# MODELANDO SOFTWARE EMBARCADO COM UML2: UM ESTUDO DE CASO

**SIEGERT, Eliane[1]; PARADA, Abilio[1] G.; MARQUES, Milena[1] R. S.; BRISOLARA, Lisane[1] B. de**

[1]Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico - CDTec
{esiegert,agparada, mrsmarques, lisane}@inf.ufpel.edu.br

## 1 INTRODUÇÃO

Os desenvolvedores de software embarcado devem lidar com aplicações cada vez mais complexas e se preocupar em produzir um produto de qualidade, com baixo custo e com curto prazo de entrega. Portanto, há uma forte motivação na adoção de novas abordagens e metodologias para conseguir acelerar o processo de desenvolvimento de um software embarcado.

A engenharia orientada a modelos (MDE) [SELIC, 2006] oferece automação e a abstração ao projeto de sistemas. Este paradigma considera modelos como os artefatos centrais do projeto de software, sendo usados em todas as fases da engenharia, não somente na documentação. Modelos são abstrações, que facilitam a compreensão do sistema e auxiliam na construção do mesmo.

A linguagem UML [OMG, 2011] é a linguagem padrão para modelagem de software, oferecendo alto nível de abstração e uma série de diagramas que permitem representar diferentes visões do sistema. A última versão da linguagem, a UML 2, oferece muitos recursos, sobretudo para a modelagem do comportamento do sistema usando diagramas de sequência [BRISOLARA, 2010].

Este artigo apresenta um estudo de caso de modelagem de um software embarcado usando a linguagem UML2. A partir deste modelo, código orientado a objetos pode ser gerado com uso de ferramentas apropriadas, seguindo princípios da engenharia orientada a modelos [PARADA, 2011].

O trabalho está organizado da seguinte maneira. Na Seção 2 é apresentada a metodologia adotada para a construção dos modelos e realização do estudo de caso. A seção 3 apresenta o modelo construído, enquanto a Seção 4 discute os resultados alcançados e a Seção 5 apresenta as principais conclusões.

## 2 METODOLOGIA

Primeiramente, foi elaborado um diagrama de casos de uso, e, a partir desta visão funcional, foram construídos os demais diagramas (um diagrama de classes e os diagramas de sequência). O diagrama de classes especifica os principais elementos estáticos do sistema, como as classes e interfaces e seus relacionamentos. Para uma visão comportamental do sistema, vários diagramas de sequência foram usados e, através do Ref (chamada de referência em UML) os diagramas são interligados, mostrando o funcionamento global do sistema. Nestes diagramas, a troca de mensagens entre os métodos, loops, condicionais, ou referências a outros diagramas de sequência são usados para representar o comportamento esperado. A ferramenta gratuita chamada Papyrus [Papyrus 2011] foi usada para a construção dos diagramas.

## 3 ESTUDO DE CASO

O estudo de caso consiste na modelagem de um software embarcado para uma máquina de lavar roupas. O sistema foi escolhido principalmente por ser um exemplo simples de software embarcado e, portanto de fácil compreensão. Esta seção apresenta e discute os principais diagramas utilizados na modelagem do sistema. Por limitações de espaço, alguns diagramas utilizados foram omitidos.

A visão estrutural do sistema foi modelada usando o diagrama de classes ilustrado na Fig. 1. Basicamente, este modelo é composto por sete classes (*MaquinaDeLavar, Timer, OpcaoLavagem, Motor, SensorDePorta, SensorDeAgua* e *SensorDeTempo*), uma classe abstrata chamada *Sensor* e uma interface, *Maquina*. As classes *MaquinaDeLavar* e *Motor* implementam a interface *Maquina*. As classes *SensorDePorta*, *SensorDeTempo* e *SensorDeAgua* são subclasses da classe abstrata *Sensor*. Neste modelo também estão representadas as associações entre as classes *MaquinaDeLavar* com a *Timer*, *Motor*, *OpcaoDeLavagem* e *SensorDeAgua*. A *MaquinaDeLavar* é a classe principal do sistema. Esta classe possui o método *main,* que representa o ponto de partida do sistema, além dos outros métodos (lavar, enxaguar, centrifugar, encher, esvaziar) que representam as funcionalidades do sistema.

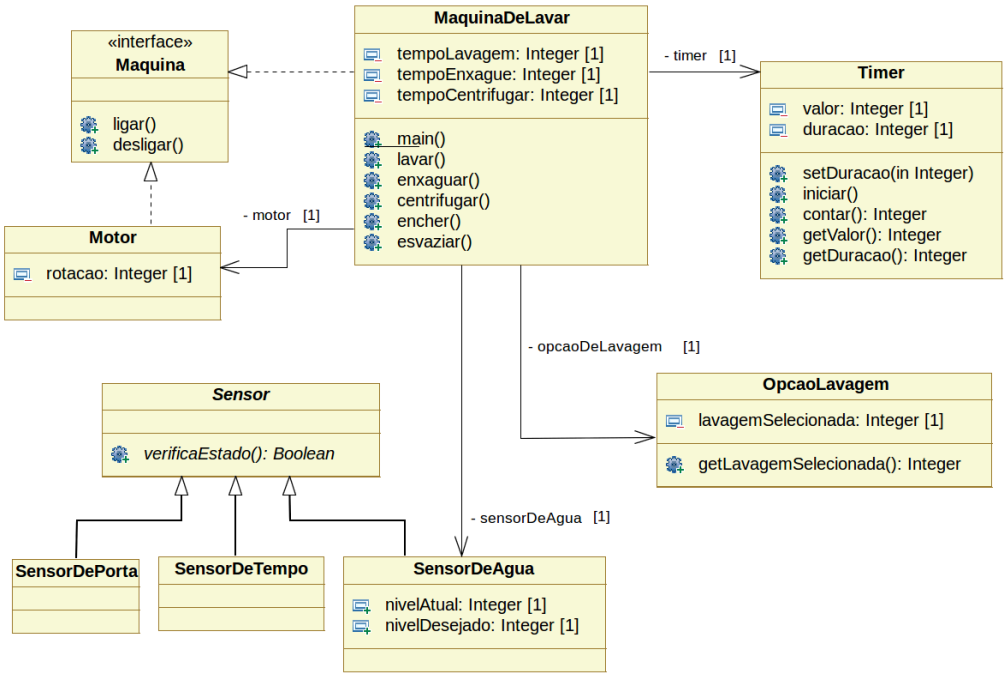


Figura 1 - Diagrama de classes - Visão estrutural da maquina de lavar

A partir do diagrama de classes, foram construídos os diagramas de sequência para representar o comportamento do sistema, definido por um conjunto de interações entre objetos. A Fig. 2 ilustra o diagrama de sequência (sd) *Principal* que representa o método *Main* da *MaquinaDeLavar*, com o nome *sd Principal* indicando que este diagrama corresponde a este método. Neste diagrama, a interação entre os objetos das classes *MaquinaDeLavar* e *OpcaoDeLavagem* estão representadas. Em primeiro lugar, a máquina de lavar deve identificar a operação selecionada. Para isso, o objeto *MaquinaDeLavar* invoca o método *getLavagemSelecionada* do objeto *OpcaoDeLavagem*. Este método retorna um valor inteiro usado para representar a opção selecionada. Então, o valor retornado é verificado pelo fragmento *alt* (recurso da UML2), que representa uma condicional com dois ou mais caminhos

possíveis, e determina qual das operações devem ser realizadas pela máquina. Se a opção 1 é selecionada, então, a operação realizada é LavagemPadrao, se a opção 2 é selecionada, então, a operação EnxagueDuplo é realizada e, se a opção 3 é selecionada a operação realizada é Centrifugar. A fim de evitar um diagrama complexo, operadores do tipo *ref* são usados para indicar que o comportamento detalhado de cada cenário está representado em outro diagrama de sequência.

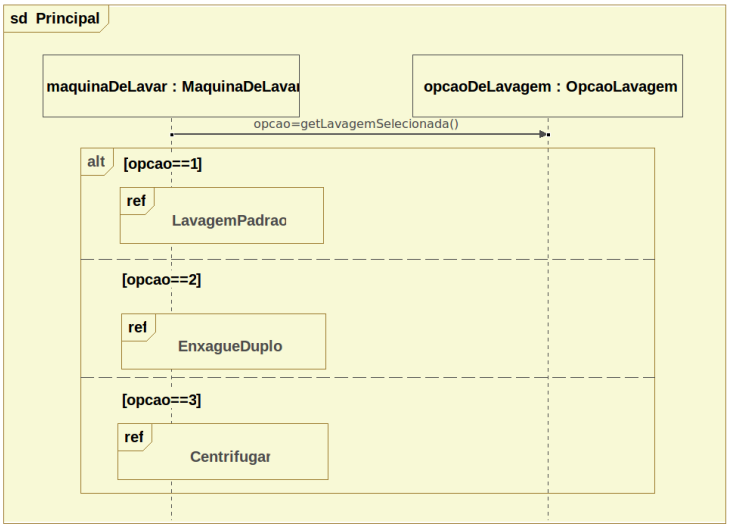


Figura 2 - Diagrama de sequencia – Visão comportamental do método principal do sistema

O diagrama ilustrado na Fig. 3 representa o comportamento detalhado do cenário correspondente ao *ref Centrifugar* da Fig. 2. Neste diagrama, a interação entre três objetos (*MaquinaDeLavar*, *Motor* e *Timer)* está representada. O cenário representado neste diagrama inicia com o objeto *MaquinaDeLavar* invocando o método *ligar()* do objeto *Motor*, seguido da invocação do método *setDuracao(tempoCentrifugar)* do objeto *timer*, passando o valor do tempo que a máquina deve ficar ligada. Depois disso, há o controle do tempo em que o motor deve ficar ligado, abstraído aqui pelo Ref *Periodo,* e detalhado em outro diagrama de sequência.

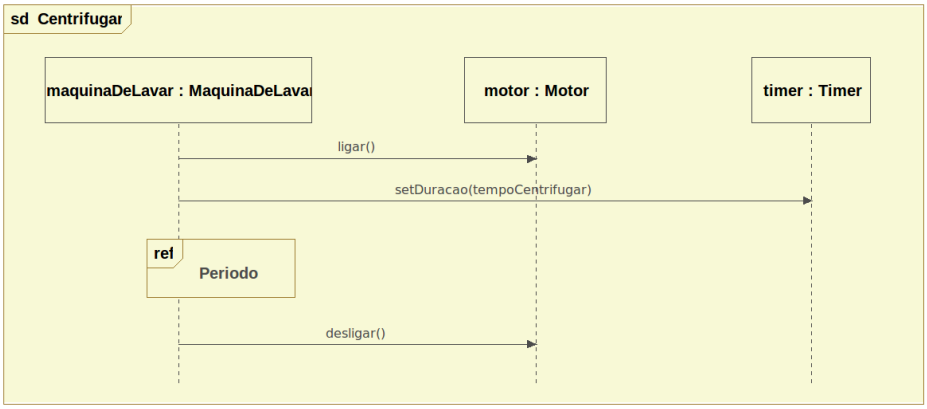


Figura3.Diagrama de sequencia - Visão comportamental da interação Centrifugar

Através das referências utilizadas nos diagramas, é possível representar todo o comportamento do sistema, no qual cada diagrama de sequência ilustra uma parte do comportamento do sistema e existe uma ligação bem definida entre todos os diagramas para que um fluxo global do sistema seja definido.

## 4 RESULTADOS E DISCUSSÃO

A UML provê um conjunto de notações de modelagem que permitem representar visões diferentes do sistema, assim, facilitando compreensão dos vários aspectos do projeto, bem como a detecção de erros. Além disso, se utilizada em conjunto com a engenharia orientada a modelos, modelos UML agilizam o desenvolvimento do sistema, visto que transformações de modelos podem ser adotadas para obter mais rapidamente o produto pronto. Contudo, a linguagem não permite especificar requisitos não funcionais, bem como para tratar requisitos temporais, é preciso usar estereótipos definidos em extensões da linguagem para sistemas de tempo-real, tais como o MARTE [OMG, 2010].

## 5 CONCLUSÃO

Este artigo apresenta um estudo de caso de modelagem de software para sistemas embarcados usando recursos suportados pela UML2. Este estudo de caso também primou por construir um modelo a partir do qual código automático pudesse ser gerado. O software usado no estudo de caso foi o controle de uma máquina de lavar usando basicamente diagramas de classes e de sequência. Como trabalhos futuros os autores pretendem realizar a modelagem do mesmo sistema usando o MARTE, uma extensão da UML para sistemas embarcados, a fim de avaliar os benefícios de seu uso.

## 6 REFERÊNCIAS

BRISOLARA, L.; KREUTZ, M. E.; CARRO, L. UML as front-end language for embedded systems design. In: GOMES, L.; FERNANDES, J. M. (Org.). **Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation**. Hershey: IGI Global, 2009. Cap. 1, p. 1-23.

PAPYRUS. (2011) Papyrus 1.12. Disponível em: <http://www.papyrusuml.org>. Acesso em Jan., 2011.

PARADA, A.; SIEGERT, E.; BRISOLARA, L. GenCode: a tool for generation of Java code from UML class model. In: **SIMPÓSIO SUL DE MICROELETRÔNICA**, 26., Novo Hamburgo, 2011. 26th South Symposium on Microelectronics, Proceedings... Novo Hamburgo: Feevale, 2011. p. 173-176.

SELIC, B. UML 2: A model-driven development tool. Model-Driven Software Development. **IBM Systems Journal**, Riverton, v. 45, n. 3, p. 607-620, 2006.

OMG. Object Management Group. UML 2. Disponível em: <http://www.omg.org>. Acesso em: Junho, 2011.

OMG. UML MARTE. Disponível em: <http://www.omgmarte.org/>. Acesso em: Dezembro, 2010.