UTILIZAÇÃO DA UML PARA ESPECIFICAÇÃO DE SISTEMAS DE TEMPO REAL

Airton Zancanaro

Evandro de Souza

Fabrício Jailson Barth

Universidade Regional de Blumenau – FURB
Rua Antônio da Veiga, 140 – Caixa Postal 1507
CEP 89010-971 Blumenau – SC
{airton, eds, fjbarth}@inf.furb.rct-sc.br

RESUMO

Este artigo tem como finalidade descrever uma metodologia capaz de especificar Sistemas de Tempo Real, que tem como requisito principal o controle do tempo. Algumas abordagens são descritas como o Diagrama de Fluxo de Dados, Redes de Petri e a Orientação a Objetos. Para este artigo foi aprofundado a Unified Modeling Language (UML), que possui características adequadas para especificar sistemas complexos. Por fim realizou-se um estudo de caso para exemplificar e testar a metodologia.

PALAVRAS-CHAVE: Unified Modeling Language (UML), Sistemas de Tempo Real

1. INTRODUÇÃO

A computação tempo real é uma das áreas mais desafiadoras e de maior demanda tecnológica da atualidade, estando diretamente ligada à aplicações que envolvem índices críticos de confiabilidade e segurança.

Sistemas de Tempo Real (STR) são sistemas encontrados normalmente em situações onde o risco é constante, podendo resultar em conseqüências negativas. Como por exemplo: sistema de automação industrial, sistema hospitalar, controle de usinas nucleares e outros.

Na especificação de STR alguns fatores que vem dificultando sua difusão é a complexidade e os riscos envolvidos. Grande parte desta complexidade está no desconhecimento de técnicas de modelagem.

Infelizmente, métodos convencionais de engenharia de software mostram-se ineficientes no desenvolvimento de STR, uma vez que estes, além de não considerarem requisitos temporais, usualmente não abordam o projeto de hardware e software de forma integrada.

Atualmente as metodologias influenciadas pelo paradigma de orientação a objetos têm recebido um maior destaque, justamente por possibilitarem de forma mais adequada o tratamento de sistemas complexos. Apesar do paradigma de orientação a objetos ter se tornado popular no desenvolvimento de sistemas computacionais sem requisitos temporais, experiências em estudos de caso relatadas em [PER94], demonstram que o paradigma de orientação a objetos é também uma ótima alternativa para a modelagem de STR, permitindo obter resultados superiores àqueles obtidos com métodos convencionais.

Este artigo visa descrever a modelagem de STR e uma ferramenta CASE para o desenvolvimento destes sistemas. Inicialmente são apresentadas definições de tempo real, relatando o seu funcionamento e classificação. Posteriormente algumas metodologias de especificação de sistemas são descritas e na seqüência são apontadas características da UML e sua adequação a STR. Por fim é utilizada a ferramenta CASE Rational Rose 98 para especificar um exemplo prático de automação de elevadores.

2. SISTEMAS DE TEMPO REAL

Um sistema computacional é classificado como tempo real quando seu correto funcionamento não depende apenas do processamento de sinais de entrada ou estímulos e sua transformação em sinais ou variáveis de saída, mas especialmente da satisfação de requisitos temporais [STA88, HAL91].

Segundo [NET99], os STR podem ser classificados de acordo com as restrições de tempo em:

Hard Real-Time: São aqueles sistemas onde é absolutamente imperativo que a resposta ocorra dentro de um limite de tempo especificado (*deadline*).

Soft Real-Time: São aqueles onde o tempo de resposta é importante, porém o sistema ainda funciona corretamente de maneira degradada se o *deadline* for ultrapassado.

Firm Real-Time: São aqueles onde o *deadline* pode ser perdido até um determinado atraso, onde o sistema ainda funciona de maneira correta e/ou degradada. Após este limite, os resultados perdem seu valor.

Além das restrições de tempo, outras restrições podem estar associadas às tarefas:

- Recursos: A tarefa pode necessitar de outros recursos além da CPU tais como: dispositivos I/O, estruturas de dados, arquivos, base de dados, etc;
- Precedência: Uma tarefa complexa, que acessa vários recursos é melhor manipulada através do seu particionamento em múltiplas sub-tarefas, relacionadas por restrições de precedência e cada uma delas requisitando um sub-conjunto dos recursos;
- Concorrência: Tarefas podem concorrer pelo acesso aos recursos desde que a consistência destes não seje violada;
- Comunicação: Tarefas cooperantes farão parte dos sistemas distribuídos. A tarefa semântica da comunicação irá variar, assim como, a estrutura de interconexão entre as tarefas comunicantes e os seus requisitos de tempo;
- Alocação de Tarefas: Quando múltiplas instâncias de uma tarefa são executadas para fins de tolerância a falhas, as diferentes instâncias deverão executar em processadores diferentes.

3. TÉCNICAS DE ESPECIFICAÇÃO DE SISTEMAS

Atualmente existem várias propostas para especificação de sistemas. Destas, algumas permitem que se modelem sistemas com o requisito tempo. Pode-se destacar as Redes de Petri, a Orientação a Objetos, Diagrama de Transição de Estados e o Diagrama de Fluxo de Dados que são formas bastante conhecidas de técnicas de especificação.

3.1 DIAGRAMA DE FLUXO DE DADOS

O Diagrama de Fluxo de Dados (DFD) é a ferramenta principal para descrever os requisitos funcionais. Ele segmenta em processos e os representa como uma rede interligada pelos fluxos de dados [HAT88].

O DFD consiste em estados e transições, como visto na figura 01, e especifica a lógica associando os fluxos de entrada e saída com as ações apropriadas.

Embora o DFD possa ser utilizado na especificação de STR ele não foi desenvolvido para este fim, não favorecendo então a especificação de sistemas em tempo real complexos.

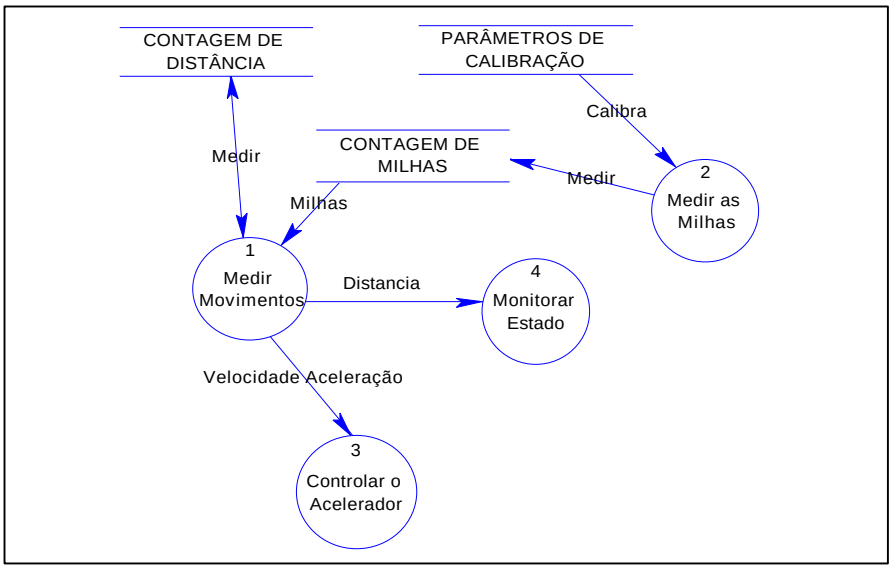

Figura 01: Exemplo de Diagrama de Fluxo de Dados [HAT88]

3.2 REDES DE PETRI

A rede de Petri é uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes [CAR98]. Na figura 02 tem-se um exemplo de Redes de Petri.

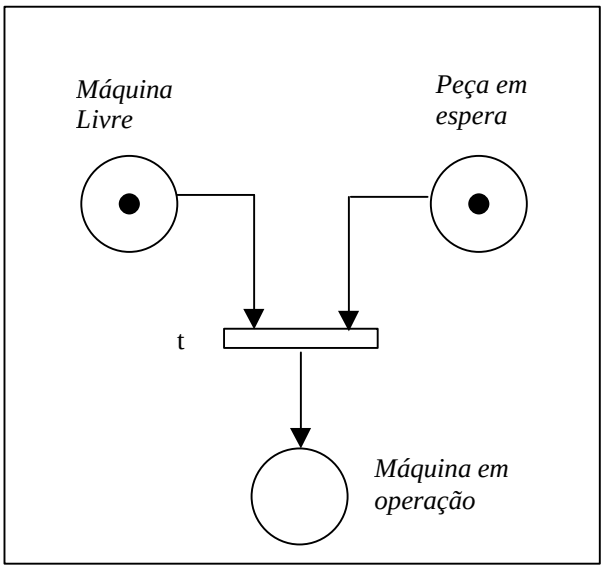

Figura 02: Exemplo de Redes de Petri [CAR98]

Entre as aplicações de rede de Petri pode-se citar: avaliação de desempenho, análise e verificação formal em sistemas discretos, protocolos de comunicação, controle de oficinas de fabricação, *concepção de software tempo real e/ou distribuído*, sistemas de informação

(organização de empresas), sistemas de transporte, logística, gerenciamento de base de dados, interface homem-máquina e multimídia.

3.3 ORIENTAÇÃO A OBJETOS

A Orientação a Objetos tem como objetivo representar o mundo real através de objetos. Esses objetos podem ser de vários tipo como por exemplo, entidades físicas (por exemplo, aviões, robôs) ou abstratas (por exemplo, listas, pilhas, filas). A característica mais importante (e diferente) da abordagem orientada a objetos para desenvolvimento de software é a unificação, através do conceito de objetos, de dois elementos que tradicionalmente tem sido considerados separadamente em paradigmas de programação tradicional: Dados e Funções [BUZ98].

Na figura 03 pode-se visualizar um diagrama de classes que é parte do processo de modelagem orientado a objetos.

Atualmente, o paradigma da orientação a objetos tem sido bastante utilizado no desenvolvimento de sistemas industriais de tempo real distribuído [PER94], uma vez que objetos permitem estruturar a informação de uma forma lógica, aumentando a tratamento de sistemas complexos.

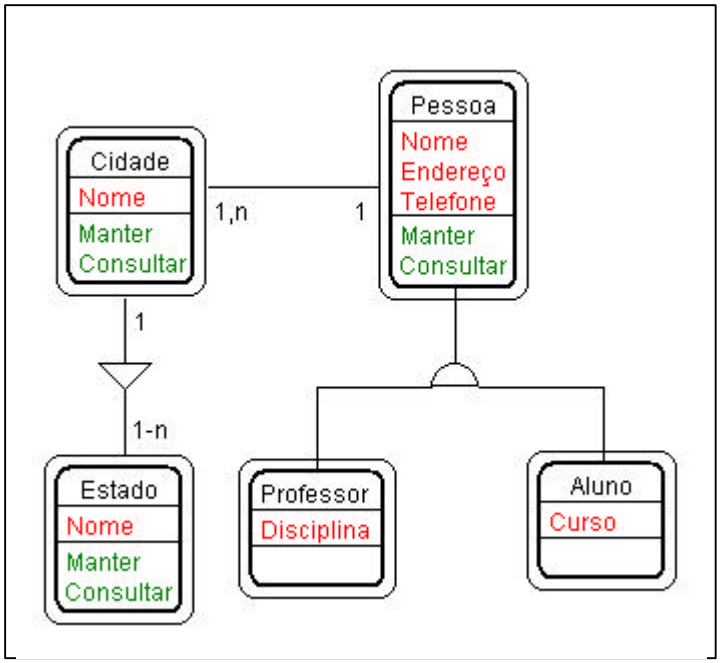

Figura 03: Exemplo de Diagrama de Classes

Uma linguagem para modelagem orientada a objetos que vem se destacando é a Unified Modeling Language (UML) devido ao fato de ser uma tentativa de unificar todas as metodologias existentes para especificação de sistemas orientados a objetos.

4. CARACTERÍSTICAS DA UNIFIED MODELING LANGUAGE (UML)

A UML é uma linguagem padrão para especificar, visualizar, documentar e construir artefatos de um sistema e pode ser utilizada com todos os processos ao longo do ciclo de desenvolvimento e através de diferentes tecnologias de implementação [FUR98].

A notação da UML é uma combinação de sintaxe vinda de várias fontes, mas com vários símbolos removidos das técnicas anteriores e com alguns símbolos novos agregados para tratar situações específicas.

O uso de objetos como elementos encapsulados de informações (dados e funções), aliados à possibilidade de abstrair-se objetos com estrutura e comportamento semelhantes na forma de classes, além do uso de conceitos como herança, instanciamento, agregação, etc., constituem uma poderosa ferramenta de modelagem para sistemas complexos pois trazem ao desenvolvedor vantagens como reutilização, facilidade de manutenção e compreensão, modularidade, concorrência e melhor gerenciamento da complexidade. Portanto a UML é usada no desenvolvimento dos mais diversos tipos de sistemas.

Ela abrange sempre qualquer característica de um sistema em um de seus diagramas e é também aplicada em diferentes fases do desenvolvimento de um sistema, desde a especificação da análise de requisitos até a finalização com a fase de testes[BAR98].

O objetivo da UML é descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos. Porém no estado atual da linguagem de modelagem UML não é possível implementar STR com perfeição, a não ser que seja realizado algumas modificações na sua metodologia.

A UML é composta basicamente pela análise: definição do sistema, descrição abrangente tendo como objetivo delimitar o sistema, diagrama de contexto, descrição de uses cases, esboço das telas, análise de verbos e substantivos e diagrama de classes; e projeto: object interaction diagram, CRC – Classes Responsabilidades Colaboradores e use case design.

Com a intenção de verificar a funcionalidade da UML estendida a Sistemas de Tempo Real, passou-se a modelagem e desenvolvimento de uma aplicação em que fosse exigido um

certo controle sobre os requisitos temporais. O objetivo desse estudo de caso é ser um exemplo prático e didático para a utilização desta metodologia.

5. ESTUDO DE CASO

Esta seção demonstra como utilizar UML para a especificação de um Sistema de Tempo Real utilizando a ferramenta CASE Rational Rose 98, devido a mesma permitir a modelagem utilizando UML e Diagrama de Transição de Estados .

Foi escolhido um problema relacionado ao funcionamento de elevadores em um edifício. Este problema trata da escolha do elevador que deva atender o chamado em menor tempo. Após a solicitação de um elevador, aquele que estiver em situação mais apropriada deverá atender o chamado, abrir a porta, fechar a porta e deslocar-se até o andar solicitado em determinado tempo.

Para a definição do sistema foram realizadas as seguintes etapas: Diagrama de Use-Case, Diagrama de Classes, Diagrama de Interação de Objetos e Diagrama de Transição de Estados.

5.1 DIAGRAMA DE USE-CASES:

O Diagrama Use-Case é uma técnica usada para descrever e definir os requisitos funcionais de um sistema [BAR98].

1. *Acionar elevador*: O ator posicionado no andar aciona o botão do elevador para cima ou para baixo;
2. *Escolher andar*: Dentro do elevador o ator aciona o botão com o respectivo número do andar que deseja alcançar;
3. *Fechar a porta*: Após o autor escolher o andar a porta do elevador deverá ser fechada;
4. *Abrir a porta*: O elevador deverá abrir a porta após a chegada do mesmo no andar indicado;
5. *Escolher elevador*: Após o autor acionar o elevador o andar escolhe qual dos elevadores atende melhor o chamado, levando em consideração o tempo de deslocamento do elevador até o andar.

A figura 04 representa o diagrama de Use-Case dos casos descritos anteriormente.

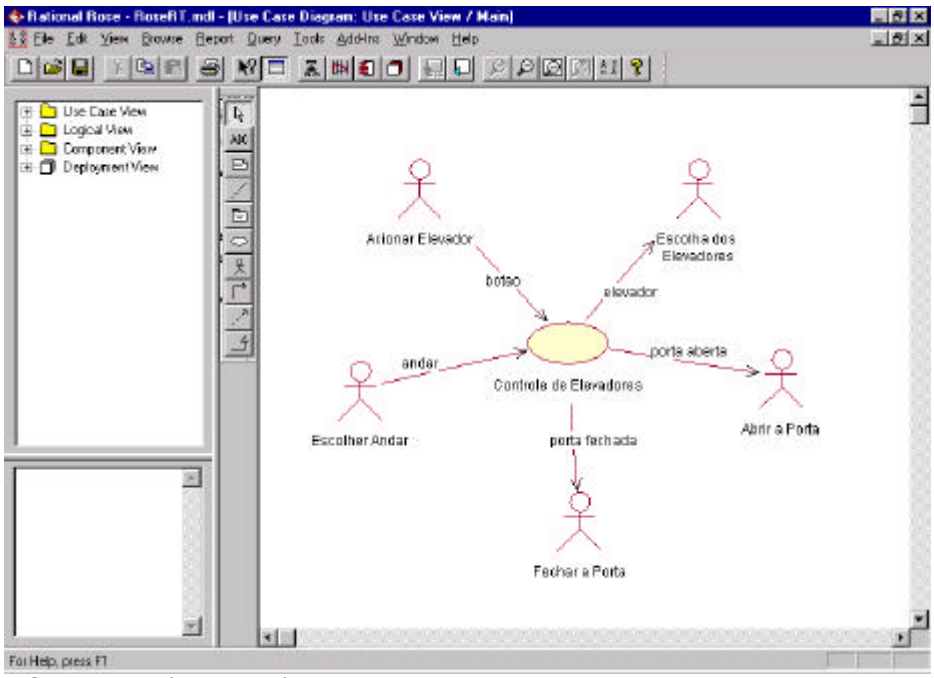

Figura 04: Diagrama de Use-Case

5.2 DIAGRAMA DE CLASSES:

O Diagrama de Classes demonstra e estrutura estática das classes de um sistema onde estas representam as entidades que são gerenciadas pela aplicação modelada.[BAR98]. O diagrama representado na figura 05 demonstram as classes, atributos, métodos e associações utilizados no sistema proposto.

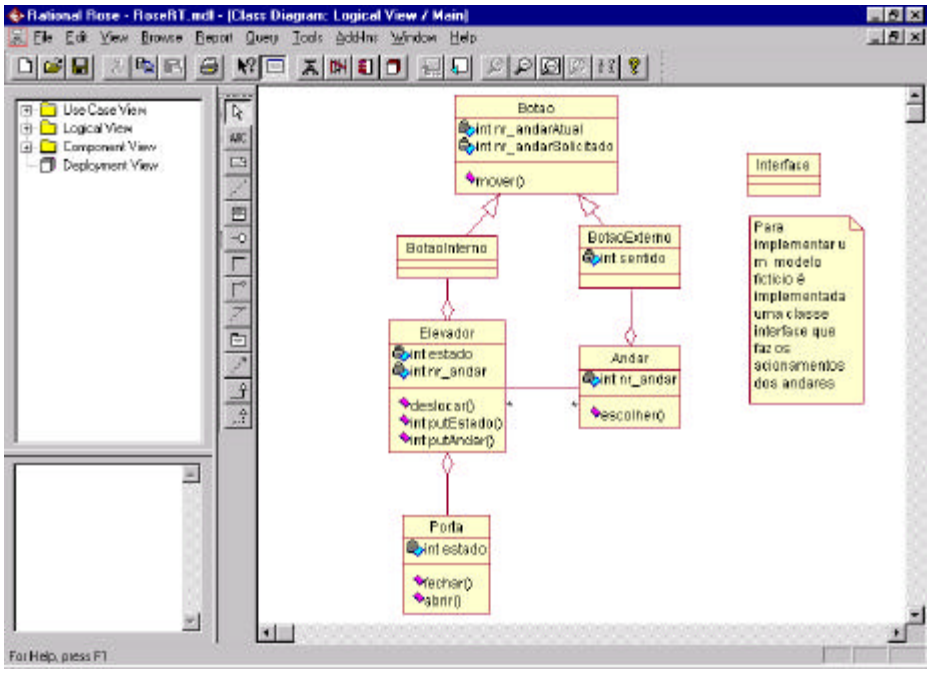

Figura 05: Diagrama de Classes

5.3 DIAGRAMA DE INTERAÇÃO DE OBJETOS:

O Diagrama de Interação mostra a colaboração dinâmica entre os vários objetos de um sistema [BAR98]. Na figura 06 pode-se observar a interação entre os objetos no cenário acionar elevador.

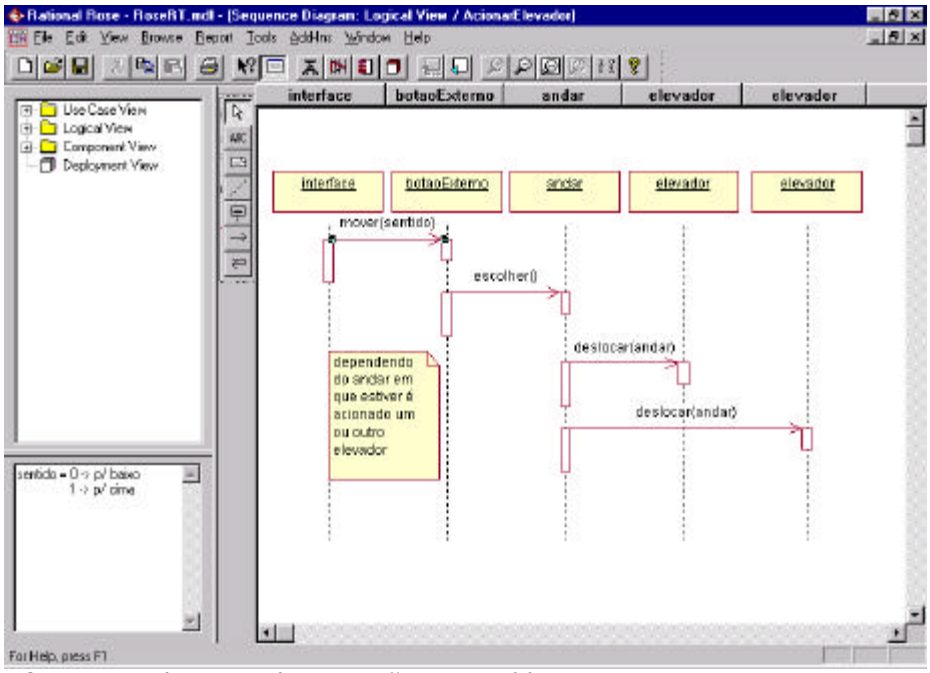

Figura 06: Diagrama de Interação entre Objetos

5.4 DIAGRAMA DE TRANSIÇÃO DE ESTADOS

O Diagrama de Transição de Estados é tipicamente um complemento para descrição das classes. Este diagrama mostra todos os estados possíveis que os objetos de uma certa classe podem encontrar e mostra também quais são os eventos do sistema que provocam tais mudanças [BAR98].

Observando a figura 07 nota-se os estados que o sistema pode assumir, essa descrição do comportamento do sistema é importante para que se possa ter uma noção bastante clara da relação tempo e estado.

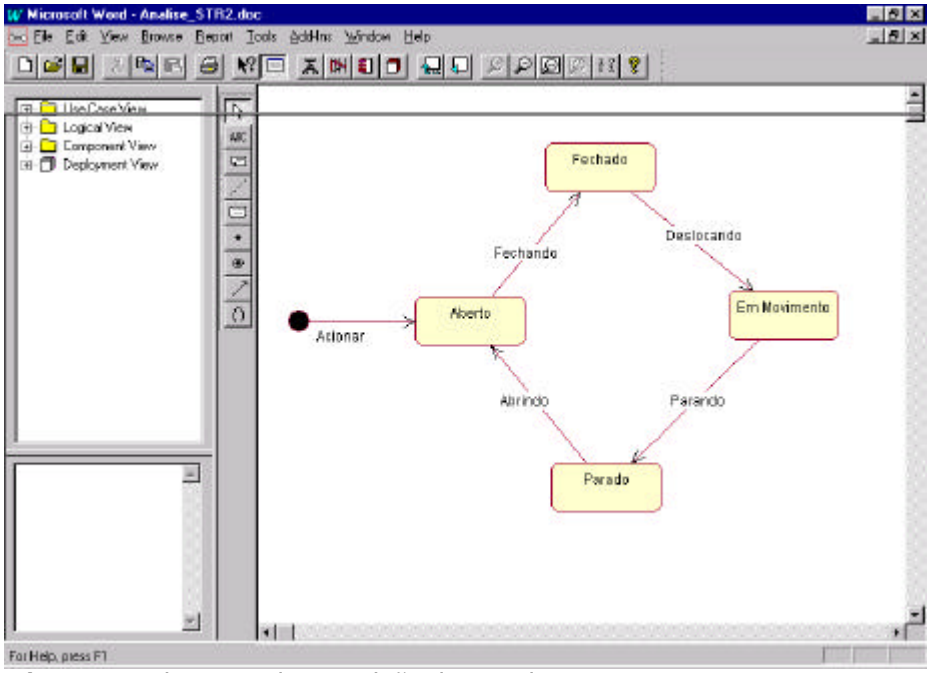

Figura 07: Diagrama de Transição de Estados

6. CONCLUSÃO

Neste texto foram apresentados conceitos e características de Sistemas de Tempo Real e metodologias adequadas para especificar tais sistemas. Entre as abordagens apresentadas optou-se pela UML, devido as suas características que facilitam a especificação de sistemas complexos.

Durante o estudo de caso realizado neste trabalho pode-se verificar como a UML e mais especificamente a ferramenta CASE Rational Rose auxiliam de forma apropriada e facilitada na especificação de Sistemas de Tempo Real.

Embora a UML não tenha sido desenvolvida especificamente para sistemas de tempo real, ela pode ser adaptada a este fim devido ao fato de possuir diversos diagramas que permitem representar a perspectiva de tempo. Em especial, pode-se destacar os Diagramas de Interação entre Objetos e o Diagrama de Transição de Estados.

O artigo demonstrou de forma didática a potencialidade de utilização da UML na especificação de Sistemas de Tempo Real a partir da perspectiva da Orientação a Objetos.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[BAR98] BARROS, Pablo. **UML - Linguagem de Modelagem Unificada** http://www.eribeiro.com.br/pablo/uml , 1998.

[BUZ98] BUZATO, Luiz E.; RUBIRA, Cecília M. F. **Construção de Sistemas Orientados a Objetos Confiáveis.** 11ª Escola de Computação, julho, 1998.

[CAR98] CARDOSO, Janette. **Uma Introdução à teoria de redes de Petri.** Anais VI Escola Regional da SBC Regional Sul. p. 182 – 194, maio. 1998.

[FUR98] FURLAN, José D. **Modelagem de Objetos através da UML.** Makron Books, São Paulo, 1998.

[HAL91] HALANG, W.; STOYENKO, A. **Construting Predictable Real Time Systems.** [S.I]: Kluwer Academic Publishers, 1991.

[HAT88] HATLEY, Derek J.; PIRBHAI, Imtiaz A. **Estratégias para especificação de sistemas em tempo real**. MacGraw-Hill, 1988.

[NET99] NETO, João. **Interface Hardware-Software.** Anais VII Escola Regional da SBC Regional Sul. p. 107 – 116, maio. 1999.

[PER94] PERREIRA, Carlos; DARSCHT, Pablo. Using Object-Orientation in Real-Time Applications: Na Experience Report. In TOOLS EUROPE, Versailles, França. Proceddings... [S.I.:s.n.], 1994.

[STA88] STANKOVIC, J. **Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems.** Computer, New York, v., n., p. 10 – 19, out. 1992.