Uma Proposta de Extensão de UML-RT para Permitir a Modelagem de Sistemas Mecatrônicos

Beatriz T. Andrade
Departamento de Ciência da Computação, Universidade Federal de Sergipe
São Cristóvão, 49100-000, Brasil
beatriz@dcce.ufs.br

Leila Silva
Departamento de Ciência da Computação, Universidade Federal de Sergipe
São Cristóvão, 49100-000, Brasil
leila@ufs.br

e

Eduardo O. Freire
Núcleo de Engenharia Elétrica, Universidade Federal de Sergipe
São Cristóvão, 49100-000, Brasil
efreire@fisica.ufs.br

Abstract

Mechatronic systems are basically compounded of hardware, software and mechanical components, and are characterized by the interaction with the environment. In general, such systems can be considered real time systems. This paper discusses the adequacy of UML-RT's use in the modeling of mechatronic systems and proposes an extension of this language in order to provide support to the modeling of the systems' physical components. With the objective of validating this approach, a case study in robotics was also developed.

Keywords: Mechatronic System's Modeling, UML-RT.

Resumo

Sistemas mecatrônicos são basicamente compostos por componentes de hardware, software e mecânicos, caracterizando-se pela interação com o ambiente e, em geral, podem ser considerados como sistemas de tempo real. Este artigo discute a adequabilidade do uso de UML-RT na modelagem de sistemas mecatrônicos e propõe uma extensão desta linguagem para prover suporte à modelagem dos componentes físicos do sistema. Um estudo de caso em robótica foi desenvolvido para validar o projeto.

Palavras Chave: Modelagem de Sistemas Mecatrônicos, UML-RT.

1 INTRODUÇÃO

A computação está cada vez mais presente na vida de todos nós. À medida que os sistemas computacionais aumentam suas funcionalidades incorporando novas tecnologias, mais diretamente eles tendem a interagir com os seres humanos e eventos reais. Neste contexto, surgem aplicações onde o tempo é considerado um componente ortogonal ao projeto. Neste tipo de aplicações, um resultado atrasado não tem valor ou pior, possui um valor negativo. Assim, não é satisfatório que o resultado esteja correto; ele tem que ser obtido em um tempo pré-definido. A sistemas com tais características, dá-se o nome de *sistemas de tempo real*.

Dentre as diversas linguagens de modelagem para sistemas de tempo real, a UML-RT (*Unified Modeling Language for Real Time*) [17] tornou-se um padrão neste contexto. A UML-RT é uma extensão da UML (*Unified Modeling Language*) [7], criada com o objetivo de incorporar conceitos da linguagem de modelagem visual ROOM (*Real Time Object Oriented Modeling*) [18] através de seus estereótipos. A linguagem ROOM é uma linguagem de domínio específico para definição de arquiteturas de sistemas de tempo real, e tem sido usada com sucesso há mais de duas décadas.

A UML-RT modela a natureza reativa dos sistemas de tempo real, e por isso se apresenta como uma linguagem bastante poderosa, acrescentando potencialidades a UML. A UML-RT fornece um modelo para análise complexa de sistemas de tempo real dirigidos a eventos, simulando a forma como os sistemas de tempo real realmente trabalham e, ajudando a construir e gerar código para aplicações críticas de tempo real. Ela foi desenvolvida para ser aplicável a uma categoria de sistemas de tempo real que compreende sistemas complexos, dirigidos a eventos, e potencialmente distribuídos. Esta categoria é a predominante entre os sistemas em tempo real.

Por outro lado, sistemas mecatrônicos são sistemas que incluem componentes de hardware, de software e mecânicos, e possuem algum tipo de interação com o ambiente [14]. Essa interação é uma característica importante, pois torna tais sistemas dependentes de eventos reais. Dessa maneira, pode-se enquadrar sistemas mecatrônicos como sistemas de tempo real. Assim, teoricamente é possível a modelagem de sistemas mecatrônicos através da UML-RT.

O uso de UML-RT para a modelagem de sistemas mecatrônicos é incipiente. Em geral, as equipes que desenvolvem projetos de sistemas deste tipo são formadas por engenheiros que desconhecem linguagens da área de Engenharia de Software. O objetivo deste artigo é investigar a adequabilidade do uso de UML-RT na modelagem de tais sistemas. Para tal, um estudo de caso em robótica é apresentado. Como UML-RT lida apenas com o aspecto lógico (software) do sistema, uma extensão desta linguagem para prover suporte à descrição dos componentes mecânicos é aqui proposta. Com esta extensão é possível então obter a visão lógica e física dos componentes da aplicação, tornando a linguagem uma ferramenta integradora das diversas áreas envolvidas na construção deste tipo de sistemas.

Embora existam trabalhos na literatura que sugerem a aplicação da UML na modelagem de sistemas mecatrônicos, este é um tema pouco explorado, sendo desconhecidos dos autores trabalhos que detalhem a aplicação de UML-RT no contexto da robótica ou que proponham extensões para a modelagem física destes sistemas.

Dentre os trabalhos relacionados a esta proposta, [13] apresenta a modelagem de um robô autônomo utilizando a UML. O objetivo é modelar um robô móvel genérico que executa uma tarefa. O processo de modelagem é iniciado com o modelo do processo do negócio, passando para diagramas UML que descrevem a solução encontrada para modelar o sistema de controle de tarefas do robô.

Por fim, alguns aspectos da implementação são discutidos, objetivando mostrar que a implementação do sistema pode ser feita a partir do modelo orientado a objetos desenvolvido.

No trabalho apresentado em [5] é estudada uma aplicação para sistemas mecatrônicos onde são usados conceitos de estereotipagem baseados em [17]. No trabalho são comparadas as vantagens e desvantagens do uso de estereótipos. Primeiramente é apresentado um problema como estudo de caso e são obtidas duas soluções. A primeira contém somente elementos padrões da UML. A segunda faz uso de um estereótipo especial, o Adaptador de Blocos de Função (FBA – *Function Block Adapter*), que tem como função fazer o mapeamento de sinais UML para sinais de *function blocks*, e vice-versa. As comparações são feitas analisando os resultados das duas opções na tarefa de integração de *function blocks* em um modelo UML.

Em [11], é feito um estudo sobre duas abordagens utilizadas na modelagem de sistemas mecatrônicos. A primeira forma é através da UML, e a segunda, da Modelica [10], uma linguagem utilizada para realizar modelagem física hierárquica orientada a objetos. A UML é aplicada no artigo para fazer o projeto do sistema mecatrônico com alto nível de abstração, enquanto a Modelica é usada para modelar e prototipar em um nível médio de abstração do projeto.

Este trabalho está organizado como descrito a seguir. A Seção 2 aborda a modelagem de sistemas mecatrônicos, enquanto a Seção 3 introduz a UML-RT. O estudo de caso é apresentado na Seção 4, onde a extensão proposta é também discutida. Por fim, na Seção 5, são apresentadas as conclusões sobre o trabalho e delineados possíveis trabalhos futuros.

2 A MODELAGEM DE SISTEMAS MECATRÔNICOS

Sistemas mecatrônicos são sistemas que interagem com o ambiente através de sensores, controladores e atuadores. A Mecatrônica pode ser entendida como a integração sinérgica da Engenharia Mecânica com a Elétrica e o controle por computador no projeto e na manufatura de produtos e processos [16].

Assim, para implementar um sistema mecatrônico normalmente são requeridos profissionais de áreas específicas que trabalham em conjunto para o desenvolvimento do sistema. Neste contexto, surge a necessidade de uma metodologia de projeto que integre os profissionais envolvidos de forma eficiente e viabilize a produção de sistemas flexíveis, confiáveis e econômicos.

Algumas metodologias para a modelagem de sistemas mecatrônicos que utilizam UML já foram propostas em [12], [13] e [2]. Por atender aos objetivos deste trabalho, adotamos aqui a proposta de [12], com o diferencial de substituir a UML por UML-RT. Esta substituição tornou mais natural a representação das características reativas do sistema mecatrônico pelo poder de representação conferido pelo uso de cápsulas ao invés dos diagramas da UML tradicional.

Em [12], inicialmente deve ser feito um levantamento de requisitos, através dos casos de uso e de cenários. A seguir, a partir dos casos de uso e de cenários, são identificados os objetos. Com isso tem-se uma versão preliminar dos diagramas de classes e de objetos. Caso posteriormente seja necessária a inclusão de novos objetos, os diagramas de classes e de objetos devem ser refeitos. A seqüência de ações descritas nos cenários é representada graficamente em um diagrama de colaboração ou de seqüência. Todos os estados por onde um objeto passa são mapeados para um diagrama de estados. Posteriormente em [11], o autor acrescenta que caso haja atividades paralelas deve ser usado o diagrama de atividades, e que a modelagem física, a simulação e a prototipação

devem ser feitas com ferramentas específicas para o tipo de sistema mecatrônico a ser implementado, onde são ressaltados o uso do Modelica [10] e do Dymola [3].

3 UML-RT

A utilização de uma linguagem de modelagem é essencial na compreensão do sistema, na sua especificação, e na comunicação entre as pessoas envolvidas. Devido à característica interdisciplinar de um sistema mecatrônico, há a necessidade de linguagens adequadas para a modelagem deste tipo de sistema.

Visando a modelagem de sistemas de tempo real, a linguagem de modelagem UML foi adaptada para prover suporte sistemas que possuem requisitos de restrição de tempo. Assim, foi desenvolvida a UML-RT, que usa os mecanismos de extensão da UML 1.1 para incorporar conceitos advindos da linguagem de modelagem visual ROOM.

Uma extensão é um recurso da UML para introduzir novas funcionalidades e adaptar sua estrutura para a modelagem de situações específicas, não definidas em sua versão original, e pode ser expressa em termos de: estereótipos (*stereotypes*), valores rotulados (*tagged values*), e limitações (*constraints*). Um estereótipo é uma extensão do vocabulário da linguagem; um valor rotulado, uma extensão para uma propriedade de um elemento do modelo; e uma limitação é uma extensão semântica da linguagem. Uma extensão pode utilizar outras extensões como pré-requisito. Na Tabela 1, estão listados os estereótipos definidos pela UML-RT e sua classe correspondente no metamodelo UML [17].

Tabela 1: Estereótipos definidos pela UML-RT.

Nome do Estereótipo	Classe do Metamodelo
protocol	*Collaboration*
protocolRole	*ClassifierRole*
port	*Class*
capsule	*Class*
chainState	*State*

Estes estereótipos podem ser utilizados no lugar das classes correspondentes do metamodelo durante a especificação. Assim, os conceitos introduzidos por cada estereótipo podem ser aplicáveis em todos os diagramas onde a classe do metamodelo é utilizada.

A UML-RT se baseia nos conceitos de *cápsulas*, *portas* e *protocolos*. Uma *cápsula* (*capsule*) representa um fluxo de controle independente em um sistema e sua arquitetura. Cápsulas se comunicam através de *portas* (*ports*), ligadas por *conectores*, e utilizando *protocolos* (*protocols*). A única forma de comunicação da cápsula com o exterior se dá através de suas portas. Da mesma maneira, de fora de uma cápsula não se pode ter acesso a nenhuma informação que não as que provêm de suas portas. A comunicação entre cápsulas é definida através das regras de protocolo (*protocolRole*), que definem o formato e ordem das mensagens trocadas através do protocolo ao qual está associado.

Cápsulas podem estar aninhadas e, neste caso, são denominadas *compostas*. A uma cápsula no nível

mais interno de aninhamento dá-se o nome de *subcápsula*. As cápsulas e suas subcápsulas também se comunicam exclusivamente através de portas. As regras de protocolo representam a perspectiva de cada um dos participantes em um cenário de comunicação. Como a porta representa o papel de participante, é designado a ela um papel de protocolo, que pode ser de uma *base* ou de *conjugado*. Estes papéis são definidos a fim de diferenciar as portas no cenário de comunicação, onde as saídas de um são as entradas de outro, e vice-versa. Assim, para que haja troca de mensagens entre duas portas, uma deve assumir o papel de base, e a outra de conjugado. A porta que possui o papel de base é representada por um quadrado preto cheio no diagrama de estrutura. A porta que possui o papel de conjugada é representada por um quadrado vazado. A Figura 1 mostra uma simplificação do diagrama de estrutura de duas cápsulas que se comunicam através de um conector que interliga suas portas.

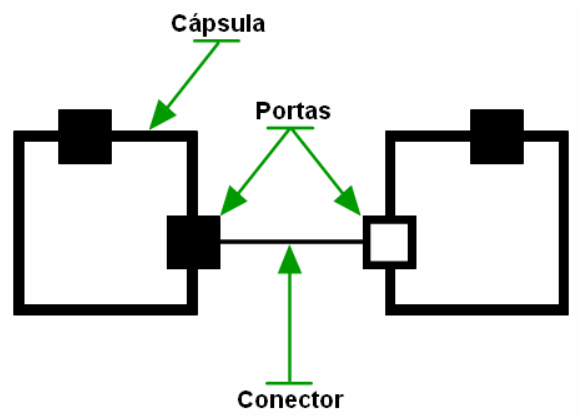

Figura 1. Simplificação do diagrama de estrutura de duas cápsulas.

Os diagramas de classe, estrutura e estados estendem conceitos da UML-RT. Na Seção 4.2 serão ilustrados tais conceitos, ao passo em que a modelagem do sistema é descrita.

4 APLICANDO UML-RT NA ESPECIFICAÇÃO DE UM SISTEMA MECATRÔNICO

Nesta seção descrevemos um sistema mecatrônico - um sistema de navegação para um robô móvel – modelado utilizando a UML-RT.

4.1 O Estudo de Caso Proposto

O sistema modelado é composto por componentes mecatrônicos (um robô móvel e uma câmera) e de software (sistema de visão e controlador instalados em um microcomputador). O objetivo do sistema é possibilitar o deslocamento de um robô móvel do ponto de origem até o destino, evitando que ele colida com possíveis obstáculos existentes no caminho. Para isso, são continuamente processadas pelo sistema de visão imagens capturadas pela câmera, com o objetivo de extrair informações sobre o robô móvel e o ambiente em que este está imerso. Estas informações são enviadas para o controlador, que decide a direção do robô e envia para ele os comandos correspondentes à ação a ser executada. O sistema é realimentado pela captura contínua de imagens.

Por ser executado em tempo real, o sistema depende do tempo de execução de suas iterações. Cada iteração é a seqüência de ações que abrange desde a aquisição da imagem até o movimento do robô, e deve ser executada em no máximo 100 ms.

4.2 Modelagem do Estudo de Caso

Na modelagem do Sistema de Navegação foram utilizadas a linguagem UML-RT e a metodologia descrita em [12], onde a linguagem utilizada nesta metodologia, a UML, foi substituída pela UML-RT com o objetivo de verificar e analisar seu desempenho na modelagem de sistemas reativos. A ferramenta utilizada para a modelagem do sistema foi a *Rational Rose Real Time* [6].

Neste artigo é abordada somente a etapa do projeto do sistema, fase na qual a UML-RT é efetivamente utilizada. Seguindo a metodologia proposta em [12], elaborou-se os diagramas de caso de uso, classes, estados, estrutura, e seqüência. Por se tratar de um número extenso de diagramas, somente apresentados neste artigo os necessários para a compreensão e ilustração do trabalho desenvolvido. A modelagem detalhada do sistema pode ser encontrada em [1].

4.2.1 Diagramas do Sistema de Navegação

Na modelagem dos requisitos funcionais (ver diagrama de casos de uso na Figura 2) foram analisadas as formas de interação do Sistema de Navegação com seu exterior, e constatou-se que o sistema possui apenas um ator, que é o ator Usuário, responsável pela calibração e definição do ponto de destino do sistema. Como entradas do sistema, são inseridos pelo usuário as cores do robô, sua posição inicial na cena e seu ponto de destino (casos de uso Calibrar Cores, Selecionar Robô e Definir Ponto de Destino, respectivamente). O conjunto destas entradas, que compõe a entrada do sistema de navegação, é denominado Informacoes. O sistema não possui saídas para o usuário além do monitoramento do robô móvel (Visualizar Informações). Além destas funcionalidades, são disponibilizadas ao usuário opções referentes à utilização do sistema (Reiniciar Sistema e Atualizar Imagem).

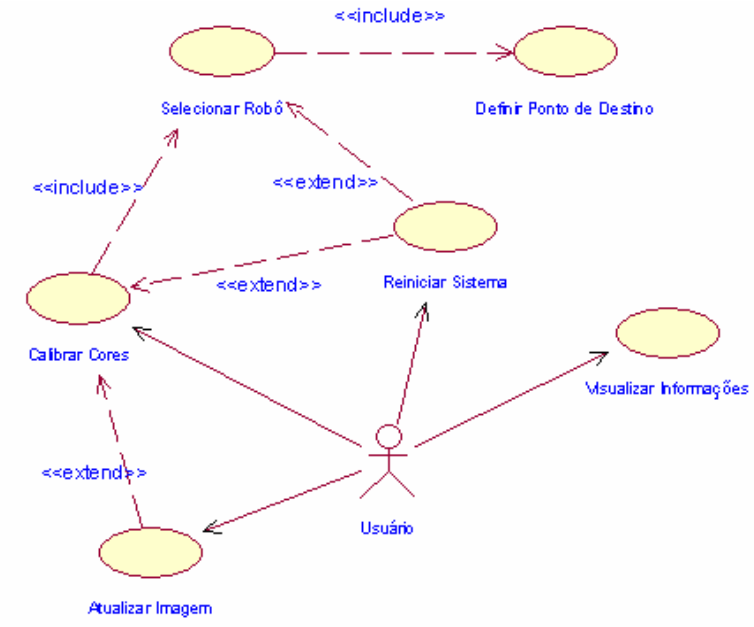

Figura 2. Diagrama de casos de uso do sistema de navegação.

O Sistema de Navegação pode ser divido em unidades independentes. Neste contexto, são mapeadas cinco unidades: a câmera, o robô, a unidade responsável pela visão do robô, a unidade responsável pelo seu controle, e um transmissor, utilizado para enviar dados do sistema de controle para o robô. Assim, pode-se mapear o próprio sistema como uma cápsula, que possui apenas uma porta, por onde entram os dados do conjunto Informações e saem imagens capturadas pela *webcam*.

Esta cápsula é então composta por cinco subcápsulas, que mapeiam as unidades independentes do sistema de navegação. O relacionamento entre estas cápsulas é definido com base nas mensagens trocadas entre elas.

A Figura 3.a apresenta o diagrama de estrutura da cápsula Sistema, a qual representa o sistema de navegação. Neste diagrama, as cápsulas são os retângulos e as portas das cápsulas são os quadrados distribuídos sobre as arestas destes. A comunicação entre portas é definida através dos conectores, que ligam cápsulas que trocam mensagens. A cápsula Sistema possui apenas uma porta (visao) que implementa protocolo PInformacoes, recebendo informações e enviando imagens.

A porta da cápsula Sistema é uma porta de transmissão (*relay*), ou seja, todos os dados que recebidos por ela são passados adiante. Neste caso, os dados recebidos são enviados para a porta sistema da cápsula Visao. A cápsula Visao se comunica com duas outras cápsulas. A primeira é a cápsula Câmera, cuja comunicação se dá através do protocolo PImagem, responsável por requisitar e receber imagens da cena do robô. Estas imagens são enviadas para o usuário através da porta sistema e são processadas internamente, a fim de extrair as características da imagem. A segunda cápsula com que a cápsula Visao se comunica é a cápsula Controle, através do protocolo PCaracteristicas, por onde envia as características extraídas da imagem processada. A cápsula Controle, por sua vez, com base nas características da imagem recebidas, calcula as ações que o robô deverá tomar e as envia para a cápsula Transmissor através do protocolo PRS_232. Por fim, a cápsula Transmissor se comunica com a cápsula Robo, retransmitindo os comandos que o robô deve executar através do protocolo PComunicacaoSerial.

Assim como acontece com a cápsula Sistema, cada uma das cápsulas nela contidas pode conter cápsulas aninhadas. Neste caso, elas contêm um símbolo no seu canto inferior direito, especificando que são compostas. Este é o caso das cápsulas Robo e Visao.

A notação gráfica das cápsulas em um diagrama de classe é representada por uma caixa com o estereótipo <<Capsule>> e um símbolo em cima, à direita. Nesta notação, a cápsula é descrita através de três ou quatro compartimentos. O primeiro lista os atributos da cápsula; o segundo, os métodos; o terceiro, as agregações, mas este não é obrigatório; e o quarto lista as portas. O diagrama de classes da cápsula Sistema é bastante simples, pois ela somente repassa mensagens através de sua porta. Este diagrama contém apenas a declaração de sua porta visão (Figura 3.b).

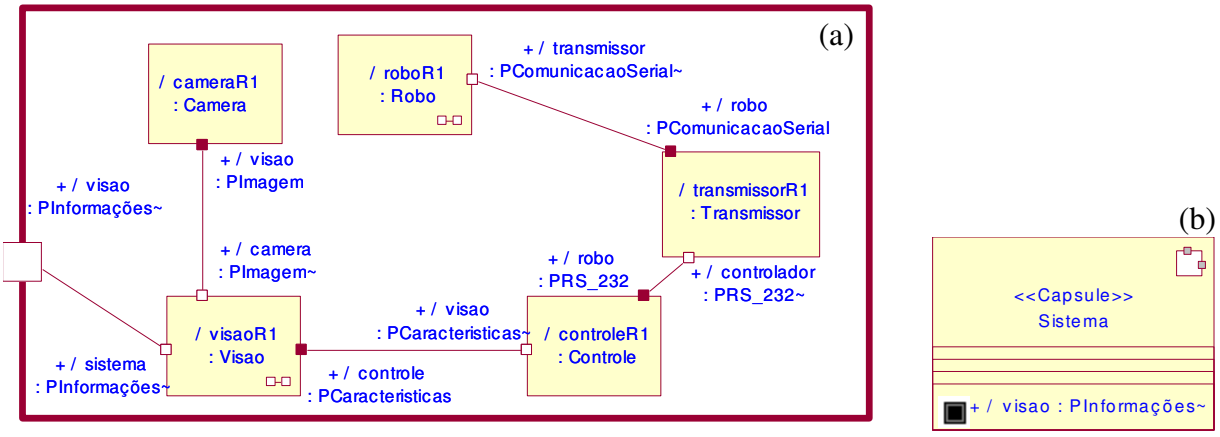

Figura 3. Diagramas de (a) estrutura e (b) classe da cápsula Sistema.

As cápsulas também podem ser representadas através de uma extensão do diagrama de estados da UML, que ilustra os estados pelos quais uma cápsula passa durante seu ciclo de vida. Além da máquina de estados da cápsula, o comportamento dos protocolos também é representado em UML-RT através do diagrama de estados.

Os diagramas de estados da UML-RT representam objetos ativos, e não possuem estados finais [15]. Em cada máquina de estados da UML-RT é suposto que existe um estado composto que possui todos os outros. A este estado é dado o nome de estado *topo* ou *S0*, e o estado inicial é implicitamente disparado quando uma instância da cápsula ou porta é criada.

O diagrama de estados da cápsula Sistema é exibido na Figura 4 . Nele, são mapeados os estados pelo qual a cápsula Sistema passa. Inicialmente ela está no estado de espera de informações, do qual só sai ao receber a mensagem contendo as informações. Ao receber a mensagem, a cápsula Sistema passa para o estado de retransmissão, onde repassa a mensagem recebida para a cápsula Visao. Após a retransmissão da mensagem, a cápsula Sistema fica no estado de espera de imagens, que termina quando a cápsula recebe uma mensagem da cápsula Visao contendo uma imagem. Ela vai então para o estado de retransmissão da imagem. Após a retransmissão, ela volta ao estado de espera por imagens, ficando assim presa em um laço. É interessante observar que da perspectiva da cápsula Sistema não se sabe para que cápsula a mensagem vai ao sair pela sua porta, nem de onde vieram as informações sobre a cena recebidas por esta cápsula, daí não interessa nesta situação saber para quem a cápsula Sistema repassa a imagem.

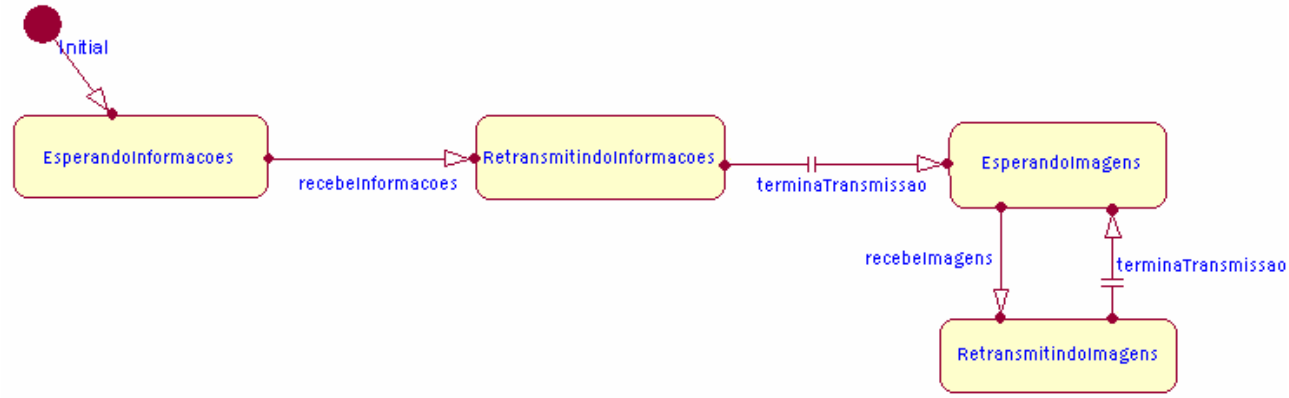

Figura 4. Diagrama de estados da cápsula Sistema.

A cápsula Sistema se comunica com a cápsula Visão através do protocolo PInformacoes. Ele é formado por dois tipos de sinais, o sinal que leva o objeto que contém informações, e o sinal que leva o objeto que contém a imagem. Na Figura 5.a tem-se o diagrama de classe do protocolo. Como o papel do conjugado pode ser derivado do papel da base através da inversão das entradas e saídas, costuma-se especificar apenas o papel da base no protocolo em seus diagramas.

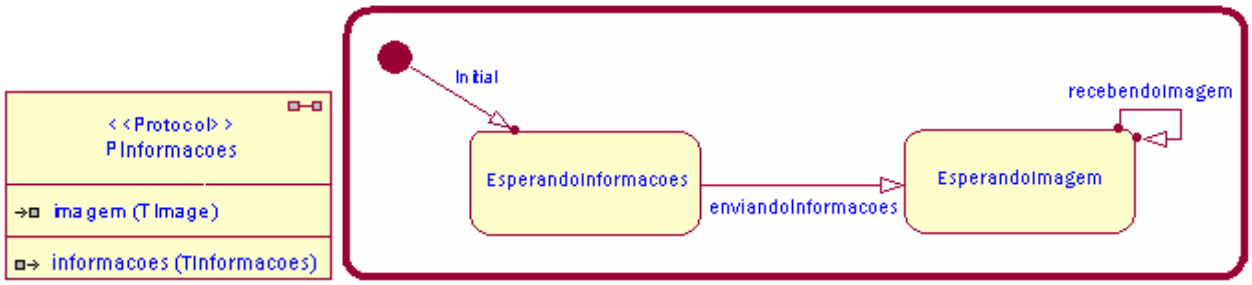

Figura 5. Diagramas de (a) classe e de (b) estados do protocolo PInformacoes.

O diagrama de classes de um protocolo é representado por uma caixa com o estereótipo <<Protocol>> e um símbolo em cima, à direita. O diagrama de classes de um protocolo é diferente do diagrama de classes que representa uma cápsula. Nele, há apenas dois compartimentos; o primeiro tem como função descrever os sinais de entrada (no protocolo PInformacoes, o sinal imagem), e o segundo, os sinais de saída (o sinal informacoes). A cápsula Sistema deve repassar um objeto do tipo TInformacoes (que contém as informações recebidas do exterior) para a cápsula Visao, e receber da cápsula Visao e repassar as imagens capturadas pela câmera. O envio das informações se dá apenas na inicialização do sistema, enquanto o envio das imagens acontece durante todo o tempo em que o robô está em funcionamento. A Figura 5.b, ilustra o diagrama de estados do protocolo PInformacoes.

4.3 Proposta de Adaptação da UML-RT para Sistemas Mecatrônicos

Durante a modelagem do estudo de caso houve dificuldades durante a especificação das cápsulas referentes aos componentes físicos do sistema (câmera e robô). Isso aconteceu devido ao fato que não é possível mapear aspectos relacionados à composição física destes elementos de forma natural na UML-RT. Por se tratar de uma linguagem que tem como foco principal a modelagem lógica do sistema, visando a geração de código, informações importantes referentes a aspectos físicos não são destacadas.

Assim, neste trabalho propomos uma adaptação para o componente cápsula. A intenção é definir uma variação de uma cápsula que especifique que ela se trata de um componente mecânico. Esse novo tipo de cápsula deve então agregar informações inerentes a um componente físico, onde são solicitadas características gerais e relevantes ao seu projeto. Na Tabela 2 listamos algumas características que podem ser usadas para definir um componente mecânico.

Com as características disponibilizadas em um diagrama, é possível ao responsável pela parte mecânica uma compreensão integrada do sistema. Identificamos que o diagrama ideal para expor tais informações é o diagrama de estrutura, devido ao fato de que ele representa as cápsulas de forma semelhante à realidade de um componente físico. Observe que um componente pode ser composto de outros componentes, assim como cápsulas podem ser compostas de subcápsulas. As informações então seriam expostas ao selecionar uma cápsula que representa um componente físico.

Tabela 2. Características para componentes mecânicos.

Nome
Descrição da finalidade do componente
Se é pré-fabricado
Caso seja pré-fabricado, qual o fabricante
Se não for pré-fabricado, declarar seus sub-componentes
Dimensões e geometria do componente
Lista dos materiais do qual o componente é composto
Definir materiais: temperaturas máxima e mínima, módulo de elasticidade, dureza, tensão de ruptura, resistência ao desgaste e custo
Nível de tensão de entrada
Grandeza física gerada

Na Figura 6 tem-se uma ilustração de como seria o diagrama de estrutura do sistema com a extensão proposta. As cápsulas que são ou possuem componentes físicos possuem o símbolo de uma porca no canto inferior esquerdo. Ao entrar no diagrama de estrutura da cápsula que contém o símbolo, o projetista poderá verificar se a própria cápsula é um componente físico, ou se ela é composta de um ou mais componentes. A cápsula que representa efetivamente parte mecânica do projeto, inclui as características da cápsula, apresentadas na Tabela 2. No exemplo da Figura 6.a, é ilustrado o diagrama de estrutura da cápsula Sistema com a aplicação da extensão proposta. Na Figura 6.b, é exibido o interior da cápsula Câmera, e suas características físicas.

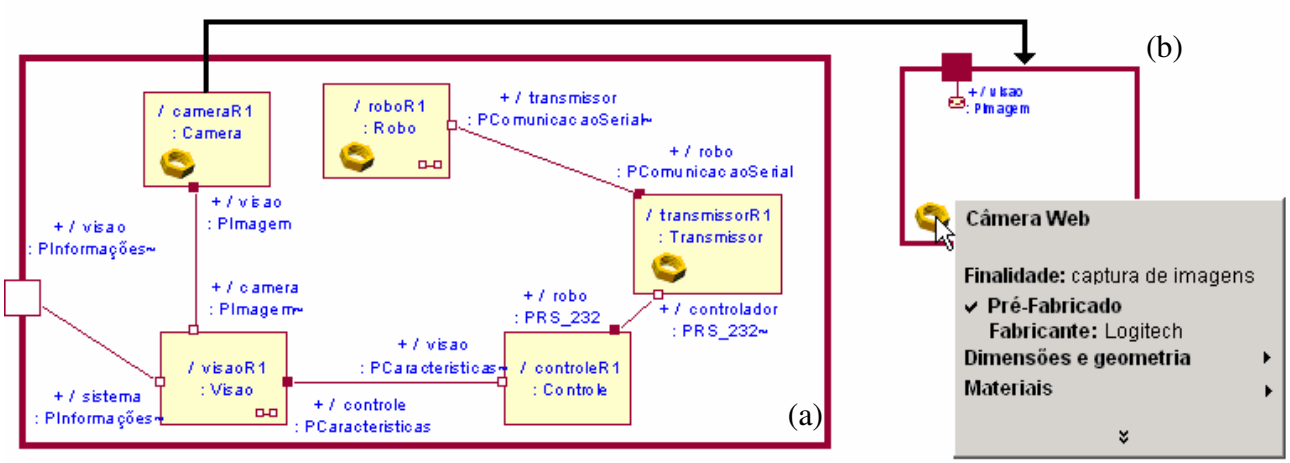

Figura 6. Exemplos de utilização do modelo proposto nas cápsulas (a) Sistema e (b) Câmera.

5 CONCLUSÕES

Considerando que sistemas mecatrônicos são sistemas de tempo real, este trabalho verifica a aplicabilidade de UML-RT na modelagem de sistemas mecatrônicos. Como não foram encontradas metodologias de projeto que utilizem a UML-RT como linguagem de modelagem de sistemas mecatrônicos, utilizou-se a metodologia apresentada em [12]. Esta metodologia foi desenvolvida com base na UML como linguagem de modelagem. A metodologia foi adaptada, fazendo-se uso das extensões introduzidas pela UML-RT nos momentos em que se fizeram necessárias. A metodologia se adaptou com facilidade à modificação, e se mostrou bastante satisfatória aos objetivos do projeto,

gerando artefatos que possibilitaram a compreensão sobre o sistema modelado.

A substituição da UML pela UML-RT na metodologia [12] aumentou o poder de representação das características reativas do sistema mecatrônico, capturando-as com mais fidelidade e oferecendo um modelo mais rico e consolidado para sua representação. A UML-RT simplificou a modelagem do sistema, e apresentou de forma bem definida seus componentes. Conceitos como cápsulas e portas foram aplicados de forma natural na análise do sistema e facilitaram bastante a modularização e a reprodução do comportamento do sistema, principalmente quando a comunicação é restrita entre estes módulos. Além disso, por ser uma linguagem visual, os membros da equipe da área de Engenharia não encontraram dificuldade em acompanhar a modelagem do sistema e beneficiaram-se da possibilidade de uma visão mais abstrata dos relacionamentos entre as partes lógica, elétrica e mecânica do sistema desenvolvido.

Durante a fase de modelagem surgiram alguns problemas, principalmente na modelagem física do sistema mecatrônico. Existem informações que não são relevantes à implementação lógica do sistema, mas que representam características importantes no projeto dos componentes mecânicos, e por isso devem constar no modelo. Geralmente as metodologias sugerem que tais componentes sejam modelados em uma linguagem específica. Em [11], por exemplo, é sugerido que o projeto de componentes seja feito no MATLAB [8] ou no Simulink [9].

Com o objetivo de atenuar tal deficiência da UML-RT, e prover uma forma de integrar melhor o projeto físico e o mecânico nesta linguagem de modelagem, propomos neste artigo uma adaptação para descrever de forma abstrata componentes mecânicos. Assim, o modelo também poderá ser útil para pessoas envolvidas na construção deste tipo de componentes.

Além da restrição na representação mecânica no sistema, percebemos limitações da linguagem para especificar requisitos de tempo, limitações estas também notadas por outros trabalhos que fizeram uso da UML-RT, como por exemplo em [15] e [4]. No entanto, como a restrição de tempo do estudo de caso não era complexa, tal problema não interferiu na sua modelagem.

Como trabalho futuro é proposto um maior refinamento no modelo construído no estudo de caso para que, com base neste, seja possível executar a implementação e testes do sistema de navegação. Por fim, é sugerida uma avaliação de outros sistemas mecatrônicos, a fim de encontrar mais características que possam ser adicionadas à adaptação da UML-RT proposta neste trabalho.

REFERÊNCIAS

[1] Andrade, B. T., Silva, L. e Freire, E. O. *Explorando UML-RT na Modelagem de Sistemas Mecatrônicos*. Trabalho de Conclusão de Curso. Departamento de Ciência da Computação e Estatística da Universidade Federal de Sergipe, maio de 2006.

[2] Bonfe, M. e Fantuzzi, C. Design and Verification of Mechatronic Object-Oriented Models for Industrial Control Systems. *Emerging Technologies and Factory Automation*, v. 2, pp. 253-260, setembro de 2003.

[3] Dymola. *Dynamic Modelling Laboratory*. Disponível na Internet via WWW. URL: http://www.dynasim.se. Última visita em abril de 2006.

[4] Gu, Z. e Shin, K. G. Synthesis of Real-Time Implementation from UML-RT Models. *2-nd RTAS Workshop an Model-Driven Embedded Systems*, maio de 2004.

[5] Heverhagen, T. e Tracht, R. Using Stereotypes of the Unified Modeling Language in Mechatronic Systems. *1st International Conference on Information Technology in Mechatronics*, pp. 333-338, outubro de 2001.

[6] IBM. *IBM Rational Rose RealTime Win Evaluation*. Disponível na Internet via WWW. URL: http://www14.software.ibm.com/webapp/download/search.jsp?rs=RATLe-ROSERTWIN-EVAL&S_TACT=104AHX11&S_CMP=TPRY. Última visita em abril de 2006.

[7] Jacobson I., Booch G. e Rumbaugh J. *The Unified Modeling Language*, Addison-Wesley, 1999.

[8] MathWorks. *MATLAB - The Language of Technical Computing*. Disponível na Internet via WWW. URL: http://www.mathworks.com/products/matlab/. Última visita em abril de 2006.

[9] MathWorks. *Simulink - Simulation and Model-Based Design*. Disponível na Internet via WWW. URL: http://www.mathworks.com/products/simulink/. Última visita em abril de 2006.

[10] Modelica. *Modeling of Complex Physical Systems*. Disponível na Internet via WWW. URL: http://www.modelica.org. Última visita em abril de 2006.

[11] Mrozek Z. Computer Aided Design of Mechatronic Systems. *International Journal of Applied Mathematics on Computer Science, n° 2*, v. 13, pp. 255-267, 2003.

[12] Mrozek, Z. Design of Mechatronic Systems with Help of UML Diagrams. *3-rd Workshop on Robot Motion and Control*, pp. 243-248, novembro de 2002.

[13] Okamoto Jr. J., Grassi Jr. V. e Corrêa F. R. Modeling Autonomous Mobile Robot System with an Object Oriented Approach. *Proceedings of the 17th International Congress of Mechanical Engineering*, novembro de 2003.

[14] Pressman R. S. *Software Engineering: A Practitioner's Approach*, McGraw-Hill, quarta edição, 1997.

[15] Ramos, R. T. *Desenvolvimento Rigoroso com UML-RT*. Dissertação de Mestrado. Centro de Informática da Universidade Federal de Pernambuco, abril de 2005.

[16] Rosario, J. M. *Princípios da Mecatrônica*. Prentice Hall, primeira edição, 2005.

[17] Selic B. e Rumbaugh J. Using UML for Modeling Complex Real Time Systems. *White Paper*. Rational Software Corporation. 1998.

[18] Selic B., Gullekson G. e Ward P. *Real-Time Object-Oriented Modeling*. John Wiley & Sons, primeira edição, 1994.