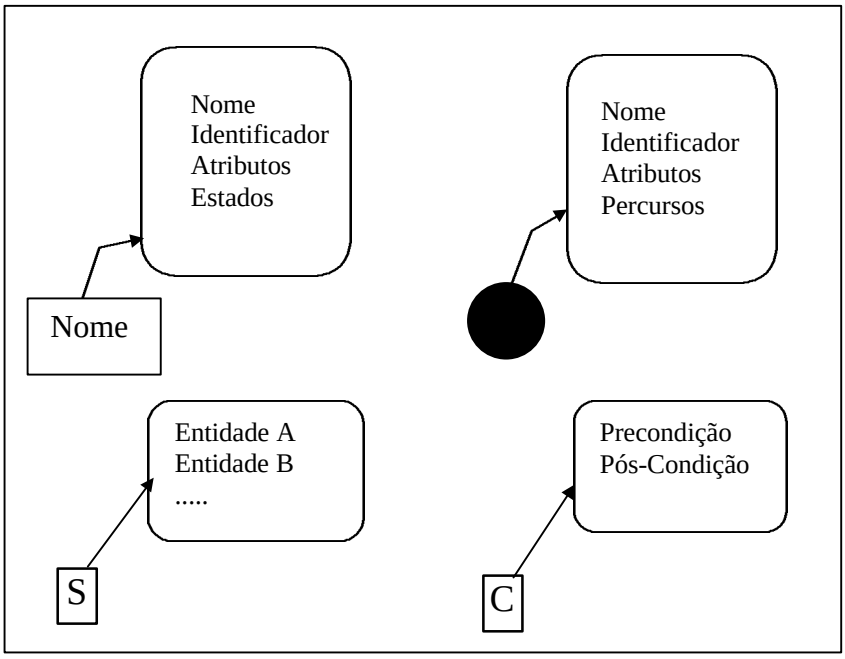

Fig. 4.3 - Apresentação das informações referentes aos elementos gráficos utilizados no modelo.

A seguir, são colocados dois exemplos simples para ilustrar o emprego da técnica e do simbolismo apresentado.

Exemplo 1: Um Satélite envia dados de telemetria para uma Estação Terrena. Estes dados movem-se do satélite para a estação terrena. O satélite possui uma posição e a antena da estação deve estar direcionada para a posição em que se encontra o satélite para recebimento dos dados durante sua passagem, estabelecendo-se assim um sincronismo entre os dois. O Satélite e a Estação Terrena podem ser vistos como entidades estáticas do modelo e os dados de telemetria compõem uma entidade dinâmica. A Figura 4.4 ilustra esta situação, mostrando um instantâneo do sistema, onde as entidades estáticas estão inativas e a entidade dinâmica está ativa.

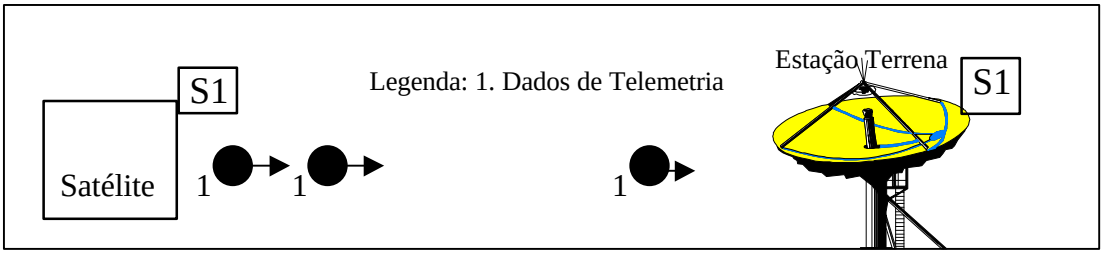

Fig. 4.4 - Satélite envia dados de telemetria para a estação terrena.

Exemplo 2: Suponhamos que um usuário deseja fazer um tratamento numa determinada imagem digital bruta, utilizando para isso um algoritmo específico, conforme mostra a Figura 4.5. Ele solicita o tratamento utilizando uma Interface Gráfica. O algoritmo de tratamento é encapsulado numa entidade chamada **Imagem**. A **Interface** e a **Imagem** podem ser vistas como entidades estáticas do modelo.

A imagem bruta e a imagem tratada podem ser vistas como entidades dinâmicas do modelo. A **Interface** envia a imagem bruta para a entidade **Imagem.** A **Imagem** cria uma nova entidade dinâmica chamada **Imagem Tratada,** a partir de alguma transformação realizada sobre a imagem bruta, e envia esta nova entidade dinâmica para a entidade **Interface** exibi-la. As entidades estáticas e o ser humano aparecem neste instantâneo como desocupados.

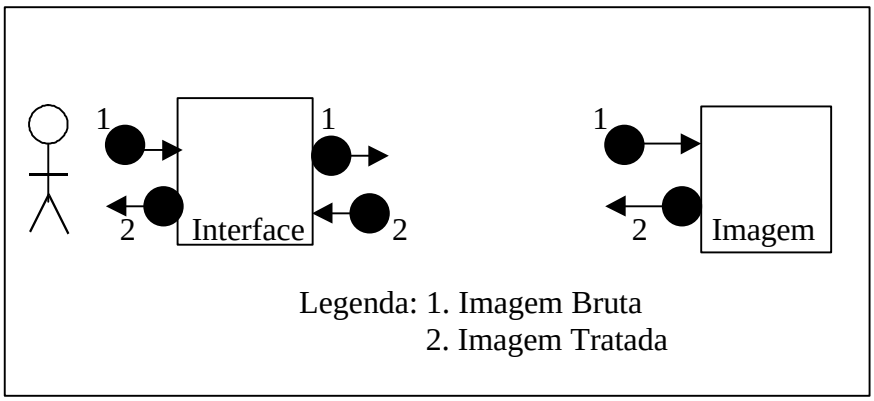

Fig. 4.5- Usuário solicita tratamento de uma imagem utilizando as entidades estáticas Interface e Imagem.

4.6 CARACTERIZAÇÃO DOS ELEMENTOS DO SIMBOLISMO

Uma vez apresentados os elementos do simbolismo utilizado pela técnica, resta agora caracterizá-los, apresentando a estrutura básica de formação de seus elementos. É com a informação contida nesta estrutura, com a informação gráfica do modelo criado e os mecanismos de simulação, que o ambiente comporá o modelo de animação.

Esta Seção apresenta os atributos da estrutura dos elementos que o modelador deve ter conhecimento, porque muitos valores destes atributos serão fornecidos por ele, durante o processo de criação do modelo de animação. Maiores detalhes da estrutura destes elementos serão apresentados no Capítulo 5.

As entidades estáticas e dinâmicas possuem alguns atributos comuns que as caracterizam e alguns que são próprios de cada uma. A Tabela 4.1 apresenta estes atributos comuns e suas descrições. As Tabelas 4.2 e 4.3 apresentam seus atributos próprios.

Todas as entidades estáticas são criadas no início da animação do modelo e mostradas na tela. Caso o usuário queira visualizar estaticamente todas entidades dinâmicas definidas, ele poderá fazê-lo a qualquer momento, selecionando uma opção que estará disponível em um dos menus na janela de interface gráfica do ambiente.

As entidades dinâmicas podem ser geradas por alguma entidade estática ou geradas a partir de um evento de geração. A geração pode ser periódica ou aleatória, a partir de alguma função de distribuição.

A entidade estática, como já mencionado anteriormente, possui uma lista dos possíveis estados que ela passa durante a animação do modelo. Estes estados serão representados por cores distintas, diferenciando-os visualmente. Durante a animação, a entidade que está em um estado diferente de desocupado, terá a cor correspondente ao estado durante o tempo em que ela permanecer nele.

Aos estados podem estar associadas pré e pós-condições, representadas como seus atributos (Tabela 4.4). No estabelecimento das pré e pós-condições poderão ser utilizadas variáveis e sentenças lógicas.

As pré e pós-condições podem também ser utilizadas para o estabelecimento de sincronismo. Uma situação deste tipo ocorre quando duas entidades estáticas distintas devem estar num determinado estado ao mesmo tempo, para a execução de alguma transformação conjunta (sincronismo entre transformações). Elas devem possuir a mesma precondição que, por exemplo, atribua o valor TRUE a uma variável INICIAR_TRANS, permitindo a entrada neste determinado estado.

TABELA 4.1 - ATRIBUTOS COMUNS DAS ENTIDADES ESTÁTICAS E DINÂMICAS

Atributos	**Descrição**
Nome	Nome lógico que deve ser fornecido pelo modelador para a entidade.
ID	Número inteiro que é dado automaticamente pelo sistema ou fornecido pelo modelador, identificando unicamente uma entidade.
Cor	Cor dada à entidade para efeitos de visualização. Inicialmente, toda entidade estática recebe a cor transparente e a entidade dinâmica a cor preto.
Figura	Contém o nome de uma figura associada a representação da entidade. De acordo com o simbolismo adotado, a Figura padrão é o retângulo para a entidade estática e o círculo para a dinâmica. Se o modelador desejar utilizar outra figura, ele deve fornecer o nome do arquivo que contém a figura (bmp, jpg, gif, dentre outros).
Status	Atributo que recebe o valor habilitado ou desabilitado. Serve para inibir o funcionamento da entidade durante a execução da animação, para testar hipóteses e verificar como o modelo reage. Isto possibilita, por exemplo, simular uma falha temporária de um equipamento, uma falha de comunicação, uma parada para manutenção, etc. Para isto ser possível, é necessário que o ambiente suporte uma simulação interativa, conforme mencionada na Seção 4.9.
Lista de Atributos:	Lista dos atributos da entidade que a definirá como uma abstração da realidade. Estes atributos não influenciam a animação e simulação do modelo; são atributos que vão sendo definidos pelo modelador, conforme ele obtém uma maior compreensão do modelo e identifica o que deve e não deve ser considerado de cada entidade. Serão utilizados posteriormente para a definição das entidades de software do sistema.
Lista de Métodos	Analogamente à lista de entidades, a lista dos métodos conterá os principais métodos ou funções que devem ser realizadas pela entidade, mas não fazem parte da animação e nem da simulação. Estes métodos vão sendo identificados à medida que o papel da entidade no sistema é melhor compreendido.

TABELA 4.2 - ATRIBUTOS PRÓPRIOS DA ENTIDADE ESTÁTICA

Atributo	**Descrição**
Estado Inicial	Contém o nome do estado inicial da entidade no início da animação Como padrão, é atribuído o estado **desocupado.**
Lista de Estados	Lista contendo todos os estados possíveis que uma entidade estática possa estar durante a animação do modelo. Os estados de uma entidade, de uma maneira indireta, representam as possíveis transformações que uma entidade realiza durante a animação.
Lista de Entidades Estáticas	Lista que contém o nome das outras entidades estáticas que a compõem, caso a entidade seja composta. Esta lista é utilizada para a criação dos vários níveis de decomposição do modelo.

TABELA 4.3 - ATRIBUTOS PRÓPRIOS DA ENTIDADE DINÂMICA

Atributo	**Descrição**
Prioridade	Estabelece um número de prioridade. Prioridades serão utilizadas para a ordenação das entidades dinâmicas em filas de acesso a uma entidade estática. As filas deverão possuir sua política de ordenação, ou serem ordenadas de acordo com o numero de prioridade definido para as entidades que estarão na fila.
Lista de Percursos	Lista contendo todos os percursos da entidade dinâmica, constando suas possíveis origens e possíveis destinos, e o tempo associado ao percurso. A uma origem da entidade dinâmica podem estar associadas mais de um destino.
Lista de Entidades Dinâmicas	Lista que contém o nome das outras entidades dinâmicas que a compõem, caso a entidade tenha sido decomposta. Esta lista é utilizada para a criação dos vários níveis de decomposição do modelo.

Outro caso possível do uso de pré e pós-condições é a mudança de estado de uma entidade condicionada à mudança de estado de uma outra entidade. Uma situação deste

tipo pode ocorrer quando uma entidade estática A realiza a transição para o estado E1 se o evento de transição T ocorrer, e tem como precondição que a entidade estática B esteja no estado E2. Neste caso, pode ser utilizada uma variável C que recebe o valor *TRUE* como pós-condição da transição de estado da entidade B para o estado E2. Esta mesma variável com o valor *TRUE* é colocada como precondição para a transição da entidade A para o estado E1.

Ainda relacionado ao uso de pré e pós-condições, outro possível caso diz respeito ao acesso de uma entidade dinâmica a uma entidade estática, quando esta possui uma fila ligada a ela. A pós-condição (condição de saída da fila) determina quando uma entidade na fila pode ter acesso à entidade estática. Por exemplo, se a entidade estática representar algum processamento feito sobre uma entidade na fila, ela só poderá receber uma nova entidade para processar quando estiver no estado desocupado. Outros casos que empregam pré e pós-condições podem ser tratados com variáveis, de maneira análoga.

A Tabela 4.4 seguinte apresenta os atributos e as respectivas descrições do estado de uma entidade estática.

Apesar do atributo Próximo Estado não estar diretamente relacionado com o estado atual, ele é uma informação necessária para que o ambiente possa criar o traçado da simulação e consequentemente, o modelo de animação, como será visto no Capítulo seguinte.

Com as informações dos estados fornecidas na modelagem de uma entidade estática, é possível construir seu Diagrama de Transição de Estados (DTE). Este diagrama poderá ser gerado automaticamente pelo ambiente, caso o modelador queira visualizá-lo. O diagrama também será útil no final, quando o modelo dinâmico já estiver definido e o próximo passo será derivar os principais objetos de software do sistema, conforme descrito na Seção 4.12.

Outro elemento proposto no simbolismo é a fila. Normalmente a fila é necessária diante de uma entidade estática quando esta estiver representando algum tipo de recurso ou processamento. Neste caso, as entidades dinâmicas ficam na fila até que a entidade

estática esteja no estado **desocupada,** tornando-se apta para receber uma nova entidade dinâmica.

TABELA 4.4 - ATRIBUTOS DO ESTADO

Atributos	**Descrição**
Nome	Nome lógico que deve ser fornecido pelo modelador para o estado
ID	Número inteiro que é dado automaticamente pelo sistema ou fornecido pelo modelador, identificando unicamente um estado.
Cor	Cor dada ao estado para efeitos de visualização. É atribuída automaticamente pelo ambiente, mas pode ser definida pelo modelador.
Tempo	Define o tempo em que a entidade permanece no estado.
Geração	Caso a entidade não fique no estado durante um tempo já pré-fixado, este atributo define uma função de distribuição para o tempo de permanência da entidade no estado.
Criação	A entidade estática pode criar entidades dinâmicas durante sua permanência no estado. Caso isto ocorra, este atributo informa o nome e a quantidade de entidades dinâmicas a serem criadas e o tempo necessário para o processo de criação. A quantidade pode ser um valor previamente fixado ou gerado aleatoriamente e o tempo para a criação pode ser fixo, aleatório ou periódico. Se for periódico, o número de vezes que o período ocorre também deve ser informado.
Precondição	Estabelece a condição necessária para que a entidade mude de estado.
Pós-condição	Estabelece a condição que deve ser satisfeita após a saída da entidade do estado.
Evento Transição	Evento que provoca a transição do estado atual para o próximo estado.
Próximo Estado	Estado que a entidade estará após o evento de transição.

Quando isto ocorre, a entidade dinâmica deixa a fila realizando um movimento até a entidade estática e um evento de transição é gerado, fazendo com que a entidade estática saia do estado desocupada. Estes eventos serão gerados automaticamente pelo ambiente, baseados na modelagem gráfica e mostrados para o usuário utilizando janelas gráficas, para que este os associe corretamente ao modelo.

A Tabela 4.5 apresenta os atributos e as respectivas descrições do elemento Fila.

Resta ainda definir os atributos dos eventos e das variáveis do modelo, que apesar de não possuírem representação gráfica, são elementos que fazem parte do modelo e são definidos pelo modelador. As variáveis serão utilizadas para o estabelecimento de pré e pós-condições. As Tabelas 4.6 e 4.7 apresentam os atributos e as descrições do elemento **evento** e do elemento **variável**, respectivamente.

A Seção seguinte mostra como o modelador interage com o ambiente e define os elementos acima colocados para a criação do modelo de animação, apresentando alguns protótipos de janelas gráficas e exemplificando o seu uso por meio de um exemplo.

TABELA 4.5 - ATRIBUTOS DA FILA

Atributos	**Descrição**
Nome	Nome lógico que deve ser fornecido pelo modelador para a fila.
ID	Número inteiro que é dado automaticamente pelo sistema ou fornecido pelo modelador, identificando unicamente uma fila.
Ordenação	Define a política de ordenação da fila, que pode ser FIFO (First In First Out), LIFO (Last In First Out) ou por prioridade, que é representada por um valor numérico. Caso a ordenação seja por prioridade, as entidades dinâmicas que chegam devem também possuir um número de prioridade definido para que a ordenação seja efetuada.
Nro Entidades Fila	Armazena o número de entidades dinâmicas que estão na fila num dado momento, esperando pelo uso de algum recurso ou aguardando a ocorrência de algum evento.

Prioridade	Número inteiro que estabelece uma prioridade de ordenação.
Evento de entrada	Evento responsável pela entrada da entidade na fila. Será gerado automaticamente pelo ambiente, baseado na modelagem gráfica.
Condição de Saída	Condição que deve ser satisfeita para que a entidade saia da fila. Normalmente esta condição estabelece que a entidade estática ao qual a fila está associada esteja no estado desocupada.
Evento de Saída	Evento que é definido pelo modelador e deve ocorrer assim que a entidade dinâmica sai da fila.

TABELA 4.6 - ATRIBUTOS DO EVENTO

Atributos	**Descrição**
Nome	Nome lógico que deve ser fornecido pelo modelador para o evento.
ID	Número inteiro que é dado automaticamente pelo sistema ou fornecido pelo modelador, identificando unicamente um evento.
Geração	Define a maneira que o evento vai ser gerado. Um evento pode ocorrer periodicamente ou ser gerado aleatoriamente de acordo com uma função de distribuição.

TABELA 4.7 - ATRIBUTOS DA VARIÁVEL

Atributos	**Descrição**
Nome	Nome lógico que deve ser fornecido pelo modelador para a variável.
ID	Número inteiro que é dado automaticamente pelo sistema ou fornecido pelo modelador, identificando unicamente uma variável.
Tipo	Define o tipo de variável (exemplos: inteira, booleana, real, string de caracteres, vetor, matriz).
Valor Inicial	Armazena o valor inicial dado para a variável.

4.7 INTERFACES COM O MODELADOR

O projeto das interfaces gráficas é uma atividade de extrema importância, em qualquer projeto de software que tenha como característica uma forte interação humana. Considerando-se esta importância, este trabalho apresenta nesta Seção alguns protótipos de interfaces com o usuário, com o propósito de exemplificar o emprego da técnica, a utilização do ambiente e as principais ferramentas de modelagem.

As interfaces gráficas devem ser projetadas de tal forma que um usuário não familiarizado com os termos técnicos, consiga utilizar as ferramentas do ambiente sem maiores dificuldades.

Como requisitos mínimos, as interfaces gráficas, além de possuírem menus adequados ao ambiente, devem também considerar:

- Um simbolismo padrão colocado numa barra de botões de símbolos;
- A possibilidade de criação de uma barra de botões de símbolos pelo modelador, adequada ao seu problema; e
- A possibilidade de criação pelo modelador de uma paleta de cores própria, associada às entidades dinâmicas e aos possíveis estados das entidades estáticas.

A seguir, utilizando um exemplo simples, a técnica de criação dos modelos de animação é ilustrada e também são mostradas as facilidades que o ambiente oferece para a esta atividade, e quais informações o modelador deve fornecer no momento da modelagem.

O exemplo 2 da Seção anterior, envolvendo o Satélite e a Estação Terrena, Figura 4.5, será novamente utilizado. Quando o ambiente é inicializado, uma janela de entrada é mostrada, conforme ilustrado na Figura 4.6.

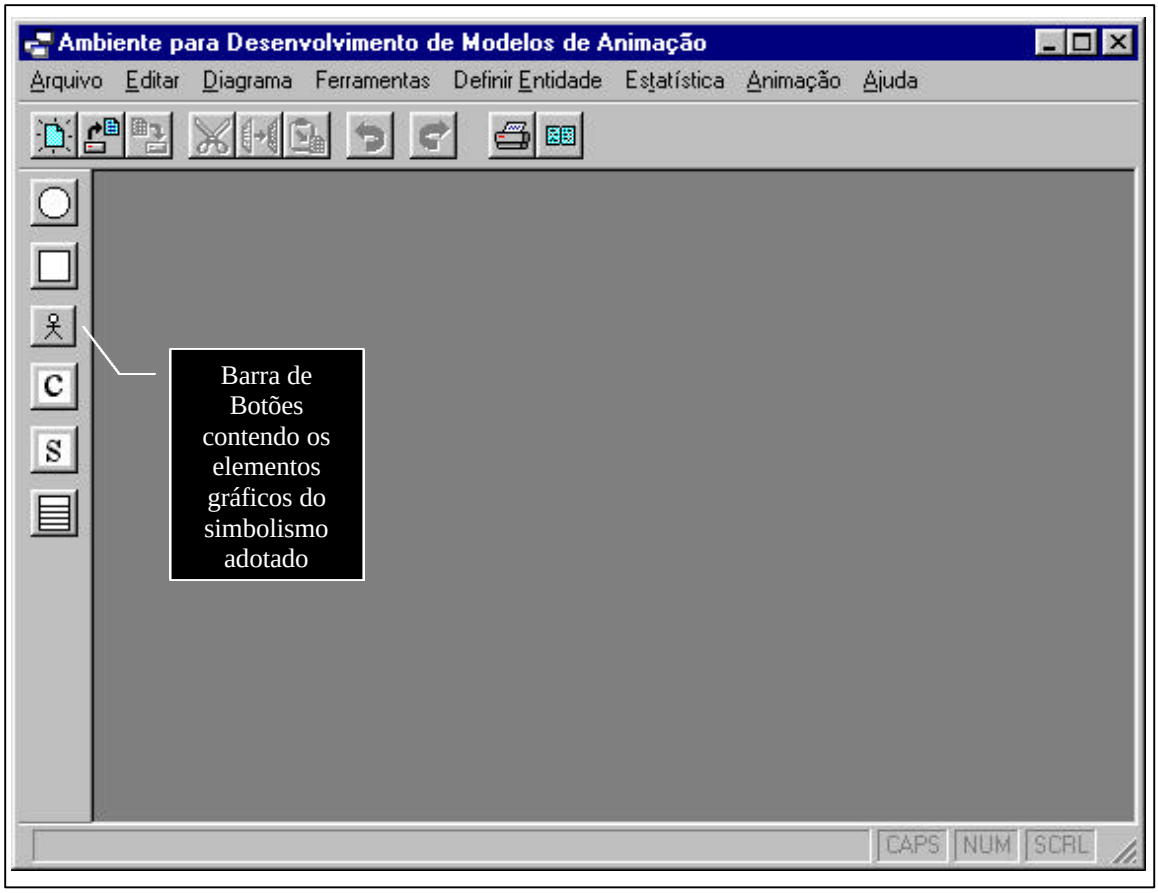

Fig. 4.6 - Protótipo da janela de entrada do ambiente.

O modelador inicia a construção do modelo, selecionando na barra de botões de símbolos, situada ao lado esquerdo da tela, o botão representativo da entidade estática. Uma vez selecionado o botão com o mouse, ele o arrasta, posicionando na área de trabalho da janela e, com o pressionar do botão esquerdo, uma janela de diálogo é mostrada para que o modelador forneça informações sobre a entidade estática a ser criada. A Figura 4.7 ilustra esta janela de diálogo.

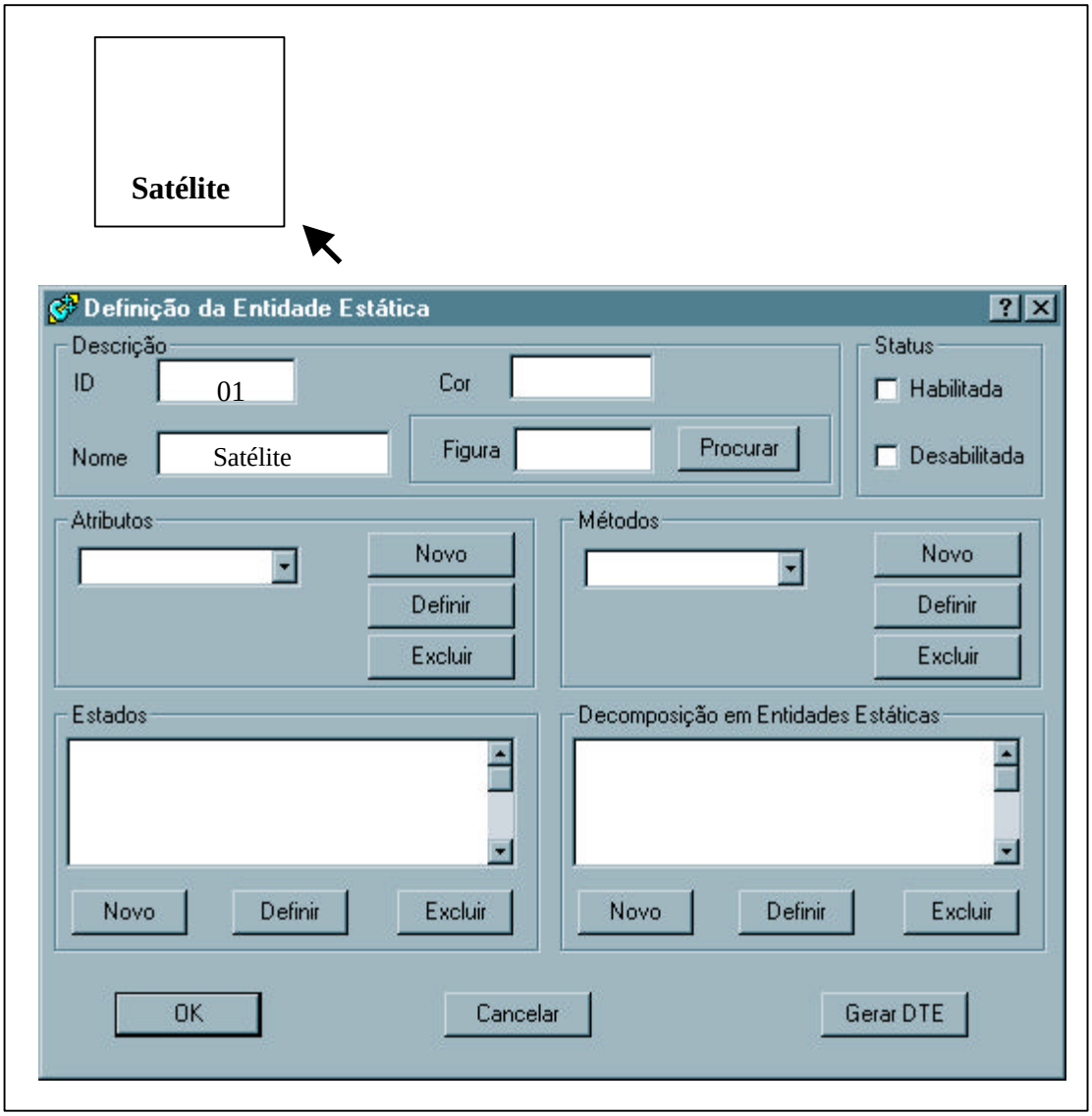

Fig. 4.7 - Definição da entidade estática Satélite.

De maneira análoga, é criada a entidade estática Estação Terrena. O próximo passo é a definição da entidade dinâmica Dados de Telemetria, utilizando o mesmo procedimento anterior. A Figura 4.8 ilustra a janela de diálogo apresentada.

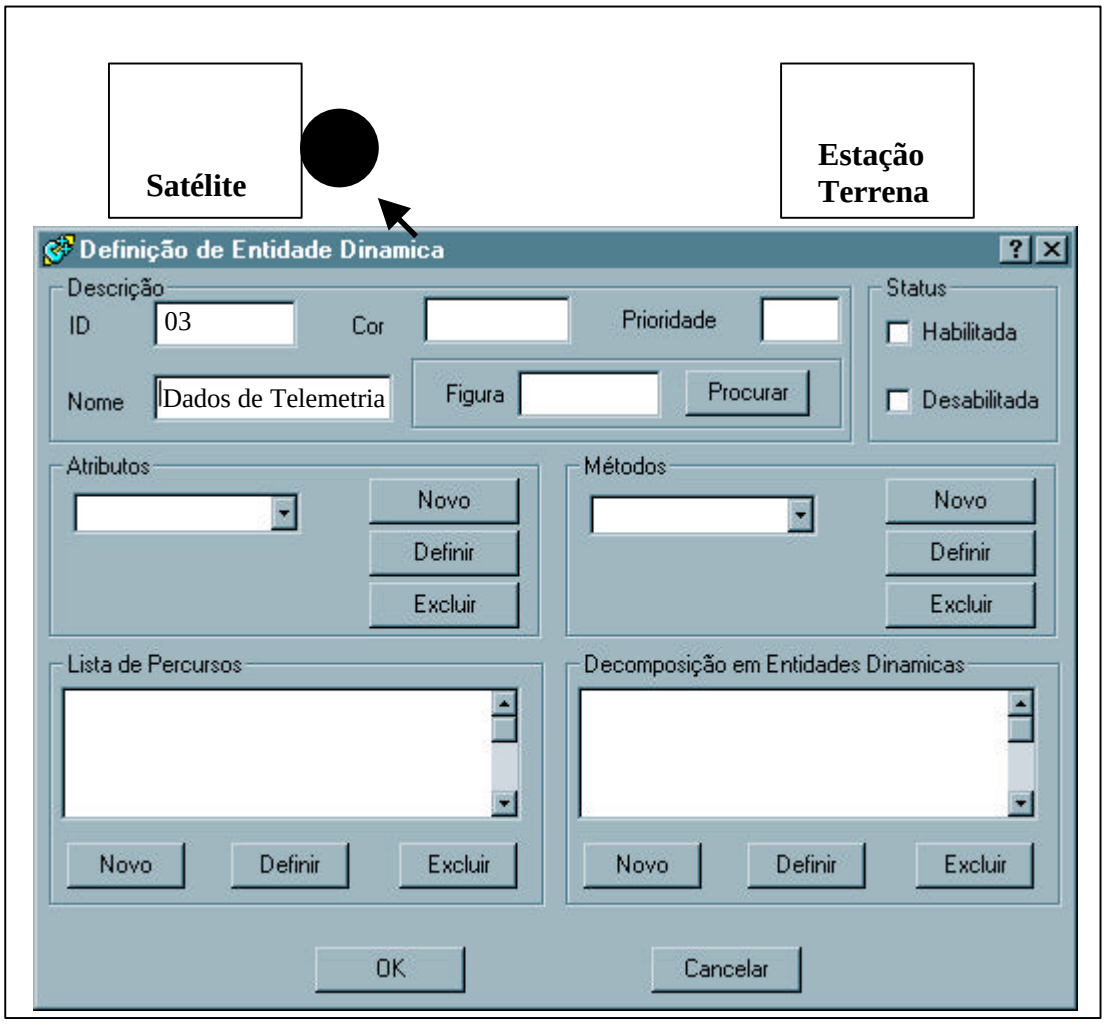

Fig. 4.8 - Definição da entidade dinâmica Dados de Telemetria.

A entidade dinâmica Dados de Telemetria move-se até a entidade estática Estação Terrena. O modelador pode definir este percurso usando a janela de definição da entidade Dados de Telemetria, no campo Lista de Percursos, ou graficamente representar este percurso, indicando a entidade ou fila de destino, conforme mostra a Figura 4.9. Quando a representação gráfica for usada, os dados de percurso são preenchidos automaticamente pelo ambiente, restando ao modelador definir o tempo deste percurso.

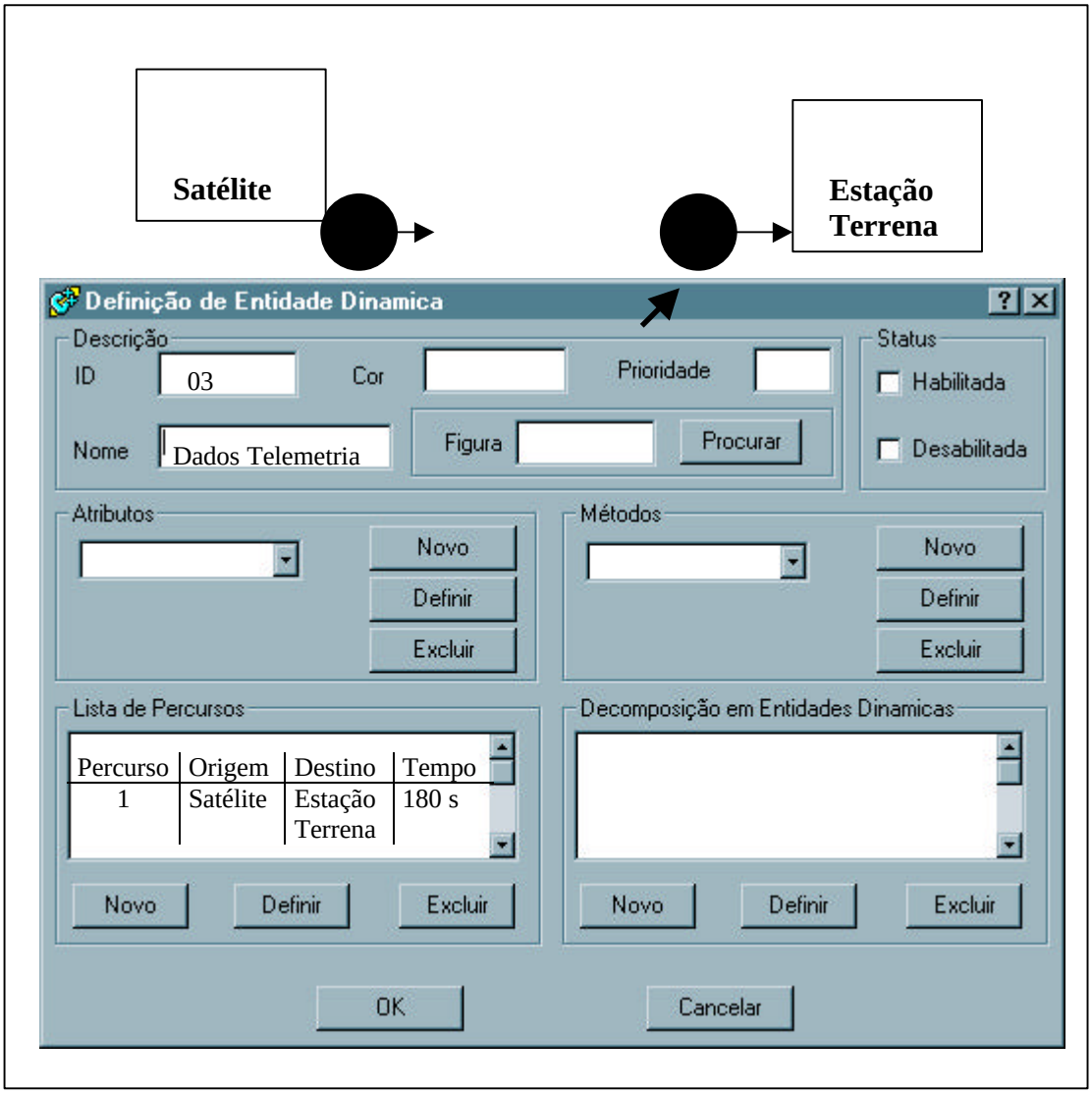

Fig. 4.9 - Definição do percurso da entidade dinâmica Dados Telemetria.

Para definir os possíveis estados da entidade estática Satélite, a janela de definição da entidade é acionada, Figura 4.7. Em seguida a definição de um novo estado é feita, pressionando-se o botão Definir do grupo Estados. À entidade estática Satélite será atribuído o estado Enviando Dados, conforme mostra a Figura 4.10. Outros estados desta mesma entidade são definidos de maneira análoga.

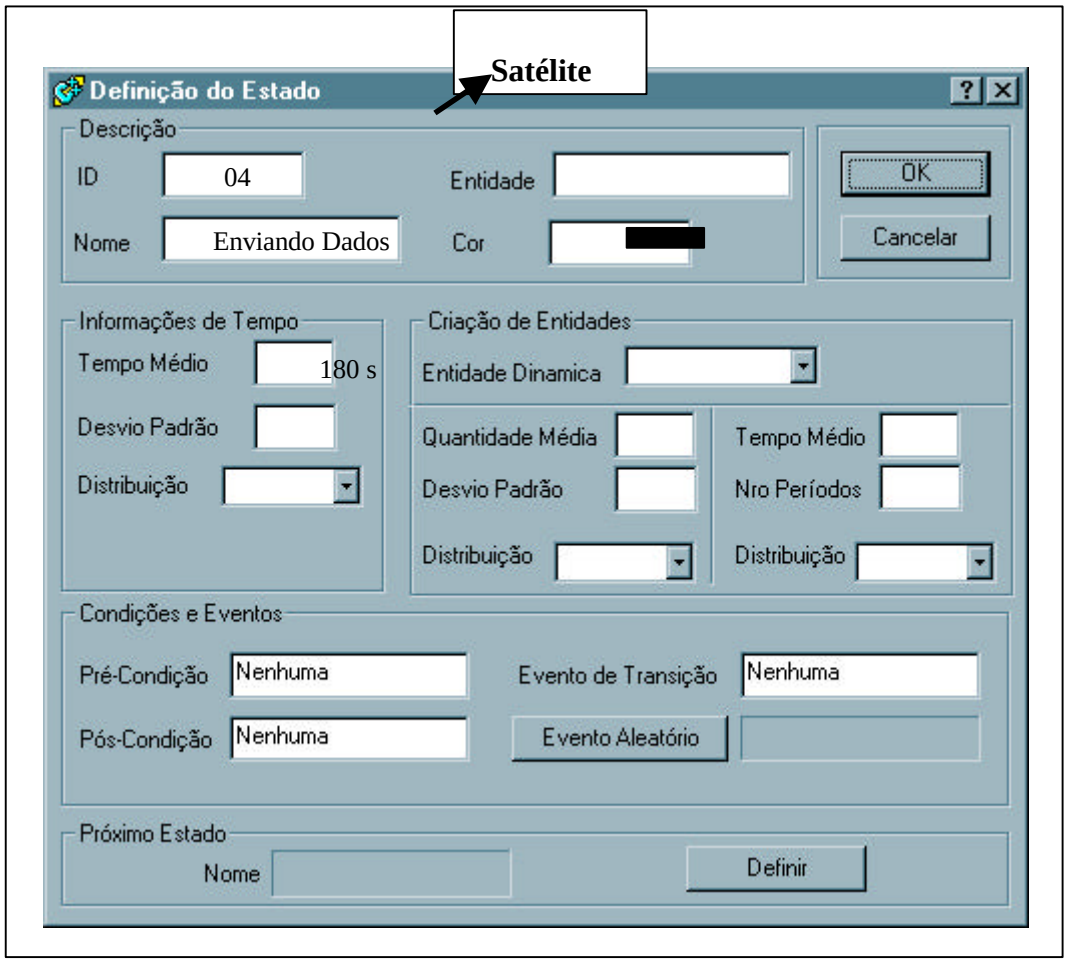

Fig. 4.10 - Definição do estado Enviando Dados da entidade estática Satélite.

Se durante o tempo de permanência no estado Enviando Dados, a entidade Satélite criar alguma entidade dinâmica, esta informação é fornecida através do grupo **Criação de Entidades** da janela de definição do estado, como pode ser visto na Figura 4.10.

O nome da entidade dinâmica deve ser fornecido ou selecionado de um conjunto já existente e a quantidade deve ser informada no campo **Quantidade Média**; caso esta quantidade seja aleatória, a função de distribuição deve ser informada. Quanto ao tempo de criação, se ele for fixo, deve ser preenchido somente o campo **Tempo Médio;** se ele for aleatório, uma função de distribuição deve ser selecionada; e se for periódico o campo **Tempo Médio** conterá o intervalo de tempo dos períodos e o campo **Nro Períodos** informará quantas vezes o período deve ser gerado.

Pode acontecer que, durante o estado Enviando Dados da entidade Satélite, exista a probabilidade de ocorrer uma falha na transmissão. Esta falha pode ser modelada por meio da geração de um evento aleatório. Este evento é definido a partir da janela de definição do estado Enviando Dados (Figura 4.10), pressionando-se o botão Evento Aleatório e provocando-se a abertura de uma nova janela para a definição deste evento. A janela de diálogo para a definição do evento Falha na Transmissão é ilustrada na Figura 4.11.

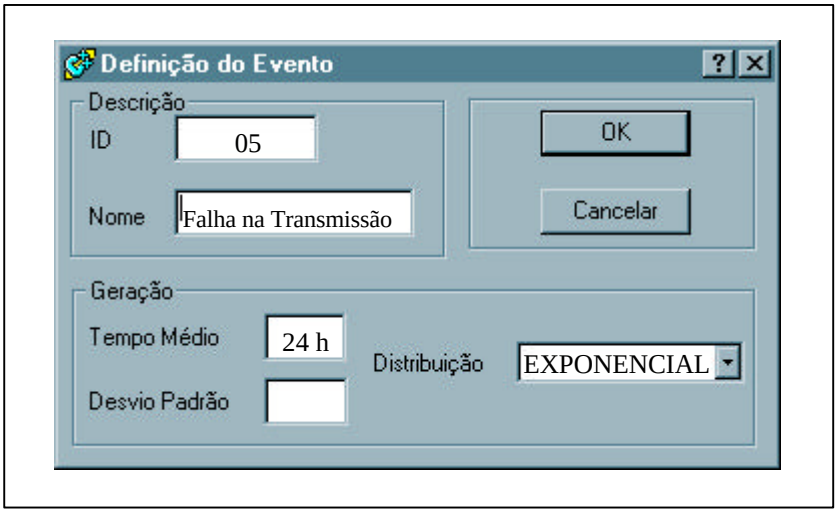

Fig. 4.11 - Definição do evento Falha na Transmissão.

Para ilustrar o emprego do elemento Fila na modelagem, é considerada a hipótese de que o recebimento de dados pela Estação Terrena é mais lento do que o envio de dados pelo Satélite. Isto causaria um acúmulo de entidades dinâmicas Dados de Telemetria, esperando para serem processadas.

Neste caso, uma fila poderá ser criada para armazenar as entidades Dados de Telemetria, correspondente ao período de uma passagem do satélite. O procedimento para a sua criação é o mesmo utilizado para a criação das entidades estáticas e dinâmicas.

A Figura 4.12 mostra a janela de definição da fila, que foi chamada **Fila de Dados**. O evento de entrada é preenchido automaticamente pelo sistema com base na informação

gráfica, conforme mostrado na Figura 4.12. A condição de saída é preenchida com Entidade Desocupada.

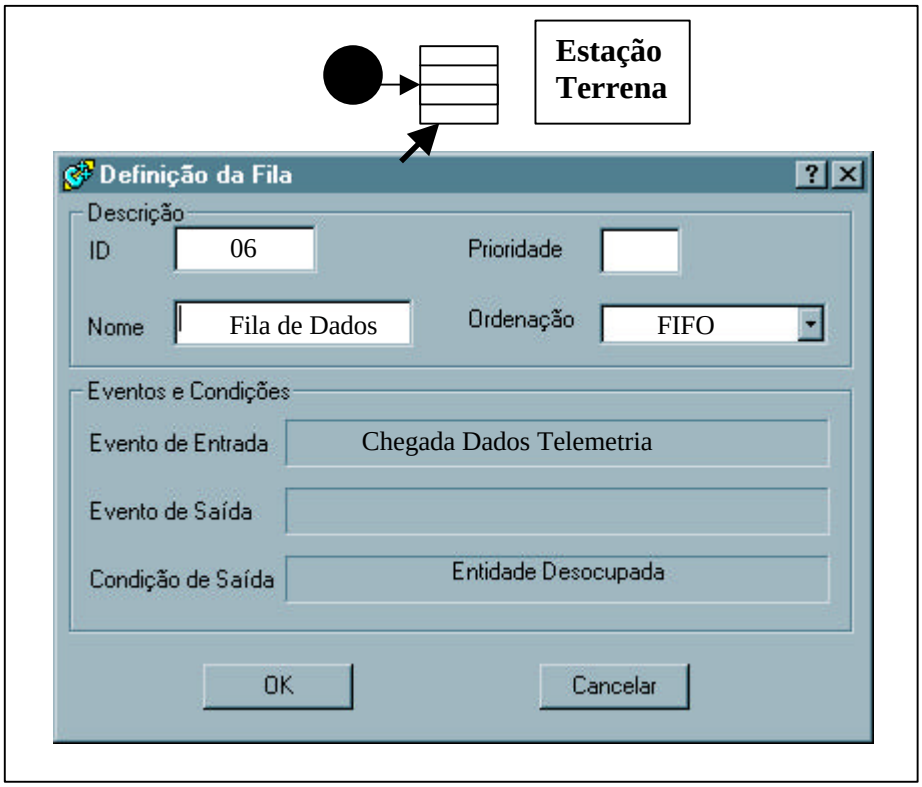

Fig. 4.12 - Definição da Fila de Dados.

As variáveis, como já mencionado anteriormente, podem ser utilizadas no modelo para definir pré e pós-condições. Elas serão variáveis globais. A Figura 4.13 mostra a janela de definição de uma variável.

Fig. 4.13 - Definição de uma variável.

4.8 ANIMAÇÃO DO MODELO NO AMBIENTE

A animação do modelo será feita por meio da **criação e destruição** de suas entidades componentes, pelo **fluxo** destas entidades, e ainda pelos vários **estados possíveis** que as entidades poderão passar durante a animação. Estes estados representam indiretamente as transformações realizadas pelas entidades.

A animação do modelo estará vinculada a tempos associados a cada um dos movimentos animados exibidos, obedecendo uma ordem cronológica.

Como já mencionado anteriormente, as entidades compostas são constituídas de mais de uma entidade básica. Este fato torna possível a decomposição da entidade em um ou mais níveis de detalhes, criando uma hierarquia de decomposição. Neste caso, a animação poderá ser visualizada nos vários níveis, considerando a entidade composta como uma entidade única no modelo, ou nos níveis inferiores de decomposição, considerando a animação de suas entidades constituintes. A seleção das entidades que serão animadas num determinado nível do modelo pode abranger uma única entidade, um conjunto de entidades selecionadas, ou todas as entidades do modelo naquele nível.

O fluxo das entidades do modelo será representado visualmente pelas várias entidades dinâmicas definidas, instanciadas num dado momento da animação do modelo, e pela dinâmica de movimento que cada entidade irá possuir.

Para as transformações que as entidades poderão sofrer ou realizar, será associado um tempo de execução, e esta transformação será representada visualmente atribuindo-se uma cor para o estado durante o qual a entidade realiza a transformação.

A representação dos vários estados possíveis que uma entidade básica ou composta possa estar durante a animação do modelo, e o tempo de permanência nestes estados, é uma informação importante que deve ser prontamente identificada durante a animação do modelo. Para que isto seja possível, é recomendável a utilização de cores distintas para representar estados distintos de uma mesma entidade ou componente.

A simulação orientada a evento foi a escolhida para simular o funcionamento do sistema e, a partir daí, gerar sua animação. Esta escolha fundamentou-se na flexibilidade em relação à oferecida pelos blocos fixos da orientação a processo, embora não seja tão

estruturada quanto esta última. Ao modelador deve caber apenas a tarefa de definir os eventos, sendo de responsabilidade do ambiente escalonar e ordenar estes eventos e prover sua ocorrência no tempo apropriado da simulação.

4.8.1 PROBLEMAS TÉCNICOS COM A ANIMAÇÃO DE UM MODELO SIMULADO

Existem alguns problemas de gerenciamento do tempo na animação de um modelo simulado. Há diferentes maneiras de gerenciar a diferença entre o tempo virtual da simulação e o tempo virtual da animação.

Para a simulação discreta de eventos, que é a que será utilizada para a animação dos modelos no ambiente sendo proposto, os mecanismos de fluxo de tempo são baseados em um relógio ou são baseados nos eventos que ocorrem. Os mecanismos de fluxo de tempo baseados nos eventos, que é o caso mais comum, consistem em passar da data de ocorrência de um evento para a data de ocorrência de outro evento. Os eventos são organizados num cronograma de tempo e dois eventos que tem a mesma data de ocorrência são descritos como simultâneos.

A animação deve dar a ilusão de um processo contínuo. Na verdade, o tempo de um sistema real é uma variável contínua. Assim, é apropriado emular um processo contínuo utilizando um relógio que sincronizará todos os movimentos animados numa freqüência aceitável para os olhos humanos, que varia de 10 a 15 imagens por segundo.

Para contornar este problema, o ambiente deverá fixar o intervalo de tempo separando os quadros de uma animação, mas também permitirá que o modelador faça a sua adequação, aumentando ou diminuindo a freqüência de apresentação destes quadros.

Segundo Hill (1996), a transformação do tempo simulado num tempo pseudo-contínuo, é essencial para manter as possibilidades de validação do modelo. A Figura 4.14 apresenta a ilustração do tempo simulado e do tempo para ser usado na animação.

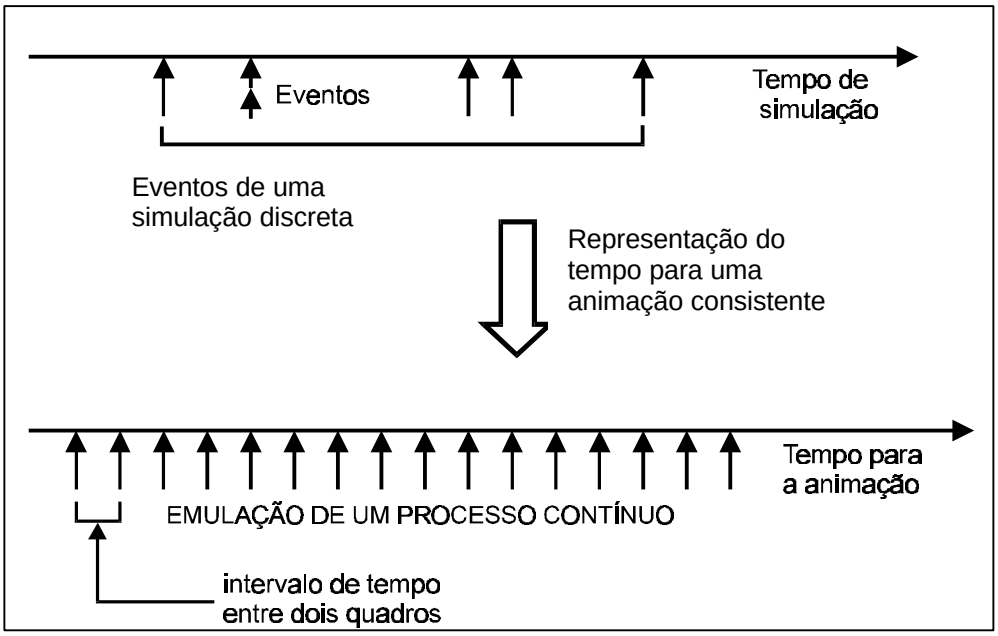

Fig. 4.14 - Representação do tempo para a animação. Adaptada de Hill, 1996.

Quando a escala de tempo escolhida para a simulação permite intervalos de tempo durante os quais nenhum evento gráfico ocorre, é fundamental contar este tempo na escala de tempo escolhida para a animação, mesmo quando nada tem que ser animado. Caso contrário, haveria um risco de chegar a conclusões erradas, quando o que está na janela não reflete o comportamento do sistema.

4.9 EMPREGO DE SIMULAÇÃO VISUAL INTERATIVA

O modelo gráfico criado será animado, tendo como base um modelo de simulação discreta de sistemas baseada em eventos. Os softwares que fazem a animação gráfica dos resultados de uma simulação deste tipo, na maioria dos casos, estão ligados à uma ferramenta específica de simulação ou à uma linguagem de simulação, conforme comentada na Seção 3.4.10.

Existem basicamente três técnicas empregadas nestes softwares de animação (Hill, 1996). A primeira técnica manuseia os resultados da simulação utilizando um arquivo pós-processamento, que é lido pelo animador após a simulação ter sido concluída.

A segunda técnica, que é chamada simulação direta, permite que a simulação e a animação sejam efetuadas ao mesmo tempo, sem possibilidade de interação.

A terceira técnica, que é a adotada para o ambiente proposto, chamada simulação visual interativa, permite que modificações sejam feitas durante a execução do modelo, com o objetivo de mostrar imediatamente os efeitos causados por estas modificações.

Com a utilização desta última técnica durante a animação de um modelo criado, o modelador poderá experimentar várias estratégias de funcionamento do sistema, incluindo a possibilidade de simular uma falha no momento da animação, que será representada por meio da desativação da entidade ou entidades correspondentes, de maneira interativa.

Este tipo de intervenção do modelador vai provocar um novo traçado da simulação do modelo, para que uma animação coerente possa ser novamente criada. A este tipo de simulação, dá-se o nome de simulação visual interativa. Com esta técnica, animação e simulação são realizadas ao mesmo tempo.

Esta técnica é criticada por especialistas em simulação (Hill, 1996) porque o emprego desorganizado de interações com o modelo pode comprometer seriamente os resultados da simulação. A interação pode falsificar completamente os cálculos estatísticos, que podem estar sendo baseados em amostras que são muito pequenas, e com isso provocar uma situação anormal, que nunca vai ser encontrada num sistema real. A interação e a seleção das opções podem invalidar um modelo.

Embora não muito utilizada nos modelos convencionais de simulação, a abordagem de simulação visual interativa é preferível neste caso, porque ela não exclui o especialista do processo de modelagem, que pode ser o próprio modelador do sistema ou a pessoa que trabalha em conjunto com este último.

A modelagem proposta envolve a criação de modelos de animação de um sistema novo, que muito dificilmente possui similar no mundo real para ser comparado. Devido ao seu conhecimento do sistema, especialistas contribuem muito para a validação do modelo. Com a experimentação de novas estratégias, o especialista pode fazer modificações diretamente no modelo e analisar o novo modelo resultante modificado.

A possibilidade de interação fornece uma abordagem mais rica comparada com a animação passiva, porque ela autoriza os especialistas a contribuírem com sua

experiência participando ativamente da busca de soluções aceitáveis, resultando numa melhor compreensão do modelo e melhoria na especificação do sistema. A Figura 4.15 mostra a seqüência de interação pretendida. As setas em preto indicam a interação.

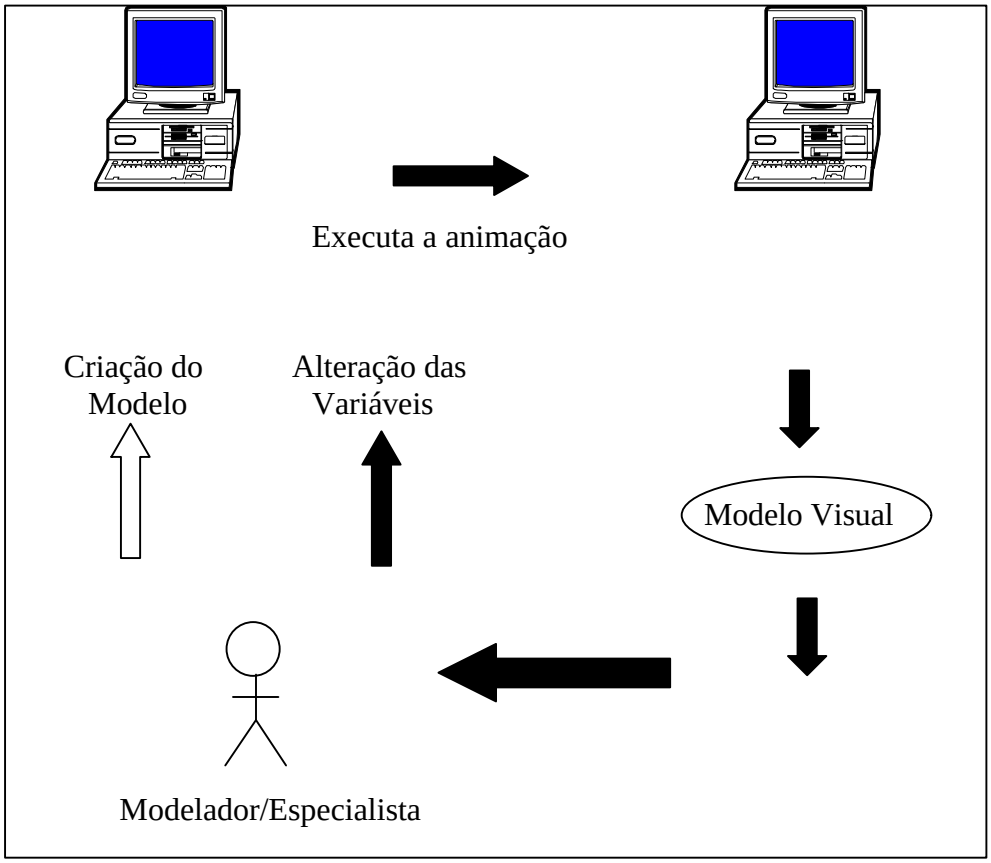

Fig. 4. 15 - Seqüência de interação.

Para evitar a possibilidade de modificar incorretamente certas entidades, as interações devem ser limitadas através de menus. Esta técnica parece restritiva, mas tem a vantagem de somente autorizar mudanças válidas no modelo.

4.9.1 MUDANÇAS PERMITIDAS

A interface gráfica com o modelador, baseada em menus, deve permitir que o ele altere interativamente:

- As funções de distribuição associadas à geração das entidades dinâmicas do modelo e à geração de eventos;
- Os tempos de percurso associados às entidades dinâmicas do modelo;

- As cores associadas às entidades estáticas e dinâmicas do modelo;
- O tempo definido para a rodada de animação;
- As entidades do modelo, inibindo a participação destas durante a animação e/ou ativando-as, após um tempo de inatividade;
- O nível de abstração do modelo, podendo alterar entre a visualização da animação dos vários subníveis criados; e
- O final de uma animação, mesmo que esta ainda não tenha esgotado o tempo definido inicialmente para a rodada.

O modelador poderá congelar um quadro qualquer da animação, permitindo assim que ele avalie melhor o modelo do sistema, e também o utilize como uma possível documentação do modelo.

4.10 CONDIÇÕES INICIAIS

Construir um modelo inicial do sistema, de tal forma que seus aspectos dinâmicos sejam representados com o uso da animação, requer que os modeladores adicionem informação relacionada à simulação às especificações existentes. Isto inclui um tempo de computação aproximado para um processo ou permanência num estado e comportamento estocástico das entradas externas, como taxa de chegada de dados e pontos de parada, com o intuito de experimentar com o sistema durante a animação.

Existem também condições que são estabelecidas antes do início da animação, que dizem respeito ao intervalo de tempo em que o modelo será animado (tal como horas, dias, semanas) e o número de rodadas ou corridas que serão executadas.

No ambiente proposto não será necessário que o modelador estabeleça, inicialmente, o intervalo de tempo da animação e nem o número de rodadas. Ele poderá iniciar a animação do modelo e interrompê-la quando desejar. Também são estabelecidos os pontos no modelo onde as coletas estatísticas serão feitas para posterior análise. O modelador poderá fixar estes pontos de coleta de dados. Apesar de a coleta de dados ficar um pouco comprometida devido ao uso de simulação interativa, a análise destes

dados pode vir a ajudar o modelador a escolher melhor as funções de distribuição utilizadas.

4.11 PROCESSO PARA A CRIAÇÃO DOS MODELOS DE ANIMAÇÃO UTILIZANDO O AMBIENTE

A criação do modelo de animação de um sistema para representar a sua dinâmica, segundo a abordagem proposta, necessitará de algumas informações que serão utilizadas somente para efeitos da simulação. Um modelo de simulação é construído de maneira transparente para o modelador, permitindo que uma animação seja criada.

Para que o processo de criação do modelo de animação seja melhor compreendido, é importante entender a sua relação com o ciclo de vida de um modelo de simulação, apresentado no Capítulo 3.

A Seção seguinte apresenta o ciclo de vida do modelo de animação, utilizando a técnica e o ambiente proposto, e traça uma relação paralela com as etapas de desenvolvimento do ciclo de vida do modelo de simulação, verificando também o que os dois ciclos apresentam em comum.

4.11.1 CICLO DE VIDA DO MODELO DE ANIMAÇÃO

O ambiente proposto deve suportar um processo de modelagem para a construção dos primeiros modelos de animação do sistema, dentro de uma abordagem evolucionária, permitindo assim que os requisitos do sistema e o seu funcionamento como um todo sejam melhor entendidos. Um modelo de animação bem construído pode ajudar na determinação de outros requisitos ainda não compreendidos ou não pensados. Além disso, com base neste modelo, um planejamento de testes poderá ser delineado.

Quando o modelo estiver completo, o modelador poderá modificar a especificação dos requisitos para incorporar o que foi aprendido. O modelo de animação é reprojetado e o processo é repetido até que se obtenha um modelo mais refinado do sistema e uma especificação de requisitos melhorada.

A seguir, é apresentado o ciclo de vida do modelo de animação proposto e os processos envolvidos na criação deste modelo, Figura 4.16. As formas de representação são

colocadas em símbolos ovais e os processos são representados por setas direcionadas. Este ciclo é comparado com as etapas de desenvolvimento do ciclo de vida de um modelo de simulação, apresentado no Capítulo 3, cujas formas de representação estão pintadas na cor cinza.

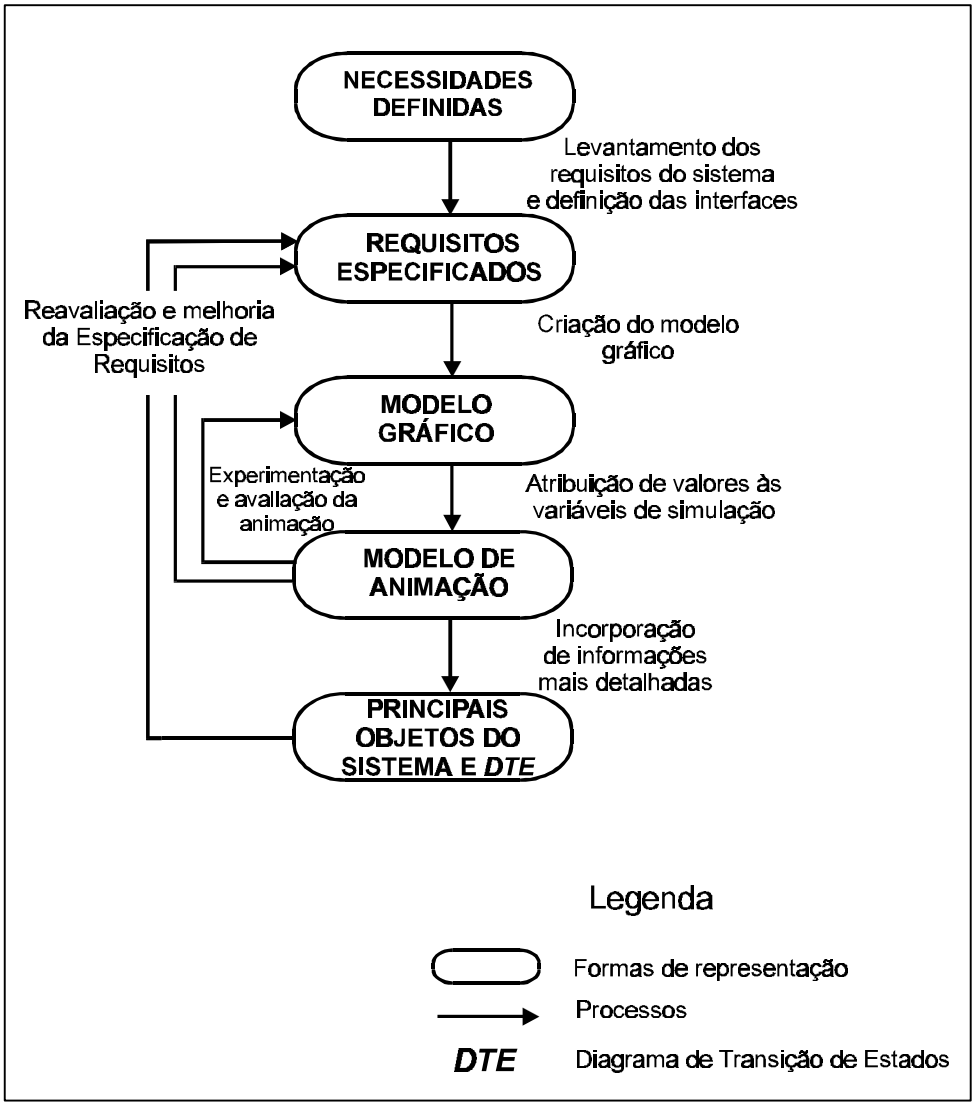

Fig. 4.16 - O ciclo de vida do modelo de animação proposto.

4.11.1.1 FORMAS DE REPRESENTAÇÃO E PROCESSOS DO CICLO DE VIDA

Os processos envolvidos no ciclo de vida do modelo de animação serão melhor explicados nesta Seção. Como as formas de representação do ciclo são resultantes destes processos, elas serão explicadas juntamente com os processos.

A primeira forma de representação, **Necessidades Definidas**, teve sua origem em um processo anterior, não contemplado por este ciclo de vida. Este processo envolve a identificação das necessidades e posterior estudo de soluções alternativas para o desenvolvimento do sistema (Pressman, 1992).

A identificação das necessidades é o ponto de partida para a evolução de um sistema computacional. Nesta fase, o analista ajuda o cliente na definição dos objetivos do sistema, estabelecendo-se quais informações serão produzidas, quais informações serão fornecidas, quais funções e desempenho do sistema são requeridos. O analista deve estar certo em distinguir o que são características críticas para o sucesso do sistema e características que seriam desejáveis, mas não essenciais.

No ciclo de vida do modelo de simulação, a **Descrição do Sistema e dos Objetivos do Estudo**, resulta de um processo onde são observadas características importantes do sistema, como sua constituição e as modificações que ele sofrerá, seu meio ambiente, seu comportamento não intuitivo, seu desempenho nas respostas, suas interdependências e sua organização.

Embora com algumas características diferentes, este processo possui pontos comuns com o correspondente processo de identificação das necessidades colocado anteriormente, principalmente no que diz respeito ao estabelecimento das fronteiras do sistema, seu meio ambiente e seu desempenho. Neste ponto do ciclo de vida do modelo de simulação, um estudo da factibilidade de se criar um modelo simulado já foi feita.

Uma vez as necessidades do sistema tenham sido estabelecidas e o estudo da factibilidade tenha sido realizado, tem início o ciclo de vida do modelo de animação conforme descrito a seguir. Vale ressaltar que, apesar dos processos estarem descritos de uma maneira seqüencial, existe interações e realimentações entre eles.

- **Levantamento dos Requisitos do Sistema e Definição das Interfaces**: este é um processo de refinamento, descoberta, modelagem e especificação. O escopo do sistema é refinado em detalhes, as principais funções e o desempenho requerido são estabelecidos. As interfaces do sistema com outros sistemas são definidas. Restrições de projeto, se existirem, também são estabelecidas neste momento. Este processo dá origem aos **Requisitos Especificados.** Esta especificação de requisitos inicial vai recebendo realimentações, conforme o sistema é melhor entendido, resultando em requisitos melhor especificados.

O processo correspondente no ciclo de vida do modelo de simulação, a **Formulação do Modelo**, trata do levantamento dos elementos essenciais do sistema e a definição de suas fronteiras, resultando no **Modelo Conceitual.** Apesar de mais informal, este processo se assemelha ao seu correspondente no ciclo de vida do modelo de animação.

- **Criação do Modelo Gráfico**: este é o processo responsável pela criação da representação gráfica do modelo de animação do sistema, segundo o simbolismo definido anteriormente. Neste processo são definidas as entidades estáticas e dinâmicas, seus possíveis estados, as filas, as interações humanas e os eventos do sistema.

No ciclo de vida do modelo de simulação, dois processos correspondem ao processo acima descrito: a **Representação do Modelo** e a **Programação**, onde o Modelo Conceitual é transcrito para um **Modelo Comunicativo**, que, na maioria das vezes, é uma representação gráfica do modelo para ser comunicada aos seres humanos, e o Modelo Comunicativo é transformado num **Modelo Programado,** que admite execução num computador. Atualmente, estes dois processos poderiam ser considerados como um só, porque os pacotes de simulação permitem que o modelador crie o seu modelo, sem se preocupar com a sua codificação.

- **Atribuição de Valores às Variáveis de Simulação**: neste processo são atribuídos valores às variáveis relacionadas à simulação, como tempos, distribuições estocásticas, e prioridades. A partir das informações contidas no modelo gráfico criado e daquelas fornecidas pelo modelador, os mecanismos de simulação

necessários para animar o modelo são acionados de maneira transparente. Como conseqüência, o escalonamento dos eventos no tempo é gerado e um **Modelo de Animação** é criado.

O processo correspondente no ciclo de vida do modelo de simulação, o **Projeto de Experimentos**, é responsável pela formulação de uma estratégia que possibilite a coleta de informação, resultando no **Modelo Experimental.** Esse processo tem um aspecto bem mais formal do que o seu correspondente no ciclo de vida do modelo de animação, pois é feita a validação do modelo com a realidade. Ele emprega uma série de técnicas que o auxiliam na análise estatística dos resultados da simulação.

- **Experimentação e Avaliação da Animação:** este processo é empírico e baseia-se fortemente na capacidade do modelador de experimentar valores diferentes das variáveis relacionadas à simulação e observar o efeito resultante na animação. É neste momento que realimentações e refinamentos nos modelos gráficos e de animação devem ser feitos, até que se consiga uma compreensão satisfatória do funcionamento do sistema como um todo, identificando seus pontos críticos de funcionamento, caso estes existam. Neste momento do processo, já é possível conduzir **uma Reavaliação e Melhoria dos Requisitos Especificados,** apreendidas durante a animação.

No ciclo de vida do modelo de simulação, o processo correspondente a este é a **Experimentação**, porém ele não faz realimentações no modelo.

- **Reavaliação e Melhoria dos Requisitos Especificados**: este processo incorpora à especificação de requisitos existentes melhorias apreendidas durante o a animação do modelo gráfico.

- **Incorporação de Informações mais Detalhadas**: este processo envolve o fornecimento de informações mais detalhadas sobre as entidades componentes do modelo, não mais com o propósito de animação, mas sim de definir melhor os futuros objetos de software do sistema, que no momento podem estar representados como entidades. É por meio deste processo, que se deseja eliminar o *gap* entre a

construção e avaliação do modelo de animação e a definição dos principais objetos de software do sistema a ser implementado.

Com as informações fornecidas para a criação do modelo de animação, principalmente às referentes aos atributos e aos estados das entidades, e com a incorporação de informações mais detalhadas, têm-se como resultado a **Descrição dos Principais Objetos de Software do Sistema**, **Diagramas de Transição de Estados das Entidades,** que podem ser gerados a partir do ambiente, e um possível esboço do **Planejamento de Testes** para o sistema que pode ser delineado pelo modelador. Neste ponto, um processo de **Reavaliação e Melhoria dos Requisitos Especificados** pode ser novamente conduzido.

O processo correspondente à este no ciclo de vida do modelo de simulação continua sendo a **Experimentação**, responsável por experimentar o modelo com um propósito específico, tendo como conseqüência os **Resultados do Modelo.** Este processo tem um aspecto bem mais formal do que o seu correspondente no ciclo de vida do modelo de animação, e emprega técnicas de análise dos resultados mais rígidas, tais como análise de sensibilidade e otimização (Welch, 1983). Baseado nestes resultados, o modelo criado pode ser alterado ou redefinido, através do processo de **Redefinição**.

Pode ser observado que, comparando o ciclo de vida do modelo de simulação com o ciclo de vida do modelo de animação, este último tem um enfoque maior na construção incremental do modelo, considerando o processo de experimentação e avaliação, e menor num processo de construção seqüencial.

4.11.1.2 HEURÍSTICAS PARA A CRIAÇÃO DO MODELO DE ANIMAÇÃO

A criação do modelo de animação envolve subjetividade e é bem diferente do modelo criado, considerando as técnicas tradicionais de especificação de sistema. Tendo isto em vista, a seguir são colocados alguns procedimentos que podem auxiliar o modelador no processo de criação deste modelo:

1) Identificação das principais entidades estáticas do sistema:

- Identificar os possíveis estados de uma entidade (transformações); e

- Atribuir tempo de permanência nos estados;

2) Identificação das principais entidades dinâmicas do sistema:

- Identificar fluxo de dados e entidades;

3) Identificação das filas;

4) Identificação dos eventos:

- Identificar os eventos periódicos; e
- Identificar os eventos aleatórios;

5) Criar as variáveis do modelo responsáveis pelo estabelecimento de pré e pós-condições e sincronismo;

6) Estabelecer as condições iniciais para a animação do modelo e acrescentar informações estocásticas ao modelo; e

7) Refinar o modelo criado, fazendo decomposições das entidades estáticas e dinâmicas para que mais detalhes sejam incorporados (*top-down*), ou reunindo estas entidades numa única entidade (*bottom-up*).

Os procedimentos apresentados foram colocadas em uma seqüência com o intuito de facilitar uma abordagem inicial, mas eles não são executados necessariamente nesta seqüência. Isto vai depender do conhecimento e experiência do modelador durante a criação do modelo.

4.12 IDENTIFICAÇÃO DOS OBJETOS DE SOFTWARE DO MODELO

O processo de criação do modelo envolve, além da criação das entidades estáticas e dinâmicas identificadas num primeiro momento, os refinamentos sucessivos destas entidades.

Estes refinamentos são feitos por meio da decomposição das entidades em outras entidades e também pela incorporação de informações a respeito de seus atributos e métodos.

No final do processo de criação do modelo, tem-se um conjunto de entidades estáticas e dinâmicas que possuem atributos e métodos específicos utilizados para compor o modelo de animação do sistema, e atributos e métodos identificados como necessários para a modelagem da entidade, mas que não participam do modelo de animação. Além disso, com as informações definidas para as entidades estáticas do modelo, é possível gerar o diagrama de transição de estados de cada uma delas.

Com as informações obtidas da modelagem e baseado nas necessidades do sistema, o modelador tem meios para identificar quais entidades ou conjunto de entidades tornariam-se objetos de software e quais não, e a partir deste ponto, adotar uma técnica de análise e projeto orientados a objetos poderia ser adotada para auxiliá-lo nos passos seguintes da construção do sistema.

No processo de identificação dos objetos de software do sistema pode ocorrer, por exemplo, que um objeto de software seja a agregação de várias entidades dinâmicas e estáticas, ou ainda que uma entidade seja composta de vários objetos de software. Além disso, entidades presentes no modelo de animação podem deixar de existir no sistema a ser construído.

A seguir é colocado um exemplo de como os objetos de software seriam obtidos a partir do modelo de animação criado.

Exemplo 3: Pedidos de compra de clientes chegam de 10 em 10 minutos para o Sistema de Vendas de uma empresa fictícia. O cliente é modelado como um Ser Humano e o Sistema de Vendas é modelado como uma entidade estática. Os pedidos de compra são entidades dinâmicas que partem de Cliente e chegam até o Sistema de Vendas. A Figura 4.18 ilustra o modelo.

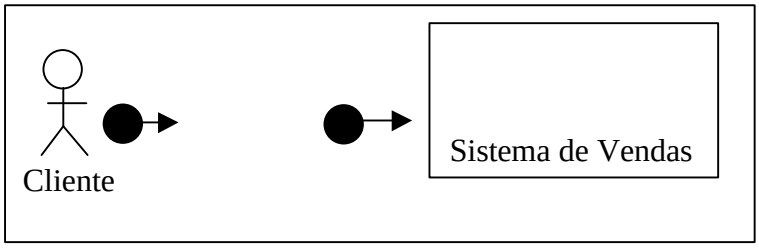

Fig. 4.17 - Exemplo de pedidos que chegam num Sistema de Vendas.

As informações iniciais que devem ser fornecidas para o elemento Ser Humano é um nome, no caso Cliente, quais os seus possíveis estados, o tempo de permanência nestes estados e com que freqüência os pedidos são gerados. Estas informações são suficientes para garantir a animação desta entidade.

Porém, conforme o modelo vai sendo refinado e melhor entendido, pode ser importante armazenar mais atributos do Cliente, como endereço, telefone, situação de crédito, quantidade de pedidos colocados, etc. Estes atributos ficam armazenados na Lista de Atributos do elemento Cliente. Analogamente, funções do Cliente como efetuar pagamento, fazer alteração de endereço e mudar o número do telefone, ficam armazenados na sua Lista de Métodos.

Tanto os atributos como os métodos acima definidos, não participam do modelo de animação criado, mas são essenciais, caso a entidade Cliente se torne um objeto de software que deva fazer parte do sistema.

4.13 MAPEAMENTO PARA OS ELEMENTOS-CHAVE

O Capítulo 2 deste trabalho identificou os elementos-chave (ECs) que deveriam estar representados na modelagem da dinâmica de um sistema. A partir desta necessidade, foi apresentado um simbolismo para criar os modelos de animação e foi concebido um ambiente para dar suporte à técnica de criação destes modelos.

O mapeamento dos elementos-chave para o modelo de animação foi feito com o intuito de verificar se eles estão sendo realmente considerados e representados neste modelo. A seguir estes elementos são listados novamente com as considerações sobre o mapeamento.

Os acontecimentos, conforme definidos anteriormente, serão representados por entidades compostas dinâmicas e/ou estáticas, envolvendo o conjunto dos possíveis estados que as entidades estáticas possam estar, os eventos associados e o movimento das entidades dinâmicas.

EC1- Seqüência dos eventos ou acontecimentos: será representado durante a animação do modelo, por meio da criação e destruição das entidades, do fluxo das entidades dinâmicas e da ativação e desativação dos estados das entidades estáticas.

EC2- Quando os acontecimentos ou eventos ocorrem no tempo: será representado durante a animação do modelo, baseado no relógio de simulação. A seqüência da animação indicará visualmente o momento de ocorrência destes acontecimentos ou eventos.

EC3- Representação de entidades componentes: representação gráfica das entidades básicas estáticas e dinâmicas e das entidades compostas estáticas e dinâmicas do modelo, através dos subníveis criados.

EC4- Fluxo de dados ou entidades (periódico ou aperiódico): será representado pelo movimento das entidades dinâmicas básicas e compostas durante a animação do modelo. A periodicidade ou não dos movimentos será determinada pelos eventos que os provocam.

EC5- Transformações: estarão embutidas nos estados das entidades estáticas, representadas por meio do tempo de permanência associado ao estado.

EC6- Paralelismo: será representado durante a animação do modelo, mostrando o movimento simultâneo das várias entidades dinâmicas e os possíveis estados das entidades estáticas. As entidades com atividade paralela, poderão também ser visualizadas quando se toma um instantâneo do modelo, identificando-as pelas cores diferentes usadas na sua representação.

EC7- Sincronismo: será representado durante a animação do modelo, mostrando o movimento síncrono das entidades envolvidas, que pode ser estabelecido com o uso de pré e pós-condições. Além disso, o uso do rótulo identificador vai facilitar a visualização das entidades em sincronia quando for tomado um instantâneo do modelo.

EC8- Concorrência por recursos: será representada por meio de filas ligadas às entidades estáticas (recursos), quando seu uso for recomendado. A fila será ordenada de forma a controlar o acesso concorrente das entidades dinâmicas ao recurso, utilizando uma política de ordenação.

EC9- Pré e pós-condições: estarão definidas no atributo condição de saída das filas e nos atributos pré e pós-condição dos estados das entidades estáticas do modelo.

Para a definição das pré e pós-condições, serão utilizados os recursos disponíveis da linguagem de simulação adotada para a implementação do ambiente.

EC10- Estabelecimento de prioridades: serão estabelecidas nas entidades dinâmicas, com a atribuição de um número de prioridade, e implementadas no modelo com o uso de filas ordenadas por prioridade.

A concepção de um ambiente de modelagem para dar suporte à técnica de criação do modelo de animação foi apresentada neste Capítulo. Os modelos construídos no ambiente possuem características que tornam possível a visualização da seqüência dos eventos, o momento em que os eventos ocorrem, as transformações realizadas pelas entidades (comportamento) e o fluxo de dados. Paralelismo, concorrência e sincronismo entre os eventos, como também as pré e pós-condições e as transformações envolvidas são consideradas e visualizadas no modelo.

Levando em conta a caracterização deste ambiente, o Capítulo seguinte apresenta a especificação e a análise de uma estrutura orientada a objetos para a sua implementação.

CAPÍTULO 5

A INFRA-ESTRUTURA ORIENTADA A OBJETO DO AMBIENTE DE MODELAGEM

5.1 ESTRUTURA DE SIMULAÇÃO PARA O MODELO DE ANIMAÇÃO

Como já ressaltado no Capítulo 3 deste trabalho, técnicas de simulação discreta orientada a eventos serão utilizadas para tornar possível a criação do modelo de animação proposto no Capítulo anterior. Estas técnicas incluem os mecanismos necessários e importantes que devem fazer parte do modelo de animação, e eles são funcionalmente descritos abaixo.

- **Mecanismo de Escalonamento dos Eventos, Ordenação Cronológica e Avanço do Relógio Simulado**.

A função primária requerida para a construção do modelo de animação envolve o escalonamento dos eventos, colocando-os em uma ordem cronológica e avançando o relógio simulado. Esta função é realizada pelo programa executivo da simulação, conforme mostra o diagrama da Figura 5.1.

Uma vez montado o calendário dos eventos, a execução das rotinas associadas a eles ocorrem em pontos apropriados do tempo simulado. Cada evento será executado em uma seqüência ordenada, com o tempo simulado sendo avançado de um evento para o próximo. Os eventos especificam a lógica que controla as mudanças que ocorrem em tempos específicos do tempo. O comportamento dinâmico é então obtido pelo processamento seqüencial dos eventos e pela coleta de dados nos tempos dos eventos. Em qualquer linguagem de simulação orientada a evento devem existir métodos para a realização de todas estas mudanças.

Os mecanismos de fluxo de tempo baseado em eventos consistem em passar da data de ocorrência de um evento para a data de ocorrência de outro. Isto causa problemas para a animação do modelo, que deve dar a ilusão de um processo contínuo. Normalmente, este problema é contornado com o uso de intervalos de tempo fixos separando os quadros de animação, conforme foi explicado na Seção 4.8.1.

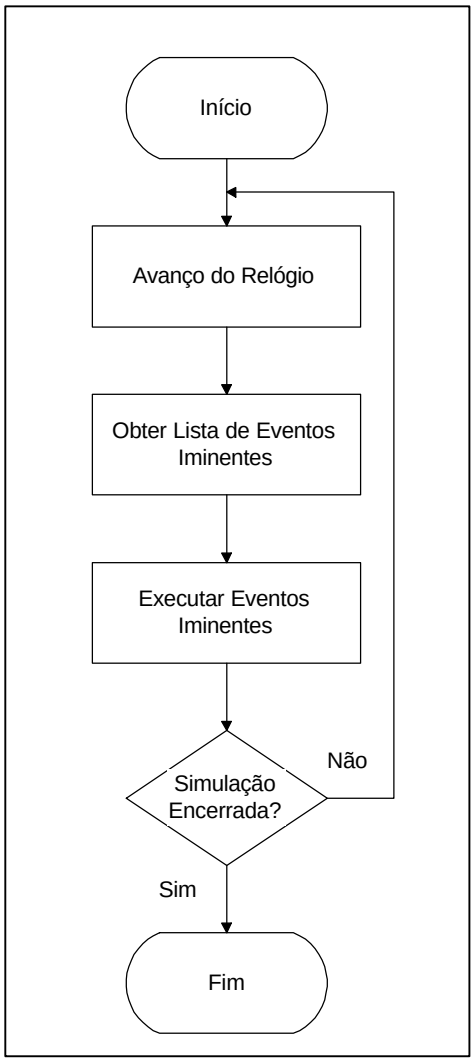

Fig. 5.1 - Programa executivo da simulação orientada a eventos.

- **Mecanismos de Manipulação de Filas.**

As filas são locais de espera para as entidades que circulam num modelo de simulação, aguardando o uso de algum de seus recursos. Quando o recurso fica liberado, algoritmos de escalonamento são utilizados para decidir qual entidade usará o recurso. Os

algoritmos de escalonamento definem as disciplinas das filas e os mecanismos necessários para a manipulação clássica de uma fila incluem inserir entidade na fila, retirar entidade da fila e contar número de entidades na fila.

- **Mecanismos de Coleta e Suporte Estatísticos.**

Durante a rodada de um modelo de simulação podem ser estabelecidos pontos onde os valores das variáveis do modelo devem ser coletados. A partir destes valores, a validade dos resultados pode ser analisada por meio de inferências estatísticas ou predições. Várias linguagens de simulação implementam técnicas adequadas para este estudo.

- **Mecanismos para Geração de Amostras Aleatórias.**

A rodada de um modelo de simulação normalmente requer a obtenção de amostras aleatórias de uma ou mais distribuições. A maneira mais utilizada de se fazer isto é, primeiro gerar uma ou mais amostras uniformemente distribuídas, entre 0 e 1, e então transformar as amostras uniformes nas novas amostras desejadas. Estas amostras independentes e uniformemente distribuídas são chamadas números aleatórios. Existem outros métodos de se fazer geração de amostras aleatórias que podem estar também implementados nas linguagens de simulação.

- **Funções de Distribuição de Probabilidade para a Geração dos Valores das Variáveis Aleatórias.**

Distribuições de probabilidade ocorrem em quase todas as partes do modelo de animação, representando o comportamento estocástico do modelo. O modo mais simples de se representar uma distribuição é utilizando funções de distribuição e funções de densidade de probabilidade. Algumas das distribuições mais utilizadas na modelagem de sistemas e presentes nas linguagens de simulação incluem: Distribuição Uniforme, Distribuição Triangular, Distribuição Exponencial, Distribuição de Poisson, Distribuição Normal, Distribuição Lognormal, Distribuição Erlang e Distribuição Gama (Soares, 1990).

Desde que estes mecanismos necessários da simulação já se encontram implementados na maioria das linguagens de simulação, estas funções já pré-programadas poderão ser

utilizadas com a adoção de uma linguagem de simulação para implementação do ambiente, reduzindo assim significativamente o tempo e o trabalho necessário.

O tipo de simulação adotada para a criação do modelo de animação será a simulação discreta orientada a eventos. A Seção 3.4.6 apresentou as principais características de uma simulação deste tipo.

A infra-estrutura orientada a objetos do ambiente, apresentada na Seção seguinte, modela os objetos responsáveis pela parte gráfica e de animação do modelo. Ela foi concebida de tal forma que a lógica dos eventos está codificada nos métodos dos objetos, ou seja, o acionamento destes métodos estará na lista dos eventos que ocorrem no modelo de animação.

A utilização de uma estrutura orientada a objetos para modelar os elementos do modelo de animação, descritos no Capítulo anterior, ajudará a derivação dos principais objetos de software do sistema, de uma maneira mais natural, no final do processo de modelagem.

Os objetos desta infra-estrutura responsáveis pela animação, incorporam em seus métodos as rotinas necessárias para que a animação interativa ocorra. O acionamento destes métodos é feito pelo programa executivo da linguagem de simulação utilizada para implementação, que os transforma em ocorrências de eventos, conforme será explicado na Seção 5.2. Todos os eventos estarão agendados no calendário de eventos. O programa executivo controla a simulação pelo avanço do relógio e chamadas às rotinas relacionadas aos eventos, em intervalos de tempo estabelecidos pelo modelador para evitar problemas de descontinuidades na animação do modelo, conforme explicado na Seção 4.8.1.

Uma linguagem como C++, por exemplo, poderia ser empregada para a criação do modelo de animação, porém um esforço considerável teria que ser direcionado na implementação do calendário de eventos e no mecanismo para o processamento destes eventos na ordem cronológica apropriada. Como esta função é comum a todos os modelos de simulação de evento discreto, várias linguagens de simulação foram desenvolvidas fornecendo estas e outras facilidades específicas.

A infra-estrutura foi concebida de tal forma que, para as atividades de projeto e implementação do ambiente, a adoção de uma linguagem de simulação discreta orientada a eventos e a objetos é a opção mais indicada. Exemplos destas linguagens incluem MODSIM (Herring, 1990), Simula 67 (Dahl et al, 1968; Kirkerud, 1989) e QNAP2 (Pistre, 1991; Hill, 1993, 1996).

5.2 A ESPECIFICAÇÃO E A ANÁLISE DA INFRA-ESTRUTURA

A seguir é apresentada a especificação e a análise da infra-estrutura do ambiente proposto. O objetivo desta análise não é identificar todos os objetos, todos os seus métodos e atributos, mas sim apresentar aqueles que são necessários para que o modelo de animação seja criado.

Uma vez identificados estes objetos, o trabalho de projeto seguinte será incorporar tecnologia aos modelos de análise, utilizando recursos específicos de uma linguagem de simulação para sua implementação.

5.2.1 CLASSES BÁSICAS PARA A ANIMAÇÃO

Os objetos de software que compõem o ambiente são divididos em dois grupos: as classes de objetos gráficos, que incluem todos os elementos do simbolismo apresentado, e as classes de objetos de apoio à animação. O primeiro grupo de objetos é responsável pela apresentação visual e animação dos elementos do modelo. O segundo grupo é o responsável pelos mecanismos de simulação que serão utilizados. Esta divisão foi feita para facilitar o entendimento e o detalhamento futuro do modelo.

No primeiro grupo, a classe de objetos gráficos se especializa em classes de entidades estáticas, entidades dinâmicas, rótuloC, rótuloS, ferramentas gráficas e filas. Existem também as classes Evento e Estado, abrangendo desta forma todos os elementos necessários para a criação do modelo. A Figura 5.2 mostra o diagrama de herança das classes deste grupo de objetos.

Os diagramas mostrados nas Figuras 5.2, 5.3 e 5.4 utilizam a notação proposta pela UML (1997). As classes são representadas em retângulos com seus métodos e atributos e as linhas direcionadas representam relacionamentos de herança entre as classes.

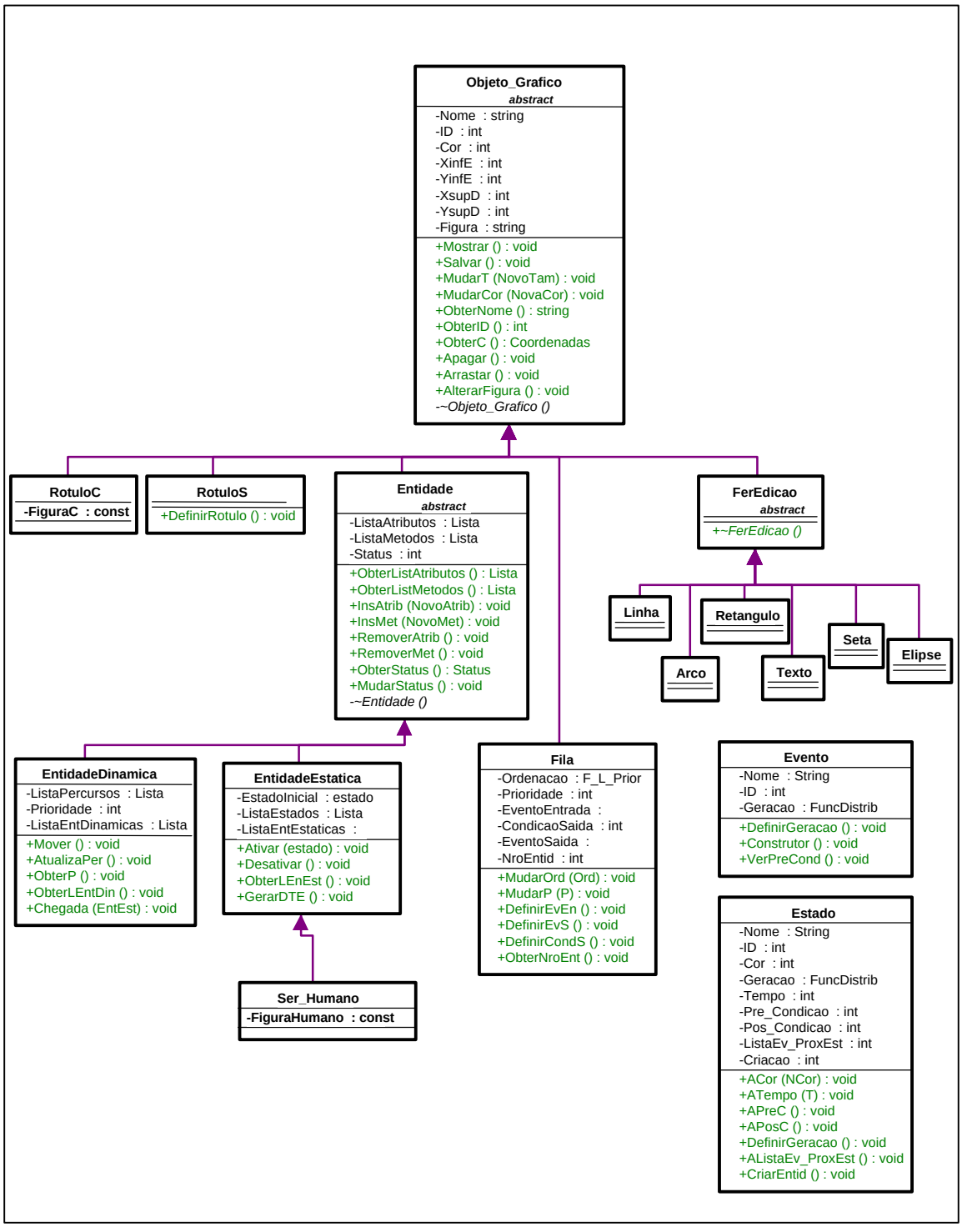

Fig. 5. 2 - Diagrama de herança do grupo objetos gráficos.

O primeiro grupo de classes resulta de uma análise de domínio, e de uma análise com o objetivo de produzir animação para o modelo criado. Ainda neste grupo está definida a

classe de ferramentas gráficas, que oferece ao modelador recursos para complementar o modelo gráfico criado, possibilitando a inclusão de textos explicativos, caixas, elipses e setas. Estas ferramentas também podem ser utilizadas para a edição de figuras simples, para substituir a representação gráfica padrão das entidades do modelo.

O segundo grupo inclui uma classe Coordenador e uma classe Variável, conforme mostra a Figura 5.3. O objeto Coordenador é o responsável pelos mecanismos de simulação que serão utilizados e pelo controle da animação. Ele irá encapsular em seus métodos as chamadas das rotinas necessárias da linguagem de simulação adotada, ou, caso seja utilizada uma linguagem de simulação orientada a objetos, será composto dos objetos responsáveis pelos mecanismos de simulação. Os objetos da classe variável representam as variáveis do modelo de animação, utilizadas para definir pré e pós-condições e os parâmetros da animação.

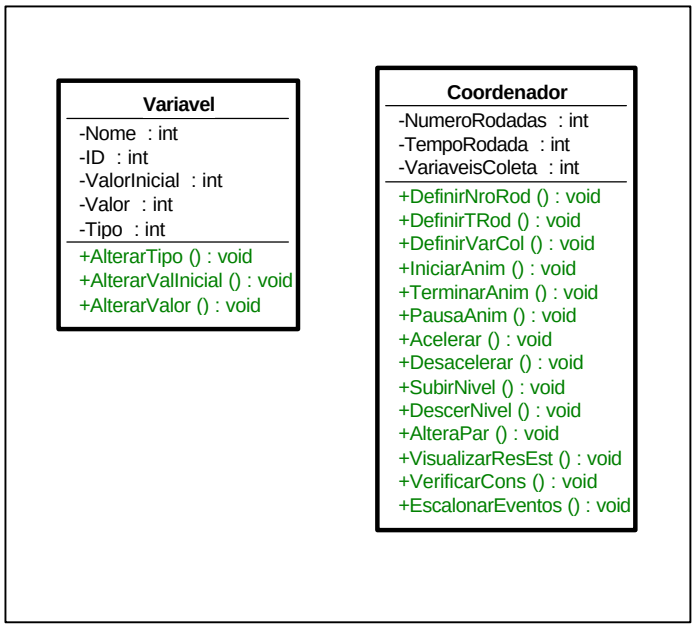

Fig. 5. 3 - Diagrama de classes do grupo de objetos de apoio à animação.

Os relacionamentos entre as classes de objetos dos dois grupos é mostrado na Figura 5.4. As linhas não direcionadas representam associações entre os objetos das classes conectadas. A cardinalidade 1..* ligada a uma classe, indica que existem um ou mais objetos da classe envolvidos na associação e a cardinalidade * indica que existem zero ou mais objetos da classe envolvidos na associação.

As seções seguintes descrevem as classes de objetos dos dois grupos, mostrando como esta estrutura torna possível a geração ordenada dos eventos num tempo simulado, para a criação do modelo animação.

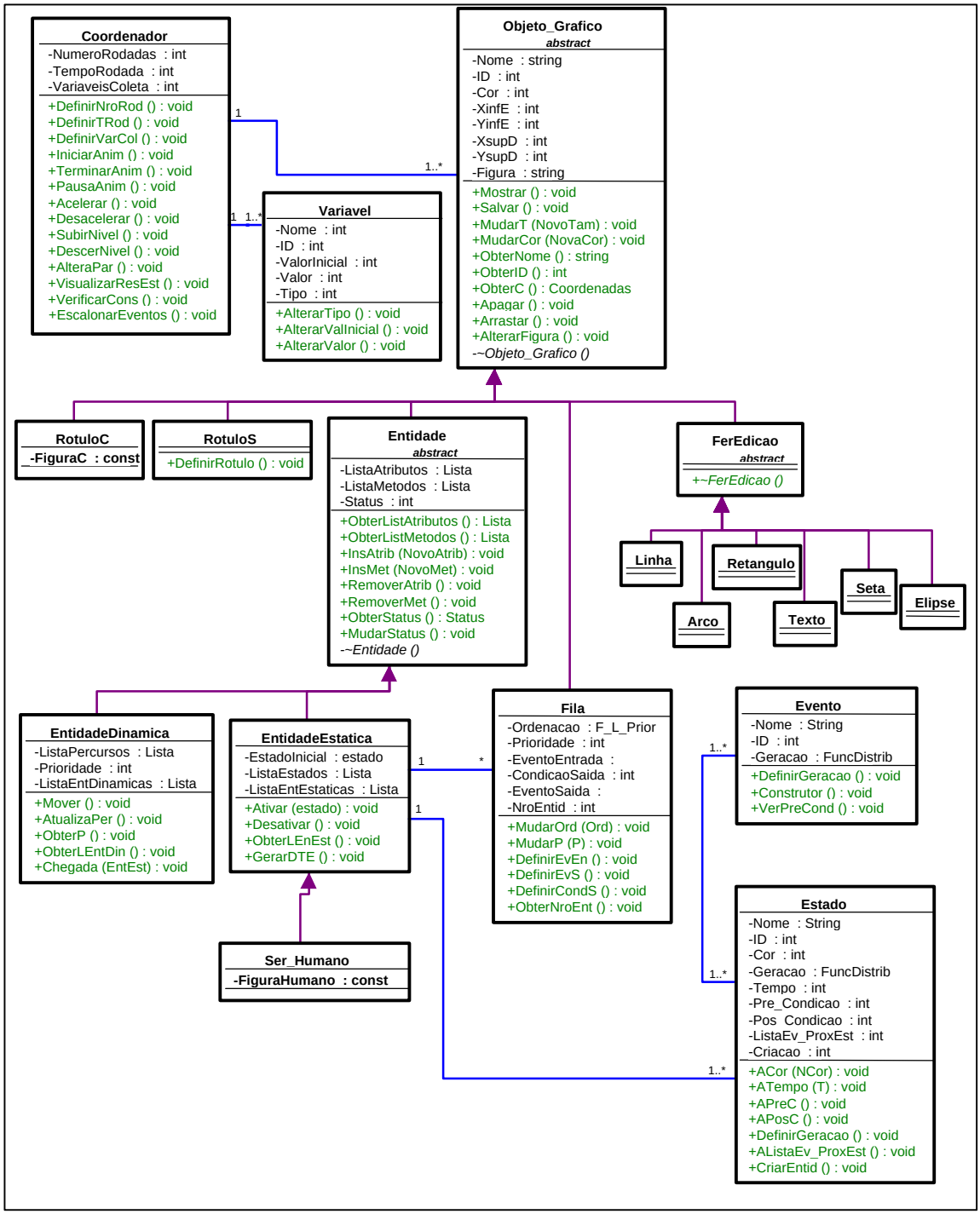

Fig. 5. 4 - Diagrama de classes mostrando o relacionamento entre os dois grupos de classes de objetos.

5.2.2 CLASSE OBJETO GRÁFICO

A super classe abstrata Objeto Gráfico possui os atributos comuns a todos os objetos gráficos do modelo. Ela armazena as coordenadas do canto inferior esquerdo (**XinfE, YinfE**) e as coordenadas do canto superior direito (**XsupD, YsupD**) dos objetos gráficos. Além disso, ela possui o atributo **Figura** que recebe um conteúdo diferente, dependendo da entidade que ela representa (retângulo, círculo, quadrado, ser humano, etc). O método AlterarFigura(NovaFig) é responsável pela alteração das figuras que representam as entidades estáticas e dinâmicas do modelo.

O atributo **Nome** armazena o nome lógico dos objetos do modelo de animação criado. O atributo **Cor** armazena a cor do objeto no momento da animação. É atribuído inicialmente a cor transparente para os objetos que representam as filas vazias e entidades estáticas do modelo, e a cor preta para os objetos que representam as entidades dinâmicas. A cor poderá ser alterada com a utilização do método MudarCor(NovaCor).

A super classe Objeto Gráfico permite a definição de métodos polimórficos para a edição dos vários tipos de objetos gráficos. Exemplos destes métodos incluem os métodos Mostrar, Salvar, MudarTamanho (NovoTam), MudarCor(NovaCor), Apagar, Arrastar e ObterCoordenadas. Os nomes dos métodos aparecem nos diagramas das figuras abreviados para melhorar a diagramação.

Os objetos gráficos criados tornam-se persistentes com a utilização do método Salvar, cuja implementação é feita nas classes derivadas. O atributo **ID** é a identificação única do objeto, que recebe um valor único na sua construção, e é utilizado para propósitos de armazenamento e recuperação do objeto.

5.2.3 CLASSES *RotuloC* E *RotuloS*

Estas classes são especializações da classe Objeto Gráfico, mostradas no diagrama da Figura 5.2. Elas são responsáveis pelos rótulos de condição C e de sincronia S, estabelecidos no simbolismo adotado.

A diferença entre a classe Objeto Gráfico e a classe RotuloC é que esta última fixa o conteúdo do seu atributo Figura como constante, não podendo mais ser alterado.

A classe RotuloS é a responsável por rotular um conjunto de entidades que estão síncronizadas. Como pode haver vários conjuntos de tais entidades, ao rótulo S é acrescentado um número inteiro, conforme foi mostrado no Capítulo 4, na descrição do simbolismo adotado. Para fazer isto, esta classe possui o método DefinirRotulo.

5.2.4 CLASSE ENTIDADE

A classe Entidade é uma classe abstrata que possui como atributos uma **Lista de Métodos**, uma **Lista de Atributos** e uma variável inteira **Status**, além dos métodos para a manipulação desses atributos, como pode ser visto na Figura 5.2.

Ela é uma classe intermediária a partir da qual as classes Entidade Estática e Entidade Dinâmica serão derivadas. Sua principal função é manter registrados os atributos e métodos das entidades que fazem parte do modelo, conforme estes atributos e métodos vão sendo identificados durante o processo de modelagem à medida que uma maior compreensão do sistema é obtida. Os atributos e métodos identificados não interferem diretamente nem no modelo de animação criado.

As entidades cujos atributos e métodos foram identificados durante a modelagem, são fortes candidatas a se tornarem objetos de software do sistema a ser implementado, ficando a cargo do modelador avaliar melhor as possibilidades no final do processo de experimentação do modelo de animação.

O atributo **Status** é utilizado para sinalizar se a entidade está habilitada ou desabilitada no momento da animação. Se o seu valor for desabilitado, a entidade não participa da animação. Este recurso é utilizado com o objetivo de testar hipóteses e verificar como o modelo reage. Isto possibilita, por exemplo, simular a falha temporária de um equipamento, uma falha de comunicação e a desativação de uma máquina para manutenção. O modelador pode mudar o valor do atributo Status de maneira interativa, e o método MudarStatus será o responsável pela alteração deste valor.

Os outros métodos, mostrados na Figura 5.2, são responsáveis pela manipulação das listas. Eles incluem ObterListaAtributos, ObterListaMétodos, InserirAtributo, InserirMétodo, RemoverAtributoLista e RemoverMétodoLista.

5.2.5 CLASSE FERRAMENTAS GRÁFICAS

A classe abstrata Ferramentas Gráficas é uma especialização da classe Objeto Gráfico e contém os atributos e métodos comuns aos objetos gráficos que são utilizados para complementar a criação do modelo. Estes objetos gráficos não se movem e não mudam de estado durante a simulação. Eles incluem **textos** de comentários, **linhas, setas, arcos, retângulos** e **elipses**, que são suas classes derivadas, conforme pode ser visto na Figura 5.2. O modelador também pode utilizá-los para a construção de figuras simples para representar graficamente uma entidade.

Os métodos de edição destes objetos, herdados da classe Objeto Gráfico, são implementados nas classes derivadas.

As linguagens de simulação orientadas a objeto, que utilizam um ambiente gráfico para a modelagem, normalmente já incorporam estas ferramentas.

5.2.6 CLASSE ENTIDADE ESTÁTICA

Esta classe é uma especialização da classe Entidade, que por sua vez é um objeto gráfico, conforme mostra o diagrama da Figura 5.2. Os objetos desta classe fazem parte da animação do modelo e por este motivo, a classe contém também os métodos necessários para isso. Os novos atributos e métodos são definidos a seguir.

- **Atributos**

  Estado Inicial: o estado inicial armazena o estado em que o objeto entidade estática se encontra no início da animação. O estado inicial padrão é “desocupado”, mas isto poderá ser alterado pelo modelador no momento da definição da entidade.

  Lista de Estados: esta lista conterá todos os possíveis estados de um objeto entidade durante a animação do modelo. Cada estado é modelado como um objeto a parte, com seus atributos e métodos próprios, conforme será explicado na Seção 5.2.11.

  Lista de Entidades Estáticas: esta lista contém os nomes de outros objetos da classe Entidade Estática, caso o objeto em questão tenha sido decomposto. Ela armazena as entidades componentes dos vários níveis de decomposição dos objetos da classe Entidade Estática do modelo.

- **Métodos**

Ativar (nome do estado): este método é o responsável pela ativação do estado da entidade estática, passado como parâmetro. Ele é acionado a partir de uma transição de estado da entidade, disparado com a chegada de um evento. O acionamento do método Ativar, marca o início do tempo de permanência da entidade no estado e este tempo é passado para o programa executivo de simulação para controle do tempo simulado. O método causa o seguinte efeito:muda a cor da entidade para a cor determinada para o estado, utilizando o método MudaCor(NovaCor), e mantém esta cor até que uma nova transição de estado ocorra.

Desativar: o evento de fim do tempo de permanência da entidade em um estado é implementado pelo método Desativar, e este evento é agendado na lista de eventos do programa executivo da simulação automaticamente após o método Ativar, caso nenhum outro evento de transição definido pelo modelador ocorra. Ele é o responsável pela transição da entidade estática para o estado desocupado, restituindo sua cor inicial. Após a transição ser efetuada, o método verifica as pós-condições estabelecidas para o estado.

ObterListaEntEst: recupera os objetos entidades estáticas que compõem o nível inferior da decomposição hierárquica do objeto entidade estática em questão, para visualização e animação.

GerarDTE: este método é responsável por gerar o diagrama de transição de estados da entidade estática, utilizando as informações das definições de estados e eventos de transição.

Todas as entidades estáticas são criadas no início da animação do modelo e mostradas na tela. Caso o usuário queira visualizar todas entidades dinâmicas definidas, de uma maneira estática, ele poderá fazê-lo a qualquer momento por meio de uma opção, que deverá ser disponibilizada em um menu na janela de interface do ambiente.

5.2.7 CLASSE SER HUMANO

Esta classe é uma especialização da classe Entidade Estática, conforme mostra a Figura 5.2. Ela é responsável pela criação do elemento Ser Humano definido no simbolismo. Os objetos desta classe recebem inicialmente a cor transparente significando ser humano desocupado, e a cor muda para preto quando o objeto ser humano está ativo, caso o modelador não tenha selecionado outra cor.

A representação gráfica de um objeto ser humano foi escolhida para representar as interfaces externas do sistema. Caso o modelador queira que um elemento Ser Humano do modelo tenha movimento, basta que ele o modele como uma entidade dinâmica e associe à sua representação uma figura humana.

5.2.8 CLASSE ENTIDADE DINÂMICA

Esta classe é uma especialização da classe Entidade, que por sua vez é um objeto gráfico, conforme mostra o diagrama da Figura 5.2. Os objetos desta classe têm movimento no modelo de animação, e por este motivo, ela contém os métodos necessários para isso. Os novos atributos e métodos são definidos a seguir.

- **Atributos**

  Prioridade: número inteiro utilizado para a ordenação das entidades dinâmicas numa fila de acesso a uma entidade estática, caso seja adotada a política de ordenação por prioridade. Este número é comparado com o número estabelecido no atributo Prioridade do objeto Fila, para determinar qual será a posição da entidade dinâmica na fila. Caso entidades com mesma prioridade cheguem à fila, será considerada a ordem de chegada como fator de desempate.

  Lista de Entidades Dinâmicas: esta lista contém os nomes de outros objetos da classe Entidade Dinâmica, caso o objeto em questão tenha sido decomposto. Ela armazena as entidades componentes dos vários níveis de decomposição dos objetos da classe Entidade Dinâmica do modelo.

  Lista de Percursos: os objetos entidades dinâmicas movimentam-se durante a animação do modelo. Este atributo armazena uma lista simples, contendo todos os

possíveis percursos do objeto dinâmico. Os elementos desta lista podem ser obtidos graficamente, com o posicionamento da entidade dinâmica nas respectivas origens e destinos, ou informados pelo modelador. Um modelo desta lista é apresentado na Figura 5.5 a seguir.

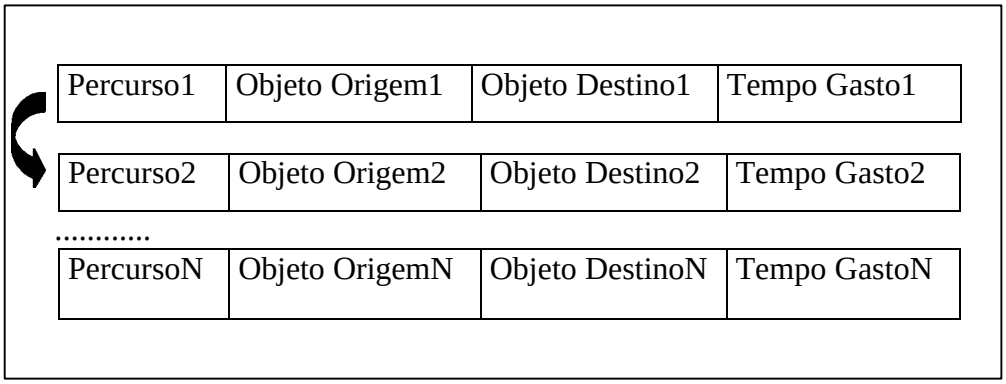

Fig. 5. 5 - Lista contendo os percursos de um objeto entidade dinâmica.

A informação sobre o tempo gasto no percurso deve ser introduzida pelo modelador. Assim que um percurso é realizado, o ponteiro de cabeça da lista passa a indicar o próximo elemento (percurso) da lista.

- **Métodos**

Mover: este é o método responsável pelo movimento da entidade dinâmica no modelo de animação. Ele consulta a Lista de Percursos da entidade dinâmica, selecionando o primeiro percurso da lista, e a move graficamente da entidade origem para a entidade destino. O acionamento deste método é um evento na lista de eventos do programa executivo da simulação e terá a duração estabelecida na variável Tempo Gasto associada ao percurso. Este tempo é passado como parâmetro para o programa executivo para controle do tempo simulado.

Com base na modelagem gráfica, o acionamento do método pode ser colocado automaticamente pelo ambiente como um evento na lista dos eventos do programa executivo da simulação. Assim que a entidade dinâmica chega ao destino, após o tempo simulado do movimento, o próprio método **Mover** dispara o método **Atualiza Percurso**, para atualizar a lista de percursos da entidade dinâmica.

Atualiza Percurso: este método simplesmente faz o próximo elemento da lista de percursos, definida para a entidade dinâmica, ser o primeiro elemento da lista.

Chegada : este método é acionado automaticamente, logo após o tempo de um percurso realizado por uma entidade dinâmica, sinalizando a entidade estática destino de sua chegada. O método Chegada é um evento na lista de eventos do programa executivo da simulação, que verifica se as precondições estabelecidas para a transição para o próximo estado da entidade destino são satisfeitas. Em caso afirmativo, a transição de estado será feita com o acionamento do método **Ativar** do objeto entidade estática, que é colocado na lista de eventos do programa executivo da simulação.

O evento de **Chegada** sempre ocorre após o acionamento do método **Mover** das entidades dinâmicas, logo o próprio evento Mover o coloca na lista dos eventos do programa executivo da simulação. A partir do modelo gráfico construído, é possível identificar os eventos **Chegada e Mover** das entidades dinâmicas automaticamente.

ObterPrioridade: recupera o atributo prioridade definido para a entidade dinâmica, para que ele possa ser comparado com o atributo Prioridade do objeto Fila e assim a posição da entidade dinâmica na fila possa ser definida.

ObterListaEntDin: recupera as entidades dinâmicas componentes do nível inferior na decomposição hierárquica da entidade dinâmica em questão, para visualização e animação.

As entidades dinâmicas são geradas ou criadas por alguma entidade estática ou ainda por meio de eventos. Sua criação, a partir de uma entidade estática, ocorre quando esta entidade estiver em um estado onde a criação da entidade dinâmica foi definida, anteriormente, pelo modelador. O acionamento do método construtor da entidade dinâmica por meio do método CriarEntidade definido para o estado, Seção 5.2.11, que passa a ser um evento na lista dos eventos do programa executivo da simulação.

A criação e o número de entidades dinâmicas criadas por uma entidade estática, pode também ocorrer de uma maneira aleatória. Neste caso a criação pode ser ativada a partir de um evento aleatório definido pelo modelador. O número de entidades criadas pode

ser fixo ou aleatoriamente gerado, a partir de uma função de distribuição. O estado responsável pela criação armazena estas informações conforme é mostrado na Seção 5.2.11.

Quando a geração é feita a partir de um evento, na definição do objeto evento é estabelecido uma função de distribuição ou a periodicidade da geração. A classe Evento é definida na Seção 5.2.10.

5.2.9 CLASSE FILA

A classe Fila é uma especialização da classe Objeto Gráfico. Os objetos desta classe recebem inicialmente a cor transparente significando fila vazia, e a cor muda para preta quando alguma entidade dinâmica está na fila. Como a representação gráfica de um objeto Fila não muda, o atributo Figura passa a ser constante, armazenando a figura representativa da fila proposta no simbolismo.

Os novos atributos e métodos da classe Fila são definidos a seguir:

- **Atributos**

  Ordenação: nome que define a política de ordenação da fila. As políticas de ordenação pode ser FIFO (First In First Out), LIFO (Last In First Out) ou P (Prioridade).

  Prioridade: número inteiro definido pelo modelador, caso o conteúdo do atributo Ordenação seja P. Este número é utilizado para definir a ordenação das entidades dinâmicas na fila.

  NroEntidades: variável inteira que armazena o número de entidades na fila. Esta informação é importante para detectar gargalos do sistema e também pontos de pouco uso.

  Evento de entrada: evento responsável pela entrada da entidade dinâmica na fila. Este evento será gerado automaticamente pelo ambiente, baseado na modelagem gráfica, que extrai o nome do objeto entidade dinâmica e o acionamento de seu método **Chegada**. Caso o evento de entrada na fila seja outro, o modelador deve definí-lo.

Condição de saída: expressão contendo a condição que deve ser satisfeita para que a entidade saia da fila. O ambiente automaticamente condicionará a saída de um objeto entidade da fila ao estado **desocupado** do objeto entidade estática, ao qual a fila está ligada.

Evento de Saída: evento gerado após a saída da entidade da fila, uma vez a condição de saída seja satisfeita. O modelador poderá definir mais de um evento de saída, caso seja necessário.

- **Métodos**

Foram definidos alguns métodos responsáveis pela manipulação dos atributos da fila. Estes atributos são preenchidos inicialmente pelo próprio construtor do objeto, mas podem ser alterados posteriormente. Os métodos definidos a seguir possibilitam estas mudanças.

MudarOrdenação: altera o conteúdo do atributo Ordenação, mudando a política de ordenação da fila.

MudarPrioridade: altera o valor do atributo Prioridade, caso a política de ordenação da fila seja por prioridade.

DefinirEvEntrada: define o evento que provoca a entrada de uma entidade dinâmica na fila.

DefinirEvSaida: define o evento que provoca a saída de uma entidade dinâmica da fila.

DefinirCondSaida: define a condição de saída de uma entidade dinâmica da fila.

ObterNroEntidades: Obtém o número de entidades dinâmicas na fila num dado momento.

Normalmente, a fila é necessária diante de uma entidade estática que está representando algum tipo de recurso ou processamento. Neste caso, as entidades dinâmicas ficam na fila até que a entidade estática fique no estado **desocupada,** tornando-se apta para receber uma nova entidade dinâmica.

Quando isto ocorre, a entidade deixa a fila realizando um movimento até a entidade estática e o evento de transição **Chegada** é gerado, fazendo com que a entidade estática saia do estado desocupada. Estes eventos serão gerados automaticamente pelo ambiente, com base na modelagem gráfica e submetidos à aprovação do usuário, para que este se certifique de que o mapeamento automático dos eventos foi feito corretamente.

Os mecanismos para a manipulação de filas estão disponíveis na maioria das linguagens de simulação, e não são considerados na análise da infra-estrutura.

5.2.10 CLASSE EVENTO

Os objetos da classe Evento são responsáveis pelos eventos que ocorrem de maneira aleatória ou periódica no modelo. Eventos podem ser gerados automaticamente, como conseqüência da animação do modelo, serem programados ou serem gerados aleatoriamente de acordo com uma função de distribuição.

Os objetos da classe Evento são definidos pelo modelador na criação do modelo de animação, por meio de uma interface gráfica, conforme foi ilustrado no Capítulo 4, ou obtidos a partir do modelo gráfico (exemplo: **Mover** e **Chegada**). Os atributos e métodos desta classe são definidos a seguir:

- **Atributos**

  Nome: atributo que contém o nome lógico do evento.

  ID: número atribuído automaticamente pelo ambiente que identifica unicamente o evento.

  Geração: atributo que armazena a função de distribuição utilizada para a geração do evento, ou o intervalo de tempo entre as gerações e o número de gerações, caso os eventos sejam periódicos.

- **Métodos**

  Construtor: este método é o responsável pela criação de um objeto evento e ele segundo a geração definida no seu atributo geração, caso o evento não seja

naturalmente gerado. O acionamento do construtor é um evento agendado na lista de eventos do programa executivo da simulação.

DefinirGeração: este método possibilita ao modelador definir e também alterar a geração dos objetos eventos.

VerificarPreCondição: este método verifica as precondições das entidades estáticas, que podem sofrer transição de estado devido a ocorrência do evento. Se estas condições são satisfeitas, a entidade muda de estado e o acionamento do método **Ativar** do objeto Entidade Estática é colocado como um dos próximos eventos da lista de eventos do programa executivo da simulação.

5.2.11 CLASSE ESTADO

Os objetos da classe Estado representam o estado de uma entidade estática, nos quais são estabelecidos o tempo de permanência da entidade estática no estado, as próximas transições, as pré e pós-condições e também uma cor para efeitos de animação.

Existe ainda um atributo Geração que é utilizado para estabelecer uma função de distribuição que pode gerar um tempo aleatório de permanência da entidade estática no estado.

Um estado poder ter associado a ele vários eventos de transição como mostra o diagrama de classes da Figura 5.4.

Durante o tempo de permanência de uma entidade em um estado, pode ocorrer a criação de novas entidades dinâmicas. Para isto, o atributo Criação e o método CriarEntidade foram definidos.

- **Atributos**

Nome: atributo que contém o nome lógico do estado.

ID: número atribuído automaticamente pelo ambiente que identifica unicamente o estado.

Cor: cor que a entidade estática tem durante sua permanência no estado.

Tempo: tempo estabelecido para a permanência da entidade no estado; este tempo pode ser fixo ou aleatório.

Geração: estabelece a função de distribuição para o tempo de permanência da entidade no estado, caso este tempo seja aleatório.

Criação: este atributo define o nome da entidade dinâmica que será criada durante a permanência da entidade estática no estado, define a quantidade de entidades a serem criadas, que pode ser fixo ou baseado em alguma função de distribuição, e o tempo necessário para o processo de criação (fixo, aleatório ou periódico).

A Figura 5.6 mostra a estrutura deste atributo. Quando a quantidade de entidades a serem produzidas é fixa, o campo Qte Média é preenchido com o valor fixado; se for aleatório, uma função de distribuição deve ser selecionada. Caso o tempo utilizado para o processo de criação for fixo, ele é informado em Tempo Médio; se for aleatório uma função de distribuição é selecionada e se for periódico, o tempo médio fornece o intervalo de tempo entre os períodos e o campo Nro Períodos deve ser preenchido com o número de períodos a serem gerados.

Nome	**Quantidade**	**Tempo para Criação**
	Qte Média – Função Distribuição- Desvio Padrão	Tempo Médio – Função Distribuição – Desvio Padrão - Nro Períodos

Fig. 5. 6 - Atributo Criação da classe Estado.

Precondição: expressão contendo a condição necessária para que a entidade mude de estado; a precondição normalmente é definida com variáveis globais, operadores lógicos e variáveis de estado do modelo. Ela é verificada pelo método **Chegada** da entidade dinâmica ou pelo método **VerificarPrecondição** de um objeto evento aleatório, desde que existe a possibilidade de um ou outro provocar a transição de estado.

Pós-Condição: expressão contendo a condição que deve ser satisfeita após a saída da entidade do estado. Esta condição normalmente é definida com variáveis globais, operadores lógicos e variáveis de estado do modelo, e é verificada pelo método

Desativar do objeto entidade estática ao qual o estado pertence, assim que o tempo de permanência no estado é esgotado.

Lista Ev-ProxEst: lista contendo o nome do evento que provoca a transição de estado da entidade e o nome do estado que a entidade estática estará após a transição (próximo estado). A Figura 5.7 ilustra a estrutura deste atributo.

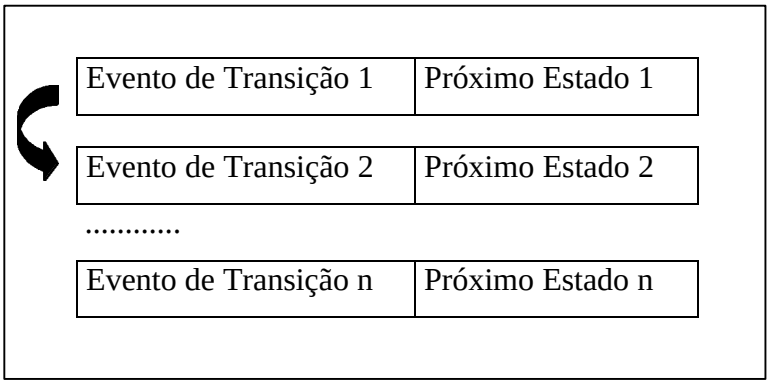

Fig. 5. 7 - Atributo Lista Ev-ProxEst da classe Estado.

- **Métodos**

Foram definidos alguns métodos responsáveis pela manipulação dos atributos do objeto estado. Estes atributos são preenchidos inicialmente pelo construtor do objeto, mas podem ser alterados posteriormente. Os métodos definidos a seguir possibilitam estas mudanças.

CriarEntidade: este método, baseado no conteúdo do atributo Criação, aciona o método construtor da entidade dinâmica a ser criada, colocando-o como um evento na lista de eventos do programa executivo da simulação. O número de vezes que o construtor será chamado vai depender do valor do atributo Qte Média.

AlterarCor: altera o conteúdo do atributo Cor definido para o estado.

AlterarTempo: altera o valor do atributo Tempo, podendo mudar a função de distribuição.

AlterarPreC: este método é responsável pela alteração da expressão de precondição.

AlterarPosC: este método é responsável pela alteração da expressão de pós-condição.

AlterarListaEv-ProxEst: este método é responsável pela alteração do atributo ListaEv-ProxEst, modificando um ou mais de seus elementos (eventos de transição e próximos estados).

DefinirGeração: este método define e altera a função de distribuição responsável por gerar o tempo de permanência da entidade estática no estado.

5.2.12 CLASSE VARIÁVEL

Esta classe é responsável pela definição das variáveis globais do modelo de animação. Variáveis globais são utilizadas para o estabelecimento de pré e pós-condições.

- **Atributos**

Nome: contém o nome lógico da variável.

ID: número inteiro atribuído automaticamente pelo ambiente que identifica unicamente a variável.

Tipo: tipo de dado que a variável irá armazenar (exemplos: inteiro, booleano, real, string de caracteres, vetor e matriz)

Valor Inicial: valor dado à variável no início da animação do modelo.

Valor: valor atual da variável, que vai sendo alterado durante a animação do modelo.

Para a manipulação das variáveis do modelo, deve ser utilizado os recursos da linguagem de simulação adotada para a implementação.

- **Métodos**

Foram definidos alguns métodos responsáveis pela manipulação dos atributos do objeto variável. Estes atributos são preenchidos inicialmente pelo seu construtor, mas podem ser alterados posteriormente. Os métodos definidos a seguir possibilitam estas mudanças.

AlterarTipo: altera o valor do Tipo da variável.

AlterarValorInicial: altera o Valor Inicial dado à variável no início da animação.

MudarValor: altera o atributo Valor da variável durante a animação do modelo.

5.2.13 CLASSE COORDENADOR

O objeto da classe Coordenador é o responsável por coordenar todo o processo de criação e animação do modelo no ambiente, disponibilizando para o modelador as funções necessárias para a manipulação do modelo de animação e o acionamento dos mecanismos da simulação, incluindo o programa executivo. Estes mecanismos foram descritos na Seção 5.1.

Existirá um único objeto da classe Coordenador durante todo o processo de criação e experimentação do modelo de animação. O objeto Coordenador funciona como um gerente, armazenando as condições iniciais estabelecidas pelo modelador para a animação e controlando o processo de interação do modelador com o modelo, utilizando os mecanismos da linguagem de simulação de maneira transparente.

Os objetos representando as entidades participantes do modelo interagem com o objeto coordenador durante a animação por meio da troca de mensagens. Esta troca de mensagens inclui, por exemplo, o envio, por um objeto do modelo, de um evento e o seu tempo de duração para ser colocado na lista de eventos programados do programa executivo da simulação.

A lista de eventos é montada no início da animação a partir do modelo gráfico criado. Quando o modelador faz interações durante a animação, esta lista de eventos é atualizada a partir de mensagens enviadas ao objeto coordenador pelos objetos componentes do modelo. A organização desta lista em uma ordem cronológica é de responsabilidade do objeto coordenador, que aciona a função de escalonamento dos eventos da linguagem de simulação utilizada, com a utilização do método **EscalonarEventos.** Deste modo é criada e mantida uma lista de eventos e uma lista de eventos iminentes, que é utilizada pelo programa executivo da simulação. A freqüência de atualização desta lista depende das interações do modelador com o modelo durante o processo de animação.

A seguir são definidos os principais atributos e métodos da classe Coordenador. A maioria das funções estabelecidas nos métodos já estão disponíveis na maioria da linguagens de simulação.

- **Atributos**

Tempo da Rodada: estabelece o tempo de duração de uma rodada de animação do modelo. Caso o modelador não queira definir um tempo para a rodada, este atributo não é preenchido.

Número de Rodadas: estabelece o número de vezes que o modelo vai ser rodado, com a duração estabelecida no atributo Tempo de Rodada.

Variáveis de Coleta: contém uma lista com os nomes das variáveis do modelo que armazenam dados das coletas estatísticas. Estes variáveis podem armazenar valores do número de entidades na fila, tempo médio que uma entidade estática fica num determinado estado, número de ocorrências de um determinado evento aleatório, etc.

- **Métodos**

DefinirNroRodadas: método que define o atributo Número de Rodadas do modelo de animação.

DefinirTempoRodada: método que define o tempo de duração de uma rodada do modelo de animação, armazenado no atributo Tempo da Rodada.

DefinirVariáveisColeta: este método permite que o modelador defina variáveis que armazenam valores que são utilizados para posterior análise estatística.

IniciarAnimação: método que inicia a animação do modelo, baseado nas condições iniciais estabelecidas.

TerminarAnimação: método que interrompe o processo de animação do modelo. Este método pode ser acionado quando não foi definido um tempo fixo para a rodada de animação, ou quando, mesmo que o tempo de rodada tenha sido definido, o modelador desejar interromper a animação do modelo.

PausaAnimação: método que provoca uma pausa no relógio de simulação, congelando os quadros de animação. É útil para análise de instantâneos do modelo. Depois que uma pausa foi solicitada pelo modelador, o modelo retoma sua animação a partir do ponto de parada.

Acelerar: este método aumenta a velocidade de exibição dos quadros de uma animação.

Desacelerar: este método diminui a velocidade de exibição dos quadros de uma animação.

DescerNível: este método, a partir de uma ou mais entidades selecionadas, cria uma nova janela gráfica para exibir um nível de decomposição destas entidades, permitindo que o modelador crie um submodelo com novas entidades estáticas e dinâmicas (processo *top-down*), ou visualize subníveis já existentes, podendo animá-los.

SubirNível: este método, a partir de algumas entidades do modelo selecionadas, cria uma nova janela gráfica para exibir a entidade que originou a decomposição. Se esta entidade não existir ela pode ser criada (processo *bottom-up*). O modelador pode visualizar a animação resultante.

AlterarParâmetros: este método permite que o modelador faça alterações de alguns parâmetros definidos para o modelo de animação, com o objetivo de realizar experimentações. Algumas destas alterações incluem habilitar e desabilitar uma entidade, alterar funções de distribuição e tempos médios, provocar a ocorrência de um evento aleatório. Estas alterações se viabilizam por meio de chamadas aos métodos específicos de cada objeto.

VerificarConsistência: este método faz uma verificação geral do modelo criado, abrangendo os seguintes pontos:

- Expressões utilizadas nas definições das condições;
- Uso correto dos rótulos C e S;
- Definição das entidades estáticas e dinâmicas e todos os seus atributos;

- Definição das filas e todos os seus atributos; e
- Definição dos eventos e variáveis e todos os seus atributos;
- Condições iniciais para o início da animação tais como tempo de uma rodada e número de rodadas.

VisualizarResultadosEstatísticos: método que permite a visualização dos resultados das coletas estatísticas, por meio de tabelas, histogramas, gráficos de barras, gráficos de linhas, gráficos tipo torta, etc. Estes recursos gráficos para apoio na análise estatística estão disponíveis na maioria das linguagens de simulação.

EscalonarEventos: este método aciona a função da linguagem de simulação, responsável por criar e manter atualizada a lista de eventos. Esta lista é utilizada pelo programa executivo da simulação durante a animação do modelo e contém a ordem cronológica de ocorrência dos eventos no tempo simulado.

5.3 DEFINIÇÃO DE PRÉ E PÓS-CONDIÇÕES

As pré e pós-condições definidas para o modelo, incluindo a condição de saída de um objeto da classe Fila, expressam parte de sua lógica e devem ser estabelecidas utilizando os recursos disponíveis da linguagem de simulação selecionada para a implementação do ambiente, não limitando o modelador no processo de criação.

As linguagens de simulação, assim como qualquer linguagem de programação, disponibilizam, no mínimo, recursos que tornam possível a criação de condições em termos de expressões aritméticas de atribuição, sentenças lógicas e expressões condicionais. Por exemplo, a condição de saída de uma entidade dinâmica da fila pode ser alterar o valor de uma variável global **C** do modelo, somando 1 ao seu valor anterior. A condição de saída da fila seria **C=C+1.**

O ambiente deve fornecer uma interface gráfica para facilitar a construção das sentenças que expressam as condições. Nesta interface as listas com os nomes de todas as entidades estáticas e dinâmicas do modelo, as filas e as variáveis globais definidas devem estar disponíveis para o modelador. Desta forma, o modelador constrói a sua expressão para definir uma condição, escolhendo um elemento da lista, sem o risco de

utilizar uma variável que não foi definida no modelo ou um operador inexistente, ocasionando inconsistências.

Uma extensão deste trabalho para a definição das pré e pós-condições é comentada no Capítulo 7. A Seção seguinte ilustra alguns problemas de inconsistências que podem ocorrer no modelo, a partir das informações fornecidas pelo modelador.

5.4 INCONSISTÊNCIAS INTRODUZIDAS PELO MODELADOR

Devido a flexibilidade do ambiente proposto, várias informações introduzidas pelo modelador durante o processo de modelagem, principalmente às relacionadas ao tempo, podem gerar inconsistências no modelo.

O modelador, durante a criação do modelo de animação, tem a liberdade de atribuir valores para as variáveis de tempo, abrangendo o tempo de permanência das entidades estáticas em um estado, a duração de alguns eventos e os tempos de movimento das entidades dinâmicas. Esta atribuição pode envolver tanto valores aleatórios como valores predeterminados. Quando os valores são gerados aleatoriamente, as inconsistências podem ocorrer mais freqüentemente.

Uma inconsistência pode ser introduzida quando o modelador define um estado de uma entidade estática, no qual ocorre a criação de entidades dinâmicas. O atributo Criação do estado conterá o nome e a quantidade de entidades dinâmicas a serem criadas e o tempo necessário para a processo de criação (fixo, periódico ou aleatório). O tempo atribuído para a entidade estática permanecer no estado não pode ser menor do que o tempo necessário para ela gerar as entidades dinâmicas definidas para aquele estado.

Outra inconsistência pode surgir quando é definido um estado específico para uma entidade estática criar entidades dinâmicas e o tempo que ela permanece no processo de criação não é compatível com o tempo atribuído para a permanência da entidade estática no estado. Por exemplo, ao tempo de permanência no estado pode ter sido atribuído o valor 10 segundos, e ao evento de criação pode ter sido atribuído uma geração periódica em intervalos médios de 1 segundo, por cinco vezes, resultando num tempo de criação aproximado de 5 segundos, incompatível com o tempo de permanência total da entidade no estado.

As inconsistências do modelo, tanto referentes ao tempo como aos outros aspectos envolvidos na criação do modelo de animação, não foram analisadas exaustivamente. Esta análise é necessária para a efetiva implementação do ambiente e é sugerida como uma extensão do trabalho, como comentado na Seção 7.3 do Capítulo 7.

O Capítulo seguinte apresenta um exemplo da criação do modelo de animação para um dos casos estudados no Capítulo 2, o Sistema de Gerência de Tráfego de Trens. O objetivo deste Capítulo é exemplificar como o modelo de animação é criado com a técnica apresentada e com as facilidades oferecidas pelo ambiente de modelagem.

Alguns refinamentos do modelo são feitos para mostrar a possibilidade de identificação de alguns dos principais objetos de software do sistema. É também ilustrado como a infra-estrutura orientada a objetos apresentada neste Capítulo possibilita a geração de uma animação baseada em simulação, por meio da apresentação de um escalonamento parcial de eventos do modelo criado.

CAPÍTULO 6

O MODELO DO SISTEMA DE GERÊNCIA DE TRÁFEGO DE TRENS

O estudo de caso Sistema de Gerência de Tráfego de Trens, apresentado no Capítulo 2, é analisado novamente neste Capítulo, com o objetivo de exemplificar o uso do simbolismo e do ambiente proposto apresentado no Capítulo 4, e também da estrutura orientada a objetos apresentada no Capítulo 5, para a criação do modelo de animação.

Inicialmente, é assumido que foi feita a identificação das principais entidades estáticas e dinâmicas pelo modelador, tendo como base a descrição do sistema apresentada no Capítulo 2, Seção 2.5.3. Após esta primeira identificação, chamada de nível 1 de abstração, são criados pelo modelador outros dois níveis do modelo, com o objetivo de ilustrar o processo de decomposição de algumas entidades. A partir do último nível de decomposição de parte do sistema, foi possível delinear alguns dos principais objetos de software.

Os níveis de abstração são identificados como níveis 1, 2 e 3, cobrindo níveis de detalhes do modelo em ordem crescente de refinamento. Os termos nível de abstração e nível do modelo são utilizados alternativamente, com o mesmo significado.

As seções seguintes apresentam as entidades estáticas e dinâmicas e as filas identificadas em cada nível de abstração, e o modelo gráfico correspondente.

6.1 IDENTIFICAÇÃO DAS ENTIDADES – NÍVEL 1

As entidades estáticas e dinâmicas identificadas no nível 1 são mostradas graficamente na Figura 6.1. O modelo gráfico deste Figura contém outros elementos gráficos que não pertencem ao simbolismo adotado, mas fazem parte de um conjunto de ferramentas gráficas que são utilizadas para complementar o modelo gráfico, conforme comentado na Seção 5.2.5 do Capítulo 5. Nesta Figura estes elementos são utilizados para

representar o trilho das ferrovias e os *links* de microondas entre as Interfaces *Wayside* e o Controlador Terminal-Terra. A legenda que identifica as entidades dinâmicas da Figura 6.1 com cores e números, é mostrada na Figura 6.2.

6.1.1 DEFINIÇÃO DAS ENTIDADES ESTÁTICAS DO NÍVEL 1

Segundo a descrição do sistema, existem vários Centros de Despacho que se comunicam trocando informações entre si. Para a simplificação deste modelo inicial foi considerado somente uma entidade estática **Centro de Despacho** e uma entidade estática **Outros Centros de Despacho,** e a comunicação entre eles só foi representada em uma direção.

As outras entidades estáticas identificadas foram: Sistema de Análise e Relatórios, Sensores do Trem, Sistema de Visualização, Unidade de Gerência de Dados, Satélite, Controlador Terminal-Terra, Sistema de Controle da Rede, Centro de Despacho, Interface *Wayside, Switch* e *Transponder*.

Neste primeiro nível do modelo, os únicos atributos das entidades estáticas que receberam valores diferentes foram Nome, ID e Lista de Estados. O restante dos atributos receberam os mesmos valores para todas as entidades identificadas, conforme é mostrado a seguir.

- Cor: transparente;
- Figura: retângulo;
- Status: habilitada;
- Lista de Atributos: vazia;
- Lista de Métodos: vazia;
- Lista de Entidades Estáticas: vazia.

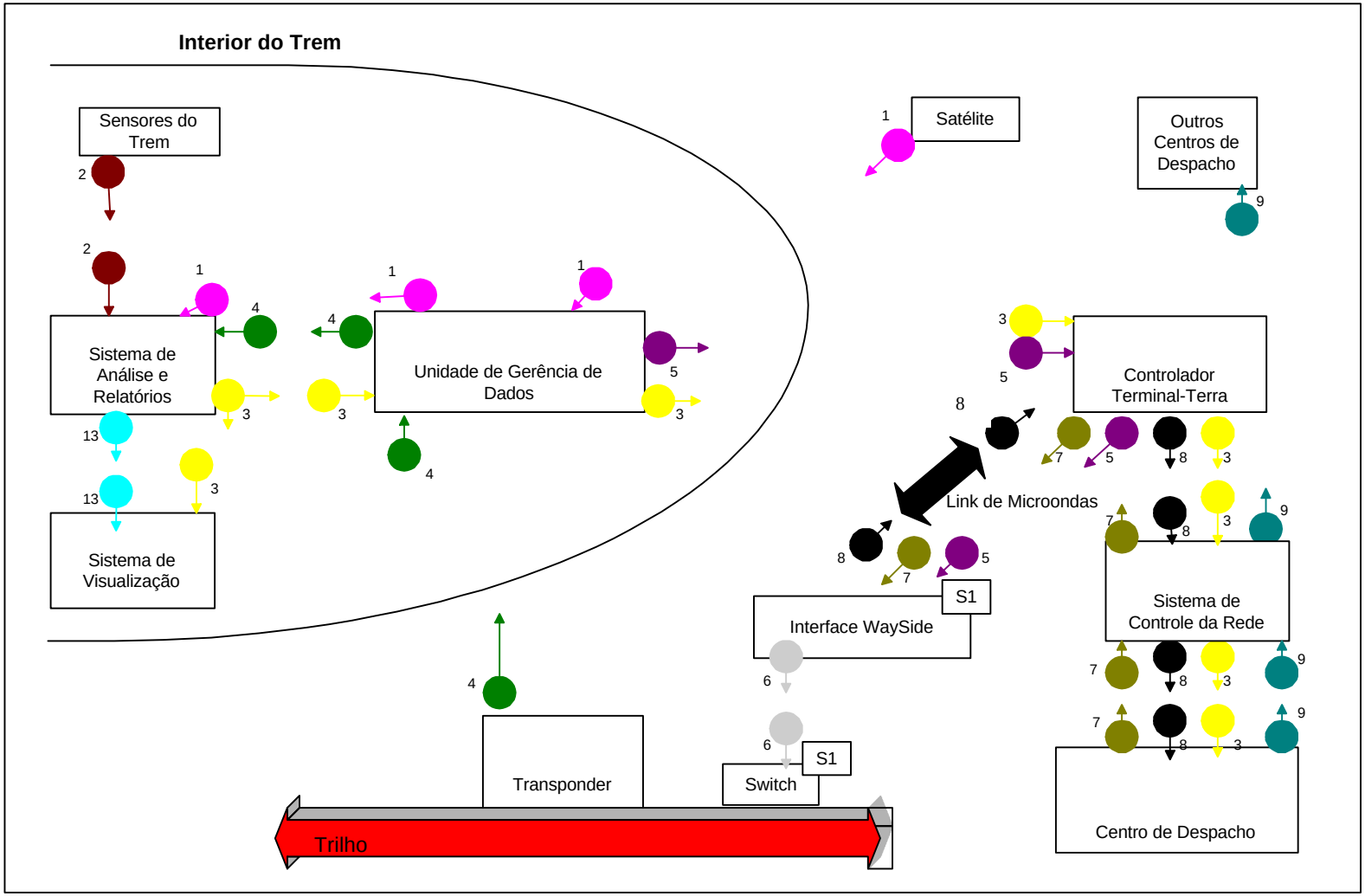

Fig. 6.1 - Modelo do sistema de Gerência de Tráfego de Trens: visão geral- nível 1.

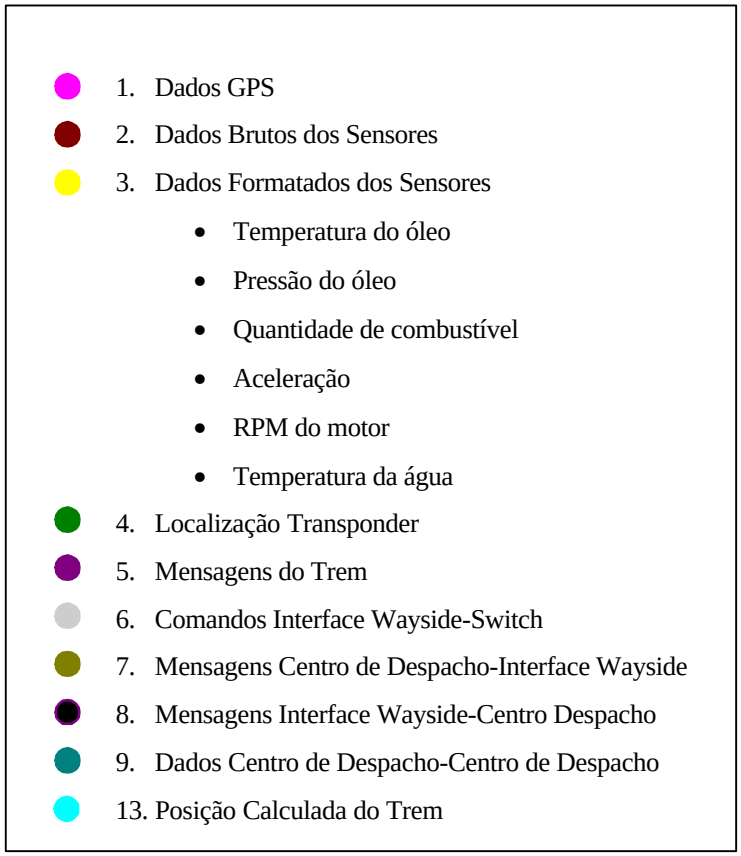

Fig. 6.2 - Legenda do modelo do sistema de Gerência de Tráfego de Trens - nível 1.

Durante o processo de criação de uma entidade estática, o modelador através das janelas gráficas do ambiente, fornece as informações sobre todos os seus possíveis estados e os eventos que provocam as transições de estado, definindo também uma cor e um tempo de permanência para cada estado.

Para facilitar a compreensão destas entidades estáticas, elas foram apresentadas utilizando diagramas de transição de estados, segundo a notação proposta por Ward e Mellor (1985). Esta notação foi a utilizada por ser parte de uma ferramenta CASE disponível para a confecção deste trabalho, mas outras poderiam ter sido utilizadas, como por exemplo, a proposta pela *Unified Modeling Language* (UML, 1997). Foram acrescentados aos diagramas informações de tempo e os estados diferentes de desocupado foram pintados com tons diferentes de cinza.

De acordo com a notação utilizada, o rótulo **C** no diagrama representa o evento de transição e o rótulo **A** a ação associada à transição. O acionamento do método Ativar da entidade estática é o evento que marca o início do tempo de permanência da entidade

em um determinado estado, representado nesta notação pela ação associada ao método Chegada.

O acionamento do método Mover de uma entidade dinâmica, normalmente está associado ao término do tempo de permanência da entidade em algum estado e, é representado pela ação associada ao método Desativar. Aos métodos Chegada e Mover foram associados os números que representam as entidades dinâmicas no modelo, referenciadas por eles.

As Figuras 6. 3 a 6.14 representam os diagramas de transição de estado das entidades estáticas componentes do modelo gráfico, Figura 6.1.

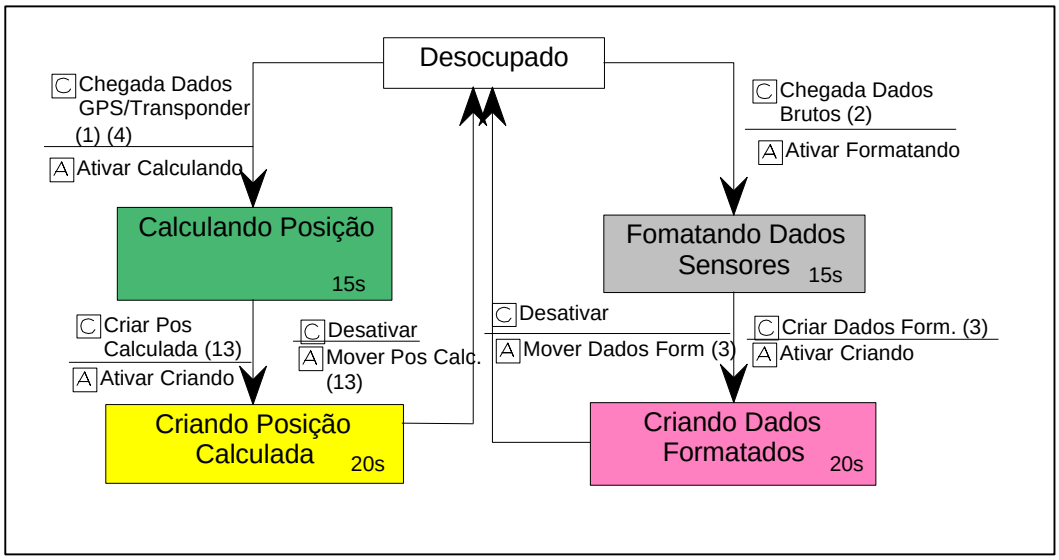

Fig. 6.3 - Diagrama de transição de estado da entidade Sistema de Análise e Relatórios.

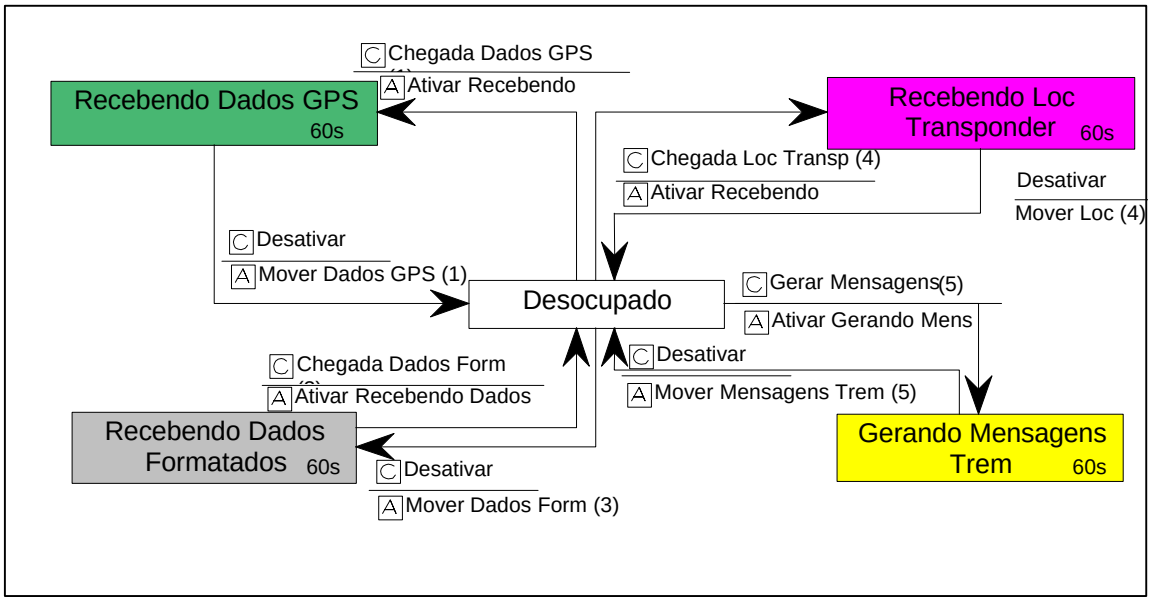

Fig. 6.4 - Diagrama de transição de estado da entidade Unidade de Gerência de Dados.

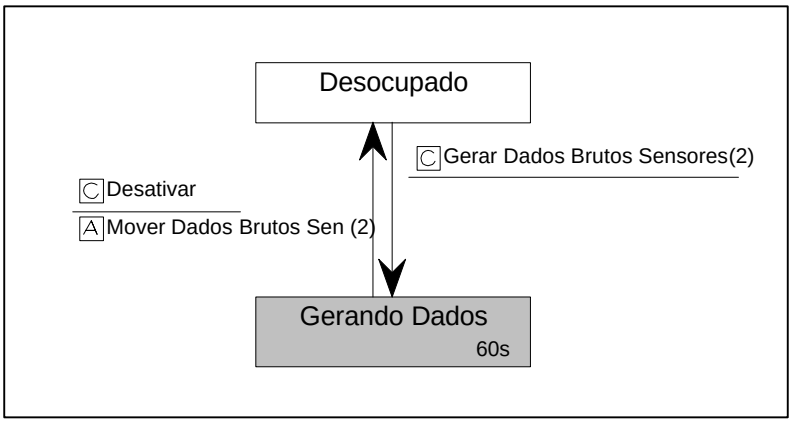

Fig. 6.5 - Diagrama de transição de estado da entidade Sensores do Trem.

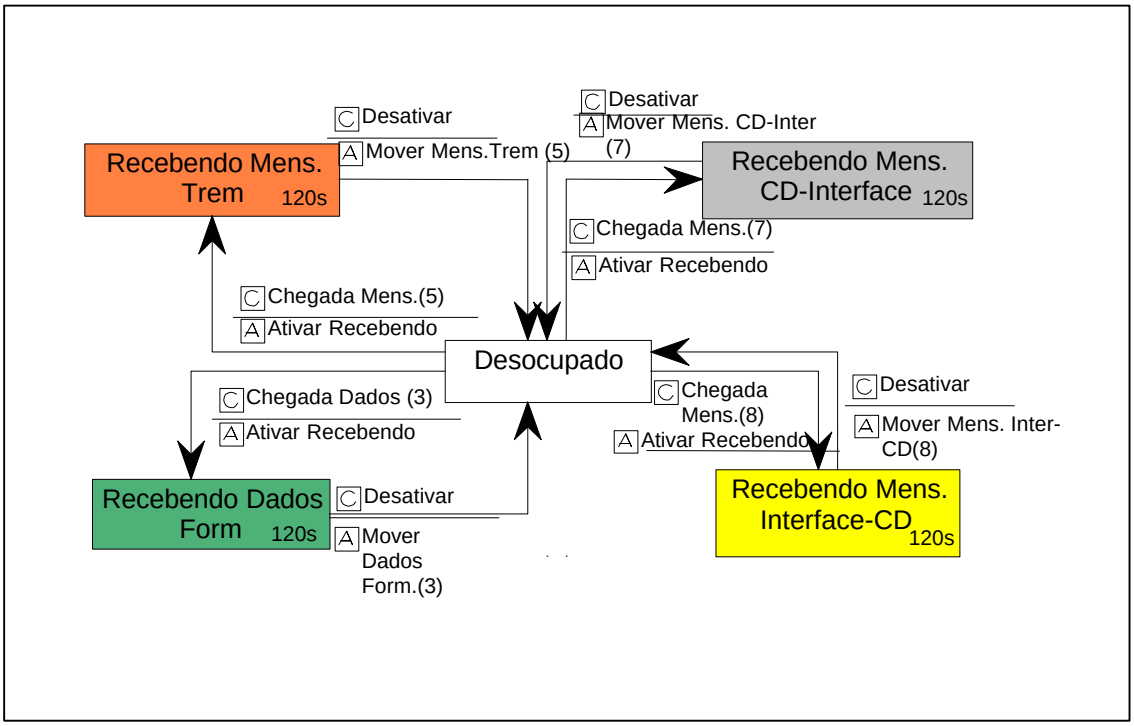

Fig. 6.6 - Diagrama de transição de estado da entidade Controlador Terminal-Terra.

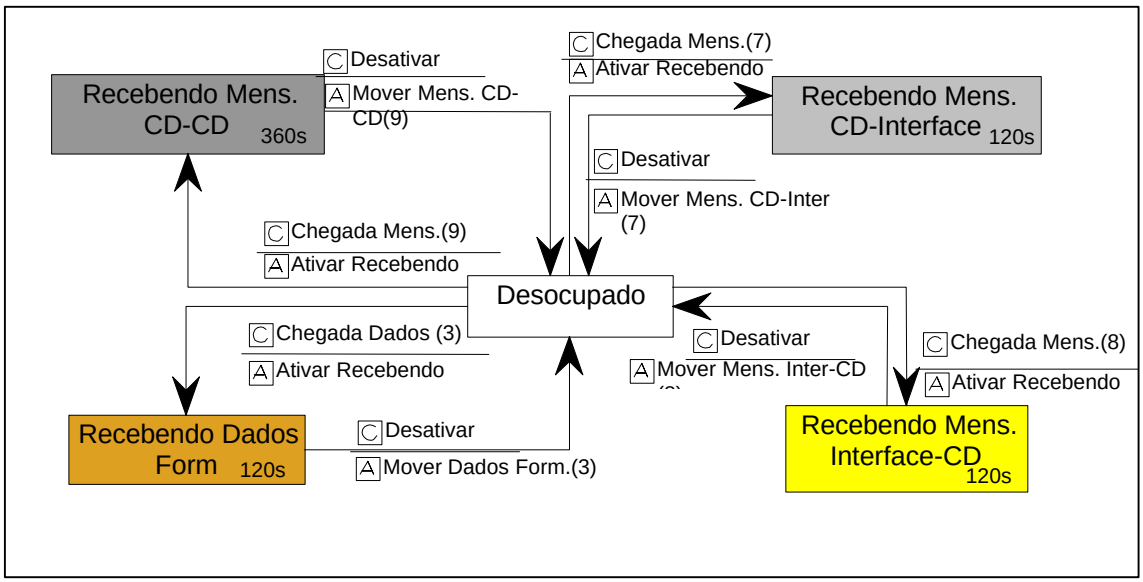

Fig. 6.7 - Diagrama de transição de estado da entidade Sistema de Controle da Rede.

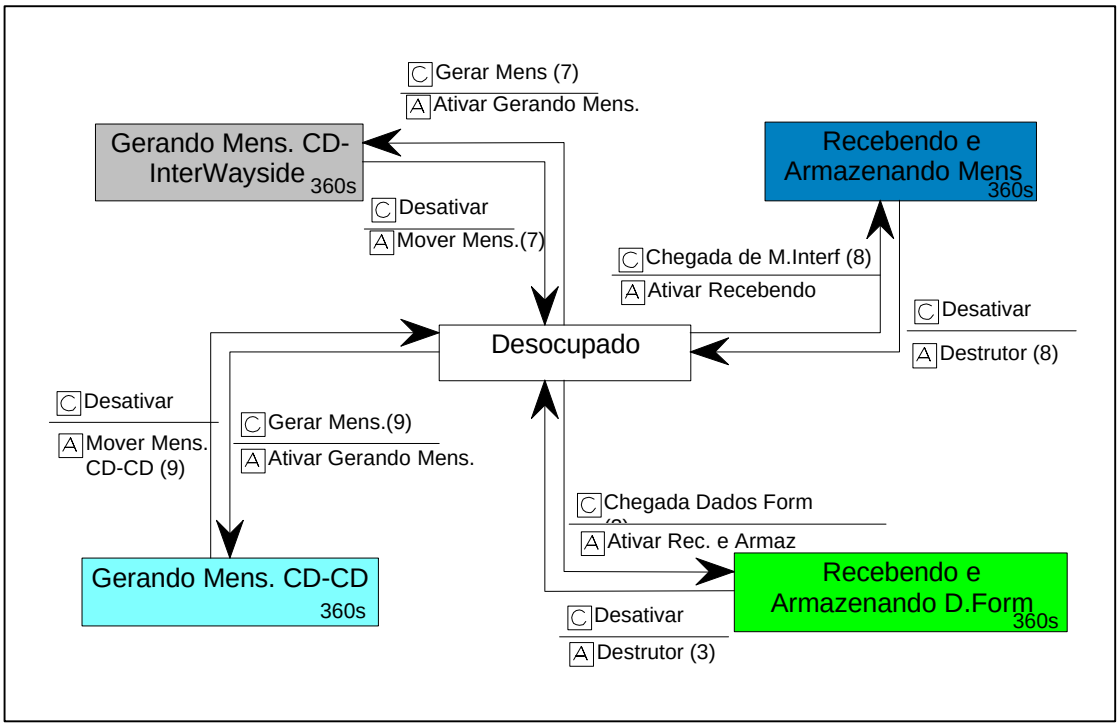

Fig. 6.8 - Diagrama de transição de estado da entidade Centro de Despacho.

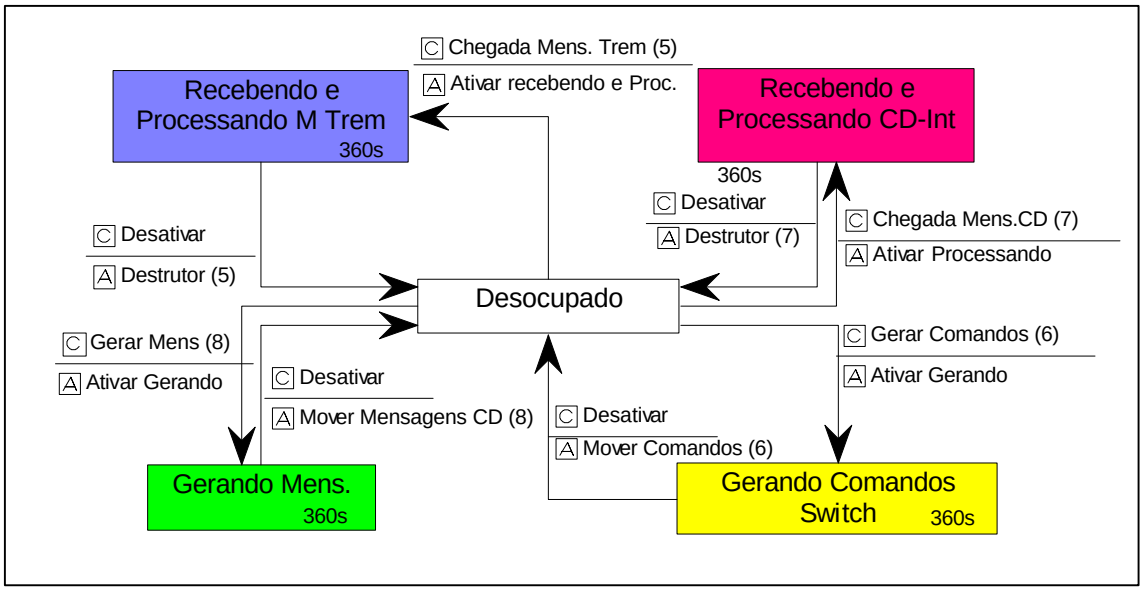

Fig. 6.9 - Diagrama de transição de estado da entidade Interface *Wayside*.

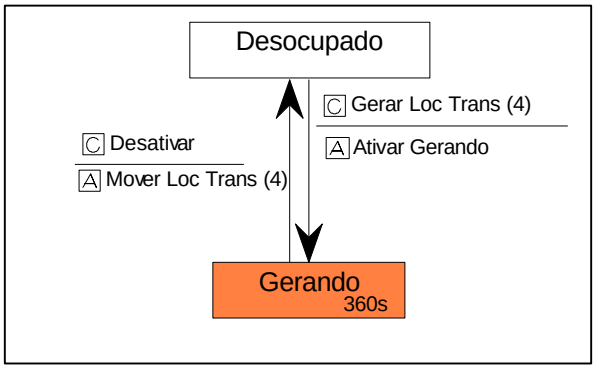

Fig. 6.10 - Diagrama de transição de estado da entidade *Transponder*.

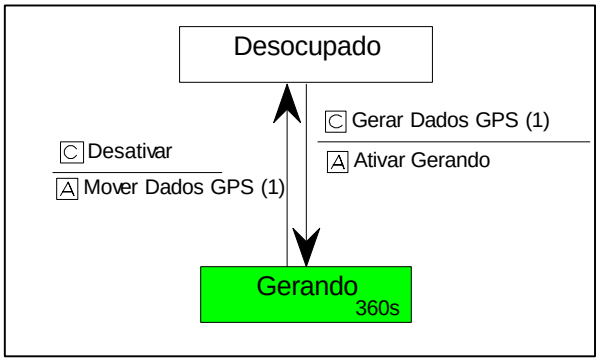

Fig. 6.11 - Diagrama de transição de estado da entidade Satélite.

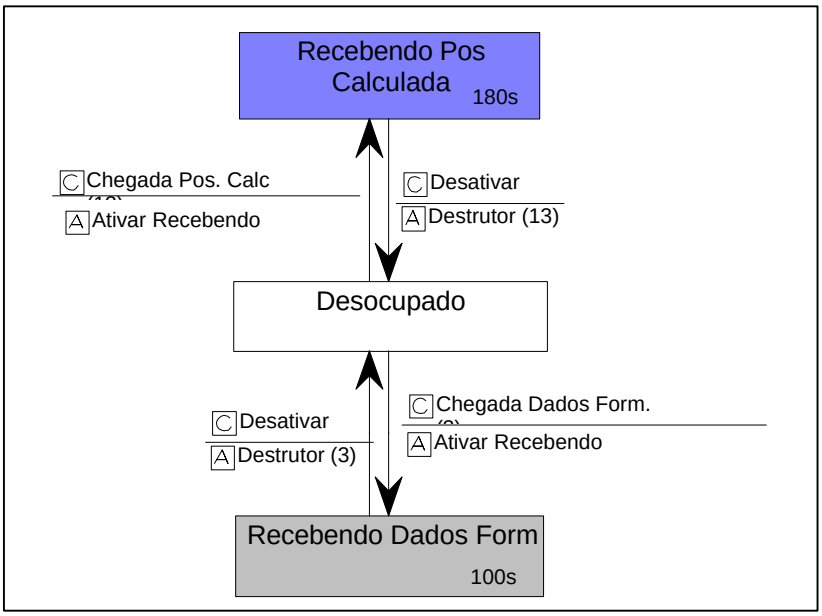

Fig. 6.12 - Diagrama de transição de estado da entidade Sistema de Visualização.

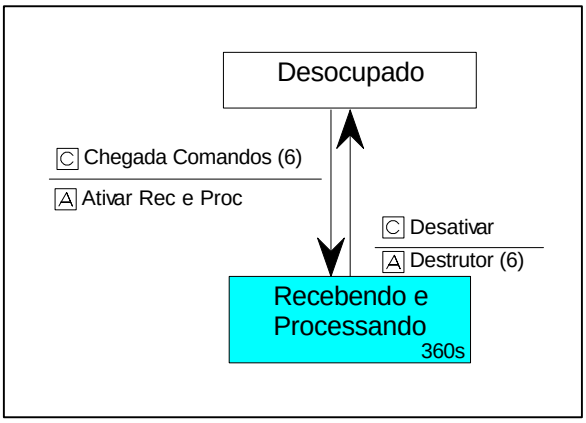

Fig. 6.13 - Diagrama de transição de estado da entidade *Switch*.

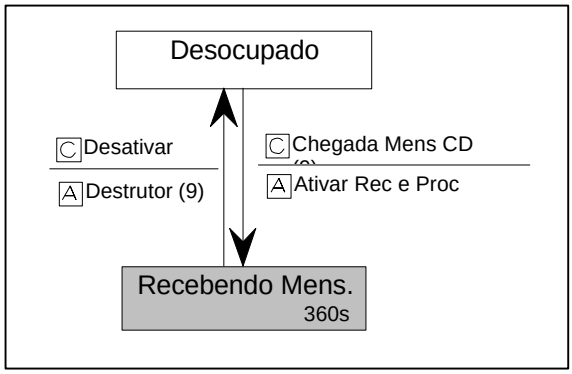

Fig. 6.14- Diagrama de transição de estado da entidade Outros Centros de Despacho.

6.1.2 DEFINIÇÃO DAS ENTIDADES DINÂMICAS DO NÍVEL 1

As entidades dinâmicas identificadas neste nível constam na Figura 6.2. A seguir, elas são apresentadas e suas respectivas listas de percursos são geradas pelo ambiente, a partir do modelo gráfico criado, onde o modelador acrescentou as informações de tempo de percurso.

Analogamente às entidades estáticas, somente alguns atributos das entidades dinâmicas receberam valores diferentes: Nome, ID, Cor e Lista de Percursos. O restante dos atributos receberam os valores mostrados a seguir.

- Lista de Atributos: vazia;
- Figura: círculo;
- Lista de Métodos: vazia;
- Lista de Entidades Dinâmicas: vazia.

Entidade Dinâmica *Dados GPS*

Atributos
Nome: Dados GPS
ID: 1
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Satélite	Unid. Gerência de Dados	60 s
2	Unid. Gerência de Dados	Sistema de Análise e Relatórios	30 s

Entidade Dinâmica *Dados Brutos Sensores*

Atributos
Nome: Dados Brutos Sensores
ID: 2
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sensores do Trem	Sistema de Análise e Relatórios	30 s

Entidade Dinâmica *Dados Formatados*

Atributos
Nome: Dados Formatados
ID: 3
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sistema de Análise e Relatórios	Sistema de Visualização	20 s
2	Sistema de Análise e Relatórios	Unid. Gerência de Dados	40 s
2	Unid. Gerência de Dados	Controlador Terminal-Terra	30 s
3	Controlador Terminal-Terra	Sistema Controle Rede	40 s
4	Sistema Controle Rede	Centro de Despacho	60 s

Entidade Dinâmica Localização *Transponder*

Atributos
Nome: Localização *Transponder*
ID: 4
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	*Transponder*	Unid. Gerência de Dados	60 s
2	Unid. Gerência de Dados	Sistema de Análise e Relatórios	30 s

Entidade Dinâmica *Mensagens do Trem*

Nome: Mensagens do Trem
ID: 5
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Unidade de Gerência de Dados	Controlador Terminal-Terra	60 s
2	Controlador Terminal Terra	Sistema de Controle da Rede	30 s
3	Sistema de Controle da Rede	Centro de Despacho	60 s

Entidade Dinâmica *Comandos Interface Wayside-Switch*

Atributos
Nome: Comandos Interface *Wayside-Switch*
ID: 6
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Interface *Wayside*	*Switch*	60 s

Entidade Dinâmica *Mensagens Centro de Despacho Interface Wayside*

Atributos
Nome: Mensagens Centro de Despacho Interface *Wayside*
ID: 7
Cor:

Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Centro de Despacho	Sistema de Controle da Rede	60 s
2	Sist. de Controle da Rede	Controlador Terminal Terra	30 s
3	Controlador Terminal-Terra	Interface *Wayside*	60 s

Entidade Dinâmica *Mensagens Interface Wayside Centro de Despacho*

Atributos
Nome: Mensagens Interface *Wayside* Centro de Despacho
ID: 8
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Interface *Wayside*	Controlador Terminal Terra	60 s
2	Controlador Terminal-Terra	Sistema de Controle da Rede	30 s
3	Sist. de Controle da Rede	Centro de Despacho	60 s

Entidade Dinâmica *Mensagens Centro de Despacho Centro de Despacho*

Atributos
Nome: Mensagens Centro de Despacho Centro de Despacho
ID: 9
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Centro de Despacho	Sistema de Controle da Rede	60 s
2	Sistema de Controle da Rede	Outros Centro de Despacho	30 s

Entidade Dinâmica *Posição Calculada do Trem*

Nome: Posição Calculada do Trem
ID: 13
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sistema de Análise e Relatórios	Sistema de Visualização	40 s

Eventos Agendados:

Nome: Limites Extrapolados (evento inicial)
ID: 34
Geração: periódico 300 em 300 s.

6.2 IDENTIFICAÇÃO DAS ENTIDADES – NÍVEL 2

No nível 2 do modelo, as entidades estáticas Sistema de Análise e Relatórios, Unidade de Gerência de Dados e Sistema de Energia, foram decompostas em novas entidades estáticas e dinâmicas, Figura 6.15, que são descritas nas seções seguintes. As alterações sofridas nas entidades já existentes também são relatadas.

6.2.1 DEFINIÇÃO DAS ENTIDADES ESTÁTICAS DO NÍVEL 2

A entidade Sistema de Análise e Relatórios passou a englobar as novas entidades estáticas Sensores e Subsistema Posição do Trem. A entidade Sensores é abstrata, diferente da entidade Sensores do Trem identificada no nível 1. Esta última representa os sensores físicos instalados no Trem.

A entidade Sensores recebe os dados brutos dos sensores físicos, formata estes dados e verifica se cada um deles está dentro do limite estabelecido para o sensor. Caso haja violação de alguns dos limites, a entidade emite alarmes ou advertências. A entidade Subsistema Posição do Trem, a partir de valores recebidos do *transponder* e do GPS e da contagem das revoluções da roda, calcula a posição do trem.

A entidade Unidade de Gerência de Dados passou a englobar as novas entidades estáticas Armazenador de Dados e Recuperador de Dados. Esta última é a responsável pelo envio de informações da Unidade de Gerência de Dados para as outras entidades do sistema. A entidade Armazenador de Dados é a responsável por tornar persistentes as entidades dinâmicas que chegam na Unidade de Gerência de Dados.

A entidade Sistema de Visualização passou a receber as novas entidades dinâmicas Alarmes/Advertências, Valores Aceleração/Freios e Posição Calculada do Trem, resultando em alterações na sua lista de estados.

O Sistema de Energia, que no nível 1 não foi definido, passa a existir neste nível, englobando uma outra entidade estática chamada Avaliador de Desempenho. Esta

última é a responsável pelo cálculo dos melhores valores de aceleração e freios, baseado nos dados vindos da Unidade de Gerência de Dados.

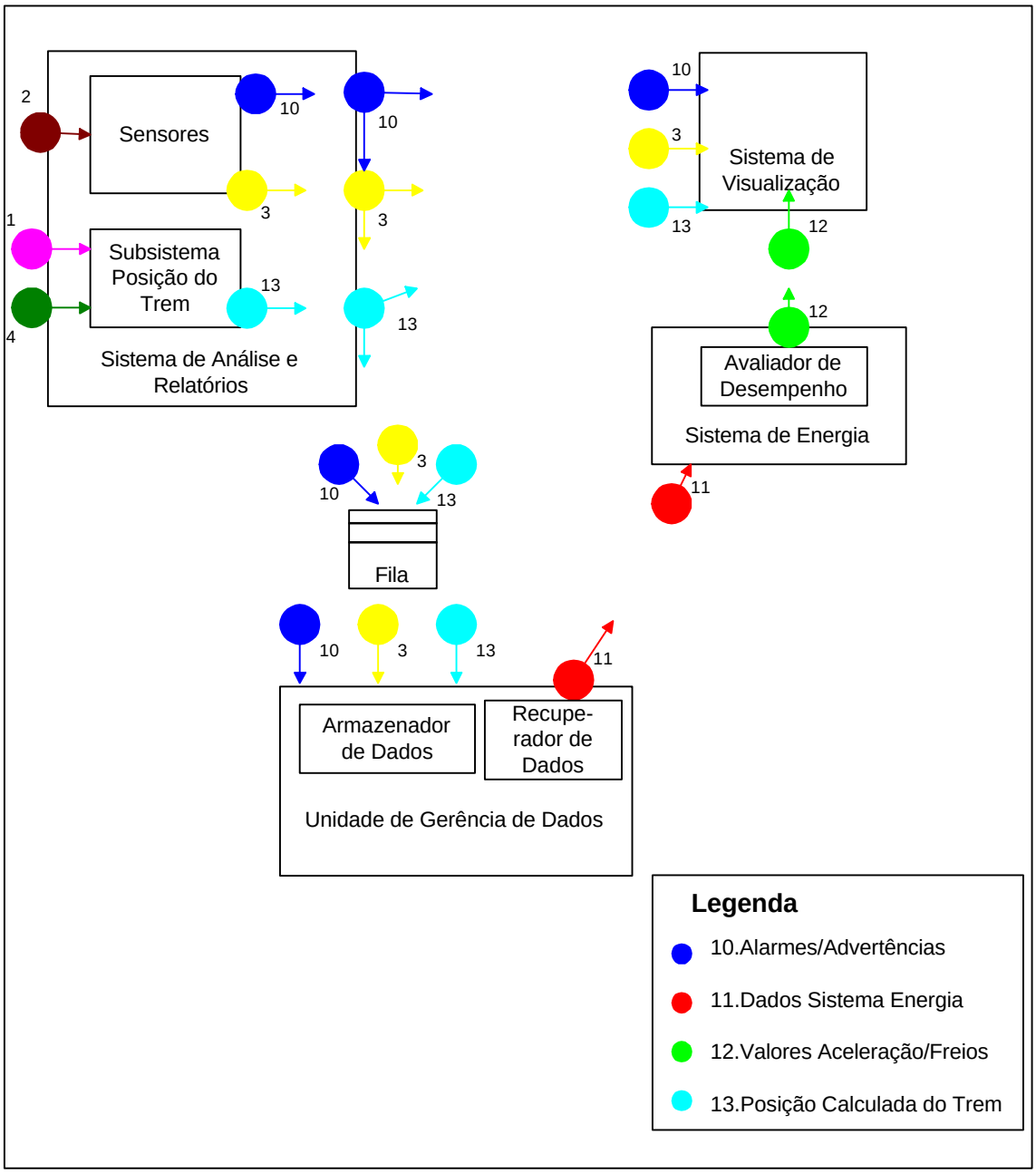

Fig. 6.15 - Sistema de Gerência de Tráfego de Trens- nível 2: interior do trem.

Foi colocada uma fila diante da entidade Unidade de Gerência de Dados para controlar o acesso à entidade Armazenador de Dados. Esta fila armazenará entidades Alarmes/Advertências, Dados Formatados e Localização *Transponder*.

O modelador, utilizando as janelas gráficas do ambiente, define as novas entidades estáticas fornecendo as informações sobre todos os seus possíveis estados e os eventos que provocam as transições de estado, atribuindo uma cor e um tempo de permanência para cada estado.

As novas entidades estáticas e a redefinição das já existentes no nível acima do modelo, são apresentadas utilizando diagramas de transição de estados, com a informação de tempo e cor, conforme mostram as Figuras de 6.16 à 6.21. Os atributos que receberam os mesmos valores para todas as entidades estáticas identificadas neste nível são:

- Cor: transparente;
- Figura: retângulo;
- Status: habilitada;
- Lista de Atributos: vazia;
- Lista de Métodos: vazia;
- Lista de Entidades Estáticas: vazia.

A fila criada é definida em seguida:

Entidade *Fila Acesso BD*

Nome: Fila Acesso BD.
ID: 28.
Ordenação: FIFO.
Prioridade: nenhuma.
Evento de entrada: Chegada Dados Formatados **ou** Chegada da Localização Trem **ou** Chegada Alarmes/Advertências.
Condição de saída: Entidade Armazenador de Dados desocupada.
Evento de Saída: nenhum.

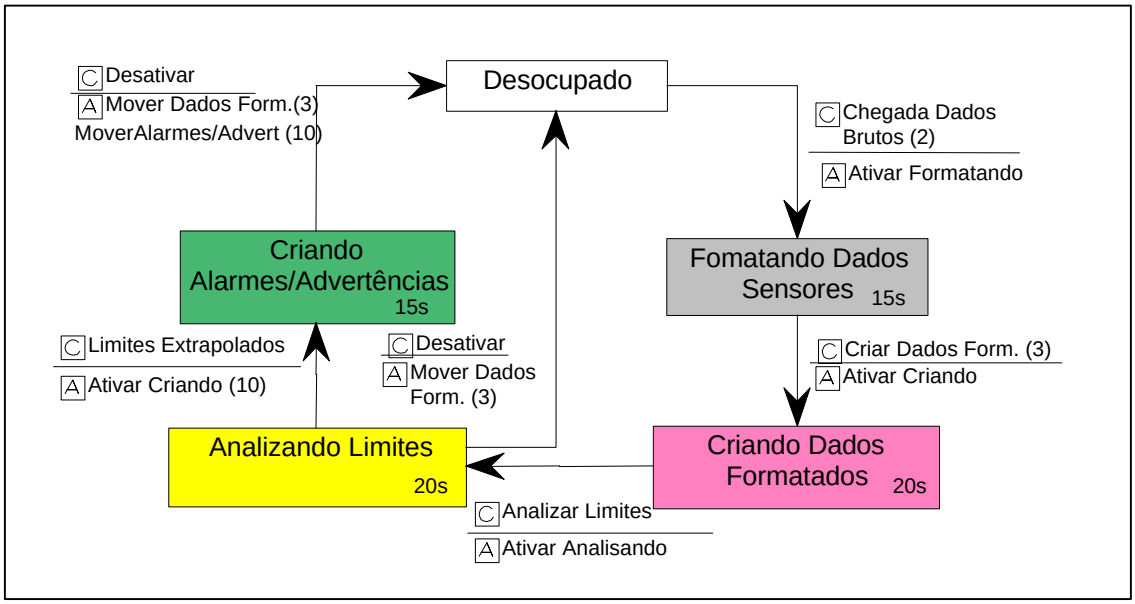

Fig. 6.16 - Diagrama de transição de estado da entidade Sensores.

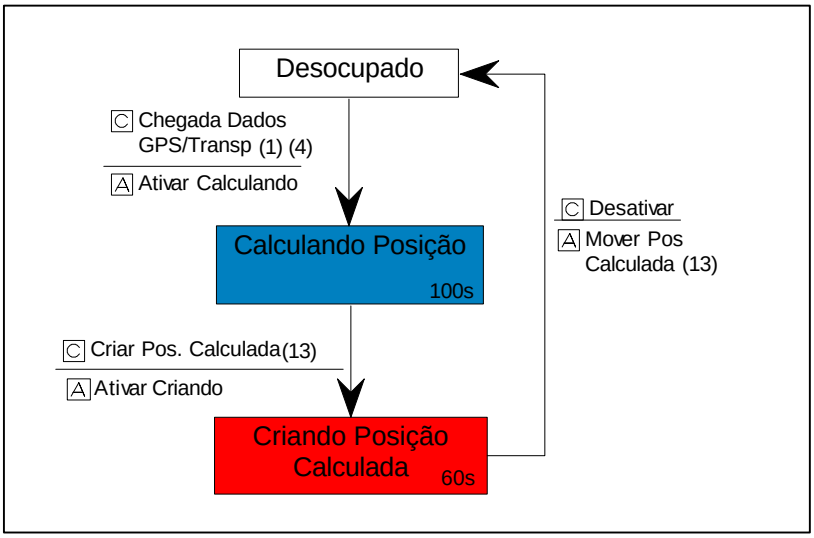

Fig. 6.17 - Diagrama de transição de estado da entidade Subsistema Posição do Trem.

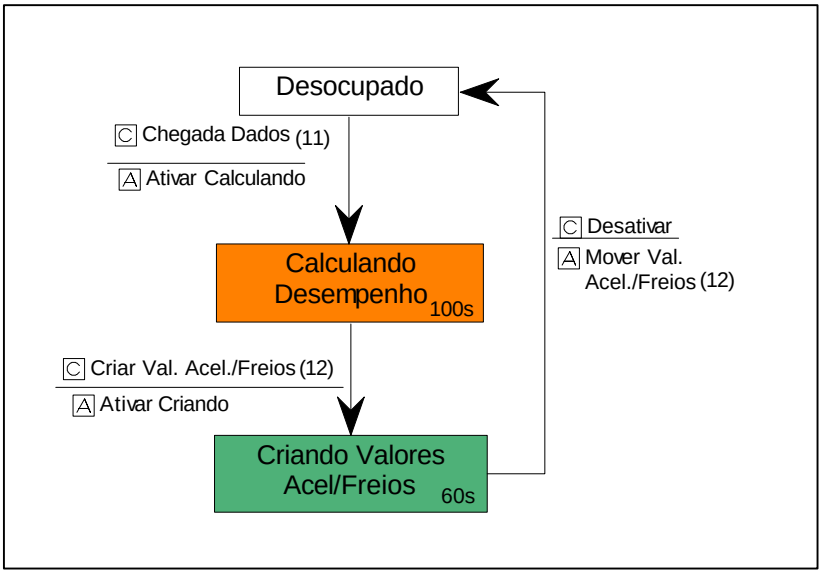

Fig. 6.18 - Diagrama de transição de estado da entidade Avaliador de Desempenho.

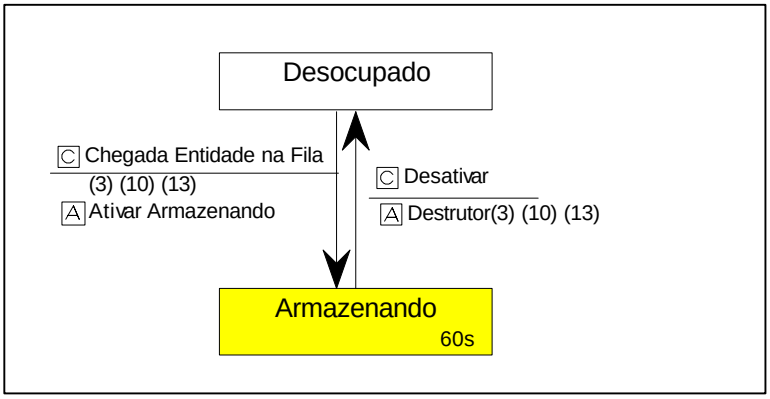

Fig. 6.19 - Diagrama de transição de estado da entidade Armazenador de Dados.

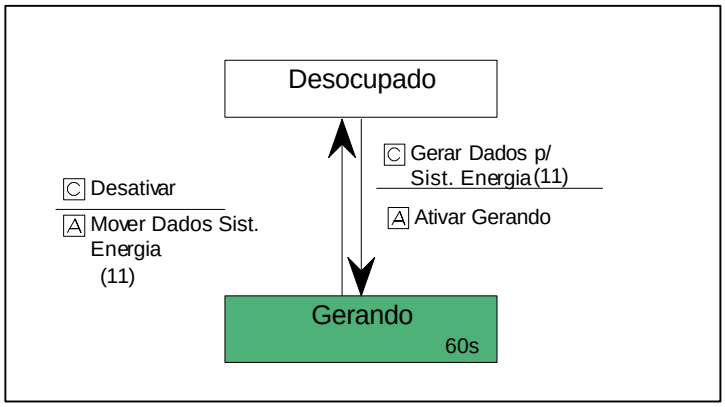

Fig. 6.20 - Diagrama de transição de estado da entidade Recuperador de Dados.

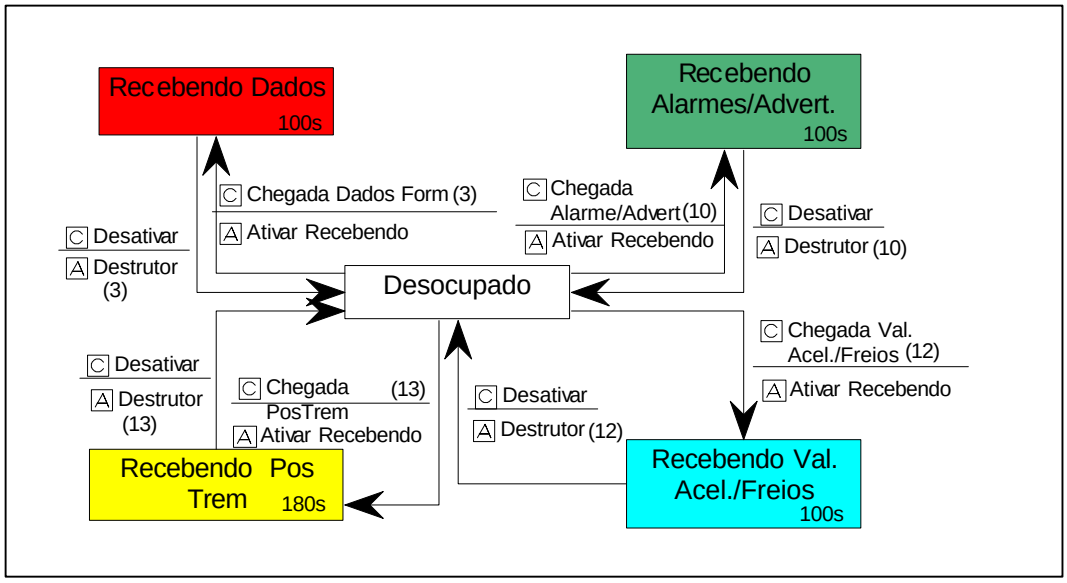

Fig. 6.21 - Diagrama de transição de estado da entidade Sistema de Visualização redefinida.

6.2.2 DEFINIÇÃO DAS ENTIDADES DINÂMICAS DO NÍVEL 2

Foram modeladas três novas entidades dinâmicas: **Alarmes/Advertências**, **Dados do Sistema de Energia**, **Valores de Aceleração/Freios**. As outras entidades dinâmicas já definidas tiveram seu percurso modificado, considerando agora as novas entidades estáticas deste nível.

A seguir são identificadas as novas entidades dinâmicas e apresentadas as modificações realizadas nas já existentes. Analogamente à modelagem feita no nível 1, existem

atributos que possuem os mesmos valores para todas as entidades, conforme listados abaixo.

- Lista de Atributos: vazia;
- Figura: círculo;
- Lista de Métodos: vazia;
- Lista de Entidades Dinâmicas: vazia.

Entidade Dinâmica *Alarmes/Advertências*

Atributos
Nome: Alarmes/Advertências
ID: 10
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sensores	Sistema de Visualização	50s
2	Sensores	Fila Acesso BD	40s
2	Fila Acesso BD	Armazenador de Dados	60s

Entidade Dinâmica *Dados Sistema de Energia*

Atributos
Nome: Dados Sistema de Energia
ID: 11
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Recuperador de Dados	Avaliador de Desempenho	80 s

Entidade Dinâmica *Valores de Aceleração e Freios*

Atributos
Nome: Valores de Aceleração e Freios
ID: 12
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Avaliador de Desempenho	Sistema de Visualização	50 s

6.2.3 ALTERAÇÃO DE ENTIDADES DINÂMICAS DEFINIDAS NO NÍVEL 1

A seguir, as entidades dinâmicas definidas no nível 1, que sofreram alterações no nível 2, são remodeladas.

Entidade Dinâmica *Dados GPS*

Atributos
Nome: Dados GPS
ID: 1
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Satélite	Unid. Gerência de Dados	60 s
2	Unid. Gerência de Dados	Subsistema Posição do Trem	30 s

Entidade Dinâmica *Dados Formatados*

Atributos
Nome: Dados Formatados
ID: 3
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sensores	Sistema de Visualização	20 s
2	Sensores	Fila Acesso BD	40 s
3	Fila Acesso BD	Armazenador de Dados	40 s
4	Armazenador de Dados	Controlador Terminal-Terra	30 s
5	Controlador Terminal-Terra	Sistema Controle Rede	40 s
6	Sistema Controle Rede	Centro de Despacho	60 s

Entidade Dinâmica *Dados Brutos Sensores*

Atributos
Nome: Dados Brutos Sensores
ID: 2
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sensores do Trem	Sensores	30 s

Entidade Dinâmica *Localização Transponder*

Atributos
Nome: Localização *Transponder*
ID: 4
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	*Transponder*	Unid. Gerência de Dados	60 s
2	Unid. Gerência de Dados	Subsistema Posição do Trem	30 s

Entidade Dinâmica *Posição Calculada do Trem*

Atributos
Nome: Posição Calculada do Trem
ID: 13
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Subsistema Posição do Trem	Sistema de Visualização	40 s
2	Subsistema Posição do Trem	Fila Acesso BD	80 s
3	Fila Acesso BD	Armazenador de Dados	40 s

Eventos Agendados:

Nome: Limites Extrapolados
ID: 33
Geração: aleatória (função distribuição)
O evento **Limites Extrapolados** é gerado de acordo com uma função de distribuição. Este evento provoca a transição do estado Analisando Limites da entidade Sensores para o estado Criando Alarmes/Advertências.

6.3 IDENTIFICAÇÃO DAS ENTIDADES – NÍVEL 3

Para a criação do modelo deste nível de abstração, foi considerada somente a entidade estática Sistema de Análise e Relatórios e a Figura 6.22 mostra a nova decomposição desta entidade.

Uma nova entidade estática foi criada pelo modelador, a entidade Conversor, e a partir das informações por ele fornecidas, ela é apresentada na Figura 6.23 utilizando um diagrama de transição de estado. Esta entidade e as alterações sofridas nas entidades já existentes são descritas nas seções seguintes.

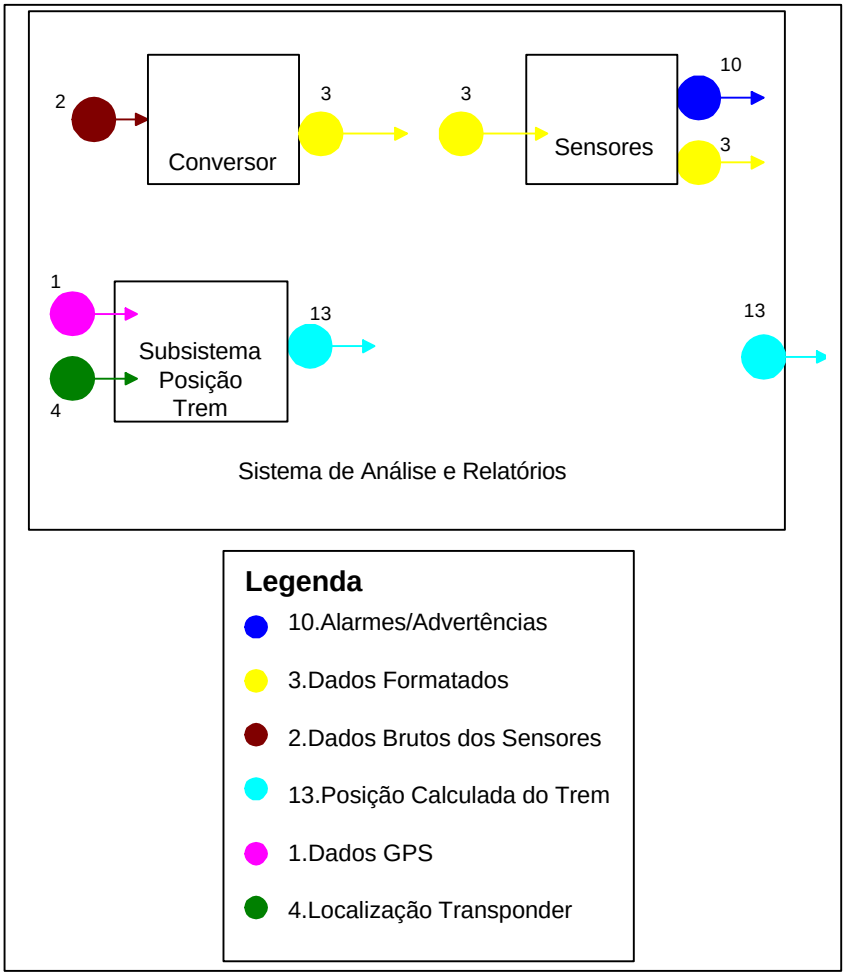

Fig. 6.22 - Sistema de Gerência de Tráfego de Trens- nível 3: Sistema de Análise e Relatórios

6.3.1 DEFINIÇÃO DAS ENTIDADES ESTÁTICAS DO NÍVEL 3

A nova entidade estática identificada, Conversor, converte dados brutos (conjunto de bits) recebidos da entidade estática Sensores do Trem em dados formatados (valores reais de temperatura, pressão, combustível, RPM) e aceleração, conforme mostra a Figura 6.23. Alguns atributos e métodos da entidade Conversor já puderam ser

identificados, conforme é mostrado a seguir nos atributos Lista de Atributos e Lista de Métodos da entidade.

Entidade Estática *Conversor*

Atributos
Nome: Conversor
ID: 32
Cor: transparente
Figura: círculo
Lista de Atributos: DadosBrutos (conjunto de bits)
Lista de Métodos: Ler Dados Brutos, Converter Dados, Formatar Dados, Criar Sensor Digital
Lista de Entidades Dinâmicas: vazia
Lista de Estados:

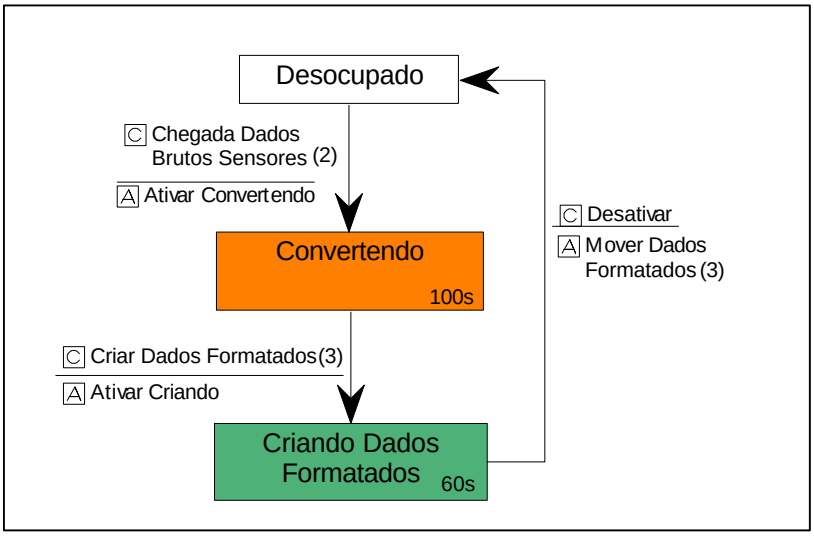

Fig. 6.23 - Diagrama de transição de estado da entidade Conversor.

6.3.2 ALTERAÇÃO DA ENTIDADE ESTÁTICA SENSORES DO NÍVEL 2

A entidade Sensores sofreu alterações na sua lista de estados com a eliminação de dois que estavam presentes no nível 2. A Figura 6.24 mostra o novo diagrama de transição de estado da entidade redefinida.

Foi possível também identificar neste nível, alguns dos métodos desta entidade, conforme é mostrado a seguir no seu atributo Lista de Métodos.

Entidade Estática *Sensores*

Atributos
Nome: Sensores
ID: 22
Cor: transparente
Figura: retângulo
Status: habilitada
Lista de Atributos: vazia
Lista de Métodos: Analisar Limite Temperatura Água, Analisar Limite Temperatura Óleo, Analisar Limite Pressão, Analisar Limite Combustível, Analisar Limite Aceleração, Analisar Limite RPM, Enviar Advertência, Enviar Alarme, Enviar Dados Formatados.
Lista de Entidades Estáticas: vazia

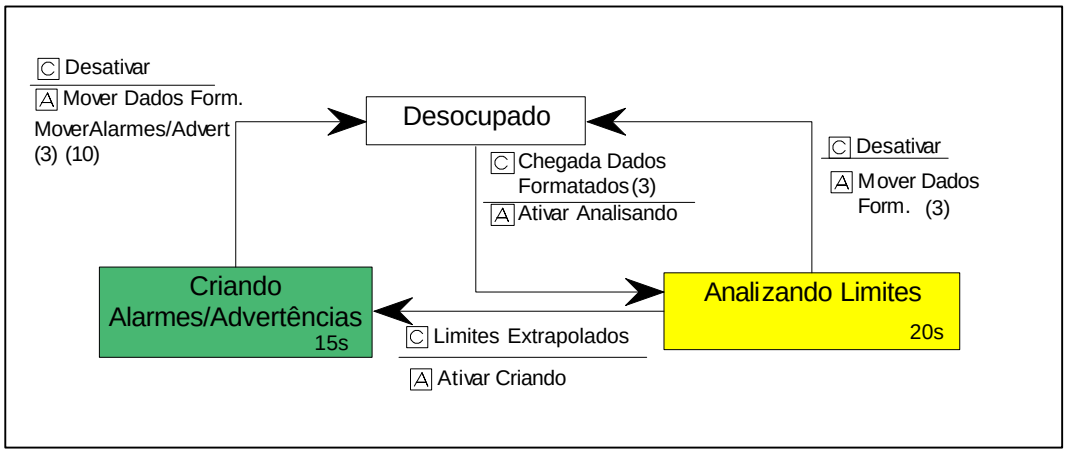

Fig. 6.24 - Diagrama de transição de estado da entidade Sensores redefinida.

6.3.3 ALTERAÇÃO DE ENTIDADES DINÂMICAS DEFINIDAS NO NÍVEL 2

As entidades dinâmicas Dados Formatados e Dados Brutos Sensores sofreram alterações na suas listas de percursos, conforme é mostrado a seguir.

Entidade Dinâmica *Dados Formatados*

Atributos
Nome: Dados Formatados
ID: 3
Cor:

Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Conversor	Sensores	20 s
2	Sensores	Fila Acesso BD	40 s
3	Sensores	Sistema de Visualização	40s
4	Fila Acesso BD	Armazenador de Dados	40 s
5	Armazenador de Dados	Controlador Terminal-Terra	30 s
6	Controlador Terminal-Terra	Sistema Controle Rede	40 s
7	Sistema Controle Rede	Centro de Despacho	60 s

Entidade Dinâmica *Dados Brutos Sensores*

Atributos
Nome: Dados Brutos Sensores
ID: 2
Cor:
Lista de Percursos:

Percurso	Origem	Destino	Tempo
1	Sensores do Trem	Conversor	30 s

6.4 IDENTIFICAÇÃO DAS CLASSES DE OBJETOS

A partir das entidades estáticas e dinâmicas identificadas no nível 3 de abstração, uma tentativa foi feita no sentido de delinear algumas classes de objetos pertencentes ao domínio do problema. Como não se dispõe de informações mais detalhadas a respeito do sistema, a tentativa teve o propósito de verificar esta possibilidade.

A entidade Sensores, por exemplo, realiza operações sobre a entidade dinâmica Dados Formatados, sendo esta última resultado da conversão dos dados brutos dos sensores para um formato digital, realizada pela entidade Conversor. Uma classe chamada Sensor Digital poderia ser criada para encapsular a entidade dinâmica Dados Formatados e a entidade estática Sensores.

A entidade estática Conversor e a entidade dinâmica Dados Brutos Sensores poderiam ser encapsuladas numa classe chamada Conversor. A entidade estática Subsistema Posição do Trem calcula a posição do trem baseada em dados (entidades dinâmicas) de GPS, do *Transponder* e da contagem de revoluções da roda. Esta entidade e as entidades

dinâmicas que ela manipula poderiam ser encapsuladas numa classe Subsistema Posição do Trem.

A Figura 6.25 mostra a modelagem destas classes e seus relacionamentos utilizando a notação UML (1997).

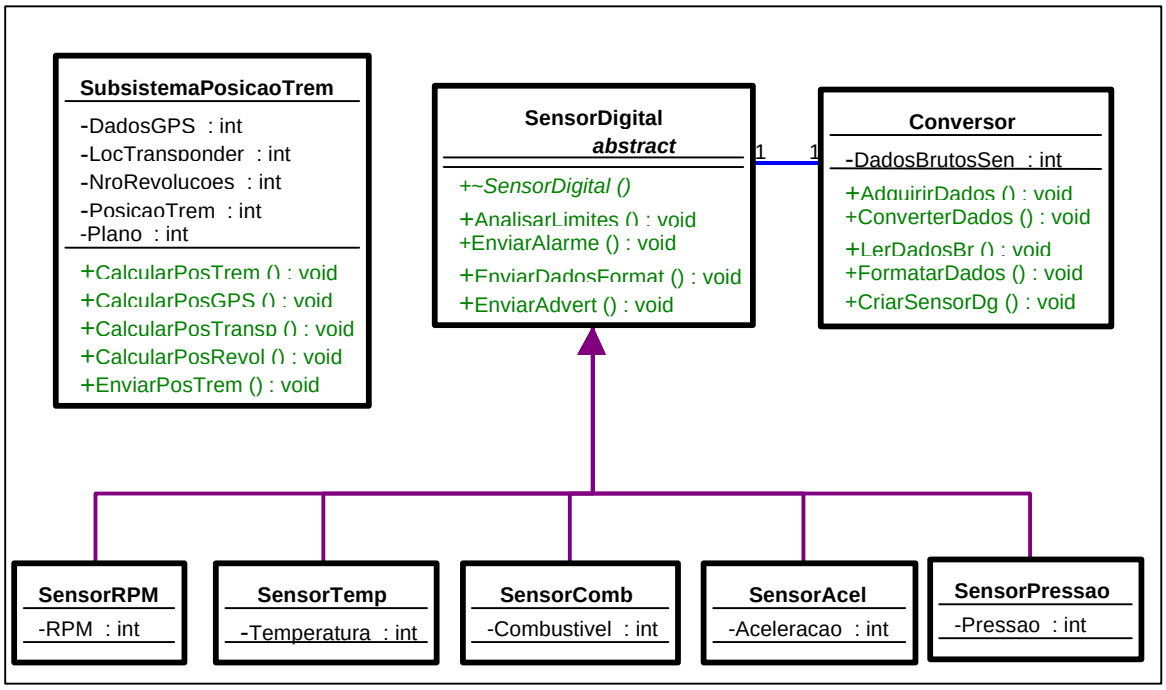

Fig. 6.25 - Diagrama mostrando a estruturação em classes de algumas entidades identificadas do Sistema de Análise e Relatórios.

No nível 3 do modelo, as entidades estáticas Conversor, Subsistema Posição do Trem e Sensores foram transformadas em objetos de maneira direta, resultando nas classes Conversor, Subsistema Posição do Trem e Sensor Digital.

No tratamento mais detalhado do objeto Sensor Digital, foram identificados outros objetos como especializações deste primeiro, que não apareceram no nível 3 do modelo criado. Talvez eles tivessem presentes no modelo se fosse criado mais um nível de refinamento relacionado à entidade Sensores.

Se a entidade estática Sistema de Análise e Relatórios não tivesse sido decomposta a partir da incorporação de mais detalhes, ela provavelmente não seria candidata a se

tornar um objeto do sistema, pois ela possui partes do sistema que não se relacionam, contrariando o conceito de objeto.

Estes aspectos do mapeamento das entidades do modelo para objetos de software do sistema, só puderam ser observados em níveis mais detalhados do modelo (nível 2 e 3). No nível 1, esta identificação dos objetos seria impossível.

Em contrapartida, se o modelo do nível 1 fosse construído num nível mais alto de abstração, talvez um objeto Trem, englobando o Subsistema Posição do Trem e o Avaliador de Desempenho tivesse sido identificado.

Com o exposto acima, pode-se observar que o mapeamento das entidades do modelo de animação para os objetos de software não é biunívoca. Depende dos níveis de abstração que foram criados para o modelo, e além disso, nem todas as entidades criadas deverão ser transformadas em objetos.

A proposta do ambiente de modelagem é fornecer ao modelador condições para que ele possa criar os modelos de animação nos níveis que lhe sejam mais convenientes, considerando as informações a respeito do sistema e do domínio do problema e seu conhecimento sobre as técnicas de orientação a objetos.

6.5 MECANISMO DE ESCALONAMENTO DOS EVENTOS

Nesta Seção, é colocado uma amostra de como será feito o escalonamento dos eventos pelas rotinas de simulação incorporadas ao ambiente, a partir das informações visuais do modelo gráfico construído, e das informações fornecidas pelo modelador, durante a criação do modelo.

Para exemplificar o mecanismo, foi selecionado o modelo do Sistema de Gerência de Tráfego de Trens, apresentado nas seções anteriores. Deste modelo, foram considerados os dois primeiros níveis de abstração criados para a apresentação dos escalonamentos parciais dos eventos, selecionando um ponto de partida e acompanhando algumas das entidades envolvidas na animação do modelo.

6.5.1 ESCALONAMENTO PARCIAL DOS EVENTOS – NÍVEL 1

Para este escalonamento parcial foi selecionado como ponto de início para a animação o evento **Gerar Dados Brutos,** que provoca a geração de dados de sensores analógicos pela entidade Sensores do Trem, e a partir daí a entidade dinâmica **Dados Brutos Sensores** e a entidade criada decorrente de seu processamento, **Dados Formatados**, foram acompanhadas até seus movimentos finais, conforme é mostrado na Tabela 6.1.

O escalonamento é parcial porque não são considerados todos os eventos que ocorrem no sistema e todas as entidades afetadas, mas só aqueles eventos que dizem respeito às entidades acima citadas. Obviamente, entre um e outro evento escalonado no exemplo, existem outros eventos ocorrendo também.

Evento Inicial:

Gerar Dados Brutos Sensores : periódico 300 em 300 s.

TABELA 6.1 - ESCALONAMENTO PARCIAL DOS EVENTOS - NÍVEL 1

	Evento	**Tempo Duração**
1	**Gerar** Dados Brutos Sensores	0 s
2	Sensores do Trem→**Ativar** (Gerando Dados Brutos Sensores)	60 s
3	Sensores do Trem→**Desativar** (Gerando Dados Brutos Sensores)	0 s
4	Dados Brutos Sensores→**Mover**(Sistema de Análise e Relatórios)	30 s
5	Dados Brutos Sensores→**AtualizaPercurso**	0 s
6	Dados Brutos Sensores→**Chegada** (Sistema de Análise e Relatórios)	0 s
7	Sistema de Análise e Relatórios→**Ativar** (Formatando)	15 s
8	Sistema de Análise e Relatórios→**Criar** (Dados Formatados)	20 s
8	Sistema de Análise e Relatórios→**Ativar** (Criando Dados Formatados)	20 s
9	Sistema de Análise e Relatórios→**Desativar** (Criando Dados Formatados)	0 s
10	Dados Formatados →**Mover** (Sistema de Visualização)	20 s
11	Dados Formatados →**AtualizaPercurso**	0 s
12	Dados Formatados→**Mover** (Unidade de Gerência de Dados)	40 s
13	Dados Formatados →**AtualizaPercurso**	0 s
14	Dados Formatados→**Chegada** (Sistema de Visualização)	0 s
15	Sistema de Visualização→**Ativar** (Recebendo Dados Formatados)	100 s
16	Dados Formatados→**Chegada** (Unidade de Gerência de Dados)	0 s
17	Unidade de Gerência de Dados→**Ativar**(Recebendo Dados Formatados)	60 s
18	Sistema de Visualização→**Desativar** (Recebendo Dados Formatados)	0 s
19	Unidade de Gerência de Dados→**Desativar**(Recebendo Dados Formatados)	0 s

20	Dados Formatados→**Mover** (Controlador Terminal-Terra)	30 s
21	DadosFormatados→**AtualizaPercurso**	0 s
22	DadosFormatados→**Chegada** (Controlador Terminal-Terra)	0 s
23	Controlador Terminal-Terra→**Ativar** (Recebendo Dados Formatados)	120 s
24	Controlador Terminal-Terra→**Desativar** (Recebendo Dados)	0 s
25	Dados Formatados→**Mover** (Sistema Controle Rede)	40 s
26	DadosFormatados→**AtualizaPercurso**	0 s
27	DadosFormatados→**Chegada** (Sistema Controle Rede)	0 s
28	Sistema Controle Rede→**Ativar** (Recebendo Dados Formatados)	120 s
29	Sistema Controle Rede→**Desativar** (Recebendo Dados Formatados)	0 s
30	Dados Formatados→**Mover** (Centro Despacho)	60 s
31	DadosFormatados→**AtualizaPercurso**	0 s
32	Dados Formatados→**Chegada** (Centro Despacho)	0 s
33	Centro Despacho→**Ativar**(Recebendo e Armazenando Dados Formatados)	360 s
34	Centro Despacho→**Desativar**(Recebendo e Armazenando Dados Formatados)	0 s
35	Dados Formatados→**Destrutor**	0 s

6.5.2 ESCALONAMENTO PARCIAL DOS EVENTOS – NÍVEL 2

Nesta Seção o escalonamento parcial é apresentado para o nível 2 de abstração e foram selecionados os mesmos pontos de início para a animação do escalonamento do nível 1, conforme retrata a Tabela 6.2. A entidade dinâmica Dados Brutos Sensores e a entidade criada decorrente de seu processamento, Dados Formatados, foram acompanhadas até seus movimentos finais. Neste escalonamento foi simulado também a geração do evento aleatório **Limites Extrapolados.**

Eventos Agendados:

Gerar Dados Brutos Sensores : evento inicial, periódico 300 em 300 s;

Limites Extrapolados: evento que pode ocorrer durante o estado Analisando Limites da entidade estática Sensores, gerado segundo uma função de distribuição.

TABELA 6.2 - ESCALONAMENTO PARCIAL DOS EVENTOS - NÍVEL 2

	Evento	**Tempo Duração**
1	**Gerar Dados Brutos Sensores**	0 s
2	Sensores do Trem→**Ativar** (Gerando Dados Brutos Sensores)	60 s
3	Sensores do Trem→**Desativar** (Gerando Dados Brutos Sensores)	0 s
4	Dados Brutos Sensores→**Mover**(Sensores)	30 s
5	Dados Brutos Sensores→**AtualizaPercurso**	0 s
6	Dados Brutos Sensores→**Chegada** (Sensores)	0 s
7	Sensores→**Ativar** (Formatando Dados Sensores)	100 s
8	Sensores→**Criar** (Dados Formatados)	0 s
9	Sensores→**Ativar** (Criando Dados Formatados)	80 s
10	Sensores→**Analisar Limites**	0 s
11	Sensores→**Ativar** (Analisando Limites)	80 s
12	**Limites Extrapolados**	0 s
13	Sensores→ **Ativar** (Criando Alarmes/Advertências)	360 s
14	Sensores→ **Desativar** (Criando Alarmes/Advertências)	0 s
15	Dados Formatados →**Mover** (Sistema de Visualização)	20 s
16	Dados Formatados →**AtualizaPercurso**	0 s
17	Dados Formatados→**Mover** (Fila Acesso BD)	40 s
18	Dados Formatados →**AtualizaPercurso**	0 s
19	Dados Formatados→**Chegada** (Sistema de Visualização)	0 s
20	Sistema de Visualização→**Ativar** (Recebendo Dados Formatados)	100 s
21	Dados Formatados→**Chegada** (Fila Acesso BD)	0 s
22	Dados Formatados→**Mover** (Armazenador de Dados) (se Armazenador de Dados desocupada)	40 s
23	Dados Formatados →**AtualizaPercurso**	0 s
24	Dados Formatados→**Chegada** (Armazenador de Dados)	0 s
25	Armazenador de Dados→**Ativar** (Armazenando)	100 s
26	Armazenador de Dados→**Desativar**(Armazenando)	0 s
27	Dados Formatados→**Mover** (Controlador Terminal-Terra)	30s
28	DadosFormatados→**AtualizaPercurso**	0 s
29	DadosFormatados→**Chegada** (Controlador Terminal-Terra)	0 s
30	Controlador Terminal-Terra→**Ativar** (Recebendo Dados Formatados)	120 s
31	Controlador Terminal-Terra→**Desativar** (Recebendo Dados)	0 s
32	Dados Formatados→**Mover** (Sistema Controle Rede)	40 s
33	DadosFormatados→**AtualizaPercurso**	0 s
34	DadosFormatados→**Chegada** (Sistema Controle Rede)	0 s
35	Sistema Controle Rede→**Ativar** (Recebendo Dados Formatados)	120 s
36	Sistema Controle Rede→**Desativar** (Recebendo Dados Formatados)	0 s

37	Dados Formatados→**Mover** (Centro Despacho)	60 s
38	DadosFormatados→**AtualizaPercurso**	0 s
39	Dados Formatados→**Chegada** (Centro Despacho)	0 s
40	Centro Despacho→**Ativar**(Recebendo e Armazenando Dados Formatados)	360 s
41	Centro Despacho→**Desativar**(Recebendo e Armazenando Dados Formatados)	0 s
42	Dados Formatados→**Destrutor**	0 s

O Capítulo seguinte apresenta as conclusões do trabalho, retomando seus principais aspectos e sua contribuição para a área de Engenharia de Software; apresentando também sugestões de possíveis trabalhos futuros.

CAPÍTULO 7

CONCLUSÕES

7.1 CONSIDERAÇÕES FINAIS

Atualmente, a demanda por sistemas de software de grande porte cada vez mais complexos, nos quais o tempo de resposta e o custo são fatores críticos, tem proporcionado grandes desafios à comunidade científica da área de Engenharia de Software.

A produção de sistemas mais confiáveis e com mais qualidade, aliada aos curtos prazos para a execução desta tarefa, tem norteado a busca de novas técnicas para serem empregadas durante a fase de desenvolvimento destes sistemas.

Nesta busca, deve ser considerada a importância de se produzir melhores especificações de requisitos e modelos iniciais, que devem levar em conta os aspectos dinâmicos e comportamentais dos sistemas. Estes aspectos envolvem a representação do que ocorre com o sistema ao longo do tempo, durante o seu funcionamento, e as ações e transformações que os seus componentes sofrem e realizam, sob certas condições ou circunstâncias.

Considerando a atual carência de técnicas que retratem esses aspectos de maneira adequada e completa, o objetivo deste trabalho foi propor uma nova abordagem para a representação dos modelos dinâmicos dos sistemas, apresentando uma nova técnica de modelagem para a especificação e representação dos aspectos dinâmicos, resultando na criação de modelos de animação. Estes modelos retratam o comportamento dinâmico dos sistemas ao longo do tempo.

Para cumprir este objetivo, inicialmente foi necessário definir quais elementos estariam envolvidos na representação dos aspectos dinâmicos para a partir daí, considerá-los no

processo de modelagem. Foram definidos dez elementos julgados chaves para a representação, conforme listados a seguir.

EC1- Seqüências de acontecimentos ou eventos;

EC2- Quando os eventos ou acontecimentos ocorrem no tempo;

EC3- Representação das entidades componentes;

EC4- Fluxos de dados ou entidades (periódico ou aperiódico);

EC5- Transformações ocorridas no decorrer do tempo que afetem a seqüência, que gerem novos componentes e entidades, que os destruam ou que os transformem;

EC6- Paralelismo na ocorrência de dois ou mais conjuntos de eventos ou acontecimentos e paralelismo entre transformações;

EC7- Sincronismo entre dois ou mais conjuntos de eventos ou acontecimentos e sincronismo entre transformações;

EC8- Concorrência por recursos;

EC9- Estabelecimento de pré e pós-condições; e

EC10- Estabelecimento de prioridades.

Para verificar a utilização destes elementos na modelagem dos aspectos dinâmicos, alguns casos foram estudados. Procurou-se considerar especificações de sistemas de diferentes áreas e cujas características evidenciassem o objeto de estudo. Foram analisados três casos, o Sistema da Fábrica de Artefatos, o Sistema de Controle de Satélites e o Sistema de Gerência de Tráfego de Trens.

Apesar de os resultados desta análise não garantirem que os dez elementos-chave sejam suficientes para a representação dos aspectos dinâmicos, foi verificado que eles são necessários para a representação das realidades envolvidas.

Baseado neste resultado, o passo seguinte foi definir uma técnica de modelagem que tornasse possível englobar os dez elementos-chave em um único modelo do sistema. Foi então proposto e definido um simbolismo para a representação gráfica do modelo e o

emprego de mecanismos de animação e de simulação discreta orientada a eventos, resultando no modelo de animação do sistema.

A técnica, basicamente, propõe a criação do modelo a partir de entidades estáticas e dinâmicas que interagem entre si, e a animação deste modelo resulta na visualização do funcionamento do sistema como um todo e, como conseqüência, os seus aspectos dinâmicos ficam melhor representados.

Para que a técnica de criação do modelo de animação fosse melhor definida, caracterizou-se um ambiente de modelagem. Este ambiente, baseado em interfaces gráficas, oferece ao modelador do sistema facilidades de uso e aprendizado e, entre outros recursos, possibilita a adequação dos símbolos gráficos utilizados e a liberdade para a construção do modelo de maneira *top-down* e *bottom-up*.

Apesar de serem utilizados mecanismos de simulação para a criação do modelo de animação, o modelador não precisa ter conhecimento prévio sobre modelos de simulação. Para a compreensão do processo de modelagem, definiu-se também um ciclo de vida para o modelo de animação proposto.

Uma vez caracterizado o ambiente de modelagem e a técnica utilizada, foi concebida uma infra-estrutura orientada a objetos para a implementação do ambiente. Na análise dos objetos componentes desta infra-estrutura, foram considerados aqueles necessários para a representação gráfica do modelo e sua animação.

Foi possível manter separadas, por meio de mecanismos de herança, as classes de objetos relacionadas à animação, das classes que contêm informações que poderão ser utilizadas posteriormente para a derivação de alguns dos principais objetos de software do sistema.

Os mecanismos de simulação envolvidos que tornam possível a animação do modelo dentro de um cronograma coerente no tempo, as funções de distribuição de probabilidade para a geração dos valores das variáveis aleatórias, a manipulação de filas, a geração de amostras aleatórias e a coleta e suporte estatísticos são funções comuns já implementadas nas linguagens de simulação e não foram consideradas para

efeito da modelagem, dado que uma linguagem deste tipo deverá ser utilizada para a implementação do ambiente.

Para exemplificar o uso da técnica proposta e da infra-estrutura orientada a objetos do ambiente de modelagem, o caso do Sistema de Gerência de Tráfego de Trens foi novamente analisado, agora com o objetivo de construir o seu modelo de animação.

Considerando a informação disponível na especificação, foram criados três níveis de abstração do modelo, por meio de sucessivos refinamentos. Estes refinamentos foram realizados em porções do modelo, não considerando o modelo como um todo.

A partir do modelo mais refinado obtido, foram derivados alguns dos principais objetos de software, que podem ser utilizados como ponto de partida para que uma análise orientada a objetos mais detalhada seja realizada e um projeto orientado a objetos possa ser conduzido.

Baseado na experiência obtida, algumas considerações também puderam ser feitas sobre o processo de identificação dos objetos, tornando claro a importância da utilização de estratégias *top-down* e *bottom-up* no processo de criação do modelo de animação.

Para exemplificar como um cronograma de eventos é montado, permitindo a animação do modelo dentro do tempo simulado, um escalonamento parcial de eventos foi apresentado, onde os eventos considerados para a simulação tornam-se chamadas dos métodos dos objetos da infra-estrutura proposta.

Utilizando o ambiente proposto, o modelo de animação pode ser usado para explorar visualmente a dinâmica do sistema, de uma maneira simples e amigável, para entender o seu comportamento e interagir para testar hipóteses relacionadas ao desempenho, à robustez e à concorrência por recursos.

A animação fornece um conjunto de informações a respeito da seqüência de eventos gerados pelo modelo, possibilita o acompanhamento paralelo das várias entidades navegando pelo sistema e também o entendimento das várias interações entre as entidades concorrentes, difíceis de compreender quando se está limitado à uma visualização estática. O comportamento transiente do sistema também pode ser representado.

Este trabalho possui limitações e não esgota o assunto a respeito da especificação e representação dos aspectos dinâmicos dos sistemas, mas utilizando uma abordagem inovadora, acrescenta melhorias na maneira como a dinâmica é especificada atualmente, abrindo caminho para que outros aspectos não cobertos possam ser pesquisados, resultando num aperfeiçoamento da proposta.

Algumas das limitações identificadas são relacionadas a seguir:

- Os dez elementos-chave considerados para a representação da dinâmica foram verificados por meio de alguns casos estudados, não sendo feita uma análise com critérios mais rígidos com o objetivo de encontrar um conjunto mínimo, necessário e suficiente de elementos que deveriam fazer parte da representação do modelo;
- A caracterização do ambiente não foi realizada de forma completa, apenas considerou-se as partes mais importantes para o entendimento da técnica e do processo de criação do modelo de animação; e
- Uma análise mais efetiva do ambiente e da infra-estrutura orientada a objetos para sua implementação, baseada no estudo de vários casos, não foi concluída.

Alguns dos aspectos envolvidos nestas limitações e outros não cobertos por este trabalho, poderão ser considerados em trabalhos futuros, conforme apresentado na Seção 7.3.

Não obstante as limitações, pode-se dizer que o objetivo do trabalho foi alcançado, advindo algumas contribuições, discutidas na Seção seguinte.

7.2 CONTRIBUIÇÕES DO TRABALHO

Este trabalho propôs uma nova abordagem para a representação do modelo dinâmico dos sistemas, nos estágios iniciais da modelagem, melhorando a especificação e a representação dos seus aspectos dinâmicos, considerando a sua importância para o entendimento do funcionamento do sistema como um todo e a atual carência de técnicas que os representem adequadamente.

Pode-se destacar as seguintes contribuições:

- **Melhoria na representação dos aspectos dinâmicos dos sistemas.**

Considerando-se a complexidade, o porte e a natureza específica dos sistemas atuais, principalmente aqueles de tempo real, a compreensão dos aspectos dinâmicos torna-se um ponto chave para a correta especificação e análise dos requisitos. Especificações incompletas e inconsistentes, geradas normalmente pela falta de compreensão do sistema, podem provocar problemas sérios numa fase avançada do desenvolvimento do software.

A criação de um modelo de animação como proposto neste trabalho, possibilita a visão do sistema como um todo, por meio da representação dos seus aspectos dinâmicos numa fase inicial da modelagem.

A animação oferece um traçado visual do funcionamento do sistema que supera a visão estática oferecida pelas ferramentas clássicas, possibilitando a visualização da seqüência de eventos, o acompanhamento simultâneo das várias entidades navegando pelo sistema, além das diversas interações entre as entidades concorrentes.

O comportamento transiente do sistema também pode ser representado e hipóteses sobre o seu funcionamento podem ser feitas.

- **Proposta do emprego de técnicas de simulação para a atividade de especificação dos requisitos dos sistemas de software.**

O emprego de técnicas de Engenharia de Software e os avanços conceituais no tocante à modelagem de sistemas têm ajudado bastante o desenvolvimento e a evolução dos programas que dão suporte ao estudo de um modelo de simulação (Lin at al, 1997).

A tecnologia de orientação a objetos, além de utilizada para o desenvolvimento dos programas, vem sendo também utilizada como base para a área de simulação.

Este trabalho, além de contribuir com a aproximação das áreas de Engenharia de Software e Simulação, explora também esta última tecnologia. Ele propõe a utilização de técnicas de simulação discreta de sistemas, orientadas a eventos, na atividade de especificação de requisitos para criar modelos de animação dos sistemas de software.

O potencial da simulação e animação é explorado, resultando numa melhor compreensão do funcionamento do sistema como um todo e uma melhor representação dos seus aspectos dinâmicos e da especificação dos requisitos.

- **Estabelecimento da conexão entre a especificação dos requisitos e o desenvolvimento do software propriamente dito.**

O processo de criação e os sucessivos refinamentos do modelo de animação do sistema incorporam informações ao modelo, que são adquiridas através do modelador e também coletadas automaticamente pelo ambiente a partir do modelo gráfico. Estas informações podem ser utilizadas não somente na fase de especificação dos requisitos, mas também em fases posteriores do desenvolvimento do software, estabelecendo uma conexão entre a especificação e o desenvolvimento propriamente dito. Um exemplo deste fato é a possibilidade de derivação de partes de alguns objetos de software importantes do sistema a ser implementado, com base nos vários níveis de abstração criados para o modelo.

- **Caracterização de um ambiente de modelagem para a criação do modelo de animação.**

Outra contribuição do trabalho foi a caracterização de um ambiente de modelagem para dar suporte ao modelador durante o processo de criação do modelo de animação. Este ambiente foi concebido de modo a tornar fácil sua utilização e não limitar a criatividade do modelador, fornecendo ferramentas que não exigem um longo tempo de aprendizagem. As características principais deste ambiente são:

1) Criação de vários níveis de abstração, dando liberdade ao modelador de trabalhar na elaboração do modelo de uma maneira top-down (criação de subníveis) e bottom-up (agregação de entidades e criação de um nível superior);

2) Possibilidade do modelador visualizar a animação em todos os níveis criados, ou fazer a seleção dos níveis desejados;

3) Alteração da representação gráfica dos elementos do simbolismo padrão adotado para adequar-se às necessidades de representação do sistema em questão; e

4) Possibilidade de o modelador interagir durante a animação do modelo, alterando parâmetros como, duração, número de rodadas e a velocidade de exibição dos quadros de animação, e também parâmetros do modelo relacionados aos tempos de ocorrência dos eventos, às funções de distribuição de probabilidade, ao tempo de permanência de uma entidade em um estado e à habilitação de uma entidade estática ou dinâmica de participar ou não da animação em um determinado momento.

- **Manipulação de eventos num nível mais alto em relação à simulação.**

No desenvolvimento de um modelo de simulação orientado a eventos, a tarefa do modelador é determinar os eventos que podem causar as mudanças nos estados do sistema, e então desenvolver a lógica associada com cada evento determinado. A simulação do sistema é então produzida pela execução da lógica associada a cada evento, em uma seqüência ordenada no tempo.

A contribuição deste trabalho neste aspecto foi dotar o ambiente de funções que tornam a lógica dos eventos transparente para o modelador, ou seja, comparando com a simulação, o modelador não precisa escrever diretamente as rotinas de tratamento dos eventos definidos no modelo. Seu trabalho resume-se em definir os eventos e associá-los às entidades estáticas do modelo, sendo de responsabilidade do ambiente ordená-los cronologicamente no tempo e provocar sua ocorrência no tempo simulado.

- **Condução à identificação de classes e objetos de software.**

Uma outra contribuição do trabalho foi no sentido de ajudar o modelador a identificar os objetos de software do sistema. O que pode ser observado é que esta identificação não se encontra necessariamente ligada a níveis mais detalhados do modelo, apesar de que nestes níveis mais detalhados as chances de uma identificação mais coerente são maiores.

A abordagem sugerida, de modelar o sistema como um conjunto de entidades estáticas e dinâmicas que interagem e o comportamento associado a elas, conduz mais naturalmente aos objetos de software do sistema.

7.3 SUGESTÕES PARA TRABALHOS FUTUROS

Podem ser destacados os seguintes pontos como objetos de trabalhos futuros:

- **Implementação de um protótipo do ambiente utilizando uma linguagem de simulação orientada a objetos.**

A implementação de um protótipo do ambiente é colocada como uma atividade a ser realizada para dar uma continuidade natural ao trabalho iniciado. Após a sua construção, o estudo de vários casos deverá ser conduzido.

O processo de construção do protótipo e os resultados obtidos com os casos estudados, ajudarão no aperfeiçoamento da proposta deste trabalho e na seleção de uma linguagem de simulação orientada a objetos para a implementação do ambiente.

- **Incorporação no ambiente de melhorias para facilitar a criação do modelo de animação pelo modelador.**

Outra extensão do trabalho seria a incorporação ao ambiente de melhorias que facilitem a criação do modelo de animação pelo modelador. Estas melhorias incluem, por exemplo, um projeto de interfaces gráficas mais adequado, levando em conta os vários perfis de modeladores que podem utilizar o ambiente, guias para auxiliá-lo nas diferentes atividades do processo de modelagem e funções específicas para a confecção de relatórios.

- **Definição das pré e pós-condições para o modelo de animação.**

As pré e pós-condições definidas no modelo de animação estão associadas às entidades estáticas e às filas e, definem parte da lógica do modelo. Para a definição destas pré e pós-condições foi proposto a utilização dos próprios recursos disponíveis na linguagem de simulação selecionada para a implementação do ambiente.

Um estudo de quais condições aparecem com maior freqüência no modelo de animação poderia ser realizado, levando-se em conta o domínio do problema, com o objetivo de propor uma maneira sistemática de se definir as pré e pós-condições. Isto poderia resultar na incorporação ao ambiente de ferramentas que ajudariam o modelador a definir as condições de uma forma semi-automática.

- **Detecção e resolução das inconsistências relacionadas às informações fornecidas pelo modelador durante a criação do modelo.**

Um outro trabalho que pode ser realizado diz respeito ao estudo, e posterior incorporação ao ambiente, de mecanismos para detectar e resolver as inconsistências relacionadas às informações fornecidas pelo modelador, durante a criação do modelo de animação.

Algumas das inconsistências detectadas poderão ser resolvidas pelo próprio ambiente, com a prévia aprovação do modelador. Outras não poderão ser resolvidas pelo ambiente, devendo ser detectadas e reportadas ao modelador.

As inconsistências surgem, em parte, devido à liberdade que o modelador tem de atribuir valores relacionados ao tempo (tempo de permanência das entidades estáticas em um estado, a duração dos eventos e os tempos de movimento das entidades dinâmicas). Esta atribuição pode envolver tanto valores aleatórios como valores já pré-determinados. Quando os valores são gerados aleatoriamente, as inconsistências podem ocorrer mais freqüentemente.

- **Estudo consistente e abrangente da derivação das classes e objetos a partir do modelo de animação.**

A atividade de identificação de alguns dos objetos de software do sistema a ser implementado, a partir do modelo de animação criado, foi apresentada no Capítulo 4, na Seção 4.12, como um processo ainda empírico e subjetivo, apenas com o propósito de mostrar a possibilidade de se realizar esta atividade.

A Seção 6.4 do Capítulo 6 também traz algumas considerações sobre este processo, baseado nos resultados obtidos com a criação do modelo de animação para o Sistema de Gerência de Tráfego de Trens.

Como uma extensão deste trabalho, um estudo mais consistente e abrangente pode ser realizado, com o objetivo de dar um tratamento mais sistemático para partes deste processo de derivação das classes e objetos que comportem esta abordagem, incorporando ao ambiente ferramentas que dêem suporte para esta atividade. Para as

partes onde não é possível uma sistematização, alternativas para o seu tratamento podem ser sugeridas ao modelador pelo ambiente.

- **Estudo mais abrangente sobre os elementos necessários para representar a dinâmica dos sistemas.**

Uma outra extensão desejável do trabalho é a realização de um estudo mais abrangente sobre os elementos necessários para a representação da dinâmica dos sistemas, com o intuito de conseguir a suficiência de um conjunto de tais elementos, para posteriormente representá-los no modelo de animação.

REFERÊNCIAS BIBLIOGRÁFICAS

Achite, M. **União das metodologias mapas use case e Coad&Yourdon para a modelagem de sistemas**. São José dos Campos. 145p. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, 1997.

Alves, M. C. B. **Especificação e modelagem de sistemas:** aspectos dinâmicos e comportamentais. São José dos Campos: CTA, 1998. 191p. (CTA/IEAv-EIN/RP-001/98).

Ambler, S. The state chart. **Software Development**, v.5, n. 5, p.83-88, May 1997.

Amyot, D. **Telepresence**: a case study of a timethread-LOTOS approach to multimedia system design. [online]. <http://www.sce.carleton.ca/ftp/pub/ UseCaseMaps/>. Fev. 1997. Ottawa: Carleton University, Dec. 1993.

Badel, M.; Eyraud, S.; Duong, D. The new simulation facilities in QNAP2. In: **European simulation symposium**, Ghent, 1991. **Proceedings**. Ghent: SCS, 1991. p.5-10.

Balci, O. **Guidelines for successful simulation studies**. Blacksburg: Virginia Technical Department of Science Computer, 1985. 25p. (TR-85-2).

Balci, O.; Bertelrud, A. I.; Esterbrook, C. M.; Nance, R. E. The visual simulation environment. In: European simulation multiconference**,** 11., Istambul, 1-4 June 1997. **Proceedings.** San Diego: SCS, 1997. p.61-68.

Balzer, R. M.; Cohen, D.; Feather, M.; Goldman, N. M.; Swartout, W.; Wile, D. S. Operational specification as the basis for specification validation. **Theory and Practice of Software Technology**, v. 5, n. 8, p.21-49, Oct. 1983.

Bicarregui, J. C.; Fitzgerald, J. S.; Lindsay, P. A.; Mooore, R.; Ritchie, B. **Proof in VDM:** a practitioner's guide. London: Springer-Verlag, 1994. 287p.

Boehm, B. W. A spiral model of software development and enhancement. **IEEE Computer**, v 6, n.5, p.61-72, May 1988.

Booch, G. **Object solutions: managing the object oriented project**. Redwood City: The Benjamin/Cummings, 1995. 314p.

Booch, G. **Object-oriented analysis and design with applications.** Redwood City: The Benjamin/ Cummings, l994. 589p.

Bordeleau, F.; Amyot, D. **LOTOS interpretation of timethreads: a method and a case study**. [online]. <http://www.sce.carleton.ca/ftp/pub/ UseCaseMaps/>. ago. 1996. Ottawa: Carleton University, Dec. 1993.

Bordeleau, F.; Buhr, R. J. A.; Obaid, A. **LOTOS interpretation of architecture-based design:** a method and a case study. [online]. <http://www.sce.carleton.ca/ftp/pub /UseCaseMaps/>. ago. 1996. Ottawa: Carleton University, Mar. 1994.

Bruyn, W.; Jensen, R.; Keskar, D.; Ward, P.T. ESML: an extended systems modeling language based on the data flow diagram. **ACM Software Engineering Notes**, v.13, n.11, p.58-67, Nov. 1987.

Buhr, R. J. A. **Use case maps for attributing behaviour to system architecture.** [online]. <http://www.sce. carleton.ca/ faculty/ buhr>. ago. 1996. Ottawa: Carleton University, 1996.

Buhr, R. J. A. **Use case maps:** a new model to bridge the gap between requeriments and detailed design. [online]. <http://www.sce.carleton.ca/ ftp/pub/UseCaseMaps/>. jan. 1996. Ottawa: Carleton University, Nov. l995.

Buhr, R. J. A.; Casselman, R. S. **Timethread-role maps for object-oriented design of real-time and distributed systems**. [online]. <http://www.sce.carleton.ca/ftp/pub/UseCaseMaps/>. jan 1996. Ottawa: Carleton University, Nov. 1994.

Buhr, R. J. A.; Casselman, R. S. **Use case maps for object-oriented systems**. Upper Saddle River: Prentice Hall, 1996. 302p.

Burns, A.; Kirkham, J. A. The construction of information management system prototyping in ADA. **Software - Practice and Experience**, v.16, n.4, p.341-350, Apr. 1986.

Coad, P.; Yourdon, E. **Object-oriented analysis..** 2.ed. Englewood Cliffs**:**Yourdon Press, l99l. 233p.

Coad, P; Mayfield, M.; North, D. **Object models:** strategies, patterns and applications. 2.ed. Englewood Cliffs: Prentice-Hall ECS Professional, 1996. 356p.

Cockburn, A. **Surviving object-oriented projects:** a manager's guide. Reading: Addison Wesley, 1998. 258p.

Coleman, D.; Arnold, P.; Bodoff S.; Dollin, C.; Gilchrist, H.; Hayes, F.; Jeremaes, P. **Object-oriented development - the fusion method**. Englewood Cliffs: Prentice Hall, 1994. 236p.

Coleman, D.; Hayes, F.; Bear, S. Introducing objectcharts or how to use satatecharts in object-oriented design. **IEEE Transactions on Software Engineering**, v.18, n.1, p. 14-19, Jan. 1992.

Coomber, C. J. Automation of transformation schemas for object-oriented simulations. In: Object-oriented simulation: reusability, adaptability, maintainability**,** New York, 1997. **Proceedings**. New York: IEEE Press, 1997. p. 221-267.

Coomber, C. J. SCHEMASIM: a simulation environment for real-time systems. **Simulation Digest**, v.8, n.3, p. 26-31, Mar.1994.

Dahl, O. J.; Myhraug, B.; Nygaard, K. **The simula common base language**. Oslo: Norwegian Computing Center, 1968. 25p. (Publication S2).

Davis, A. M. **Software requirements:** objects, functions and states. Englewood Cliffs: Prentice Hall PTR, 1993. 287p.

DeMarco, T. **Structured analysis and systems specification.** Englewood Cliffs: Prentice Hall, 1978. 254p.

Ebert, C. Dealing with nonfunctional requirements in lager software systems. [CD ROM]. **Annals of Software Engineering**, v.3, p.367-395, 1997.

Embley, D. W.; Kurtz, B. D.; Woodfield, S. N. **Objetct oriented analysis - a model-driven approach**. Englewood Cliffs: Prentice Hall, 1992. 287p.

Felder, M.; Morzenti, A. Validating real-time systems by history-checking TRIO specifications. **ACM Transactions on Software Engineering and Methodology**, v.3, n.4, p.308-339, Oct. 1994.

Fitzgerald, K. Vulnerability exposed in AT&T 9-hours glitch. **The Institute - IEEE Spectrum,** v.14, n.3, p. 21-34, Mar. 1990.

Fuchs, N. Specifications are (preferably) executable. **Software Engineering Journal**, v.7, n.5, p.323-334, Apr. 1992.

Gane, C.; Sarson, T. **Structured systems analysis**. Englewood Cliffs: Prentice Hall, 1979. 267p.

Gaskell, C.; Phillips, R. Executable specifications and CASE. **Software Engineering Journal**, v.9, n. 7, p. 174-182, July 1994.

Gennaro, G. **Hierarchical object oriented requirements analysis.** [online]. <http://esapub.esrin.esa.it/pff/pffv5n3/genv5n3.htm>. abr. 1996. Mathematics and Software Division, ESTEC (ESA), Dec. 1995.

Goldberg, A.; Rubin, K. **Succeding with objects:** decision frameworks for project management. New York: Addison Wesley, 1995. 340p.

Gordon, G. **System simulation**. Hemel Hempstead: Prentice Hall, 1978. 313p.

Gullekson, G. **An interative development process for complex event–driven systems.** [online]. <http://www.objectime.on.ca/>. mar. 1996. Ontário: ObjecTime Limited, 1995a.

Gullekson, G. **Using an object-oriented methodology for an ATM protocol stack.** [online]. <http://www.objectime.on.ca/>. mar 1996. Ontário: ObjecTime Limited, 1995b.

Harbert, A.; Lively, W.; Sheppard, S. A graphical specification system for user inteface design. **IEEE Software**, v 23, n. 7, p. 12-20, July 1990.

Harel, D. On visual formalism. **Comunication of ACM**, v.31, n.5, p.514-517, May 1988.

Harel, D.; Gery, E. Executable object modeling with statecharts. [online]. <http://www.ilogix.com/fs_prod.htm>. fev. 1997. I-Logix, 1997.

Harel, D.; Lachover, H.; Naamad, A.; Pnueli, A.; Politi, M.; Sherman, R.; Shtull-Trauring, A.; Trakhtenbrot, M. STATEMATE: a working environment for the development of complex reactive systems. **IEEE Transactions on Software Engineering**, v.16, n.4, p.403-414, Apr. 1990.

Harel, D; Politi, M. **Modeling reactive systems using statecharts: The statemate approach**. [online]. <http://www. ilogix.com/fs_prod.htm>. mar.1996. I-Logix, 1996.

Hatley, D. J.; Pirbhai, A. I. **Strategies for real time system specifications.** New York: Dorset House, 1987. 326p.

Heimdahl, M. P. E.; Leveson, N. G. Completeness and consistency analysis of state-based requirements. In: International conference on software engineering, 17., Seatle, 23-30 Apr.1995. **Proceedings**. Seatle: SES,1995. p.3-14.

Henderson-Sellers, B. **Object-oriented metrics measures of complexity**. Upper Saddle River: Prentice Hall PTR, 1996. 234p.

Herring, C. ModSim: a new object-oriented simulation language. In: SCS multi-conference on OO simulation, San Diego, 1990. **Proceedings**. San Diego: SCS, 1990. p. 55-60.

Hill, D. R. C. Enchancing the QNAP2 object-oriented simulation language for manufacturing modeling. In: European simulation multi-conference, Lyon, 7-9 June 1993. **Proceedings.** Lyon: ESS, 1993. p. 35-40.

Hill, D. R. C. **Object-oriented analysis and simulation**. Reading: Addison-Wesley, 1996. 291p.

Hrischuk, C.; Rolia, J.; Woodside, C. M. **Automatic generation of a performance model using an object-oriented prototype.** [online]. <http://www.sce.carleton.ca/ftp/pub/>. ago. 1996. Ottawa: Carleton University, Jan. l995.

Humphrey, W. S. **Managing the software process.** Reading: Addison Wesley, 1989. 494p.

Instituto Nacional de Pesquisas Espaciais.Centro de Rastreio e Controle de Satélites (INPE/CRC). **Descrição interna do controle de satélites.** versão 1.0. São José dos Campos, 1989. 448p. (CCSSCSDIS_SWR_0002).

Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G. **Object-oriented software engineering - a use case driven approach.** 4.ed. Reading: Addison Wesley, l992. 528p.

Jorgensen, P. C.; Erickson, C. Object-oriented integration testing. **Communications of the ACM**, v.37, n.9, p.30-38, Sep. 1994.

Joyce, E. Software bugs: A matter of life and liability. **Datamation**, v. 15, n.5, p. 88-92, May 1987.

Kang, K. C.; Ko, K. I. PARTS: A temporal logic-based real-time software specification and verification method. In: International conference on software engineering, 17., Seatle, 23-30 Apr. 1995. **Proceedings**. Seatle: SES, 1995a. p. 169-176.

Kang, K. C.; Ko, K. I. PARTS: A temporal logic-based real-time software specification method supporting multiple viewpoints. In: International conference on software engineering, 17., Seatle, 23-30 April 1995. **Proceedings**. Seatle: SES, 1995b. p.177-184.

Kang, K. C.; Lee, K. W.; Lee, J. Y.; Kim, G. J. ASADAL/SIM: an incremental multi-level simulation and analysis tool for real -time software specifications. **Software - Practice and Experience**, v.28, n.4, p.445-462, Apr. 1998.

Kienbaum, G. S. **Simulação discreta de sistemas:** notas de aulas do curso CAP-259. São José dos Campos: Instituto Nacional de Pesquisas Espaciais, 1996.

Kirkerud, B. **Object-oriented programming with Simula**. Harlow: Addison-Wesley, 1989. 295p.

Kiviat, P. J.; Villanueva, R.; Markowitz, H. **The SIMSCRIPT II programming language.** Englewood Cliffs: Pentice Hall, 1969. 437p.

Kreutzer, W. **Sytem simulation: programming styles and languages**. Reading: Addison Wesley, 1986. 373p.

Law, A. M.; Larmey, C. S. **An introduction to simulation using SIMSCRIPT II.5**. Los Angeles: C.A.C.I, 1984. 225p.

Lea, R. J.; Chung, C. G. Rapid prototyping from structured analysis: executable specifications approach. **Information Software Technology**, v.32, n.9, p.589-597, Sep. 1990.

Leveson, N. G; Turner, C. S. An investigation of the Therac 25 accidents. **IEEE Computer**, v. 26, n.7, p.18-41, July 1993.

Lilegdon, W. R.; O'Reilly, J. J. **SLAM II quick reference manual**. New York: Pritsker & Associates, 1987. 142p.

Lin, C. Y.; Abdel - Hamid, T.; Sherif, J. S. Software engineering process simulation model (SEPS). **Jounal of Software Systems**, v.38, n. 10, p.263-277, Oct. 1997.

Lions, J. L. **Inquiry board ariane 5 flight 501 failure.** [online]. <http:// www.esrin.esa.it/htdocs/tidc/Press/ Press96/ariane5rep.html>. set. 1996. Report by the Inquiry Board, Paris, July 1996.

Martin, R. C. **Designing object-oriented C++ applications using the Booch method.** Englewood Cliffs: Prentice Hall, 1995. 368p.

Meyer, B. **Object-oriented analysis.** Englewood Cliffs: Prentice Hall**,** 1992. 257p.

Micro Analysis & Design Simulation Software. **Getting started with MicroSaint for Windows.** Boulder, 1992. 200p.

Miner, R. J.; Rolston, L. J. **MAP/1 user's manual**. version 3.0. New York: Pritsker & Associates, 1986. 275p.

Nance, R. E. **Model representation in discrete event simulation:** the conical methodology. Blacksburg: Virginia Technical. Department. of Computer Science, Mar. 1981. 27p.

Neumann, P. G. Risks to the public in computers and related systems. [online]. **ACM SIGSOFT Software Engineering Notes**, v.9, n.3, p.61-64, Mar. 1990. <http://catless. ncl.ac.uk/Risks>. mar. 1997.

Nicol, J. R.; Wilkes, C. T.; Manola, F. A. Object-orientation in heterogeneous distributed computing systems. **IEEE Computer**, v.26, n.6, p.57-67, June 1993.

Orfali, R.; Harkey D.; Edwards, J. **The essential distributed objects survival guide.** New York: John Wiley & Sons, l996. 534p.

Otte, R.; Patrick, P.; Roy, M. **Understanding CORBA**. Upper Saddle River: Prentice Hall PTR, 1996. 185p.

Ozcam, M. B.; Siddiqi, J. A rapid prototyping environment based on executable specifications. In: IEEE region annual international conference, 10., Singapore, 22-26 Aug. 1994. **Proceedings.** Piscataway: CODEN, 1995. v.2, p. 709-795.

Pegden, C. D.; Shannon, R. E.; Sadowski, R. P. **Introduction to simulation using SIMAN**. Maidenhead: McGraw-Hill, 1990. 380p.

Péralti, M.; Decotigne, J. A design framework for real-time reactive applications. In: IEEE international conference on industrial eletronics, control and instrumentation, 21., Orlando, 6-10 Nov. 1995. **Proceedings.** Los Alamitos: CODEN, 1995. v.1, p. 144-149.

Peterson, J. L. **Petri net theory and the modeling of systems.** Englewood Cliffs: Prentice Hall, 1981. 465p.

Polito, J. **The safeguards network analysis procedure (SNAP):** a user's manual. New York: Pritsker & Associates, 1983. 297p.

Pountain, D.; **Best CASE scenario:** choosing OO methods. [online]. <http://www.byte.com/art/9608/sec16/art.htm>. Byte Magazine, Agu 1996.

Pressman, R. S. **Software engineering - a practitioner's approach**. 3.ed. Singapore: McGraw-Hill, 1992. 793p.

Priste, C. **The QNAP2 language and the object-oriented approach**. Clermont-Ferrand II: Blaise Pascal University, 1991. 23p. (Draft Engineer Report).

Pritsker, A. A. B. **Introduction to simulation and SLAM II.** New York: John Wiley & Sons, 1986. 364p.

Pritsker, A. A. B. **The GASP IV simulation language.** New York: John Wiley & Sons, 1974. 473p.

Rolina, T. **Verification of systems specifications:** a case study in the automotive industry. [online]. <http://www.ilogix.com/fs_about.htm>. abr. 1998. I-Logix, 1998.

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W. **Object-oriented modeling and design.** Englewood Cliffs: Prentice Hall, 1991. 387p.

Sant'Anna, N. M. **Um ambiente experimental para a engenharia de software.** São José dos Campos. 187p. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, 1993. (INPE-5540-TDI/528).

Santos, J. **Tradução automática de modelos informais para especificações formais de sistemas de software**. São José dos Campos. 297p. Tese (Doutorado em Ciência) – Instituto Tecnológico de Aeronáutica, 1996.

Schriber, T. J. **An introduction to simulation using GPSS/H**. New York: John Wiley & Sons, 1991. 274p.

Selic, B. **An efficient object-oriented variation of the statecharts formalism for distributed real-time systems.** [online]. <http://www.objectime.on.ca/>. nov. 1996. Ontário: ObjecTime Limited, 1993.

Selic, B. **Real-time object-oriented modeling**. New York: John Wiley & Sons, 1994. 525p.

Selic, B.; Gulkerson, G.; McGee, J.; Engelberg, L. **ROOM: an object-oriented methodology for developing real-time systems**. [online]. <http://www.objectime.on.ca>. nov. 1996. Ottawa: Bell-Northern Research, 1992.

Shannon, R. E. **Systems simulation: art and science**. Englewood Cliffs: Prentice Hall,1975. 432p.

Shlaer, S.; Mellor, S. J. **Análise de sistemas orientada para objetos.** Trad. de Anna Terzi Giova. São Paulo: Mc Graw Hill,1990. Inglês. 178p.

Soares, L. F. G. **Modelagem e simulação discreta de sistemas**. São Paulo: IME –USP (VII Escola de Computação), 1990. 250p.

Software Engineering Institute (SEI), **Capability model for software.** Pittsburgh**:** Carnigie Mellon University, l99l. 326p. (CMU/SEI -91-TR-24)

Sommerville, I. **Software engineering**. 3.ed. Reading: Addison Wesley, 1989. 458p.

Tate, G. Prototyping: helping to build the right software. **Information Software Technology**, v.32, n.4, p.237-244, Apr. 1990.

Taylor, D. **Object technology - a manager's guide**. Reading: Addison Wesley, 1998. 283p.

Unified Modeling Language (UML). **Notation guide version 1.0.** [online]. <http://www.rational.com>. fev. 1997. Santa Clara: Rational Rose Corporation, Jan. 1997.

Urban, J. E.; Joo, H. Executable specifications for distributed software systems. In: IEEE Computer Society Workshop on Future Trends of Distributed Computing

Systems, 5., Cheju Island, 28-30 Aug. 1995. **Proceedings.** Los Alamitos: CODEN, 1995. v.1, p.257-265.

Waltham, B. BEST/1 - MVS User's Guide; BEST/1 - VM User's Guide; BEST/1 - SNA User's Guide. Vienna: BGS Systems, 1983.

Ward, P. T. The transformation schema: an extension of the data flow diagram to represent control and timing. **IEEE Transactions on Software Engineering**, v.12, n.2, p.17-22, Feb. 1986.

Ward, P. T.; Mellor, S. J. **Structured development for real-time systems**. Englewood Cliffs:Prentice Hall, 1985. 465p.

Welch, P. D. **The statistical analysis of simulation results**: Computer Performance Handbook. Lavenberg: Academic, 1983. 782p.

Wienand, W. **Object-oriented programming:** KISS. [online]. <http://www.pass-consulting.com/passage01/forum1_en.htm>. mar. 1997.

Wirfs-Brock, R.; Wilkerson, B.; Wiener, L. **Designing object-oriented software.** Englewood Cliffs: Prentice Hall, 1990. 396p.

Wortman, D. B.; Duket, S. D.; Seifert, D. J; Hann, R. L.; Chubb, G.P. **The SAINT user's manual**. Menlo Park: Aerospacial Medical Research Laboratory, June 1978. 22p. (AMRL-TR-76-22).

Yourdon, E. **Análise estruturada moderna**. 3.ed. Trad. de Dalton Conde de Alencar. Rio de Janeiro: Editora Campus, 1990. 836p.

Zave, P. An insider evaluation of PAISLey. **IEEE Transactions on Software Engineering**, v.17, n.3, p.212-225, Mar. 1991.

Zeigler, B. P. **Theory of modelling and simulation**. New York: John Wiley & Sons, 1976. 528p.

Zeigler, B. P.; De Wael, L. Towards a knowledge base implementation of multifaceted modeling methodology. **Simulation Series Proceedings**, v.18, n.2, p.42-51, Feb. 1986.