8ª MOSTRA ACADÊMICA UNIMEP
26 a 28/10 de 2010
Tema: Desafios da Educação Superior na Agenda do Novo Milênio

18º Congresso de Iniciação Científica

UM ESTUDO EXPLORATÓRIO SOBRE TÉCNICAS DE MODELAGEM DE REQUISITOS DE SOFTWARE PARA SISTEMA EMBARCADO

Autor(es)

MARINA CALÇA

Orientador(es)

LUIZ EDUARDO GALVÃO MARTINS

Apoio Financeiro

PIBIC/CNPQ

1. Introdução

As pessoas estão cada vez mais dependentes da tecnologia, mas não sabem que por de trás de alguns equipamentos tecnológicos existem um pequeno computador embutido em um sistema maior, chamado de sistema embarcado. Quem controla todas as funções do sistema embarcado são os microcontroladores e o sistema embarcado pode ser estruturado em cinco segmentos: dispositivo de controle, dispositivo físico, *interface* de usuário, ambiente e usuário (BROY, 1997).

É nas atividades da Engenharia de Requisitos que os dados são modelados e documentados, além de ocorrer a comunicação entre os analistas e os *stackholders*. As atividades que foram propostas por Kotonya e Sommerville apud Calça (2009) são: elicitação, análise, documentação e validação de requisitos. Poucos desenvolvedores de *firmware* utilizam técnicas de modelagem para os requisitos de sistema, causando aumento no custo e no tempo de desenvolvimento do *software*.

Os primeiros trabalhos sobre técnicas de modelagem surgiram no final da década de 1960, mas a primeira abordagem surgiu com o tema “projeto estruturado”. A técnica de modelagem é a primeira representação gráfica do sistema e são utilizados para descrever os requisitos de sistema. Uma técnica amplamente utilizada é documentar a especificação do sistema como um conjunto de diagramas que expressam diversos aspectos do sistema (SOMMERVILLE, 2003).

O modelo de análise tem três objetivos: descrever o que o cliente necessita, estabelecer a base para criação de um projeto de *software* e definir um conjunto de requisitos que possam ser validados quando o *software* for construído (PRESSMAN, 2006). Na década de 1980, surgiram muitas deficiências nas técnicas de modelagem e novas técnicas foram empregadas, como as extensões em tempo real ao DFD (muitas vezes sistemas em tempo real são sistemas embarcados), UML e Redes de *Petri*.

O Diagrama de Fluxo de Dados (DFD) é uma técnica gráfica que descreve o fluxo de informação e as transformações que são aplicadas à medida que os dados se movimentam da entrada para a saída (PRESSMAN, 1995). O objetivo do DFD é modelar sistemas em qualquer nível de abstração.

Muitas aplicações de *software* são dependentes do tempo e processam mais informações orientadas ao controle do que dados

(PRESSMAN, 1995). Para modelar sistemas em tempo real uma série de extensões foi proposta, desenvolvida por *Ward/Mellor* e *Hatley/Pirbhai*.

As extensões de *Ward/Mellor* ampliam a notação básica da análise estruturada para acomodar as exigências impostas por um sistema de tempo-real (PRESSMAN, 1995). As extensões de *Hatley/Pirbhai* concentram-se menos na criação de símbolos gráficos adicionais e mais na representação e especificação dos aspectos orientados ao controle do *software*.

A diferença das duas extensões é que as extensões de *Hatley/Pirbhai* sugerem que as notações sólidas sejam separadas das pontilhadas, para depois definir um diagrama de fluxo de controle (DFC). O DFC não é representado diretamente no DFD, uma referência notacional a uma especificação de controle CSPEC é utilizada. A CSPEC é vista como uma "janela" que controla os processos representados pelo DFD. O DFC mostra como os eventos fluem entre os processos e ilustra eventos externos que ativam os processos.

Os métodos orientados a objetos aumentaram, causando uma complicação em construir um diagrama de classes, pois era necessário saber todos os métodos. Ivar Jacobson, Grady Booch e Jim Rumbaugh unificaram os métodos existentes para uma única ferramenta, com seu nome inicial Método Unificado 0.8. Em 1995, incluíram os métodos OOSE (*Object-Oriented Software Engineering*) e *Objectory* a esta ferramenta e em 1996 deu-se o nome de UML (*Unified Modeling Language*) 0.9. Após a sua versão 1.3, em 1999, começou a ser padronizada pela OMG (*Object Management Group*). Logo veio a UML 2.0 composta por 13 diagramas e descrição dos Casos de Uso. Os diagramas com visões estáticas do sistema são: classes, objetos, componentes, casos de uso, implantação e pacotes; os diagramas com visões dinâmicas do sistema são: seqüência, comunicação, *Timing Diagram*, gráficos de estado, atividades e visão geral de interação. Os diagramas abordados nesta pesquisa são: classes, casos de uso, gráficos de estados, sequência, atividades e *Timing Diagram*.

O diagrama de classes mostra um conjunto de classes, *interfaces*, colaborações e seus relacionamentos (BOOCH; RUMBAUGH; JACOBSON, 2005). O diagrama de classes serve como base para os diagramas de componentes e implantação. Também dão suporte para requisitos funcionais, são um tipo especial de diagrama, mas diferenciam pelo seu conteúdo particular.

O diagrama de casos de uso tem um papel central para modelagem do comportamento de um sistema, de um subsistema ou de uma classe (BOOCH; RUMBAUGH; JACOBSON, 2005). O diagrama de casos de uso contém assunto, atores, casos de uso e relacionamentos de dependência, generalização e associação.

O diagrama de gráficos de estados veio antes da UML, em 1987 com o trabalho de Harel. O diagrama de gráficos de estado modela o tempo de vida de um objeto e também modela o comportamento de um objeto reativo. Também mostra uma "máquina de estados", dando ênfase ao fluxo de controle de um estado para o outro. O diagrama de gráficos de estado pode ser anexado a classes, casos de uso ou a sistemas inteiros para visualizar, especificar, construir e documentar a dinâmica de um objeto individual (BOOCH; RUMBAUGH; JACOBSON, 2005). Ele deve ser utilizado quando tiver uma classe com um ou mais atributos, que reflita o estado de seus objetos em um determinado tempo, e que seu atributo mereça ser modelado para simplificar sua complexidade (MEDEIROS, 2004).

O diagrama de sequência mostra interações entre objetos na ordenação temporal das mensagens. Também serve para melhorar o diagrama de classes, permitindo que retire métodos e atributos desnecessários de um conjunto de classes (MEDEIROS, 2004). Os elementos são objetos e as mensagens que fazem comunicação entre os objetos.

O diagrama de atividades é um caso especial de diagramas de gráficos de estados, em que todos ou a maioria dos estados são estados de atividades e todas ou a maioria das transições são ativadas pela conclusão de atividades no estado de origem (BOOCH; RUMBAUGH; JACOBSON, 2005). O objetivo de um diagrama de atividades é mostrar o fluxo de controle de uma atividade para outra.

O *Timing Diagram* é um diagrama novo da UML 2.0 e classificado como diagrama de interação entre um ou mais objetos em um determinado tempo, situação ou condição, chamada de *lifeline*. A proposta inicial é mostrar a evolução do tempo ao longo de um eixo com suas variações e de forma linear.

A Rede de *Petri* surgiu quando um aluno de doutorado Carl Adam Petri defendeu sua tese intitulada de *Kommunikation mit Automaden* (*Communication with Automata* – Comunicação com Autômatos) em 1962 (PENHA; FREITAS; MARTINS, 2004). O objetivo dessa tese era desenvolver um modelo em que as máquinas de estado fossem capazes de se comunicar. Hoje há variações do modelo original de Redes de *Petri*, como coloridas, temporizadas e estocásticas. Em 1970 sua definição foi padronizada depois do trabalho de Holt e Commoner.

2. Objetivos

O objetivo geral deste projeto foi realizar um estudo exploratório sobre técnicas de modelagem de requisitos de *software*, com a intenção de elencar um conjunto de técnicas que sejam adequadas à modelagem de requisitos de *software* para sistemas embarcados.

Os objetivos específicos do projeto foram identificar um conjunto de técnicas de modelagem de requisitos com potencial promissor para o segmento de sistemas embarcados; compor um mapa de técnicas de modelagem de requisitos, que de forma integrada possam auxiliar engenheiros de *software* a produzir melhores especificações de *software* para sistemas embarcados; e apresentar uma análise qualitativa das técnicas de modelagem de requisitos investigadas.

3. Desenvolvimento

Para concluir este projeto de iniciação científica foi desenvolvido um estudo de caso com o objetivo de modelar um sistema embarcado que controla e monitora a temperatura em um ambiente fechado, adotando-se as técnicas de modelagem estudadas, que são: extensões ao DFD em tempo real de *Ward/Mellor* e *Hatley/Pirbhai*, diagrama de classes, diagrama de casos de uso, diagrama de gráficos de estados, diagrama de seqüência, diagrama de atividades, *Timing Diagram* e Redes de *Petri*. Esse sistema embarcado foi especificado no projeto de iniciação científica “Em busca de um processo de engenharia de requisitos para o desenvolvimento de *software* embarcado” (CALÇA, 2009).

Também foi elaborada uma lista com as principais características dos sistemas embarcados que precisam ser modeladas, além de fazer um mapeamento da utilidade das técnicas de modelagem estudadas com as características dos sistemas embarcados; foi feita uma identificação de como as técnicas de modelagem atendem as características dos sistemas embarcados; foi elaborada uma tabela relacionando as características dos sistemas embarcados com as técnicas de modelagem e discussão dos resultados (vide Figura 1 uma parte da tabela).

A Figura 2 apresenta o diagrama de casos de uso para o sistema embarcado analisado, foi desenvolvido um único diagrama com a visão geral; a Figura 3 apresenta uma parte da modelagem do diagrama de sequência que é armazenar dados da leitura, além dessa parte existem mais três: digitar temperatura mínima e máxima, digitar tempo de aquisição e interromper armazenamento.

4. Resultado e Discussão

Muitas pessoas não sabem que estão rodeadas de sistemas embarcados, mas no seu cotidiano há vários equipamentos que os contêm, como exemplo brinquedos, eletrônicos, eletrodomésticos etc. Como a área de engenharia de requisitos para sistemas embarcados é utilizada por poucos, esse projeto de iniciação científica teve como objetivo realizar um estudo exploratório sobre técnicas de modelagem de requisitos de *software* embarcado, com a intenção de elencar um conjunto de técnicas que sejam adequadas à modelagem de requisitos de *software* embarcado.

Os requisitos de sistema descrevem-se utilizando as técnicas de modelagem, pois são destinados a implementação do sistema; permitindo que os desenvolvedores e os usuários possam ter a visão do sistema ao longo do seu desenvolvimento e também verificar a consistência.

Com a modelagem do sistema embarcado analisado foi possível determinar a importância de modelar os aspectos dinâmicos como as interações entre seus componentes. Para modelar os sistemas embarcados utilizando o *Timing Diagram* é preciso verificar se o sistema precisa respeitar o tempo exato das atividades, se há comprometimento na segurança do sistema maior em relação ao lugar, pessoas etc. Já a técnica de modelagem Redes de *Petri* serve como complemento para verificar as interações entre dispositivos de controle e dispositivos físicos ou até mesmo com a *interface* de usuário.

5. Considerações Finais

As técnicas de modelagem são muito importantes para verificar a consistência dos sistemas embarcados. Como os sistemas embarcados são diferentes de outros sistemas foi proposta uma lista para identificar as principais características dos sistemas embarcados, para analisar qual das necessidades as técnicas de modelagem atendem. Como o objetivo geral foi realizar um estudo exploratório sobre técnicas de modelagem de requisitos de *software*, com a intenção de elencar um conjunto de técnicas que sejam adequadas à modelagem de requisitos de *software* para sistemas embarcados, foi possível concluir que é necessária uma técnica de modelagem para modelar aspectos estáticos dos sistemas embarcados para detalhar os dispositivos físicos, através da técnica de modelagem diagrama de classe.

Já para modelar os aspectos dinâmicos dos sistemas embarcados foi possível determinar dois conjuntos de técnicas podem modelar esses aspectos: o primeiro conjunto são as técnicas de modelagem diagrama de seqüência e diagrama de casos de uso e o segundo conjunto são os diagramas de gráficos de estados e de atividades; para visualizar o sistema inteiro pode-se utilizar alguma das duas extensões de *Ward/Mellor* ou *Hatley/Pirbhai*. Já a utilização do *Timing Diagram* é possível quando há necessidade do tempo exato de cada ação, para não ter comprometimento com a segurança do sistema embarcado. E as Redes de *Petri* servem como complemento para a visualização das interações que ocorrem no sistema.

Como sistemas embarcados é uma área que está em franco crescimento, e ainda demandará muita pesquisa e desenvolvimento para melhorar a qualidade dos *softwares* que implementam as funcionalidades necessárias, essa pesquisa foi um complemento da pesquisa anterior, “Em busca de um processo de engenharia de requisitos para o desenvolvimento de *software* embarcado” (CALÇA, 2009) para avançar nas especificações de requisitos para sistemas embarcados.

Referências Bibliográficas

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2ª Ed. Rio de Janeiro: Elsevier, 2005;

BROY, M. ***Requirements Engineering for Embedded Systems,*** (FEmSys’97)
Workshop on Formal Design of Safety Critical Embedded Systems, Germany, 1997;

CALÇA, Marina. **Em busca de um processo de engenharia de requisitos para o desenvolvimento de *software* embarcado**. 2009. (Projeto de iniciação científica - PIBIC) – Universidade Metodista de Piracicaba, Piracicaba, SP;

MEDEIROS, Ernani Sales de. **Desenvolvendo *software* com UML 2.0: definitivo.** 1ª Ed. São Paulo: Pearson Makron Books, 2004.

[OMG'09] OMG. ***Unified Modeling LanguageTM (OMG UML), Superstructure.*** 2009. *Version 2.2.*

PENHA, Dulcinéia Oliveira da; FREITA, Henrique Cota de; MARTINS, Carlos Augusto Paiva da Silva. **Modelagem de Sistemas Computacionais usando Redes de *Petri:* aplicação em projeto, análise e avaliação.** IV Escola Regional de Informática RJ/ES, 2004.

PRESSMAN, Roger S. **Engenharia de *software*.** 1ª ed. São Paulo: Makron Books, 1995;

PRESSMAN, Roger S. **Engenharia de *software.*** 6ª ed. São Paulo: McGraw-Hill, 2006.

SOMMERVILLE, Ian. **Engenharia de *software*.** 6ª ed. São Paulo: Addison Wesley, 2003;

Anexos

	Diagrama de Sequência	Diagrama de Classe	Diagrama de Casos de Uso
Microcontroladores e suas portas		Atributos para identificar o microcontrolador, atributos para identificar a ligação de cada pino com os	
Sensores e atuadores com seus tempos exatos		Atributos para identificar os atuadores	
Ambiente e suas variáveis físicas			
Interação entre microcontrolador com dispositivos físicos	Ações sequenciais entre microcontroladores e dispositivo físico	Operações entre os microcontroladores e dispositivos físicos	Captura das funcionalidades do microcontrolador sobre os dispositivos físicos
Interação entre dispositivos físicos com o ambiente			Captura dos relacionamentos do microcontrolador com os dispositivos físicos que monitoram e controlam o ambiente
Interação entre *interface* de usuário com dispositivo de controle	Ações sequenciais entre usuário e dispositivo de controle	Operações entre os microcontroladores e a interface de usuário, operações entre a interface de usuário e o microcontrolador	Captura dos relacionamentos da interface de usuário com o dispositivo de controle. Esse relacionamento é feito através do display LCD, botões e teclados.
Requisitos de tempo real			
Requisitos de Usabilidade		Consistência da interface de usuário	
Requisitos de Desempenho	tempo de resposta, taxa de transferência		
Requisitos de Confiabilidade			
Requisitos de Segurança			

Figura 1 - Análise das técnicas de modelagem Diagrama de sequência, Diagrama de Classe e Diagrama de Casos de Uso.

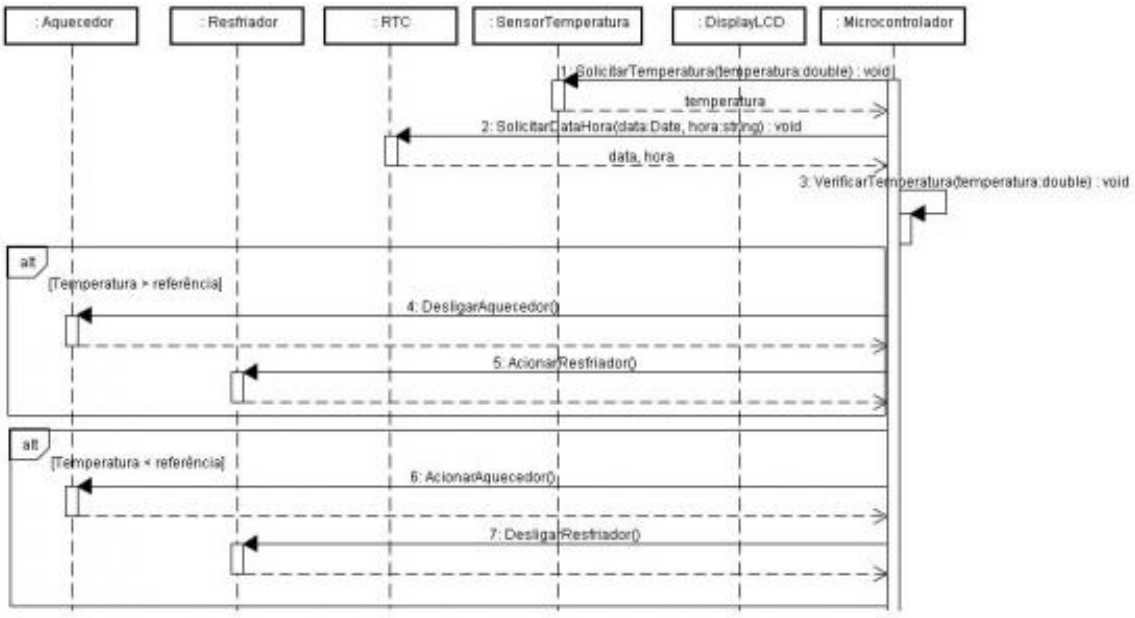

Figura 3 – Diagrama de Sequência parte 1 para a modelagem armazenar dados de leitura.

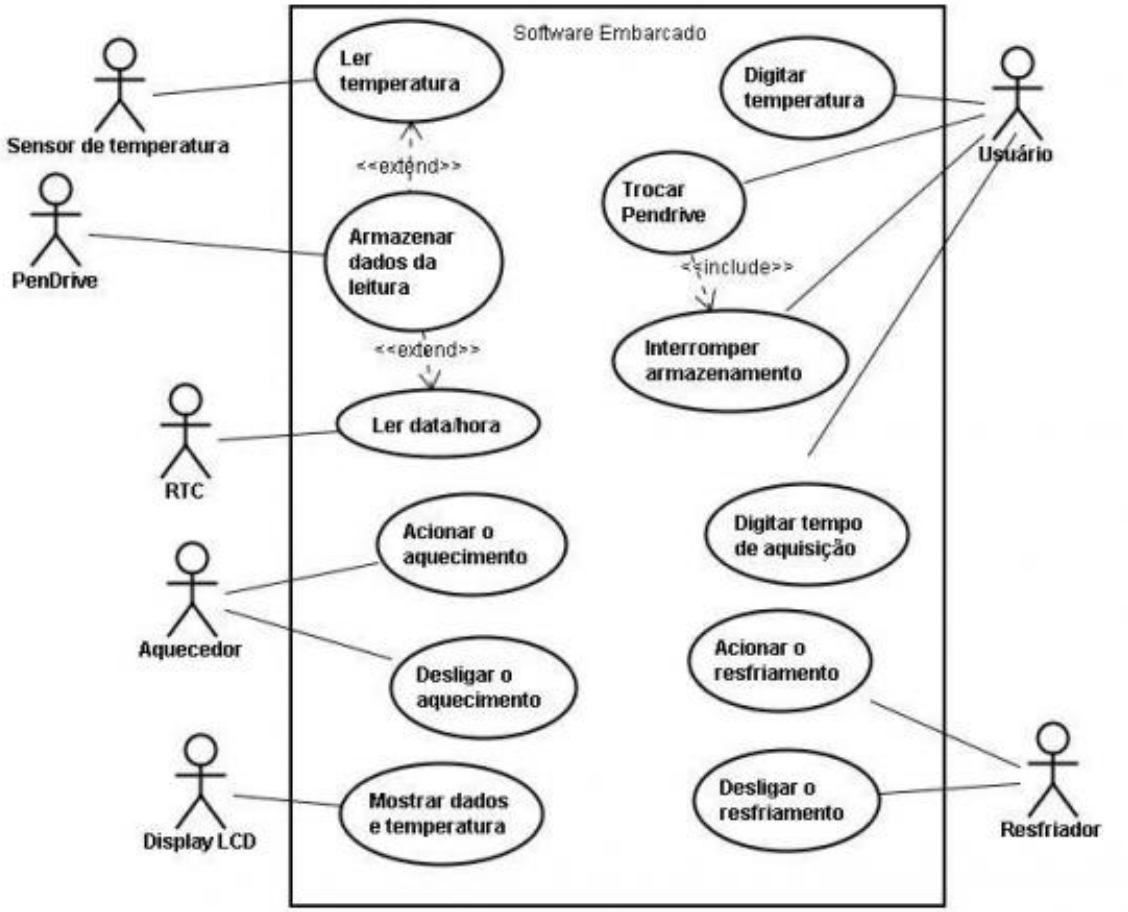

Figura 2 - Diagrama de Caso de Uso para sistema embarcado analisado.