JORGE RADY DE ALMEIDA JUNIOR

STAD - UMA FERRAMENTA PARA REPRESENTAÇÃO E SIMULAÇÃO DE SISTEMAS ATRAVÉS DE STATECHARTS ADAPTATIVOS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

São Paulo
1995

JORGE RADY DE ALMEIDA JUNIOR

STAD - UMA FERRAMENTA PARA REPRESENTAÇÃO E SIMULAÇÃO DE SISTEMAS ATRAVÉS DE STATECHARTS ADAPTATIVOS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

Área de Concentração:
Engenharia de Computação

Orientador
Prof. Dr. João José Neto

1995

Almeida Junior, Jorge Rady

STAD - Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos. São Paulo. 1995 202p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Statecharts adaptativos 2. Engenharia de software I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t

Ao meu pai,

à minha esposa e

à minha mãe (in memoriam)

AGRADECIMENTOS

Ao amigo e orientador Prof. Dr. João José Neto, pela orientação segura e sempre prestativa, possibilitando a concepção e elaboração deste trabalho.

Aos colegas da FDTE e da Escola Politécnica, que de alguma forma colaboraram através de apoio, idéias e colaborações prestadas.

À minha querida mãe que sempre esteve presente espiritualmente na realização deste trabalho.

A todos os meus familiares, pela compreensão e paciência com que por tantos anos suportaram minha distância durante a elaboração deste trabalho.

SUMÁRIO

RESUMO

Este trabalho apresenta uma contribuição para a evolução da notação conhecida como Statechart, através da incorporação de algumas características inéditas, provenientes da teoria de Autômatos Adaptativos.

Um Statechart construído desta forma torna-se capaz de modificar sua configuração em resposta às entradas impostas ao sistema por ele representado.

Desenvolve-se, para tais Statecharts Adaptativos, uma ferramenta de análise e desenvolvimento. Essa ferramenta, chamada STAD, constitui um editor e simulador de sistemas representados por meio de Statecharts Adaptativos.

Essa ferramenta, além de propiciar uma visão prática da teoria, comprovando sua viabilidade, possibilita uma forma bastante útil de difusão destes conceitos, bem como sua utilização no projeto e representação de sistemas reais.

ABSTRACT

This works presents a contribuition to the evolution of Statecharts through the inclusion of some new characteristics, taken from the theory of Adptative Automata

Building Statecharts that way make them able to be a self modifying device, which change in response to their inputs.

STAD has been created as a development and analysis tool for Adaptative Statecharts – modelled system, and it offers edition and simulation capabilities.

Besides provides a practical view of the theory, STAD has been shown helpful in representing real systems, as an excellent tool in their design phase, and for educational purposes.

1. INTRODUÇÃO

Neste capítulo são apresentados os principais pontos que nortearam a realização deste trabalho, expondo-se suas motivações e trabalhos relacionados desenvolvidos, finalizando com a descrição da organização da estrutura desta tese.

1.1 Motivação e Objetivos

A tecnologia na área dos computadores vem apresentando grande evolução, propiciando sua utilização nas mais diversas áreas de aplicação. Os problemas com a segurança foram aumentando e tornando-se críticos à medida que essas aplicações foram incluindo áreas onde uma falha pudesse levar a conseqüências mais drásticas, tais como danos à vida humana e à propriedade.

Atualmente, os controles feitos por computador tendem a ser bem mais complexos, identificando-se sistemas recentes que não poderiam ter sido construídos anteriormente sem a utilização de computadores. Estes são agora utilizados para controlar sistemas de transporte público, plantas industriais e químicas, plantas nucleares, sistemas de armamentos e aero-espaciais e aparelhagem médica.

O aumento do número de acidentes com esses novos sistemas pode ser visto da seguinte forma: os computadores aumentam a complexidade do sistema não apenas pela complexidade inerente do software, mas também porque o software é bastante flexível, permitindo a construção de sistemas mais complexos e de um maior número de tipos de controle.

A utilização dos sistemas de computação nesses campos ocasionou um aumento da responsabilidade atribuída a esses sistemas, levando à necessidade de um maior controle de qualidade em sua produção.

Para se atingir esses objetivos, os problemas ainda não solucionados incluem a adoção de uma metodologia de desenvolvimento de software, o modelamento, análise e projeto da interface homem-máquina e a determinação dos riscos, vitais para o sucesso desses sistemas, bem como uma estratégia adequada de certificação de programas.

Quanto à metodologia de desenvolvimento, é a aplicação de método formais para o projeto de software que vem sendo mais explorada. Os defensores dos métodos formais afirmam que estes podem revolucionar o desenvolvimento de software. Seus opositores dizem que tais métodos são muito difíceis de se aplicar.

Outro aspecto de fundamental relevância nesses sistemas é a existência de uma boa interação homem-máquina, que é importante não apenas nos produtos que o engenheiro de software projeta e implementa para os usuários, mas também para que o próprio engenheiro possa bem projetar, entender e manter programas complexos.

A engenharia de software é uma atividade de projeto, sendo que muitos problemas não estruturados devem ser resolvidos por exploração e eliminação de erros. Isto é especialmente aplicável à interação homem-máquina, pois não há uma teoria completa e fechada sobre o assunto. Diversos métodos e ferramentas desenvolvidas para a engenharia de software tradicional não se aplicam ao software para a interação homem-máquina.

Dessa forma, o objetivo deste trabalho de tese de doutorado é apresentar algumas técnicas utilizadas no desenvolvimento de sistemas de software, destacando-se suas principais etapas e características e desenvolver uma ferramenta para a representação de sistemas com a capacidade de auxiliar os profissionais envolvidos nessa tarefa.

O ideal é que um usuário possa criar uma especificação para o seu sistema ou para o seu software através da utilização de uma ferramenta que automatize a maioria das tarefas necessárias. O ambiente para a utilização dessa ferramenta deve ser o mais amigável possível, o que pode ser conseguido principalmente pela utilização de interfaces gráficas com o usuário.

Assim, a tese se concentrará na pesquisa de métodos formais de especificação e de desenvolvimento de software, com o auxílio de modernas interfaces gráficas com o usuário, de forma a se poder garantir um ambiente de desenvolvimento eficiente.

A idéia é utilizar a representação conhecida como Statechart para representar visualmente as especificações, bem como se poder simulá-las e executá-las, de forma a se obter subsídios sobre sua correção.

1.2. Histórico da Área e Trabalhos Relacionados

O crescente reconhecimento do software como um elemento dos mais importantes nos mais diversos sistemas atualmente desenvolvidos e a elevação dos custos de manutenção decorrentes de suas falhas, vêm criando motivação suficiente para um planejamento eficaz das atividades de desenvolvimento.

Quando do desenvolvimento dos primeiros programas para computadores, praticamente não eram disponíveis técnicas ou métodos formais para o seu aprimoramento. Com a evolução dos sistemas e o conseqüente aumento de custos relativos ao software, realizaram-se estudos visando solucionar os problemas decorrentes dos longos prazos para a sua conclusão, da impossibilidade de detecção de todos os erros e da dificuldade de avaliação do estágio de desenvolvimento de um programa (ICHIKAWA, 1990).

Com o desenvolvimento da engenharia de software, a tarefa de escrever um programa ficou facilitada, pois foram aprimoradas técnicas e procedimentos adequados à especificação, programação, testes e manutenção de software.

Nos últimos anos a velocidade, capacidade de memória e a confiabilidade do hardware dos computadores vêm aumentando significativamente, o que não se pode afirmar com relação ao software. Certamente a confiabilidade do software tem aumentado, mas erros ainda existem como fato normal em praticamente todos os programas, desde os mais simples até os mais complexos.

Ambientes de suporte formais podem ser divididos em duas categorias:

- a primeira contém o ambiente dedicado para um determinado método de projeto;
- a segunda contém as ferramentas de uso geral que podem ser usadas para o suporte formal. Essas ferramentas são mais conhecidas como provadores de teoremas, podendo ainda ser subdivididas em provadores voltados para uma lógica específica e provadores genéricos.

Diversos sistemas têm sido desenvolvidos com a finalidade de se facilitar a representação das informações. Entre eles está o sistema chamado ARIES (JONHSON, 1992), que tem como finalidade representar e apresentar o conhecimento. As principais atividades do analista neste sistema são:

- aquisição das informações sobre o problema por meio de entrevistas e documentos existentes;
- raciocínio: análise e combinação das informações obtidas;
- evolução: modificação dos requisitos e especificações;
- apresentação das informações obtidas e processadas.

Longos tempos de treinamento e a resultante perda de produtividade podem ser parcialmente amenizados por uma máquina virtual que permaneceria a mesma em todas as unidades de manutenção do software. Um componente dessa máquina virtual (SHEPARD, 1992) seria a futura geração de IPSE - "Integrated Project Support Environment". Esse IPSE pode executar automaticamente algumas tarefas, dar uma orientação para outras e retirar dos usuários tarefas administrativas.

Os primeiros experimentos com provadores de teoremas foram dedicados a provas de teoremas matemáticos, e evoluiram para o desenvolvimento formal de software.

O sistema ARGONAUTA (MARANINCHI, 1989) foi projetado para descrever, especificar e verificar sistemas reativos tais como protocolos de comunicação, aplicações de tempo real e interfaces homem-máquina. Baseia-se no ARGOS, que é uma linguagem gráfica cuja sintaxe é uma extensão dos statecharts.

O sistema STATEMATE (HAREL, et al, 1988b) é um ambiente de trabalho gráfico destinado a especificação, análise, projeto e documentação de grandes e complexos sistemas reativos. Permite ao usuário preparar, analisar e depurar de forma diagramática descrições do sistema em desenvolvimento dos pontos de vista da estrutural, funcional e comportamental, sendo que a representação do aspecto comportamental é feita por meio dos statecharts.

Também é apresentado um editor gráfico para a representação de statecharts (BATISTA NETO, 1991), bem como um simulador para a sua execução (MASIERO, 1990). O objetivo dessas duas ferramentas é o de validar statecharts complexos, servindo de apoio aos métodos de especificação de sistemas de tempo real.

Outro ambiente de desenvolvimento descrito (ELIAS, 1992) transforma especificações definidas em termo de statecharts em programas funcionalmente equivalentes da linguagem C. O ambiente encoraja o usuário a manter a descrição de sistemas em níveis mais altos de abstração utilizando-se da notação dos statecharts.

Uma técnica apresentada permite implementar o controle de complexos diálogos de interface com o usuário. Baseia-se na especificação do fluxo de controle dirigido a eventos descrito por um dialeto de statecharts determinístico (LUCENA, 1993). O controle do diálogo é feito por meio de statecharts, que por sua vez é convertido em tabelas manipuladas em tempo real.

Uma axiomatização para os statecharts é apresentada (HOOMAN, 1992), baseando-se em uma sintaxe textual desses últimos. Dessa maneira as propriedades de um statechart podem ser derivadas das propriedades de seus componentes, sem se referir à sua estrutrura interna.

1.3. Estrutura da Tese

Neste primeiro capítulo é feita uma apresentação do trabalho realizado, descrevendo-se suas motivações, um histórico da área e seus trabalhos relacionados, bem como a descrição da organização física do texto que a compõem.

No segundo capítulo são apresentados os principais aspectos que devem nortear o desenvolvimento de um sistema, desde a sua concepção até a fase de depuração e testes. São expostos os passos seguidos desde a análise de um sistema que necessita de desenvolvimento, passando-se pela descrição da importância das especificações, pelos ciclos de vida do desenvolvimento e chegando-se até as ferramentas para o desenvolvimento de sistemas.

No terceiro capítulo são apresentados os principais modelos utilizados na representação de sistemas computadorizados, sendo que após uma análise desses modelos, apresentam-se as razões para a seleção dos statecharts como a ferramenta a ser empregada. São então descritos detalhadamente os statecharts, bem como é feita uma breve descrição da teoria de autômatos, como base para a extensão dos statecharts para a sua forma adaptativa.

No quarto capítulo é descrita detalhadamente a ferramenta desenvolvida para a edição, simulação e execução dos statecharts adaptativos. São expostas as principais características da ferramenta, descrevendo-se suas peculiaridades, suas limitações, bem como alguns exemplos de utilização.

No quinto capítulo são feitas considerações a respeito da importância e da utilidade do trabalho desenvolvido, apontando-se suas utilizações potenciais e relatando-se perspectivas para a realização de trabalhos futuros.

No anexo A é apresentado um manual para a utilização da ferramenta, construída como aplicação à teoria apresentada.

No anexo B é apresentada a estrutura das tabelas presentes no Banco de Dados que implementa a ferramenta desenvolvida.

Finalmente são apresentadas as referências bibliográficas citadas no decorrer do texto e utilizadas na sua elaboração.

2. O DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE

Vários projetos de desenvolvimento de software são caracterizados por problemas que podem levar a atrasos, aumento de custos e insatisfação dos usuários. Esses problemas são de natureza técnica ou gerencial (LIN, 1989)

Diversas variáveis afetam o processo de desenvolvimento de um software, entre elas: nível da força de trabalho, orçamento, prazos, produtividade, número de erros detectados.

Alguns problemas gerenciais são causados por alocação insuficiente de tempo e recursos, alteração dos requisitos do cliente, produtividade menor que a esperada e avaliação inadequada do real progresso do projeto.

No que se refere à metodologia de desenvolvimento, uma corrente que vem ganhando mais destaque é a da aplicação de métodos formais para o projeto de software.

Os defensores dos métodos formais afirmam que estes podem revolucionar o desenvolvimento de software. Seus opositores argumentam que tais métodos são muito difíceis de se aplicar (HAL, 1990).

O fato é que os métodos formais utilizados no desenvolvimento de sistemas de computação são baseados em conceitos matemáticos. Esses métodos formais permitem a utilização de estruturas dentro das quais se pode especificar, desenvolver e verificar programas de uma forma sistemática.

A demanda por software de qualidade vem aumentando, pois as aplicações tendem a se tornar maiores e mais complexas, e por conseqüência, mais difíceis de serem desenvolvidas. As atenções estão mudando do processamento para a simulação e integração de sistemas, de sistemas centralizados para sistemas distribuídos e do texto para gráficos e multimídia (NERSON, 1992).

A competição do mercado exige tempos menores de desenvolvimento, o que pode resultar na entrega de produtos sem os níveis adequados de qualidade.

A importância da portabilidade das aplicações entre diversas plataformas de hardware (também em constante e rápida evolução) e a necessidade de entendimento e aprendizado por parte de usuários finais de diferentes formações, tornam as coisas ainda mais difíceis e complexas.

Dessa maneira, torna-se de fundamental importância uma forma de análise de sistemas, tal que possibilite um desenvolvimento harmonioso das diversas atividades envolvidas. Nos próximos itens são apresentados os principais conceitos envolvidos na análise, bem como os principais tipos de sistemas existentes e os ciclos de vida de desenvolvimento mais utilizados.

Maiores detalhes sobre os conceitos envolvidos no desenvolvimento de sistemas podem ser encontradas em (GENERAL ELECTRIC, 1986), (HAREL, 1992), (PRESSMAN, 1992) e (YOURDON, 1992).

2.1. Análise de Sistemas

Não se pode falar muito sobre a análise de sistemas, sem antes se ter uma definição clara do que seja um sistema. Se se dispuser de um certo conhecimento da teoria de sistemas, ter-se-á melhores condições para se projetar sistemas estáveis e confiáveis.

A palavra "*sistema*" é um dos termos mais utilizados do vocabulário técnico (PRESSMAN, 1992). Fala-se em sistemas políticos, sistemas educacionais, sistemas aviônicos, sistemas de manufatura, sistemas bancários e sistemas metroviários. A palavra, considerada isoladamente, pouco significa. Utiliza-se o adjetivo descritivo do sistema para se entender o contexto no qual a palavra é usada.

Muitas são as definições encontradas, sem no entanto haver um sinônimo para a palavra sistema. Baseando-se em uma das definições encontradas, define-se um sistema baseado em computadores como :

> *Um conjunto ou arranjo de elementos organizados para cumprir algum método, procedimento ou controle pelo processamento da informação.* (PRESSMAN, 1992)

Os elementos de um sistema baseado em computadores e que estão representados na figura 2.1, são os seguintes:

- **Software**: programas, estruturas de dados e a documentação relativa que serve para implementar a lógica, procedimentos ou controles necessários.
- **Hardware**: componentes eletrônicos que implementam a capacidade computacional e dispositivos eletro-mecânicos que possibilitam funções de comunicação com o universo exterior ao sistema.
- **Pessoas**: usuários e operadores do hardware e do software.
- **Base de Dados**: uma coleção organizada de informações cujo acesso é feito via software.
- **Documentação**: manuais, formulários e outras informações descritivas que explicam o uso e/ou operação do sistema.
- **Procedimentos**: ações a serem executadas que definem o uso específico de cada elemento do sistema.

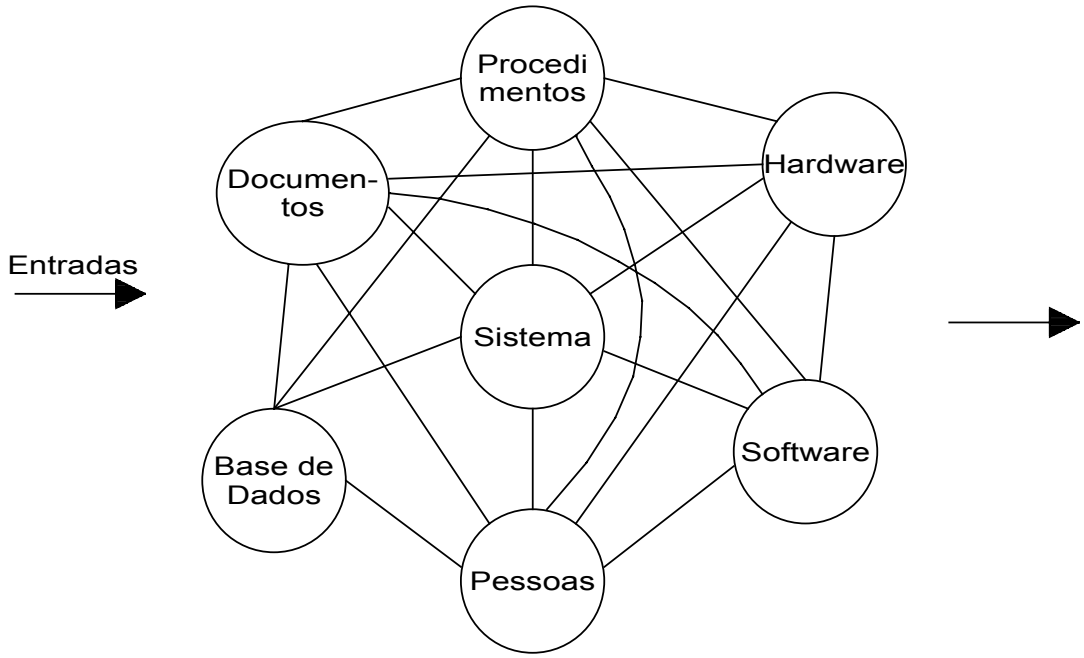


Figura 2.1 - Elementos Componentes de um Sistema

Tais elementos se combinam em uma variedade absurdamente grande de modos, formando os diversos tipos de sistemas existentes.

Um dos maiores problemas encontrados na análise de um sistema refere-se ao estudo do domínio do problema e à identificação de suas características (COAD, 1992). Na prática, os analistas precisam investigar o domínio do problema e as responsabilidades do sistema neste domínio. Um analista precisa especificar requisitos, concisamente reunidos de forma que as pessoas possam ler e entender o que o analista acredita que sejam tais requisitos. Mas a compreensão do domínio do problema é realmente o ponto-chave da análise de sistemas.

Algumas estatísticas ajudam a justificar melhor a busca de soluções para os problemas do desenvolvimento de sistemas de software. Segundo Gibbs (GIBBS, 1994), a cada 6 sistemas de grande porte colocados em operação, 2 outros que estavam em funcionamento são desativados por falhas na operação. O cronograma previsto é ultrapassado em 50% dos casos, bem como 75% dos sistemas apresentam falhas em porções do código que implementam funções não muito utilizadas.

O desafio da análise também requer a comunicação efetiva. Um analista precisa se comunicar durante todo o seu trabalho de análise. Ele tem de se comunicar para extrair informações sobre o domínio do problema e sobre os requisitos do cliente.

Os requisitos de um sistema sempre estarão mudando. A gerência ou os clientes podem impor uma estabilização artificial dos requisitos em um ponto particular do tempo. Mas os requisitos verdadeiros, ou seja, o sistema realmente desejado, continuará a evoluir. Muitos fatores afetam este conjunto instável de requisitos: clientes, competidores, legisladores e técnicos.

Nos próximos itens são apresentados os principais aspectos envolvidos em uma análise de sistemas, sendo que maiores detalhes podem ser encontrados em (COAD, 1992), (MOJDEHBAKHSH, 1994) e (YOURDON, 1992)

2.1.1. O Analista de Sistemas

Um analista de sistemas, ao trabalhar em projetos de desenvolvimento de sistemas, terá de lidar com diversos tipos de pessoas. A equipe variará de projeto para projeto, as personalidades serão extremamente diferentes umas das outras e o número de pessoas com quem se deve interagir certamente apresentará uma grande variação.

Desta forma, um analista de sistemas necessita, mais do que do simples conhecimento da tecnologia de computadores, também de aptidões interpessoais, pois em boa parte do seu tempo, estará trabalhando com pessoas que falam uma linguagem inteiramente diferente da linguagem técnica dos computadores. Assim, é importante saber o que essas pessoas esperam do analista e o que o analista pode esperar delas.

O analista de sistemas é um membro essencial de qualquer equipe de desenvolvimento de sistemas. O analista desempenha diversas funções, entre elas a de separar, de suas verdadeiras causas, os sintomas do problema do usuário. Com o seu conhecimento da tecnologia de processamento, o analista deve auxiliar o usuário a explorar as novas e úteis aplicações dos computadores, bem como novas maneiras de o usuário conduzir seus negócios.

O analista de sistemas muitas vezes se entre usuários, gerentes, programadores e auditores, que freqüentemente se desentendem entre si. É função do analista procurar obter um consenso entre todos esses grupos, e não tentar impor sua opinião.

Assim, o analista de sistemas precisa, não só ser capaz de desenhar fluxogramas e outros diagramas técnicos, mas também ter habilidade com pessoas, para entrevistar usuários, mediar desentendimentos, ter conhecimento de aplicações para compreender a empresa do usuário, além de ser capaz de visualizar o sistema sob diversas perspectivas, de modo que possa subdividi-lo em subsistemas de diversos níveis.

Entre os grupos de pessoas com as quais o analista tem contato, o mais importante é o usuário. Usuário é a pessoa (ou grupo de pessoas) para quem o sistema está sendo construído. Ele é a pessoa que deve ser entrevistada diversas vezes, para se determinar quais características o novo sistema deverá exibir para que seja bem sucedido.

Nos casos de grandes organizações, nem sempre o analista tem contato direto com o usuário final do sistema. Muitas vezes os contatos são mantidos entre agentes contratantes ou outros sistemas administrativos, podendo, assim, gerar mal-entendidos. Assim, aquilo que o verdadeiro usuário gostaria que o sistema fizesse pode ser mal transmitido para o analista de sistemas. Da mesma forma, aquilo que o analista de sistemas imagina que está criando para o usuário também pode ser mal comunicado. Estas imprecisões podem perdurar até que todo o sistema esteja pronto, quando serão muito mais difíceis e onerosas as correções.

Portanto, sempre que possível, o analista deve tentar estabelecer contato direto com o usuário final, mesmo que outras pessoas estejam envolvidas como intermediárias. Se não for possível a comunicação direta com o usuário, então a documentação produzida pelo analista torna-se ainda mais crítica, pois será o seu principal (senão o único) meio de comunicação.

Outro grupo importante é o dos projetistas. Um projetista de sistemas é uma pessoa que recebe a saída do trabalho de análise de sistemas e tem com função transformar uma lista dos requisitos do usuário, em um projeto arquitetural de alto nível, que fornecerá a estrutura com a qual os programadores poderão trabalhar.

Pode ocorrer de o analista de sistemas e o projetista serem a mesma pessoa, ou membros do mesmo grupo em uma organização. Ainda que sejam pessoas diferentes, é importante para o analista e para o projetista de sistemas manterem comunicação constante durante todo o projeto. A razão para isso é a grande interação existente entre a análise e o projeto de sistemas: o analista de sistemas deve fornecer informações suficientemente detalhadas para que o projetista elabore um projeto tecnologicamente bom, e o projetista de sistemas deve fornecer informações suficientemente acuradas para que o analista de sistemas possa dizer se os requisitos do usuário que ele está documentando são tecnologicamente viáveis. Fundamentando-se nas informações recebidas do projetista, a analista pode ter de negociar com o usuário para modificar seus requisitos.

2.1.2. Modelagem de Sistemas

Uma das tarefas mais extensas que o analista de sistemas tem de desenvolver é a modelagem do sistema que o usuário deseja. A finalidade da construção de um modelo é realçar os aspectos mais importantes de um sistema, enquanto que outros aspectos são apenas citados no modelo. Dessa forma, é possível a comunicação com o usuário de uma maneira objetiva, sem se deixar levar pelos aspectos considerados irrelevantes. Se for verificado que o conhecimento do analista sobre os requisitos do usuário estava incorreto, pode-se modificar o modelo, ou mesmo abandoná-lo e construir um novo modelo, se necessário.

Desta forma, as finalidades da utilização de ferramentas de modelagem são:

- manter a atenção principalmente nas características mais importantes do sistema;
- ter a possibilidade de discutir modificações e correções nos requisitos do usuário, com baixo custo e mínimo risco;
- verificar se o ambiente do usuário está documentado de forma que os projetistas e programadores tenham informações suficientes para construir o sistema.

As ferramentas gráficas de modelagem fornecem uma forma clara e de fácil leitura para o analista apresentar aos usuários os principais componentes do modelo, bem como as interfaces entre tais componentes. As ferramentas textuais de modelagem fornecem definições precisas do significado dos componentes e de suas interfaces.

A modelagem de processos de software se volta explicitamente aos fenômenos que ocorrem durante a criação e evolução do software.

O desenvolvimento de software é um desafio para a modelagem de processos devido à criatividade envolvida, tanto na análise dos requisitos e de seu projeto, como na coordenação de equipes durante o desenvolvimento de sistemas complexos.

As pesquisas relativas à modelagem de software visam muitos objetivos, cujas principais categorias são:

- facilitar o entendimento e a comunicação humanos, o que requer a habilidade de compartilhar representações comuns;

- suportar melhorias no processo, tarefa que requer uma base para definição e análise do processo;
- suportar gerenciamento do processo, o que requer uma definição precisa deste, com base na qual o comportamento real do processo possa ser comparado;
- orientação para a automatização do processo, o que requer ferramentas automáticas para manipular descrições do processo;
- suporte para execução automática, o que requer uma base computacional para controle do comportamento em um ambiente automatizado.

Um modelo é uma representação abstrata da realidade, que pode excluir muitos detalhes do mundo real. O propósito de um modelo é reduzir a complexidade quando do entendimento ou interação com um fenômeno, através da eliminação de detalhes que não influenciam significativamente seu comportamento. Um modelo revela aquilo que seu criador acredita ser importante para a compreensão e predição dos fenômenos. A seleção de limites do fenômeno a ser modelado depende do uso que o modelo irá ter.

2.1.3. Metas do Desenvolvimento de Sistemas

Ao se executar uma tarefa de desenvolvimento de um sistema, os aspectos prioritários a serem considerados são: produtividade, confiabilidade e de manutenção desse sistema (YOURDON, 1992). Esses aspectos são descritos a seguir.

2.1.3.1. Produtividade

Um dos problemas mais graves enfrentados na atividade de desenvolvimento de sistemas é o da produtividade. Exigem-se, a cada dia, sistemas mais sofisticados, cada vez com prazos menores de entrega. Os dois aspectos mais importantes deste problema são a demanda reprimida por novos sistemas e o tempo necessário para a construção de cada um deles. Para tentar solucionar estes problemas, vêm sendo utilizadas as seguintes técnicas:

Contratação de maior número de programadores e analistas de sistemas. Com um número maior de profissionais consegue-se obter maior rendimento, tomando-se o cuidado de também se ter um gerenciamento eficiente de todos os envolvidos no desenvolvimento.

Deixar que os usuários desenvolvam seus próprios sistemas. Isto tem sido possível com os computadores pessoais e as linguagens de quarta geração.

Utilização de melhores linguagens de programação. As novas linguagens de quarta geração podem aumentar a produtividade da programação até por um fator igual a 10.

Ataque ao problema da manutenção. Uma técnica adotada é a de migrar os programas antigos (cuja lógica pode já ter sido alterada, tornando-os de compreensão muito difícil) para programas bem estruturados em novas e melhores linguagens. Como auxílio podem ser utilizados pacotes automatizados de documentação, que produzem fluxogramas ou dicionários de dados, por exemplo, diretamente a partir do código, facilitando futuras manutenções.

Observe-se que esta técnica não é muito incentivada, porque pode fazer com que os projetistas deixem de documentar pontos importantes do projeto.

Controles de engenharia de software. Compreende a utilização de técnicas de programação, projeto e análise estruturada, bem como os controles relacionados, como as métricas de software, as provas de correção de programas e o controle de qualidade de software.

Ferramentas automatizadas para desenvolvimento de sistemas. Constituem os CASEs mais recentemente desenvolvidos, que realizam muitas das tarefas mecânicas, geralmente realizadas por analistas e programadores.

2.1.3.2. Confiabilidade

Outro aspecto a ser considerado pelas pessoas envolvidas no desenvolvimento de um sistema é o da confiabilidade. O tempo muito grande dispendido em testes e depuração de erros (tipicamente 50% do tempo total de desenvolvimento de um sistema) e a baixa produtividade podem até ser tolerados se, como resultados, forem obtidos sistemas altamente confiáveis e de fácil manutenção. Isto, no entanto, não é o que vem ocorrendo, ou seja, os sistemas produzidos contêm muitos erros e são muito difíceis de modificar.

O problema está no fato de que erros em programas não são fáceis de corrigir. Quando alguém percebe que o software não está funcionando corretamente, deve ser disparado o processo de correção, ou seja, o programador deve identificar a fonte e a natureza do erro, bem como um modo de corrigi-lo, preferencialmente sem afetar nenhum outro aspecto da operação do sistema.

2.1.3.3. Manutenção

A correção de erros é um dos aspectos da manutenção, que também está vinculada à modificação de um sistema em conseqüência de alterações no hardware, às modificações de aspectos operacionais e às modificações decorrentes de alterações de requisitos.

Para se obter uma boa característica de manutenção de um sistema deve ser utilizado um modelo acurado e atualizado dos requisitos do sistema. Um aspecto positivo é que as especificações funcionais evoluiram de páginas e páginas de puro texto para um formato gráfico, inicialmente feito à mão, e, mais recentemente, gerado e mantido por ferramentas automatizadas de projeto, facilitando sobremaneira a tarefa de manutenção.

2.2. Especificações

As especificações têm um papel central no desenvolvimento de software, pois definem todas as características requeridas do software a ser implementado, formando o ponto de partida de qualquer processo de desenvolvimento de um programa.

As especificações são também um importante meio de comunicação entre o usuário e o projetista, podendo até desempenhar o papel de um contrato formal, que pode ser utilizado para se verificar se o sistema implementado corresponde ao que foi solicitado.

Para definir as características de um sistema de forma precisa e concisa, é necessário que as especificações sejam escritas em uma linguagem formal e fortemente expressiva. Para uma análise imediata das conseqüências das especificações e para uma avaliação precoce, é inclusive desejável que tais especificações sejam executáveis.

Desde que a correta operação do software é definida pelos requisitos, e que estes são formalizados pela especificação, devem ser utilizados todos os meios

para sua validação, considerando-se tanto os requisitos implícitos, como os explícitos.

O processo de definição dos requisitos, destinado a resolver estes conflitos, deve ser feito por meio de uma técnica de análise que capture adequadamente as diferentes perspectivas, seus atributos e suas relações.

A análise dos requisitos de um sistema permite o estabelecimento das especificações, que são a base dos esforços subseqüentes. As atividades realizadas consistem na análise de:

- **Exigências do Sistema**: começa com o entendimento das informações que o sistema irá processar. Concorrentemente, o fluxo de informações e os requisitos operacionais são definidos;
- **Ambiente do Usuário**: é estudado para determinar como o sistema será utilizado;
- **Sistemas Similares ou Interfaces**: sistemas, subsistemas, procedimentos, ferramentas e técnicas já existentes são avaliadas para determinar sua influência no desenvolvimento de um novo sistema. Sistemas e subsistemas existentes pedem interfaces e organização de dados. Procedimentos relacionados podem impor restrições ao projeto. Ferramentas disponíveis podem afetar estimativas de custo e cronograma.

Uma especificação de software deve descrever o comportamento desejado de um futuro sistema de software, isto é, a especificação deve formar um modelo conceitual do sistema.

Tradicionalmente as especificações vêm sendo escritas em linguagem natural, mas estas estão perdendo terreno para as linguagens de formais de especificação, cujas vantagens residem em prevenir mais facilmente problemas de perda, ambigüidade ou inconsistência de informações.

Se a linguagem de especificação formal for executável, há uma vantagem adicional, que é a representação do comportamento do software a ser implementado. O comportamento do software, dentro de seu ambiente, pode ser avaliado antes de ele realmente existir. Isto permite que se faça uma validação prévia, o que aumenta a correção e a confiabilidade do software, reduzindo o custo e o tempo de desenvolvimento.

Especificações executáveis também podem servir como protótipos que permitam experiências com diferentes requisitos, ou mesmo com possíveis evoluções futuras do sistema.

O detalhamento dos requisitos é a elucidação, análise e classificação dos requisitos de software e do sistema. Em modelos recentes de ciclo de vida do software, o detalhamento dos requisitos é uma atividade maior e contínua, cuja função mais importante é a de especificar um sistema que corresponda às reais necessidades do usuário.

Os principais aspectos referentes à especificação de sistemas podem ser encontrados em (GABRIELIAN, 1991), (GIRGIS, 1992), (KOTONYA, 1992), (PALMER, 1992), (PORISINI, 1991), (POTTS, 1994), (SIDDIGI, 1994) e (WILLIAMS, 1994).

2.2.1. Problemas das Especificações

As especificações feitas até o final dos anos 70 eram constituídas por documentos muito grandes e feitas simplesmente no idioma corrente. Era preciso ler a especificação do começo ao fim para poder compreendê-la. Havia muitas redundâncias, com a mesma informação sendo repetida em diversas partes do documento. Além disso possibilitavam a adoção de interpretações ambíguas pelos usuários, projetistas, analistas e programadores. Por todos esses aspectos, qualquer manutenção na especificação tornava-se um trabalho muito extenso.

Por mais de uma década vem ocorrendo insatisfação com as estratégias convencionais para o desenvolvimento de sistemas e um crescente número de propostas para substituí-las por novos métodos mais formais. Isto se deve ao fato de que os métodos de desenvolvimento convencionais não mais comportam o tamanho e a complexidade dos sistemas atualmente implementados (HATLEY, 1988).

A inadequação dos métodos de desenvolvimento se tornou evidente, inicialmente, em sistemas comerciais e de negócios, tendo sido em seguida detectada também em outras áreas.

A preocupação com a correta especificação de sistemas de computação tem origem devido, principalmente, aos seguintes fatores:

- **obsolescência**: com o "envelhecimento" do software já instalado surge a necessidade de engenharia reversa e de técnicas de atualização para manter esses sistemas ainda em operação;
- **segurança**: tanto a vida das pessoas, quanto a integridade física dos sistemas, depende cada vez mais de programas de computador, fazendo com que o interesse pela qualidade do software. assumisse lugar primordial no desenvolvimento de sistemas.
- **complexidade**: aplicações como processamento paralelo tornam mais importante a tarefa de verificação e validação no que se refere à escalação de tarefas e sincronização;
- **condições críticas**: a utilização de software em sistemas que possam causar danos físicos ou materiais (afetando a segurança) levam a uma necessidade maior do emprego de técnicas formais de especificação;
- **reutilização**: leva à necessidade de revisão de especificações;
- **realizabilidade**: com a utilização da prototipagem rápida, que pode demonstrar a possibilidade, ou não, de realização de um sistema, também se faz necessária a utilização de especificações confiáveis para a avaliação dos protótipos;

Tendo em vista esses problemas, foram sendo desenvolvidas técnicas que permitissem a estruturação do projeto e da programação, que gradualmente vêm sendo adotadas por grande parte das empresas. Especificações geradas de forma estruturada se apoiam em ferramentas gráficas de análise, complementadas por informações textuais. Convém que sejam particionadas de modo tal que partes individuais da especificação possam ser lidas independentemente das outras. A redundância deve ser a mínima possível, para facilitar, quando necessário, o trabalho de alteração de requisitos.

2.2.2. Formalização de Especificações

Os métodos formais permitem que se especifique, desenvolva e verifique um sistema baseado em computadores, através da aplicação de uma notação matemática rigorosa. A utilização de uma linguagem de especificação formal proporciona meios para especificar um sistema de tal modo que sejam obtidas a consistência, a completeza e a correção desejadas.

Quando um método formal é utilizado durante o desenvolvimento de um sistema, provê mecanismos para eliminar a maioria dos problemas que ocorrem, como por exemplo ambigüidade, incompleteza e inconsistência, que podem ser descobertos mais facilmente.

Quando um método formal é utilizado no projeto de um sistema, pode revelar falhas que só seriam detectadas durante o teste e a depuração. Na fase final do desenvolvimento de software, um método formal pode assegurar a correta implementação do sistema.

A razão pela qual os métodos formais podem proporcionar soluções eficientes para o problema da especificação e projeto é que eles fazem com que se considere o problema do software de uma maneira análoga a um cálculo algébrico ou a uma prova de geometria analítica.

Os métodos formais utilizados no desenvolvimento de sistemas de computadores são técnicas matematicamente baseadas para descrever as propriedades de um sistema. Esses métodos formais provêm estruturas dentro das quais se pode especificar, desenvolver e verificar sistemas de uma forma metódica.

Um método pode ser considerado formal se tiver uma base matemática, tipicamente uma linguagem de especificação formal. Esta base fornece os meios para definir precisamente noções como consistência, completeza, especificação e correção.

Por razões de consistência, as informações presentes em determinados pontos da especificação não devem ser contraditos em outra parte da mesma. A consistência é assegurada pela prova matemática de que fatos iniciais possam ser mapeados em afirmações posteriores dentro da especificação.

A completeza é difícil de ser atingida mesmo com métodos formais. Alguns aspectos podem ser esquecidos e outros podem ser intencionalmente omitidos para que o projetista tenha mais liberdade na época da implementação.

Os métodos formais não devem ser vistos como uma panacéia, mas sim como um passo que deve ser dado para colocar a engenharia de software sobre uma base sólida, como outros campos da engenharia.

Muito daquilo que é dito a respeito dos métodos formais baseia-se em afirmações, não em fatos. Daí algumas opiniões acerca dos métodos formais seriam exageradas, adquirindo o status de mitos. Os mitos mais comuns, referentes aos métodos formais, são (HAL, 1990):

- os métodos formais podem garantir que um software seja perfeito;
- os métodos formais funcionam de forma a provar que um programa esteja correto;
- apenas os sistemas críticos podem se beneficiar com seu uso;
- envolvem aspectos matemáticos complexos;
- aumentam o custo de desenvolvimento;
- são incompreensíveis para os clientes;
- ninguém os utiliza em projetos reais.

Os métodos formais fornecem ferramentas poderosas. No lugar dos mitos acima apresentados, podem ser colocados os seguintes fatos (HAL, 1990):

- os métodos formais são muito úteis na detecção de erros nas primeiras fases do ciclo de vida do software (definição dos requisitos e das especificações), podendo praticamente eliminar certas classes de erros;
- fazem o projetista raciocinar bem mais sobre o sistema proposto;
- são úteis para quase todas as aplicações;
- são baseados em especificações matemáticas, muito mais simples de se entender do que programas;
- podem diminuir o custo de desenvolvimento;

- podem auxiliar a compreensão do produto pelos clientes;
- podem ser usados com sucesso em projetos práticos da indústria.

Maiores detalhes a respeito da formalização de especificações podem ser encontrados em (GERHART, 1990), (GERHART, 1994), (MEYER, 1985), (WING, 1990), (FUCHS, 1992) e (NOTA, 1992).

2.2.3. Linguagens de Especificações

Várias pesquisas têm sido feitas com relação a linguagens de especificação formais. A disponibilidade dessas linguagens permite que uma fase de testes seja realizada logo após, ou até mesmo concorrentemente com a fase de especificação dos requisitos. Além disso, é possível utilizar essas linguagens para rapidamente se obter protótipos, permitindo a obtenção de informações do comportamento do sistema desejado, inclusive em resposta a eventos externos.

As pesquisas apontam na direção de uma linguagem de alto nível que permita a execução direta de especificações formais. Testar a especificação de um sistema significa verificar as dependências entre os possíveis comportamentos do sistema e as características dos sinais externos e estímulos produzidos pelo ambiente.

Uma linguagem de especificação formal é usualmente composta por três componentes primários: uma sintaxe que define a notação específica com a qual a especificação é representada; uma semântica que define o universo de objetos que podem ser usados para descrever o sistema e um conjunto de relações que define as regras que indicam quais objetos satisfazem as especificações.

A sintaxe formal de uma linguagem de especificação deve permitir que os requisitos ou o projeto sejam interpretados de apenas um modo, eliminando a ambigüidade que pode ocorrer quando uma linguagem natural ou uma notação gráfica é utilizada.

2.2.4. Reutilização

A reutilização de especificações envolve a transformação de uma rede de domínio e de conhecimento que representa uma solução, sugerindo que a reutilização de especificações durante a análise de requisitos é uma forma de solução de problemas análogos.

O potencial dessa analogia é o de recuperar o conhecimento de um domínio e aplicá-lo a outros domínios.

A reutilização de especificações envolve dois problemas: recuperar a especificação correta, desenvolver e adaptar essa especificação para o novo domínio.

O sucesso na reutilização de especificações pode ajudar a superar as dificuldades iniciais de engenheiros de software durante os primeiros estágios do desenvolvimento do software.

A reutilização de especificações incentiva uma utilização maior de prototipação na análise de requisitos. A prototipação, por sua vez, pode encorajar avaliações de especificações mais freqüentes, implicando benefícios na reutilização de especificações.

A reutilização de especificações deve melhorar a produtividade no desenvolvimento de software, proporcionando aos projetistas maior rapidez durante a fase de análise de requisitos. Pode também beneficiar na melhoria da qualidade das especificações, bem como aumentar o conhecimento e experiência do engenheiro para solucionar problemas análogos.

Maiores detalhes a respeito da reutilização de especificações podem ser obtidos em (DIAZ, 1991), (HAREL, 1988a) e (MAIDEN, 1992).

2.2.5. Métodos Estruturados

A análise estruturada é composta por técnicas de modelagem dos elementos e do fluxo de informações. Um sistema baseado em computadores é representado por uma simples transformação de informações. A chave é representar a informação fornecida como entrada e a informação produzida como saída.

Os métodos estruturados envolvem ferramentas e técnicas de modelagem que iluminam certos aspectos do sistema desejado durante o processo de especificação. Diferentes métodos são apropriados para diferentes fases da especificação.

Os métodos estruturados são potencialmente úteis para o desenvolvimento de qualquer tipo de sistema, tais como os de tempo real, de processamento de dados ou de controle de processos. O aspecto importante a se considerar é que eles são sistemas e não apenas um conjunto de componentes de hardware e de software. Os componentes devem ser analisados em conjunto, inter-relacionando-se entre si de modo a cumprir a função para a qual foram projetados.

A especificação deve definir qual é o problema que o sistema deve resolver (os requisitos) e a forma como o sistema deve ser estruturado (a arquitetura). Há a necessidade de capturar qual é o problema e considerar as opções para se encontrar a sua melhor solução. Para estabelecer o problema e obter sua solução constroem-se dois modelos do sistema: o modelo de requisitos e o modelo de arquitetura.

O modelo de requisitos é construído como um modelo tecnologicamente independente dos requisitos essenciais do sistema, e descreve aquilo que o sistema deve fazer. É constituído por diagramas de fluxo de dados e de controle, especificação de processos e de controles, especificações de tempo, e um dicionário de requisitos.

O modelo de arquitetura é um modelo tecnologicamente dependente da estrutura do sistema, e descreve a forma como o sistema deve ser projetado. É constituído por diagramas de fluxo, diagramas de interconexão de arquitetura, especificações de módulos de arquitetura e sua interconexão, e por um dicionário de arquitetura.

Os métodos estruturados incluem ferramentas que assistem em certas partes do processo de desenvolvimento de sistema, não substituindo ou eliminando as práticas usuais de gerenciamento, que devem continuar existindo.

2.3 Principais Tipos de Sistemas

Existem muitos tipos diferentes de sistemas, sendo que praticamente em todas as atividades realizadas pelo ser humano há um sistema ou um componente de algum sistema. Dessa forma, é conveniente organizar os principais tipos de sistemas em categorias, e como os sistemas de interesse, neste trabalho, são os de processamento, pode-se dividi-los nas seguintes categorias: naturais e feitos pelo homem, on line, tempo real, de apoio à decisão e baseados no conhecimento (YOURDON, 1992).

Alguns princípios gerais, de particular interesse para o tratamento sobre os tipos de sistemas, são os seguintes:

Quanto mais especializado é um sistema, menos capaz ele é de se adaptar a circunstâncias diferentes. Quanto mais um sistema for de emprego geral, menos otimizado será para uma situação específica. Porém quanto mais um sistema for otimizado para uma situação específica, menos adaptável ele será a nova circunstâncias.

Quanto maior for um sistema, maior o número de seus recursos que serão destinados à manutenção diária. Um sistema de grande porte exige, habitualmente, enormes esforços nas áreas de verificação de erros, cópias, manutenção, segurança e documentação.

Os sistemas fazem parte de sistemas maiores e podem ser divididos em sistemas menores. Este princípio sugere uma maneira de organizar um sistema que se deseja desenvolver, através de sua divisão em sistemas menores. Sugere ainda que a definição do sistema que se deseja desenvolver é arbitrária — pode-se escolher um sistema ligeiramente menor ou ligeiramente maior. A escolha do escopo de um sistema, e da cuidadosa definição do que faz e do que não faz parte do sistema, é uma atividade bastante importante.

Os sistemas crescem. Isto ocorre com muitos dos sistemas desenvolvidos. Por exemplo, um sistema de informações cresce para incluir mais software do que estava previsto originalmente, mais dados, mais funções e mais usuários.

Nos itens seguintes são descritos os principais tipos de sistemas existentes, cuja classificação baseou-se na descrição de Yourdon (YOURDON, 1992). Outras informações podem ser obtidas em (CRAIGEN, 1994) e (LEVESON, 1990).

2.3.1. Sistemas Naturais e Feitos pelo Homem

Os sistemas, em sua grande maioria, não são feitos pelo homem. Sistemas são encontrados na natureza, servindo, de modo geral, a seus próprios propósitos. É conveniente dividir esses sistemas em duas subcategorias: sistemas físicos e sistemas vivos.

Dentre os sistemas físicos podem ser citados, por exemplo: sistemas estelares (galáxias, sistemas solares), sistemas geológicos (rios, montanhas) ou sistemas moleculares (organizações de átomos). Os sistemas físicos merecem atenção, pois às vezes o homem tenta modificá-los. Além disso, os sistema feitos pelo homem devem interagir harmoniosamente com os sistemas físicos.

Fazem parte de sistemas vivos animais e plantas, incluindo os seres humanos. Algumas das propriedades e características dos sistemas vivos podem ser utilizadas para auxiliar a ilustrar e compreender melhor os sistemas feitos pelo homem, pois muitas vezes podem ser utilizadas analogias existentes entre ambos os tipos de sistemas.

Dentre os sistemas feitos pelo homem podem ser citados, por exemplo: sistemas sociais (leis, costumes), sistemas de transporte (redes rodoviárias, petroleiros), sistemas de comunicação (telefone, telex), sistemas de manufatura (fábricas, linhas de montagem) e sistemas financeiros (contabilidade, controle de estoques).

Atualmente, dos sistemas citados, a maior parte se utiliza de computadores. Na verdade, muitos deles não poderiam manter-se sem a ajuda dos computadores, embora a origem de alguns desses sistemas seja anterior ao surgimento dessas máquinas. Alguns ainda não estão totalmente computadorizados, outros têm um computador como componente, mas podem conter também mais componentes não computadorizados.

A decisão a tomar se um sistema feito pelo homem deve ou não ser computadorizado, baseia-se na análise e na modelagem do comportamento do sistema.

2.3.2. Sistemas On Line

Pode-se definir sistemas on-line como sendo aqueles que recebem como entradas dados trazidos diretamente do local em que foram gerados. Além disso, são também os sistemas cujas saídas, ou resultados do processamento, são dirigidos diretamente para onde sejam necessários (YOURDON, 1992).

Em um sistema on-line, geralmente, os dados são inseridos no sistema de processamento e dele recuperados de forma remota, ou seja, os usuários do sistema interagem com o computador através de terminais que podem estar, ou não, distantes do(s) computador(es) de controle do sistema.

Outro aspecto a destacar em um sistema on-line é o fato de que os dados armazenados (arquivos ou banco de dados) são normalmente organizados de forma tal que os dados individuais possam ser recuperados ou modificados rapidamente, sem a necessidade de acesso suplementar a outros dados do sistema.

Devido à característica de que os sistemas on-line precisam, geralmente, recuperar dados rapidamente (para obter respostas a consultas dos usuários), torna-se muito importante projetar os arquivos e os bancos de dados de maneira que sejam tão eficientes quanto possível. Muitas vezes ocorre que os cálculos executados por um sistema on-line sejam relativamente simples, enquanto que a estrutura e organização dos dados se mostram bastante complexas. Daí a importância das ferramentas de modelagem de dados.

A decisão de se construir um sistema que seja ou não on-line é uma decisão de implementação, e não algo que deva ser determinado pelo analista de sistemas, mas sim pelos implementadores do sistema. Como essa decisão tem, em geral, grande impacto sobre os usuários do sistema, na maioria dos casos estes últimos desejam e devem participar dessa discussão.

2.3.3. Sistemas de Tempo Real

Os sistemas de tempo real são muitas vezes considerados como variações dos sistemas on-line. No entanto, é importante estabelecer uma distinção entre eles.

Um sistema de tempo real deve ter uma reação imediata a estímulos ocorridos no mundo real. A reação imediata significa receber os estímulos, executar o processamento e gerar as saídas necessárias em um tempo menor ou igual ao exigido pela característica do sistema sob análise.

Alguns exemplos de sistemas de tempo real são sistemas de controle de processos (monitoração e controle de refinarias, indústrias químicas), sistemas de caixa automático (caixas eletrônicos de bancos), sistemas de obtenção de dados de alta velocidade (recebimento de dados de satélites), sistemas de orientação de mísseis, sistemas de comutação telefônica e sistemas de monitoração de pacientes.

Em um sistema de tempo real as atenções estão voltadas para se verificar se o computador pode responder com rapidez suficiente aos estímulos do ambiente, sob o risco de o sistema sair fora de controle. Por exemplo, se o sistema não for suficientemente rápido, os dados que chegam podem ser perdidos, ou um míssil pode sair completamente de sua trajetória pretendida. Em constraste, um sistema on-line que não possua a capacidade de responder rapidamente a uma solicitação, pode apenas "irritar" seus usuários, não causando maiores danos, tanto ao ambiente, quanto aos usuários.

Nos sistemas on-line a interação é, geralmente, feita com pessoas, enquanto que nos sistemas de tempo real, a interação tanto pode ser feita com pessoas, quanto com o ambiente, que é normalmente autônomo, e muitas vezes hostil.

Um maior detalhamento de sistemas de tempo real é encontrado em (GOMAA, 1986).

2.3.4. Sistemas de Apoio à Decisão

Muitas empresas vêm concentrando a sua atuação nos sistemas de apoio à decisão, que constituem um avanço em relação aos sistemas de processamento de ações (sistemas de pagamento, de processamento de pedidos e de contabilidade), dominantes desde que se começou a se automatizar o processamento de informações.

Os sistemas de apoio à decisão não tomam decisões si mesmos, mas auxiliam os profissionais de uma organização a tomarem decisões mais acertadas e bem informadas sobre vários aspectos da operação. Estes sistemas têm como característica serem passivos, ou seja, não funcionam de uma forma autônoma, sendo ativados externamente, apenas quando isso se faz necessário.

Alguns exemplos de sistemas de apoio à decisão são: programas de planilhas eletrônicas, sistemas de análise estatística e programas de previsões mercadológicas. Outra atividade executada pelos sistemas de apoio à decisão é que eles, além de recuperar e apresentar os dados solicitados, também executam análises matemáticas e estatísticas sobre os mesmos. Têm também, geralmente, a capacidade de apresentar as informações sob várias formas gráficas (tabelas, diagramas), bem como relatórios convencionais.

2.3.5. Sistemas Baseados no Conhecimento

Os sistemas baseados no conhecimento, ou sistemas especialistas, são relativamente novos na indústria do processamento. Em um sistema baseado no conhecimento há grande quantidade de informações a serem utilizadas em determinada tarefa. Os sistemas especialistas são uma espécie de sistemas baseados no conhecimento, embora os dois nomes sejam, muita vezes, empregados indistintamente.

Pode-se definir um sistema especialista como sendo um programa que tem armazenado o conhecimento e a capacidade que permitem que funcione como especialista. Atuar como especialista significa, por exemplo, que o programa consiga atingir o nível de desempenho de um médico, em diagnose e terapêutica, ou de pessoas de grande experiência, em exercer tarefas de engenharia, tarefas científicas ou administrativas. O sistema especialista é um auxílio intelectual de alto nível para o seu usuário.

Um sistema especialista pode, em geral, exibir o caminho seguido, ou seja, explicar as linhas de raciocínio que conduzem a seus resultados. De maneira similar, também podem explicar porque rejeitaram certas linhas de raciocínio e escolheram outras. Essa é uma das principais característica dos sistemas especialistas, sendo muito importante, pois a utilização de um sistema especialista depende de sua credibilidade junto aos usuários, e esta credibilidade surge pelo fato de seu comportamento ser transparente e auto-explicativo.

Convém observar que os sistemas especialistas devem se tornar, gradualmente, componentes cada vez mais importantes dos sistemas típicos de processamento, embora sua utilização, atualmente, ainda não seja tão intensa.

2.3.6. Sistemas Transformacionais e Sistemas Reativos

Um sistema reativo caracteriza-se por ser dirigido por eventos, tendo de reagir continuamente a estímulos internos e externos. O comportamento de um sistema reativo é o conjunto de seqüências permitidas de eventos de entrada e saída, condições e ações, talvez com alguma informação adicional como restrições de tempo.

Já os sistemas transformacionais são descritos por uma relação entre valores de entrada e de saída. Eles lêem valores de entrada, produzem, eventualmente de forma não determinística, um valor de saída e terminam o processamento. Têm uma estrutura linear e apenas seus estados inicial e final são de interesse para o usuário.

Os sistema reativos não processam uma função, mas sim realizam uma interação contínua com o ambiente. Sistemas reativos são encontrados em qualquer parte, especialmente no mundo real. Entre eles incluem-se, por exemplo, relógios digitais, aparelhos de televisão, software de interação com o usuário.

Dessa forma, as ferramentas de descrição tradicionais, já bastante estudadas e utilizadas para a formalização de sistemas transformacionais, não se adaptam tão bem para a descrição de sistemas reativos.

Em um sistema reativo é importante saber-se o momento em que uma nova entrada é ativada, pois o estado interno do sistema nesse instante é importante para que seja determinada a reação ao novo estímulo. Daí porque mais atenção deve ser dada aos estados intermediários, o que não ocorre com os sistemas tranformacionais.

A entidade elementar de observação de um sistema reativo é o evento. O ambiente envia eventos ao sistema para disparar o processamento, o sistema reage ao ambiente, enviando ou gerando eventos. Eventos são também o meio de comunicação entre as partes do sistema. Devido ao fato de se querer especificar sistemas reativos de uma maneira discreta, os eventos são sinais discretos que ocorrem em um ponto no tempo. Eventos têm duração zero e, conseqüentemente, o programa que gera os eventos também deve ter duração zero. Eventos são gerados nas transições de um estado para outro e como não consomem tempo, todo o tempo é consumido nos estados.

Uma razão importante para esta escolha é que, em sistemas reativos, as entradas podem ocorrer a qualquer instante e sempre deve estar claro em qual estado o sistema se encontra. Desde que as transições têm duração zero, não há períodos de transição entre um determinado estado e uma entrada que ocorra, ou seja, o próximo estado que o sistema deve assumir sempre está muito bem definido.

2.4. Ciclo de Vida de Sistemas

Um dos objetivos pelos quais se deve adotar um ciclo de vida para o desenvolvimento de um sistema é que isto proporciona uma definição mais clara das atividades a serem executadas, ou seja, as atividades são organizadas de maneira a se otimizarem os trabalhos e seus resultados. Outra finalidade é a de se obter consistência entre muitos sistemas desenvolvidos na mesma organização, podendo-se ainda introduzir pontos de verificação para o controle gerencial de decisões.

As atividades mais importantes em cada fase do ciclo de vida do software podem ser definidas pelos produtos finais de cada fase. Na fase de especificação dos requisitos, a principal preocupação é com a análise dos requisitos e com a definição de interfaces. Na fase de projeto, a ênfase é dada no projeto em si e na qualidade do projeto, visando à confiabilidade, eficiência e manutenção.

Embora cada fase tenha um enfoque principal, todas atividades têm continuidade em todas etapas do ciclo. O que muda é o grau de atenção ou importância atribuídos a cada atividade, dependendo da etapa do ciclo.

Por exemplo, a codificação de um módulo propriamente dito tem início apenas após o detalhamento do projeto desse módulo. No entanto, há atividades de codificação anteriores a serem realizadas, tais como, por exemplo, o planejamento do método de codificação, a aquisição de ferramentas de software, ou mesmo o teste de alguns tipos de algoritmos.

A substituição do modelo convencional clássico de desenvolvimento de software por modelos alternativos, tais como a prototipação ou a reutilização de software, tem um profundo efeito na atividade de gerenciamento da configuração de software (BERSOFF, 1991).

O propósito do gerenciamento é tratar as alterações através do processo de desenvolvimento de software. As alterações ocorrem nas seguintes etapas:

- durante a **criação**, quando é feita a especificação dos requisitos;
- durante o **desenvolvimento**, quando são feitos o projeto, a codificação e os testes;
- durante as **melhorias**, quando os requisitos são alterados;
- durante as **adaptações**, quando erros são detectados.

Os principais modelos de ciclos de vida utilizados para o desenvolvimento de sistemas, tais como o Ciclo Clássico, a Reutilização e a Prototipação são detalhados em (BERSOFF, 1991), (ROOK, 1990) e (PRESSMAN, 1992).

2.5. Certificação de Sistemas

Para se validar um programa de computador é necessário submetê-lo a um sistema de testes rigorosos. No entanto há um problema fundamental, de que o número de combinações de entrada a serem testadas é muito grande, tornando impraticável a execução de todos os possíveis casos de teste (MUSA, 1989).

Dessa forma, para validar um software deve-se estruturar o problema de modo que se possam aplicar conceitos estatísticos, de modo a se obter medidas de confiabilidade de software. Os conceitos básicos da confiabilidade de software são:

- **Execução**: em cada execução há um mapeamento entre os conjuntos das variáveis de entrada e de saída, utilizando-se um determinado tempo de processamento. A especificação desse mapeamento é um requisito para a execução;
- **Erros e falhas de software**: quando o software é executado e as saídas não estão de acordo com os requisitos, isto significa a ocorrência de uma falha. Para evitar que essa falha ocorra em uma execução subseqüente, o programa deve ser modificado, ou seja, o erro que causou a falha deve ser corrigido;
- **Perfil operacional**: uma determinada entrada presente em um determinado ponto da execução, é considerada como tendo sido selecionada aleatoriamente. No entanto, determinadas entradas ocorrem com maior freqüência que outras. A especificação dessas entradas e sua freqüência relativa é chamada perfil operacional do programa;
- O **tempo de execução** é a dimensão básica da medida de confiabilidade (tempo médio decorrido entre falhas). Dada uma seleção aleatória de entradas, é o tempo de execução que reflete a utilização do software.

Como exemplos de técnicas a serem aplicadas na certificação de um programa de computador estão os testes e a verificação e validação, abaixo descritos, conforme apresentado em (ACKERMAN, 1989), (DUKE, 1989), (DUNHAM, 1989) e (HALLANG, 1994).

2.5.1. Verificação e Validação

Verificação é o processo pelo qual se determina se o software realiza o que está especificado. É acompanhada pela realização de testes individuais para cada tarefa específica do software.

Validação é uma tarefa mais ampla que visa determinar se o sistema do qual o software faz parte está de acordo com os requisitos.

Os requisitos impostos na verificação e validação são determinados pela condições críticas do sistema. A razão para se identificar essas condições é estabelecer as conseqüências de uma falha e determinar o nível de controle de configuração e de testes a que o sistema deve ser submetido. Isto também serve para estabelecer limites nos recursos destinados ao processo de verificação e validação.

Embora as linhas gerais do processo de verificação e validação possam ser estabelecidas, os detalhes devem ser adaptados a cada aplicação.

2.5.2. Testes

Os objetivos desta fase são satisfeitos pela execução de uma série de testes que começam com a revisão do projeto e culminam com testes de validação do software.

O objetivo do teste de software é forçar o mesmo a falhar, de forma que erros possam ser mais facilmente detectados. Ou seja, um teste de sucesso é aquele que detecta erros (GENERAL ELECTRIC, 1986).

Uma falha de software consiste na ocorrência de um erro que pode ser classificado em:

- não-conformidade com as especificações;
- não-conformidade com a documentação;
- violação de restrições de projeto;
- uma situação em que o software não atenda a alguma característica desejada pelo usuário, embora tal característica possa não fazer parte das especificações do sistema.

A aplicação de medidas de confiabilidade durante o teste de um sistema fornece informações quantitativas do processo de validação. Assim, o teste de um sistema tem duas funções distintas:

- Validar as funções executadas pelo software, de acordo com os requisitos;
- Melhorar a qualidade do software completo, detectando falhas e corrigindo os erros que levam às falhas.

A medida da confiabilidade está relacionada principalmente à primeira função, pois a detecção e correção de falhas não faz parte do escopo das medidas, embora colabore para a concentração das atenções em determinadas partes, baseado no perfil operacional (MUSA, 1989).

Assim, para se efetuar a medida de confiabilidade de software durante o teste de um sistema é necessário:

- Selecionar casos de teste na mesma proporção das freqüências especificadas no perfil operacional;
- Registrar o tempo de execução entre falhas;
- Continuar o teste, corrigindo as falhas detectadas, até que o nível de confiabilidade atingido seja melhor ou igual ao especificado.

2.6. Interface Homem-Máquina

A interface homem-máquina é o meio de que o usuário dispõe para interagir com o produto desenvolvido. Do ponto de vista do usuário é a interface homem-máquina que permite, por exemplo, a um piloto, comandar um moderno avião, ou a um cliente, transferir dinheiro entre agências bancárias. A interface é, na maioria dos casos, a visão que o usuário tem do sistema com o qual está interagindo. Se ela for bem projetada, o usuário se sentirá mais inclinado a utilizar e aceitar melhor o software desenvolvido.

Ao se considerar um sistema interativo com a utilização de software, o fator humano tem diferentes significados. Inicialmente pode-se pensar na percepção visual, na memória humana e no raciocínio dedutivo e indutivo. Em outro nível deve-se entender o usuário e seu comportamento. Finalmente, devem ser entendidas as tarefas que o sistema realiza para o usuário e as tarefas que o sistema requer que o usuário realize como parte da interação homem-máquina.

A interface com o usuário constitui-se no mecanismo pelo qual é estabelecido o diálogo entre a máquina e o homem. Se os fatores humanos foram corretamente considerados, o diálogo deverá ser fácil e um ritmo adequado poderá ser estabelecido entre o usuário e o programa. Se esses fatores não foram considerados, o sistema será quase sempre visto como hostil.

Os estilos da interação homem-máquina obedecem a uma gama de opções que estão vinculadas à evolução histórica dos computadores e de seus dispositivos de acionamento. Como o hardware se tornou mais sofisticado, as opções para os estilos de interações cresceram. Assim que os engenheiros de software aprenderam mais sobre os fatores humanos e seu impacto no projeto de interfaces, surgiram as modernas interfaces orientadas a janelas, baseadas em dispositivos de apontamento e clique. Os benefícios advindos dessas interfaces foram:

- Diferentes tipos de informação podem ser mostrados simultaneamente, permitindo ao usuário chavear de contexto sem perder a conexão visual com outro trabalho.
- Diversas tarefas iterativas distintas são disponíveis por meio de menus "pull-down", permitindo ao usuário realizar, de maneira simples, tarefas de controle e diálogo.
- A utilização de ícones gráficos, botões e barras de rolagem, entre outros, reduziu a quantidade de digitação, o que aumenta a eficiência da interação com a máquina.

Nos próximos itens são descritos alguns dos principais aspectos que envolvem Interfaces Homem-Máquina, tendo como base as descrições de (HARBERT, 1990), (HOLLAND, 1992), (MARCUS, 1993), (MORSE, 1993), (MYERS, 1989), (MYERS, 1992), (NIELSEN, 1993a) e (NIELSEN, 1993b)

2.6.1. Evolução das Interfaces Homem-Máquina

Convém destacar como ocorreu o desenvolvimento das interfaces com o usuário ao longo do tempo, desde os primeiros sistemas onde havia algum tipo de automação (MANDELKERN, 1993):

- Na década de 40 não havia uma interface propriamente dita, mas sim uma interação física direta com mecanismos de emulação de sistemas reais;
- Na década de 50 as interfaces eletro-mecânicas tornaram-se comuns. Neste ponto a interface homem-máquina evoluiu para a sua primeira linguagem alfanumérica de comunicação;
- Na década de 60 começaram a ser adotadas soluções puramente eletrônicas. Mostradores eletrônicos alfanuméricos ou tubos de raios catódicos gráficos foram conectados a computadores de grande porte ("mainframes") em centros de processamento de dados. Este foi o início das interfaces gráficas modernas;
- Na década de 70 surgiram as telas gráficas de alta resolução em ambientes de computação distribuídos. Os primeiros dispositivos de apontamento ("mouses"), ícones e interfaces baseadas em "menus" começaram a aparecer nos laboratórios de pesquisa;
- Finalmente, na década de 80 surgiram as interfaces gráficas com o usuário ("GUIs - Graphical User Interfaces") em computadores pessoais e estações de trabalho. Isto foi possível devido ao desenvolvimento e barateamento dessas máquinas, com alto desempenho da CPU, vários megabytes de memória, disco rígido de grande capacidade e monitores coloridos de alta resolução, além de recursos de interconexão por meio de redes;
- A década de 90 está vendo a capacidade dos computadores aumentar continuamente, tornando possível o avanço das interfaces gráficas com o usuário. A potência de processamento aumenta exponencialmente, enquanto que o custo cai na mesma proporção (MANDELKERN, 1993). O mesmo ocorre com a capacidade de armazenamento. Os dispositivos de apontamento e posicionamento, bem como os monitores, também devem sofrer grande processo de evolução. Todos esses avanços tornarão possíveis novas formas de interação homem-máquina, mas apenas se as ferramentas de desenvolvimento de software puderem aproveitar todo esse potencial de hardware.

O meio mais seguro para gerar interfaces de boa qualidade é testar protótipos com os usuários, modificando o projeto com base em seus comentários.

A interface homem-máquina é a janela do usuário para o processo. O usuário deve ter todas as ferramentas necessárias para responder, manipular e analisar todas as informações do processo.

A redefinição da interface homem-máquina é um fator importante para a aceitação dos computadores por um número cada vez maior de usuários não-especialistas em informática. O surgimento de sistemas operacionais gráficos possibilitou vantagens não só em termos de interfaces, mas também de entrada/saída.

2.6.2. Especificação de Interfaces

O software necessário para gerenciar um sistema de interface com o usuário contribui em muito na complexidade e custo de desenvolvimento, pois deve interagir tanto com o usuário, quanto com outros módulos de software do sistema.

A interface entre módulos de software é relativamente estática quando comparada com a interface entre usuário e computador, que não fica bem definida até muito perto da época de instalação do software. A interface de uma aplicação com o usuário pode se alterar em resposta a solicitações de mudanças por parte do usuário.

Várias ferramentas foram desenvolvidas para ajudar na criação de componentes de interface com o usuário. Estas ferramentas e seus ambientes, denominados de sistemas de gerenciamento de interface com o usuário, permitem, em geral, especificar graficamente as interfaces.

Com o passar dos anos as interfaces como o usuário passaram a cuidar de tarefas cada vez mais complexas, tornando-se mais acessíveis e tratáveis. Como exemplo, citam-se os antigos editores de texto orientados a linha, em comparação com os sistemas de editoração atualmente existentes.

A complexidade da interface é função do número de usuários potenciais (do próprio programador, indo até um grande número de usuários com pouco conhecimento), exigindo também diferentes estratégias de implementação, com os correspondentes custos aumentando progressivamente.

O melhor paradigma para a criação de software para a interação homem-máquina é um modelo de comunicação e uma rápida prototipação que suporte a co-evolução das especificações e da implementação. A comunicação entre usuários, projetistas e implementadores é vital (FISHER, 1989).

As linhas gerais seguidas quando da especificação de interfaces homem-máquina são (PRESSMAN, 1992):

- Ser consistente: utilizam-se formatos consistentes para a seleção de menus, entrada de comandos e exibição de dados, entre outros.
- Oferecer realimentação compreensível: é oferecida ao usuário uma realimentação visual e/ou sonora para assegurar que uma comunicação bidirecional seja corretamente estabelecida.
- Questionar sobre qualquer ação destrutiva (ação de exclusão) não convencional.
- Permitir fácil reversão das ações realizadas: por exemplo, uma ação de exclusão.
- Reduzir a quantidade de informações que devem ser memorizadas entre ações: não se espera que o usuário necessite lembrar listas ou códigos.
- Dividir as atividades pela sua função e organizar a tela dentro deste espírito.
- Prover recursos de auxílio on-line sensíveis ao contexto.
- Mostrar apenas as informações que sejam relevantes ao contexto em que estiverem sendo exibidas.
- Utilizar formas de apresentação dos dados que permitam uma rápida assimilação da informação (gráficos ou cartas).
- Produzir mensagens de erro úteis e com significado.
- Permitir ao usuário personalizar suas entradas.
- Desativar comandos que sejam inadequados ao contexto que estiver sendo exibido.

Se o ambiente se altera, o problema também pode se alterar, tornando uma determinada estratégia não apropriada. Essa alteração pode causar aumento muito grande dos custos, forçando uma mudança de estratégia, que pode ser evolucionária (ajustes em uma estratégia) ou revolucionária (alteração total de uma estratégia).

2.6.3. Interfaces Gráficas

As interfaces gráficas como o usuário - GUIs ("Graphical User Interfaces") são uma das mais revolucionárias mudanças na evolução dos sistemas de computação modernos. Em curto espaço de tempo (10 anos) a expectativa de interação evoluiu das "teletypes" para o ambiente Windows, ícones, menus e "mouses". Esta evolução melhorou o acesso e a utilização dos computadores ao público em geral (MANDELKERN, 1993).

O mundo das interfaces não é perfeito, completo e estático. Ainda há muito a ser feito em termos de aumentar a produtividade dos usuários, padronizar operações entre as diversas arquiteturas e adaptar a interface homem-máquina a aplicações não tradicionais.

As GUIs típicas representam a estrutura de um escritório, com arquivos, documentos e ações típicas desse ambiente. Contudo, esse "escritório virtual" não é apropriado para outras aplicações, tais como a sala de controle de uma planta nuclear. Em tais casos é desejável prover a interface de metáforas que sejam mais representativas ao paradigma físico do ambiente.

Para fazer interfaces melhores é necessário dispor-se de boas ferramentas de construção, de forma a tornar o desenvolvimento de GUIs mais fácil, mais rápido e menos caro para os projetistas e eventualmente para usuários finais. Este problema não é único dos GUIs, mas também se aplica ao desenvolvimento de software.

As GUIs modernas são orientadas a objeto. O usuário primeiro acessa o objeto de interesse e o modifica, operando sobre ele. A razão de se utilizar a orientação a objetos é de representar os objetos de interesse ao usuário para permitir manipulação direta (ícones são bons para representar objetos, mas pobres para representar ações).

2.7. Ferramentas para o Desenvolvimento de Sistemas

Uma ferramenta para o desenvolvimento de sistemas pode ser definida como algo que substitua ou auxilie o trabalho manual do analista, projetista, programador e até do usuário final. Assim, algumas das características que uma ferramenta para auxílio ao desenvolvimento de sistemas deve possuir são (LIN, 1989):

Linguagem de programação de alto nível, incluindo-se as linguagens de quarta geração, que permitem o uso de sentenças semelhantes à linguagem natural, de alto nível.

Listagens de referências cruzadas, programas de impressão de relatórios, e outros programas utilitários que oferecem informações auxiliares e estatísticas sobre seus programas.

Ferramentas de teste e de depuração, simuladores e congêneres, que prestam informações ao programador sobre o comportamento dinâmico de seu programa em tempo de execução.

Pacotes de controle de código fonte, que impedem que um programador faça modificações não autorizadas em versões oficiais dos programas.

Representação gráfica para os processos do sistema.

As ferramentas com que o analista e o projetista podem contar estão relacionadas principalmente com o desenvolvimento eficiente dos modelos em que se fundamenta a construção de um sistema. Nos próximo itens são descritos os principais aspectos que envolvem essas ferramentas, baseando-se em (JACOB, 1985), (JACOBSON, 1993), (LIN, 1989), (NERSON, 1992) e (SNYDER, 1993).

2.7.1 Ambientes de Desenvolvimento

Os projetistas de software necessitam de técnicas sistemáticas e integradas para aprimorar a qualidade de seus produtos. A produtividade e a qualidade de um projeto são aperfeiçoadas com a utilização de um ambiente de suporte integrado, facilitando um melhor aproveitamento de pessoas e recursos.

Utilizar uma linguagem de programação por si só é útil para algumas aplicações. Contudo, para outras aplicações, tais linguagens não proporcionam uma utilização com a eficiência esperada (RAEDER, 1985).

Para auxiliar nestas tarefas, foram criados ambientes de desenvolvimento que suportam, de diversas maneiras, o processo de desenvolvimento de programas, melhorando e aumentando a capacidade das linguagens.

Muitos ambientes de desenvolvimento contêm diversas ferramentas de software, cada uma destinada à execução de tarefas específicas. A integração ocorre em dois níveis: interno, integrando-se os componentes do sistema, e externo, referindo-se à interface com o usuário.

O objetivo fundamental de um ambiente de desenvolvimento de software é o de automatizar as tarefas de produção deste. Um componente importante nesta automatização é a chamada e controle das ferramentas do ambiente (GARLAN, 1990).

Ambientes de desenvolvimento modernos têm se concentrado em mecanismos implícitos de chamada de ferramentas. Em vez de o usuário acionar explicitamente uma determinada ferramenta, é o ambiente de desenvolvimento que garante a chamada de ferramentas, no tempo devido. Por exemplo, ao fim da edição de um módulo, o ambiente acionaria automaticamente o compilador.

2.7.2. Orientação a Objetos

Sistemas orientados a objetos são construídos com modelos também orientados a objetos, cuja modelagem impõe sua descrição, suas relações e a forma como interagem dinamicamente. De posse de um modelo, podem ser discutidos os objetos e suas relações.

A construção de um sistema orientado a objetos impõe a construção de modelos que devem representar a implementação do sistema. Como a implementação é descrita em uma linguagem orientada a objetos, tanto o programador, quanto o computador podem compreendê-la.

Na tecnologia orientada a objetos, a atenção muda do trabalho de desenvolvimento para o produto, ou seja, para o sistema resultante.

Um objeto é uma entidade cuja função é prover serviços aos seus clientes. A natureza exata do cliente e do serviço dependem de cada sistema particular. Em geral, um cliente é uma pessoa ou um programa e um serviço pode ser

qualquer atividade que possa ser realizada, decorrente do atendimento de uma requisição do cliente.

Quando comparado à tecnologia tradicional, o desenvolvimento orientado a objetos é um processo sem emendas, ou seja, não há troca de paradigmas entre as fases do ciclo de desenvolvimento do software. Princípios uniformes se aplicam ao longo do processo de desenvolvimento.

2.7.3. Programação Gráfica

A obtenção de informações é muito melhor e mais rápida se feita através de figuras e gráficos do que em textos escritos, devido aos seguintes fatores (RAEDER, 1985):

- A visão fornece **acesso instantâneo e aleatório** para qualquer parte de uma figura, enquanto que o acesso a um texto é seqüencial;
- Uma imagem geralmente consegue uma **trasferência de informações bem mais rápida** do que um texto, tanto no acesso, como na decodificaçào;
- A utilização de figuras, que representam o mundo real, para ilustrar idéias abstratas **facilita o raciocínio** sobre o problema;
- Desde que os **objetos estejam representados visualmente**, não é necessário se referir e eles pelos nomes;
- Enquanto as **figuras refletem o mundo real**, o texto apenas faz referência ao mesmo;

Os modos gráficos disponíveis atualmente em microcomputadores e estações de trabalho podem eliminar a necessidade de conversão dos algoritmos em seqüências de comandos tradicionais das linguagens atuais. O desafio é, dessa forma, obter uma linguagem de programação visual natural e de uso conveniente.

As linguagens de programação visual podem ser divididas em duas categorias. Na primeira, o objeto a ser representado é estático (menu, "layout" de uma tela, esquema). Na segunda categoria o problema é a representação de entes abstratos (seqüência de tempo, hierarquia, comandos condicionais).

Uma linguagem gráfica deve considerar a forma estática e a forma como a mesma é utilizada para descrever a interface com o usuário. Uma representação visual para este propósito deve descrever o comportamento externo

(visível ao usuário) de forma precisa, sem deixar dúvida quanto ao comportamento do sistema para todas as entradas possíveis.

Outra característica de uma linguagem gráfica é a separação entre a função e sua implementação, sendo descrito o comportamento da interface de forma precisa. A representação visual deve ser mais fácil de entender e deve necessitar menor esforço para ser produzida do que o software tradicional.

2.7.4. Técnicas de Quarta Geração

O termo Técnicas de Quarta Geração (4GT - "fourth-generation techniques") compreende diversas ferramentas de software com uma propriedade comum: permitem ao projetista de software especificar alguma característica em alto nível. A ferramenta gera o código-fonte baseada na especificação fornecida. O paradigma 4GT para a engenharia de software focaliza sua atenção na habilidade de se especificar um sistema em um nível que seja o mais próximo possível da linguagem natural.

Um ambiente de desenvolvimento de software que suporte o paradigma 4GT inclui, normalmente, algumas das seguintes ferramentas: linguagem não procedural para consultas a bases de dados, geração de relatórios, manipulação de dados, definição e interação de tela, geração de código e gráficos de alto nível.

Com relação à utilização das técnicas de quarta geração há duas correntes distintas. Uma que afirma que há uma redução substancial no tempo de desenvolvimento de sistemas, acompanhado de um aumento significativo da produtividade. Outra corrente rebate, dizendo que o paradigma 4GT não tem linguagens tão fáceis de serem assimiladas e utilizadas, não produz códigos eficientes e que os sistemas são difíceis de serem mantidos.

De qualquer maneira, as técnicas de quarta geração vêm tendo utilização crescente, e a tendência demonstrada é que este paradigma vá contribuir cada vez mais com a demanda em ascensão constante para a produção de software.

3. MODELOS PARA A REPRESENTAÇÃO DE SISTEMAS

Um modelo é uma representação de um sistema complexo que se deseja estudar. Modelos de sistemas são construídos de forma a focalizar suas características mais importantes e possibilitar o estudo de alterações e correções a custos relativamente baixos e com risco reduzido.

Um modelo bem elaborado permite que a comunicação entre o projetista de sistemas e o usuário se faça de maneira mais clara e precisa. Outra razão para a construção de um modelo é para que se entenda o ambiente do usuário e o documente, de maneira que projetistas e programadores possam construir o sistema desejado.

Uma definição mais formal para um modelo é a seguinte:

> *Um modelo define como um processo deve ser feito no contexto das metas, objetivos e restrições de um projeto. O modelo contém seqüências de tarefas que devem ser executadas para se obter os objetivos desejados.* (KRASNER, 1992)

Há diversos tipos de modelos: narrativos, protótipos e vários modelos gráficos. Cada usuário, geralmente, necessita de diferentes ferramentas de modelagem, seja pela experiência anterior ou porque não se adapta a certos tipos de diagramas. Cada projeto necessita da utilização de diferentes ferramentas de modelagem, tendo em vista os padrões de documentação impostos por organizações externas. E diferentes tipos de sistemas podem exigir modelos diferentes para realçar adequadamente as diversas possíveis características importantes. Muitos sistemas exigem modelos múltiplos, pois cada modelo enfatiza um número limitado de aspectos do sistema, em detrimento de outros.

É conveniente que o modelo a ser utilizado atenda às seguintes características (YOURDON, 1992):

- Ser gráfico, com adequado detalhamento textual de apoio. A escolha por um modelo gráfico apoia-se no fato de que uma figura adequadamente escolhida contém, na maioria da vezes, uma quantidade de informações muito grande, de forma concisa e compacta. Isso não significa que uma figura possa descrever, necessariamente, tudo sobre um sistema, sendo necessária uma complementação com documentos textuais de apoio.

- Permitir que o sistema seja visualizado de forma hierárquica, na forma "top-down". Isto se aplica para sistemas de médio e grande porte, onde se torna muito difícil, de outra forma, se entender o sistema como um todo. Desse modo convém que os modelos possibilitem retratar partes individuais do sistema de forma isolada, além de se contar com uma maneira de se passar facilmente de uma parte a outra do modelo do sistema.
- Ter mínima redundância. É desejável que o modelo se mantenha atualizado sempre que possível. Assim, se o sistema se alterar, o modelo deve permitir modificações. Se apenas um aspecto local do sistema for modificado, é desejável modificar apenas um aspecto local correspondente do modelo. Se forem necessárias múltiplas mudanças existe a possibilidade de elas não serem feitas, ou de que sejam feitas de maneira incorreta, tornando o modelo não aderente ao sistema real.
- Ajudar o leitor a prognosticar o comportamento do sistema. O modelo deve conter informações suficientes que permitam ao leitor determinar o estado que o sistema assumirá quando determinadas condições ocorrerem.
- Ser transparente para o leitor. É desejável que um modelo seja tão natural que o leitor não possa distinguir com facilidade entre a representação de um sistema, e o próprio sistema.

Em (CURTIS, 1992), (KRASNER, 1992), (LINDLAND, 1994) e (SHEPARD, 1992) essas características são descritas de uma forma mais detalhada.

Nos próximos itens são descritas as principais notações usualmente empregadas para a representação de sistemas e é feita uma avaliação das características de cada uma delas. Também é detalhada a representação conhecida como Statecharts, bem como uma extensão destes, que são os Statecharts adaptativos, propostos nesta tese.

3.1. Principais Notações para a Representação de Sistemas

As principais características que fazem com que uma notação possa representar um sistema são (DAVIS, 1988):

- Quando a técnica é propriamente utilizada, a especificação dos requisitos do sistema torna-se útil e compreensível para os usuários que não entendam de computação, bem como serve efetivamente como base para o projeto e os testes do sistema.

- São previstas verificações com a finalidade de prevenir ambigüidades, incompletezas e inconsistências.
- Incentivo ao analista a pensar e escrever em termos da visão externa do produto e não de seus componentes internos.

Auxílio na organização da informação.

- Adequação para cada aplicação particular.

Nos próximos itens são apresentadas as principais, dentre as inúmeras notações existentes, empregadas para a análise e a conseqüente representação de sistemas. A classificação das notações apresentadas baseou-se em (YOURDON, 1992) e (ROMAN, 1993).

3.1.1. Diagrama de Fluxo de Dados

O Diagrama de Fluxo de Dados - DFD – é uma ferramenta de modelagem que permite imaginar um sistema como sendo uma rede de processos funcionais interligados por transições e armazenamento de dados.

Um DFD é uma notação gráfica que mostra o fluxo de informação e a transformação aplicada sempre que os dados se movem da entrada para a saída. Um DFD pode ser particionado em níveis que representam sucessivos detalhamentos da informação da funcionalidade do sistema.

Os DFDs são utilizados principalmente em sistemas cujas funções sejam de fundamental importância e mais complexas que os dados por ele manipulados (PRESSMAN, 1992).

Os componentes típicos de um DFD são:

- **Processos**: são mostrados como círculos no diagrama. Representam as diversas funções individuais que o sistema executa. Funções transformam entradas em saídas.
- **Fluxos**: são denotados por arcos direcionados, representando as conexões entre os processos, indicando a informação que os processos exigem como entrada e/ou as informações que eles geram como saída.
- **Depósitos de Dados**: representados por duas linhas paralelas ou por uma elipse, mostrando coleções de dados que o sistema deve manter na memória por um

período de tempo. Concluída a construção do sistema, os depósitos existirão, tipicamente, como arquivos ou bancos de dados.

- **Terminadores**: representados por retângulos, mostram as entidades externas com as quais o sistema se comunica. Os terminadores são, tipicamente, indivíduos, grupos de pessoas, sistemas externos de computação ou organizações externas.

A figura 3.1 apresenta um exemplo de um DFD.

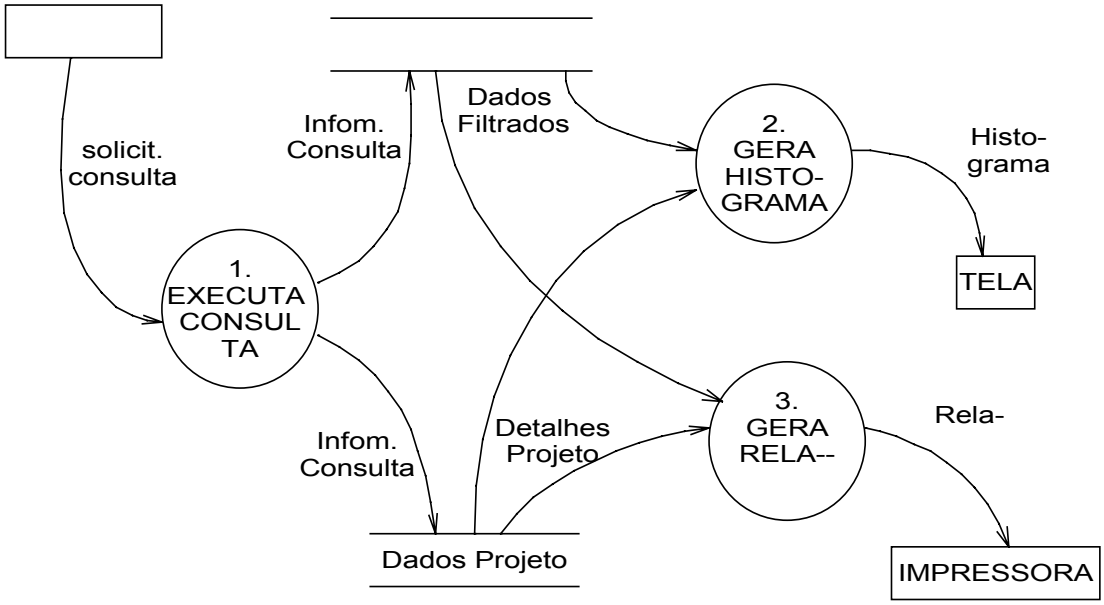


Figura 3.1 - Exemplo de um Diagrama de Fluxo de Dados

Podem ser citadas algumas diretrizes básicas que auxiliam na tarefa de se desenhar um DFD:

- Escolher nomes significativos para os processos, fluxos, depósitos e terminadores.
- Numerar os processos.
- Desenhar um DFD tantas vezes quantas forem necessárias até obter uma boa estética.
- Evitar fazer DFDs complexos demais.
- Certificar-se de que o DFD seja internamente consistente, além de manter a consistência com DFDs de outros níveis de detalhamento

3.1.2. Dicionário de Dados

O DFD, acima apresentado, permite a obtenção de uma visão geral dos principais componentes funcionais do sistema, não fornecendo, porém, detalhes sobre esses componentes. Assim, com a finalidade de se detalhar quais informações são transformadas e da forma como são transformadas, são utilizadas ferramentas de suporte textual de modelagem, tais como os Dicionários de Dados e a Especificação de Processos. A figura 3.2 apresenta um exemplo de um Dicionário de Dados.

companhia =	nome_cia + CGC_cia + IE_cia + endereço_cia
nome_cia =	{caractere válido}
endereço_cia =	{caractere válido}
CGC_cia =	{dígito válido}
IE_cia =	{dígito válido}
caractere válido =	[A-Z/a-z/'/-/ /]
dígito válido =	[1-0]

Figura 3.2 - Exemplo de um Dicionário de Dados

O Dicionário de Dados, é constituido por uma listagem organizada de todos os elementos de dados pertinentes ao sistema, contendo definições precisas e rigorosas para que possam ser conhecidas todas as entrada, saídas, componentes de depósitos e cálculos intermediários. O Dicionário de Dados contêm os seguintes elementos:

- O significado dos fluxos e depósitos mostrados nos DFDs.
- A composição de pacotes agregados de dados que se movimentam pelos fluxos, isto é, pacotes complexos (como o endereço de um cliente) que podem ser divididos em itens mais elementares (como cidade, estado e código postal).
- A composição dos pacotes de dados nos depósitos.
- Os valores e unidades relevantes de partes elementares de informações dos fluxos e depósitos de dados.
- Os detalhes dos relacionamentos entre os depósitos realçados em um diagrama Entidades-Relacionamentos.

3.1.3. Especificações de Processos.

Existem diversas ferramentas que podem ser utilizadas para produzir uma Especificação de Processos: tabelas de decisão, linguagem estruturada, condições pré/pós, fluxogramas e diagramas de Nassi-Shneiderman, entre outras. Embora a linguagem estruturada seja a ferramenta mais utilizada, pode ser empregado qualquer método, desde que a especificação de processos seja expressa de uma forma tal que possa ser facilmente verificada pelo usuário e pelo analista de sistemas e ser efetivamente comunicada às diversas audiências envolvidas.

As ferramentas utilizadas para a Especificação de Processos são a seguir descritas.

a) Linguagem Estruturada

Uma linguagem estruturada se constitui em um subconjunto da linguagem normal com algumas restrições quanto ao tipo de sentenças que podem ser utilizadas e à maneira como essas sentenças podem ser reunidas. Seu objetivo é obter um equilíbrio entre a linguagem de programação formal e a informalidade e legibilidade da linguagem natural, utilizada normalmente. A figura 3.3 apresenta um exemplo de uma especificação de processos representada em linguagem natural.

1. SE não houver trem nos próximo 5 circuitos de via ENTÃO

    a) Libera código de velocidade máxima para o trem

    b) Fecha todas as portas do trem

    c) Emite aviso sonoro de partida

Figura 3.3 - Exemplo de Especificação de Processos representada em Linguagem Natural

b) Condições Pré/Pós

As Condições Pré-Pós constituem um modo de se descrever a função a ser executada por um processo, sem que seja necessário detalhar o algoritmo ou o procedimento que será empregado. Essa visão é útil quando se deseja permitir ao programador que explore alguns dos muitos algoritmos disponíveis , sem se envolver pessoalmente em detalhes e, principalmente, em discussões com o usuário sobre os métodos de desenvolvimento a serem utilizados nesses algoritmos.

Uma condição é composta por duas partes principais: as pré-condições e as pós-condições. As pré-condições descrevem tudo que deve ser

verdadeiro antes que o processo inicie seu funcionamento. Costumeiramente as pré-condições descrevem quais entradas devem estar disponíveis, que relacionamentos devem existir entre as entradas e entre estas e os depósitos de dados.

As pós-condições descrevem, de forma análoga, os fatos que devem ser constatados quando o processo terminar sua tarefa. As pós-condições habitualmente descrevem as saídas que serão geradas pelo processo, os relacionamentos que existirão entre os valores de saída e os valores originais de entrada e entre os valores de saída e os valores contidos nos depósitos de dados.

A figura 3.4 apresenta um exemplo de uma especificação de processos representada em condições pré/pós.

Pré-condição 1

Não há trem ocupando os próximos 5 circuitos de via

Pós-condição 1

Libera código de velocidade máxima, fecha todas as portas e emite

aviso de partida

Figura 3.4 - Exemplo de Especificação de Processos representada em Condições Pré/Pós

c) Tabelas de Decisões

Há casos em que tanto a linguagem estruturada, quanto as condições pré/pós não são adequadas para a confecção de uma especificação de processos. Isso é especialmente verdadeiro quando o processo deve produzir alguma saída ou executar ações com base em decisões complexas. Se as decisões forem baseadas em diversas variáveis, e se estas puderem assumir muitos valores diferentes, então a lógica expressa pela linguagem estruturada ou pela condições pré/pós será provavelmente tão complexa que o usuário não a entenderá. Neste caso, a notação mais recomendável será então a Tabela de Decisões, pois esta relaciona todas as entradas relevantes com todas as ações relevantes, de uma forma bastante completa, evitando omissões.

Uma vantagem da utilização de Tabelas de Decisões é que elas não impõem qualquer forma de implementação em particular. Isto é, quando o analista entrega a Tabela de Decisões ao projetista/programador, este tem total liberdade de escolha em termos de estratégia de sua implementação.

A figura 3.5 apresenta um exemplo de uma especificação de processos representada em uma tabela de decisões.

	CV1	CV2	CV3	CV4	CV5	CV6	CV7	CV8
Ocupação		S	N	N	N	N	N	S
Veloc. Máx. (Km/h)	44	44	44	30	30	30	0	0
Aceleração	S	S	S	N	N	N	N	N

Figura 3.5 - Exemplo de Especificação de Processos representada em uma Tabela de Decisões

d) Fluxogramas

Os Fluxogramas permitem representar a lógica de procedimentos de um programa de computador, sendo que alguns analistas os utilizam como meio de documentação da especificação de processos.

Os Fluxogramas podem ser empregados para descrever uma lógica detalhada, desde que sejam utilizados símbolos equivalentes às construções da linguagem estruturada. Para criar um Fluxograma estruturado deve-se organizar sua lógica com combinações aninhadas dos símbolos de Fluxogramas, sob o risco de o fluxograma se tornar extremamente complicado e difícil de ser interpretado.

A figura 3.6 apresenta um exemplo de uma especificação de processos representada em um fluxograma.

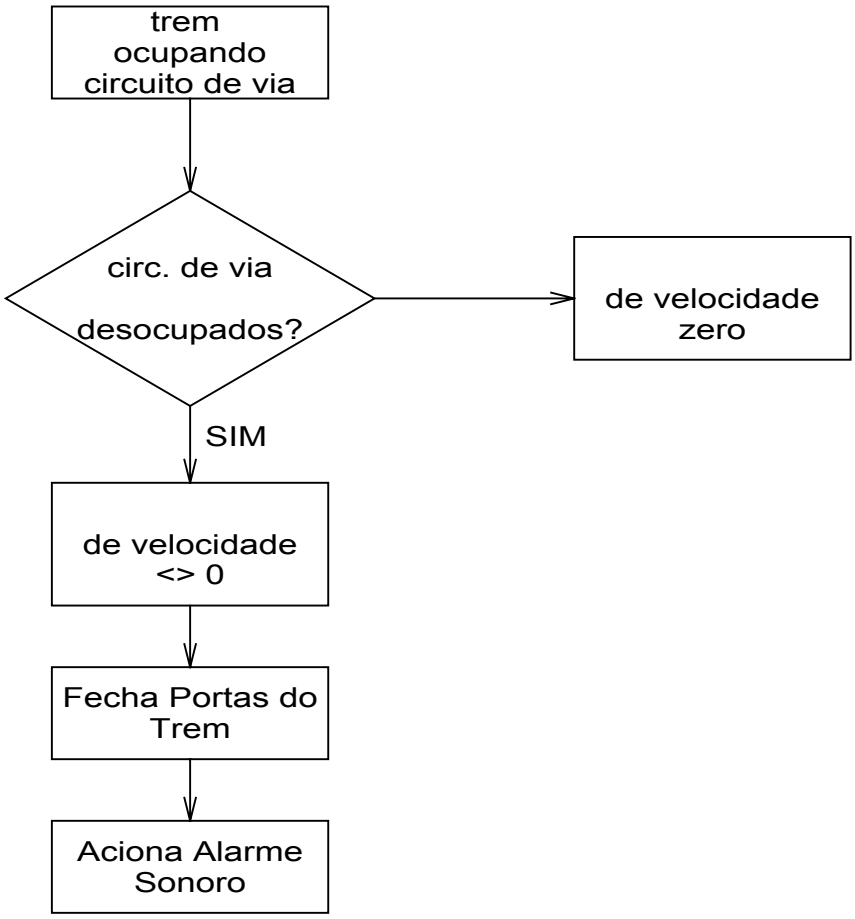


Figura 3.6 - Exemplo de um Fluxograma

e) Diagramas de Nassi-Shneiderman

Os diagramas de Nassi-Shneiderman são geralmente mais organizados, mais estruturados e mais abrangentes que os fluxogramas comuns. Por esse motivo, eles são por vezes preferidos como ferramentas para a criação de especificações de processos. Entretanto eles exigem, normalmente, uma quantidade não trivial de gráficos, o que é desvantajoso.

A figura 3.7 apresenta um exemplo de uma especificação de processos representada em um diagrama de Nassi-Shneiderman.

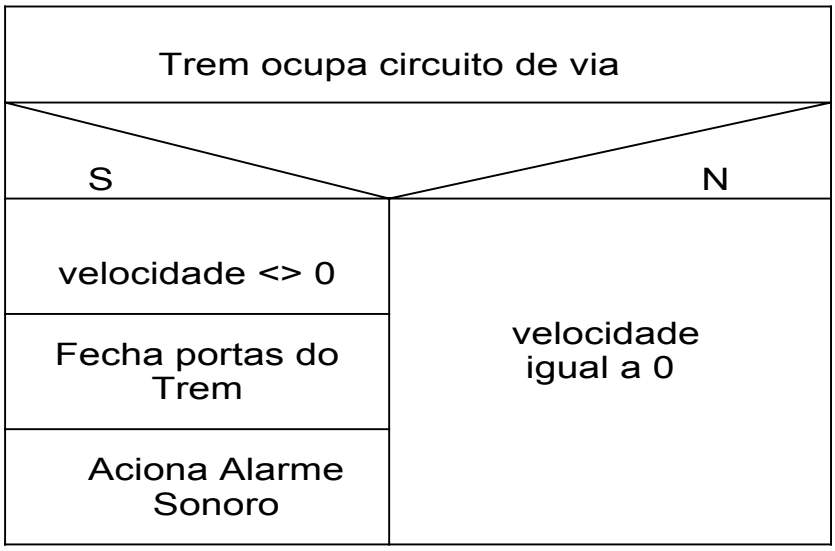


Figura 3.7 - Exemplo de uma Especificação de Processos utilizando um Diagrama de Nassi-Shneiderman

3.1.4. Diagrama Entidades-Relacionamentos

É função dos sistemas armazenar e utilizar informações sobre o ambiente com o qual interagem. Algumas vezes as informações são mínimas, porém na maioria dos sistemas atuais são bastante complexas. É de interesse saber não somente detalhes da informação contida em cada depósito, mas também que relacionamentos existem entre os Depósitos de Dados. Este aspecto do sistema é realçado pelo Diagramas de Entidades-Relacionamentos E-R.

Os componentes de um diagrama E-R são:

- **Tipos de Objetos**: representados por um retângulo, indicando uma coleção ou conjunto do mundo real cujos membros desempenham um papel no sistema que está sendo desenvolvido.
- **Relacionamentos**: representados por losangos, indicando um conjunto de associações entre os Tipos de Objetos que são interligados por linhas aos Relacionamentos.

A figura 3.8 apresenta um exemplo de um Diagrama Entidades-Relacionamentos.

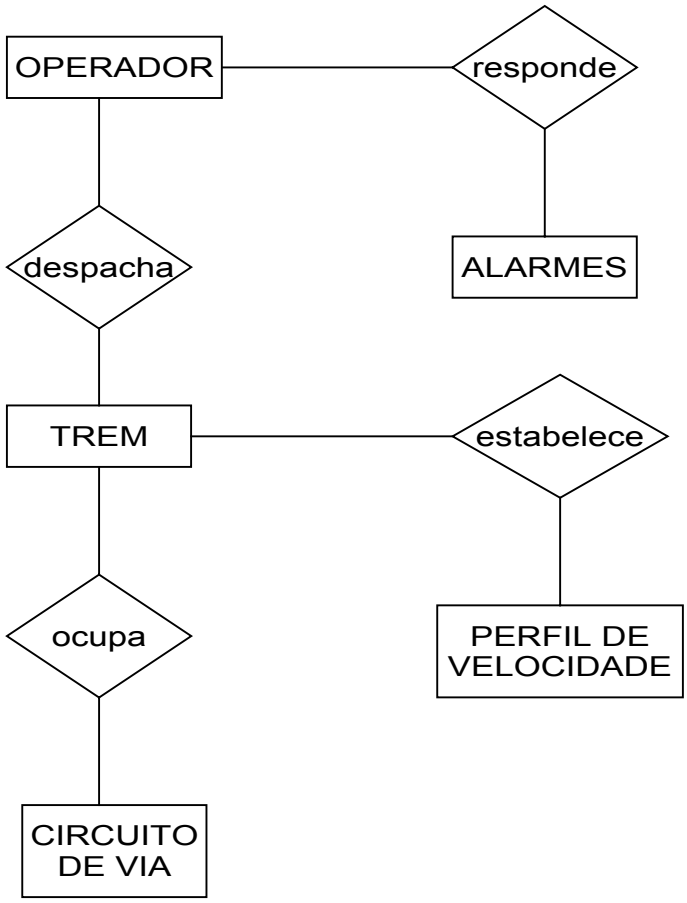


Figura 3.8 - Exemplo de um Diagrama Entidades-Relacionamentos

3.1.5. Diagrama de Transições de Estado

Outro aspecto a ser considerado em diversos sistemas complexos é o comportamento dependente do tempo, da seqüência na qual se tem acesso aos dados e da execução das funções. Em alguns sistemas comerciais de processamento, isso não é um aspecto importante a ser destacado, uma vez que a seqüência é, em princípio, trivial. Entretanto, muito sistemas on-line e de tempo real têm complexos relacionamentos de tempo, que devem ser modelados tão cuidadosamente quanto as funções e relacionamentos de dados.

A ferramenta de modelagem utilizada para descrever esse aspecto do comportamento de um sistema é o Diagrama de Transições de Estado - DTE. Nesse diagrama os retângulos representam os estados em que o sistema pode estar. Cada estado, portanto, representa um período de tempo durante o qual o sistema exibe algum comportamento observável. As linhas que interligam os retângulos mostram a mudança de estado ou transições de um estado para outro. Associada a cada mudança de estado há uma ou mais condições (eventos ou circunstâncias que causaram a mudança de estado) e zero ou mais ações (resposta, saída ou atividade que ocorre como parte da mudança de estado). A figura 3.9 apresenta um exemplo de um DTE.

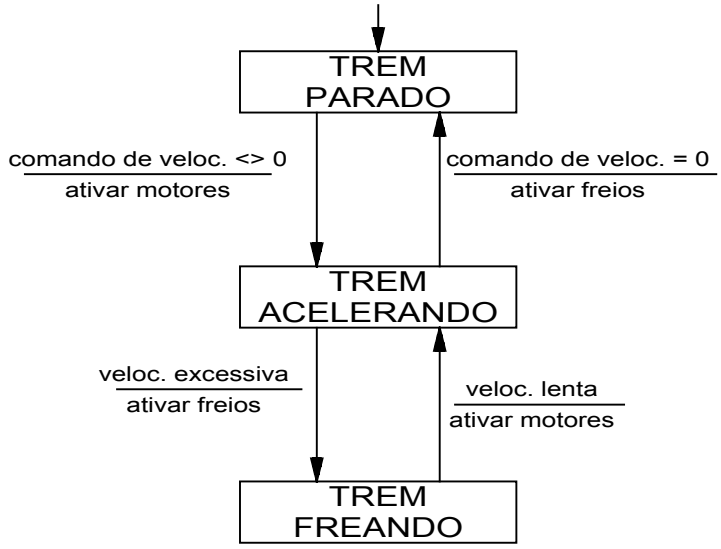


Figura 3.9 - Exemplo de um Diagrama de Transições de Estado

3.1.6. Diagrama Estrutural

Uma ferramenta de modelagem gráfica utilizada para representar a hierarquia de módulos de software é o Diagrama Estrutural. Nesse diagrama cada retângulo representa um módulo. As linhas que interligam os retângulos representam chamadas de módulos. O diagrama também apresenta os parâmetros de entrada passados para cada módulo chamado e os parâmetros de saída retornados pelo módulo quando ele termina sua tarefa e restitui o controle ao módulo chamador. A figura 3.10 apresenta um exemplo de um Diagrama Estrutural.

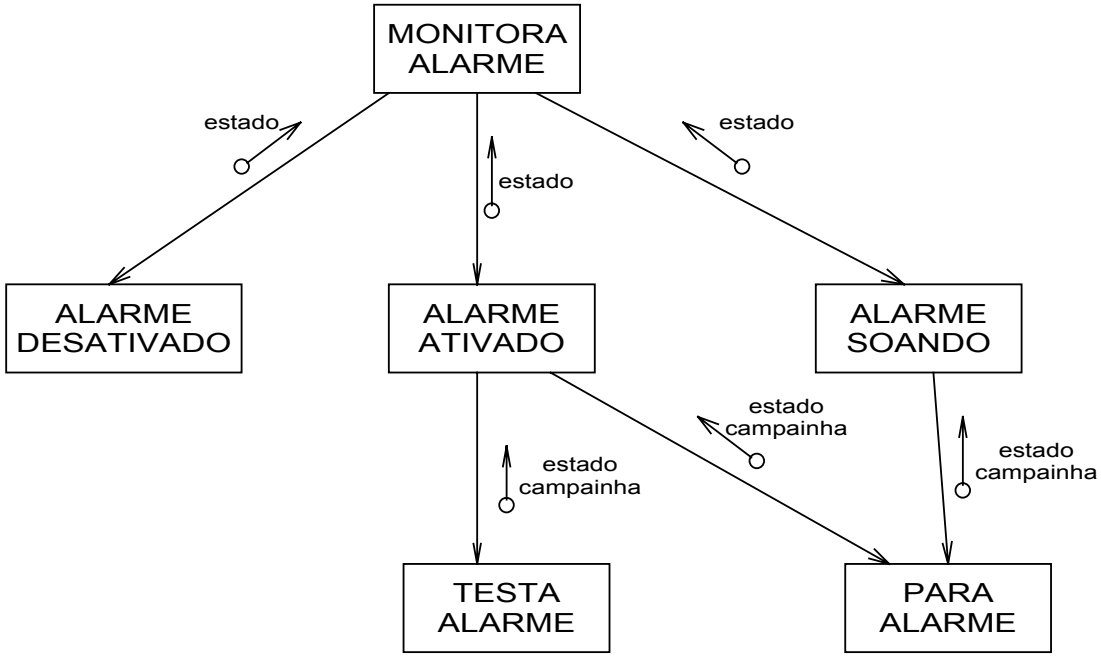


Figura 3.10 - Exemplo de um Diagrama Estrutural

3.1.7. Statecharts

A literatura da engenharia de software e de sistemas é quase unânime em reconhecer a existência de um problema maior na especificação e projeto de sistemas reativos complexos. Um sistema reativo, em contraste com um sistema transformacional, caracteriza-se por ter de reagir continuamente a estímulos internos e externos. Exemplos incluem telefones, redes de comunicação, sistemas instalados a bordo de aeronaves e a interface homem-máquina de diversos softwares (HAREL, 1987a).

A maior dificuldade reside em como se descrever o comportamento reativo de modo claro e realístico, e ao mesmo tempo, formal e rigoroso o suficiente que possa ser simulado por um computador.

Entre as soluções propostas para uma correta representação da característica reativa dos sistemas estão as Redes de Petri (AGERWALA, 1979) e a lógica temporal. Uma solução mais recente é representada pelos Statecharts, que constituem uma versão mais atual e abrangente do formalismo clássico das máquinas de estado finito e dos diagramas de transição de estado.

Os Statecharts são capazes de representar o comportamento dinâmico ou os aspectos de controle dos sistemas que descrevem. No Statechart os estados podem ser encadeados e ligados de variadas formas, e conseguem, assim, representar os sistemas em seus comportamentos seqüencial ou concorrente.

A figura 3.11 apresenta um exemplo de um Statechart para um sistema instalado a bordo de aviões. No Statechart os retângulos com bordas arredondadas, também chamados bolhas, representam os estados, enquanto que as ligações que interligam as bolhas representam as transições que ocorrem na mudança de um estado para outro. Cada bolha tem um nome que a identifica. A cada transição é associado um evento cuja ocorrência, ocasiona a mudança de estado. Condições e disparos de eventos também podem ser associadas a uma transição. O significada de uma condição é que a transição só ocorre se a condição especificada for obedecida. No caso de disparo, representa o acionamento de um outro evento pertencente ao Statechart.

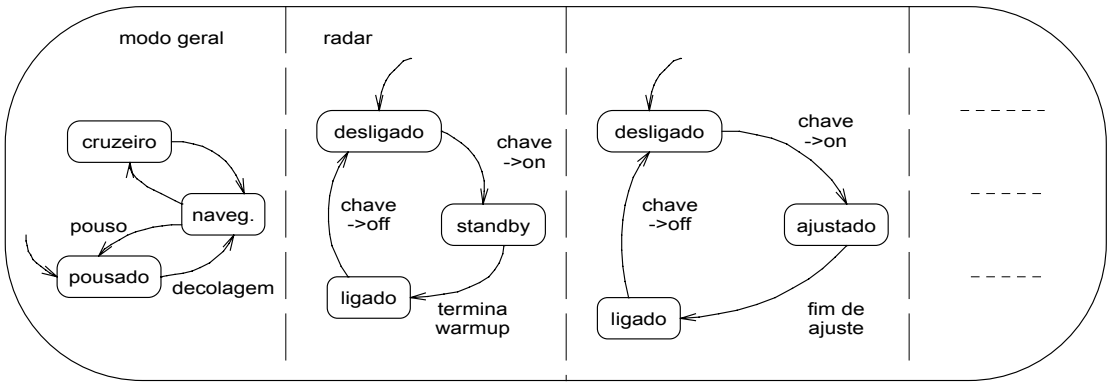


Figura 3.11 - Exemplo de um Statechart

3.1.8. Comparação e Avaliação

Qualquer comparação entre as técnicas apresentadas deve ser feita sob a premissa de que qualquer uma delas, se usada corretamente, pode proporcionar um auxílio inestimável ao projeto. Inversamente, mesmo a melhor das técnicas, se aplicada de maneira incorreta, adiciona muito pouco ao desenvolvimento do projeto. Tendo isto em vista, podem ser analisados alguns critérios de comparação (PRESSMAN, 1992).

- **Modularidade**: a notação deve suportar o desenvolvimento modular do software e prover meios para se especificar as interfaces.
- **Simplicidade**: a notação deve ser relativamente simples de se aprender, e fácil de ser utilizada e lida.
- **Facilidade de Edição**: o projeto pode requerer modificações durante seu desenvolvimento e manutenções, e por isso mesmo a representação deve ser facilmente editável para auxiliar nas alterações.
- **Leitura por meio de Máquina**: uma notação que possa ser lida diretamente por um computador é desejável, pois as ferramentas automatizadas de projeto vêm sendo cada vez mais adotadas.
- **Manutenção**: a manutenção do software é a fase mais cara de seu ciclo de vida, e uma manutenção na configuração de um software acarreta a manutenção de sua representação de projeto. Daí a necessidade de que a notação empregada possibilite que se executem manutenções de maneira simples e completa.
- **Direcionamento para Estrutura**: uma notação que force ou direcione para a utilização de técnicas estruturadas proporciona melhores práticas de projeto.
- **Processamento Automático**: Um projeto detalhado contém informações que podem ser processadas para proporcionar ao projetista novas e melhores visões a respeito da qualidade de um projeto. Tais visões podem ser melhoradas por meio de relatórios feitos automaticamente.
- **Representação de Dados**: a habilidade para representar dados locais e globais é um elemento essencial do projeto detalhado. Idealmente, uma notação adequada deveria representar tais dados de forma direta.
- **Verificação Lógica**: a verificação automática da lógica de projeto é um objetivo permanente durante a fase de teste do software. Uma notação que aumente a capacidade de verificar a lógica aumenta em muito a adequação dos testes.

- **Facilidade de Codificação**: uma notação que possa ser convertida facilmente para um código-fonte reduz em muito a introdução de erros nesta fase.

Basicamente cada um dos modelos acima apresentados focaliza um diferente aspecto de um sistema: o DFD mostra as funções, o ER realça os relacionamentos entre os dados e o DTE enfatiza o comportamento dependente do tempo do sistema. É útil estudar cada um desses aspectos isoladamente, pois a complexidade de um sistema típico é grande. Por outro lado, essas três visões do sistema devem permanecer consistentes e compatíveis entre si.

A maioria dos analistas de sistemas utiliza apenas uma ferramenta para elaborar todas as especificações, o que nem sempre permite representar adequadamente todos os aspectos do sistema. Deve-se então empregar uma combinação das ferramentas de representação, dependendo da preferência do usuário, da preferência do analista e principalmente da natureza dos diversos projetos a serem representados.

Qualquer técnica de documentação que permita descrever corretamente as exigências do usuário e que comunique efetivamente essas exigências é aceitável.

Há razões para a não adequação das máquinas de estado finito e dos diagramas de transição de estados para a representação de sistemas, que são:

- Diagramas de estado são planos e não oferecem um meio natural para se representar as noções de profundidade, hierarquia ou modularidade.
- Diagramas de estado não são "econômicos" para representar transições. Um evento que cause a mesma transição em um grande número de estados (por exemplo, uma interrupção) deve ser ligado a cada um desses estados separadamente, resultando em um número muito grande de linhas no diagrama.
- Os diagramas de estado são seqüenciais por natureza, não representando a concorrência.

Ao se utilizar uma representação gráfica para capturar a profundidade e a hierarquia de um sistema não há vantagem ou razão em se empregar árvores ou grafos lineares, pois estes não fazem uso da área do diagrama. Nessas representações não é utilizada a localização relativa dos elementos no diagrama (HAREL, 1987b).

Assim, visando à representação de eventos ortogonais, ou seja, simultâneos, bem como à escolha de uma representação gráfica cômoda e econômica em termos do número de ligações entre os estados, optou-se pelo desenvolvimento e aprofundamento na modelagem com o auxílio de Statecharts.

Nos próximos itens são descritos com maiores detalhes os elementos componentes de um Statechart e é feita sua extensão para comportar a característica adaptativa de forma similar à que foi introduzida nos autômatos adaptativos (JOSÉ NETO, 1993). Dessa forma, no final do capítulo é descrita a operação dos Statecharts adaptativos, ou seja, Statecharts que podem modificar-se como resposta aos seus eventos de entrada.

3.2. Descrição Detalhada de Statecharts

Neste item é feita, apresentada inicialmente, uma formulação rápida dos diversos tipos de autômatos, desde os finitos, passando pelos autômatos de pilha e chegando aos autômatos adaptativos. Em seguida é realizada uma descrição detalhada dos Statecharts convencionais, e por fim dos Statecharts adaptativos, utilizados no presente trabalho como fundamento teórico da ferramenta desenvolvida como parte prática desta pesquisa.

3.2.1. Autômatos Finitos

São dispositivos sem memória, com um conjunto finito fixo de estados, que reconhecem linguagens regulares com base em uma função de transição. Podem ser representados por:

a = (Q, ###, P, q_0, F) onde:

- **Q** é o conjunto finito não vazio de estados do autômato finito

- ### é o conjunto finito não vazio de símbolos de entrada, denominado alfabeto de entrada do autômato finito

- **q_0** é o estado inicial e corresponde a um elemento do conjunto de estados

  (**$q_0 \in Q$**)

- **F** é um subconjunto de **Q** e seus elementos são todos os estados finais do autômato **($F \subseteq Q$)**

- **P** é uma função de transição de estados, representada por um conjunto de produções do tipo: **(q, sα): → (q', s'α)** onde:

  - **q** é o estado corrente (**q** ∈ **Q**)

  - **s** é o símbolo consumido da cadeia de entrada (**s** ∈ ###)

  - ### é o restante da cadeia de entrada (irrelevante) (### ∈ ###*)

  - **q'** é o novo estado depois da execução (nova situação) (**q'** ∈ **Q**)

  - **s'** é o símbolo reposto na cadeia de entrada (opcional) (**s'** ∈ **{s, ε}**)

No caso do reconhecimento de uma cadeia **ω** ∈ ###*por **a**, partindo-se do estado q_0, vai-se consumindo **ω** símbolo a símbolo, pela aplicação das produções de **P**, até esgotar a cadeia **ω**. Se então o estado corrente pertencer ao conjunto **F**, diz-se que **ω** é uma sentença da linguagem, caso contrário não o será.

A linguagem definida por **a** é o conjunto de todas as cadeias **ω** que conduzirem **a**, do estado inicial q_0, a algum estado final de **F**.

3.2.2. Autômatos de Pilha

São dispositivos com memória organizada como pilha e constituídos de uma coleção de submáquinas que operam como autômatos finitos, mas que podem chamar uma à outra como se fossem "sub-rotinas", salvando uma informação de retorno em uma pilha nesta ocasião, e retornando ao estado salvo na pilha ao atingir uma situação final. São capazes de reconhecer linguagens livres de contexto. Podem ser representados por:

M = (Q, A, Σ, Γ, P, Z_0, q_0, F) onde:

- **Q** é o conjunto finito não-vazio de estados do autômato de pilha

- **A** é o conjunto de submáquinas $a_i = (Q_i, \Sigma_i, P_i, q_{0i}, F_i)$, **i = 1...n**, onde:

  - $Q_i \subseteq Q$ é o conjunto dos estados q_{ij} da submáquina a_i

  - $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da submáquina a_i

  - $P_i \subseteq P$ é o conjunto de produções da submáquina a_i

  - $q_{0i} \subseteq Q_i$ é o estado de entrada da submáquina a_i

- $\mathbf{F_i \subseteq F}$ é o conjunto dos estados de saída da submáquina $\mathbf{a_i}$

- Σ é o conjunto finito não-vazio de símbolos de entrada, denominado alfabeto de entrada do autômato de pilha **M**

- Γ é o conjunto finito não-vazio de símbolos de pilha, formando o alfabeto de pilha de **M**

- $\mathbf{Z_0}$ é o símbolo de pilha vazia ($\mathbf{Z_0} \in \Gamma$)

- $\mathbf{q_0}$ é o estado inicial e corresponde a um elemento do conjunto de estados ($\mathbf{q_0} \in \mathbf{Q}$)

- **F** é um subconjunto de **Q** e contém apenas os estados finais contidos em **Q** do autômato **M** ($\mathbf{F \subseteq Q}$)

- **P** é o conjunto das produções, que definem as regras de movimentação do autômato de pilha, e são do tipo: $\mathbf{(\gamma g, e, s\alpha): \rightarrow (\gamma g', e', s'\alpha)}$, onde:

  - γ é o restante da pilha (irrelevante) ($\gamma \in \Gamma^*$)

  - **g** é o topo da pilha (desempilhado pela produção) ($\gamma \in \Gamma$)

  - **e** é o estado corrente ($\mathbf{e} \in \mathbf{Q}$)

  - **s** é o símbolo consumido da cadeia de entrada ($\mathbf{s} \in \Sigma$)

  - α é o resto da cadeia de entrada (irrelevante) ($\alpha \in \Sigma^*$)

  - **g'** é o novo topo da pilha (empilhado pela produção) ($\mathbf{g'} \in \Gamma$)

  - **e'** é o novo estado ($\mathbf{e'} \in \mathbf{Q}$)

  - **s'** é o símbolo reposto na cadeia de entrada ($\mathbf{s'} \in \{s, \varepsilon\}$)

Consegue-se obter autômatos deste tipo que utilizam a pilha apenas quando forem detectadas construções aninháveis na cadeia de entrada, se for mantida coerência entre os estados e o conteúdo da pilha (JOSÉ NETO, 1993).

O autômato de pilha deve ser levado a uma condição inicial, em que a pilha deve estar vazia e o autômato em seu estado inicial, sempre que uma nova cadeia for submetida para ser reconhecida.

No caso do reconhecimento de uma cadeia ω por **M**, partindo-se do estado $q_{00} \in Q_0$, ω vai sendo consumido pela transições, que podem usar a pilha para promover chamadas/retornos de sub-máquinas, até que ω se esgote, deixando a pilha vazia em algum estado final. A linguagem definida por **M** é a coleção de todas as cadeias nestas condições.

3.2.3. Autômatos Adaptativos

Os autômatos adaptativos são dispositivos constituídos de um autômato (finito ou de pilha) inicial, que opera convencionalmente até a execução de alguma transição especial, acionada pela detecção de uma construção dependente de contexto. Nesta ocasião pode ocorrer uma alteração estrutural no autômato. Isto se repete até que a cadeia de entrada se esgote, em uma situação final da máquina de estados corrente. Os autômatos adaptativos são capazes de reconhecer linguagens sensíveis ao contexto.

Um autômato adaptativo **A** apresenta os seguintes elementos:

- ω, que é a cadeia de entrada a ser reconhecida ($\omega \in \Sigma^*$)

- E_0 , que é a máquina de estados que implementa **A** no início da operação

- E_m , que é a máquina de estados que implementa **A** no final da operação

- E_i , que é a máquina de estados que implementa **A** após **i** transições adaptativas

A aceitação de $\omega = \alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\ \ldots\ \alpha_m$ percorre uma trajetória $< E_0 , \alpha_0> \rightarrow < E_1 , \alpha_1> \ldots \rightarrow < E_m , \alpha_m>$, onde $< E_j , \alpha_j> \rightarrow < E_{j+1} , \alpha_{j+1}>$, ($0 \leq j \leq m$) representa o reconhecimento de $\#\#\#_j$ por E_j, ao final da qual é executada uma transição adaptativa P_j que altera E_j para E_{j+1}, onde P_j é representado por:

$P_j = (t_j, \mathcal{A}_j, u_j, \mathcal{B}_j)$ onde:

- t_j é a configuração anterior do autômato adaptativo **A**

- $\mathcal{A}_j$ é uma função adaptativa anterior, executada antes da mudança de configuração (opcional)

- u_j é a configuração posterior do autômato adaptativo **A**

- $\mathcal{B}_j$ é uma função adaptativa posterior, executada após a mudança de configuração (opcional)

As Funções Adaptativas podem ser representas pela seguinte nônupla:

$P_j = (\mathcal{F}, \mathcal{P}, \mathcal{V}, \mathcal{G}, \mathcal{C}, \mathcal{E}, \mathcal{I}, \mathcal{A}, \mathcal{B})$ onde:

- $\mathcal{F}$ é nome da função adaptativa

- $\mathcal{P}$ é a lista $(\rho_1, \rho_2, ...)$ dos parâmetros formais da função $\mathcal{F}$

- $\mathcal{V}$ é o conjunto de nomes de variáveis cujos valores são desconhecidos no instante da chamada da função, e que são preenchidos, uma única vez, a cada execução da função adaptativa

- $\mathcal{G}$ é o conjunto de nomes de seus geradores, ou seja, variáveis especiais que são automaticamente preenchidas a cada chamada da função adaptativa

- $\mathcal{C}$ é o conjunto de padrões de produções a serem consultadas para preenchimento das variáveis indefinidas referenciadas

- $\mathcal{E}$ é o conjunto de padrões de produções a serem consultadas para o preenchimento das variáveis indefinidas referenciadas, para depois serem eliminadas do conjunto de produções da máquina de estado corrente

- $\mathcal{I}$ é o conjunto de padrões de produções a serem inseridas no conjunto das produções da máquina de estados corrente

- $\mathcal{A}$ é uma ação adaptativa a ser efetuada antes da execução de $\mathcal{F}$

- $\mathcal{B}$ é uma ação adaptativa a ser efetuada após a execução de $\mathcal{F}$

Para se efetuar a chamada de uma Função Adaptativa deve-se indicar o nome da função e a lista de argumentos. Uma Ação Adaptativa representa uma chamada de uma Função Adaptativa, e é descrita por um par ordenado $(\mathcal{F}, \Lambda)$, em que $\mathcal{F}$ é o nome da Função a ser chamada e Λ é a seqüência de argumentos $(\eta_1, \eta_2, ...)$ passados para $\mathcal{F}$.

Um mecanismo de passagem de parâmetros atribui valores aos parâmetros, à entrada da função, sendo que esses parâmetros não podem ser alterados em outra situação.

No mecanismo de alteração de valores, os parâmetros utilizam o mecanismo de passagem de parâmetros, as variáveis são preenchidas uma só vez por ações de inspeção ou de eliminação de produções, e os geradores são preenchidos automaticamente com valores únicos, a cada entrada na função. Todos estes elementos se tornam inalteráveis após o seu preenchimento, o que pode resultar em um valor "desconhecido" para aqueles que não forem preenchidos.

Define-se uma configuração $\mathbf{t_k}$ do autômato adaptativo no instante **k** como: $\mathbf{t_k = (E_k, \gamma_k, q_k, \omega_k)}$ onde:

- $\mathbf{E_k}$ é a máquina de estados que implementa $\mathcal{F}$ no instante **k**

- γ_k é o conteúdo da pilha no instante **k**

- $\mathbf{q_k}$ é o estado corrente de $\mathbf{E_k}$ no instante **k**

- ω_k é a cadeia de entrada a ser processada por $\mathbf{E_k}$ no instante **k**

Sejam as seguintes configurações inicial e final, respectivamente: $\mathbf{t_0 = (E_0, Z_0, q_0, \omega)}$ e $\mathbf{t_f = (E_f, Z_0, q_f, \varepsilon)}$. O autômato evolui de uma configuração para outra a cada transição, resultante da aplicação de uma produção do tipo:

$\mathbf{(\gamma g, e, s\alpha): \mathcal{A}, \rightarrow (\gamma g', e', s'\alpha), \mathcal{B}}$ onde:

- $\mathbf{(\gamma g, e, s\alpha)}$ é a situação do autômato adaptativo anterior à aplicação da produção

- $\mathcal{A}$ é a ação adaptativa anterior (opcional)

- $\mathbf{(\gamma g', e', s'\alpha)}$ é a situação do autômato adaptativo posterior à aplicação da produção

- $\mathcal{B}$ é a ação adaptativa posterior (opcional)

Há vários tipos possíveis de transições:

Transições adaptativas - contêm ações $\mathcal{A}$ e/ou $\mathcal{B}$.

Transições entre sub-máquinas (chamadas e retornos) - exibem **g** e/ou **g'** explícito, com $\mathcal{A}$ e $\mathcal{B}$ omitidos.

Transições internas - **e** e **e'** pertencem à mesma sub-máquina, com **g** e **g'** omitidos, bem como $\mathcal{A}$ e $\mathcal{B}$.

Transições em vazio - **s** e **s'** omitidos (incondicional) ou .**s = s'** explícitos (condicional)

Há mudança de configuração a cada aplicação de uma transição adaptativa.

A linguagem reconhecida pelo autômato adaptativo é o conjunto de todas as cadeias ### que podem ser reconhecidas de acordo com o critério descrito.

Para se operar o autômato adaptativo, executa-se o seguinte procedimento:

No início, o autômato deve ser colocado em uma situação inicial (Z_0, e_0, α_0), ou seja, a pilha está vazia, e o autômato, em seu estado inicial e_0, recebe a cadeia inicial α_0 para análise.

Estando, depois da aplicação de um certo número de produções, na situação $(\gamma\eta, e, \sigma\alpha)$, o autômato pesquisa seu conjunto de produções em busca de uma que seja aplicável ao caso, ou seja, uma produção da forma desejada:

existindo uma única produção aplicável, a transição é deterministicamente executada;

havendo mais de uma produção aplicável, todas elas são executadas em paralelo (situação não-determinística);

transições com consumo de átomos têm precedência sobre transições em vazio;

transições internas são prioritárias;

não havendo produção aplicável em um estado não final, a cadeia α_0 é rejeitada;

não havendo produção aplicável em um estado final, a cadeia $\#\#\#_0$ é considerada aceita se tiver sido totalmente consumida, desde que a pilha esteja vazia, caso contrário, α_0 é também rejeitada;

o processo se repete até que uma situação final seja atingida (Z_0, e_f, ε) ou que α_0 seja rejeitada

A linguagem definida pelo autômato adaptativo é o conjunto de todas as cadeias α_0 que levem o autômato a uma situação final de aceitação.

A execução das transições no autômato adaptativo se faz da seguinte forma:

- Para uma produção aplicável à situação corrente:
  $(\gamma\eta, e, \sigma\alpha)$: $\mathcal{A}\#\#\#(\gamma\eta', e', \sigma'\alpha)$, $\mathcal{B}$

- se $\mathcal{A}$ estiver presente, é executada em primeiro lugar; se $\mathcal{A}$ eliminar a produção corrente, a execução desta produção é descontinuada, voltando-se a buscar nova produção.
- se η estiver especificada, deverá ser eliminada do topo da pilha.
- se η' estiver presente, deverá ser empilhada.
- se σ estiver especificada, deverá ser consumida da cadeia de entrada.
- se σ' estiver presente, deverá ser incluída na cadeia de entrada, em substituição a . ###
- o estado **e** é abandonado, e o novo estado será **e'**
- se $\mathcal{B}$ estiver declarada, será executada por último.

As ações adaptativas $\mathcal{A}$ e $\mathcal{B}$ são da forma $\mathbf{F}(\varphi_1, \varphi_2, \ldots \varphi_p)$ com φ_i argumentos a serem passados como parâmetros de **F**. Sua execução compreende, na seguinte ordem:

- passagem dos parâmetros e preenchimento dos geradores
- execução da ação adaptativa anterior da função
- execução das ações básicas de inspeção
- execução das ações básicas de eliminação
- execução das ações básicas de inclusão
- execução da ação adaptativa posterior da função

As ações adaptativas elementares assumem o seguinte formato:
prefixo [*padrão da produção*], onde o *prefixo* representa um dos três tipos de ações adaptativas elementares a serem executadas: "**?**" (ação de inspeção), "**-**" (ação de eliminação) e "+" (ação de inserção), enquanto que o *padrão da produção* corresponde a uma produção parametrizada à qual devem ser aplicadas as ações.

As ações básicas de inspeção são denotadas da seguinte forma:
? [$(\gamma\eta, e, s\alpha)$: $\mathcal{A} \rightarrow (\gamma\eta', e', s'\alpha)$, $\mathcal{B}$ **]**. O conjunto de produções do autômato é pesquisado quanto à ocorrência de produções da forma indicada. As variáveis utilizadas para denotar η**, e, s,** η'**, e', s'**, bem como os nomes e os argumentos das funções representadas por $\mathcal{A}$ e $\mathcal{B}$, são preenchidas com valores correspondentes da produção encontrada como resultado da inspeção. Se nenhuma produção desta forma for localizada, as variáveis indicadas serão marcadas como indefinidas. Havendo

mais de uma produção nestas condições, tem-se um não-determinismo, e os vários casos são tratados em paralelo. Estas ações básica não alteram o conjunto de produções.

As ações básicas de eliminação são denotadas da seguinte forma:
- [(γη, e, sα): $\mathcal{A} \rightarrow$ **(γη', e', s'α)**, $\mathcal{B}$ **]**. Procede-se como no caso das ações de inspeção, e, adicionalmente, elimina-se do conjunto de produções a produção encontrada. Se nenhuma produção desta forma for encontrada, ou se alguma variável de seu conjunto de variáveis estiver indefinida, nada será efetuado.

As ações básicas de inclusão são da seguinte forma:
+ [(γη, e, sα): $\mathcal{A} \rightarrow$ **(γη', e', s'α)**, $\mathcal{B}$ **]**. Procede-se como no caso das ações de inspeção, e, adicionalmente, inclui-se no conjunto de produções a produção indicada. Se nenhuma produção desta forma for encontrada, ou se alguma variável de seu conjunto de variáveis estiver indefinida, nada será feito.

Os autômatos adaptativos foram propostos como uma ferramenta alternativa para a formalização de linguagens sensíveis ao contexto, pois possibilita uma obtenção relativamente simples e eficiente de reconhecedores para essa classe de linguagens. Tal eficiência é conseguida pela estrutura hierárquica imposta aos autômatos adaptativos, bem como pela forma de operação destes autômatos, que se modificam conforme os dados de entrada. São criadas e eliminadas transições à medida que a cadeia de entrada vai sendo consumida, passando o autômato por configurações intermediárias, até se chegar à configuração final, em que toda a cadeia tenha sido consumida.

Uma descrição mais detalhada da teoria de autômatos finitos e de pilha e de autômatos adaptativos pode ser encontrada em (JOSÉ NETO, 1987), (JOSÉ NETO, 1993) e em (JOSÉ NETO, 1994).

3.2.4. Statecharts Convencionais

Os Statecharts permitem representar sistemas de forma mais sucinta que com o auxílio das máquinas de estado finito, o que é obtido pela adição de hierarquia, concorrência e comunicação broadcast.

A notação dos Statecharts representa uma extensão dos diagramas de transição convencionais que suporta a definição de um contexto incremental graças à possibilidade de decomposição hierárquica, da presença de um fluxo de controle baseado no histórico, das definições explícitas do comportamento concorrente, bem como de recursos de comunicação entre componentes concorrentes (ELIAS, 1992).

Informalmente, pode-se dizer que um Statechart consiste de uma rede de bolhas aninhadas, desenhadas como retângulos com os cantos arredondados, descrevendo um contexto específico incrementalmente. Na proposta de Harel (HAREL, 1987b), uma bolha pode ser decomposta de duas formas: bolhas XOR e bolhas AND. Se uma determinada bolha estiver ativa (for o estado corrente) e ela foi decomposta em outras bolhas XOR, então apenas uma de suas bolhas descendentes será ativada. Se, por outro lado, ela foi decomposta em bolhas AND, todas suas bolhas descendentes serão ativadas quando seu antecessor for ativado. Desta maneira, as bolhas AND são um meio de descrever o comportamento paralelo de forma explícita.

Uma transição acontece na dependência de condições restritivas e na ocorrência de eventos específicos, sendo representada por meio de arcos direcionados. Os eventos e condições restritivas que disparam uma transição, bem como as ações específicas a serem realizadas quando de sua ocorrência são representados como atributos do arco correspondente. Ações associadas a transições de estado podem ser utilizadas para gerar novos eventos, os quais podem ter efeito sobre o outras transições dos componentes concorrentes. A comunicação entre os componentes concorrentes pode ser estabelecida pela geração de eventos internos, ou seja, gerados diretamente pelos processos representados pelas Bolhas, e portanto internamente ao Statechart.

Ações podem ser vinculadas a bolhas. Neste caso podem ser de três tipos: ações executadas sempre que a correspondente bolha estiver ativa, ações executadas sempre que a bolha correspondente não estiver ativa e aquelas cuja execução se inicia no momento de ativação e termina por ocasião de desativação da bolha em questão.

Além das ações, um atributo de história pode ser associado às bolhas. Atributos de história levam ao processo de ativação de bolhas. Sempre que uma bolha com o atributo de história for ativada, verifica-se em seu Statechart de detalhamento quais foram as bolhas que estavam ativas na última vez em que se realizou a simulação de tal detalhamento, com a finalidade de que se ativem essas bolhas componentes de seu detalhamento. Se não houver uma história nos descendentes ou se for a primeira vez em que a bolha com o atributo de história foi ativada, então seus descendentes "default", se existirem, é que serão ativados.

Se um Statechart deve ser simulado, então a ativação do processo é iniciada pelas bolhas de nível mais alto. Uma vez concluído este processo, o Statechart é capaz de reagir a eventos. O conjunto de bolhas ativas de um Statechart em uma situação estável é referido como a configuração ou como o estado global do Statechart. Sempre que uma transição for disparada, a configuração do Statechart é afetada como conseqüência de transições realizadas.

Um Statechart é simulado em passos de tempo. Todas as transições presentes em um passo são executadas simultaneamente. Para evitar indefinições que podem ocorrer em alguns casos, pelo fato de serem gerados eventos que gerem ações que possam alterar alguma transição já efetuada, dividem-se os passos em micro-passos. Desse modo, se uma determinada transição disparar outra transição, estas serão realizadas em micro-passos subseqüentes dentro do mesmo passo. Assim, uma transição realizada em um determinado micro-passo, não afeta transições realizadas em micro-passos anteriores.

Nos próximos itens é apresentada uma formulação dos Statecharts convencionais, sua denotação gráfica, bem como sua aplicação prática, através de alguns exemplos. A descrição dos Statecharts aqui apresentada baseou-se em (HAREL, 1987a), (HAREL, 1987b), (HAREL, 1988a), (HOOMAN, 1992) e (HUIZING, 1991).

3.2.4.1. Formulação de Statecharts Convencionais

A formalização da sintaxe e da semântica dos Statecharts Convencionais, definida por Harel (HAREL, 1987a), é aqui representada, tendo como objetivos não só a exposição de tal formulação, mas também possibilitar sua extensão para os Statecharts Adaptativos, incorporando à teoria dos Statecharts Convencionais conceitos extraídos da formulação dos Autômatos Adaptativos.

Um Statechart convencional é composto por bolhas. O conjunto de bolhas **B** é definido conjuntamente com a Função de Hierarquia (ρ), Função de Tipo (ψ), conjunto de símbolos de História (**H**) e Função "Default" (δ).

A **Função Hierarquia**: ρ**: B** $\rightarrow$ $2^{\mathbf{B}}$, define para cada bolha, suas sub-bolhas. Se ρ**(x) =** ρ**(y)** $\Rightarrow$ **x = y**. Existe uma bolha única **r** $\in$ **B** tal que $\forall$**b** $\in$ **B**, **r** $\notin$ ρ**(b)**, **r** é a raiz do Statechart. [1]

A **Função Tipo**: Ψ**:S** $\rightarrow$ **{ AND, OR }**, define para cada bolha, o seu tipo. Se ρ**(b)** $\neq$ $\varnothing$ e Ψ**(b) = OR**, então ρ**(b)** é uma composição XOR de **b**, o que significa que quando o sistema estiver na bolha **b**, estará em uma e apenas uma de suas sub-bolhas. Se ρ**(b)** $\neq$ $\varnothing$ e Ψ**(b) = AND**, então ρ**(b)** é uma composição AND de **b**, o que significa que quando o sistema estiver na bolha **b**, estará simultaneamente em todas as suas sub-bolhas.

O conjunto de **Símbolos de História** relaciona-se com o conjunto de bolhas pela função γ**: H** $\rightarrow$ **B**$_{,}$, tal que γ **(**h_1**)** = γ **(**h_2**)** $\Rightarrow$ h_1 **=** h_2 e γ**(H)** é um subconjunto do conjunto de bolhas OR, ou seja, apenas uma bolha OR pode ter um símbolo de história associada a ela. Os símbolos de história definem a memorização de uma determinada configuração de bolhas ativas do sistema, possibilitando sua restauração.

A **Função "Default":** δ**:B** $\rightarrow$ $2^{\mathbf{B} \cup \mathbf{H}}$, define para uma bolha **b** um conjunto de bolhas e símbolos de história que são contidos na bolha. δ**(b)** é o conjunto "default" de **b**.

---

[1] Para S = {S_0, S_1, …, S_n} tem-se por definição: 2^S = {$\varnothing$, { S_0}, { S_1}, . . . , { S_n}, {S_0, S_1}, . . . {S_{n-1}, S_n}, {S_0, S_1, S_2}, . . ., {S_{n-2}, S_{n-1}, S_n},. . ., {S_0, S_1, . . ., S_n}

Define-se uma **ligação** de uma bolha **B** para uma bolha e **B'** como uma conexão entre tais bolhas, simbolizada por um arco orientado, com uma seta apontando para **B'**.

A uma ligação associa-se um **rótulo**. O conjunto de rótulos **L** é o conjunto de pares **ExA**, para **l** $\in$ **L**, **e** $\in$ **E**, **a** $\in$ **A**, **c** $\in$ **C**: **l = (e[c], a) = e[c] / a**, significando que a transição é disparada pelo evento **e**, gerando a ação **a**, desde que a condição **c** seja respeitada. **E** é o conjunto de eventos, **A** é o conjunto de ações e **C** é o conjunto de condições, conforme abaixo definidos. (Obs.:a definição de estado é feita após a definição do conjunto de ações)

- **Condições**: considerando-se C_p o conjunto de condições primitivas (true e false), define-se indutivamente o conjunto de condições **C** como descrito a seguir:
  - **T, F** $\in$ **C**, sendo **T** e **F** correspondentes às constantes lógicas verdadeiro e falso, respectivamente
  - Se **c** $\in$ C_p, então **c** $\in$ **C**
  - Se **b** $\in$ **B**, então **in(b)** $\in$ **C**; **in(b) = T** quando o sistema está em algum estado **s** tal que **b** $\in$ **S** e , e **in(b) = F** se o sistema não está em algum estado **s** tal que **b** $\in$ **S** (**in(b) = F**)
  - Se **c** $\in$ **C** $\Rightarrow$ valor atual da condição = **current (c)** $\in$ **C**
  - Se $c_1, c_2 \in C \Rightarrow c_1 \vee c_2, c_1 \wedge c_2, \neg c_1 \in C$

- **Eventos**: O conjunto de eventos primitivos é representado por $\mathbf{E_p}$. Por evento primitivo entende-se qualquer evento gerado externamente à lógica do Statechart, que possa alterar o conjunto de bolhas ativas. O conjunto dos eventos **E** é definido da seguinte forma:

  - Se λ é o evento nulo, então $\lambda \in \mathbf{E}$

  - Se $\mathbf{e} \in \mathbf{E_p} \Rightarrow \mathbf{e} \in \mathbf{E}$

  - Se $\mathbf{c} \in \mathbf{C} \Rightarrow \mathbf{true\ (c)}, \mathbf{false\ (c)} \in \mathbf{E}$

  - Se $\mathbf{b} \in \mathbf{B} \Rightarrow$ ocorrência de entrada na bolha **b**, quando o sistema estiver em algum estado **s**, tal que $\mathbf{b} \in \mathbf{S} \rightarrow \mathbf{exit(b)} \in \mathbf{E}$

  - Se $\mathbf{b} \in \mathbf{B} \Rightarrow$ ocorrência de saída da bolha **b**, quando o sistema estiver em algum estado **s**, tal que $\mathbf{b} \in \mathbf{S\ s} \rightarrow \mathbf{entered(s)} \in \mathbf{E}$

  - Se $\mathbf{e_1}, \mathbf{e_2} \in \mathbf{E}$, então $\mathbf{e_1} \vee \mathbf{e_2}, \mathbf{e_1} \wedge \mathbf{e_2} \in \mathbf{E}$

  - Se $\mathbf{e} \in \mathbf{E}, \mathbf{c} \in \mathbf{C}$, então $\mathbf{e[c]} \in \mathbf{E}$

- **Ações**: O conjunto de ações (eventos gerados pela lógica do Statechart) **A** é definido como:

  - Se μ é a ação nula $\Rightarrow \mu \in \mathbf{A}$

  - Se $\mathbf{a_i} \in \mathbf{A}, \mathbf{i = 0, ..., n}, \Rightarrow \mathbf{a_0}, \mathbf{...}, \mathbf{a_n} \in \mathbf{A}$

Seja o rótulo I_1 associado a uma ligação entre duas bolhas B_1 e B_1': $I_1 = E_1\ [C_1] / A_1$, conforme mostra a figura 3.12. Supondo-se que a bolha B_1 esteja ativa (**in(B_1) = T**), que ocorra o evento E_1 e a que a condição C_1 seja verdadeira (**current [C_1] = T**), ocorre uma **produção**: $(B_1, E_1, C_1, A_1)_t \rightarrow (B_1')_{t+\Delta t}$ definida pela mudança de bolha ativa de B_1 para B_1', com a conseqüente geração da ação A_1. Esta produção, assim definida corresponde a uma **transição elementar**.

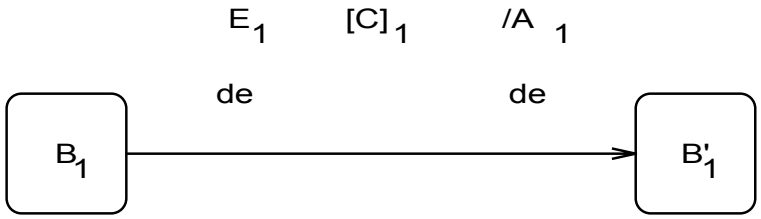


Figura 3.12 - Representação de uma Transição Elementar em um Statechart

Define-se um **estado** – **S** – do Statechart como sendo composto pelos elementos do conjunto 2^B, onde **B** é o conjunto de Bolhas ativas: **in(B_1) = T**, . . ., **in(B_n) = T**.

$(B_1, E, C_1, A_1)_t \rightarrow (B_1')_{t+\Delta t}$
. . . . . .
. . . . . .
$(B_n, E, C_n, A_n)_t \rightarrow (B_n')_{t+\Delta t}$

O conjunto $\{B_1, \ldots, B_n\} \subseteq B$ corresponde ao estado **S** e o conjunto $\{B'_1, \ldots, B'_n\} \subseteq B$ corresponde ao estado **S'**. Desta forma, um estado **S** é composto do conjunto das bolhas que estiverem ativas em determinado instante de tempo

No que se refere à semântica dos Statecharts, esta baseia-se em uma seqüência de instantes de tempo $\{\sigma_i\}_{i>0}$, Os intervalos básicos são definidos por $I_i = (\sigma_i, \sigma_{i+1})$. Em σ_{i+1} o sistema reage a um estímulo externo ocorrido no intervalo I_i. Um estímulo externo associado com σ_{i+1} é uma tripla (Π, Θ, ξ), onde Π é um conjunto de eventos externos ocorrendo em I_i; Θ é um conjunto de condições cujo valor é verdadeiro em I_i e ξ é uma função determinada pelo ambiente externo, tal que para uma variável $\xi(\nu) = x$ se o valor de ν for x em I_i.

Uma configuração do sistema **SC** associada ao instante σ_{i+1} é uma quádrupla (X, Π, Θ, ξ), onde **X** é a configuração de estados, e (Π, Θ, ξ) é um estímulo externo associado a σ_{i+1}.

A reação do sistema em qualquer instante é composta pelo conjunto de transições realizadas naquele instante e pelo conjunto de eventos gerados quando tais transições são realizadas. Então a reação do sistema é um par (Γ, Π^e), onde Γ é um conjunto de transições chamado **passo** e Π^e é um conjunto de eventos gerados por Γ. Dada uma configuração de sistema $SC = (X, \Pi, \Theta, \xi)$, define-se a reação do sistema $\{SR = (\Gamma, \Pi^e)\}$ como um conjunto das possíveis reações a **SC**.

Seja **SC** uma configuração de sistema no instante **i** e $\Gamma = (\mu\Gamma_0, \ldots \mu\Gamma_m)$ um passo de **SC** visto a um instante de tempo σ_i.

Uma transição $\mu\Gamma$ é definida como o conjunto de triplas $\mu T \subset 2^S x L x 2^S \cup H$, sendo $\mu T = \{t_0, \ldots t_n\}$, onde $t_i = (SX_i, (e_i, a_i), Sy_i)$. Conforme já definido para uma transição elementar. $a = a_0, \ldots, a_n$ é a ação causada por $\mu\Gamma$, . $e = e_0, \ldots, e_n$ são os eventos primitivos e **SX** e **SY** são estados do Statechart.

Dada uma configuração **SC**, a reação do sistema a um evento externo é composta por uma seqüência de micro-passos. O primeiro micro-passo é definido como um conjunto de transições cujo disparo ocorre em **SC**. Este primeiro micro-passo implica em uma micro-configuração do sistema, a qual pode reagir a outro micro-passo, e assim por diante.

Uma micro-configuração de sistema **μSC**, com relação a uma configuração **SC = (X, Π, Θ, ξ)**, é uma quíntupla **μSC = (μX, μΠ, μΘ, μξ, μY)** onde:

- **μX** é uma configuração parcial de estados: $\mu X \subseteq X$
- $\mu\Pi \in E_p \cup \{exit(s), entered(s) \mid s \in S\}$, $\mu\Pi \subseteq \Pi$
- $\mu\Theta \subseteq \{current(c) \mid c \in C_p\}$
- **μξ** é uma função que atribui valores a variáveis correntes
- **μY** é uma configuração parcial de estados, isto é, **μξ** é uma função de $\{current(\nu) \mid \nu \in V_p\}$ para o conjunto de números.

Dada duas micro-configurações de sistema **μSC = (μX, μΠ, μΘ, μξ, μY)** e $\mu SC_1 = (\mu X_1, \mu\Pi_1, \mu\Theta_1, \mu\xi_1, \mu Y_1)$, ambas em relação a **SC = (X, Π, Θ, ξ)**, diz-se que **μSC** é um possível sucessor de μSC_1 se: $\mu X \subseteq \mu X_1$, $\mu\Pi_1 \subseteq \mu\Pi$ e $\mu Y_1 \subseteq \mu Y$.

Um passo faz com que o sistema atinja uma nova configuração, que é igual à última micro-configuração atigida pela seqüência de micro-passos componente de um passo. Desde que não haja mais transições habilitadas do último micro-passo, o sistema aguarda um intervalo de tempo para a ocorrêcia de novos estímulos externos. Os eventos, conjuntamente com novos valores atribuidos a condições e variáveis são a saída do sistema ao ambiente externo.

Uma execução do sistema é uma seqüência $\{(SC_i, \Gamma_i, \Pi_i^e)\}_{i\geq 0}$, onde:

- $SC_0 = (X_0, \Pi_0, \Theta_0, \xi_0)$ X_0 é a configuração de estados inicial e (Π_0, Θ_0, ξ_0) são estímulos externos ocorrendo no primeiro intervalo de tempo I_0
- Γ_i é um passo de SC_i no instante σ_{i+1}, $i \geq 0$
- SC_{i+1} é a configuração alcançada pela aplicação de uma transição Γ_i a partir da configuração SC_i, $i \geq 0$
- Π_i^e é o conjunto de eventos gerados, aplicando-se uma transição por Γ_i, $i \geq 0$

Seja **SC** uma configuração de sistema. O sistema é não determinístico em **SC** se existirem duas reações diferentes (Γ_1, Π_1^e), (Γ_2, Π_2^e) tal que $\Pi_1^e \neq \Pi_2^e$ ou $\Gamma_1 \neq \Gamma_2$

3.2.4.2. Exemplos de Statecharts Convencionais

Neste item são apresentados alguns exemplos no emprego da notação de Statecharts, com a finalidade de se obter uma maior familiarização e compreensão com a teoria apresentada.

a) Detalhamento de Bolhas

No exemplo da figura 3.13, onde o diagrama 3.13a executa a mesma função do diagrama 3.13b. Os símbolos **e, f, g** e **h** representam os eventos que disparam as transições e o termo entre parênteses representa uma condição. As bolhas representadas na figura 3.13a são **A**, **B** e **C**. Assim, **g(B)** dispara a transição de **A** para **C** se **g** ocorrer desde que bolha **B** esteja ativa. A bolha **D** é o XOR de das bolhas **A** e **C**, de forma que estar na bolha **D** significa estar em na bolha **A** ou na bolha **C**, mas não em ambas. O principal aspecto a ser destacado neste exemplo é o evento **f**, que deixa o contorno da bolha **D**, por definição se aplica tanto à bolha **A**, quanto à bolha **C.**

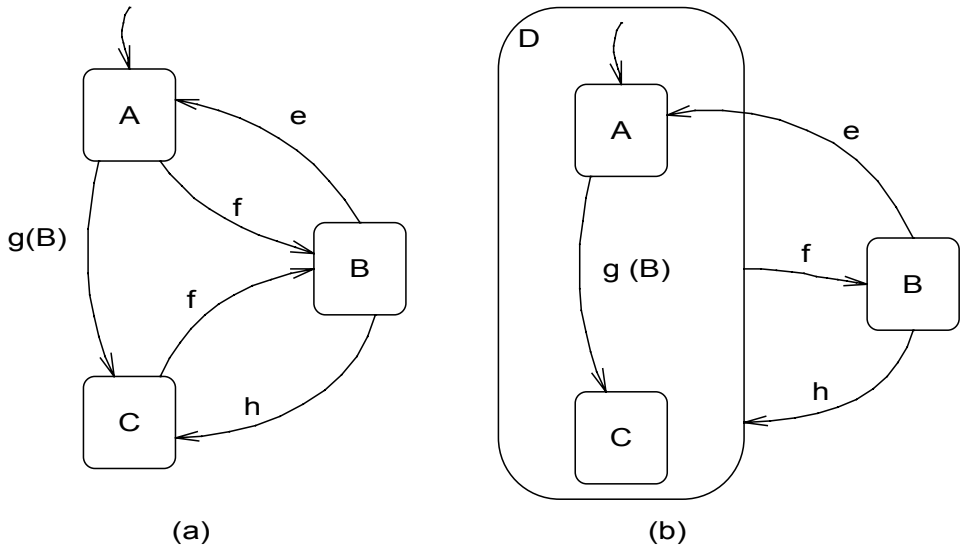


Figura 3.13 - Statechart representando o Detalhamento de Bolhas

Existe ainda a seta de "default", representada por um pequeno arco terminado por uma seta. Na figura 3.13a bolha **A** é a bolha "default" entre as bolhas **A, B** e **C**.

b) Ortogonalidade

No exemplo da figura 3.14, é ilustrado o conceito de ortogonalidade, que representa uma decomposição AND, que é o dual da decomposição XOR de bolhas. O diagrama da figura 3.14b executa a mesma função do diagrama da figura 3.14a. Na figura 3.14b a bolha **Y** é composta por dois componentes ortogonais: as bolhas **A** e **D**; para estar em **Y** deve-se estar em **A** e **D**, e por conseqüência, em **B** e **F**, que são as bolhas "default" de **A** e **D**, respectivamente.

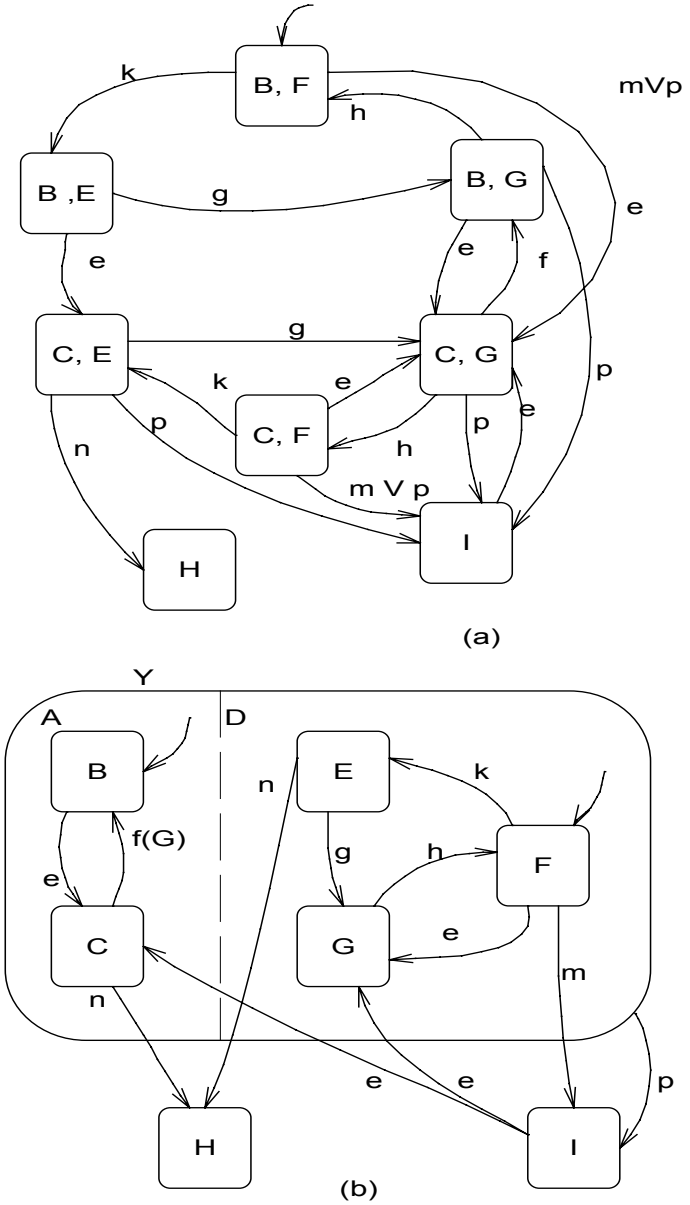


Figura 3.14 - Statechart Representando a Ortogonalidade

Note-se a simultaneidade de transições que acontece quando o evento **e** ocorre na configuração de bolhas **B** e **F** (a configuração se torna **C, G**), bem como a divisão e a reunião de transições que entram e saem do estado **Y** (transição **n** vinda das bolhas **C** e **E** para a bolha **H** e a transição **e** saindo da bolha **I** para as bolhas **C** e **G**). Note-se também a condição **(G)** acoplada ao evento **f** saindo do estado **C**, na figura 3.14b e como ela se reflete na figura 3.14a.

Este exemplo ilustra bem a conveniência da utilização da notação dos Statecharts, devido à grande redução obtida no número de arcos, bem como de seus cruzamentos, facilitando sobremaneira a visualização das informações.

c) Conceito de História

O exemplo de figura 3.15 refere-se à característica que os Statecharts apresentam em armazenar a informação da última bolha ativa. Nesta figura há um identificador history (círculo com a letra **H**) na bolha **A**, indicando que quando se entrar nesta bolha, deve ser ativada a última bolha visitada, ignorando-se a bolha "default". A bolha "default" é ativada quando ocorre a primeira entrada na bolha **A**.

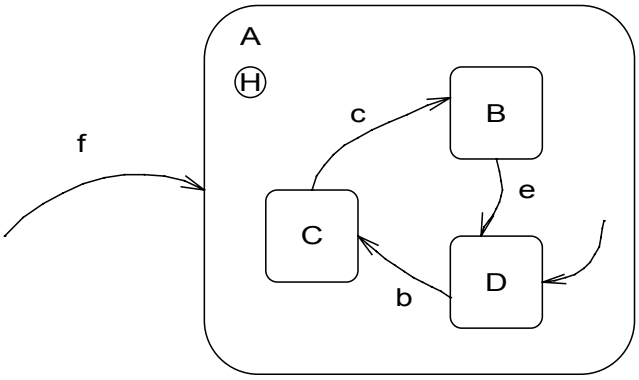


Figura 3.15 - Statechart Representando o Conceito de História

d) Uma Transição Disparando outras Transições

A figura 3.16 ilustra um exemplo da inclusão em um Statechart de transições que disparam outras transições. Se o estado corrente for **(B, F, I)** e ocorrer o evento externo **m**, a próxima configuração assumida será **(C, G, H)** em virtude da ativação do evento **c** pela bolha **H** e pelo conseqüente disparo de duas transições nas bolhas **A** e **D**. Se, neste instante, o evento **n** ocorrer, a nova configuração será **(B, E, I)** em virtude das transições gerarem eventos que disparam outras transições.

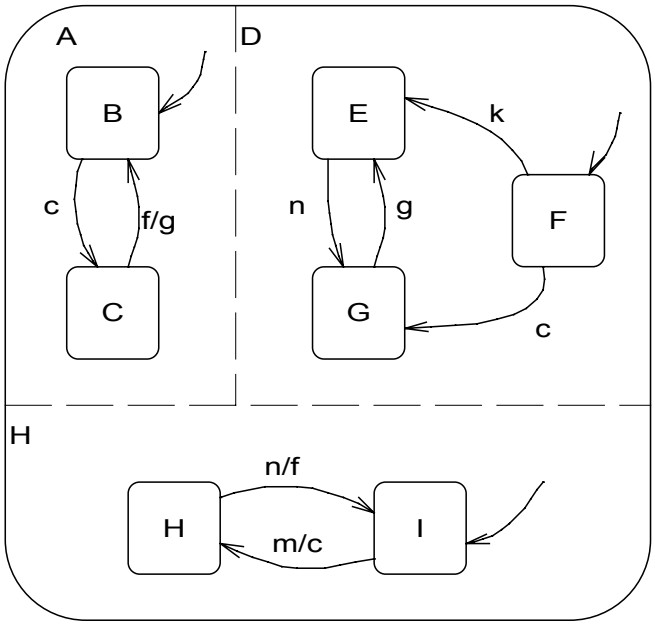


Figura 3.16 - Statechart representando Transições que disparam outras Transições

e) Execução em Micro-Passos

Na figura 3.17 é ilustrado um exemplo relativo à importância da divisão do conceito de passos em micro-passos. Nesta figura, se na configuração **(A1, B1)** ocorrer o evento externo **¬a** (**NOT a**), é disparada a transição de **A1** para **A2**, fazendo com que seja acionado o evento **b**, que por sua vez faz com que ocorra a transição de **B1** para **B2**. No entanto, o evento **b** aciona o evento **a**. Se essas duas transições fossem realizadas no mesmo intervalo de tempo, ficar-se-ia na dúvida se a transição de **A1** para **A2** seria desfeita. No entanto, como essa transições são realizadas em instantes diferentes, ou seja, em micro-passos distintos, a nova configuração dos sistema após a ocorrência do evento **¬a** será **(A2, B2)**.

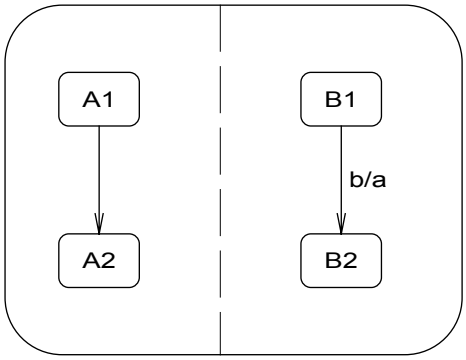


Figura 3.17 - Statechart Representando Execução em Micro-Passos

3.2.5. Statecharts Adaptativos

De forma similar aos autômatos adaptativos, pode-se definir os Statecharts adaptativos como sendo constituídos por um Statechart convencional inicial, que transita entre os seus estados de maneira convencional até a execução de alguma transição que contenha uma chamada a uma função adaptativa.

Formalmente um Statechart Adaptativo **SA** compõe-se de:

- ###, que é a seqüência dos eventos externos independentes a serem tratados

- $\mathbf{SA_0}$, que é o statechart convencional que representa **SA** no início da operação

- $\mathbf{SA_m}$, que é o statechart convencional que representa **SA** no final da operação

- $\mathbf{SA_i}$, que é o statechart convencional que representa **SA** após **i** transições adaptativas

As transições realizadas em decorrência da seqüência de eventos externos independentes $\omega = \alpha_0\ \alpha_1\ \alpha_2\ \alpha_3\ \ldots\ldots\ \alpha_m$ fazem com que se percorra uma trajetória

$\langle SA_0, \alpha_0 \rangle \rightarrow \langle SA_1, \alpha_1 \rangle \ \ldots \rightarrow \langle SA_m, \alpha_m \rangle$, onde $\langle SA_j, \alpha_j \rangle \rightarrow \langle SA_{j+1}, \alpha_{j+1} \rangle$,

($0 \leq j < m$) representa a transição realizada em decorrência do evento α_j por **SA**, ao final da qual é executada uma transição adaptativa P_j que altera SA_j para SA_{j+1}, conforme indicado na figura 3.18.

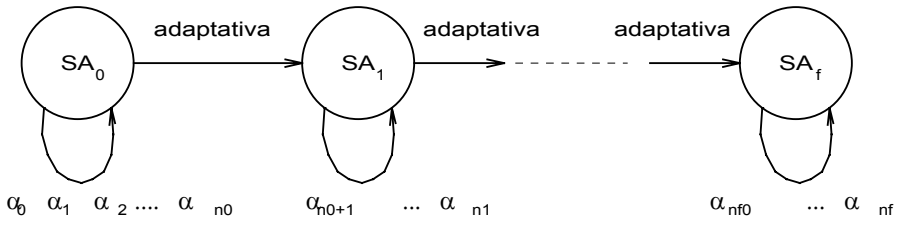


Figura 3.18- Representação de Transições em um Statechart Adaptativo

Uma transição em um Statechart Adaptativo pode ser representada por:

$$(X_i, \Pi_i, \Theta_i, \xi_i, \Gamma_i, \Pi_i^e, \mathcal{Y}_j) \rightarrow (X_{i+1}, \Pi_{i+1}, \Theta_{i+1}, \xi_{i+1}, \Gamma_{i+1}, \Pi_{i+1}^e, \mathcal{Z}_j),$$

sendo:

- $(X_j, \Pi_j, \Theta_j, \xi_j, \Gamma_i, \Pi_i^e)$ é a configuração do statechart antes da aplicação de um passo

- $(X_{j+1}, \Pi_{j+1}, \Theta_{j+1}, \xi_{j+1}, \Gamma_{i+1}, \Pi_{i+1}^e)$ é a configuração do statechart após a aplicação de um passo

- $\mathcal{Y}_j$, $\mathcal{Z}_j$ representam o conjunto de eventuais chamadas de funções adaptativas para cada passo

Os tipos possíveis de transições são:

Transições adaptativas – contêm ações $\mathcal{Y}$ e/ou $\mathcal{Z}$.

Transições não-adaptativas – com $\mathcal{Y}$ e $\mathcal{Z}$ omitidos.

As Funções Adaptativas podem ser representadas pela mesma nônupla utilizada nos Autômatos Adaptativos, desde que seus componentes $\mathcal{C}$, $\mathcal{E}$ e $\mathcal{I}$ sejam interpretados à luz das produções que definem transições do Statechart Adaptativo.

Para se efetuar a chamada de uma Função Adaptativa deve-se indicar o nome da função e a lista de argumentos. Uma Ação Adaptativa representa uma chamada de uma Função Adaptativa, e é descrita por um par ordenado $(\mathcal{F}, \Omega)$, em que $\mathcal{F}$ é o nome da Função a ser chamada e Ω é a seqüência de argumentos $(\sigma_1, \sigma_2, ...)$ passados para $\mathcal{F}$.

Um mecanismo de passagem de parâmetros atribui valores aos parâmetros, à entrada da função, sendo que esses parâmetros não podem ser alterados em outra situação.

No mecanismo de alteração de valores, os parâmetros utilizam o mecanismo de passagem de parâmetros, as variáveis são preenchidas uma só vez por ações de inspeção ou de eliminação de produções, e os geradores são preenchidos automaticamente com valores únicos, a cada entrada na função. Todos estes elementos se tornam inalteráveis após o seu preenchimento, o que pode resultar em um valor "desconhecido" para aqueles que não forem preenchidos.

Estando o Statechart Adaptativo em sua situação inicial, recebe a seqüência de eventos externos. As transições são executadas conforme essa seqüência, consumindo os eventos externos, repetindo-se o processo até o fim da seqüência. Para cada evento recebido verifica-se qual ou quais são as produções a serem executadas.

No que se refere à execução dos Statecharts Adaptativos, se $\mathcal{A}$ estiver presente, é executada em primeiro lugar, enquanto que se $\mathcal{B}$ estiver declarada, será executada por último.

As ações adaptativas elementares utilizadas para os Statecharts Adaptativos são as mesmas utilizadas nos Autômatos Adaptativos, ou seja, têm o seguinte formato: *prefixo* [*padrão da produção*], onde o *prefixo* representa um dos três tipos de ações adaptativas elementares a serem executadas: "**?**" (ação de inspeção), "**-**" (ação de eliminação) e "+" (ação de inserção), enquanto que o *padrão da produção* representa uma produção parametrizada à qual devem ser aplicadas as ações. A produção aqui citada é aquela referente aos Statecharts Adaptativos, ou seja:

$$(X_i, \Pi_i, \Theta_i, \xi_i, \Gamma_i, \Pi_i^e, \mathcal{Y}_j) \rightarrow (X_{i+1}, \Pi_{i+1}, \Theta_{i+1}, \xi_{i+1}, \Gamma_{i+1}, \Pi_{i+1}^e, \mathcal{Z}_j),$$

O modo de operação dessas ações é similar ao relatado na descrição dos Autômatos Adaptativos.

Como ilustração é apresentado um exemplo da execução de uma ação adaptativa, conforme ilustrado na figura 3.19.

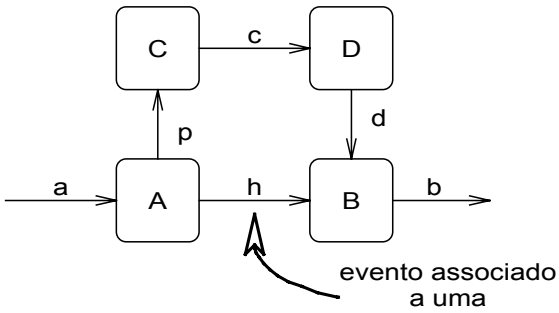
Na figura 3.19a é mostrado um trecho do Statechart original. Na ligação com o evento **h** está associada a uma chamada da função adaptativa **F**.

No caso do exemplo, a função adaptativa **F**, conforme mostra a figura 3.19b, é composta pela adição das bolhas **T** e **U** e pelas ligações **x**, **y** e **z**, além de realizar a eliminação da ligação **c** presente no Statechart original.

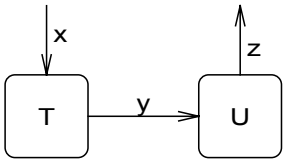
Quando o Statechart tiver como bolha corrente a bolha **A**, será, inicialmente realizada a transição para a bolha **B**, supondo-se que a ação adaptativa associada ao evento **h** é do tipo posterior. Após a ocorrência desta transição é que a ação adaptativa será realizada, fazendo com que a nova configuração do Statechart seja aquela apresentada na figura 3.19c.

Portanto, conforme se pode notar pela figura 3.19c, a ação adaptativa realizada teve como conseqüências a adição das bolhas **T1** e **U1**, além das ligações **x1**, **y1** e **z1**, que são instâncias dos elementos da função adaptativa. A posição em que os elementos criados pela ação adaptativa foram inseridos deve ser previamente determinada. Além da inclusão desses elementos, a ação adaptativa ocasionou a eliminação da ligação **c**.

No capítulo 4, item 4.4.3, é apresentado um exemplo completo da utilização da característica adaptativa incorporada aos Statecharts, conforme aqui descrito.



(a) Trecho do Statechart Original



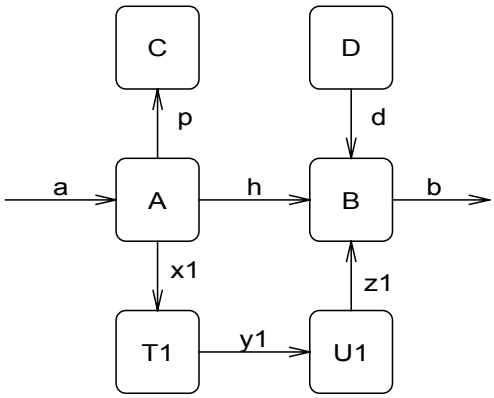


Figura 3.19 - Statechart com Ação Adaptativa

CAPÍTULO 4 - O GERADOR DE STATECHARTS ADAPTATIVOS – STAD

Neste capítulo é apresentada a ferramenta desenvolvida para a edição e simulação de Statecharts Adaptativos – o STAD. São descritas suas principais características, incluindo-se as funções nela implementadas, bem como o modo como cada elemento é utilizado pelo sistema desenvolvido.

Finalizando o capítulo, são apresentados alguns exemplos que ilustram sua capacidade na geração e simulação de Statecharts Adaptativos, além de demonstrar seu modo de utilização

4.1. A FERRAMENTA DESENVOLVIDA

Neste item são descritas as características principais implementadas na ferramenta desenvolvida, ou seja, o Gerador e Executor de Statecharts Adaptativos, bem como são tecidos comentários acerca de sua implementação.

4.1.1. Linguagem de Desenvolvimento

Para que fosse possível proceder o desenvolvimento do Gerador de Statecharts Adaptativos, necessitou-se, antes de mais nada, escolher o ambiente de desenvolvimento mais adequado para que se conseguisse atingir com eficiência o objetivo pretendido.

Dentre as ferramentas disponíveis, o software que se mostrou mais apropriado para este desenvolvimento foi o Microsoft Visual Basic, principalmente pelos recursos oferecidos para a criação de interfaces amigáveis com o usuário e pela simplicidade com que permite associar eventos aos objetos necessários para a geração de interface gráfica homem-máquina adequada à representação dos Statecharts Adaptativos.

O Visual Basic é tido como uma das melhores ferramentas de programação já criadas para PC (HOLZNER, 1994). A criação e utilização do Visual Basic é essencialmente visual, visto que, no paradigma a que se refere, antes de escrever qualquer linha de código, projeta-se a interface homem-máquina do produto.

No ambiente DOS, para criar telas elaboradas e agradáveis, o programador necessita escrever muitas linhas de código. No Visual Basic para Windows, ao contrário, todos os recursos de tela (menus, botões de comando, caixas de texto, etc) já estão disponíveis para uso, sem a necessidade de se programá-los

explicitamente, restando ao programador apenas a tarefa de instanciá-los adequadamente.

Em linhas gerais, a confecção de um programa neste ambiente ocorre em três etapas: o traçado das telas, a determinação de propriedades (características) para os objetos nelas inseridos, e, por fim, a vinculação desses objetos a comandos de programação. A tarefa de criação de uma janela com alguns objetos em seu interior é feita selecionando-se a arrastando-se os mesmos para a posição desejada, com a utilização do mouse.

Da mesma forma, o tamanho desses objetos também é alterável de forma bastante simples, ou seja, marcando-se o objeto e deformando-o pelo acionamento de pontos sensíveis que aparecem em suas extremidades. Também a tarefa de escrita do código é facilitada, pois um editor dirigido por sintaxe auxilia o programador a redigir seu programa, informando-o sobre cada erro detectado.

Todos os recursos operacionais deste ambiente são reforçados pela presença de um sistema de ajuda on-line muito claro e minuncioso. A maioria dos comandos e informações presentes nessa ajuda é complementada por exemplos elucidativos. (MACHADO, 1992)

Dessa forma, por todos os motivos expostos, optou-se pela utilização do Visual Basic para o desenvolvimento da ferramenta de edição e execução de Statecharts Adaptativos aqui apresentada.

4.1.2. Características Gerais do Editor e Simulador de Statecharts Adaptativos – STAD

O Editor e Simulador de Statecharts Adaptativos – STAD – é uma ferramenta de grande valia e de fácil utilização, para a edição e simulação do comportamento de sistemas, tanto comuns, quanto de tempo real.

Cada sistema, que tenha seus dados incluídos no modelo desenvolvido com o auxílio do STAD, é tratado como um projeto. Dentro de cada projeto podem ser criados tantos níveis de detalhamento, quantos forem necessários.

Todos os dados referentes aos projetos cadastrados na ferramenta são armazenados em um Banco de Dados, implementado através do Gerenciador de Banco de Dados Microsoft ACCESS. O Visual Basic permite manipular tabelas criadas por esse gerenciador, tendo a possibilidade de executar consultas, alterar os dados de um determinado registro, excluir ou acrescentar registros.

Dessa forma, ao se criar um projeto, vão sendo acrescentados os registros correspondentes às tabelas do Banco de Dados. Ao se abrir um projeto, é feita uma consulta às tabelas para se obter os dados necessários para a montagem dos Statecharts já cadastrados.

Para a montagem dos Statecharts, foi implementado um editor totalmente gráfico. Assim, as Bolhas e as Ligações entre elas podem ser criadas de forma bastante simples e auto-explicativa. Ao se inserir uma Bolha em um Statechart, é possível estabelecer que esta possa ser detalhada, o mesmo podendo ser feito com as Bolhas criadas nos detalhamentos, à medida da necessidade, em tantos níveis quantos forem necessários. A figura 4.1 apresenta um exemplo ilustrativo da edição de um Statechart com o auxílio da ferramenta..

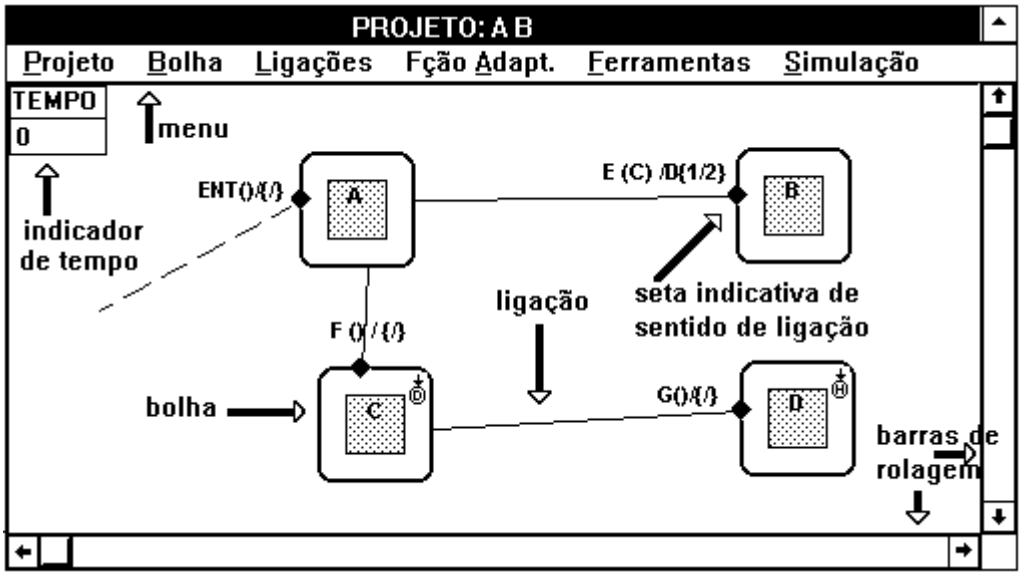


Figura 4.1. Exemplo de Edição de um Statechart

Cada Bolha é identificada por um nome e não pode haver mais de uma Bolha com o mesmo nome. São implementadas Bolhas dos tipos AND, OR e Primitivas. Os dois primeiros tipos são os mesmos especificados na sintaxe apresentada no capítulo 3. As Bolhas Primitivas são uma extensão criada para permitir o cálculo de expressões, possibilitando a execução até os níveis mais primitivos de detalhamento.

Na tela, o detalhamento das Bolhas em níveis, não é feito, fisicamente no interior da representação de cada Bolha, mas através da abertura de uma nova

janela. Foi adotada esta técnica, pois apenas uma janela do ambiente Windows não comportaria um detalhamento muito extenso.

De forma análoga aos Statecharts, as funções adaptativas também são editadas através de janelas do ambiente. Tanto a janela de edição de Statecharts, quanto a janela de edição das funções adaptativas, são dotadas de barras de rolagem, de maneira a permitir a edição e a conseqüente visualização de Statecharts cuja representação gráfica ultrapasse as dimensões de sua janela.

A forma de representação das Bolhas AND também difere daquela sugerida por Harel (HAREL, 1987b), que desenha essas Bolhas em uma Bolha maior, cuja área é dividida por linhas tracejadas, conforme apresentado no capítulo 3. Na ferramenta desenvolvida, as Bolhas AND têm o mesmo formato das Bolhas OR, sendo identificadas pela diferença de cor. As Bolhas do tipo AND são representadas com a cor amarela, as do tipo OR com a cor azul e as Bolhas primitivas, com a cor cinza.

Quanto às Ligações entre as Bolhas, estas podem ser dependentes e independentes. As Ligações do tipo independente representam eventos gerados externamente ao Statechart, provocando as transições iniciais. As Ligações dependentes representam eventos gerados internamente ao Statechart que estiver sendo executado, eventos estes causadores das transições intermediárias. As Ligações independentes são representadas por linhas tracejadas, e as dependentes por linhas contínuas.

Uma Ligação é caracterizada pela composição dos elementos seguintes: um evento, uma condição, um disparo, um tempo de atraso e um tempo mínimo. O evento é o elemento que dispara a transição representada pela Ligação. Esta transição ocorre se a condição indicada for satisfeita. Esta condição indica que uma determinada Bolha deve estar ativa, ou seja, deve ser a Bolha corrente, para que a transição seja efetuada. O disparo indica que uma outra Ligação deve ser ativada quando da ocorrência da transição em questão. O tempo de atraso indica quanto tempo que deve decorrer para que a transição seja efetuada, e o tempo mínimo indica o mínimo intervalo de tempo necessário para que a transição possa ocorrer.

O único elemento obrigatório na caracterização da Ligação é o evento. Se o campo de condição não for especificado, nenhuma condição será testada. Caso o campo de disparo não seja fornecido, não haverá Ligação a ser

ativada. Se os campos de tempo não forem especificados, não haverá quaisquer atrasos ou tempo mínimo de transição a serem respeitados. Na bolha-destino, junto à ligação é representada uma seta, definindo o sentido dessa ligação.

O tempo de execução de um Statechart é indicado pelo campo numérico rotulado **TEMPO** no canto esquerdo superior da janela principal de edição, conforme mostra a figura 4.1. Há uma função, acessível ao usuário, que permite zerar o valor desse campo quando for necessário.

É possível verificar se um nível de detalhamento do Statechart está compatível com os níveis imediatamente anterior e posterior. A compatibilidade em questão refere-se às entradas e saídas de cada Bolha, ou seja, cada entrada e cada saída de uma determinada Bolha deve obrigatoriamente estar presente nas representações de seus níveis anterior e posterior de detalhamento.

É possível, também, se obter uma visão compacta e resumida de cada Statechart editado pelo STAD, o que é muito útil nos casos de Statecharts extensos. Dispõe-se também de uma função que permite ao usuário obter a árvore completa da montagem do projeto, cuja utilidade está no fato de permitir que se visualize a estrutura integral de detalhamento do mesmo.

Outra função disponível na ferramenta permite a impressão da parte do Statechart que estiver sendo exibida na janela de edição.

Para uma visualização detalhada das tabelas do Banco de Dados, é dada a oportunidade de um acesso direto aos seus registros, por intermédio da incorporação à ferramenta, de um programa integrante do pacote do Visual Basic, denominado VISDATA. No entanto não é recomendável o uso deste recurso para a alteração de dados das tabelas, pois alterações indevidas em algum dos registros do Banco de Dados pode levar à sua inconsistência, e a conseqüente impossibilidade de utilização de um projeto já cadastrado. Este acesso direto ao Banco de Dados componente do STAD é uma função utilizada na fase de depuração do programa, podendo ser retirada da implementação final da ferramenta sem nenhum prejuízo à sua utilização.

Para a edição de Funções Adaptativas é utilizado o mesmo editor empregado para a montagem de Statecharts. Assim que uma Função Adaptativa é selecionada para ser utilizada por uma determinada Ligação, passa a ter uma instância, que é a Ação Adaptativa correspondente a esta instância.

Os tipos de ações que integram uma função adaptativa são a adição de Bolhas e Ligações e a exclusão de Ligações. A função de consulta não foi implementada na primeira versão da ferramenta. Na teoria apresentada para os autômatos adaptativos, a exclusão de Ligações depende da realização de uma consulta. No entanto, na ferramenta implementada, não é necessário se realizar uma consulta para se efetuar uma exclusão, bastando que se especifique qual é a Ligação a ser excluída. No primeiro acionamento da Ação Adaptativa, o comando de exclusão da Ligação especificada é executado. Se a Ação Adaptativa for acionada outras vezes, o mesmo comando de exclusão é ignorado.

Uma função adaptativa contém parâmetros que devem receber atribuições de valores quando da associação a uma ligação. Os parâmetros possíveis de serem definidos são a origem e o destino de ligações, o nome de bolhas, o nome de eventos associados às ligações e a definição de ligação a ser excluída.

Um Statechart pode ainda ser simulado, ou seja, a partir de um estado inicial, podem ser ativadas Ligações/Eventos que causam a transição entre os Estados ou Bolhas do Statechart. A execução de um Statechart pode ser feita de forma automática ou passo a passo. Para se atingir esse estado inicial existe uma função que desativa todas as Bolhas e Ligações eventualmente ativas em determinado momento da simulação. Há também uma função para se apagar a informação de qual foi último estado visitado, o que é útil quando se tem Bolhas com a característica de "história".

Durante a simulação de um Statechart, quando uma Bolha é ativada, ou seja, quando ela é a Bolha corrente (associando-se com o estado corrente em um autômato), a borda de sua representação é desenhada com linhas mais grossas e com a cor azul. Da mesma forma, quando da ocorrência de uma transição, a representação gráfica da Ligação responsável por ela também assume a cor azul, também representada através de linhas mais espessas.

Na simulação de um Statechart não é feita a verificação se apenas uma bolha OR em cada nível de detalhamento está ativada em um determinado instante. Isto possibilita a utilização do STAD para a representação e simulação de sistemas da mesma forma que se fossem representados através de autômatos.

4.2. DETALHAMENTO DO STAD

Neste item são detalhados os principais aspectos da ferramenta desenvolvida, procurando-se, em cada item, mostrar as características mais importantes e seu papel no contexto global de utilização.

Maiores detalhes a respeito do modo de utilização da ferramenta desenvolvida são apresentados no Manual de Operação do STAD, no Anexo A.

4.2.1. Banco de Dados

A estrutura dos Projetos, ou seja, seus elementos, são armazenados em um Banco de Dados, desenvolvido utilizando-se o Gerenciador de Banco de Dados ACCESS versão 1.0. Desta forma, antes de se executar qualquer operação em um Projeto, é necessário que seja aberto esse Banco de Dados, para que se tenha acesso ao detalhamento de cada Projeto já cadastrado, ou para que se faça a abertura de novos Projetos. O nome do arquivo que contém o Banco de Dados utilizado neste sistema é "**tese1.mdb**". A abertura deste Banco de Dados é feita logo no início de utilização do STAD.

Uma descrição dos campos das tabelas utilizadas no Banco de Dados é apresentado adiante, no anexo B.

4.2.2. Edição de Projetos

As funções implementadas na ferramenta, que permitem a manipulação de um Projeto são:

a) Criação

Esta função permite a criação de um novo Projeto. Sempre, ao se criar um Projeto, é necessário definir o seu nome, o qual não pode coincidir com o nome de qualquer outro Projeto já existente. Ao se iniciar a criação de um Projeto, a edição de seus elementos inicia-se sempre pelo nível zero de detalhamento. Tais verificações são efetuadas automaticamente pela ferramenta.

b) Alteração/Edição

Esta função permite a alteração ou a edição de um Projeto já cadastrado no sistema. Sempre que ocorrer a abertura de um Projeto para alteração, ela será feita em seu nível zero de detalhamento. Se um Projeto já estiver sendo exibido na janela correspondente, este será automaticamente fechado antes de se carregar o Projeto selecionado para alteração/edição, de forma que, em cada instante, apenas um projeto, no máximo, esteja aberto na ferramenta..

c) Exclusão

Esta função permite a eliminação de um Projeto já cadastrado no sistema. Desta forma, possibilita-se excluir do sistema um Projeto que já não tenha mais utilidade. Para se excluir o Projeto selecionado é necessário, antes que se efetue a confirmação de sua exclusão, o que ocorre por intermédio da exibição de uma mensagem solicitando a confirmação ou não da exclusão do Projeto selecionado. Se, for confirmada a exclusão do Projeto, o mesmo será completamente retirado completamente das tabelas do Banco de Dados.

d) Alteração de Nome

Esta função permite a alteração do nome de um Projeto já existente com outro nome. Assim, possibilita-se a alteração do nome de um Projeto para um nome que venha a se mostrar mais conveniente. O novo nome escolhido par o Projeto não pode coincidir com o nome de qualquer dos Projetos já existentes no Banco de Dados do sistema na ocasião.

e) Cópia de Projeto

Esta função permite que se efetue a cópia dos dados de um Projeto em um novo Projeto, com outro nome. A finalidade desta função é a de possibilitar que se salve uma determinada configuração, enquanto se testam novas evoluções do

Projeto original. O nome escolhido para a reprodução do Projeto não pode coincidir com o nome de nenhum dos Projetos já existentes no Banco de Dados do sistema.

4.2.3. Edição de Bolhas

As representação de bolhas, adotadas na implementação da ferramenta correspondem ao ao elemento **B** descrito no item 3.2.4.1, de formalização dos Statecharts. As funções implementadas na ferramenta, que permitem a edição das Bolhas de um Statechart, são:

a) Criação de Bolhas

Esta função permite a criação de Bolhas em um Statechart, sendo possível a criação de três tipos de Bolhas: OR, AND e Primitiva. As Bolhas dos tipos OR e AND não podem ter nomes repetidos, nem nomes deixados em branco. Se ocorrer a tentativa de levar o projeto a uma dessas duas condições, mensagens de erro serão exibidas, solicitando-se sua correção.

O modo de se indicar a posição de uma Bolha na janela de edição de Statecharts é comum para estes três tipos de Bolhas, devendo-se, para tanto, acionar o mouse na posição desejada para a Bolha. Quando o mouse é acionado, é exibida a moldura de representação da Bolha, cuja posição poderá ser então melhor determinada arrastando-se o indicador do mouse até que a Bolha atinja a posição desejada.

Uma Bolha do tipo OR pode ainda ser de um dos seguintes tipos:

- **Default**: é a Bolha que assumida como a Bolha corrente quando se executa um detalhamento pela primeira vez;
- **History**: indica que, no detalhamento de uma Bolha deste tipo, a última Bolha visitada é a que será a assumida como a Bolha corrente quando se executar tal detalhamento uma próxima vez; e
- **Comum**: refere-se a Bolhas que não são nem do tipo Default, nem do tipo History.

Uma Bolha do tipo Primitiva pode ainda ser de um dos seguinte tipos:

- **soma**: permite a operação de soma entre dois números reais (X + Y);
- **subtração**: permite a operação de subtração entre dois números reais (X - Y);
- **multiplicação**: permite a operação de multiplicação entre dois números reais (X * Y);

- **divisão**: permite a operação de divisão entre dois números reais (X / Y);
- **exponenciação**: permite a operação de exponenciação entre dois números reais (X ^ Y);
- **igual**: permite verificar se dois valores são iguais (X = Y ?) ;
- **diferente**: permite verificar se dois valores são diferentes (X <> Y ?);
- **menor**: permite verificar se um valor é menor que outro (X < Y ?);
- **maior**: permite se verificar se um valor é maior que outro (X > Y ?);
- **maior ou igual**: permite verificar se um valor é maior ou igual a outro (X >= Y ?);
- **menor ou igual**: permite verificar se um valor é menor ou igual a outro (X <= Y ?);
- **OR**: permite a operação lógica OR entre dois valores booleanos (X OR Y);
- **AND**: permite a operação lógica AND entre dois valores booleanos (X AND Y);
- **NOT**: permite a operação lógica NOT de um valor booleano (NOT X);
- **IF THEN**: permite o teste de uma condição sobre um valor e a execução da seqüência de Bolhas representada após este teste, em caso da condição resultar verdadeira (IF X THEN);
- **IF ELSE**: permite o teste de uma condição sobre um valor e a execução da seqüência de Bolhas representada após este teste, em caso da condição resultar falsa (IF X ELSE);
- **DO WHILE**: permite a execução da seqüência de Bolhas representada no Statechart enquanto a condição X for verdadeira (DO ... WHILE X);

Quando uma Bolha Primitiva é criada, designam-se os nomes de seus parâmetros de entrada como sendo X e Y (ou apenas X nos quatro últimos casos acima listados). Para que uma Bolha Primitiva possa ser simulada, é necessário que seus parâmetros de entrada sejam associados às Ligações que têm como destino tal Bolha. Após realizada essa associação, os nomes dos parâmetros de entrada assumem o nome dos eventos das Ligações designadas para isso.

As bolhas primitivas constituem-se em um tipo de bolha OR e foram implementadas para possibilitar uma simulação dos Statecharts a nível de valores, permitindo montar um sistema desde os seus componentes mais básicos.

b) Alteração de Bolhas

Esta função permite a alteração do nome de Bolhas de um Statechart. Podem ser alterados o campo do nome da Bolha, e seu Tipo da Bolha (no caso de Bolhas OR). O novo nome escolhido para a Bolha não pode coincidir com o nome de qualquer das Bolhas já existentes no Projeto que estiver sendo editado.

c) Exclusão de Bolhas

Esta função permite a exclusão de Bolhas de um Statechart. Para se excluir a Bolha selecionada, é necessário, antes, que seja confirmada a sua exclusão, o que é efetuado por intermédio da exibição de uma mensagem, solicitando a confirmação ou não da exclusão da Bolha selecionada Caso seja confirmada a exclusão da Bolha, a mesma é eliminada, bem como todas as eventuais Ligações existentes, tendo como origem ou destino a Bolha excluída. Tanto a Bolha, como essas Ligações, são então retiradas das tabelas do Banco de Dados.

d) Movimentação de Bolhas

Esta função permite a movimentação física das representações das Bolhas dentro da janela de edição de Statecharts. Através desta função é possível alterar o posicionamento de Bolhas e de Ligações a elas associadas, conforme se mostre necessário, com a montagem do Statechart em um determinado nível de detalhamento. O modo de se efetuar esta movimentação é comum para os três tipos de Bolhas, devendo-se selecionar a Bolha que se quer mover, através do acionamento do mouse posicionado na área mais externa de representação da Bolha, e arrastando-a para o novo local selecionado. Eventuais Ligações existentes na Bolha também serão movidas automaticamente, adequando-se à nova posição da Bolha.

e) Alteração do Tamanho de Bolhas

Esta função permite a alteração do tamanho de Bolhas dentro da janela de edição de Statecharts. Através desta função é possível alterar o tamanho da representação das Bolhas, conforme se mostre necessário, com a montagem do Statechart de um determinado nível de detalhamento. O modo de se efetuar esta alteração de tamanho é comum para os três tipos de Bolhas, devendo-se selecionar a Bolha que se quer alterar, por meio do acionamento do botão do mouse posicionado na área mais externa de representação da Bolha, arrastando as bordas laterais direita e inferior da mesma, até que seja atingido o novo tamanho desejado.

4.2.4 Edição de Ligações

As ligações aqui representadas correspondem ao conjunto de "labels" **L** descrito no item 3.2.4.1 da formalização de Statecharts. As funções implementadas na ferramenta, que permitem a edição de Ligações de um Statechart, são:

a) Criação

Esta função permite a criação de Ligações em um Statechart, permitindo que sejam gerados os seguintes tipos de Ligações:

- **Dependente Interna**: permite a criação de uma Ligação Dependente Interna, ou seja, uma Ligação que tem origem e destino em Bolhas do Statechart correntemente em uso e cuja transição depende exclusivamente de um evento originado internamente ao Statechart;
- **Dependente Externa de Entrada**: permite a criação de uma Ligação Dependente Externa de Entrada, ou seja, uma Ligação que tem origem externa e destino em uma Bolha do Statechart e cuja transição depende exclusivamente um evento originado internamente ao Statechart;
- **Dependente Externa de Saída**: permite a criação de uma Ligação Dependente Externa de Saída, ou seja, uma Ligação que tem origem em uma Bolha e destino externo ao Statechart e cuja transição depende exclusivamente de um evento originado internamente ao Statechart;
- **Independente Interna**: permite a criação de uma Ligação Independente Interna, ou seja, uma Ligação que tem origem e destino em Bolhas do próprio Statechart, e cuja transição depende de um evento originado externamente ao Statechart;

- **Independente Externa de Entrada**: permite a criação de uma Ligação Independente Externa de Entrada, ou seja, uma Ligação que tem origem externa e destino em uma Bolha do Statechart e cuja transição depende exclusivamente de um evento originado externamente ao Statechart;
- **Independente Externa de Saída**: permite a criação de uma Ligação Independente Externa de Saída, ou seja, uma Ligação que tem origem em uma Bolha e destino externo ao Statechart e cuja transição depende exclusivamente de um evento originado externamente ao Statechart.

A forma através da qual se pode traçar graficamente o caminho de uma Ligação, é comum às Ligações de todos esses tipos especificados, bastando selecionar, com o mouse, a região externa da Bolha que se deseja como origem de uma Ligação, e, mantendo pressionado o botão do mouse, arrastar seu ponteiro até a posição de destino do segmento de reta que se deseja traçar. Desta forma o ponteiro do mouse vai sendo deslocado na tela, e à medida que vai se deslocando, traça, como linha quebrada, a Ligação desejada. A origem da Ligação com relação à borda da Bolha é automaticamente posicionada pela ferramenta, assim que o botão do mouse é liberado. Se o local da tela onde o botão do mouse deixou de ser pressionado for uma região onde não exista alguma Bolha, deverá ser repetida a mesma operação de pressionar o botão do mouse e arrastá-lo até o local pretendido, e assim sucessivamente, até que o botão do mouse seja liberado na área externa de alguma Bolha que se tenha escolhido com sendo o destino da Ligação. A posição do destino da Ligação com relação à borda da Bolha é automaticamente escolhida pela ferramenta, assim que o botão do mouse é liberado. Tratando-se de uma Ligação de entrada, o ponto inicial da Ligação deve ser uma região onde não exista Bolha, o mesmo ocorrendo se for uma Ligação de saída, e neste caso o ponto final da Ligação também deve ser alguma região onde não exista Bolha.

Após a escolha do caminho físico da representação gráfica de uma Ligação, deve-se especificar seus parâmetros, ou seja:

- **Evento**: Representa o nome que a Ligação recebe quando é criada. Um evento não pode ter seu nome deixado em branco. Se isto ocorrer, é exibida uma mensagem de erro informando do ocorrido e solicitando sua correção.

- **Condição**: Representa a condição que deve ser satisfeita para que a transição para a Bolha-destino seja executada. A condição deve ser preenchida com o nome de uma Bolha. Se a Bolha selecionada for a Bolha corrente no momento de se executar a transição da Ligação que aqui estiver sendo criada, tal transição é realizada. Caso contrário, a transição não ocorre.
- **Disparo**: Representa o nome de um evento que deve ser acionado quando da execução da Ligação que aqui estiver sendo criada. O disparo deve ser preenchido com o nome de um outro evento. Quando da execução desta Ligação, o evento marcado como disparo, deve ser ativado.
- **Atraso**: Representa o atraso que deve ocorrer para que seja realizada a transição representada pela Ligação.
- **Instante de Disparo**: Representa o tempo mínimo absoluto decorrido desde o início da execução do Statechart, para que a seja efetuada a transição para a Bolha-destino.
- **Valor**: Representa o valor assumido pela Ligação quando de sua criação, tendo como finalidade a execução de cálculos no caso de seu destino ser uma Bolha Primitiva.
- **Tipo de Ligação**: Representa o tipo de Ligação que estiver sendo criada, podendo ou não ser Adaptativa. Neste último caso é possível associar uma Função Adaptativa à Ligação em questão.

b) Alteração

Esta função permite a alteração dos parâmetros de uma Ligação, ou seja, os nomes de seu Evento, Condição, Disparo, Atraso, Tempo de Disparo, Valor e Tipo de Ligação, conforme detalhados acima. Na alteração destes parâmetros, o novo nome atribuído ao Evento não pode ser deixado em branco. Se isto acontecer, uma mensagem de erro é exibida, solicitando a correção do erro.

O modo de se selecionar a Ligação a ser alterada consiste em selecionar-se, com o mouse, qualquer região extrema de algum segmento da Ligação que se deseje alterar. Isto promove a abertura de uma janela para as alterações que se deseja efetuar.

c) Exclusão

Esta função permite a eliminação de Ligações de um Statechart. Para se excluir uma Ligação selecionada é necessário, antes, confirmar a sua exclusão. Isto é feito através da exibição de a uma mensagem solicitando a resposta se a Ligação selecionada deve ou não ser excluída. Se a exclusão for confirmada, a Ligação é suprimida, tanto do Statechart, como das tabelas do Banco de Dados utilizado pelo sistema. Se a Ligação, selecionada para ser excluída, estiver associada a qualquer Função Adaptativa, será exibida uma mensagem informando que a mesma só poderá ser excluída após a remoção da associação com a Função Adaptativa.

Assim como para a alteração de parâmetros de uma Ligação, o modo de se escolher a Ligação a ser excluída é feito selecionando-se, através do acionamento do botão esquerdo do mouse, qualquer região extrema de algum dos segmentos que representam a Ligação que se deseja excluir.

4.2.5. Níveis de Detalhamento

Os níveis de detalhamento aqui implementados correspondem à função ρ descrita no item 3.2.4.1 de formalização de Statecharts.

Esta função permite ao usuário da ferramenta efetuar o detalhamento de uma Bolha. Qualquer Bolha do tipo OR ou AND pode ser detalhada em um nível inferior de detalhamento. Podem ser feitos tantos níveis de detalhamento quantos forem necessários. Para cada nível de detalhamento que se crie, é aberta uma nova janela para a edição do Statechart correspondente. O detalhamento vai sendo executado no sentido da profundidade da árvore de hierarquia das Bolhas. Para que se efetue o detalhamento de uma outra Bolha de um dado Statechart, é necessário que se retorne ao nível de detalhamento ao qual pertence a Bolha que se deseja detalhar.

Para se indicar uma Bolha a ser detalhada, é necessário selecioná-la com a ajuda do mouse, acionando-o sobre a área mais externa da Bolha selecionada. Após ter sido selecionada a Bolha, é exibida uma nova janela de edição de Statecharts, permitindo a edição do detalhamento da Bolha selecionada. Se já havia algum detalhamento previamente editado para a Bolha selecionada, o mesmo será exibido, permitindo que se façam as alterações adicionais necessárias. Caso contrário, uma janela vazia será aberta.

4.2.6. Funções Auxiliares

Foram implementadas algumas funções que auxiliam o usuário, principalmente, a verificar o estado da edição de cada Statechart. Estas funções são:

a) Exibição de Zoom

Esta função permite que se realize a montagem e exibição de uma vista reduzida, correspondente ao Statechart da Bolha que estiver sendo detalhada. Tal exibição deste é feita por meio de um quadro contendo a respectiva vista reduzida do detalhamento da Bolha. A montagem deste zoom visa permitir uma visualização completa do Statechart da Bolha que estiver sendo detalhado, principalmente no caso de este ultrapassar as dimensões máximas da Janela Principal. A figura 4.2. apresenta o formato desse zoom.

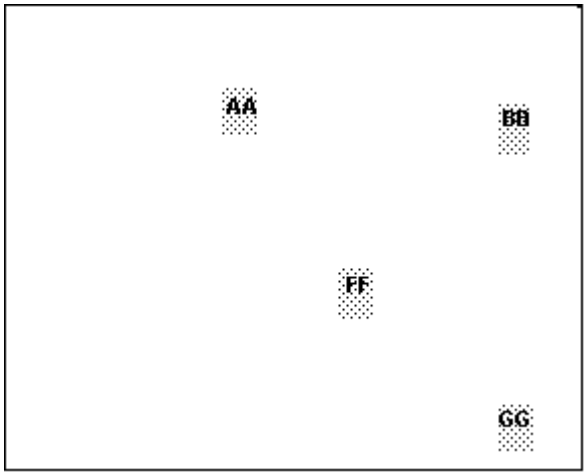


Figura 4.2 - Formato da exibição de Zoom

b) Lista de Inconsistências

Esta função permite que se realize uma verificação das inconsistências das Ligações de origem ou destino de cada Bolha, com seus níveis de detalhamento imediatamente inferior e superior. As inconsistências testadas referem-se às seguintes verificações:

- se uma determinada Ligação que consta em um dado nível de detalhamento do Statechart também existe nos níveis inferior e superior de detalhamento da Bolha;
- se existe realmente uma Bolha no nível de detalhamento correspondente, cujo nome seja o mesmo de uma determinada condição associada a uma Ligação; e

- se existe realmente uma Ligação no nível de detalhamento correspondente, cujo nome de evento associado seja o mesmo de um determinado disparo associado a uma Ligação.

c) Montagem de Árvore

Esta função permite que se realize a montagem e exibição de uma árvore que contém a estrutura hierárquica das Bolhas representadas no Projeto que estiver sendo montado. A exibição desta árvore é feita por meio de um quadro, contendo a respectiva árvore do Projeto. A montagem desta árvore visa facilitar uma visualização completa de toda a estrutura do Projeto que estiver sendo montado, em todos os seus níveis de detalhamento. A figura 4.3 apresenta o formato dessa árvore.

Figura 4.3 - Formato da Árvore de Hierarquia de Bolhas

d) Zera Relógio

Esta função tem como finalidade iniciar a contagem de tempo no relógio utilizado para a marcação, em época de execução, do tempo decorrido a partir do início da simulação do Statechart de um Projeto.

e) Impressão do Statechart

Esta função tem como finalidade permitir a impressão da parte da janela que estiver sendo exibida, quando da edição de um determinado detalhamento de um Statechart. A área impressa é a correntemente exibida na janela, no momento da solicitação da impressão. Emite-se uma mensagem solicitando confirmação ou não da área de impressão. Se esta for confirmada, ocorre a impressão, caso a impressora esteja conectada e ligada. Caso contrário, impressão não será realizada, sendo então emitida uma mensagem de erro. Não há a opção de se imprimir toda a área da janela em que se encontra representado um Statechart, caso suas dimensões ultrapassem as dimensões máximas da janela.

f) Acesso ao Banco de Dados

Esta função tem como finalidade permitir a chamada do programa VISDATA, conforme já explicado no item 4.1.2. A finalidade desta chamada ao Banco de Dados é possibilitar a confirmação de informações exibidas de forma gráfica, pelo editor e executor de Statecharts Adaptativos, bem como permitir eventuais correções que se façam necessárias, em decorrência de algum erro que possa vir a ocorrer com o programa. O programa VISDATA contém um "help" bastante satisfatório, podendo ser utilizado em caso de dúvida na utilização de qualquer de suas funções.

4.2.7. Funções Adaptativas

A implementação das funções adaptativas inseridas na ferramenta se baseia na teoria apresentada no item 3.2.5.

As Funções Adaptativas têm a finalidade de incluir a característica adaptativa nos Statecharts, fazendo com que, quando se fizer necessário, se possa associar a uma ligação, presente em um Statechart, uma Ação Adaptativa. Tal ação constitui uma instância de uma Função Adaptativa representada no STAD.

Dessa forma, neste item são descritas as características referentes à montagem e edição de Funções Adaptativas.

a) Criação

Esta função permite a criação de uma nova Função Adaptativa. Sempre, ao se criar uma Função Adaptativa, é necessário definir o seu nome, que também não pode coincidir com o nome de outra Função já existente.

b) Alteração/Edição

Esta função permite a alteração ou a edição de uma Função Adaptativa já cadastrada para o Projeto que estiver sendo detalhado. Se uma Função já estiver sendo exibida na janela correspondente, é apagada da janela antes de se carregar a Função selecionada para alteração/edição.

c) Exclusão

Esta função permite a exclusão de uma Função Adaptativa já cadastrada no sistema. Desta forma, possibilita-se eliminar uma Função Adaptativa que já não tenha mais utilidade no Banco de Dados. Para se excluir a Função selecionada é necessário, que antes seja confirmada a sua exclusão, pela resposta a uma mensagem solicitando a confirmação ou não da exclusão da Função selecionada. Se for confirmada a exclusão da Função, esta será retirada das tabelas do Banco de Dados.

d) Alteração de Nome

Esta função permite a alteração do nome de uma Função Adaptativa anteriormente criada com outro nome. Assim, possibilita-se a alteração do nome de uma Função para um nome que venha a se mostrar mais conveniente. O novo nome escolhido para a Função não pode coincidir com o nome de qualquer das Funções já existentes no Projeto que estiver sendo editado.

e) Cópia de Função

Esta função permite que se efetue a cópia de todos os dados de uma Função Adaptativa em uma nova Função, com outro nome. A finalidade desta função é a de possibilitar que se salve, com outro nome, a versão corrente de uma determinada Função, enquanto se testam novas configurações na Função original. O nome escolhido para a reprodução da Função Adaptativa não pode coincidir com o nome de uma das Funções já existentes no Projeto que estiver sendo editado.

f) Elementos de Funções Adaptativas

A representação das Funções Adaptativas é feita da mesma forma que os elementos convencionais dos Statecharts. Bolhas e Ligações que se queiram adicionar assumem a mesma representação das Bolhas e Ligações anteriormente apresentadas.

Os elementos que podem estar presentes em uma Função Adaptativa são: adição de Bolha, adição de Ligação e Exclusão de Ligação.

Quando uma chamada de uma Função Adaptativa é associada a uma Ligação, é criada uma condição para, toda vez que tal ligação for executada, venha a

ser também executada, como decorrência, um Ação Adaptativa. O que distingue a representação dos elementos de uma Ação Adaptativa daquela dos elementos do Statechart convencional é que no primeiro caso as bordas das Bolhas e os segmentos das Ligações são tracejados, enquanto que no segundo caso são contínuos.

O modo como os elementos de uma Função Adaptativa são montados na janela de edição é o mesmo que foi previamente descrito para a montagem de elementos não-adaptativos do Statechart. Dessa forma, a criação, alteração e exclusão de Bolhas e Ligações é realizada da mesma maneira que para os elementos não-adaptativos.

A única diferença com relação aos elementos não-adaptativos é a existência de quatro Bolhas com moldura mais grossa na janela de montagem de Funções Adaptativas. A finalidade destas Bolhas é a de servir como origem/destino de Ligações que se deseja que a respectiva origem ou destino seja um parâmetro da Função Adaptativa que se estiver montando. Se uma destas Bolhas for origem/destino de uma Ligação, esta aparecerá na lista de parâmetros da janela de seleção de parâmetros da Ação Adaptativa como origem/destino dessa Ligação, e portanto como um parâmetro a ser definido.

Os nomes das Bolhas e dos Eventos das Ligações podem ser selecionados para que sejam parâmetros da Função Adaptativa

g) Designação de Funções Adaptativas

Esta função permite que se realize a associação a uma Ligação de uma Função Adaptativa a ser escolhida entre as Funções Adaptativas já existentes no Projeto que estiver sendo exibido. Após ter sido feita esta seleção, é criada uma instância da Função Adaptativa, que é a Ação Adaptativa. A associação de uma Função Adaptativa também pode ser desfeita. A instância da Função Adaptativa associada, ou seja, a Ação Adaptativa, é inserida no Statechart correspondente, na janela de edição, em uma posição apontada pelo usuário. Se uma outra Ação Adaptativa já estava associada à Ligação, é removida do Statechart antes da inserção da nova Ação associada

A execução da Ação Adaptativa pode ocorrer antes ou após a execução da transição associada à Ligação em questão. Dessa forma, há a possibilidade da escolha da ordem de execução da Ação Adaptativa, que pode ser anterior ou posterior. Se for escolhida a opção anterior, a Ação deverá ser realizada antes da transição para as Bolhas correspondentes. Se for escolhida a opção

posterior, a Ação deverá ser realizada após a transição para as Bolhas correspondentes.

h) Designação de Argumentos

Esta função permite que se faça a alteração de argumentos parâmetros de uma Ação Adaptativa. Para tanto, deve ser selecionada uma Ligação que tenha uma Ação Adaptativa associada, o que é feito acionando-se o mouse, em qualquer região extrema de qualquer segmento da Ligação escolhida.

Os argumentos cuja atribuição pode ser realizada são: origem e destino de Ligações, nome de Bolhas, nome de evento de Ligações e nome de Ligação a ser excluída. Uma ligação inserida por meio de uma Ação Adaptativa pode ter sua origem e seu destino definidos como argumentos, ou seja, a origem e o destino desta Ligação poderão ser alterados quando da simulação do Statechart. O mesmo ocorre com o nome de Bolhas, com o nome de evento associado à Ligação e com a seleção de Ligação a ser excluída.

4.2.8. Simulação

A simulação consiste em se fazer com que o Statechart representado seja estimulado, inicialmente por meio dos eventos externos, para que as Bolhas correspondentes sejam ativadas, ou ainda, sejam as Bolhas correntes. A simulação se inicia pelo nível zero de detalhamento de um Projeto e pode prosseguir até quantos níveis de detalhamento tenham sido construídos.

a) Modo de Simulação

Esta função possibilita que se faça a seleção do modo de operação da ferramenta entre os modos manual e automático de simulação de Statecharts. Durante uma simulação, no modo manual, para que o Statechart possa transitar uma vez entre os seus estados, é necessário acionar a função execução de passo. Durante uma execução, no modo automático, o Statechart transita entre os estados automaticamente, sem a necessidade de acionamento da função de execução de passo. As transições são executadas até que não haja mais possibilidade de alteração do estado do Statechart, no detalhamento de Bolha que estiver sendo executado.

b) Modo de Operação

Esta função possibilita que se faça uma escolha entre os modos de operação sem detalhamento, detalhamento completo e seleciona detalhamento. No modo sem detalhamento, apenas o nível 0 do Statechart é executado, não sendo considerados eventuais detalhamentos de Bolhas existentes. No modo com detalhamento completo, todos os níveis de detalhamento de todas as Bolhas são executados, sem necessidade de confirmação por parte do usuário. No modo de seleção de detalhamento, é exibida uma mensagem solicitando a confirmação da execução do detalhamento das Bolhas correspondentes. Se houver confirmação, o detalhamento em questão será executado.

c) Entradas Externas

Esta função permite a exibição de uma lista que contém todas as entradas independentes externas ao nível de detalhamento que estiver sendo exibido. Esta lista possibilita a ativação manual de qualquer das entradas independentes externas, quando da execução do Statechart. Quando esta lista já estiver sendo exibida, e a função for selecionada, é feita uma "limpeza" do Statechart, ou seja, os elementos criados ou excluídos por meio de ações adaptativas executadas são restituídos ao seu estado original, ou seja, os elementos criados são excluídos e os elementos excluídos são restituídos ao Statechart.

d) Execução de Passo

Esta função permite que se execute um passo do Statechart, ou seja, durante uma execução, o Statechart transita uma vez entre os estados a cada chamada desta função. As transições são executadas até que não haja mais possibilidade de alteração do estado do Statechart, no detalhamento de Bolha que estiver sendo executado.

e) Valores

Esta função permite a exibição em uma janela aberta especialmente para este fim, dos valores que as Ligações assumem conforme o Statechart vai sendo executado/simulado. Para se obter esses valores (entradas e saída de uma Bolha), deve-se selecionar a Bolha à qual estão associados os valores se deseja visualizar.

f) Atraso/Tempo de Disparo

A função de atraso permite que, durante a execução de um Statechart, uma determinada transição para uma Bolha-destino de uma Ligação só seja efetuada após transcorrido o tempo especificado no campo atraso desta Ligação.

A função tempo de disparo permite que, durante a execução de um Statechart, uma determinada transição para uma Bolha-destino de uma Ligação só seja efetuada quando o intervalo especificado no campo tempo de disparo desta Ligação já houver transcorrido.

Para que esse tempos possam ser conferidos pelo usuário, na janela de edição principal há um marcador que indica o número de unidades de tempo transcorridas desde o início da execução do Statechart em um determinado detalhamento, pois o tempo marcado não é global.

g) Limpa e Limpa História

A função Limpa permite colocar o Statechart em uma situação inicial em que os elementos criados ou excluídos por meio de ações adaptativas executadas são restituídos à sua situação original, ou seja, os elementos criados adaptativamente são excluídos e os elementos excluídos adaptativamente são restituídos ao Statechart.

A função Limpa História permite eliminar a memória da informação do último estado visitado por um determinado detalhamento de uma Bolha. Esta função tem utilidade quando o detalhamento de uma Bolha do tipo History é executado.

4.2.9 Modo de Simulação

Apresenta-se a seguir o modo como é feita a simulação de um Statechart Adaptativo, no STAD, com o auxílio do exemplo visualizado na figura 4.4. Na parte (a) dessa figura, está representado o Statechart original, que é constituído pelas bolhas **A** e **B** e pelas ligações **x**, **y** e **z**. À ligação **y** está associada uma Ação Adaptativa, representada pelas bolhas **C1** e **D1** e pelas ligações **s1**, **t1** e **u1**. Note-se que os elementos da Ação Adaptativa são representados por símbolos com bordas e segmentos tracejados.

Para que se possa verificar se há alguma ação adaptativa associada a uma ligação, deve-se selecionar a opção de alteração dos atributos da ligação. Se já houver uma ação adaptativa associada à ligação, seu nome é exibido em uma janela específica para a seleção da ação adaptativa associada, bem como para se definir se tal ação deve ser executada anterior ou posteriormente à transição indicada pela ligação. Se ainda não houver uma ação adaptativa associada à ligação selecionada, o campo reservado para o nome dessa ação é deixado em branco.

A cada associação de uma ligação com uma ação adaptativa, são atribuídos índices aos elementos criados por meio dessa ação. Assim, na figura 4.4, os elementos criados pela ação adaptativa têm o índice **1** (**C1, D1, s1, t1** e **u1**). Se a mesma ação for associada a outra ligação, os elementos criados terão o índice **2** (**C2, D2, s2, t2** e **u2**).

Na figura 4.4b, está representada, com linha mais grossa, a ativação da ligação **x**. Em um passo seguinte, verifica-se que a ativação da ligação **x** provoca a ativação da bolha **A**. Em um próximo passo da simulação, deve ser ativada a ligação **y**. No entanto, como à ligação **y** está associada uma Ação Adaptativa, e supondo-se que tal ação seja do tipo anterior, o que ocorre é a realização dessa Ação Adaptativa, antes da ativação de **y**. Tal realização é representada na figura 4.4d, na qual se nota que os elementos da Ação Adaptativa estão representados não mais por linhas tracejadas, mas sim por linhas contínuas, indicando que já fazem parte do Statechart, como elementos que podem ser ativados.

Finalmente, na figura 4.4e está representada a ativação das ligações **y** (original do Statechart) e **s1** (criada adaptativamente).

Em qualquer instante da simulação, se for acionada a opção Limpa mencionada no item 4.2.8g, a configuração do Statechart voltará a ser a da figura 4.4a.

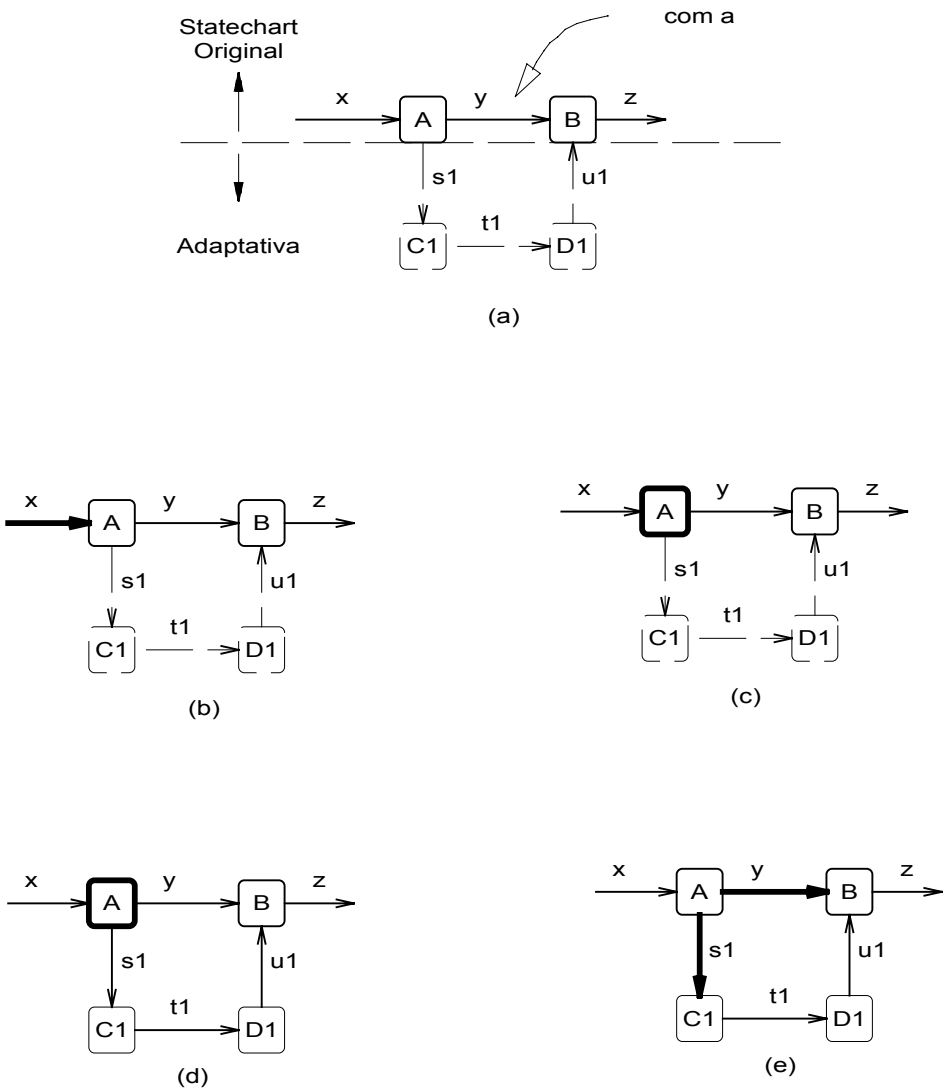


Figura 4.4 - Exemplo de Simulação de um Statechart Adaptativo

4.3. ESTRUTURA DO STAD

Neste item é descrita a constituição interna do sistema STAD, visando propiciar uma compreensão de sua estrutura. A descrição é feita por intermédio de Statecharts, embora a estrutura seqüencial do programa não permita explorar todo o potencial da representação. Tal descrição serve como um primeiro exemplo prático da utilização da notação dos Statecharts.

O STAD é constituído basicamente por módulos que manipulam as informações contidas no Banco de Dados do sistema. A figura 4.5 apresenta a estrutura geral do software do STAD. Há um módulo que possibilita a Entrada no Sistema, sendo que após o seu reconhecimento, evolui-se para o módulo de Seleção de Arquivos, no qual deve ser carregado o arquivo que contém o Banco de Dados, que por sua vez contém as informações sobre os projetos armazenados no sistema.

Após ter sido feita a seleção desse arquivo, deve ser feita a abertura de um projeto já existente, ou a criação de um novo projeto, o que é feito por intermédio do módulo de Abertura de Projeto. Tendo ocorrido a seleção de um projeto, segue-se o módulo de Gerenciamento dos Comandos, que é o responsável pelo tratamento de todos os comandos presentes no sistema. Os comandos disponíveis são os de Tratamento de Projetos, Tratamento de Funções Adaptativas, Tratamento de Funções Auxiliares, Tratamento de Simulação, Edição de Bolhas e Edição de Ligações.

A solicitação para a utilização de cada um desses módulos é feita por intermédio de um comando acionado pelo usuário do sistema, e a volta ao módulo de controle é feita por meio do comando de reconhecimento, também acionado pelo usuário, em cada um dos respectivos módulos. Este também é o meio para a ativação dos demais módulos descritos nos detalhamentos abaixo apresentados.

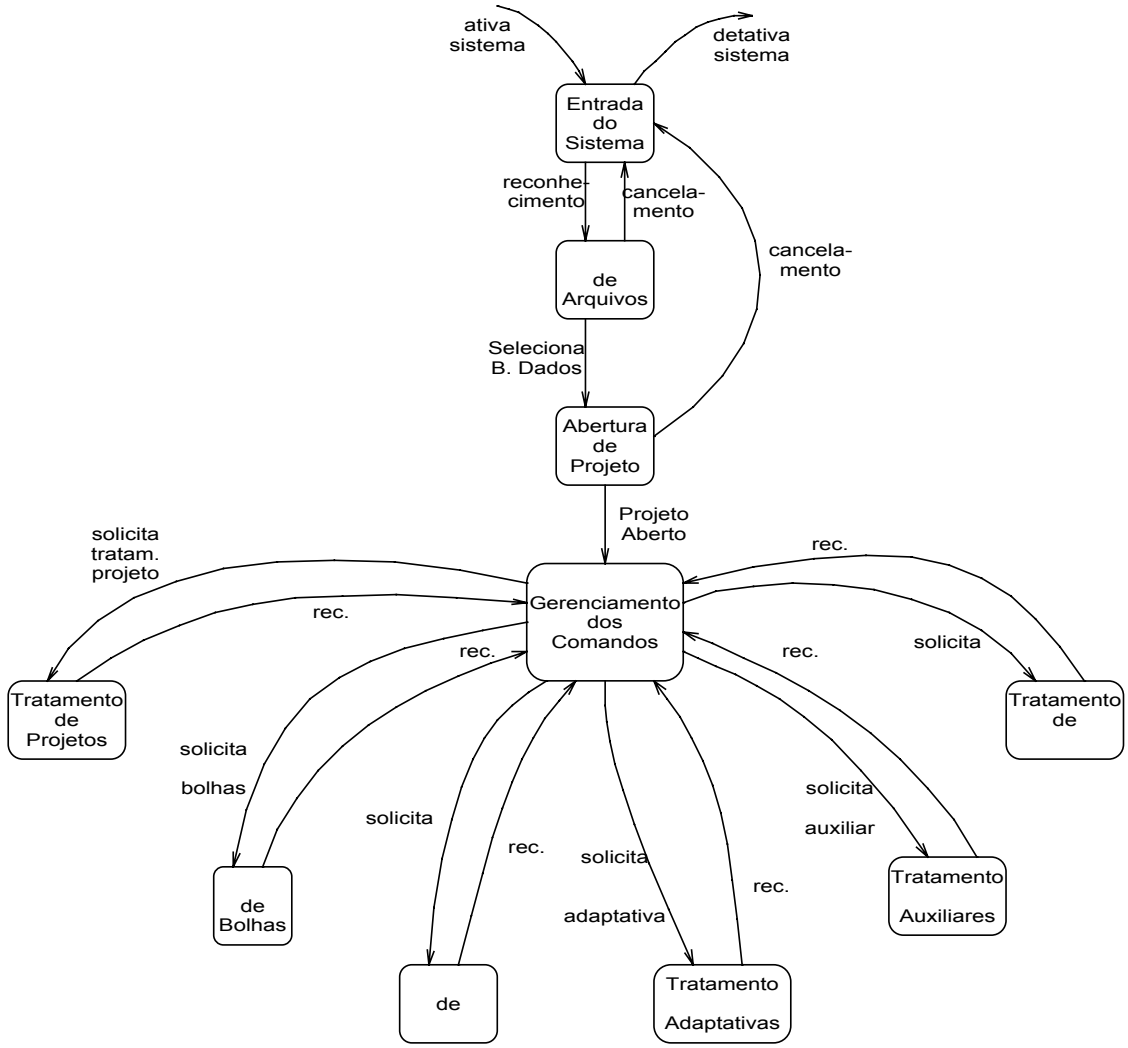


Figura 4.5 - Estrutura Geral do Software do STAD

O módulo de Tratamento de Projetos é detalhado na figura 4.6. Este módulo é composto por um módulo central que é o de Controle dos Comandos de Projeto, que executa o controle de todos os comandos relativos a um projeto, ou seja, Abertura e Exclusão de Projeto, Criação de Novo Projeto, Alteração de Nome de Projeto e Salvamento de Projeto com outro Nome. Nesse três últimos casos também é feita a verificação de igualdade do nome de Projeto com os nomes dos Projetos já existentes, de forma a não permitir duplicação de nomes. As funções executadas nesses módulos já foram descritas no item 4.2.2. Todas essas operações são acompanhadas da correspondente atualização do Banco de Dados.

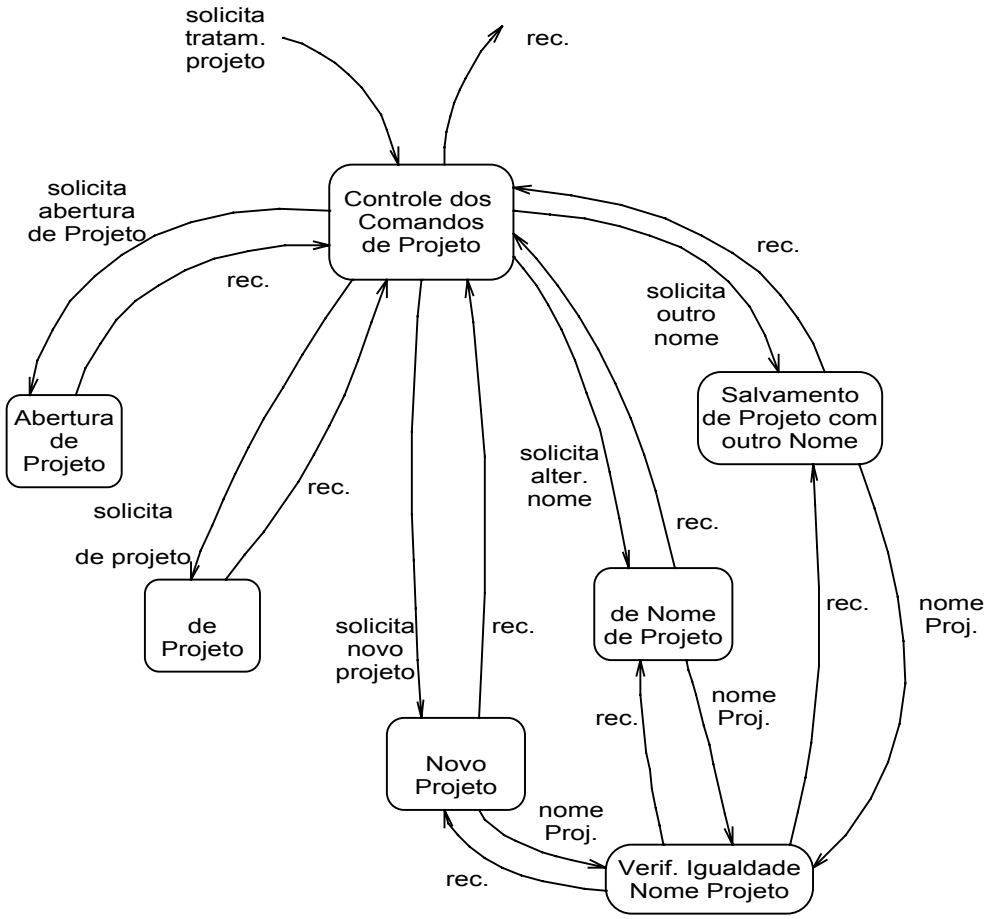


Figura 4.6 - Detalhamento do Módulo de Tratamento de Projetos

O módulo de Edição de Bolhas é detalhado na figura 4.7. Este módulo é composto por um módulo central, que é o de Controle da Edição de Bolhas, que executa o controle de todos os módulos relativos à edição de uma bolha. Desse modo, é feito o controle da criação de todos os tipos de bolhas permitidos no sistema, ou seja, bolhas OR, AND e Primitivas, acompanhado de seu respectivo posicionamento no Statechart. Através da seleção de uma bolha são acionados os demais módulos, ou seja, a movimentação, exclusão, alteração de tamanho e de nome e tipo de uma bolha, bem como a atribuição de parâmetros de bolhas primitivas. As funções executadas por esses módulo são aquelas descritas no item 4.2.3 acima apresentado. Todas essas operações atualizam o Banco de Dados do sistema.

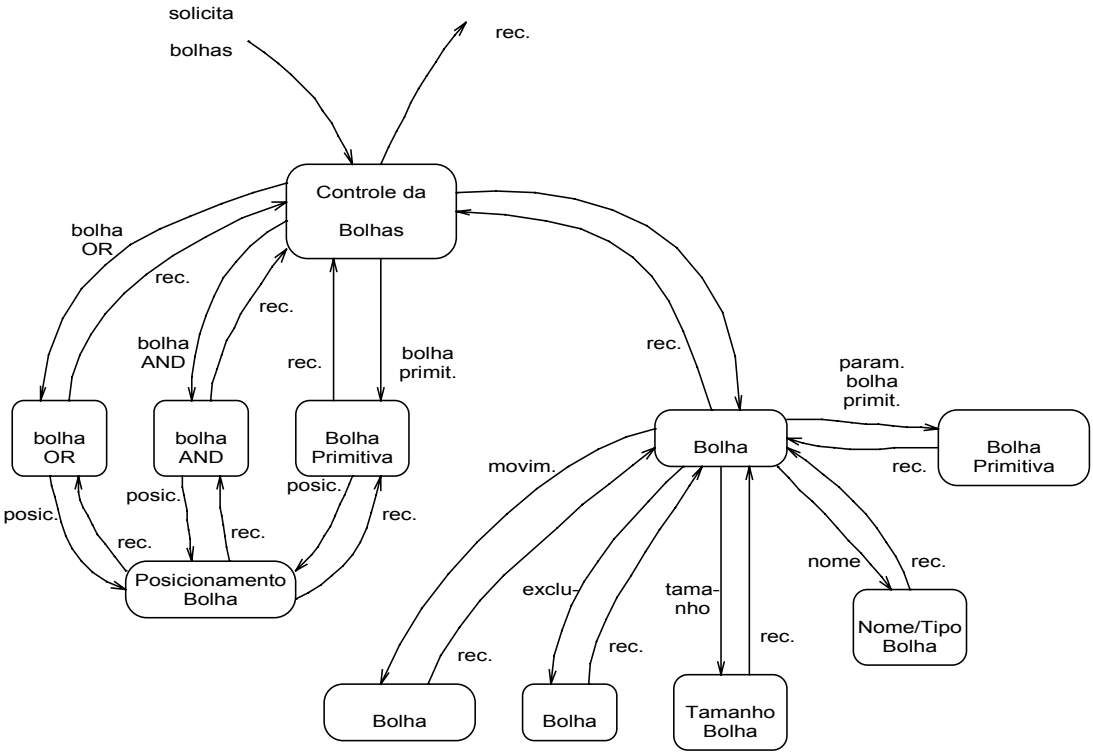


Figura 4.7 - Detalhamento do Módulo de Tratamento de Edição de Bolhas

O módulo de Edição de Ligações é detalhado na figura 4.8. Assim como no módulo de Edição de Bolhas, este módulo é composto por um módulo central, que é o de Controle da Edição de Ligações, que executa o controle de todos os módulos relativos à edição de uma ligação. Além das funções de edição de ligações, também é função deste módulo realizar a Seleção de Ação Adaptativa e de seus parâmetros. As funções executadas por esses módulo já foram apresentadas no item 4.2.4. Do mesmo modo, essas operações são acompanhadas das respectivas atualizações do Banco de Dados.

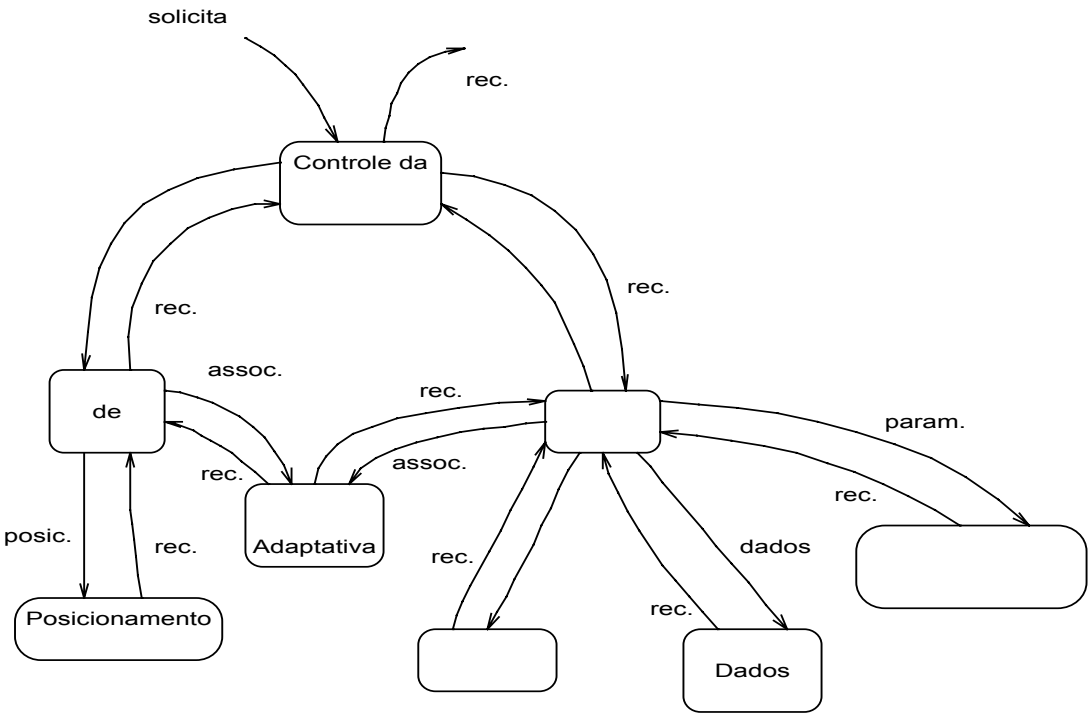


Figura 4.8 - Detalhamento do Módulo de Tratamento de Edição de Ligações

O módulo de Tratamento de Funções Auxiliares é detalhado na figura 4.9. Este módulo é composto pelos módulos que tratam as solicitações do usuário relativas a essas funções: Impressão de Tela, Edição de Banco de Dados, geração de Zoom, Árvore de Hierarquia e Lista de Inconsistências, funções estas já descritas no item 4.2.6.

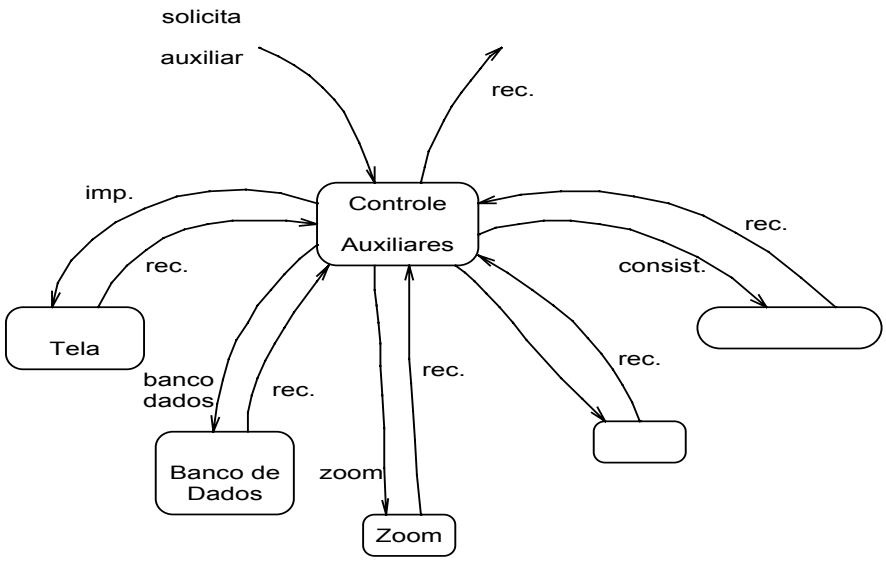


Figura 4.9 - Detalhamento do Módulo de Tratamento de Funções Auxiliares

O módulo de Tratamento de Funções Adaptativas é detalhado na figura 4.10. Este módulo é composto pelo Controle de Funções Adaptativas (que gerencia a Criação de Funções Adaptativas) e pela Edição de Elementos de Funções Adaptativas, que são detalhados nas figuras 4.11 e 4.12, respectivamente.

O módulo de Criação de Funções Adaptativas trata da criação, abertura, exclusão, alteração de nome e salvamento com outro nome de Funções Adaptativas, e efetua testes de consistência que impedem a criação de funções homônimas às já existentes. Estas atividades já foram descritas no item 4.2.7.

O módulo de Edição de Elementos de Funções Adaptativas gerencia a criação ou eliminação de elementos que compõem as Funções Adaptativas, conforme já descrito no item 4.2.7f.

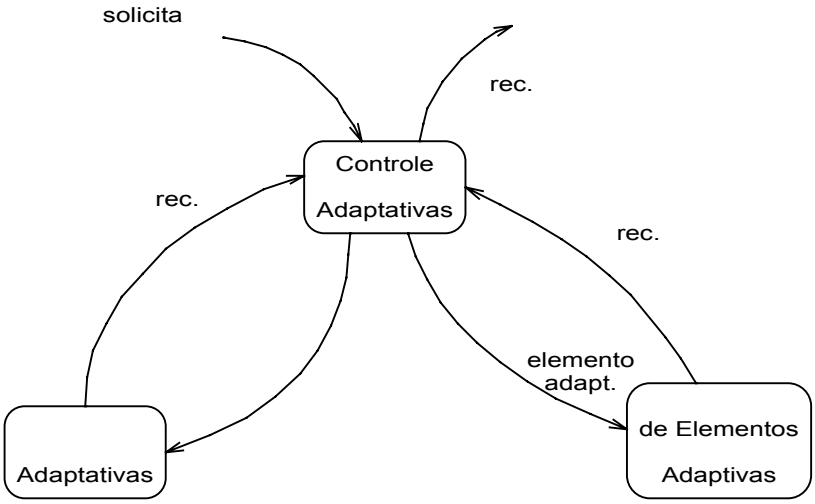


Figura 4.10 - Detalhamento do Módulo de Tratamento de Funções Adaptativas

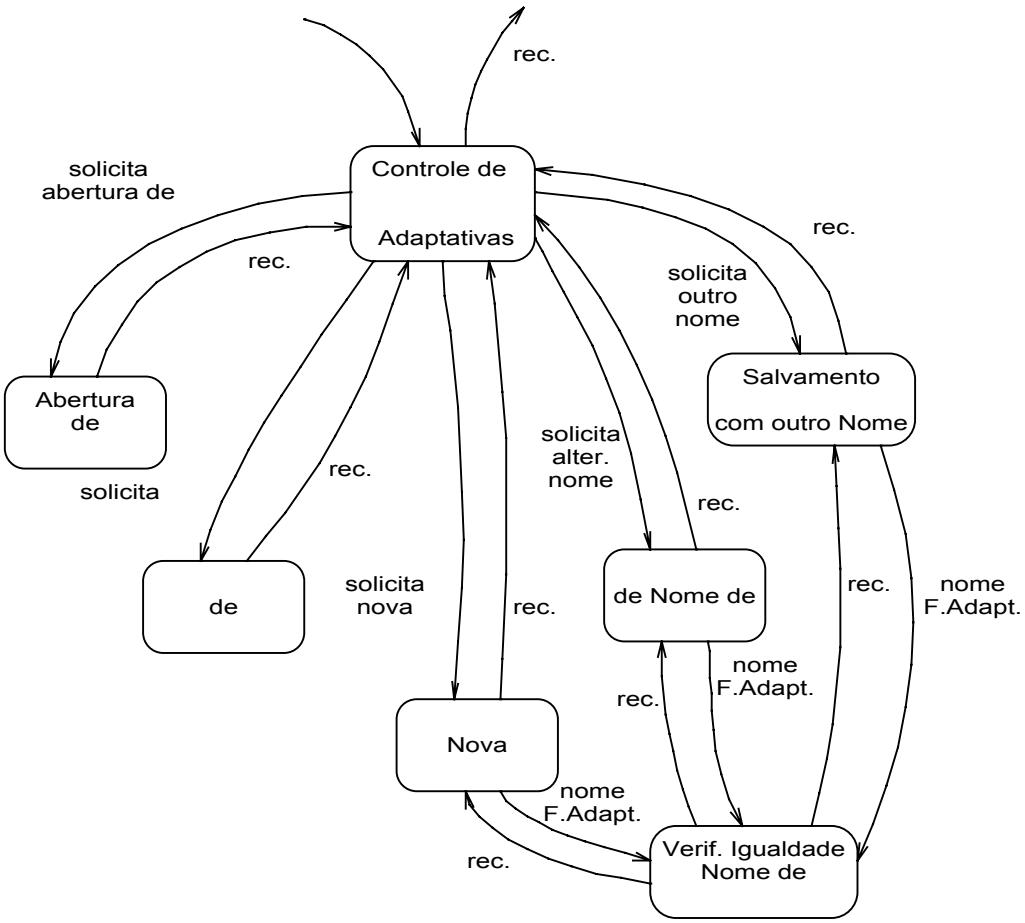


Figura 4.11 - Detalhamento do Módulo de Tratamento de Criação de Funções Adaptativas

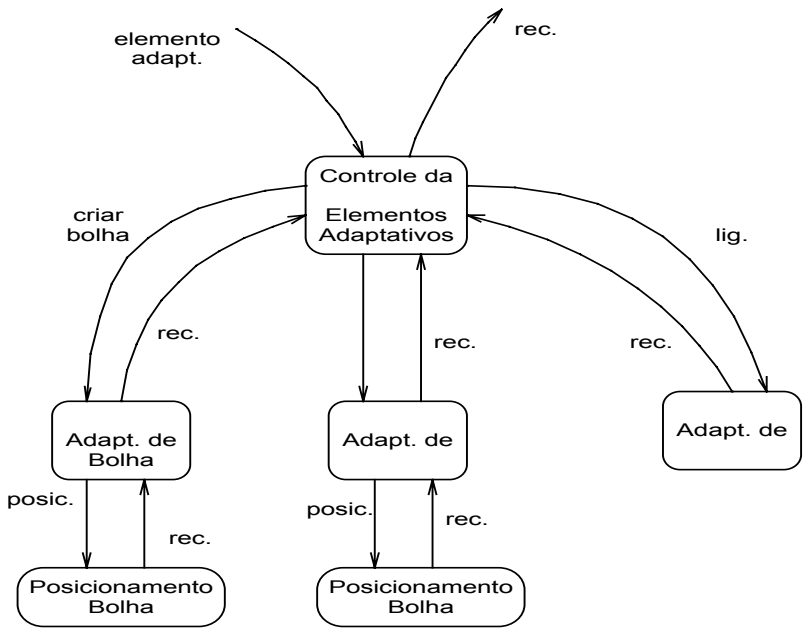


Figura 4.12 - Detalhamento do Módulo de Edição de Elementos de Funções Adaptativas

O módulo de Tratamento de Simulação é detalhado na figura 4.13. Este módulo é composto pelas funções de reinício do relógio do STAD (Zera Relógio), Determinação do Modo de Simulação (manual ou automático), Determinação do Nível de Detalhamento (se os detalhamentos devem ou não ser simulados), pela volta do Statechart às condições iniciais antes da simulação (Limpa) e pelo Passo de simulação, que é detalhado na figura 4.14.

Por essa figura nota-se que são inicialmente ativadas as bolhas do tipo "default" ou as bolhas descendentes de uma bolha do tipo "history". Com a execução de um passo de simulação, são ativadas as ligações que têm como origem essas bolhas. Outra forma de se ativar uma ligação é a ocorrência de um evento externo. A partir da ativação de uma ligação, e pela ocorrência de um novo passo de simulação, são ativadas as bolhas que têm como destino as ligações ativas. De uma bolha ativa é possível a ativação de outras ligações, através de um novo passo de simulação, ou então a execução da simulação dos níveis ascendente ou descendente de detalhamento da bolha, proporcionando assim uma forma estruturada de execução para os Statecharts.

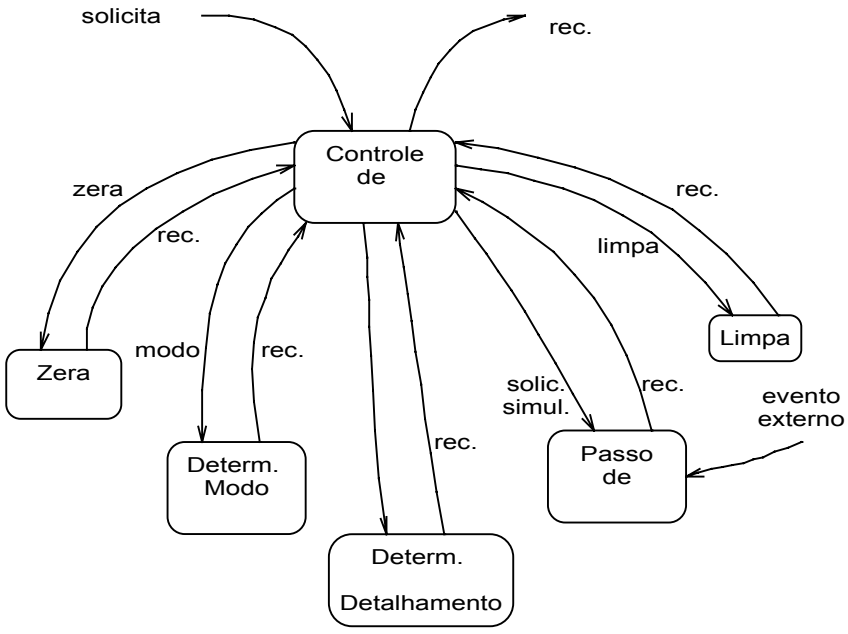


Figura 4.13 - Detalhamento do Módulo de Tratamento de Simulação

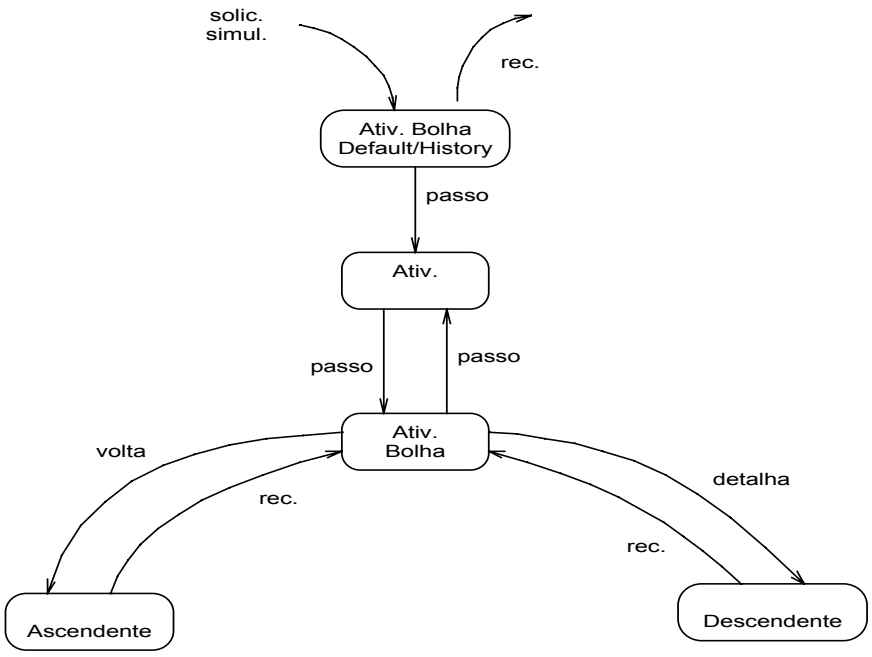


Figura 4.14 - Detalhamento do Módulo de Simulação de Statecharts

Quanto às informações relativas a cada Projeto, todas são armazenadas no Banco de Dados "tese1.mdb", conforme já citado. Este Banco de Dados é composto por quatro tabelas:

Tabela Bolhas: contém dados relativos às bolhas componentes de cada um dos projetos armazenados no sistema. Os dados armazenados de cada bolha são:

- Projeto do qual a Bolha faz parte;
- Nível de detalhamento;
- Tipo: OR (comum, "default", ou 'history"), AND ou Primitiva;
- Nome;
- Localização física da Bolha dentro do Statechart;
- Tamanho das bordas;
- Ascendente, no caso de se estar realizando um detalhamento;
- Nome da uma função ou ação adaptativa de que fizer parte, se for o caso; e
- Estado da simulação (se é ou não a bolha corrente durante uma simulação)

Tabela Ligações: contém dados relativos às ligações componentes de cada um dos projetos armazenados no sistema. Os dados armazenados de cada ligação são

- Projeto do qual a Ligação faz parte;
- Nível de detalhamento;
- Tipo: Dependente ou Independente, Interna ou Externa;
- Evento, Condição, Disparo, Atraso, Tempo de Disparo;
- Localização física da Ligação dentro do Statechart;
- Bolhas origem e destino;
- Ascendente, no caso de se estar realizando um detalhamento;
- Nome da uma função ou ação adaptativa de que fizer parte, se for o caso; e
- Estado da simulação (se é ou não a ligação corrente durante uma simulação)

Tabela Setas: contém dados relativos à representação das setas que compõem cada uma das ligações componentes de cada um dos projetos armazenados no sistema. Os dados armazenados de cada seta são:

- Projeto do qual a Seta faz parte;
- Nível de detalhamento;
- Localização física da seta dentro do Statechart;
- Ascendente, no caso de se estar realizando um detalhamento;
- Nome da uma função ou ação adaptativa de que fizer parte, se for o caso; e

Tabela Adaptativa: contém dados relativos à montagem das Funções Adaptativas e Ações Adaptativas componentes de cada um dos projetos armazenados no sistema. Os dados armazenados em cada registro são:

- Projeto do qual a Função ou Ação Adaptativa faz parte;
- Tipo de ação a ser realizada (inclusão ou exclusão de elemento);
- Tipo de elemento a ser incluído ou excluído (bolha ou ligação - só inclusão);
- Bolhas origem e destino, quando se tratar do caso de ligação.

4.4. Exemplos de Aplicação

Neste item são apresentados alguns exemplos de aplicação da ferramenta desenvolvida. O primeiro exemplo é o de uma pilha, cuja finalidade é a de demonstrar o funcionamento do STAD, principalmente no que se refere à utilização de funções adaptativas.

O segundo exemplo refere-se à representação de um relógio digital, que constitui um exemplo clássico da teoria de Statecharts, com o objetivo de demonstrar a capacidade de representação dos Statecharts pelo STAD sem, no entanto, se fazer uso de funções adaptativas.

No terceiro exemplo procura-se representar um sistema de controle de movimentação de trens em um trecho de um pátio ferroviário de manobras. Por meio deste exemplo procura-se apresentar uma situação prática, na qual se possa fazer utilização efetiva da característica adaptativa dos Statecharts Adaptativos oferecidos pelo STAD.

4.4.1. Pilha

Neste exemplo clássico é apresentado um statechart que simula o funcionamento de uma pilha. O statechart inicial é apresentado na figura 4.15. Nessa figura, as bolhas, representadas com um contorno contínuo, representam a situação do statechart antes da execução de qualquer ação adaptativa. A ação adaptativa selecionada é a que aparece nessa mesma figura, com as bolhas representadas com contornos tracejados.

A ligação associada à ação adaptativa é a representada pelo evento **##** que liga as bolhas **1** e **2**. As ações básicas executadas por esta Ação Adaptativa são as seguintes:

- Adição das bolha **C-1** e **D-1.**
- Adição das ligações **BETA-1**, **#-1**, **##-1** e **@-1**.
- Exclusão da ligação **#** ou **#-1.**

Após a execução de alguns ciclos de simulação deste statechart, obtém-se a configuração apresentada na figura 4.16.

Através deste exemplo foi ilustrada a capacidade do STAD de operar com Funções Adaptativas, verificando-se as ações de inclusão de bolhas e de ligações, bem como a exclusão de ligações.

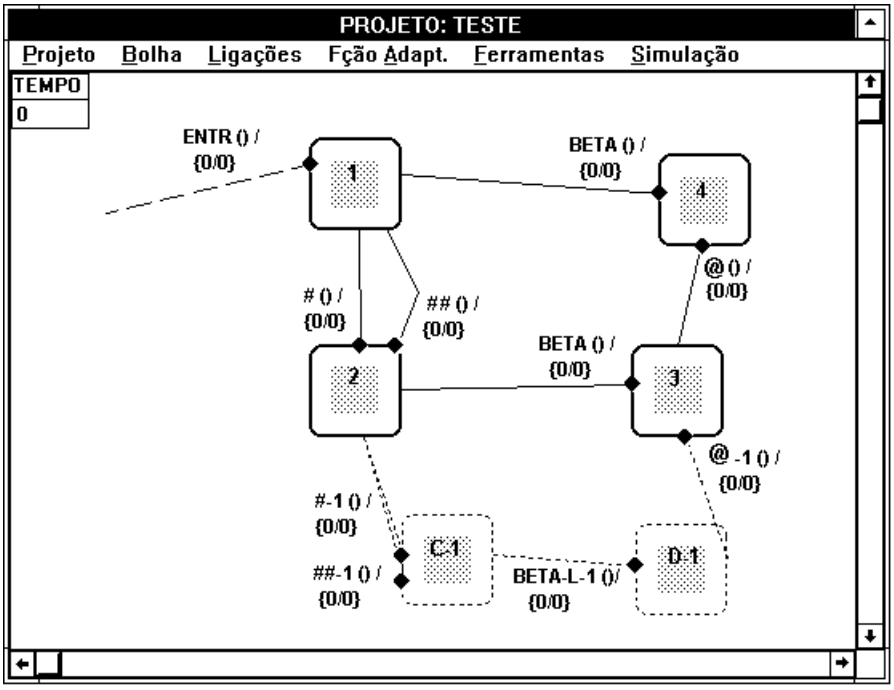


Figura 4.15 - Statechart Adaptativo do Projeto "TESTE" que simula uma Pilha

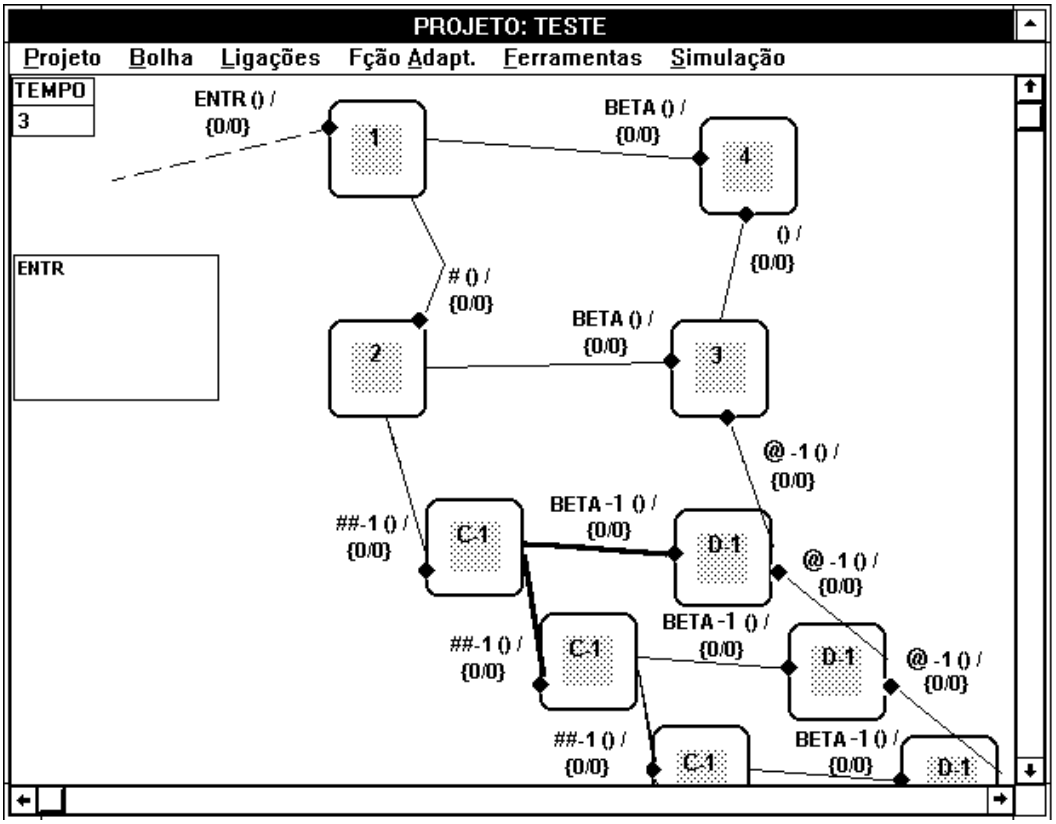


Figura 4.16 - Statechart Adaptativo do Projeto "TESTE" representando a Pilha após a execução de alguns ciclos de simulação

4.4.2. Relógio Digital

Neste outro exemplo clássico é apresentado um Statechart Adaptativo que modela um , relógio digital, apresentado diversas vezes na bibliografia (HAREL, 1987), (HAREL, 1988), (LUCENA, 1993) e (BATISTA NETO, 1991).

O relógio é composto por quatro botões, denominados de **A, B, C** e **D.** A ação de pressionar um deste botões causa a ocorrência dos eventos externos que disparam transições no statechart construído.

Na figura 4.17 é apresentada a configuração do nível 0 do projeto **"RELÓGIO"**, que é composto por duas bolhas, uma representando o relógio em funcionamento (**ALIVE**) e a outra, que é a bolha "default", o relógio parado (**DEAD**).

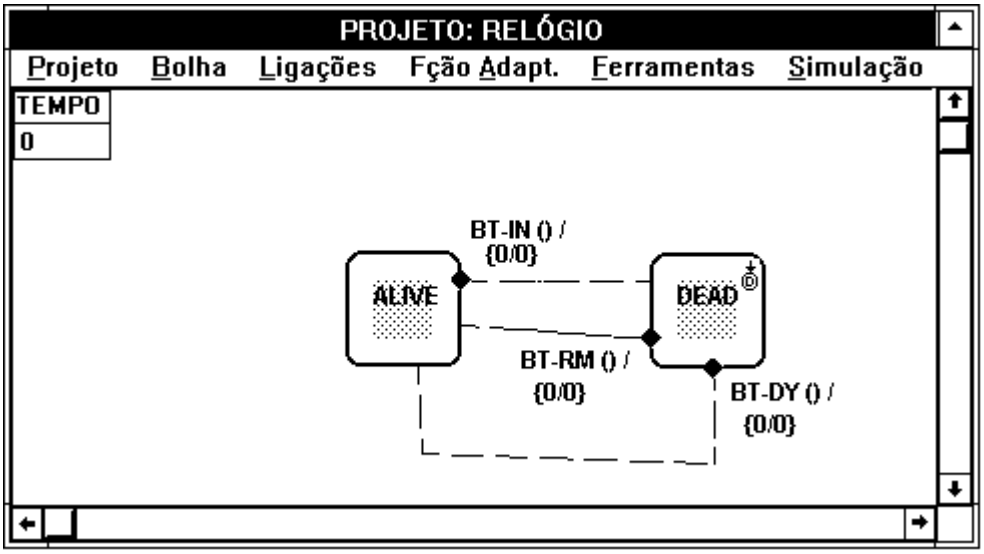


Figura 4.17 - Statechart do nível zero do Projeto "RELÓGIO"

Na figura 4.18. é mostrada a forma como é efetuado o detalhamento da bolha **ALIVE**, composta por cinco bolhas do tipo AND. Dessa forma, em qualquer instante, o relógio está operando em todas estas bolhas, indicando que todos esses estados são concorrentes, ou seja, todas as bolhas deste detalhamento estão ativas. A bolha **MAIN** indica a exibição das principais funções do relógio, tais como o acerto de hora, o acerto do momento do alarme e a ativação do cronômetro. A bolha **POWER** indica a presença ou não de tensão para o funcionamento do relógio. A bolha **ALARM-ST** indica o estado do alarme (habilitado ou desabilitado). A bolha **LIGHT** indica se a luz do relógio está ou não acesa. A bolha **CHIME-ST** indica o estado da função "chime" (toque de um beep de hora em hora) do relógio (habilitada ou desabilitada).

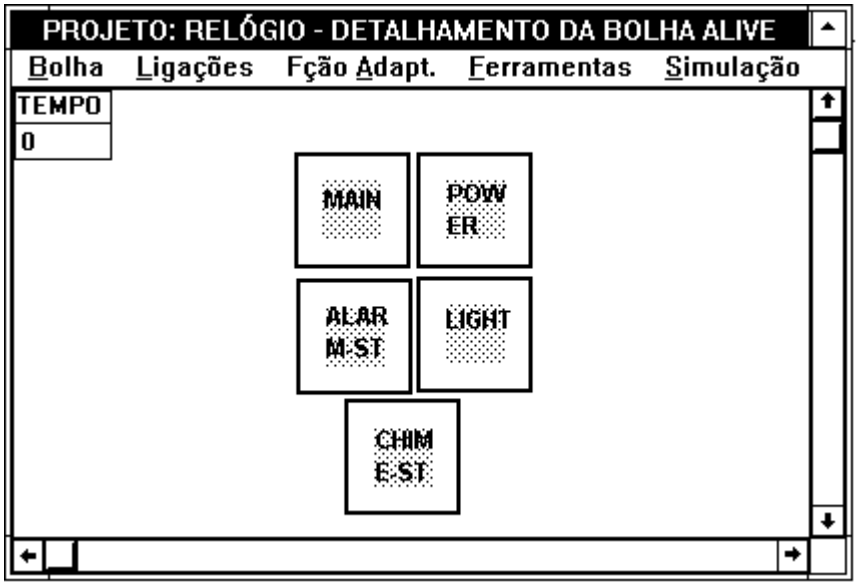


Figura 4.18 - Detalhamento da Bolha ALIVE do Projeto "RELÓGIO"

Na figura 4.19 é apresentado o detalhamento da bolha **MAIN**, que se decompõe nas bolhas **BEEP** e **DISPLAY**. A bolha **BEEP** é ativada na ocasião em que o horário indicado pelo relógio coincidir com o horário programado para toque do alarme
(**T-HITS-TM**), ao passo que a bolha **DISPLAY**, que é a bolha "default" neste detalhamento, é atingida sempre que algum botão for pressionado, ou então após transcorrido um intervalo de 30 segundos sem que seja pressionado nenhum botão..

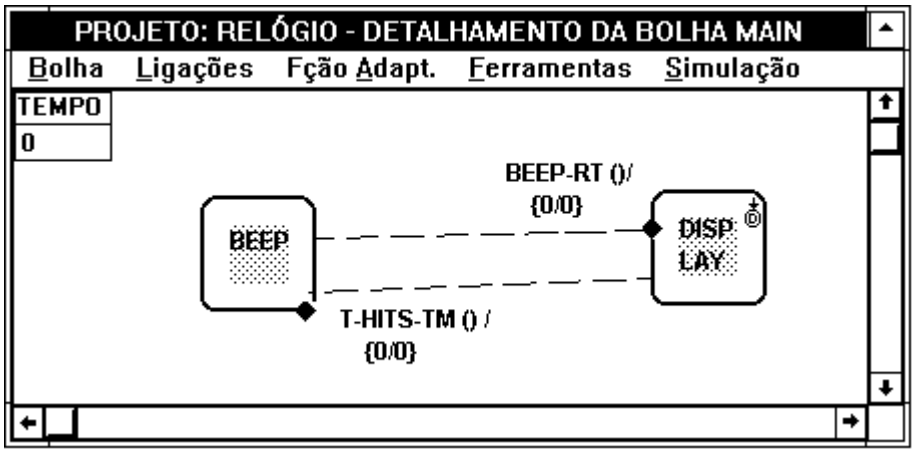


Figura 4.19 - Detalhamento da Bolha MAIN do Projeto "RELÓGIO"

Na figura 4.20 é apresentado o detalhamento da bolha **DISPLAY**. A transição entre as bolhas decorre dos eventos de serem pressionados os botões **A, B, C** e **D**, qualquer que seja a ordem do acionamento desse botões**.**

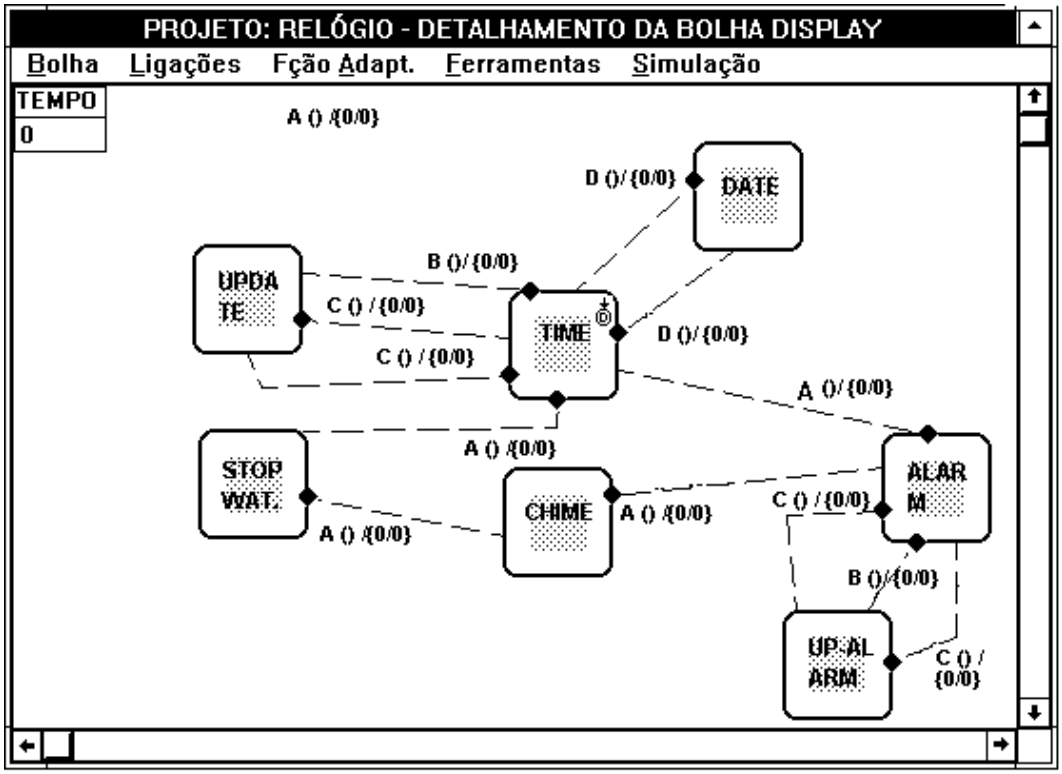


Figura 4.20 - Detalhamento da Bolha DISPLAY do Projeto “RELÓGIO”

O intuito da apresentação desses statecharts de detalhamento é apenas o de ressaltar os aspectos principais de sua utilização, através da ferramenta desenvolvida. Por esse motivo não são apresentados os demais statecharts que completam o detalhamento do projeto utilizado como exemplo, os quais são construídos de forma .análoga.

Este exemplo ilustrou a capacidade que o STAD possui de representar, por meio dos Statecharts, sistemas relativamente complexos, permitindo o detalhamento passo a passo de todas as funções, estabelecendo os níveis hierárquicos mais convenientes para cada sistema.

4.4.3. Simulação do Controle de Movimentação de Trens

Este exemplo procura demonstrar a capacidade do STAD em lidar com problemas de complexidade maior e mais realísticos. Corresponde à simulação do controle de movimentação de trens em um trecho de pátio de uma linha ferroviária.

Nos pátios ferroviários é comum a realização de manobras com os trens, com o objetivo de reunir ou separar os vagões que formam as composições, sendo que tais manobras são realizadas em velocidades muito baixas (até um máximo de 20 Km/h). Assim, as restrições de movimentação (intertravamento) que devem impor a abertura e fechamento dos sinais, bem como promover a movimentação dos aparelhos de mudança de via (máquinas de chave), e para a abertura e fechamento de sinais de acesso, é diferente do tipo de controle que deve existir para uma via onde os trens trafegam em altas velocidades, onde uma colisão entre trens ou um descarrilhamento teria conseqüências bastante danosas.

Outro aspecto a ser considerado é que, em um pátio de manobras ferroviário, são relativamente comuns as alterações de configuração da via, ou seja, enquanto alguns trechos de via podem vir a ser construídos, outros podem ser removidos, o mesmo ocorrendo com relação aos aparelhos de mudança de via e aos sinaleiros de acesso. Assim, este exemplo permite demonstrar a utilidade prática de modelagem através do uso dos Autômatos Adaptativos oferecidos pelo STAD, pois isto possibilita uma utilização efetiva das características adaptativas do formalismo implementado.

Um aparelho de mudança de via, como o nome indica, tem como função permitir a mudança de via de um trem, para efeitos de manobra ou por motivos operacionais. Diz-se que um aparelho de mudança de via está na posição normal, quando um trem, ao passar por esse aparelho, permanecer na mesma via em que se encontrava. De forma similar, um aparelho de mudança de via se diz na posição reversa, quando um trem, ao passar por esse aparelho, se transfere da via em que trafegava para outra via, ligada à via anterior por seu intermédio.

Diz-se que um determinado trecho está ocupado se houver um trem sobre o mesmo, e que tal trecho está desocupado se não houver trem sobre ele.

A nomenclatura utilizada no decorrer do exemplo é de **circuito de via** para os trechos em que o pátio se divide, **máquina de chave** para os aparelhos de

mudança de via e **sinal** para os sinaleiros de controle de acesso. Há ainda os **botões** que controlam a solicitação da abertura dos sinais.

Este exemplo é apresentado figura 4.21, cuja parte **a** apresenta a via em sua configuração inicial. Na parte **b** apresenta-se a via em sua configuração final, após a adição de alguns elementos.

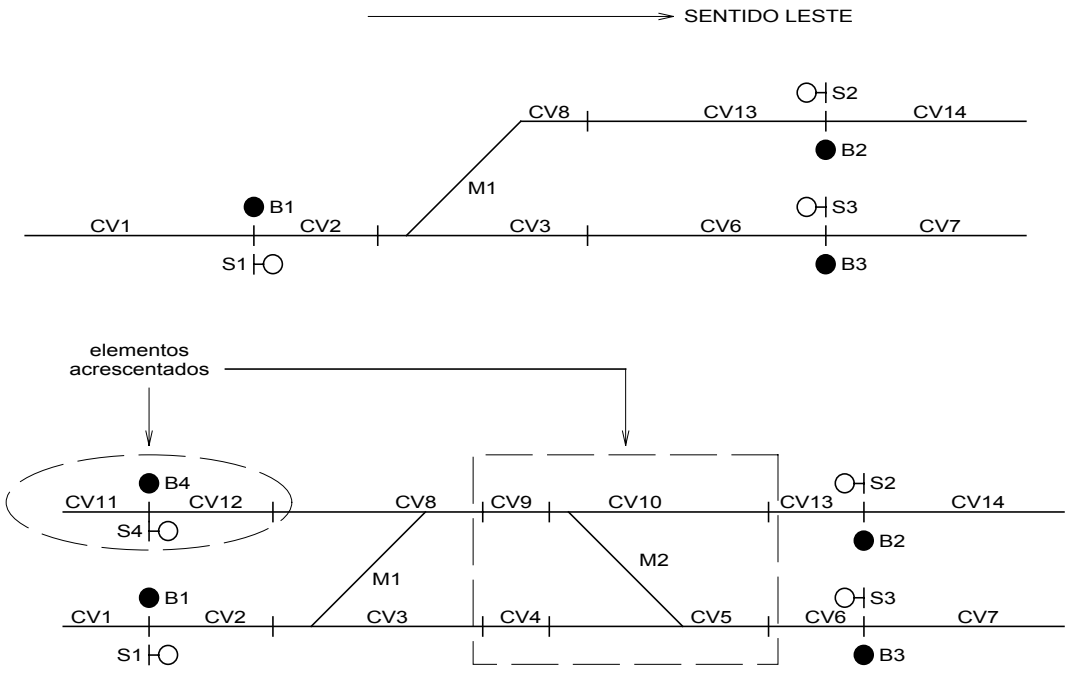


Figura 4.21 - Configuração de um trecho do Pátio de Manobras

Pela figura 4.21a, nota-se que a via em sua configuração inicial é composta pelos circuitos de via **CV1, CV2, CV3, CV6, CV7, CV8, CV13** e **CV14**, pelos sinais **S1, S2** e **S3**, pela máquina de chave **M1** e pelo botões **B1, B2** e **B3**. A configuração final, vista na figura 4.21b, é a mesma da figura 4.21a acrescida dos circuitos de via **CV4, CV5, CV9, CV10, CV11** e **CV12**, do sinal **S4,** da máquina de chave **M2** e do botão **B4**.

Os sinais têm como função proteger a entrada de trens no circuito de via imediatamente posterior a cada sinal, respeitando-se o sentido de movimentação do trem. Portanto, no exemplo da figura 4.21, o sinal **S1** protege a entrada de um trem no circuito de via **CV2**, ou seja, enquanto o sinal **S1** estiver fechado (ou com o aspecto vermelho aceso), não é permitida a ocupação de **CV2**, que só é liberada quando **S1** estiver aberto (ou com o aspecto verde aceso).

Para se alterar o posicionamento de uma máquina de chave basta que se gere um comando para que a sua posição seja alterada.

Para se abrir um sinal que libere a passagem de um trem, deve-se estabelecer uma rota, ou seja, determinar um botão ou sinal de origem e um botão ou sinal de destino da rota. No exemplo da figura 4.21, uma rota com origem em **B1** e destino em **B3** é denominada rota **B1-B3**.

Como o intuito de um pátio de manobras é o de permitir que se realizem junções e separações de vagões, conforme já citado, para se estabelecer uma rota verifica-se apenas o posicionamento das máquinas de chave. Dessa forma, no exemplo da figura 4.21, para se armar a rota **B1-B2**, é necessário pressionar, inicialmente o botão **B1**, que é a origem de rota, e em seguida o botão **B2**, que é o seu destino. Para que a rota possa ser completada, é necessário apenas que a máquina de chave **M1** esteja posicionada em reverso e a máquina **M2**, em normal. Não são verificadas eventuais ocupações nos circuitos de via, pelo motivo acima mencionado.

Para que um sinal, aberto pela solicitação de abertura de uma rota, seja fechado, basta pressionar novamente o botão correspondente à origem dessa rota. A ocupação dos circuitos de via que compõem a rota não causa o fechamento do sinal.

Este sistema foi modelado por intermédio dos Statecharts Adaptativos do STAD. Por motivos gráficos de adequação do tamanho das figuras ao presente texto, as figuras aqui apresentadas não são aquelas obtidas diretamente pala utilização da ferramenta.

Na figura 4.22 é apresentado o modelo utilizado, na forma de grandes blocos. Para se iniciar a simulação do sistema é necessário gerar o evento **inicia sistema,** e a partir da bolha **INÍCIO**, acionar o evento **solicita principal** ou **solicita manutenção**, conforme se queira entrar no **MÓDULO PRINCIPAL** ou **MÓDULO MANUTENÇÃO**. O **MÓDULO PRINCIPAL** é detalhado na figura 4.23, sendo composto pelos módulos de **Situação dos Circuitos de Via**, **Posicionamento das Máquinas de Chave** e **Aspecto dos Sinais**.

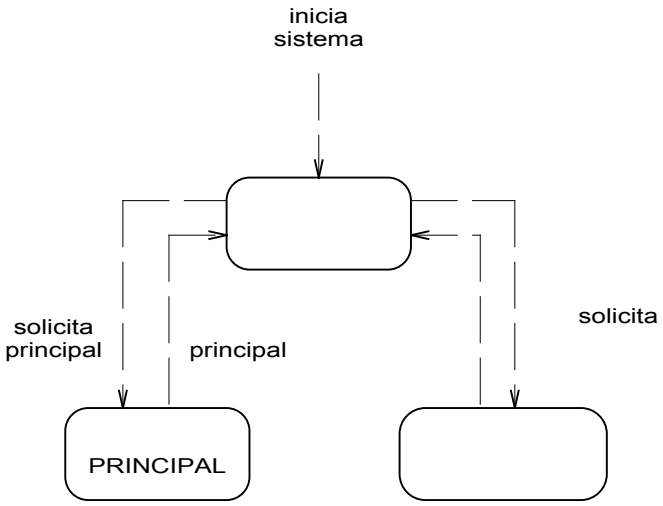


Figura 4.22 - Blocos Principais constituintes do Sistema de Controle de Tráfego de Trens

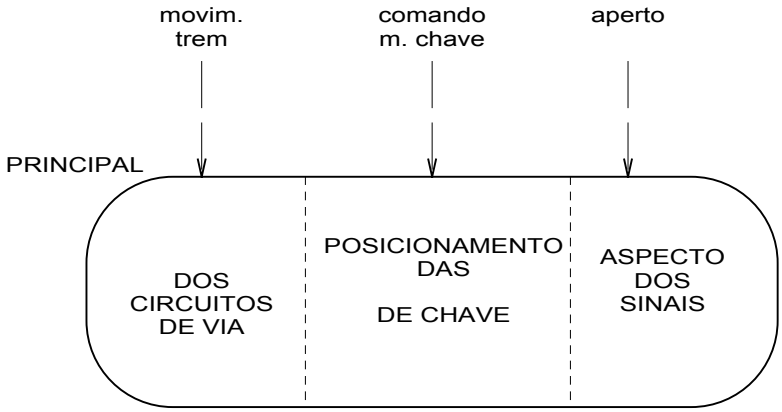


Figura 4.23 - Detalhamento do Módulo Principal do Sistema de Controle de Tráfego de Trens

A bolha de **Situação dos Circuitos de Via** é a responsável pela representação da movimentação dos trens, alterando-se a ocupação dos circuitos de via e verificando-se o aspecto dos sinais e o posicionamento das máquinas de chave. Na figura 4.24 é apresentada a configuração do Statechart, considerando-se a via da figura 4.21a.

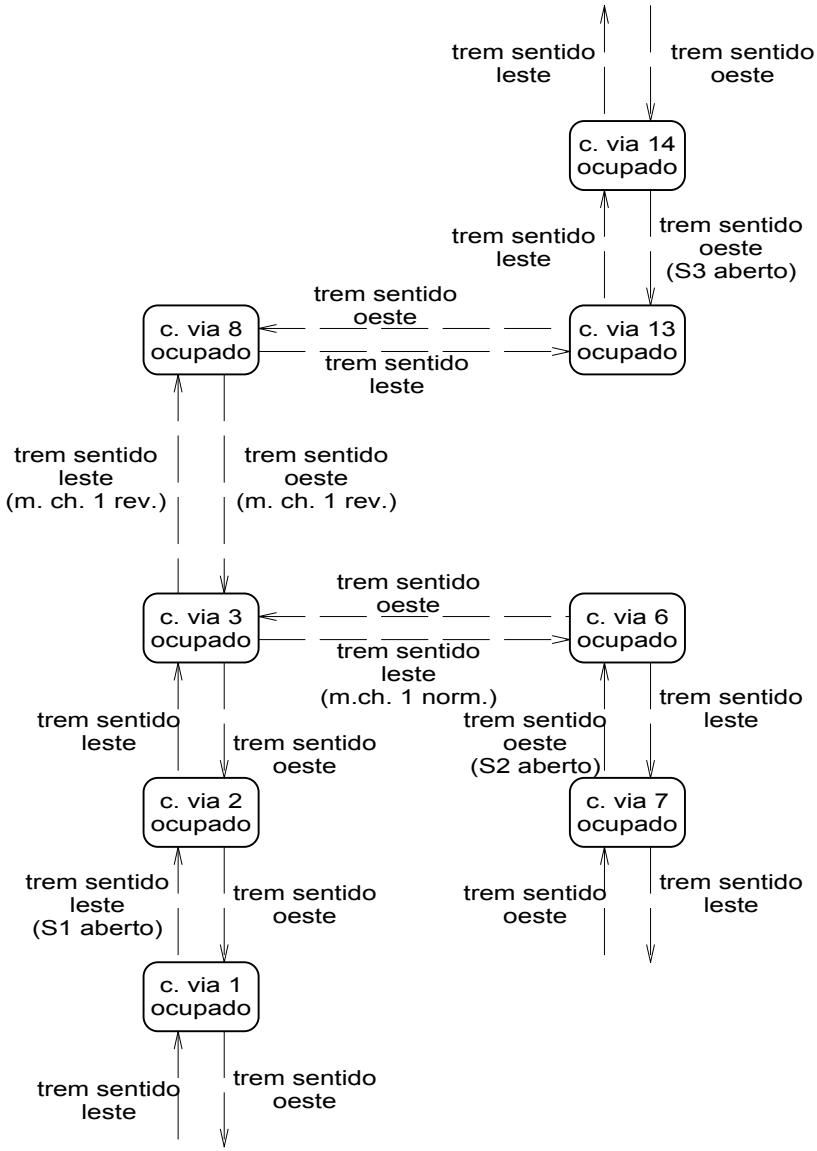


Figura 4.24 - Detalhamento da Bolha de Situação dos Circuitos de Via - Situação Inicial

A bolha de **Posicionamento das Máquinas de Chave** é a responsável pela representação da posição das máquinas de chave, à medida que forem sendo gerados os comandos para o seu posicionamento. Nas figura 4.25 é apresentada a configuração do Statechart, considerando-se a via da figura 4.21a.

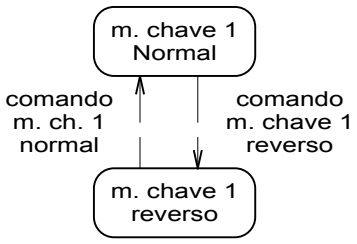


Figura 4.25 - Detalhamento da Bolha de Posicionamento de Máquinas de Chave - Situação Inicial

A bolha de **Aspecto dos Sinais** é a responsável pela representação do aspecto dos sinais e pela solicitação de abertura de rotas, considerando-se os eventos de aperto de botões. Na figura 4.26 é apresentada a configuração do Statechart, considerando-se a via da figura 4.21a.

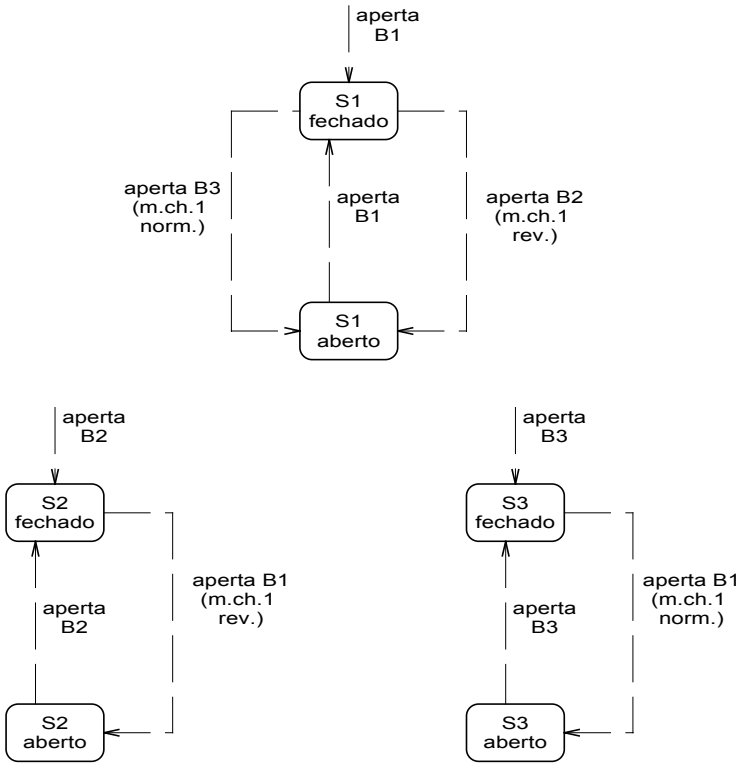


Figura 4.26 - Detalhamento da Bolha Aspecto dos Sinais - Situação Final

Na figura 4.27 são apresentadas as funções adaptativas utilizadas para a representação das alterações ocorridas na configuração da via. Na figura 4.27a é apresentada a Função **F1**, que consiste de uma bolha que representa a criação de um novo circuito e duas ligações que representam a possibilidade de movimentação de trens. Na figura 4.27b apresenta-se a Função **F2**, que também consiste de uma bolha que representa a criação de um novo circuito e quatro ligações que representam a possibilidade de movimentação de trens. Este é o caso de um circuito de via que contenha uma máquina de chave.

Na figura 4.27c são apresentadas duas bolhas que simbolizam a criação de uma nova máquina de chave, com as posições normal e reverso e os comando para posicionamento em normal e reverso. Esta é a Função **F3**. Na figura 4.27d é apresentada a função que cria um novo sinal e um novo botão, representado a Função **F4**, enquanto que na figura 4.27e representa-se a Função **F5**, composta por duas bolhas que simbolizam a possibilidade de se armar uma rota, com as ligações associadas ao aperto de um botão e à verificação do posicionamento de duas máquinas de chave. Finalmente, na figura 4.27f representa-se a Função **F6**, que se constitui na eliminação de até 4 ligações.

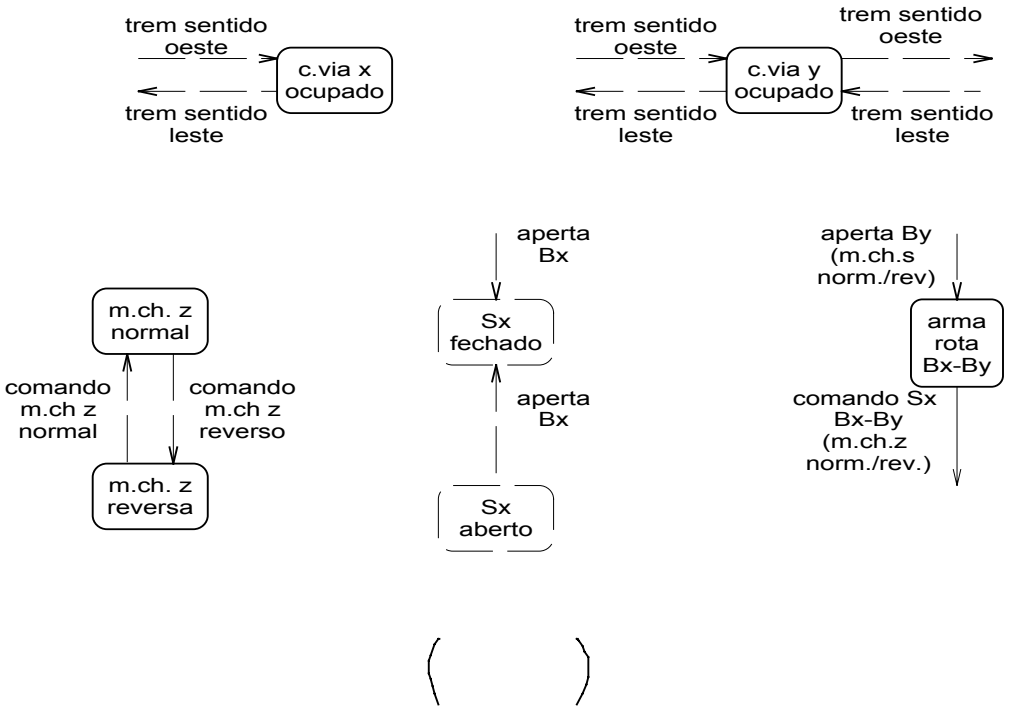
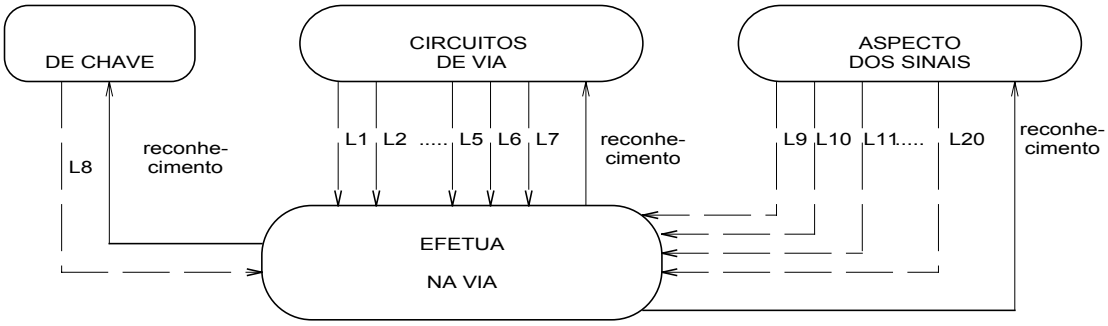


Figura 4.27- Funções Adaptativas utilizadas no Sistema de Simulação e Controle de Trens

Na figura 4.28 é detalhado o **MÓDULO MANUTENÇÃO**, que é composto pelas ligações associadas às Ações Adaptativas que efetuam a alteração do Statechart original, correspondente à via da figura 4.21a, para a via da figura 4.21b. A ativação de cada uma das ligações habilita a execução da Ação Adaptativa associada, e por conseqüência, da alteração da configuração da via do pátio de manobras.



"trem sentido oeste" (entre as bolhas "c.via 8 ocupado" e "c.via 13 ocupado") e

as bolhas "c.via 3 ocupado" e "c.via 6 ocupado")

"aperta B2 (m. ch. 1 rev.)", "aperta B1 (m.ch. 1 rev.)" e

Figura 4.28- Detalhamento do Módulo Manutenção do Sistema de Simulação e Controle de Trens

Desta forma, após a execução de todas a Ações Adaptativas previstas na figura 4.28, os Statecharts correspondentes ao detalhamento das Bolhas **Situação dos Circuitos de Via**, **Posicionamento das Máquinas de Chave** e **Aspecto dos Sinais**, assumem a configuração das figuras 4.29, 4.30 e 4.31, respectivamente.

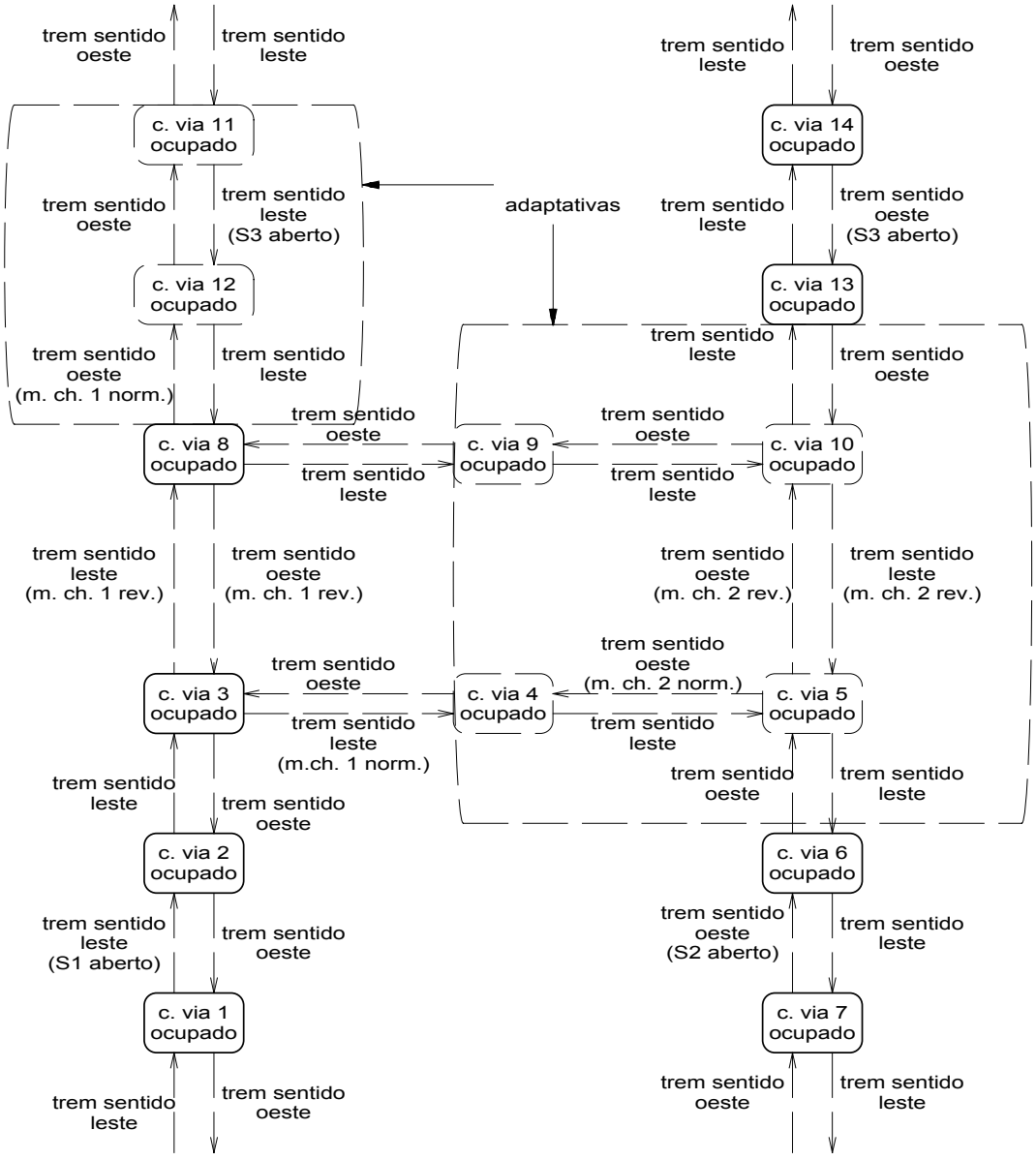


Figura 4.29 - Detalhamento da Bolha de Situação Final dos Circuitos de Via - Situação Final

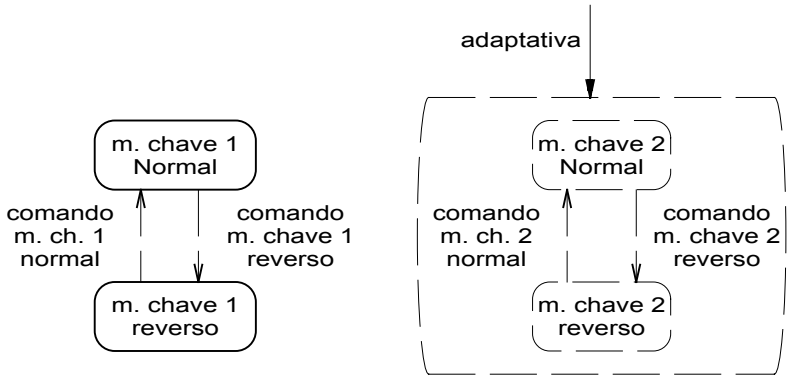


Figura 4.30 - Detalhamento da Bolha de Posicionamento de Máquinas de Chave - Situação Final

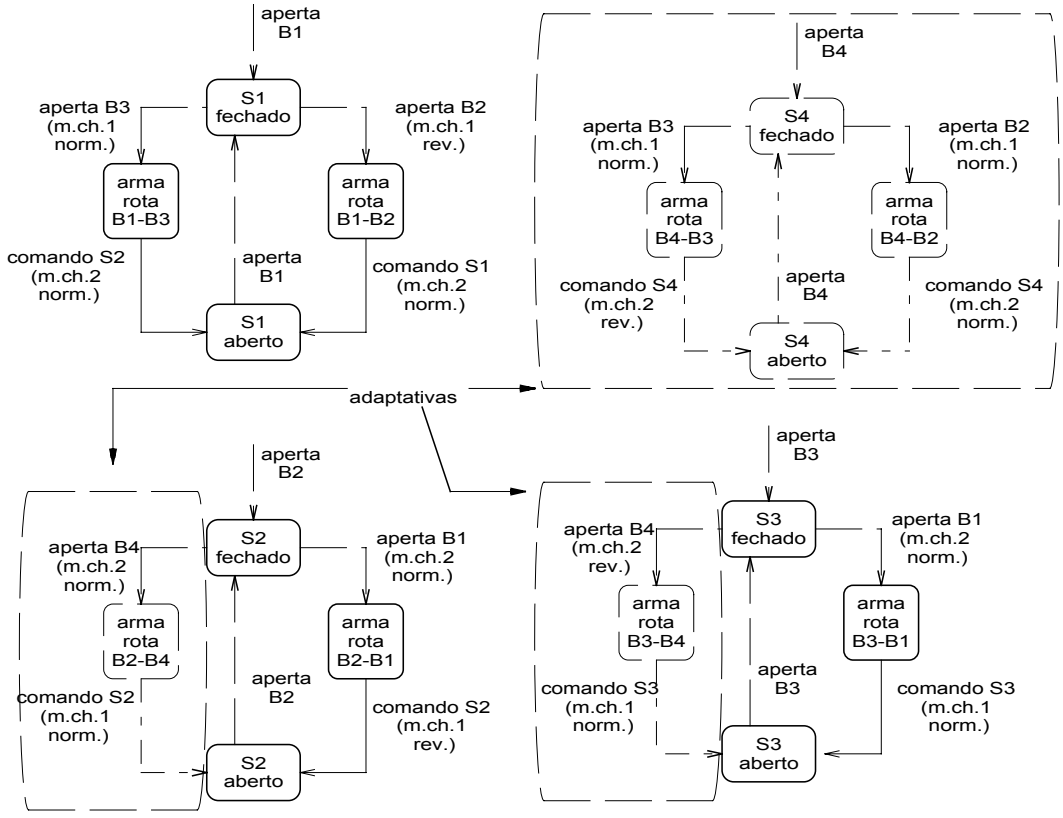


Figura 4.31 - Detalhamento da Bolha Aspecto dos Sinais - Situação Final

Este exemplo de aplicação foi submetido ao STAD, obtendo-se os resultados esperados, permitindo a representação eficiente e o estudo detalhado de sistemas que possuam características que possam aproveitar o potencial de utilização das funções adaptativas.

5. CONSIDERAÇÕES FINAIS

Neste capítulo são comentados os aspectos relevantes deste trabalho, levantando-se suas potencialidades, sua aplicabilidade e as perspectivas para prosseguimento da pesquisa, destacando-se os aspectos de maior interesse, e as principais contribuições do trabalho.

5.1. Contribuições do Trabalho

Os Statecharts têm potencial para se integrar às linguagens e metodologias visuais, possuindo elementos de estrutura apropriada para explorar as vantagens da interação gráfica. Dessa forma, os problemas técnicos e científicos poderão ser solucionados através de ferramentas gráfico-visuais, melhores e de menor custo de desenvolvimento que as ferramentas anteriormente utilizadas.

A característica adaptativa da ferramenta permite que se representem sistemas nos quais seja mais efetiva a utilização de uma notação que suporte o emprego de funções adaptativas. Através da utilização de elementos criados ou excluídos por intermédio de ações adaptativas, possibilita-se a representação de sistemas cuja configuração seja, ou possa ser alterada dinamicamente.

Desse modo, os Statecharts Adaptativos constituem uma ferramenta teórica, implementada de forma prática pelo STAD, de grande valia e utilidade na representação dessa característica dinâmica de alguns sistemas, .

Os principais itens estudados e pesquisados foram a teoria existente, tanto para os Statecharts convencionais, quanto para os Autômatos Adaptativos. Do estudo desses dois aspectos pode-se concluir pela viabilidade da união das características dessas duas formas de representação, resultando nos Statecharts Adaptativos.

Possibilitou-se ainda a representação de eventos concorrentes, uma das principais vantagens da utilização dos Statecharts, em adição à sua característica adaptativa

Por permitirem uma representação hierárquica, concisa e completa, os Statecharts Adaptativos superam uma dificuldade apresentada por muitas representações, as quais geram diagramas muito grandes e densos, dificultando a compreensão por parte de analistas, projetistas e usuários.

A forma adotada na ferramenta, que permite a divisão do detalhamento em janelas separadas, também se revelou bastante apropriada, possibilitando uma separação maior das funções do sistema que estiver sendo desenvolvido, embora isto dificulte um pouco uma visão mais global do projeto.

A ferramenta desenvolvida revelou-se um editor bastante cômodo e de utilização relativamente simples, graças à sua interface homem-máquina moderna e amigável. As funções relativas à edição de bolhas e ligações, tais como criação, exclusão e movimentação foram de implementação bastante trabalhosa, sendo que o resultado final obtido foi plenamente satisfatório. A maioria das ações a serem executadas pelo usuário é feita quase que exclusivamente por intermédio do acionamento do mouse, o que facilita sobremaneira a edição dos statecharts.

A possibilidade adicional, oferecida pela ferramenta, de simulação da execução dos statecharts, facilita em muito ao usuário a compreensão do sistema em estudo, permitindo que se façam simulações em alto nível, sobre o modelo de tal sistema, possibilitando assim, a realização de correções já nas primeiras fases do seu ciclo de desenvolvimento.

As funções auxiliares incorporadas à ferramenta, tais como as responsáveis pela possibilidade de visualização da árvore de hierarquia do projeto e pelo zoom das bolhas, também se mostraram muito práticas, por permitirem um melhor acompanhamento da evolução da edição do sistema.

Outro aspecto a ser ressaltado é a possibilidade do acesso ao banco de dados que contém os registros com as informações dos elementos do sistema, permitindo não só o acompanhamento da evolução do sistema, mas também eventuais correções que se deseje efetuar diretamente nos registros. Este aspecto ressalta o lado didático da ferramenta. Embora esta característica tenha sido um importante aliado na fase de depuração do STAD, poderá vir a constituir um problema para um usuário convencional, pois alguma alteração indevida em registros do Banco de Dados pode comprometer seus resultados. Por isto, na versão definitiva tal função será restrita a usuários privilegiados apenas.

A criação e edição das funções adaptativas, por utilizarem as mesmas rotinas de edição dos elementos não-adaptativos, também se revelaram relativamente simples de serem manipuladas. Com a possibilidade oferecida de atribuição de parâmetros aos argumentos das ações adaptativas associadas às ligações, permite-se

que sejam representados os mais diversos tipos de ações necessárias em sistemas que tenham a característica adaptativa. As ações adaptativas associadas às ligações possibilitam, quando da simulação do Statechart, a visualização, passo a passo, da evolução estrutural do sistema sob análise.

Um dos aspectos mais importantes a se ressaltar é a possibilidade de se representar aspectos temporais, permitindo o estudo de sistema de tempo real, de uma forma bastante prática.

5.2. Conclusões

Os Statecharts constituem uma excelente ferramenta gráfica para a representação de sistemas, devido à sua característica de reunir e de englobar um maior número de informações em um desenho não muito complexo, que permite uma leitura fluente e de fácil compreensão.

No que tange à característica adaptativa incorporada aos Statecharts, pode-se afirmar que contribuiu sobremaneira para enriquecer e completar as potencialidades dos Statecharts convencionais.

As Funções Adaptativas e suas instâncias, as Ações Adaptativas, são um meio excelente para se representar características do comportamento dinâmico de sistemas, permitindo que se estude com muito mais precisão e se formalize com rigor sistemas que possuam características que variem conforme a sua história, como é o caso, por exemplo, de sistemas que utilizem a inteligência artificial como ferramenta de desenvolvimento.

A ferramenta desenvolvida tem grande potencial didático e pedagógico, pois ao se realizar um estudo tanto da teoria de Statecharts, quanto da teoria de Autômatos Adaptativos, oferece uma ferramenta disponível e completa que permite visualizar com facilidade todos os pontos da teoria.

Dessa forma, o aprendizado torna-se bem mais facilitado, pois permite um acompanhamento prático de uma parte teórica nem sempre muito simples de ser compreendida.

Como prosseguimento do trabalho podem ser citados alguns aspectos a serem incorporados à ferramenta, a saber:

- a criação de um auxílio “on-line”;

- a geração de um aviso ou a interrupção da simulação de um Statechart quando de tentativa da ativação de mais de uma bolha OR em um determinado detalhamento;
- uma execução real (criação de código executável) do sistema representado, em lugar da simulação atualmente efetuada;
- a inclusão da função de consulta, presente na teoria de autômatos adaptativos, mas não incorporada nesta versão da ferramenta.
- a possibilidade de se salvar uma configuração de uma simulação de um sistema como uma nova configuração original, eventualmente com outro nome de projeto.
- a inclusão da probabilidade nos eventos externos, permitindo que o estudo de um sistema seja feito de forma mais automatizada, com a própria ferramenta gerando os eventos, de forma a respeitar as probabilidades introduzidas.

- a inclusão de sincronização e comunicação entre Statecharts.

- a possibilidade de alteração dinâmica do aspecto temporal.

Outra evolução a ser feita neste trabalho é a continuação do estudo desta e de outras técnicas a serem adotadas para a representação de sistemas, bem como a dos próprios Statecharts, e da teoria de autômatos adaptativos.

Concluindo-se, pode-se afirmar que a construção do STAD demonstrou, de forma prática, a possibilidade da utilização da representação de sistemas por meio de Statecharts, aliado à teoria de autômatos adaptativos.

A concepção dos Statecharts adaptativos, fundamentada na inclusão da propriedade adaptativa aos Statecharts constitui o aspecto de maior relevância deste trabalho, pois representou um avanço conceitual em relação aos statecharts convencionais. Com os statecharts adaptativos, possibilita-se a representação de sistemas com a capacidade de, com base na experiência adquirida durante o seu processamento, aprender e reter este conhecimento, através de uma auto-modificação topológica comandada pelos estímulos de entrada recebidos.

REFERÊNCIAS BIBLIOGRÁFICAS

ACKERMAN, A.F.; BUCHWALD, L.S.; LEWSKI, F. H. Software inspections: an effective verification process. **IEEE Software**, v.6, n.3, p.31-6, May 1989.

AGERWALA, T. Putting Petri nets to work. **Computer**, v.12, n.12, p.85-94, Dec. 1979.

BATISTA NETO, J.E.S. **Um editor gráfico para statecharts.** São Carlos, 1991, 118p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.

BERSOFF, E.H.; DAVIS, A.M.. Impacts of life cycle models on software. **Communications of the ACM**, v. 34, n.8, p.104-17, Aug. 1991.

COAD, P.; YOURDON, E. **Análise baseada em objetos.** Rio de Janeiro, Editora Campus, 1992.

CRAIGEN, D.; GERHART, S.; RALSTON T. Case study: darlington nuclear generating station, Paris metro signaling system, Traffic alert and collision avoidance system, Multinet gateway system. **IEEE Software**, v.12, n.1, p.30-9, Jan. 1994.

CURTIS, B.; KELLNER, M. I.; OVER, J. Process modeling. **Communications of the ACM**, v.35, n.9, p.75-90, Sept. 1992.

DAVIS, A.M. A comparision of techniques for the specification of external system behavior. **Communications of the ACM,** v.31, n.9**,** p.1098-115, Sept. 1988.

DIAZ, R.P.. Implementation faceted classification for software reuse. **Communications of the ACM**, v.34, n.5, p.88-97, May 1991.

DUKE, E.L. V&V on flight and mission-critical software. **IEEE Software**, v.6, n.3, p.39-45, May 1989.

DUNHAM, J.R. V&V in the next decade. **IEEE Software**, v.6, n.3, p.47-53, May 1989.

ELIAS, V.G.S.; LIESENBERG, H. **Debugging aids for statechart-based systems.** Campinas, Departamento de Ciência da Computação da UNICAMP, 1992. p.1-11 (Relatório Técnico DCC-00/92).

FISCHER, G. Human-computer interaction software: lessons learned, challenges ahead. **IEEE Software**, v.6, n.1, p. 44-52, Jan. 1989.

FUCHS, N.E. Specifications are (preferably) executable. **Software Engineering Journal**, v.7, n.9, p. 323-34, Sept. 1992.

GABRIELIAN, A.; FRANKLIN, M.K. Multilevel specification of real-time systems. **Communications of the ACM**, v.34, n.5, p. 50-60, May 1991.

GARLAN, D.; ILIAS, E. Low-cost, adaptable tool integration policies for integrated environments. **Communnications of ACM**, v.33, n.12, p.1-10, Dec. 1990.

GERHART, S.L. Applications of formal methods: developing virtuoso software. **IEEE Software**, v.7, n.5, p.7-10, Sept. 1990.

GERHART, S.; CRAIGEN, D.; RALSTON, T. Experience with formal methods in critical systems. **IEEE Software**, v.11, n.1, p.21-28, Jan. 1994

GENERAL ELECTRIC CORPORATION. **Software engineering handbook.** New York**,** 1986.

GIBBS, W.W. Software's chronic crisis. **Scientific American.** v.271, n.3, p. 72-81, Sept. 1994.

GIRGIS, M.R. An experimental evaluation of a symbolic execution system. **Software Engineering Journal**, v.7, n.7, p.285-90, July l992.

GOMAA, H. Software development of real time systems. **Communicatios of the ACM**, v.29, n.7, p.657-68, July 1986.

HAL, A. Seven myths of formal methods. **IEEE Software**, v.7, n.5, p.11-9, Sept. 1990.

HALLANG, W.A.; KRÄMER, B.J. Safety assurance in process control. **IEEE Software**, v.11, n.1, p.61-7, Jan. 1994.

HARBERT, A.; LIVELY, W.; SHEPPARD, S. A graphical specification system for user-interface design. **IEEE Software**, v.7, n.4, p.12-20, July 1990.

HAREL, D.; PNUELI, A.; SCHMIDT, J.P.; SHERMAN, R. On the formal semantics of statecharts. In: SYMPOSYUM ON LOGIC IN COMPUTER SCIENCE, 2°, Ithaca, NY, 1987. **Proceedings.** New York, IEEE Press, 1987a, p.54-64.

HAREL, D. Statecharts: a visual formalism for complex systems. **Science of Computer Programming**, v.8, n.3, p.231- 74, Aug. 1987b.

HAREL, D. On visual formalisms. **Communications of the ACM**, v.31, n.5, p.11-9, May 1988a.

HAREL, D.; et al. Statemate: A working environment for the development of complex reactive systems. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 10°, Singapore, 1988. **Proceedings.** Los Alamitos, CS Press, 1988b, p. 25-35

HAREL, D. Bitting the silver bullet. **Computer**, v.25, n.1, p.8-20, Jan. 1992.

HATLEY, D.; PIRBHAI, I.A. **Strategies for real time system specification.** New York, Dorset, 1988.

HOLLAND, J.; JENSEN, B.; TOMASELLO, M.; HEIDRICH, G.; ETTERICH, D. The process of selecting man-machine interface software package for use in a process control system. **ISA Transactions**, v.31, n.3, p.101-10, Sept. 1992

HOLZNER, S. **Visual Basic for Windows** versão 3.0**.** 3.ed. Rio de Janeiro, Campus, 1994

HOOMAN, J.J.M.; RAMESH, S.; ROEVER, W.P. A compositional axiomatization of statecharts. **Theoretical Computer Science**, v.101, p.45-65, 1992.

HUIZING, C; ROEVER, W.P. Introduction to design choices in the semantics of statecharts. **Informations Processing Letters,** v.37, n.4, p.205-13, Feb. 1991.

ICHIKAWA, T.; HIRAKAWA, M. Iconic programming: where to go?. **IEEE Software**, v.7, n.6, p.63-8, Nov. 1990.

JACOB, R.J.K. A state transition diagram language for visual programming. **Computer**, v.18, n.8, p.51-9, Aug. 1985

JACOBSON, I. Is object technology software's industrial platform?. **IEEE Software**, v.10, n.1, p.24-30, Jan. 1993.

JOHNSON, W.L., FEATHER, M.S., HARRIS, D.R. Representation and presentation of requirements knowledge. **IEEE Transactions on Software Engineering**, v.18, n.10, p.853-69, Oct. 1992.

JOSÉ NETO, JOÃO. **Introdução à compilação.** Rio de Janeiro, Livros Técnicos e Científicos, 1987.

JOSÉ NETO, JOÃO. **Contribuições à metodologia de construção de compiladores,** São Paulo, 1993, 272p. Tese (Livre Docência) - Escola Politécnica, Universidade de São Paulo.

JOSÉ NETO, JOÃO. Adaptative automata for context-sensitive languages. **ACM Sigplan Notices,** v.29, n.9, p.115-24, Sept. 1994

KOTONYA, G.; SOMMERVILLE, I. Viewpoints for requirements definition. **Software Engineering Journal**, v.7, n.11, p.375-87, Nov. 1992

KRASNER, H.; et al. Lessons learned from a software process modeling. **Communications of the ACM**, v.35, n.9, p.91-100, Sept. 1992.

LEVESON, N.G. The challenge of building process-control software. **IEEE Software**, v.7, n.6, p.55-68, Nov. 1990.

LIN, C.Y.; LEVARY, R.R. Computer-aided software development process design. **IEEE Transactions on Software Engineering**, v.15, n.9, p.1025-37, Sept. 1989.

LINDLAND, O.I.; SINDRE, G.; SOLVBERG, A. Understanding quality in conceptual modeling. **IEEE Software**, v.11, n,2, p.42-9, Mar. 1994.

LUCENA, F.N.; LIESENBERG, H. **Programming dialogue control of user interfaces using statecharts.** Campinas, Departamento de Ciência da Computação da UNICAMP, 1993. p.1-47, (Relatório Técnico DCC-28/93).

MACHADO, C. Linguagem feita para os olhos. **Exame Informática**, v.6, n.70, p.44-6, jan. 1992.

MAIDEN, N.A.; SUTCLIFFE, A.G. Exploiting reusable specifications through analogy. **Communications of the ACM**, v.35, n.4, p.55-64, Apr. 1992.

MANDELKERN, D. GUIs The next generation. **Communications of the ACM**, v.36, n.4, p.36-9, Apr. 1993

MARANINCHI, F. Argonaute: graphical description, semantics and verification of reactive systems by using a process algebra. In: WORKSHOP ON AUTOMATIC VERIFICATION METHODS FOR FINITE STATE SYSTEMS, Berlin, 1989. **Proceedings.** New York, LNCS, 1989, p.38-53.

MARCUS, A. Human communications issues in advanced UIs. **Communications of the ACM**, v.36, n.4, p. 101-9, Apr. 1993.

MASIERO, P.C.; FORTES, R.P.M.; BATISTA NETO, J.E.S. Edição e simulação do aspecto comportamental de sistemas de tempo real. In: SEMINÁRIO INTEGRADO DE HARDWARE E SOFTWARE, 18°, Santos, 1990. **Anais.** p.45-61.

MEYER, B. On formalism in specifications. **IEEE Software**, v.2, n.1, p.6-25, Jan. 1985.

MOJDEHBAKHSH, R.; TSAI, W.T.; KIRANI, S.; ELLIOTT, L. Retrofiting software safety in an implantable medical device. **IEEE Software**, v.11, n.1, p.41-50, Jan. 1994.

MORSE, A.; REYNOLDS, G. Overcoming current growth limits in UI development. **Communications of the ACM**, v.36, n.4, p.73-81, Apr. 1993.

MUSA, J.D.; ACKERMAN A.F. Quantifying software validation: when to stop testing?. **IEEE Software**, v.6, n.3, p.19-27, May 1989

MYERS, B.A. User interface tools: introduction and survey. **IEEE Software**, v.6, n.1, p.15-23, Jan. 1989.

MYERS, B.A. Demonstrational interfaces: a step beyond direct manipulation. **Computer**, v.25, n.8, p.61-73, Aug. 1992.

NERSON, J.M. Applying object-oriented analysis and design. **Communications of the ACM**, v.35, n.9, p.63-74, Sept. 1992.

NIELSEN, J. Noncommand user interfaces. **Communications of the ACM**, v.36, n.4, p.83-99, Apr. 1993a.

NIELSEN, J. Iterative user-interface design. **Computer**, v.26, n.11, p.32-41, Nov. 1993b.

NOTA, G.; PACINI, G. Querying of executable software specifications. **IEEE Transactions on Software Engineering**, v.18, n.8, p. 705-16, Aug. 1992.

PALMER, J.D.; FIELDS, N.A. An integrated environment for requirements engineering. **IEEE Software**, v.9, n.3, p.80-5, May 1992.

PORISINI, A. C.; DE PAOLI, F.; G., MANDRIOLI, D. Software specialization via symbolic execution. **IEEE Transactions on Software Engineering**, v.17, n.9, p.884-99, Sept. 1991.

POTTS, C.; TAKAHASHI, K., ANTÓN, A.I. Inquiry-based requirements analisys. **IEEE Software**, v.11, n.2, p.21-32, Mar. 1994.

PRESSMAN, R. S. **Software Engineering -** a practiner's approach**.** 3.ed., New York, Mac Graw Hill, 1992

RAEDER, G. A survey of current graphical programming techniques. **Computer**, v.18, n.8, p.11-25, Aug. 1985.

ROMAN, G.C.; COX, K.C. A taxonomy of program visualization systems. **Computer**, v.26, n.12, p.11-24, Dec. 1993.

ROOK, P. Software development process models. **Software Reliability Handbook**, London, Elsevier, 1990, p.413-39.

SHEPARD, T.; SIBBALD, S.; WORTLEY, C. A visual software process language. **Communications of the ACM**, v.35, n.4, p.37- 43, Apr. 1992.

SIDDIGI, J. Challenging universal truths of requirements engineering **IEEE Software**, v.11, n.2, p.18-9, Mar. 1994.

SNYDER, A. The essence of objects: concepts and terms. **IEEE Software**, v.10, n.1, p.31-42, Jan. 1993.

WILLIAMS, L.G. Assessment of safety-critical specifications. **IEEE Software**, v.11, n.1, p.51-60, Jan. 1994.

WING, J. M. A specifier's introduction to formal methods. **Computer**, v.23, n.9, p.8-24, Sept. 1990.

YOUDON, E. **Análise estruturada moderna.** Rio de Janeiro, Campus, 1992.

ANEXO A - MANUAL DE OPERAÇÃO DO STAD

Neste anexo é apresentado o manual de operações do STAD. No item A.1 descrevem-se os componentes presentes nas diversas janelas que fazem parte do STAD. No item A.2 são descritas, de forma sucinta, as principais funções disponíveis, enquanto que no item A.3 é feita a exibição das janelas do sistema, seguida da descrição dos elementos componentes de cada uma delas.

A.1. Elementos Componentes das Janelas

Os elementos presentes nas janelas que fazem parte do STAD são:

- **Botões**: possibilitam a seleção de funções disponíveis no STAD, conforme a tela que estiver sendo exibida. A seleção de um botão pode ser efetuada, da mesma forma que qualquer botão em ambiente Windows, por meio do seu acionamento pelo "mouse" ou, alternativamente, por meio de acionamentos sucessivos da tecla "Tab", seguida da tecla "Enter"

- **Menus**: contêm a indicação de uma série de comandos, agrupados por funções, possibilitando o acionamento de funções disponíveis no STAD. A seleção de um item do menu pode ser efetuada, como qualquer menu em ambiente Windows, por meio do acionamento pelo "mouse" ou pela tecla de atalho "Alt" seguida da letra grifada no menu.

- **Campos**: são representados por caixas onde deve ser digitado um texto que fornece informações ao STAD, tais como nomes ou valores necessários quando se estiver entrando com os dados relativos a um projeto.

- **Listas**: são representadas por caixas que contêm um conjunto de nomes ou valores, cujo contexto sempre está ligado ao tipo de informações presentes ou requisitadas na tela que estiver sendo exibida.

A.2. Funções Disponíveis no STAD

As principais funções disponíveis no STAD são apresentadas, de forma resumida, na tabela abaixo. Esta tabela é composta pela descrição de cada função (coluna FUNÇÃO), pelo comando ou ação necessário para ativar a função que estiver sendo descrita (coluna COMANDO/AÇÃO) e pela janela em que tal comando ou ação deve ser realizado para que se obtenha a função desejada (coluna JANELA). a coluna COMANDO/AÇÃO pode conter uma seqüência de operações que efetuem uma função.

FUNÇÃO	COMANDO/AÇÃO	JANELA
Abertura do arquivo do Banco de Dados do STAD ("tese1.mdb")	Botão Confirma ou duplo clique no nome do arquivo	Seleção de Arquivos
Abertura de Projeto	Menu Projeto-Abrir Projeto	Principal
	Seleção de Projeto e Botão Confirma	Abrir/Excluir Projeto
Exclusão de Projeto	Menu Projeto-Excluir Projeto	Principal
	Seleção de Projeto e Botão Confirma	Abrir/Excluir Projeto
Criação de Novo Projeto	Menu Projeto-Novo Projeto	Principal
	Seleção de Nome de Projeto e Botão Confirma	Novo Projeto
Alteração de Nome de Projeto	Menu Projeto-Salvar Como	Principal
	Seleção de Projeto e de novo nome e Botão Confirma	Salvar Como/Alterar Nome de Projeto
Salvamento de Projeto com outro Nome	Menu Projeto-Alterar Nome	Principal
	Seleção de Projeto e de outro Nome e Botão Confirma	Salvar Como/Alterar Nome de Projeto
Criação de Bolha OR	Menu Bolha-Criar Bolha OR e posicionamento da Bolha	Principal

	Seleção de Nome e Tipo e Botão Confirma	Criar Bolha OR
Criação de Bolha AND	Menu Bolha-Criar Bolha AND e posicionamento da Bolha	Principal
	Seleção de Nome e Botão Confirma	Criar Bolha AND
Criação de Bolha Primitiva	Menu Bolha-Bolha Primitiva-Criar e Posicionamento da Bolha	Principal
	Seleção de Tipo e Botão Confirma	Bolhas Primitivas

FUNÇÃO	COMANDO/AÇÃO	JANELA
Alteração de Parâmetros de Bolha Primitiva	Menu Bolha Primitiva-Parâmetros e seleção da Bolha	Principal
	Seleção de Parâmetros e Botão Atribui	Parâmetros da Bolha Primitiva
Movimentação de Bolhas	Menu Bolha-Mover, seleção e posicionamento da Bolha	Principal
Alteração de Tamanho de Bolhas	Menu Bolha-Tamanho, seleção e alteração de tamanho da Bolha	Principal
Exclusão de Bolhas	Menu Bolha-Excluir e seleção da Bolha	Principal
Alteração do Nome/Tipo de Bolhas	Menu Bolha-Alterar e seleção da Bolha	Principal
	Escolha de novo nome e tipo e Botão Confirma	Alteração de Bolhas
Detalhamento de Bolhas	Menu Bolha-Detalhar Mais e seleção da Bolha	Principal
Criação de Ligações	Menu Ligações-Criar Dependente/Independente Interna/Externa Entrada/Saída e posicionamento da Ligação	Principal
	Seleção de Atributos e Botão Confirma	Dados da Ligação
	Associação de Função Adaptativa, de Ordem de Execução e Botão Confirma	Seleção de Função Adaptativa
Exclusão de Ligações	Menu Ligações-Excluir e seleção da Ligação	Principal
Alteração de Atributos de Ligações	Menu Ligações-Alterar e seleção da Ligação	Principal

	Escolha de novos Atributos e e Botão Confirma ou Adaptativa	Alteração de Dados da Ligação
	Associação de Função Adaptativa, de Ordem de Execução e Botão Confirma	Seleção de Função Adaptativa
Alteração de Parâmetros de Ação Adaptativa associada	Menu Ligações-Alterar Parâmetros	Principal
	Seleção de Parâmetros (Botões >>1 e >>2) e Botão Volta	Seleção de Parâmetros

FUNÇÃO	COMANDO/AÇÃO	JANELA
Impressão de conteúdo de Janela	Menu Ferramentas-Imprimir Formulário	Principal
Acesso ao Banco de Dados ("tese1.mdb")	Menu Ferramentas-Abrir Banco de Dados	Principal
	Programa VISDATA	VISDATA
Testes de Inconsistências entre os níveis de detalhamento	Menu Simulação-Inconsistências	Principal
	Botão Volta	Inconsistências
Exibição de Árvore de hierarquia	Menu Simulação-Árvore	Principal
Exibição de Vista Panorâmica	Menu Simulação-Zoom	Principal
Reinício do Relógio	Menu Simulação-Zera Relógio	Principal
Exibição de Entradas Externas	Menu Simulação-Disparo	Principal
Obtenção de Valores decorrentes de uma simulação	Menu Simulação-Valores e seleção de uma Bolha Primitiva	Principal
	Botão Volta	Valores de Execução
Seleção do Modo de Execução da Simulação	Menu Simulação-Modo de Execução Manual/Automático	Principal
Seleção dos níveis a serem simulados	Menu Simulação-Execução Sem Detalhamento/ Detalhamento Completo/Seleciona Detalhamento	Principal
Execução de um passo de simulação	Menu Simulação-Passo ou "CTRL + P"	Principal

Retorno ao Statechart antes da simulação de ações adaptativas	Menu Simulação-Limpa	Principal
Cancelamento da informação de último estado visitado	Menu Simulação-Limpa História	Principal
Abertura de Função Adaptativa	Menu Fção Adap-Editar	Principal
	Menu Função Adapt-Criar	Montagem de Função Adaptativa
	Seleção de Função Adaptativa e Botão Confirma	Alt./Exc Funções Adaptativas
	Seleção de Tipo de Elemento e Botão OK ou Seleção de Elemento e Botão Exclui	Seleção de Elementos de Função Adaptativa

FUNÇÃO	COMANDO/AÇÃO	JANELA
Exclusão de Função Adaptativa	Menu Fção Adap-Editar	Principal
	Menu Função Adapt-Excluir	Montagem de Função Adaptativa
	Seleção de Função Adaptativa e Botão Confirma	Alt./Exc Funções Adaptativas
Criação de Nova Função Adaptativa	Menu Fção Adap-Editar	Principal
	Menu Função Adapt-Criar	Montagem de Função Adaptativa
	Seleção de Nome para a Função Adaptativa e Botão Confirma	Nova Função Adaptativa

	Seleção de Tipo de Elemento e Botão OK ou Seleção de Elemento e Botão Exclui	Seleção de Elementos de Função Adaptativa
Alteração de Nome de Função Adaptativa	Menu Fção Adap-Editar	Principal
	Menu Função Adapt-Alterar Nome	Montagem de Função Adaptativa
	Seleção de novo Nome para a Função Adaptativa e Botão Confirma	Salvar Como/Alt Nome Função Adapt.
Salvamento de Função Adaptativa com outro Nome	Menu Fção Adap-Editar	Principal
	Menu Função Adapt-Salvar Como	Montagem de Função Adaptativa
	Seleção de Outro Nome para a Função Adaptativa e Botão Confirma	Salvar Como/Alt Nome Função Adapt.

Nas descrições apresentadas para cada janela, as alusões a nomes de arquivos , projetos, bolhas e funções adaptativas, entre outros, referem-se aos exemplos ilustrados nas figuras das telas que acompanham a descrição.

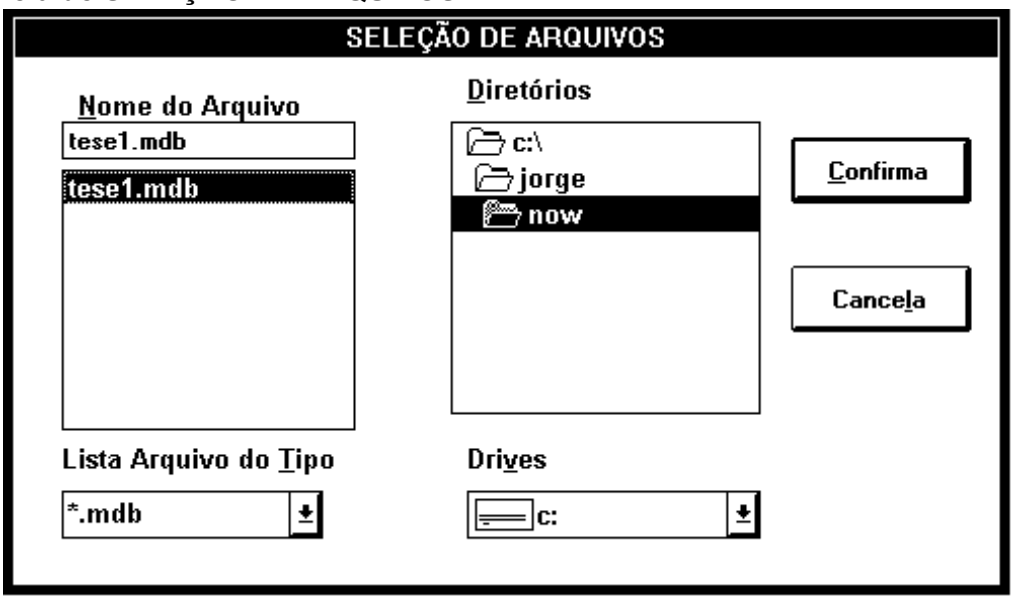
Janela de ABERTURA

Botões

Continua: Causa a abertura da janela de **Seleção de Arquivos**

Sai do Sistema: Causa o término da execução do STAD - Sistema para Criação e Execução de Statecharts Adaptativos.

Janela de SELEÇÃO DE ARQUIVOS



Botões

***Confirma**:* Ocasiona a abertura do arquivo especificado na lista **Nome do Arquivo**. O nome do arquivo selecionado deve ser "**tese1.mdb**", que é o arquivo que contém a estrutura apropriada do Banco de Dados utilizado no programa. Se o arquivo selecionado for diferente de "**tese1.mdb**", uma mensagem de erro é exibida.

***Cancela**:* Faz com que a janela de **Seleção de Arquivos** seja fechada, retornado-se para a janela de **Abertura**.

Listas

***Nome do Arquivo**:* esta lista exibe os nomes de todos os arquivos contidos no diretório selecionado na Lista de **Diretórios** e com a extensão especificada em **Lista Arquivos do Tipo** (*.mdb ou *.*).

Um duplo clique sobre o nome do arquivo selecionado faz com que o mesmo seja aberto.

***Lista Arquivos do Tipo**:* esta lista exibe os diretórios contidos no Drive selecionado na Lista **Drives**.

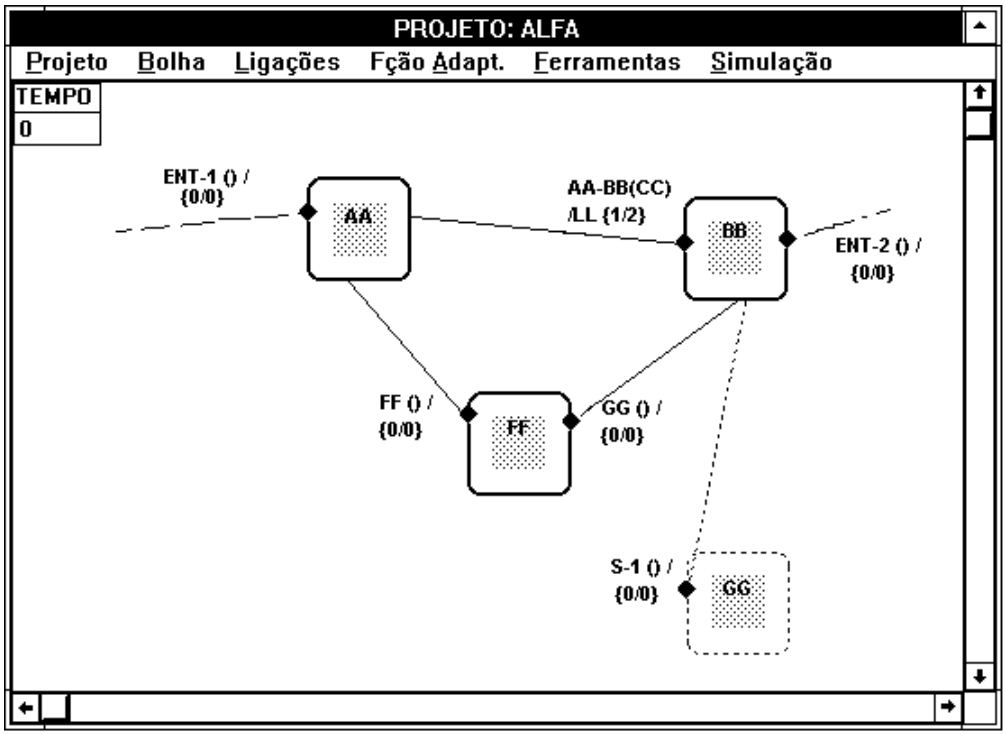
***Diretórios**:* esta lista permite a seleção do diretório que contém os arquivos a serem exibidos na Lista **Nome do Arquivo**. Um duplo clique sobre o nome do diretório selecionado faz com que o mesmo seja aberto.

***Drives**:* esta lista permite a seleção do Drive em que se deseja abrir o arquivo com o banco de dados, ou seja "**tese1.mdb**".

Campos

***Nome do Arquivo**:* este campo exibe o nome do arquivo selecionado da Lista **Nome do Arquivo**.

Janela PRINCIPAL



Menus:

Projeto - Abrir Projeto*:* Habilita a função de abertura de um Projeto já existente, causando a abertura da Janela **Abrir/Excluir Projeto**.

Projeto - Novo Projeto*:* Habilita a função de criação de um novo Projeto, causando a abertura da Janela **Novo Projeto**.

Projeto - Alterar Nome*:* Habilita a função de alteração de nome de um Projeto já existente, fazendo com que seja aberta a Janela **Salvar Como/Alterar Nome de Projeto**.

Projeto - Salvar Como*:* Habilita a função de se salvar um Projeto já existente com outro nome, fazendo com que seja aberta a Janela **Salvar Como/Alterar Nome de Projeto**.

Projeto - Excluir Projeto*:* Habilita a função de exclusão de um Projeto já existente, causando a abertura da Janela **Abrir/Excluir Projeto**.

Projeto - Fim*:* Habilita o fechamento da Janela **Principal** e a volta para a Janela de **Abertura**. Se um Projeto estiver aberto na Janela **Principal**, no momento da seleção desta opção do menu, o mesmo será fechado.

***Bolha - Criar Bolha OR**:* Habilita a criação de uma Bolha do tipo OR, fazendo com que seja aberta a Janela **Criar Bolha OR**.

***Bolha - Criar Bolha AND**:* Habilita a criação de uma Bolha do tipo AND, fazendo com que seja aberta a Janela **Criar Bolha AND**.

***Bolha - Bolha Primitiva - Criar**:* Habilita a criação de uma Bolha Primitiva, fazendo com que seja aberta a Janela **Bolhas Primitivas**.

***Bolha - Bolha Primitiva - Parâmetros**:* Habilita a função de determinação dos parâmetros de uma Bolha Primitiva, fazendo com que seja aberta a Janela **Parâmetros da Bolha Primitiva**.

***Bolha - Mover**:* Habilita a função de movimentação de Bolhas na Janela **Principal**. Deve-se selecionar a bolha que se deseja mover, conforme detalhado no item A.3 e arrastá-la, com o botão do mouse pressionado, para o novo local selecionado. Eventuais ligações existentes na Bolha também serão movidas em conformidade com sua nova posição.

Bolha - Tamanho: Habilita a função de alteração de tamanho de Bolhas na Janela **Principal**. Deve-se selecionar a bolha cujo tamanho se deseja alterar, conforme detalhado no item A.3 e arrastar suas bordas laterais direita e inferior até que se atinja o novo tamanho desejado.

***Bolha - Excluir**:* Habilita a função de exclusão de Bolhas na Janela **Principal**. Deve-se selecionar a bolha que se deseja excluir, conforme detalhado no item A.3. Após a bolha ter sido selecionada, é exibida uma mensagem para confirmar ou não sua exclusão. Se for escolhida a opção de cancelamento da exclusão, nada é feito. Se for escolhida a opção de confirmação da exclusão, a bolha é excluída, bem como eventuais ligações existentes, que tenham como origem ou destino a Bolha excluída.

***Bolha - Alterar**:* Habilita a função de alteração de Bolhas na Janela **Principal**. Deve-se selecionar a bolha que se deseja alterar, conforme detalhado no item A.3, fazendo com que seja exibida a Janela **Alteração de Bolhas**.

***Bolha - Detalhar Mais**:* Habilita a função de se fazer um detalhamento de Bolhas na Janela **Principal**. Deve-se selecionar a bolha que se quer detalhar, conforme detalhado no item A.3, fazendo com que seja exibida outra Janela semelhante à **Principal**, permitindo a edição do detalhamento da Bolha selecionada.

***Bolha - Detalhar Voltar**:* Desabilita a função de detalhamento de Bolhas na Janela **Principal**. A Janela que estiver aberta com o detalhamento é fechada,

voltando-se para o nível imediatamente superior. Se estiver sendo detalhado o nível zero, apenas é exibida uma mensagem de erro, indicando que não se pode voltar além deste nível, que é o mais geral do Statechart.

***Ligações - Criar Dependente Interna**:* Habilita a função de criação de uma Ligação Dependente Interna. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Criar Dependente Externa Entrada**:* Habilita a função de criação de uma Ligação Dependente Externa de Entrada. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Criar Dependente Externa Saída**:* Habilita a função de criação de uma Ligação Dependente Externa de Saída. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Criar Independente Interna**:* Habilita a função de criação de uma Ligação Independente Interna. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Criar Independente Externa Entrada**:* Habilita a função de criação de uma Ligação Independente Externa de Entrada. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Criar Independente Externa Saída**:* Habilita a função de criação de uma Ligação Independente Externa de Saída. O modo de realização dessa ligação encontra-se descrito no item A.3.

***Ligações - Excluir**:* Habilita a função de exclusão de uma Ligação. Deve-se selecionar a ligação a ser excluída, conforme detalhado no item A.3, fazendo com que seja exibida uma mensagem, solicitando a confirmação ou não da exclusão da Ligação. Se a exclusão for confirmada, a Ligação será apagada, caso contrário, nada será feito. Se a Ligação selecionada para ser excluída estiver associada a qualquer Ação Adaptativa, é exibida uma mensagem informando que a mesma só poderá ser excluída após a remoção dessa associação.

***Ligações - Alterar**:* Habilita a função de alteração de uma Ligação. Deve-se selecionar a ligação a ser alterada, conforme detalhado no item A.3, fazendo com que seja aberta a Janela de **Alteração de Dados da Ligação**.

***Ligações - Alterar Parâmetros**:* Habilita a função de alteração de parâmetros de uma Ação Adaptativa previamente associada à ligação selecionada. Deve-se selecionar a ligação a ter os parâmetros alterados, conforme detalhado no item A.3, fazendo com que seja aberta a Janela de **Seleção de Parâmetros**.

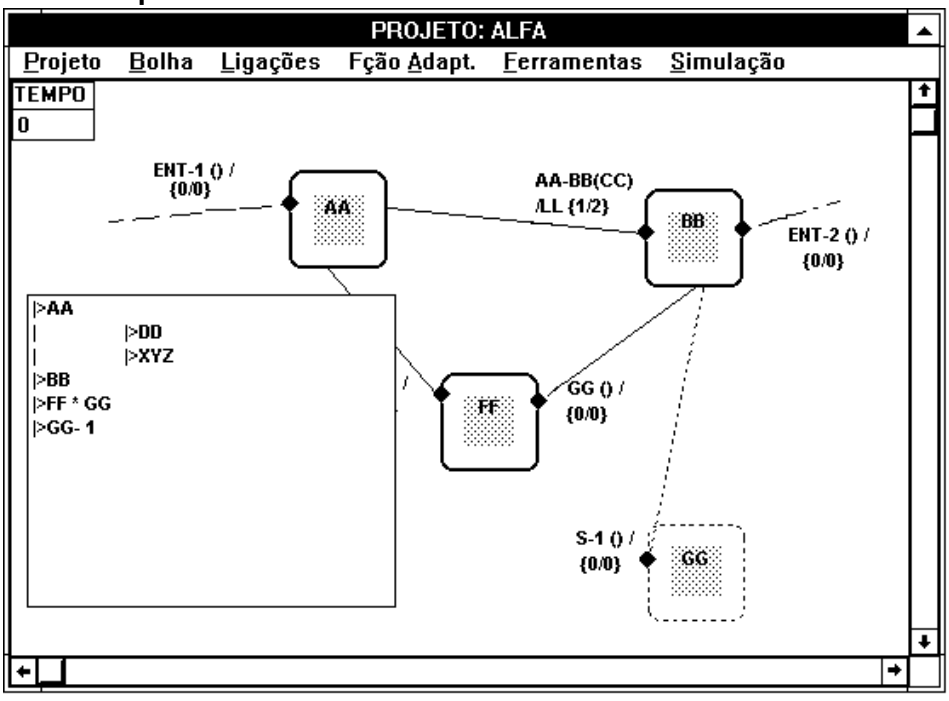
***Fção Adap. - Editar**:* Habilita a função de edição de Funções Adaptativas, abrindo a Janela de **Montagem de Função Adaptativa**.

***Ferramentas - Imprimir Formulário**:* Habilita a impressão da parte da Janela **Principal** que estiver sendo mostrada, fazendo com que seja exibida uma mensagem solicitando confirmação ou não da área do Statechart a ser impressa. Uma vez confirmada, ocorre a impressão, desde que a impressora esteja conectada e ligada. Caso contrário, não ocorre a impressão, sendo exibida uma mensagem de erro.

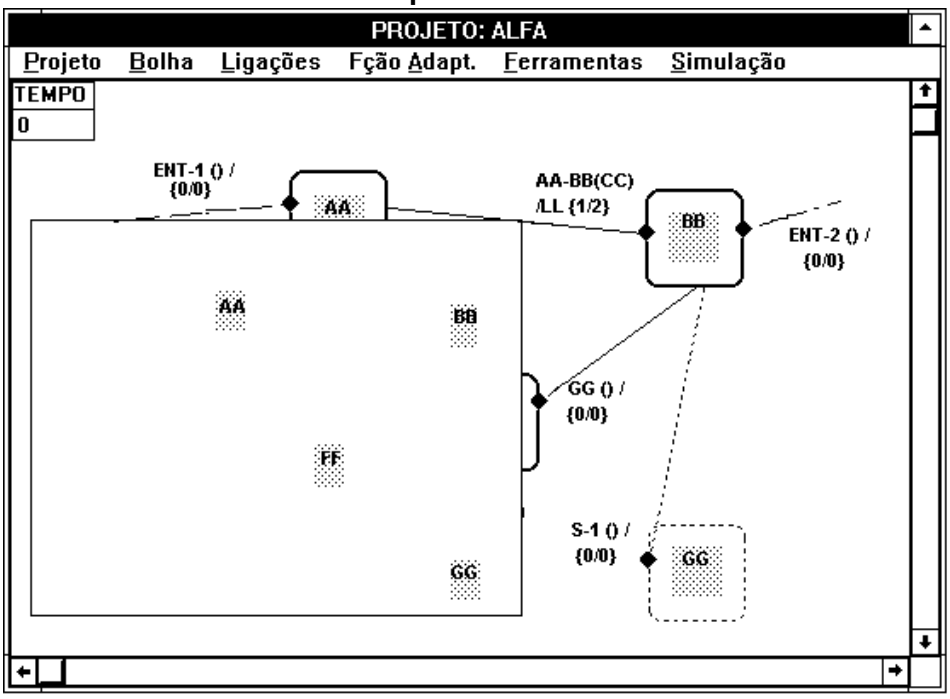
***Ferramentas - Abrir Banco de Dados**:* Realiza a chamada do programa **VISDATA**, que permite o acesso a todas as informações contidas no banco de dados desenvolvido para este editor de Statecharts Adaptativos. Sua finalidade é a de possibilitar a confirmação ou correção de informações exibidas de forma gráfica, pelo editor e executor de Statecharts Adaptativos.

***Simulação - Inconsistências**:* Habilita a realização de testes de inconsistências das entradas/saídas de cada Bolha, com seu respectivo detalhamento, se este existir. São consideradas inconsistências: uma determinada ligação, definida em um dado nível não existe nos níveis imediatamente inferior e superior de detalhamento da Bolha; uma determinada condição não corresponde a uma das Bolhas existentes no nível de detalhamento correspondente; um determinado disparo de outro evento não corresponde a uma Ligação no nível de detalhamento correspondente. Ao ser acionada esta opção do menu é aberta a **Janela de Inconsistências**.

***Simulação - Árvore**:* Controla a exibição da árvore que representa o Projeto que está sendo montado, fazendo com que, caso isto ainda não seja verdade, seja exibido um quadro com a respectiva árvore do Projeto, conforme mostra a Janela abaixo. A finalidade desta árvore é permitir uma visualização completa de toda a estrutura do Statechart que estiver sendo exibido, em todos os seus níveis de detalhamento. Caso este quadro já esteja sendo exibido, o mesmo é retirado da Janela **Principal**.



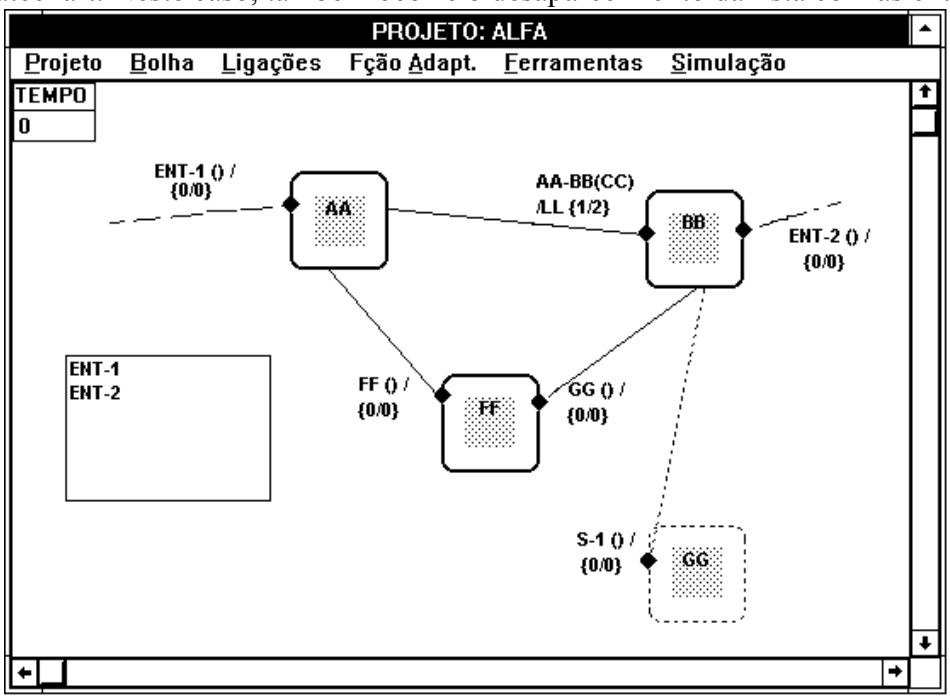
***Simulação - Zoom**:* Controla a exibição de uma vista reduzida que representa o Statechart da Bolha que estiver sendo detalhada, fazendo com que, caso não esteja presente na tela, seja exibido um quadro com a respectiva vista reduzida do detalhamento da Bolha, conforme mostra a Janela abaixo. A finalidade deste zoom é permitir uma visualização completa do Statechart a que pertence a Bolha que estiver sendo detalhada, principalmente no caso de este ultrapassar as dimensões máximas da Janela **Principal**. Caso o zoom já estiver sendo exibido, o mesmo é retirado da Janela **Principal**.



***Simulação - Zera Relógio**:* Habilita a função de iniciar em zero o Campo **Tempo**, que simula um relógio utilizado para a marcação do tempo transcorrido desde o início da simulação do Statechart do Projeto que estiver sendo utilizado. Ao ser acionada esta opção do menu, o valor do Campo **Tempo** é feito igual a zero.

***Simulação - Valores**:* Habilita a função de exibição dos valores que as Ligações vão assumindo conforme o Statechart vai sendo simulado, fazendo com que seja aberta a Janela **Valores de Execução**.

***Simulação - Disparo**:* Habilita a exibição de uma lista que contém todas as entradas independentes externas do nível corrente de detalhamento do statechart que estiver sendo exibido. É apresentada uma lista com as entrada correspondentes, conforme se vê na janela mostrada abaixo. Caso esta lista já esteja sendo exibida, uma "limpeza" será feita no Statechart, sendo que os elementos que tenham sido criados ou excluídos como resultado de ações adaptativas executadas, são restituídos ao seu estado original. Assim, os elementos criados são excluídos e os elementos excluídos são restituídos ao Statechart. Neste caso, também ocorre o desaparecimento da lista com as entradas.



***Simulação - Modo de Execução - Manual**:* Habilita a execução manual da simulação do Statechart. Para que durante uma execução, o Statechart possa transitar entre os estados, é necessário acionar a opção **Simulação - Passo** do menu, ou teclar "Ctrl + P".

***Simulação - Modo de Execução - Automático**:* Habilita a execução automática da simulação do Statechart, ou seja, durante uma execução, o Statechart transita entre as bolhas automaticamente, sem a necessidade de acionamento da opção **Simulação - Passo** do menu. As transições são executadas até que não haja mais possibilidade de alteração do estado do Statechart, no detalhamento de Bolha que estiver sendo executado. Observe-se que, enquanto houver um loop no Statechart, este será executado indefinidamente, até que seja acionada outra opção do menu.

***Simulação - Execução - Sem Detalhamento**:* Habilita a execução da simulação do Statechart apenas em seu nível zero de detalhamento. Eventuais detalhamentos de Bolhas existentes não serão executados enquanto esta opção estiver selecionada.

***Simulação - Execução - Detalhamento Completo**:* Habilita a execução completa da simulação do Statechart, ou seja, todos os níveis de detalhamento de todas as Bolhas são executados, sem necessidade de confirmação por parte do usuário.

***Simulação - Execução - Seleciona Detalhamento**:* Possibilita a seleção dos níveis de detalhamento cuja simulação deva ser executada. Quando uma Bolha for a Bolha corrente, é exibida uma mensagem solicitando a confirmação da execução da simulação do detalhamento da mesma. Caso seja confirmada a execução do detalhamento, é exibida outra Janela deste tipo, permitindo o detalhamento da execução da simulação do detalhamento da Bolha em questão.

***Simulação - Passo**:* Habilita a execução de outro passo do Statechart, ou seja, durante uma execução, o Statechart efetua uma transição entre bolhas a cada seleção deste item do menu. Como tecla de atalho a esta opção do menu, pode ser utilizado o comando "Ctrl + P".

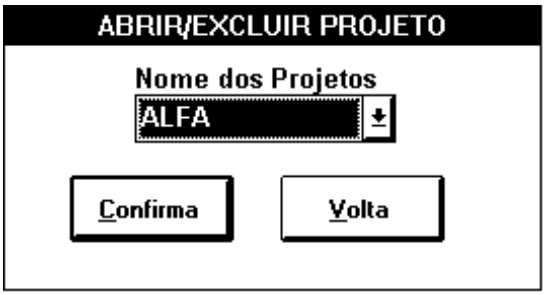
***Simulação - Limpa**:* Restaura a forma original do Statechart. Os elementos que tenham sido criados ou excluídos como resultado de ações adaptativas executadas são restituídos à sua situação original, ou seja, todos os elementos criados são excluídos e todos os elementos excluídos são restituídos ao Statechart.

***Simulação - Limpa História**:* Aciona a função de se limpar a informação da última bolha visitada em uma simulação.

Campo:

***TEMPO**:* Indica o número de unidades de tempo transcorridas desde o início da execução do Statechart em um determinado detalhamento. Quando o detalhamento de uma Bolha é solicitado, este campo aparece com o valor zero, ao ser exibida a Janela com o Statechart detalhado.

Janela ABRIR/EXCLUIR PROJETO



Botões:

Confirma: Se a opção selecionada no menu da Janela **Principal** foi **Projeto - Abrir Projeto**, ocorre a abertura do projeto especificado na lista **Nome dos Projetos**. Se não for selecionado nenhum Projeto desta lista, uma mensagem é exibida, solicitando-se que seja escolhido um Projeto. Após ter sido selecionado um Projeto e ter sido pressionado este botão, tal Projeto é aberto na Janela **Principal**, em seu nível zero de detalhamento, e esta Janela (**Abrir/Excluir Projeto**) é fechada. Se um Projeto já estiver sendo exibido na Janela **Principal**, este é fechado antes de se carregar o Projeto aqui selecionado.

Se a opção selecionada no menu da Janela **Principal** foi **Projeto - Excluir Projeto**, ocorre a exibição de uma mensagem solicitando a confirmação ou não da exclusão do Projeto selecionado na lista **Nome dos Projetos**.

Para os dois casos acima descritos, se não for selecionado nenhum Projeto desta lista, uma mensagem é exibida, solicitando-se que seja escolhido um Projeto. Se, for confirmada a exclusão do Projeto, o mesmo é retirado completamente das tabelas do Banco de Dados. Caso contrário, a operação de exclusão é cancelada.

Volta*:* Faz com que esta Janela seja fechada, retornado-se para a Janela **Principal**. Neste caso, nenhum Projeto é aberto ou excluído.

Listas

Nome dos Projetos*:* Contém o nome de todos os Projetos existentes no Banco de Dados utilizado no programa corrente.

Janela NOVO PROJETO

Botões

***Confirma**:* Causa a criação do projeto especificado no campo **Nome do Novo Projeto**. Se este campo estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome desejado para o novo Projeto. Se o **Nome do Novo Projeto** fornecido coincidir com o de um dos projetos já existentes, uma mensagem é exibida, informando o fato. Após ter sido selecionado um nome válido para o Projeto e ter sido pressionado este botão, tal Projeto é aberto na Janela **Principal**, em seu nível zero de detalhamento (porém sem nenhum elemento), e a Janela **Novo Projeto** é fechada. Se um Projeto já estiver sendo exibido na Janela **Principal**, este será fechado antes de se iniciar a criação do Projeto indicado.

***Volta**:* Faz com que esta Janela seja fechada, retornado-se para a Janela **Principal**. Neste caso, nenhum **Novo Projeto** é criado.

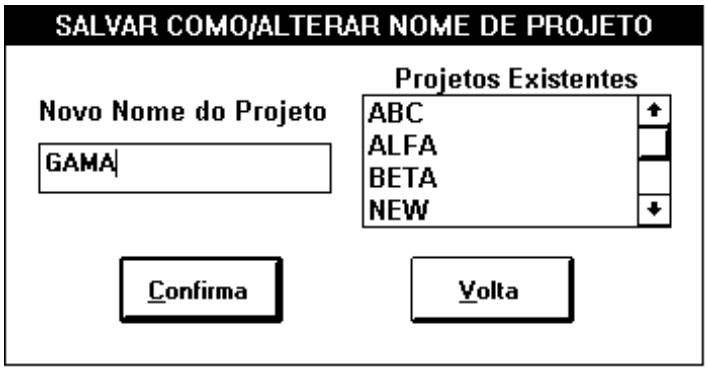
Listas

***Projetos Existentes**:* Contém o nome de todos os Projetos existentes no Banco de Dados utilizado no programa.

Campos

***Nome do Novo Projeto**:* Deve ser preenchido com o nome desejado para o **Novo Projeto**.

Janela SALVAR COMO/ALTERAR NOME DE PROJETO



Botões:

***Confirma**:* Se a opção selecionada no menu da Janela **Principal** foi **Projeto - Alterar Nome**, ocorre a alteração de nome do Projeto especificado na lista **Projetos Existentes** para o nome indicado no campo **Novo Nome do Projeto**.

Se a opção selecionada no menu da Janela **Principal** foi **Projeto - Salvar Como**, ocorre a inclusão, na Base de Dados, de um Projeto exatamente igual ao especificado na lista **Projetos Existentes**, porém apenas com o nome alterado para o nome indicado no campo **Novo Nome do Projeto**.
Para os dois casos acima descritos, se não for selecionado nenhum Projeto da lista **Projetos Existentes**, uma mensagem é exibida, solicitando-se que seja escolhido um dos Projetos. Se o **Novo Nome do Projeto** fornecido coincidir com o de um dos Projetos já existentes, uma mensagem neste sentido é exibida.

***Volta**:* Faz com que a Janela **Salvar Como/Alterar Nome de Projeto** seja fechada, retornado-se para a Janela **Principal**. Neste caso, não ocorre nem a alteração de nome nem a inclusão de um novo Projeto.

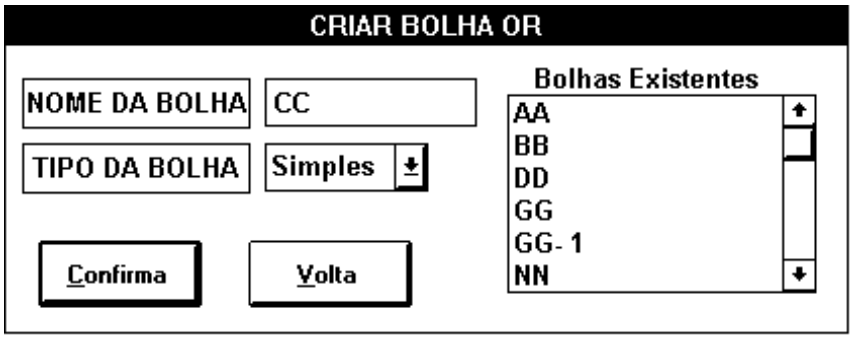
Listas

***Projetos Existentes**:* Contém o nome de todos os Projetos existentes no Banco de Dados.

Campos

***Novo Nome do Projeto**:* Deve ser preenchido com o **Novo Nome** pretendido para o Projeto, tanto no caso de mudança de nome, quanto no caso de se salvar o projeto com outro nome.

Janela CRIAR BOLHA OR



Botões:

***Confirma**:* Permite a criação de uma Bolha do tipo OR, com o nome especificado no campo **Nome da Bolha** e com o tipo especificado na lista **Tipo da Bolha**. Se o campo **Nome da Bolha** estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome desejado para a nova Bolha. Se o nome da nova Bolha coincidir com o de uma das Bolhas já existentes, uma mensagem será exibida, informando que o nome de Bolha pretendido já está sendo utilizado. Após ter sido selecionado um nome válido para a Bolha e ter sido pressionado este botão, esta Janela (**Criar Bolha OR**) é fechada, retornando-se o comando para a Janela de **Principal**, devendo-se então escolher a posição da nova bolha, conforme indicado no item A.3.

***Volta**:* Faz com que esta Janela seja fechada, retornado-se para a Janela **Principal**. Neste caso, não ocorre a criação de uma Bolha do tipo OR.

Listas

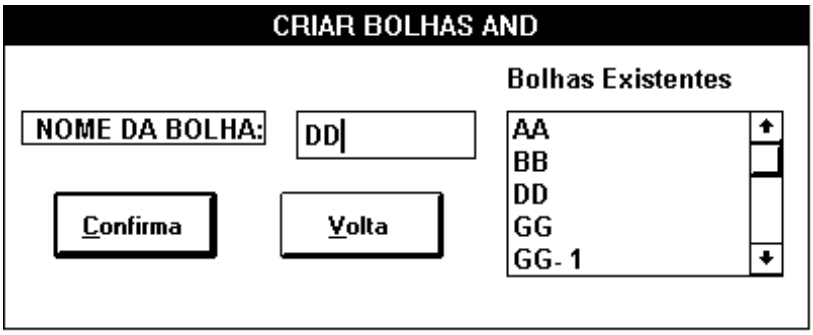
***Bolhas Existentes**:* Contém os nomes das Bolhas dos tipos OR e AND criadas até a ocasião.

***Tipo da Bolha**:* Contém os tipos de Bolha OR possíveis de serem criados: Comum, Default e History.

Campos

***Nome da Bolha**:* Deve ser preenchido com o nome selecionado para a Bolha tipo OR que estiver sendo criada.

Janela CRIAR BOLHA AND



Botões

***Confirma**:* Permite a criação de uma Bolha do tipo AND, com o nome especificado no campo **Nome da Bolha**. Se este campo estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome desejado para a nova Bolha. Se o nome da nova Bolha coincidir com uma das Bolhas já existentes, uma mensagem acusará o fato. Após ter sido selecionado um nome para a Bolha e ter sido pressionado este botão, esta Janela (**Criar Bolha AND**) é fechada, voltando-se o comando para a **Janela Principal**, quando então a posição da nova bolha deve ser escolhida, conforme indicado no item A.3.

***Volta**:* Faz com que esta Janela seja fechada, retornado-se para a **Janela Principal**. Neste caso, não ocorre a criação de uma Bolha do tipo AND.

Listas

***Bolhas Existentes**:* Contém os nomes das Bolhas dos tipos OR e AND criadas.

Campos

***Nome da Bolha**:* Deve ser preenchido com o nome selecionado para a Bolha tipo AND que estiver sendo criada.

Janela ALTERAÇÃO DE BOLHAS

Botões

***Confirma**:* Permite a alteração do nome de uma Bolha e de seu tipo (no caso de Bolhas OR). Quando esta Janela é exibida, os campos **Nome Atual** e **Novo Nome** são preenchidos com o nome atual da Bolha que estiver sendo alterada. Podem ser alterados o campo **Nome Atual** e a lista **Tipo da Bolha** (no caso de Bolhas OR). Se o campo **Novo Nome** estiver em branco, uma mensagem é exibida, solicitando que seja fornecido o nome desejado para a Bolha. Se o novo nome da Bolha coincidir com o de uma das Bolhas já existentes, uma mensagem indicará que o nome pretendido para a Bolha já foi anteriormente utilizado. Após essas seleções terem sido feitas e ter sido pressionado este botão, a **Janela Alteração de Bolhas** é fechada, voltando-se o comando para a **Janela Principal**.

***Volta**:* Faz com que esta **Janela Alteração de Bolhas** seja fechada, retornado-se para a **Janela Principal**. Neste caso, não ocorre a alteração da Bolha selecionada.

Listas

***Outras Bolhas Existentes**:* Contém os nomes das demais Bolhas dos tipos OR e AND criadas anteriormente.

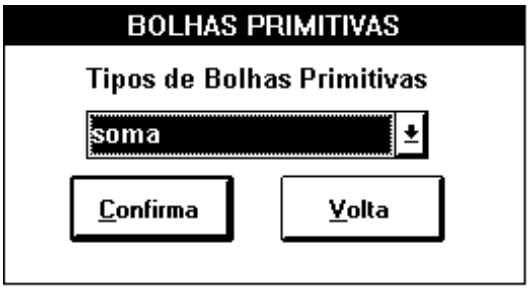
***Tipo da Bolha**:* Contém os tipos de Bolha OR possíveis de serem criados: Comum, Default e History.

Campos

***Nome Atual**:* Contém o nome atual da Bolha que estiver sendo alterada.

***Novo Nome**:* Deve ser preenchido com o novo nome selecionado para a Bolha que estiver sendo alterada.

Janela BOLHAS PRIMITIVAS



Botões

***Confirma**:* Permite a criação de uma Bolha primitiva, do tipo especificado na lista **Tipos de Bolhas Primitivas**, fazendo com que a **Janela Bolhas Primitivas** seja fechada, voltando-se o comando para a **Janela Principal,** quando então deve-se selecionar a posição da nova bolha, conforme indicado no item A.3. Antes de serem selecionados os parâmetros de uma Bolha Primitiva, assume-se que os parâmetros tenham o nome de **X** e **Y**.

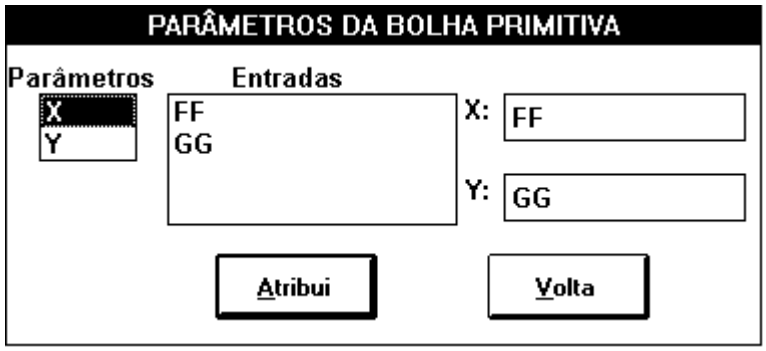
Os nomes **X** e **Y** indicam, no contexto do STAD, que os parâmetros de uma Bolha Primitiva ainda não foram associados a qualquer Ligação cujo destino seja a Bolha e questão. Se a designação dos parâmetros não tiver sido feita, quando for ser realizada a simulação do Statechart, será assumido o valor zero ou "false" como entrada da Bolha Primitiva em questão. O significado dos parâmetros **X** e **Y** é aquele explicado no item 4.2.3a.

***Volta**:* Faz com que a Janela **Bolhas Primitivas** seja fechada, retornado-se o comando para a Janela **Principal**. Neste caso, não ocorre a criação de uma Bolha Primitiva.

Listas

***Tipos de Bolhas Primitivas**:* Contém os tipos de **Bolhas Primitivas** possíveis de serem criados: soma, subtração, multiplicação, divisão, exponenciação, igual, diferente, menor, maior, maior ou igual, menor ou igual, OR, AND, NOT, IF THEN, IF ELSE e DO WHILE.

Janela PARÂMETROS DA BOLHA PRIMITIVA



Botões

Atribui*:* Executa a atribuição do parâmetro selecionado na lista **Parâmetros** com a entrada selecionada da lista **entradas**. O significado dos parâmetros **X** e **Y** é aquele explicado no item 4.2.3a.

Volta*:* Faz com que a Janela **Parâmetros da Bolha Primitiva** seja fechada, retornado-se o comando para a Janela **Principal**. As atribuições confirmadas por meio do botão **Atribui**, são mantidas.

Listas

Parâmetros*:* Contém o nome dos parâmetros: X e Y.

Entradas*:* Contém as entradas da Bolha.

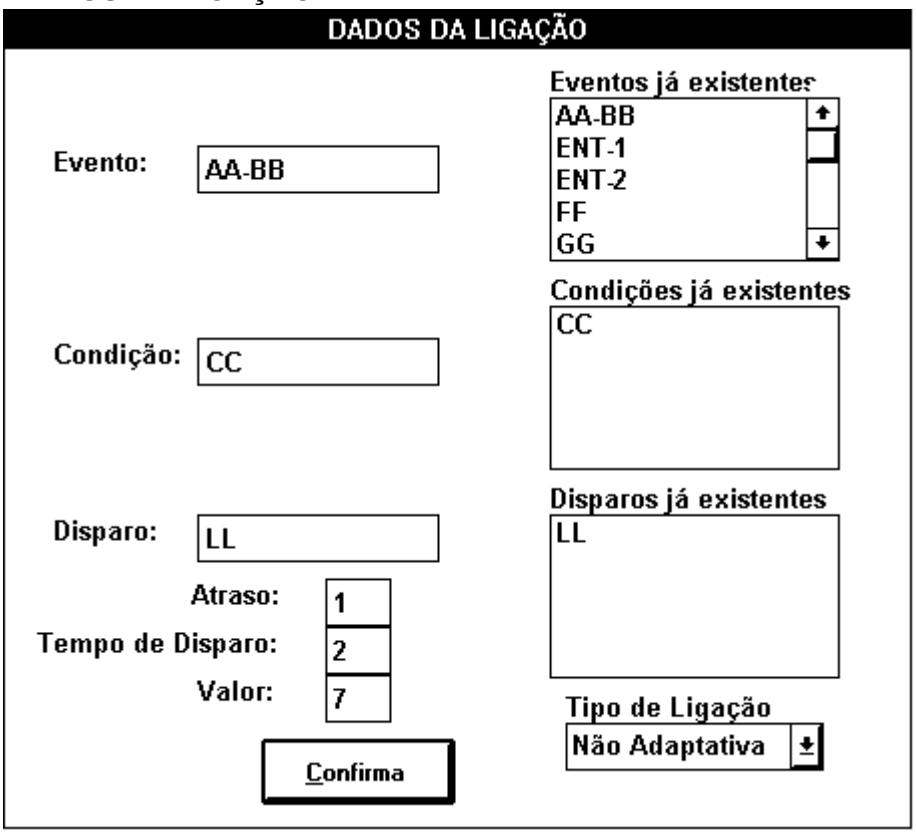
Campos

X*:* Contém o nome da entrada atribuído ao parâmetro **X**.

Y*:* Contém o nome da entrada atribuído ao parâmetro **Y**.

Obs.: o significado dos parâmetros **X** e **Y** é explicado no texto da Janela **Bolha Primitivas**.

Janela DADOS DA LIGAÇÃO



Botões

Confirma: Efetua o salvamento, no Banco de Dados, dos parâmetros fornecidos na Janela **Dados da Ligação**. Se o campo **Evento** estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome do **Evento** desejado para a nova Ligação. Após ter sido selecionado um nome para o **Evento** e ter sido pressionado este botão, a Janela **Dados da Ligação** é fechada, voltando-se o comando para a Janela **Principal**, caso, na lista **Tipo de Ligação**, tenha sido selecionada a opção **Não Adaptativa**. Se houver sido selecionada a opção **Adaptativa** nesta lista, é aberta a Janela **Seleção de Função Adaptativa**, para que uma Função Adaptativa possa ser associada à Ligação que estiver sendo criada.

Listas

***Eventos já Existentes**:* Contém os nomes de todos os **Eventos** criados anteriormente.

***Condições já Existentes**:* Contém os nomes de todas as **Condições** criadas anteriormente.

***Disparos já Existentes**:* Contém os nomes de todos os **Disparos** criados anteriormente.

***Tipo de Ligação**:* Contém os tipos de Ligação possíveis de serem criados: **Adaptativa** e **Não Adaptativa**.

Campos

***Evento**:* Deve ser preenchido com o nome selecionado para o **Evento** da Ligação que estiver sendo criada.

***Condição**:* Deve ser preenchido com o nome selecionado para a **Condição** da Ligação que estiver sendo criada.

***Disparo**:* Deve ser preenchido com o nome selecionado para o **Disparo** da Ligação que estiver sendo criada.

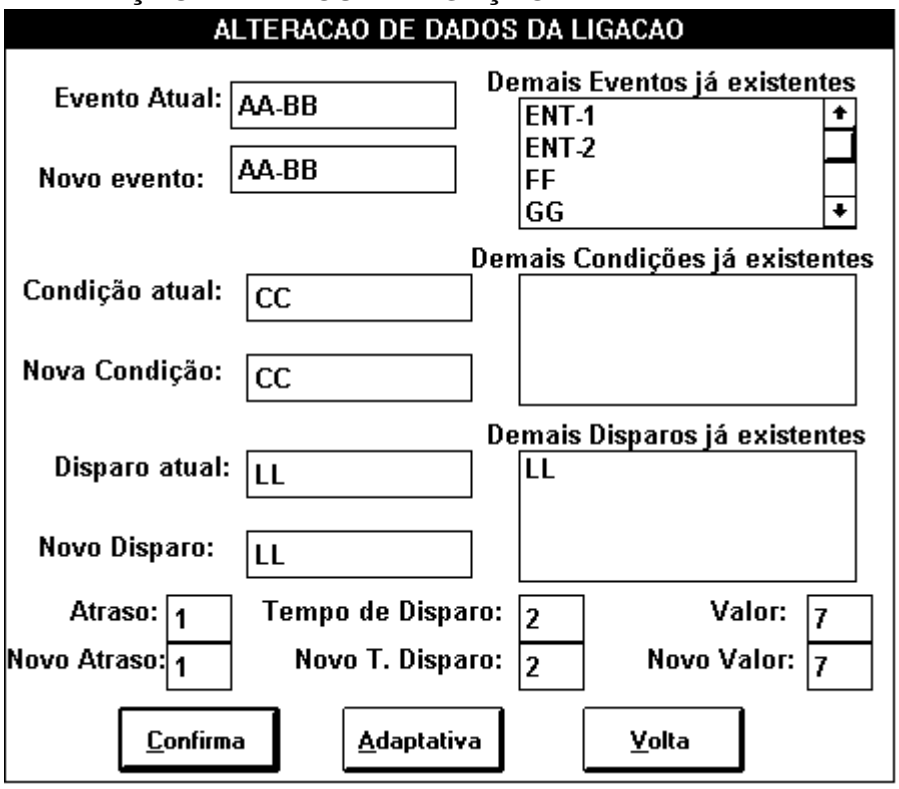
***Atraso**:* Deve ser preenchido com o valor do Atraso selecionado para a Ligação que estiver sendo criada.

***Tempo de Disparo**:* Deve ser preenchido com o valor do **Tempo de Disparo** selecionado para a Ligação que estiver sendo criada.

***Valor**:* Deve ser preenchido com o Valor selecionado para a Ligação que estiver sendo criada.

Obs.: o significado dos campos é descrito no item 4.2.4b do corpo da Tese.

Janela ALTERAÇÃO DE DADOS DA LIGAÇÃO



Botões

***Confirma**:* Efetua o salvamento, no Banco de Dados, dos parâmetros alterados na Janela **Alteração de Dados da Ligação**. Se o campo **Novo Evento** estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome do Evento desejado para a Ligação que estiver sendo alterada. Após ter sido selecionado um nome para o Evento e ter sido pressionado este botão, a Janela **Alteração de Dados da Ligação** é fechada, voltando-se o comando para a Janela **Principal**.

***Adaptativa**:* Habilita a função de alteração da **Função Adaptativa** que estiver sendo utilizada, ou, caso nenhuma Função esteja sendo utilizada, permite a escolha da Função desejada. Ao ser pressionado este botão, a Janela **Alteração de Dados da Ligação** é fechada, e é aberta a Janela **Seleção de Função Adaptativa.**

***Volta**:* Faz com que a **Janela Alteração de Dados da Ligação** seja fechada, retornado-se para a **Janela Principal**. Neste caso, não ocorre a alteração dos parâmetros da Ligação selecionada.

Listas

***Demais Eventos já Existentes**:* Contém os nomes dos demais **Eventos** criados anteriormente.

***Demais Condições já Existentes**:* Contém os nomes das demais **Condições** criadas anteriormente.

***Demais Disparos já Existentes**:* Contém os nomes dos demais **Disparos** criados anteriormente.

Campos

***Evento Atual**:* Contém o nome atual do **Evento** associado à Ligação que estiver sendo alterada.

***Novo Evento**:* Deve ser preenchido com o novo nome selecionado para o **Evento** associado à Ligação que estiver sendo alterada.

***Condição Atual**:* Contém o nome atual da **Condição** associada à Ligação que estiver sendo alterada.

***Nova Condição**:* Deve ser preenchido com o novo nome selecionado para a **Condição** associada à Ligação que estiver sendo alterada.

***Disparo Atual**:* Contém o nome atual do **Disparo** associado à Ligação que estiver sendo alterada.

***Novo Disparo**:* Deve ser preenchido com o novo nome selecionado para o **Disparo** associado à Ligação que estiver sendo alterada.

***Atraso**:* Contém o valor do **Atraso** atual associado à Ligação que estiver sendo alterada.

***Novo Atraso**:* Deve ser preenchido com o novo valor do **Atraso** associado à Ligação que estiver sendo alterada.

***Tempo de Disparo**:* Contém o valor do **Tempo de Disparo** atual associado à Ligação que estiver sendo alterada.

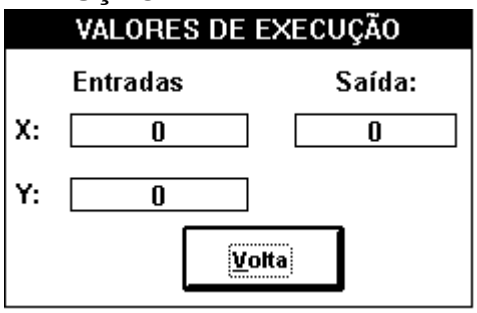
***Novo T. Disparo**:* Deve ser preenchido com o novo valor do **Tempo de Disparo** associado à Ligação que estiver sendo alterada.

***Valor**:* Contém o **Valor** atual associado à Ligação que estiver sendo alterada.

***Novo Valor**:* Deve ser preenchido com o **Novo Valor** associado à Ligação que estiver sendo alterada.

Obs.: o significado de cada um desses campos é descrito em 4.2.4b.

Janela VALORES DE EXECUÇÃO



Botões

Volta*:* Faz com que a Janela **Valores de Execução** seja fechada, retornado-se o comando para a Janela **Principal**.

Campos

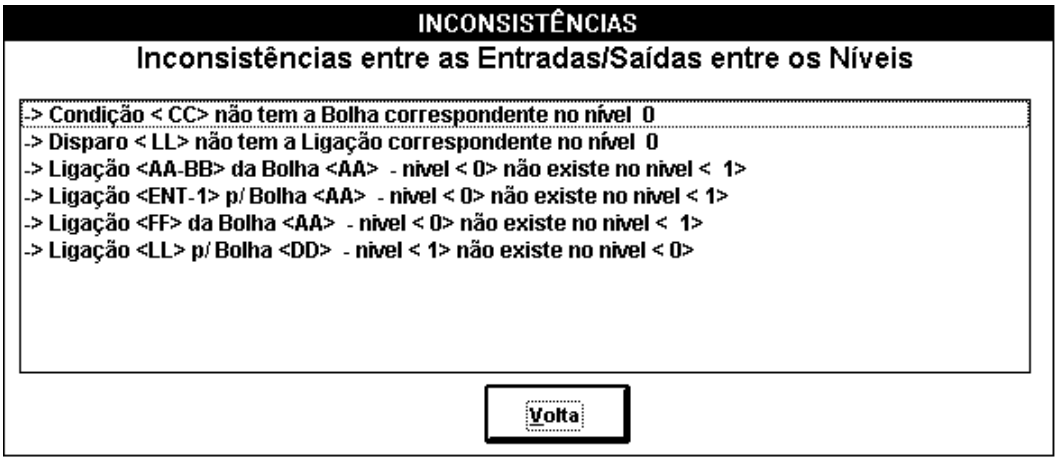
Entradas - X*:* Contém o valor assumido, no momento da abertura da Janela **Valores de Execução**, pela entrada designada como parâmetro **X**.

Entradas - Y*:* Contém o valor assumido, no momento da abertura da Janela **Valores de Execução**, pela entrada designada como parâmetro **Y**.

Saída*:* Contém o valor assumido, no momento da abertura da Janela **Valores de Execução**, pelas saídas da Bolha.

Obs.: O significado dos parâmetros **X** e **Y** é aquele explicado no item 4.2.3a.

Janela INCONSISTÊNCIAS



Botões

Volta: Faz com que a Janela **Inconsistências** seja fechada, retornado-se o comando para a Janela **Principal**.

Listas

Inconsistências entre as Entradas/Saídas entre os Níveis: Contém as inconsistências detectadas no detalhamento do Projeto correto. As inconsistências presentes nesta lista são aquelas já apresentadas na descrição da Janela **Principal**.

- Janela SELEÇÃO DE FUNÇÃO ADAPTATIVA

Botões

Confirma: Causa a escolha da Função Adaptativa selecionada da lista **Funções Adaptativas Existentes**, caso esta não seja a mesma já exibida no campo **Função Adaptativa Utilizada**. Se na lista **Funções Adaptativas Existentes** tiver sido selecionada a opção "nenhuma", e se uma Função já estava associada à Ligação, esta é removida do Statechart correspondente, na Janela **Principal**. A Ação Adaptativa correspondente à Função Adaptativa selecionada é inserida no Statechart correspondente, na Janela **Principal**, sendo antes exibida uma mensagem, solicitando o posicionamento do primeiro elemento a ser criado pela Ação Adaptativa. Este posicionamento é feito pelo acionamento do botão do mouse na posição da Janela **Principal**, em que se deseja inserir o elemento citado na mensagem exibida. Se uma outra Função já estava sendo utilizada, será removida do Statechart antes da inserção da nova Função selecionada.

Volta: Faz com que a Janela **Seleção de Função Adaptativa** seja fechada, retornado-se o comando para a Janela **Principal**. Neste caso, não é realizada nenhuma seleção de Função Adaptativa, seja para a criação de uma Ligação, seja para a sua alteração.

Listas

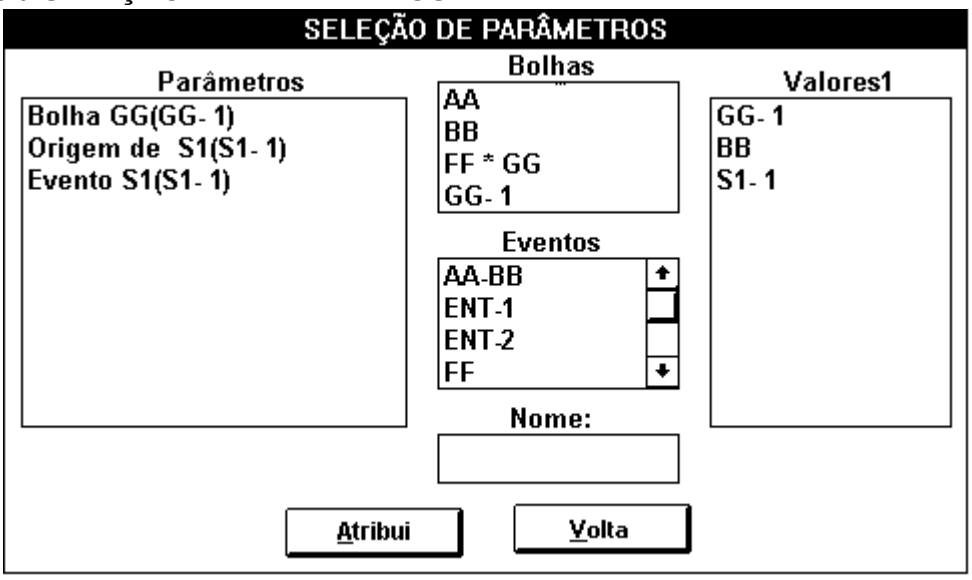
Funções Adaptativas Existentes: Contém o nome de todas as Funções Adaptativas existentes para o Projeto que estiver sendo utilizado.

Ordem de Execução: Contém a ordem de execução da Função Adaptativa: **Anterior** (a Função é executada antes da transição para as Bolhas correspondentes) e **Posterior** (a Função é executada após a transição para as Bolhas correspondentes).

Campos

Função Adaptativa Utilizada*:* Contém o nome da Função Adaptativa que estiver associada à Ligação que estiver sendo criada ou alterada. Se não houver nenhuma Função associada, este campo é preenchido com o valor "nenhuma".

Janela SELEÇÃO DE PARÂMETROS



Botões

***Atribui**:* Executa o preenchimento do parâmetro selecionado da lista **Parâmetros** com os valores das listas **Bolhas** e **Eventos** e do campo **Nome**. No caso de estar sendo feita a atribuição de um parâmetro a uma Bolha ou a um Evento, deve ser preenchido o campo **Nome** com o nome desejado para o elemento em questão. No caso de estar sendo feita a atribuição a uma origem ou destino de uma Ligação, deve ser selecionada uma Bolha da lista **Bolhas**, que servirá como a origem ou destino apontada pela lista **Parâmetros**. Finalmente, no caso de estar sendo feita a atribuição a uma exclusão de Ligação, deve ser selecionado um Evento da lista **Eventos**, que será excluído quando da execução da Função Adaptativa. Os parâmetros atribuídos são exibidos na lista **Valores**.

***Volta**:* Faz com que a Janela **Seleção de Parâmetros** seja fechada, retornado-se o comando para a Janela **Principal**. As atribuições realizadas por meio do botão **Atribui** são mantidas.

Listas

***Parâmetros**:* Contém a relação dos parâmetros que podem ser alterados. Esses parâmetros são: nome de Bolhas e de Eventos, origem e destino de Ligações e Eventos a serem excluídos. Quando é selecionado um elemento desta lista, a lista **Valores** também passa a apontar para a linha correspondente ao parâmetro apontado pela lista **Parâmetros**.

***Valores**:* Contém a relação dos valores que os parâmetros irão assumir quando da execução da Função Adaptativa selecionada. Quando é selecionado um elemento desta lista, a lista **Parâmetros** também passam a apontar para a linha correspondente ao valor apontado pela lista **Valores**.

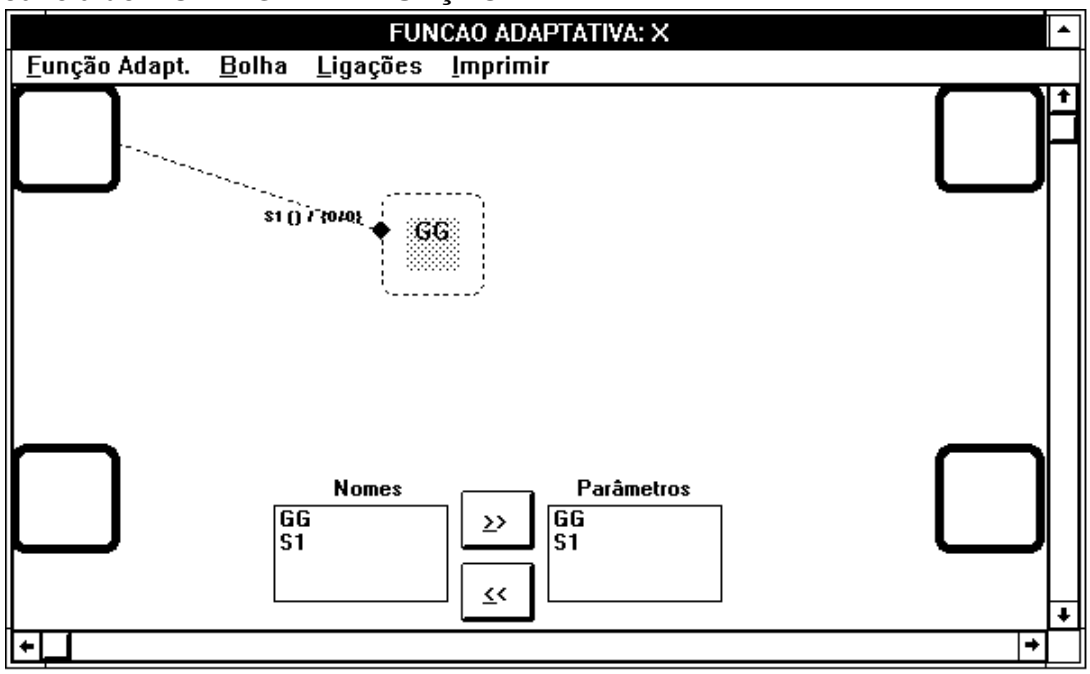
***Bolhas**:* Contém a relação de todas as Bolhas existentes no nível de detalhamento a que pertence a Bolha que estiver sendo editada.

***Eventos**:* Contém a relação de todos os Eventos existentes no nível de detalhamento a que pertence a Bolha que estiver sendo editada.

Campos

***Nome**:* Deve ser preenchido com o nome que a Bolha ou o Evento selecionados na lista **Parâmetros** deve assumir.

Janela de MONTAGEM DE FUNÇÃO ADAPTATIVA



Botões

>>*:* Faz com que o elemento selecionado na lista **Nomes** passe a ser um parâmetro da Função Adaptativa, o que é indicado pela sua presença na lista **Parâmetros**. Se um elemento selecionado já fizer parte da lista **Parâmetros**, nenhuma ação é executada.

<<*:* Faz com que o elemento selecionado na lista **Parâmetros** deixe de ser um parâmetro da Função Adaptativa, o que é indicado pela sua retirada da lista **Parâmetros**.

Listas

Nomes*:* Contém todos os nomes de Bolhas e de Eventos criados para a Função Adaptativa que estiver sendo exibida na Janela de **Montagem de Função Adaptativa**.

Parâmetros*:* Contém os nomes de Bolhas e de Eventos que foram selecionados para serem parâmetros da Função Adaptativa.

Menu

Função Adapt - Criar*:* Habilita a função de criação de uma nova Função Adaptativa, fazendo com que seja aberta a Janela **Nova Função Adaptativa**.

***Função Adapt - Alterar**:* Habilita a função de abertura de uma Função Adaptativa já existente, fazendo com que seja aberta a Janela **Alteração/Exclusão de Funções Adaptativas**.

***Função Adapt - Alterar Nome**:* Habilita a função que permite a alteração de nome de Função Adaptativa, fazendo com que seja aberta a Janela **Salvar Como/Alterar Nome de Função Adaptativa**.

***Função Adapt - Salvar Como**:* Habilita a função que permite salvar uma Função Adaptativa já existente com outro nome, fazendo com que seja aberta a Janela **Salvar Como/Alterar Nome de Função Adaptativa**.

***Função Adapt - Excluir**:* Habilita a função de exclusão de uma Função Adaptativa já existente, fazendo com que seja aberta a Janela **Alteração/Exclusão de Funções Adaptativas**.

***Função Adapt - Voltar**:* Faz com que a Janela **de Montagem de Função Adaptativa** seja fechada, retornado-se para a Janela **Principal**. As edições realizadas nas Funções Adaptativas são armazenadas no Banco de Dados do pelo sistema.

***Bolha - Mover**:* Habilita a função de movimentação de Bolhas dentro da Janela de **Montagem de Função Adaptativa**. Deve-se selecionar a bolha que se deseje mover, conforme detalhado no item A.3, e arrastá-la, com o botão do mouse pressionado, para o local desejado. Eventuais ligações existentes na Bolha serão automaticamente movidas, em conformidade com sua nova posição

***Bolha - Tamanho**:* Habilita a função de alteração de tamanho de Bolhas dentro desta Janela. Deve-se selecionar a bolha cujo tamanho se deseje alterar, conforme detalhado no item A.3, e arrastar suas bordas laterais direita e inferior até que se atinja o tamanho desejado

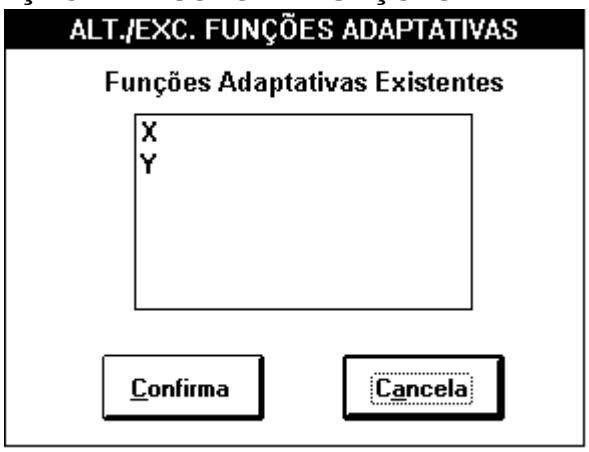
***Ligações - Adaptativa**:* Habilita a função de alteração da Função Adaptativa que estiver sendo utilizada, ou, caso não haja nenhuma Função sendo utilizada, permite a escolha da Função desejada. Ao ser pressionado este botão é aberta a Janela **Seleção de Função Adaptativa**.

Imprimir - Formulário: Habilita a impressão do conteúdo da Janela que estiver sendo exibida, emitindo uma mensagem de pedido de confirmação ou não da região do Statechart a ser impressa.

Bolhas

***4 Bolhas com moldura grossa**:* Estas Bolhas, automaticamente inseridas pelo sistema, têm a finalidade de servir como origem ou destino de Ligações cuja origem ou destino deva ser um parâmetro da Função Adaptativa que se estiver montando. Desta forma, se uma destas Bolhas for origem de uma Ligação, esta aparecerá, na lista **Parâmetros** da Janela de **Seleção de Parâmetros**, como Origem dessa Ligação, e portanto como um parâmetro a ser definido. O mesmo ocorre se uma destas Bolhas for destino de uma Ligação, com a diferença que deverá aparecer como Destino dessa Ligação.

Janela de ALTERAÇÃO/EXCLUSÃO DE FUNÇÕES ADAPTATIVAS



Botões

***Confirma**:* Se a opção selecionada no menu da Janela de **Montagem de Função Adaptativa** foi **Função Adapt - Alterar**, ocorre a abertura da Função Adaptativa especificada na lista **Funções Adaptativas Existentes**. Após ter sido selecionada uma Função e ter sido pressionado este botão, a Função é aberta na Janela de **Montagem de Função Adaptativa** e a Janela de **Alteração/Exclusão de Funções Adaptativas** é fechada. Também é aberta a janela de **Seleção de Elementos de Função Adaptativa**. Se uma Função já estiver sendo exibida na Janela de **Montagem de Função Adaptativa**, tal função será retirada antes de se carregar a Função aqui selecionada.

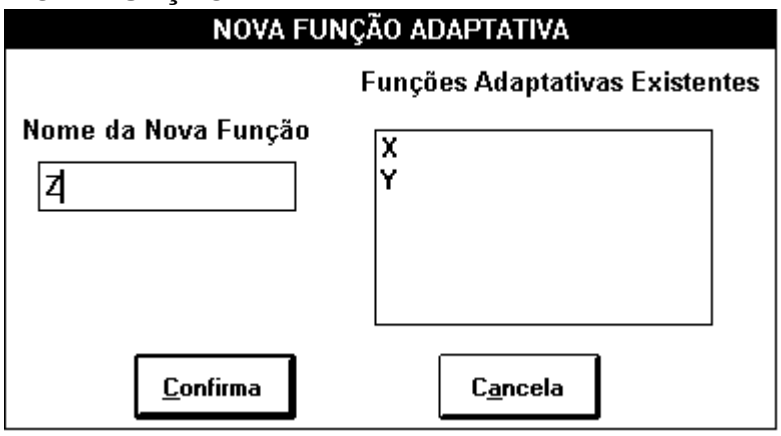
Se a opção selecionada no menu da Janela de **Montagem** foi **Função Adapt - Excluir**, ocorre a exibição de uma mensagem solicitando a confirmação ou não da exclusão da Função Adaptativa selecionada na lista **Funções Adaptativas Existentes**. Se, for confirmada a exclusão da Função, a mesma é retirada completamente das tabelas do Banco de Dados. Caso contrário, a operação de exclusão é cancelada

***Cancela**:* Faz com que a Janela **Alt./Exc. Funções Adaptativas** seja fechada, retornado-se para a Janela de **Montagem de Funções Adaptativas**. Neste caso, nenhuma Função é aberta ou excluída.

Listas

***Funções Adaptativas Existentes**:* Contém o nome de todas as Funções Adaptativas existentes no Projeto que estiver sendo utilizado.

- Janela de NOVA FUNÇÃO ADAPTATIVA



Botões

Confirma*:* Causa a criação da Função especificada no campo **Nome da Nova Função**. Se este campo estiver em branco, uma mensagem é exibida, solicitando-se que seja fornecido o nome desejado para a nova Função. Se o **Nome da Nova Função** fornecido coincidir com o de uma das Funções já existentes, uma mensagem é exibida, informando que o nome de Função pretendido já é utilizado. Após ter sido selecionado um nome para a Função e ter sido pressionado este botão, tal Função é aberta na Janela de **Montagem de Funções Adaptativas** (porém sem nenhum elemento), e a Janela **Nova Função Adaptativa** é fechada. Também é aberta a janela de **Seleção de Elementos de Função Adaptativa**. Se uma Função já estiver sendo exibida na Janela de Montagem, esta é fechada antes de se iniciar a criação da Função aqui selecionada.

Cancela*:* Faz com que a Janela **Nova Função Adaptativa** seja fechada, retornado-se para a Janela de **Montagem de Funções Adaptativas**. Neste caso, nenhuma Função Adaptativa nova é criada.

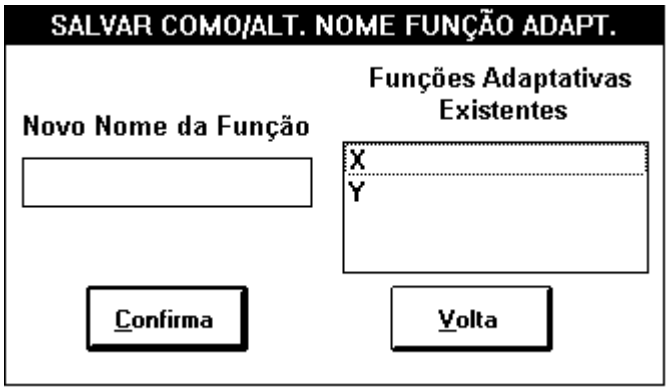
Listas

Funções Adaptativas Existentes*:* Contém o nome de todas as Funções Adaptativas existentes no Projeto que estiver sendo utilizado.

Campos

Nome da Nova Função*:* Deve ser preenchido com o nome desejado para a nova Função Adaptativa.

Janela SALVAR COMO/ALTERAR NOME DE FUNÇÃO ADAPTATIVA



Botões

***Confirma**:* Se a opção selecionada no menu da Janela de **Montagem de Funções Adaptativas** foi **Função Adapt - Alterar Nome**, ocorre a alteração de nome da Função especificada na lista **Funções Adaptativas Existentes** para o nome indicado no campo **Novo Nome da Função**.

Se a opção selecionada no menu da Janela de **Montagem de Funções Adaptativas** foi **Função Adapt - Salvar Como**, ocorre a inclusão, na Base de Dados, de uma Função exatamente igual à especificada na lista **Funções Adaptativas Existentes**, porém com o nome alterado para o nome indicado no campo **Novo Nome da Função**.

Nos dois casos acima descritos, se não for selecionada nenhuma Função da lista **Funções Adaptativas Existentes**, uma mensagem é exibida, solicitando-se que seja escolhida uma das Funções. Se o **Novo Nome** fornecido para a função coincidir com o de uma das Funções já existentes, uma mensagem será exibida, informando que o nome de Função pretendido já é utilizado

***Volta**:* Faz com que esta Janela seja fechada, retornado-se para a Janela de **Montagem de Funções Adaptativas**. Neste caso, não ocorre nem a alteração de nome nem a inclusão de uma nova Função.

Listas

***Funções Adaptativas Existentes**:* Contém o nome de todas as Funções Adaptativas existentes no Projeto que estiver sendo utilizado.

Campos

***Novo Nome da Função**:* Deve ser preenchido com o novo nome pretendido para a Função Adaptativa, tanto no caso de mudança de nome, quanto no caso de se salvar a Função Adaptativa com outro nome.

- Janela de SELEÇÃO DE ELEMENTOS DE FUNÇÃO ADAPTATIVA

Botões

***OK**:* Permite a inclusão, na Função Adaptativa que estiver sendo montada, do elemento selecionado da lista **Tipo de Elemento**. Após ter sido selecionado um elemento desta lista e ter sido pressionado este botão, a Janela de **Seleção de Elementos de Função Adaptativa** é fechada, e, dependendo do **Tipo de Elemento** selecionado é efetuada a ação correspondente, conforme a seguir descrito:

- **Ligação**: é exibida uma mensagem solicitando que se acione o botão do mouse sobre a Janela **Montagem de Funções Adaptativas**, para que a inclusão deste elemento seja confirmada. Se for digitada a tecla "Esc", a inclusão não é realizada.

+ **Bolha OR**: Habilita a criação de uma Bolha do tipo OR, através da abertura da Janela **Criar Bolha OR**.

+ **Bolha AND**: Habilita a criação de uma Bolha do tipo AND, através da abertura da Janela **Criar Bolha AND.**

+ **Ligação Dependente Interna**: Habilita a criação de uma Ligação Dependente Interna. O modo de realização dessa ligação encontra-se descrita no item A.3.

+ **Ligação Dependente Externa Entrada**: Habilita a criação de uma Ligação Dependente Externa de Entrada. A forma de realização dessa ligação está descrito no item A.3.

+ **Ligação Dependente Externa Saída**: Habilita a criação de uma Ligação Dependente Externa de Saída. A maneira de criar esta ligação está descrita no item A.3.

+ **Ligação Independente Interna**: Habilita a criação de uma Ligação Independente Interna. O modo de realização dessa ligação encontra-se no item A.3.

+ **Ligação Independente Externa Entrada**: Habilita a criação de uma Ligação Independente Externa de Entrada. O modo de realização dessa ligação é encontrado no item A.3.

+ **Ligação Independente Externa Saída**: Habilita a criação de uma Ligação Independente Externa de Saída. A forma de realização dessa ligação está indicada no item A.3.

***Exclui**:* Realiza a exclusão do elemento criado na Função Adaptativa e selecionado na lista **Elementos**. O elemento excluído é removido tanto do Banco de Dados, quanto da Janela de **Montagem de Funções Adaptativas**.

***Volta**:* Faz com que a Janela de **Seleção de Elementos de Função Adaptativa** seja fechada, retornado-se para a Janela de **Montagem de Funções Adaptativas**. As edições realizadas na Função Adaptativa que estiver sendo montada são armazenadas no Banco de Dados.

Listas

***Tipo de Elemento**:* Contém os tipos de Elementos possíveis de serem criados:

- **Ligação**: quando executado, em uma Ação Adaptativa, realiza a eliminação de uma Ligação especificada na Janela de **Seleção de Parâmetros**.

+ **Bolha OR**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Bolha do tipo OR especificada na Janela de **Seleção de Parâmetros**.

+ **Bolha AND**: quando executado, em uma Ação Adaptativa, efetua a criação de uma Bolha do tipo AND especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Dependente Interna**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Dependente Interna especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Dependente Externa Entrada**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Dependente Externa de Entrada especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Dependente Externa Saída**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Dependente Externa de Saída especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Independente Interna**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Independente Interna especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Independente Externa Entrada**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Independente Externa de Entrada especificada na Janela de **Seleção de Parâmetros**.

+ **Ligação Independente Externa Saída**: quando executado, em uma Ação Adaptativa, realiza a criação de uma Ligação Independente Externa de Saída especificada na Janela de **Seleção de Parâmetros**.

***Elementos**:* Contém a relação dos elementos criados na Função Adaptativa, na ordem de sua criação.

A.3. Comandos Auxiliares

Neste item é descrito o modo de operação de alguns dos comandos auxiliares disponíveis no STAD.

a) Seleção de Bolha: Para se selecionar uma bolha deve-se acionar o mouse na área mais externa da bolha desejada, entre o texto com o nome da bolha e a sua borda externa.

b) Seleção de Ligação: Para se selecionar uma ligação deve-se acionar o mouse em qualquer extremidade de algum segmento que compõe a representação da ligação que se deseje selecionar.

c) Criação de Bolha: Para se criar uma bolha deve-se, após selecionar o item adequado no menu, selecionar com o mouse a posição em que se deseja que a nova Bolha seja inserida. Ao ser pressionado o botão de acionamento do mouse, será exibida a moldura da nova Bolha, cuja posição pode ser escolhida pelo usuário arrastando-se o indicador do mouse até o local desejado, onde então deverá ser liberado este botão. O nome da Bolha será exibido na parte central de sua representação.

d) Criação de Ligação: Para se criar uma ligação deve-se, após acionar o item correspondente do menu, selecionar, com o mouse, a região do contorno da Bolha que se deseja seja a origem da ligação, e mantendo-se pressionado o botão de acionamento do mouse, arrastar seu ponteiro até a posição de destino do segmento de reta, que vai sendo traçado conforme o ponteiro do mouse vai sendo deslocado na tela. A posição da origem da ligação com relação à borda da Bolha é acertada assim que o botão do mouse é liberado. Se o local da tela onde o botão do mouse deixou de ser pressionado for uma região onde não exista Bolha, a mesma operação de pressionar o botão do mouse e arrastá-lo até o local pretendido deve ser repetida, até que o botão do mouse seja liberado na área externa de uma Bolha que se deseja seja o destino da ligação. A posição do destino da ligação com relação à borda da Bolha é automaticamente acertada assim que o botão do mouse é liberado. Neste caso, é exibida uma mensagem para que seja confirmada ou não a ligação. Se a ligação não for confirmada, todos os segmentos de reta que a compõem são apagados. Caso a ligação seja confirmada, é aberta a **Janela de Dados da Ligação**.

Se a ligação for do tipo dependente, os segmentos de reta traçados são representados com linhas sólidas.

Se a ligação for do tipo independente, os segmentos de reta traçados são representados em linhas tracejadas. Se corresponder a uma ligação de entrada, a origem do primeiro segmento de reta traçado deve ser uma região da tela onde não existam bolhas. Se se referir a uma ligação de saída, o destino do último segmento de reta traçado deve ser uma região da tela onde não existam bolhas.

ANEXO B - ESTRUTURA DAS TABELAS DO BANCO DE DADOS

Neste anexo é descrita a estrutura das tabelas empregadas na montagem do Banco de Dados utilizado para armazenamento das informações relativas a cada projeto.

B1. Campos da Tabela Bolhas

Projeto: tipo texto, com 20 caracteres, contém o nome do Projeto que estiver sendo detalhado.

Nivel: tipo numérico inteiro, indica o nível de detalhamento do Projeto que estiver sendo detalhado (nível mais global = 0, níveis inferiores = 1, 2, 3, ...).

Pai: tipo texto, com 20 caracteres, contém o nome da Bolha ancestral do Statechart que estiver sendo detalhado (na bolha mais global este campo não é preenchido).

Nome: tipo texto, com 20 caracteres, contém o nome da Bolha detalhada no registro.

Numero: tipo numérico inteiro, contém um número de referência para a Bolha detalhada no registro.

Filha: tipo texto, com 20 caracteres, indica se a Bolha detalhada no registro possui ou não Statecharts de detalhamento.

Cima: tipo numérico inteiro, contém a abcissa do canto superior esquerdo da Bolha, com relação à origem da janela, ou seja, também em relação ao canto superior esquerdo da janela em que a Bolha estiver sendo exibida.

Esquerda: tipo numérico inteiro, contém a ordenada do canto superior esquerdo da Bolha, com relação à origem da janela, ou seja, também em relação ao canto superior esquerdo da janela em que a Bolha estiver sendo exibida.

Largura: tipo numérico inteiro, contém a largura da Bolha detalhada no registro.

Altura: tipo numérico inteiro, contém a altura da Bolha detalhada no registro.

Tipo: tipo texto, com 1 caractere, contém o tipo de Bolha detalhada no registro ("A" = AND, "D" = Default, "H" = History ou "C" = Comum).

Estado: tipo texto, com 10 caracteres, indica o estado em que se encontra a Bolha durante uma execução do Statechart (Ativo ou Inativo).

History: tipo numérico inteiro, quando igual a "1" indica que a Bolha em questão estava ativa no instante em que o Statechart deixou de ser executado. Quando igual a "0", significa que a Bolha em questão estava inativa nesse instante.

Executado: tipo texto, com 5 caracteres, só é utilizado no caso de Ações Adaptativas, e indica, quando igual a "SIM", que a Bolha em questão já foi inserida ou removida do Statechart correspondente, caso contrário, quando igual a "NAO", que tal Bolha ainda não foi inserida ou removida do Statechart.

Adaptativo: tipo texto, com 10 caracteres, indica, quando igual a "SIM", que a Bolha faz parte de uma Função Adaptativa; quando igual a "DUP" que a Bolha não faz parte de uma Ação Adaptativa, quando igual a "PROV", que a Bolha em questão foi gerada por intermédio da execução de uma Ação Adaptativa, e quando igual a "NAO" que a Bolha não faz parte de nenhuma Ação ou Função Adaptativa.

Metanome: tipo texto, com 20 caracteres, contém o metanome da Bolha detalhada no registro. Este campo tem utilidade quando é alterado o nome de alguma Bolha em uma Ação Adaptativa.

Nome_Funcao: tipo texto, com 20 caracteres, quando a Bolha está montada em uma Função Adaptativa indica o nome da Função que está criando a Bolha; quando a Bolha está montada fora de uma Função Adaptativa indica o número da Ligação que está acionando a Função.

Param1: tipo texto, com 20 caracteres, indica qual é o parâmetro X (Ligação que representa o parâmetro X) de uma Bolha Primitiva.

Param2: tipo texto, com 20 caracteres, indica qual é o parâmetro Y (Ligação que representa o parâmetro Y) de uma Bolha Primitiva.

Nome_Aux: tipo texto, com 20 caracteres, utilizada no caso de uma Ligação de uma Função Adaptativa acionar outra Função Adaptativa, sendo preenchido com o nome da Função acionada.

B2. Campos da Tabela Ligações

Projeto: tipo texto, com 20 caracteres, contém o nome do Projeto que estiver sendo detalhado.

Nivel: tipo numérico inteiro, indica o nível de detalhamento do Projeto que estiver sendo detalhado (nível mais global = 0, níveis inferiores = 1, 2, 3, ...).

Pai: tipo texto, com 20 caracteres, contém o nome da Bolha ancestral do Statechart que estiver sendo detalhado (na bolha mais global este campo não é preenchido).

Primeiro: tipo texto, com 1 caractere, indica, quando igual a "P" que o segmento que estiver sendo detalhado é o primeiro segmento da Ligação, caso contrário, quando não preenchido, indica que o segmento não é o primeiro da ligação.

Ultimo: tipo texto, com 1 caractere, indica, quando igual a "U" que o segmento que estiver sendo detalhado é o último segmento da Ligação, caso contrário, quando não preenchido, indica que o segmento não é o último da ligação.

Origem: tipo numérico inteiro, contém o número de referência da Bolha-origem da Ligação que estiver sendo detalhada.

Destino: tipo numérico inteiro, contém o número de referência da Bolha-destino da Ligação que estiver sendo detalhada.

Num: tipo numérico inteiro, contém um número de referência para a Bolha detalhada no registro.

X1: tipo numérico inteiro, contém a abcissa da origem do Segmento detalhado no registro, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que o segmento estiver sendo exibido.

Y1: tipo numérico inteiro, contém a ordenada da origem do Segmento detalhado no registro, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que o segmento estiver sendo exibido.

X2: tipo numérico inteiro, contém a abcissa do destino do Segmento detalhado no registro, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que o segmento estiver sendo exibido.

Y2: tipo numérico inteiro, contém a ordenada do destino do Segmento detalhado no registro, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que o segmento estiver sendo exibido.

Tipo: tipo texto, com 3 caracteres, indica o tipo da Ligação: "I" (interna), "EE" (externa de entrada), "ES" (externa de saída), "II" (interna independente), "EEI" (externa de entrada independente) ou "ESI" (externa de saída independente).

Evento: tipo texto, com 20 caracteres, contém o nome do evento da Ligação que estiver sendo detalhada no registro.

Condicao: tipo texto, com 20 caracteres, contém o nome da condição (nome de uma Bolha) a ser respeitada para que possa ocorrer a transição indicada pela Ligação detalhada no registro.

Disparo: tipo texto, com 20 caracteres, contém o nome do disparo a ser realizado (nome de um Evento) quando da ocorrência da transição indicada pela Ligação detalhada no registro.

Posicao: tipo texto, com 1 caractere, indica a posição da Bolha em que o primeiro e o último segmentos que representam uma Ligação se conectam à Bolha de origem e destino, respectivamente. A posição é representada pelas letras "E", "D", "S" ou "I" , indicando se a conexão é efetuada à esquerda, à direita, na parte superior ou na parte inferior da Bolha, respectivamente.

Cima: tipo numérico inteiro, indica a abcissa do canto superior esquerdo do texto que contém as informações (evento, condição, disparo, atraso e tempo de disparo) sobre a Ligação detalhada no registro, com relação à origem da janela do texto exibido, ou seja, em relação ao seu canto superior esquerdo.

Esquerda: tipo numérico inteiro, representa a ordenada do canto superior esquerdo do texto que contém as informações (evento, condição, disparo, atraso e tempo de disparo) sobre a Ligação detalhada no registro, com relação à origem, ou seja, ao canto superior esquerdo da janela em que o texto estiver sendo exibido.

Estado: tipo texto, com 10 caracteres, indica o estado em que se encontra a Ligação durante uma execução do Statechart (Ativo ou Inativo).

Nome_Origem: tipo texto, com 20 caracteres, contém o nome da Bolha-origem da Ligação que estiver sendo detalhada.

Nome_Destino: tipo texto, com 20 caracteres, contém o nome da Bolha-destino da Ligação que estiver sendo detalhada.

Executado: tipo texto, com 5 caracteres, só é utilizado no caso de Ações Adaptativas, e indica quando igual a "SIM" que a Ligação em questão já foi inserida ou removida do Statechart correspondente, caso contrário, quando igual a "NAO", indica que a Ligação ainda não foi inserida ou removida do Statechart.

Adaptativo: tipo texto, com 10 caracteres, indica, quando igual a "SIM", que a Ligação faz parte de uma Função Adaptativa; quando igual a "DUP" que a Ligação não faz parte de uma Ação Adaptativa, quando igual a "PROV", que a Ligação em questão foi gerada por intermédio da execução de uma Ação Adaptativa, e quando igual a "NAO" que a Ligação não faz parte de nenhuma Ação ou Função Adaptativa.

Atraso: tipo numérico inteiro, contém o atraso que deve ocorrer para que a transição para a Bolha-destino seja efetuada. A unidade de tempo utilizada é o número de passos de simulação executados.

T_Disparo: tipo numérico inteiro, contém o tempo mínimo absoluto (que deve decorrer a partir do início da execução do Statechart), para que a transição para a Bolha-destino seja efetuada. A unidade de tempo utilizada é o número de passos de simulação executados.

T_Restante: tipo numérico inteiro, contém o tempo que deve ainda transcorrer (levando-se em conta o atraso e o tempo de disparo) para que a transição para a Bolha-destino seja efetuada. A unidade de tempo utilizada é o número de passos de simulação executados.

Nome_Funcao: tipo texto, com 20 caracteres. Quando a Ligação está montada em uma Função Adaptativa indica o nome da Função que está criando a Ligação; quando a Ligação está montada fora de uma Função Adaptativa indica o número da Ligação que está acionando a Função.

Metanome: tipo texto, com 20 caracteres, contém o metanome do evento associado à Ligação que estiver sendo detalhada no registro do Banco de Dados. Este campo tem utilidade quando é alterado o nome de algum evento de Ligação em uma Ação Adaptativa.

Funcao_Usada: tipo texto, com 20 caracteres, campo auxiliar que tem a mesma função do campo ***Nome_Funcao*.**

Execucao: tipo texto, com 10 caracteres, indica se a execução da Ação Adaptativa deve ser feita anterior ("Anterior") ou posteriormente ("Posterior") à transição apontada pela Ligação.

Valor: tipo texto, com 20 caracteres, contém o valor assumido pela Ligação, conforme o Statechart vai sendo executado.

Nome_Aux: tipo texto, com 20 caracteres, utilizada no caso de uma Ligação pertencente a Função Adaptativa estar associada ao acionamento de outra Função Adaptativa, sendo preenchido com o nome da Função acionada.

B3. Campos da Tabela Setas

Projeto: tipo texto, com 20 caracteres, contém o nome do Projeto em uso.

Nivel: tipo numérico inteiro, indica o nível de detalhamento do Projeto em uso (nível mais global = 0, níveis inferiores = 1, 2, 3, ...).

Pai: tipo texto, com 20 caracteres, contém o nome da Bolha ancestral do Statechart que estiver sendo detalhado (na bolha mais global este campo não é preenchido).

Numero: tipo numérico inteiro, contém um número de referência para a Ligação que estiver sendo detalhada.

Esquerda: tipo numérico inteiro, contém a abcissa do canto superior esquerdo da seta que estiver sendo detalhada, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que a seta estiver sendo exibida.

Cima: tipo numérico inteiro, contém a ordenada do canto superior esquerdo da seta que estiver sendo detalhada, com relação à origem da janela, ou seja, em relação ao canto superior esquerdo da janela em que a seta estiver sendo exibida.

Adaptativo: tipo texto, com 10 caracteres, indica, quando igual a "SIM", que a Seta faz parte de uma Função Adaptativa; quando igual a "DUP" que a Seta não faz parte de uma Ação Adaptativa, quando igual a "PROV", que a Seta em questão foi gerada por intermédio da execução de uma Ação Adaptativa, e quando igual a "NAO" que a Seta não faz parte de nenhuma Ação ou Função Adaptativa.

Executado: tipo texto, com 10 caracteres, só é utilizado no caso de Ações Adaptativas, e indica quando igual a "SIM", que a Seta em questão já foi inserida ou removida do Statechart correspondente; caso contrário, quando igual a "NAO", que tal Seta ainda não foi inserida ou removida do Statechart.

Nome_Funcao: tipo texto, com 20 caracteres. Quando a Seta está sendo utilizada na montagem de uma Função Adaptativa, indica o nome da Função que está criando a Seta; quando a Seta está sendo utilizada fora de uma Função Adaptativa, indica o número da Ligação que está promovendo a execução de uma Ação Adaptativa.

Metanome: tipo texto, com 20 caracteres, contém o metanome do evento da Ligação da qual a Seta que estiver sendo detalhada no faz parte.

Nome_Aux: tipo texto, com 20 caracteres, utilizada no caso de uma transição em uma Função Adaptativa promover a execução de uma Ação Adaptativa. Portanto, este campo contém o nome desta Ação Adaptativa.

B4. Campos da Tabela Adaptativa

Projeto: tipo texto, com 20 caracteres, contém o nome do Projeto que estiver sendo detalhado.

Nivel: tipo numérico inteiro, contém o nível de detalhamento do Projeto que estiver sendo detalhado (nível mais global = 0, níveis inferiores = 1, 2, 3, ...).

Pai: tipo texto, com 20 caracteres, contém o nome da Bolha ancestral do Statechart que estiver sendo detalhado (na bolha mais global este campo não é preenchido).

Nome: tipo texto, com 20 caracteres, contém o nome da Função Adaptativa que estiver sendo detalhada; no caso de uma Função acionar outra Função, é preenchido com "@@@".

Nome_Aux: tipo texto, com 20 caracteres, utilizada no caso de uma Ligação de uma Função Adaptativa acionar outra Função Adaptativa, sendo preenchido com o nome da Função acionada.

Nome_Geral: tipo texto, com 20 caracteres, contém o nome da Função Adaptativa que estiver sendo detalhada.

Adaptativo: tipo texto, com 5 caracteres, indica, quando igual a "SIM", que o registro do Banco de Dados é parte do detalhamento de uma Função Adaptativa; caso contrário, quando igual a "NAO", que foi resultado da execução de uma Ação Adaptativa, ou seja, que o registro foi criado a partir do acionamento de uma Função Adaptativa.

Tipo: tipo texto, com 5 caracteres, indica qual o tipo de ação adaptativa a ser realizada pela Função Adaptativa, ou seja, se é uma ação de inclusão de um elemento ("+") ou de exclusão de elemento ("-").

Elemento: tipo texto, com 10 caracteres, indica o elemento que estiver sendo incluído (Bolha ou Ligação) ou removido (Ligação) por meio da Função Adaptativa.

Meta_Nomevento: tipo texto, com 20 caracteres, contém o metanome do elemento que estiver sendo incluído ou removido por meio da Função Adaptativa.

Origem: tipo texto, com 20 caracteres, contém o nome da Bolha-origem da Ligação que estiver sendo definida na Função Adaptativa.

Destino: tipo texto, com 20 caracteres, contém o nome da Bolha-destino da Ligação que estiver sendo definida na Função Adaptativa.

Num: tipo numérico inteiro, indica o número da Ligação, ou seja, da transição que estiver promovendo a execução de uma Ação Adaptativa; no caso de uma transição de uma Função promover o a execução de outra Ação Adaptativa, este campo é preenchido com o número da Ligação que efetuou o primeira acionamento de uma Função.

Ligacao: tipo numérico inteiro, indica o número da Ligação, ou seja, da transição que estiver promovendo a execução de um Ação Adaptativa

Execucao: tipo texto, com 10 caracteres, indica se a execução da Ação Adaptativa deve ser feita anterior ("Anterior") ou posteriormente ("Posterior") à da transição apontada pela Ligação.

P1: tipo texto, com 5 caracteres, indica se o nome da Bolha ou da Ligação definida em uma Função Adaptativa é ou não um parâmetro passível de alteração quando utilizado em uma Ação Adaptativa.

P2: tipo texto, com 5 caracteres, indica se a origem de uma Ligação definida em uma Função Adaptativa é ou não um parâmetro passível de alteração quando utilizada em uma Ação Adaptativa.

P3: tipo texto, com 5 caracteres, indica se o destino de uma Ligação definida em uma Função Adaptativa é ou não um parâmetro passível de alteração quando utilizada em uma Ação Adaptativa.