GERAÇÃO AUTOMÁTICA DE ESPECIFICAÇÕES EXECUTÁVEIS EM SYSTEMC A PARTIR DE MODELOS SYSML

KEYLA GUIMARÃES MACHARET BRASIL

GERAÇÃO AUTOMÁTICA DE ESPECIFICAÇÕES EXECUTÁVEIS EM SYSTEMC A PARTIR DE MODELOS SYSML

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

ORIENTADOR: DIÓGENES CECÍLIO DA SILVA JÚNIOR

Belo Horizonte

Novembro de 2011

Macharet Brasil, Keyla Guimarães
M149g Geração automática de Especificações Executáveis em SystemC a partir de modelos SysML / Keyla Guimarães Macharet Brasil. — Belo Horizonte, 2011
xxiv, 143 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Diógenes Cecílio da Silva Júnior

1. Unified Modeling Language. 2. SoC Design. 3. SysML. 4. SystemC. 5. Model-driven Architecture. I. Título.

CDU 621.3(043)

Dedico este trabalho à minha família que sempre me apoiou.

Agradecimentos

Agradeço a Deus por ter me dado inteligência e sabedoria para vencer essa etapa tão importante na minha vida.

Agradeço à minha família por sempre me apoiar. Ao meus pais e avós por sempre me incentivarem desde muito cedo na busca constante do conhecimento. E ao meu irmão Douglas por me ajudar, dar carona, tirar dúvidas de LaTeX, fazendo com que o processo fosse um pouco mais fácil.

Agradeço especialmente ao meu marido Timóteo, que me apoiou em todos os momentos e nunca me deixou desistir. Agradeço o carinho e a compreensão nos momentos em que todo o meu tempo era gasto com os estudos. Sem você eu não teria chegado até aqui.

Ao meu orientador, Professor PhD. Diógenes Cecílio da Silva Jr, agradeço pela paciência, confiança e por todos os ensinamentos.

Este espaço é dedicado a todos aqueles que de alguma forma contribuíram para que este desafio fosse superado. A todos eles, deixo aqui meu sincero agradecimento.

"Adquire a sabedoria, adquire o entendimento;
não te esqueças nem te desvies das palavras da minha boca."
(Provérbios 4:5)

Resumo

A modelagem de sistemas embutidos é atualmente uma importante área de pesquisa e o projeto de sistemas se torna crítico à medida que a tecnologia de implementação avança na direção de circuitos integrados cada vez mais complexos e com a pressão para redução do tempo para lançamento do produto no mercado. Devido a esses problemas reconhecemos a necessidade do desenvolvimento de metodologias para reduzirem o custo e o tempo gasto durante as fases do projeto e desenvolvimento de sistemas. SysML, System Modeling Language - linguagem de modelagem de sistemas, é uma notação dedicada no nível de sistema baseada na UML e proposta pela OMG. Esta notação está ganhando bastante importância nos últimos anos como padrão para projetos no nível de sistema, pois é uma linguagem independente de plataforma, que permite o projeto de um sistema sem o conhecimento de detalhes de implementação.

Neste trabalho está proposta uma ferramenta de apoio a uma metodologia para tradução automática de alguns diagramas SysML (Diagrama de Definição de Bloco, Diagrama Interno de Bloco, Diagrama de Transição de Estados e Diagrama de Sequência) para uma especificação executável em SystemC e a demonstração do uso através de exemplos. A abordagem proposta tem algumas vantagens, tais como: menor custo devido à mudanças no projeto, documentação mais fácil e melhor entendimento do projeto pelos interessados.

A ferramenta desenvolvida, **SysMLToSystemC**, foi validada através da aplicação em três sistemas de naturezas diferentes, sendo dois sistemas orientados a processamento de dados e um orientado a controle. Em todos os três casos, o comportamento da especificação executável gerada automaticamente com o uso da ferramenta desenvolvida foi igual ao comportamento do sistema original ou de acordo com o esperado - como no caso no qual não havia um sistema original sendo executado para comparação.

Palavras-chave: UML, Linguagem de Modelagem Unificada, Projeto de Sistemas Embutidos, SysML, SystemC, Arquitetura Dirigida a Modelos.

Abstract

Embedded system modeling is an important area of research nowadays, and system design become critical as implementation technology progress toward complex integrated circuits and the time-to-market pressure continues to grow. Due to those problems we recognize the necessity to develop methodologies to reduce cost and time spent during all the design phases and system development. SysML, System Modeling Language, is a dedicated system level UML-based notation proposed by OMG. The notation is gaining a lot of importance as a system level design standard because it is a platform independent language, which makes possible the design of systems without knowing implementation details.

In this work we propose a tool to support a methodology for automatic translation of some SysML Diagrams (Block Definition Diagram, Internal Block Diagram, State Machine Diagram and Sequence Diagram) to an executable specification in SystemC and demonstrate the usage through a few examples. The proposed approach have some advantages, such as: smaller cost due to project changes, easier documentation and better understanding of the project by stakeholders.

The created tool, **SysMLToSystemC**, was validated by its usage in three systems of different natures: two data-oriented systems and one control-driven system. In all three cases, the behavior of the executable specification, automatically generated by the developed tool, was similar to the behavior of the original system or as expected - which is the case of one of the systems with no original source code for comparison.

Keywords: Unified Modeling Language, SoC Design, SysML, SystemC, Model-driven Architecture.

Lista de Figuras

2.1 Níveis de Modelagem [Grötker et al., 2002]. 10
2.2 Uso das linguagens [Black et al., 2009]. 12
2.3 Arquitetura da linguagem SystemC [Initiative, 2005]. 14
2.4 Modelos de Abstração [Cai & Gajski, 2003]. 16
2.5 Relação entre UML 2 e SysML [SysML, 2006]. 19
2.6 Diagramas SysML e relação deles com UML 2 [SysML, 2006]. 20
2.7 Notação utilizada para Blocos. 21
2.8 Notação utilizada para Blocos. 22
2.9 Mundo de Newton [Friedenthal et al., 2008]. 22
2.10 Notação utilizada para Diagrama Interno de Bloco. 23
2.11 Biblioteca de restrições do mundo de Newton [Friedenthal et al., 2008]. . . 24
2.12 Utilização do Bloco de Restrição num Diagrama Paramétrico [Friedenthal et al., 2008]. 25
2.13 Alocação de atividades para partes. 25

3.1 Procedimento de Mapeamento SysML para SystemC. 28
3.2 Ferramenta Altova UModel. 29
3.3 Menu para exportação XMI. 31
3.4 Menu do plugin **SysMLToSystemC**. 32
3.5 Plugin **SysMLToSystemC**. 33
3.6 Diagrama de Classes representando os Blocos e suas informações. 33
3.7 Diagrama de Classes representando os Diagramas de Transição de Estados. 34
3.8 Diagrama de Classes representando os Diagramas de Sequência. 34
3.9 Modelo de Testbench. 36
3.10 Adicionando código fonte para operação a partir de arquivo. 37
3.11 Salvando arquivo texto com operação de leitura de dados. 38
3.12 Estrutura do Arquivo XML Intermediário. 39

4.1 Diagrama de Definição de Bloco Pipeline. 47
4.2 Opções escolhidas para geração da Especificação Executável. 48
4.3 Adicionando código para função **numgen()**. 48
4.4 Diagrama Interno de Bloco Calculator. 53
4.5 Diagrama de Transição de Estados Calculator. 53
4.6 Seleção de opções para geração da Especificação Executável SystemC. . . . 54
4.7 Diagrama de Definição de Bloco Máquina de Lavar. 59
4.8 Diagrama Interno de Bloco Máquina de Lavar. 60
4.9 Opções escolhidas para geração da Especificação Executável da Máquina de Lavar. 61
4.10 Diagrama de Sequência Máquina de Lavar. 62

A.1 Criação do Plugin para Altova UModel 77
A.2 Código fonte para abertura da janela do Plugin 78
A.3 Processo de instalação do Plugin 80
A.4 Confirmação de Plugin instalado com sucesso 81

B.1 Arquivos de saída gerados para o sistema Calculator 103

Lista de Tabelas

4.1 Comparação Pipeline Original x Especificação Executável. 50

Lista de Abreviações

API	Application Programming Interface
CI	Circuitos Integrados
DUV	Device Under Validation
IEEE	Institute of Electrical and Electronic Engineers
INCOSE	International Council on Systems Engineering
MDA	Model Driven Architecture
MDE	Model-driven engineering
MDT	Model Development Tools
MOF	Meta-Object Facility
OMG	Object Management Group
OSCI	Open SystemC Initiative
PIM	Platform Independent Model
PSM	Platform Specific Model
RFP	Request for Proposal
RTL	Register transfer level
SAM	System Architectural Model
SoC	System on a Chip
SysML	Systems Modeling Language
TLM	Transaction Level Modeling
UML	Unified Modeling Language
XMI	XML Metadata Interchage

Sumário

Agradecimentos **ix**

Resumo **xiii**

Abstract **xv**

Lista de Figuras **xvii**

Lista de Tabelas **xix**

Lista de Abreviações **xxi**

1 Introdução **1**

1.1 Modelagem em SysML ou UML 3
1.2 Motivação 4
1.3 Objetivo 4
1.4 Organização do Documento 5

2 Revisão da Literatura **7**

2.1 Arquitetura Dirigida pelo Modelo 9
2.1.1 Modelo Independente de Plataforma (PIM) 9
2.1.2 Modelo Específico de Plataforma (PSM) 9
2.2 Níveis de Modelagem 10
2.2.1 Especificação Executável 11
2.3 SystemC 12
2.3.1 Estrutura da Linguagem 13
2.3.2 Transaction Level Modeling (TLM) 15
2.3.3 SystemC TLM 17
2.4 SysML 18
2.4.1 Arquitetura da Linguagem 18

2.4.2 Diagramas SysML 18
2.4.3 Ferramentas de Modelagem 24
2.4.4 Benefícios da utilização do SysML 26

3 Metodologia de Mapeamento de SysML para SystemC 27
3.1 Altova UModel 28
3.2 Metodologia Detalhada 29
3.2.1 1ª Etapa: Modelagem em Diagramas SysML 30
3.2.2 2ª Etapa: Exportação de Diagramas SysML como arquivo XMI 31
3.2.3 3ª Etapa: Importação de arquivo XMI para ferramenta **SysML-ToSystemC** 32
3.2.4 4ª Etapa: Escolha de opções relativas ao SystemC a ser gerado . 35
3.2.5 5ª Etapa: Geração de Especificação Executável em SystemC . . 38

4 Estudos de Caso 45
4.1 Pipeline 46
4.1.1 Comparação com o sistema original 50
4.2 Calculadora 53
4.2.1 Resultado da Execução 58
4.3 Máquina de Lavar 59
4.3.1 Resultado da Execução 66

5 Conclusão 67
5.1 Contribuições do Trabalho 68
5.2 Trabalhos Futuros 69

Referências Bibliográficas 73

Apêndice A Desenvolvimento do *Plugin* para ferramenta Altova UModel 77
A.1 Instalação do *Plugin* na ferramenta Altova UModel 79

Apêndice B Arquivos de Estilo XSLT 83
B.1 Tutorial XSLT 87

Anexo A Arquivos de saída do Pipeline 91

Anexo B Arquivos de saída da Calculadora 103

Anexo C Arquivos de saída da Máquina de Lavar 115

Capítulo 1

Introdução

O projeto de sistemas tem se tornado crítico devido a evolução da tecnologia de implementação e o aumento da complexidade dos circuitos integrados (CIs). CIs complexos passaram a integrar a maioria dos elementos funcionais de um produto final em um único chip de sistema (System on Chip - SoC) [Badawy & Jullien, 2003]. Neste trabalho será utilizada a definição de SoC para representar CIs complexos que podem consistir de um ou mais núcleos programáveis, através de barramentos locais, e eventualmente até de unidades de memória.

Com a utilização em massa de SoCs no domínio das telecomunicações, multimídia e eletrônicos comuns [Benini & De Micheli, 2002], o desenvolvimento destes enfrenta a pressão contínua para menor tempo de colocação de produto no mercado [Keutzer et al., 2000]. Devido a estes fatores, novos métodos tem sido desenvolvidos na tentativa de diminuir custo e tempo gasto nas etapas de modelagem e desenvolvimento do projeto. A reutilização de código é uma maneira efetiva de reduzir custo e tempo de desenvolvimento, e é de grande importância já que o tempo e o custo de um projeto são fatores principais no processo de tomada de decisão de projetistas de sistema [Keutzer et al., 2000]. Além disso a eficiência, confiabilidade e robustez são essenciais para o projeto, modelagem, implementação e desenvolvimento de sistemas concorrentes.

A Arquitetura Dirigida pelo Modelo (MDA - Model Driven Architecture) [Miller et al., 2001] é uma metodologia para desenvolvimento de software criada pelo Object Management Group (OMG) [Object Management Group, 2010]. O ponto chave para essa metodologia é a importância da modelagem no processo de desenvolvimento do software. Nessa metodologia se inicia o desenvolvimento com um modelo abstrato do sistema e após um processo de refinamento dos modelos é gerada a representação concreta, um código fonte, do sistema de software.

A Arquitetura Dirigida pelo Modelo pode ser dividida em três etapas:

1. Construção de um modelo com um alto nível de abstração, independente de qualquer tecnologia. Esse modelo é chamado de Modelo Independente de Plataforma (PIM, Platform Independent Model). Neste trabalho será utilizada a linguagem SysML [SysML, 2008], Systems Modeling Language, para a descrição do modelo.

2. A segunda etapa é a transformação do PIM em um ou mais Modelos Específicos de Plataforma (PSM, Platform Specific Model). Um PSM é mais específico para o sistema em termos de tecnologia de implementação. Para validar o modelo PIM é utilizada uma linguagem de modelagem específica de plataforma, que neste trabalho será o SystemC [Iniciative, 2010].

3. A terceira e última etapa é transformar um PSM em código.

A transformação de *Unified Modeling Language* (UML) para SystemC tem sido alvo de vários trabalhos hà muito tempo [Prevostini & Zamsa, 2007]. Como o padrão SysML é relativamente recente, pode-se dizer que a transformação de SysML para SystemC é algo novo e ainda pouco pesquisado e será alvo deste trabalho. A geração automática de código fonte em SystemC a partir de uma especificação em SysML irá beneficiar o ciclo de vida dos projetos de sistema das seguintes maneiras [Raslan & Sameh, 2007b]:

- Preencherá o espaço de abstração entre a especificação e o ciclo de projeto no nível de transferência de registro (RTL - *Register transfer level*)[1].
- Aumentará a produtividade do projeto.
- Criará uma especificação executável.
- Diminuirá o espaço entre projetistas de software e hardware, melhorando também a integração entre software e hardware e diminuindo a chance de problemas.

[1]Projetos RTL são ainda o ponto inicial do desenvolvimento de sistemas digitais integrados, tais como microcontroladores e microprocessadores. Neste tipo de projeto, o comportamento do circuito é descrito em termos do fluxo de sinais ou transferência de dados entre os registradores presentes no hardware e as operações lógicas conduzidas com estes sinais. Embora seja um projeto de hardware, sua descrição é feita através de linguagens de descrição de hardware, ou seja, via software. As linguagens mais utilizadas são o VHDL e o Verilog. A descrição RTL é usualmente convertida para a descrição de circuitos em *gate-level* por uma ferramenta de síntese lógica. Esta síntese resultante representa a descrição física do circuito.

1.1 Modelagem em SysML ou UML

A UML é uma linguagem de modelagem estabelecida no campo de desenvolvimento de software, é um padrão internacional especificado pelo OMG e também é aceito como padrão ISO (ISO/IEC 19501) [Weilkiens, 2007]. Apesar disso, a linguagem apresentava falhas na utilização em sistemas de engenharia. Entre os problemas observados havia a falta de diagramas específicos para a modelagem de requisitos. Uma outra dificuldade ocorria pelo fato da UML ter sido desenvolvida fortemente caracterizada pela orientação a objetos específicos para o desenvolvimento de software ao passo que a modelagem dos sistemas de engenharia é interdiciplinar.

Várias iniciativas de padronizar os processos de engenharia de sistemas surgiram até o inicio dos anos 2000 mas nenhuma linguagem de modelagem havia sido criada, o que dificultava a comunicação em projetos multidiciplinares e exigia a utilização de diversas ferramentas diferentes. Em 2001, o International Council on Systems Engineering (INCOSE) e o OMG decidiram criar um padrão de linguagem UML específica para sistemas de engenharia. A linguagem UML foi escolhida, pois atendia aos requisitos, já era amplamente utilizada e aceita, possuía muitas ferramentas de modelagem e poderia ser adaptada para a nova necessidade. Sua propriedade de extensão através do mecanismo de estereótipos permite a definição e adaptação para um domínio específico. As extensões mais importantes para a criação da linguagem SysML são:

- As classes em UML são chamadas de *blocos* em SysML e o diagrama de classe passou a se chamar *diagrama de definição de bloco*. O diagrama de estrutura composta agora se chama *diagrama interno de bloco* em SysML.
- Dois novos tipos de diagramas foram criados: diagrama paramétrico e diagrama de requisitos.
- Atende ao padrão ISO AP-233, um protocolo de troca de dados para sistemas de engenharia entre diferentes ferramentas.
- Omissão de alguns elementos UML específicos para desenvolvimento de software.

1.2 Motivação

A medida que a tecnologia de implementação de sistemas embutidos avança na direção de circuitos integrados cada vez mais complexos e com a pressão para redução do tempo para lançamento do produto no mercado, a utilização de metodologias para reduzir o custo e o tempo gasto durante as fases do projeto e desenvolvimento de sistemas se torna necessária. A metodologia utilizada trará as seguintes vantagens:

- Facilidade e rapidez na detecção de erros, que podem ser corrigidos nas primeiras etapas do projeto.
- Alteração no projeto com menor custo.
- Redução do tempo gasto entre as etapas de concepção, validação e teste do sistema.
- Facilidade de documentação do sistema e reutilização de módulos desenvolvidos.
- Melhor entendimento do sistema por todos os envolvidos no projeto.
- Aumento da qualidade na interação com o cliente.

1.3 Objetivo

O objetivo deste trabalho é elaborar uma ferramenta de apoio a uma metodologia de desenvolvimento de SoCs que reduza o esforço gasto pelo projetista nas etapas de concepção, modelagem e validação do projeto e que reduza também a possibilidade de erros sintáticos, devido à falta de conhecimento da linguagem SystemC pelo projetista.

A proposta é partir de uma especificação do SoC em alto nível, através de sua descrição gráfica *SysML* e chegar a um modelo de referência executável, com a geração de uma especificação executável em SystemC.

Como consequência deste trabalho foi desenvolvida uma ferramenta de nome **SysMLToSystemC**, para tradução automatizada de descrições *SysML* para código SystemC. A ferramenta escrita em C♯.NET, é um *plugin* que deve ser instalado e utilizado com a ferramenta *Altova UModel* [UModel, 2011] - a ser detalhada adiante - de modelagem *SysML*.

1.4 Organização do Documento

Os capítulos desse trabalho estão organizados da seguinte forma:

- No Capítulo 2 é apresentada uma revisão bibliográfica contendo diversas propostas de metodologias para a modelagem e projeto de sistemas embutidos usando SysML, UML e SystemC desenvolvidas nos últimos anos. São apresentados também os conceitos e as definições básicas para o desenvolvimento dos capítulos seguintes.

- No Capítulo 3 é apresentada a proposta de uma metodologia para geração de código executável SystemC a partir de um modelo de sistema desenhado em SysML. Nesse capítulo também é apresentada a ferramenta proposta neste trabalho bem como as ferramentas utilizadas no seu desenvolvimento, implementação e funcionamento.

- O Capítulo 4 mostra a aplicação da metodologia e da ferramenta desenvolvida em alguns estudos de caso, assim como uma avaliação dos resultados obtidos.

- No Capítulo 5 são apresentadas as considerações finais e algumas propostas de tarefas futuras para este trabalho.

Capítulo 2

Revisão da Literatura

Uma das primeiras tentativas de utilização de UML no projeto de SoCs foi feita por Pauwels [Pauwels et al., 2004] em 2004 e apresentou alguns benefícios do uso combinado de modelos UML e SystemC para documentação e especificação de sistemas. Na tentativa de utilizar UML no projeto de software e hardware, Riccobene [Riccobene, 2005] começou a extender a linguagem de modelagem UML para permitir a modelagem de um modelo SystemC em UML. Essa extensão foi feita através do mecanismo de perfis UML que adicionam uma coleção de notações específicas de um domínio através de estereótipos. Naquele trabalho foi adicionada a capacidade de geração de código SystemC a partir de um modelo UML utilizando a capacidade de geração de *script* da ferramenta de modelagem UML utilizada e também através da exportação e formatação de modelos XML Metadata Interchage (XMI) [OMG, 2007]. Riccobene [Riccobene et al., 2005] também apresentou um perfil UML 2.0 para a linguagem SystemC explorando as capacidades de definir linguagens de modelagens da arquitetura dirigida a modelos, independente de plataformas. Xi [Xi et al., 2005] também tem uma abordagem semelhante. Em Boutekkouk [Boutekkouk, 2010] pode ser encontrada uma abordagem para geração automática de código SystemC em estágios iniciais de projetos de SoC a partir de diagramas UML de sequência e de atividade.

Outra forma de combinar UML e SystemC foi abordada por Andersson [Andersson & Höst, 2008] com a transformação automática entre UML e SystemC sendo feita baseada em regras de mapeamento. Em Kreku [Kreku et al., 2007], o mapeamento entre a modelagem UML e o código em SystemC foi definido de forma que as regras de transformação permitem a geração de código parcialmente automática.

Partindo das tentativas iniciais, um perfil UML específico para modelagem de sistemas (SysML - Systems Modeling Language) foi desenvolvido como um padrão oficial OMG em Julho de 2006 [SysML, 2006]. Durante o ano seguinte, alguns trabalhos

de mapeamento de diagramas SysML para SystemC com geração automática de código fonte foram desenvolvidos, os quais podem ser citados Raslan [Raslan & Sameh, 2007b] e Prevostini [Prevostini & Zamsa, 2007]. Raslan [Raslan & Sameh, 2007b] descreve uma abordagem para tradução de diagramas estruturais e comportamentais SysML mas não mostra exemplos do código SystemC gerado. Prevostini [Prevostini & Zamsa, 2007] traduz o Diagrama de Definição de Bloco e o Diagrama Interno de Bloco criando somente a estrutura dos módulos e utiliza o Diagrama de Atividade somente para alocar operações aos seus respectivos blocos.

Nos anos de 2006 e 2007, dois trabalhos no tópico da geração de código a partir de uma modelagem do sistema foram desenvolvidos no laboratório LabSCI/UFMG [Papa, 2006] [DIAS, 2007]. A principal diferença deste trabalho em relação àqueles está na utilização da modelagem em SysML, quando ambos utilizaram UML. O código gerado pelo trabalho de Papa [Papa, 2006] utiliza *templates*. Os códigos gerados na forma de *template* ficam maiores e menos legíveis para o projetista. Foi analisado também que os códigos na forma de *template* não funcionam corretamente quando são gerados utilizando a separação de arquivos .h e .cpp. Devido a esses motivos, a ferramenta **SysMLToSystemC** não faz a geração da especificação executável utilizando *templates*. O trabalho SRD desenvolvido por Dias [DIAS, 2007] é focado no refinamento sucessivo do código em SystemC, tópico que não é alvo deste trabalho. O Diagrama de Sequência foi utilizado pelo primeiro para obter as conexões e comunicações entre blocos, ao passo que neste trabalho essas informações são obtidas através dos diagramas estruturais SysML. E nenhum deles utilizou o Diagrama de Transição de Estados para descrever o comportamento de processos.

Este trabalho apresenta uma ferramenta que utiliza a modelagem de um sistema através dos diagramas estruturais SysML - Diagrama de Definição de Bloco e Diagrama Interno de Bloco - e dos diagramas comportamentais de Transição de Estados e de Sequência para geração automática de uma especificação executável SystemC. Os módulos SystemC são gerados com suas portas e processos. Nesta abordagem os diagramas de Transição de Estados são utilizados para descrever processos orientados a controles, podendo associá-los ao comportamento de uma operação de um bloco e automaticamente gerando código SystemC associado. O Diagrama de Sequência tem seus dados salvos no arquivo que contém a descrição do sistema para ser utilizado em trabalho a ser desenvolvido posteriormente. O Diagrama de Sequência também foi utilizado para confirmar os processos a serem chamados em cada módulo. A ferramenta também gera o arquivo principal SystemC e a função $sc\_main()$ com a inicialização dos módulos SystemC, criação dos canais para conexão entre módulos e associação entre portas e canais de conexão.

2.1 Arquitetura Dirigida pelo Modelo

No final de 2001, o OMG lançou o modelo MDA (Model-Driven Architecture) como abordagem na utilização de modelos no desenvolvimento de software. A Arquitetura Dirigida pelo Modelo, MDA, teve início na idéia de separar a especificação de operação de um sistema dos detalhes de como o sistema usa as capacidades da plataforma [Miller et al., 2003].

Os três objetivos principais da MDA são portabilidade, interoperabilidade e reuso, através da separação arquitetural. MDA provê uma abordagem que possibilita:

- Especificação de sistemas independentes de plataforma;
- Especificação de plataformas;
- Escolha de uma plataforma específica para um sistema;
- Transformação de uma especificação de sistema para uma plataforma específica.

O foco do trabalho descrito nesta dissertação será baseado nesse último ponto, onde será feita a transformação de uma especificação de sistema para uma plataforma específica. Alguns conceitos básicos utilizados nessa arquitetura serão detalhados nas seções 2.1.1 e 2.1.2.

2.1.1 Modelo Independente de Plataforma (PIM)

Um modelo independente de plataforma, cuja sigla PIM refere-se a *Platform Independent Model*, é um modelo cuja visão é focada na operação de um sistema, escondendo detalhes de uma plataforma específica. Essa visão mostra partes da especificação que não mudam de uma plataforma para outra. Um PIM é descrito em um grau de independência tal da plataforma que pode ser adequado para o uso em várias plataformas diferentes que sejam similares.

O modelo independente da plataforma deve ser descrito por uma linguagem de modelagem de propósitos gerais ou em uma linguagem específica da área na qual o sistema especificado será usado. Neste trabalho o PIM será descrito utilizando a linguagem de modelagem SysML descrita na Seção 2.4.

2.1.2 Modelo Específico de Plataforma (PSM)

Um modelo específico de plataforma, cuja sigla PSM refere-se a *Platform Specific Model*, é um modelo cujo ponto de vista combina a visão independente da plataforma com um foco adicional nos detalhes de como o sistema utilizará a plataforma específica.

Um PSM combina a especificação no modelo PIM com os detalhes que especificam como aquele sistema utiliza um tipo específico de plataforma.

2.2 Níveis de Modelagem

Utilizando linguagens baseadas em C e C++, como SystemC, pode-se utilizar uma metodologia de refinamentos sucessivos para o desenvolvimento de um projeto de sistema embutido. Esta metodologia consiste em partir de uma especificação executável e refinar o projeto agregando detalhes em cada nível até um nível que possibilite a síntese. Estes detalhes referem-se, por exemplo, a temporização ou comunicação.

Grötker et al [Grötker et al., 2002] separa os níveis de modelagem conforme pode ser visto na Figura 2.1.

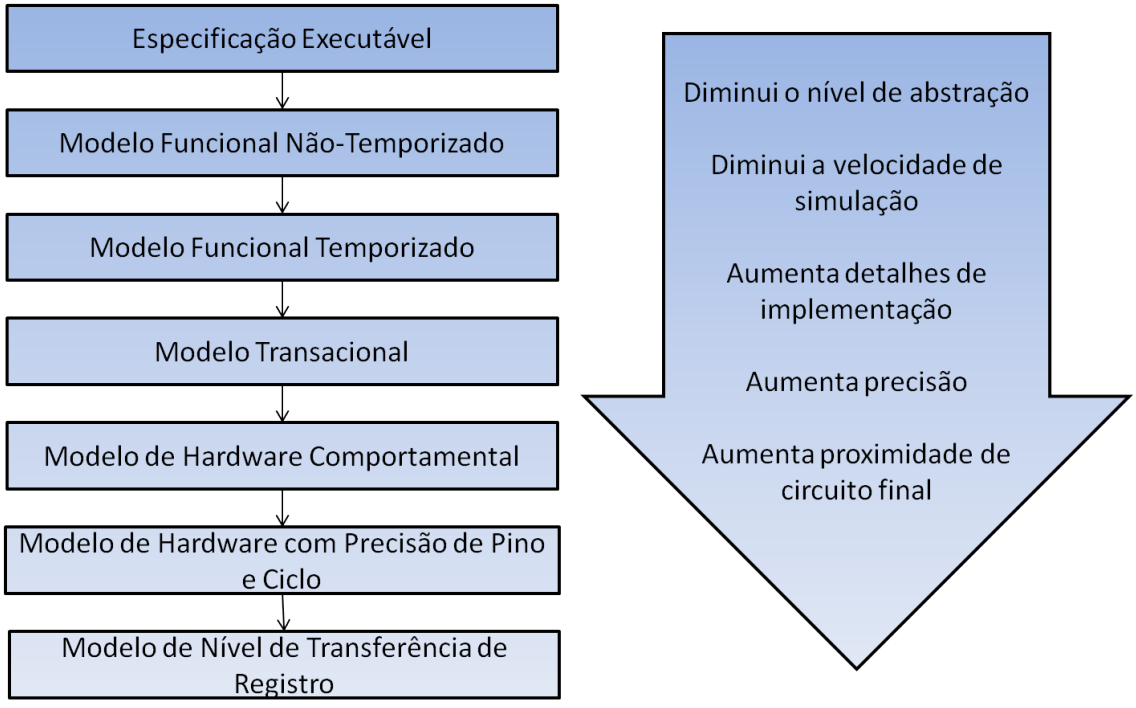


Figura 2.1: Níveis de Modelagem [Grötker et al., 2002].

Neste trabalho será gerada uma especificação executável em SystemC, primeiro nível de modelagem. O código fonte gerado e um arquivo XML com todas as informações será utilizado posteriormente, por outro trabalho, para a realização de refinamentos sucessivos.

2.2.1 Especificação Executável

A especificação executável é uma descrição comportamental cujo propósito principal é permitir a verificação de um determinado comportamento do sistema [Bailey et al., 2005]. Para as primeiras simulações os projetistas usam esse modelo com fluxo de sinal contínuo. Este modelo de computação assume que não há ordem de execução nem períodos de tempo pré-definidos.

SystemC é uma linguagem amplamente aceita pela comunidade de projetos e provê bibliotecas para o desenvolvimento de especificações executáveis que podem ser utilizadas para simulações rápidas, exploração de algoritmos variados, validação e otimização do projeto. A especificação executável é essencialmente um programa C++ acrescido de mecanismos de sincronização entre processos e um simulador de processos concorrentes, que exibe o mesmo comportamento do sistema real quando executado.

Existem vários benefícios na criação de especificações executáveis precisas de seu sistema complexo no início do fluxo de projeto. Entre eles [Initiative, 2002]:

- Evita inconsistências e erros, e ajuda a assegurar que a especificação esteja completa, já que, ao criar a especificação executável, está sendo criado um programa que deve se comportar da mesma forma que o sistema.

- Garante a não ambiguidade de interpretação da especificação. Sempre que houver dúvidas, a especificação executável pode ser executada para confirmar se o sistema está fazendo o que era esperado.

- Ajuda a validar as funcionalidades do sistema antes da implementação ser iniciada.

- O *testbench* utilizado para testar a especificação executável pode ser refinado e utilizado para testar a implementação da especificação, reduzindo o tempo gasto em verificação.

2.3 SystemC

Apesar de ser conhecida como uma linguagem, SystemC é um conjunto de classes C++, de código aberto, desenvolvidas pelo grupo *Open SystemC Initiative* [Iniciative, 2010] e aprovada pelo *Institute of Electrical and Electronic Engineers (IEEE) Standards Association* como o padrão IEEE 1666 [Initiative, 2005] em 2005.

Um dos principais objetivos do SystemC é permitir a modelagem no nível de sistema, isto é, permitir a modelagem de sistemas num nível de abstração mais alto do que o nível de registradores, incluindo sistemas que podem ser desenvolvidos em software, hardware ou numa combinação dos dois [Grötker et al., 2002]. Várias linguagens surgiram para tratar os diversos aspectos do projeto de sistemas. C e C++ são as mais utilizadas atualmente para o desenvolvimento de software em mais baixo nível em projetos de sistemas embutidos. As linguagens de descrição de hardware VHDL e Verilog são utilizadas para simulação digital de circuitos. Mais recentemente a linguagem SystemVerilog surgiu para facilitar a verificação funcional de circuitos complexos. Outras linguagens e ferramentas, como MATLAB, por exemplo, são utilizadas para capturar requisitos do sistema e desenvolver algoritmos de processamento de sinais [Black et al., 2009]. Na Figura 2.2 pode ser visto o campo de atuação de cada linguagem ou ferramenta, mas, no comparativo, SystemC é a mais abrangente.

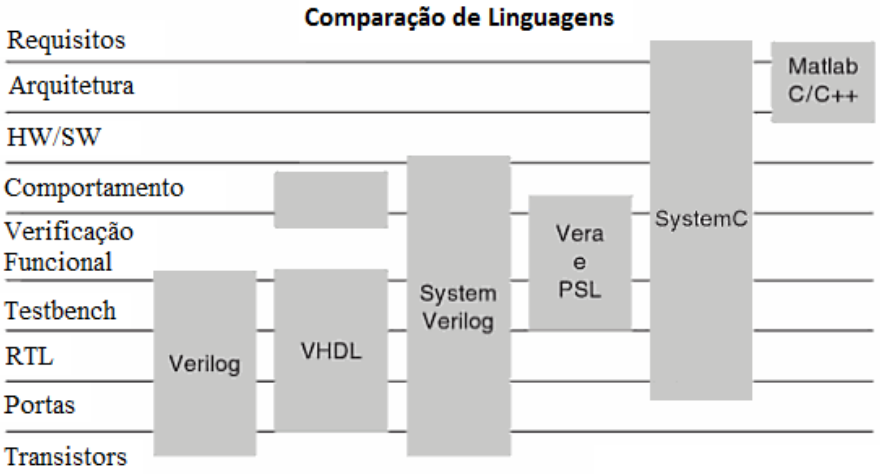


Figura 2.2: Uso das linguagens [Black et al., 2009].

SystemC é aplicada à modelagem no nível de sistema, exploração arquitetural, modelagem de performance, desenvolvimento de software, verificação funcional, síntese em alto nível e síntese RTL.

2.3.1 Estrutura da Linguagem

As classes possuem uma estrutura de programação intuitiva e de fácil aprendizado. Algumas características a serem ressaltadas são a existência de um núcleo exclusivo para simulações, suporte à concorrência e a hierarquia de módulos. A possibilidade de descrever e testar módulos em vários níveis de abstração também pode ser vista como uma vantagem em relação a outras linguagens de descrição de software existentes [Grötker et al., 2002]. A biblioteca de classes SystemC suporta a modelagem funcional de sistemas através de [Initiative, 2005]:

- Decomposição hierárquica do sistema em módulos;
- Conectividade estrutural entre módulos usando portas;
- Agendamento e sincronização de processos concorrentes usando eventos;
- Passagem de tempo simulado;
- Separação entre computação (processos) e comunicação (canais);
- Refinamento independente de computação e comunicação utilizando interfaces;
- Tipos de dados orientados ao hardware para modelagem de lógica digital e aritmética de ponto fixo.

Na Figura 2.3 pode ser vista a arquitetura de uma aplicação em SystemC. Os blocos sombreados representam a biblioteca de classes SystemC. O bloco C++ na parte de baixo da figura representa as bibliotecas padrão ou proprietárias C++, que tem relacionamento com metodologias de projeto e verificação ou canais de comunicação específicos.

As classes da biblioteca são separadas em quatro categorias:

Núcleo da linguagem

Contém o mecanismo de simulação com o escalonador de processos. Processos são executados como resposta a notificação de eventos. Eventos são notificados em pontos específicos da simulação. Em caso de eventos ordenados no tempo o escalonador é determinístico. No caso de eventos que ocorrem no mesmo momento no tempo, o escalonador é não-determinístico[1]

[1]Não se pode saber de antemão qual será o resultado da execução de um algoritmo não determinístico.

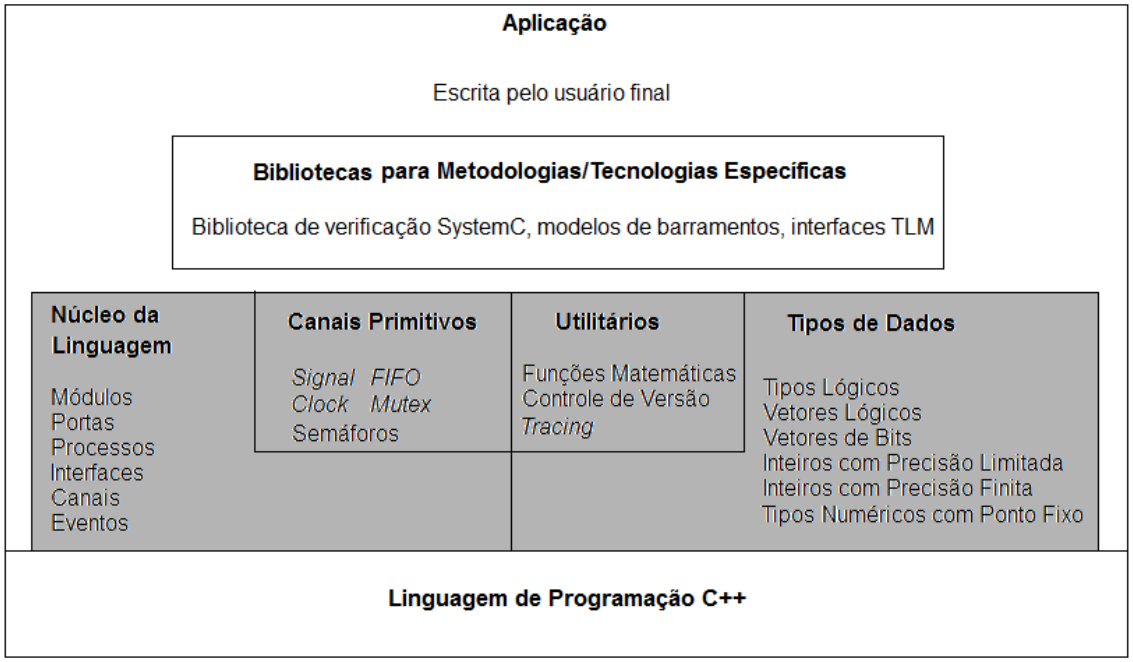


Figura 2.3: Arquitetura da linguagem SystemC [Initiative, 2005].

Tipos de dados SystemC

Todos os tipos nativos de C++ são suportados e mais alguns tipos de dados adicionais aplicáveis a aplicações de hardware digital. Entre os tipos específicos do SystemC temos inteiros com precisão limitada e finita, com e sem sinal, tipos numéricos com ponto fixo, tipos lógicos com aceitação de valores lógicos 0, 1, desconhecido e de alta impedância; vetores lógicos e de bits.

Canais Primitivos

A biblioteca SystemC define um número de canais primitivos predefinidos, são eles: *buffer, fifo, mutex, semáforo e signal.* Os canais primitivos aceitam ações de leitura e escrita e podem ser utilizados individualmente ou combinados como parte de um mecanismo de comunicação mais complexo.

Utilitários

Entre algumas funções utilitárias providas pelo SystemC podem ser citadas funções matemáticas (abs, min, max), funções para controle de versão e *copyright*, funções de *trace* para facilitar a depuração do código.

2.3.2 Transaction Level Modeling (TLM)

Com o aumento da complexidade dos SoCs e a pressão do mercado, o nível de abstração tem sido elevado para o nível de sistema de forma a aumentar a produtividade do projeto. A necessidade pelo nível de abstração mais alto gerou um interesse maior na modelagem, síntese e análise no nível da transação [Cai & Gajski, 2003].

Um padrão industrial para TLM ajudaria a aumentar a produtividade dos engenheiros de software, arquitetos, engenheiros de implementação e verificação. Entretanto, o aumento da produtividade só será alcançado se o padrão preencher alguns critérios [Rose et al., 2005]:

- O novo padrão deve ser fácil, eficiente e seguro para utilização no ambiente concorrente;
- Deve permitir reutilização de módulos entre projetos e entre diferentes níveis de abstração em um mesmo projeto;
- A modelagem de hardware e software deve ser fácil, assim como o limite entre ambos;
- Deve permitir a utilização de componentes genéricos durante o projeto.

Num modelo TLM os detalhes dos componentes de comunicação ficam separados dos detalhes dos componentes de computação. A comunicação é modelada através de canais, enquanto as requisições de transações são feitas através de funções de interface no modelo desses canais. Com a diminuição de detalhes da comunicação e computação, que poderão ser adicionados mais tarde, a simulação é mais rápida e a exploração e validação de projetos alternativos pode ser feita num nível de abstração mais alto.

2.3.2.1 Níveis de Abstração

Existem várias terminologias para definir os níveis de abstração utilizados nos modelos de sistema. O primeiro conceito importante para entender TLM é que o sistema de comunicação e computação podem ser desenvolvidos e refinados independentemente [Black et al., 2009]. Nessa terminologia podemos dividir a granularidade do tempo para comunicação e computação em 3 categorias:

Não Temporizado
Representa a funcionalidade pura do projeto sem qualquer detalhe de implementação.

Tempo Aproximado

Contém detalhes de implementação no nível de sistema, tais como a arquitetura de sistema selecionada, mapeamento das relações entre os processos da especificação do sistema e os elementos de processamento da arquitetura de sistema.

Temporizado por Ciclo

Contém detalhes de implementação dos níveis de sistema e do nível RTL, de forma que a estimativa do ciclo pode ser obtida com precisão.

Os modelos de abstração existentes com base nas possíveis combinações de cada categoria de tempo para comunicação e computação são apresentados na Figura 2.4.

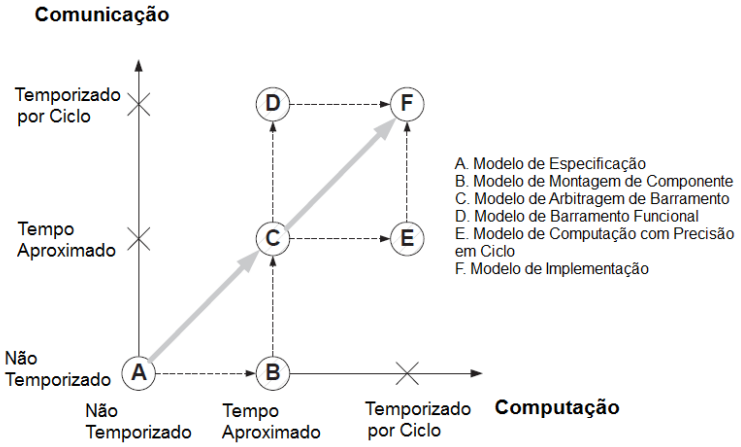


Figura 2.4: Modelos de Abstração [Cai & Gajski, 2003].

O modelo F, que tem precisão de ciclo para comunicação e computação geralmente se refere ao modelo RTL. Esse modelo é geralmente usado para síntese automática de portas lógicas.

O modelo A, sem precisão de tempo tanto na comunicação quanto na computação é conhecido como System Architectural Model (SAM). O modelo SAM é muito usado para refinamento de algoritmos e pode ter comunicação e computação refinadas para precisão aproximada de tempo.

Os outros quatro modelos - B, C, D, E - podem ser considerados TLM e tem ou comunicação ou computação com precisão aproximada de tempo. Esses modelos se baseiam na medição estimada de tempo.

A especificação executável gerada por este trabalho se enquadra no modelo A, pois não há precisão de tempo na comunicação e na computação.

2.3.3 SystemC TLM

As primeiras versões do SystemC permitiam construções RTL e possuíam conceitos primários de canais, que direcionavam para uma modelagem futura mais abstrata do que RTL. Por volta dos anos 2000, a Open SystemC Initiative (OSCI) começou a mudar a biblioteca devido aos pedidos por conceitos mais abstratos de desenvolvimento e necessidade de suporte à modelagem dirigida a especificação de eventos do sistema e não mais implementação baseadas em *clocks*. Nesse momento nasce a nova versão 2.0 com suporte à construções e modelagens no nível de sistema, como novos canais e novas entradas [Ghenassia, 2006].

SystemC 2.0 facilita o desenvolvimento de modelos TLM, que são modelos em um nível de abstração mais alto. A natureza modular da biblioteca também promove a reutilização de componentes desenvolvidos entre sistemas diferentes [Pasricha, 2002]. SystemC 2.0 define canais primitivos para comunicação de transações, mas também permite que o usuário defina canais num nível mais alto que sejam desenhados de acordo com a necessidade do projeto. Comunicação nos modelos TLM são garantidos através da utilização desses canais primitivos, enquanto a sincronização é baseada em eventos. Além disso, SystemC permite refinamentos sucessivos e a utilização de módulos em diferentes níveis de abstração em um mesmo projeto.

2.4 SysML

A linguagem de modelagem de sistemas do Object Management Group (OMG SysML™) é uma linguagem de modelagem gráfica para especificar, analisar, projetar e verificar sistemas complexos que podem incluir hardware, software, informações e procedimentos [SysML, 2008]. A linguagem provê uma representação gráfica com base semântica para modelagem de requisitos, comportamento, estrutura e parâmetros de um sistema.

Historicamente a linguagem surgiu após a chamada para propostas (RFP) [OMG, 2003] desenvolvida pelo OMG em conjunto com o INCOSE para criação de uma linguagem UML específica para sistemas de engenharia e foi publicada em Março de 2003. Essa RFP especificava os requisitos necessários para modificar a linguagem UML existente de acordo com as necessidades da comunidade de sistemas de engenharia. A especificação da linguagem SysML foi desenvolvida em resposta a esses requisitos por um grupo composto por fabricantes de ferramentas, usuários, pesquisadores e representantes do governo. O OMG anunciou a adoção do OMG SysML™em 6 de Julho de 2006 e a versão mais nova da linguagem, OMG SysML™v1.1 foi publicada em 2 de Novembro de 2008.

2.4.1 Arquitetura da Linguagem

SysML reutiliza um sub-conjunto da UML 2 e provê as extensões necessárias para satisfazer os requisitos de UML para engenharia de sistemas. A relação existente entre a linguagem de modelagem UML 2 e SysML pode ser visualizada através do diagrama Venn da Figura 2.5. Nesta figura há um círculo compreendendo a linguagem UML 2 e outro representando a linguagem SysML. Na intersecção dos círculos indicamos as construções da modelagem UML que SysML reutiliza. SysML também define novas construções para modelagem que não substituem nenhuma construção da UML 2. Há também uma parte da UML 2 que não é utilizada pelo SysML.

2.4.2 Diagramas SysML

SysML inclui nove diagramas, conforme pode ser visto na Figura 2.6. Cada diagrama pode ser descrito resumidamente da seguinte forma [Friedenthal et al., 2008]:

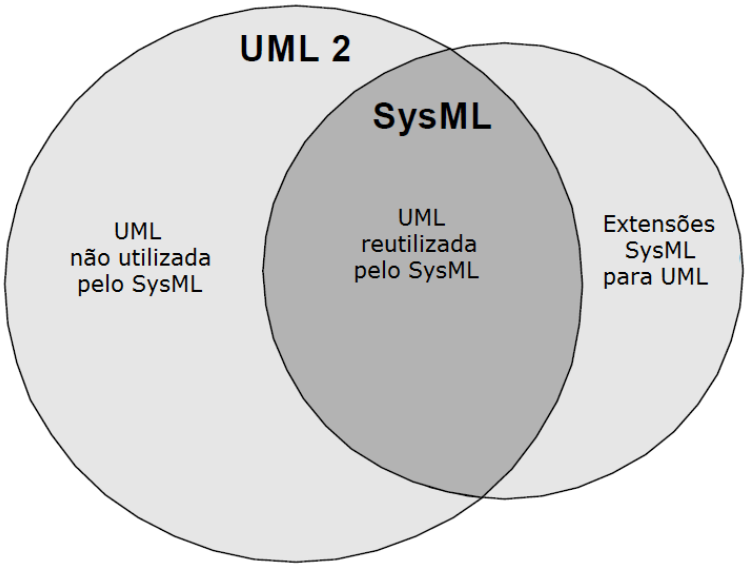


Figura 2.5: Relação entre UML 2 e SysML [SysML, 2006].

- **Diagrama de Requisitos:** Representa requisitos e seus relacionamentos com outros requisitos, elementos do projeto e casos de teste para permitir a rastreabilidade dos requisitos. Esse diagrama não existe na linguagem UML.

- **Diagrama de Atividade:** Representa o comportamento em termos da ordenação das ações baseada na disponibilidade das entradas, saídas, controles e como as ações transformam as entradas em saídas. Esse diagrama é uma modificação do diagrama de atividade existente na UML.

- **Diagrama de Sequência:** Representa o comportamento em termos da sequência de mensagens trocadas entre as partes. É o mesmo diagrama de sequência já existente na UML.

- **Diagrama de Transição de Estados:** Representa o comportamento de uma entidade em termos de suas transições entre estados disparados por eventos. Mesmo diagrama de transição de estados da UML.

- **Diagrama de Caso de Uso:** Representa a funcionalidade em termos de como um sistema ou uma entidade é usado por um ator externo para alcançar um conjunto de objetivos. Mesmo diagrama de caso de uso da UML.

- **Diagrama de Definição de Bloco:** Representa os elementos estruturais chamados blocos, suas composições e classificação. Esse diagrama é uma modificação do diagrama de classes da UML.

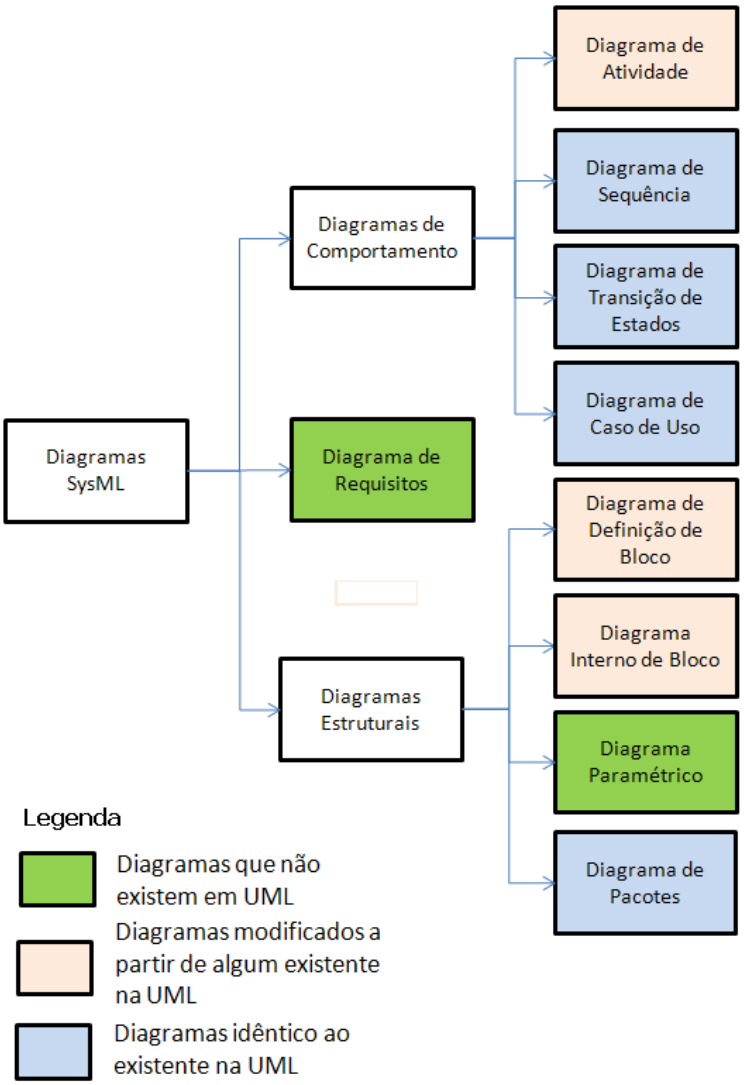


Figura 2.6: Diagramas SysML e relação deles com UML 2 [SysML, 2006].

- **Diagrama Interno de Bloco:** Representa a interconexão e interfaces entre as partes de um bloco. Esse diagrama é uma modificação do diagrama de estruturas compostas.
- **Diagrama Paramétrico:** Representa restrições nos valores de propriedades. Esse diagrama não existe na linguagem UML.
- **Diagrama de Pacotes:** Representa a organização do modelo em termos de pacotes que contém os elementos do modelo. Esse diagrama é o mesmo diagrama de pacotes existente na linguagem UML.

As subseções seguintes descreverão com mais detalhes os principais conceitos e diagramas descritos pelo SysML.

2.4.2.1 Elemento Bloco

O bloco é o elemento básico de descrição estrutural de um sistema no SysML e pode ser utilizado para representar hardware, software, instalações ou qualquer outro elemento do sistema, componente ou item que transitará através do sistema. Ele pode ser usado também para descrever entidades conceituais e abstrações lógicas.

- **Propriedades** são as características estruturais primárias de um bloco. Propriedades relativas a parte descrevem a hierarquia de um bloco provendo assim um mecanismo para definir a parte no contexto de um todo. As propriedades relativas a valor descrevem os intervalos válidos para os valores de uma propriedade, dimensões e unidades de medida. Na Figura 2.7 pode ser visto um exemplo de notação utilizada para descrição de blocos com seus valores, operações e restrições.

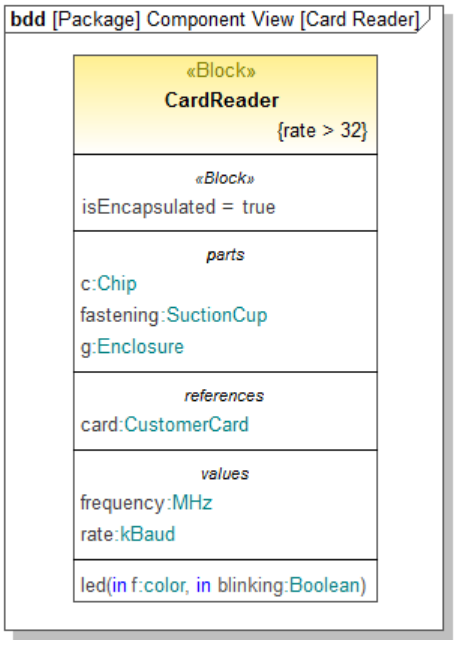


Figura 2.7: Notação utilizada para Blocos.

- **Portas** são características estruturais que descrevem os pontos de interação de um bloco com outros blocos e são usadas para conectar as partes de um bloco. As portas em SysML pode ser de dois tipos:

  Portas de fluxo Esse tipo é usado para especificar as entradas e saídas do bloco. Através dessas portas teremos o fluxo de informação entre blocos. Na Figura 2.8 há um exemplo da notação de portas de fluxo.

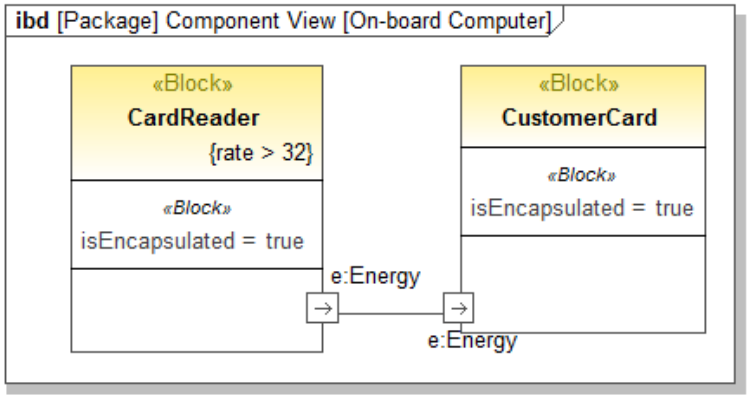


Figura 2.8: Notação utilizada para Blocos.

Portas padrão Usadas para especificar os tipos de serviço que um bloco provê ou utiliza. Responsável por manipular as requisições e invocar serviços disponibilizados por outros blocos.

2.4.2.2 Diagrama de Definição de Bloco

Esse diagrama é um dos diagramas estruturais da linguagem. Ele define características de blocos e o relacionamento entre eles, tais como associações, generalizações e dependências. O diagrama captura a definição de blocos em termos das propriedades, operações e relacionamentos como uma hierarquia do sistema ou uma árvore de classificações, como pode ser visto nas Figuras 2.7 e 2.9.
O diagrama de definição de bloco é baseado no diagrama de classe da UML, com restrições e extensões definidas pelo SysML.

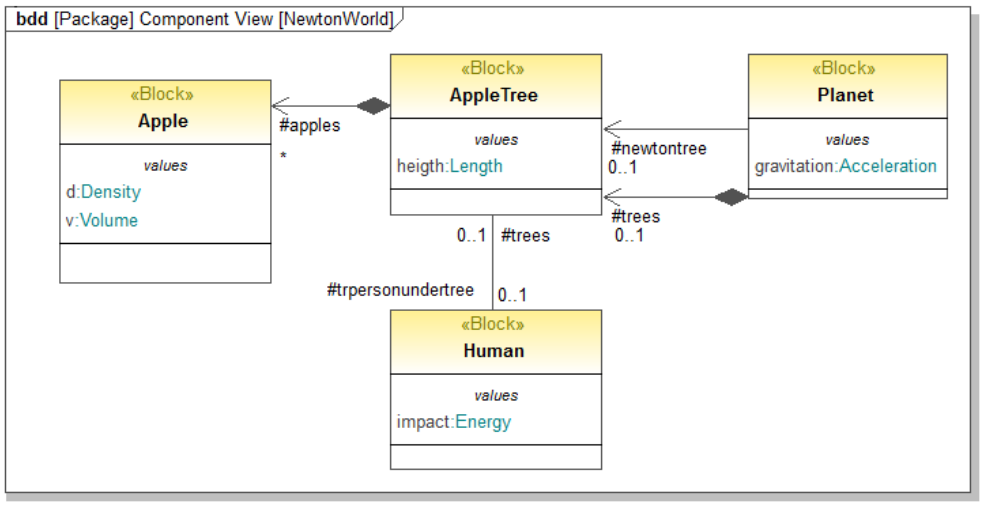


Figura 2.9: Mundo de Newton [Friedenthal et al., 2008].

2.4.2.3 Diagrama Interno de Bloco

O diagrama interno de bloco (DIB) é usado para descrever a estrutura interna de um bloco, seus atributos e propriedades e as conexões entre partes de um bloco. Esse diagrama permite ao projetista detalhar aspectos estruturais do modelo. O diagrama interno de bloco é equivalente ao diagrama de estruturas compostas existentes na UML. Na Figura 2.10 pode ser observado um Diagrama Interno de Bloco.

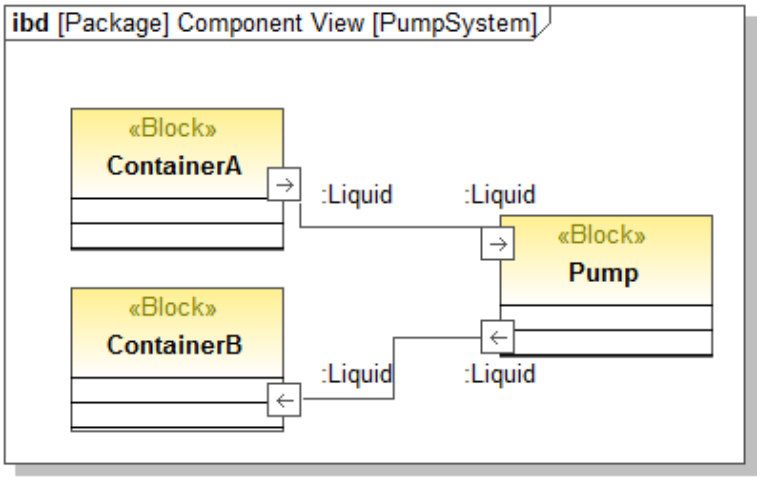


Figura 2.10: Notação utilizada para Diagrama Interno de Bloco.

O diagrama interno de bloco mostra aspectos mais detalhados da estrutura do que o diagrama de definição de bloco. Por exemplo, com ele pode ser detalhado como as propriedades de um bloco - portas e atributos - se relacionam. O diagrama de definição de bloco faz descrições no nível dos tipos, enquanto o diagrama interno de bloco descreve no nível funcional.

2.4.2.4 Diagrama Paramétrico

SysML permite a definição de relacionamentos paramétricos entre propriedades de blocos. O diagrama paramétrico representa restrições em valores de propriedade do sistema, tais como desempenho e confiabilidade, e serve como meio para integrar os modelos das fases de especificação e projeto com modelos da fase de análise.

Uma definição que deve ser conhecida nesse ponto é a do **Bloco de Restrições**. Descrito através de um diagrama de definição de bloco, ele descreve restrições na estrutura do sistema e parâmetros necessários. O bloco de restrição é definido livre de contexto, permitindo a criação de uma biblioteca de restrições que pode ser facilmente reutilizada em vários projetos.

Após a criação do bloco de restrições, ele pode ser atribuído a um modelo através do diagrama paramétrico. Para facilitar o entendimento desse conceito foi utilizado um exemplo descrito por Friedenthal [Friedenthal et al., 2008]. A Figura 2.9 descreve o sistema simplificado do mundo de Newton, numa tentativa de modelar a famosa queda da maçã na cabeça de Newton.

No bloco de restrições da Figura 2.11 estão os parâmetros e restrições das leis da física de Newton que devem ser aplicadas no episódio da queda da maça.

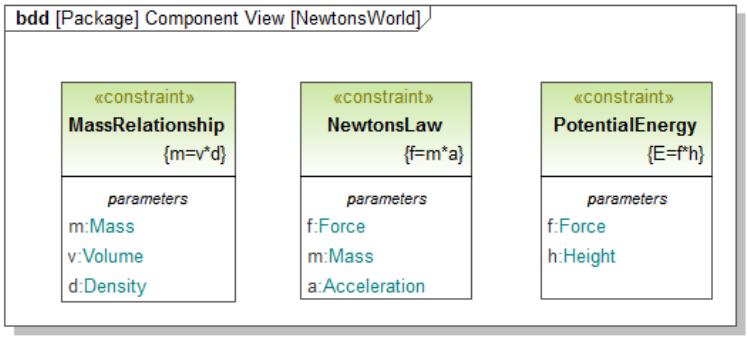


Figura 2.11: Biblioteca de restrições do mundo de Newton [Friedenthal et al., 2008].

A seguir, pode ser observada a aplicação das restrições ao modelo no diagrama paramétrico da Figura 2.12.

2.4.2.5 Alocação

Um aspecto interessante do SysML é a possibilidade de alocação de comportamentos a partes. Alocações podem ser descritas em diagramas de atividades, conforme Figura 2.13, onde cada atividade está alocada para uma parte de um bloco chamado *SoC1*.

O diagrama da Figura 2.13 descreve o comportamento do *SoC1*, que irá executar a função **getValue** e, em seguida, a função **increaseValue**, buscando um determinado valor e incrementado-o. A seguir o *SoC1* verifica se $value1 > value2$, conforme restrição a ser satisfeita. Se a restrição for satisfeita continua lendo e incrementando valores, caso contrário, para.

2.4.3 Ferramentas de Modelagem

Existem várias ferramentas de modelagem disponíveis no mercado que permitem a modelagem de diagramas SysML. Outras ainda estão em processo de modificação para atenderem a especificação SysML do OMG. Abaixo há uma lista de ferramentas

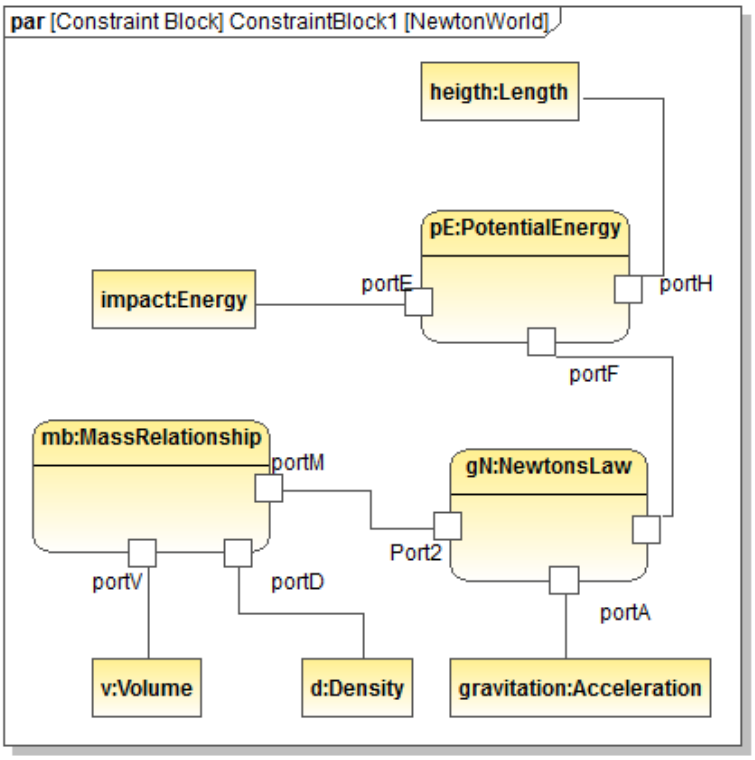


Figura 2.12: Utilização do Bloco de Restrição num Diagrama Paramétrico [Friedenthal et al., 2008].

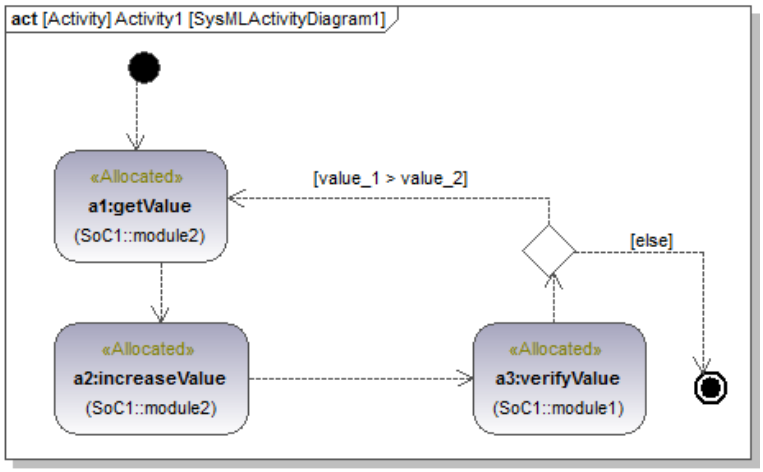


Figura 2.13: Alocação de atividades para partes.

com suporte a SysML. A listagem foi encontrada no fórum SysML [Forum, 2010] e no site SysML do OMG [Language, 2010].

Ferramentas pagas ou extensões para ferramentas pagas:

- Altova UModel;
- Artisan Studio (Artisan Software);

- CoFluent Design;
- SysML Toolkit (EmbeddedPlus) + Rational Software Modeler/Architect (IBM);
- Rhapsody (IBM/Telelogic);
- TAU G2 (IBM/Telelogic);
- InterCAX;
- MagicDraw + SysML plugin (No Magic);
- Software Stencils - Microsoft Visio SysML and UML templates;
- Enterprise Architect + MDG Technology for SysML (Sparx Systems).

Ferramentas com código fonte aberto:

- Papyrus for SysML (Open Source Eclipse Modeling Tool);
- TOPCASED-SysML (TOPCASED Modeling Framework Open Source Project).

2.4.4 Benefícios da utilização do SysML

- SysML melhora a comunicação, pois provê uma linguagem formal para compartilhar as informações do sistema entre todos os envolvidos no projeto. Por ser baseado na UML, SysML garante que o fluxo de informações entre os engenheiros de sistemas e os engenheiros de software seja mais preciso e formal;
- O suporte à modelagem de requisitos provê a habilidade de avaliar o impacto na mudança de requisitos na arquitetura do sistema;
- SysML é uma linguagem precisa e inclui suporte a restrições e análise paramétrica que permite a análise e simulação de modelos;
- É um padrão aberto que suporta XMI e ISO 10303-303 (AP233), permitindo a troca de informações com outras ferramentas de engenharia de sistemas.

Capítulo 3

Metodologia de Mapeamento de SysML para SystemC

Este trabalho apresenta a ferramenta **SysMLToSystemC** como um *plugin* para a ferramenta Altova UModel, que converte a descrição SysML para especificação executável SystemC.

Algumas tentativas foram feitas de mapeamento entre diagramas SysML para código em SystemC [Raslan & Sameh, 2007a] [Prevostini & Zamsa, 2007], mas as tentativas foram restritas a poucos diagramas da linguagem e a geração de código era somente de blocos e não de um projeto completo e que pudesse ser executado. A proposta deste trabalho é fazer um mapeamento mais abrangente do que os propostos anteriormente na literatura, permitindo o desenvolvimento de sistemas mais complexos através da ferramenta e com geração de código executável em SystemC. O procedimento de transformação utilizado para tradução dos diagramas SysML para código fonte SystemC está mostrado na Figura 3.1.

O projetista fará a modelagem do SoC utilizando a ferramenta Altova UModel [UModel, 2011]. Ele deve exportar o projeto para XMI através da própria ferramenta e executar o plugin desenvolvido para a geração da especificação executável em SystemC.

Na próxima seção serão mostradas algumas informações sobre a ferramenta Altova UModel e sobre a ferramenta utilizada para desenvolvimento do *plugin*. Na Seção 3.2 será explicado detalhadamente cada passo do procedimento de mapeamento SysML para SystemC.

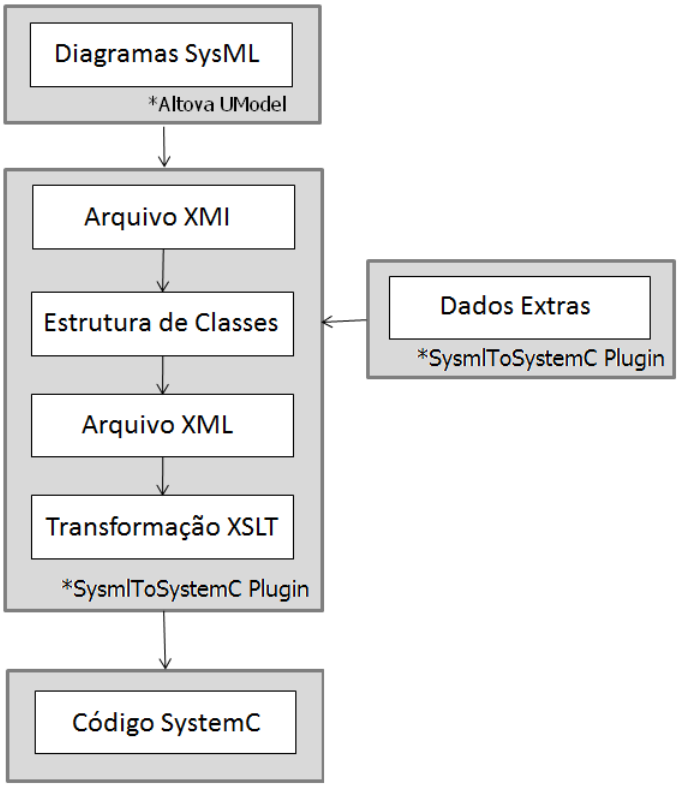


Figura 3.1: Procedimento de Mapeamento SysML para SystemC.

3.1 Altova UModel

Altova UModel [UModel, 2011] é uma ferramenta UML de modelagem de software e desenvolvimento de aplicações gráficas. A ferramenta permite a modelagem de todos os tipos de diagramas existentes na UML 2.3 e em SysML. Entre suas características três foram decisivas para a escolha da ferramenta:

- **SysML para modelagem de sistemas embutidos:** A ferramenta permite a modelagem gráfica de todos os diagramas do padrão OMG SysML e garante alguma coerência de blocos utilizados em diferentes diagramas. A possibilidade de modelagem gráfica SysML era o requisito principal para a escolha da ferramenta a ser utilizada neste trabalho.

- **Exportação de modelos SysML/UML para formato XMI:** A ferramenta permite a exportação de modelos SysML para o formato padronizado XMI em sua mais recente versão - XMI 2.1. A exportação é feita de forma simples e o arquivo XMI gerado é a entrada utilizada pela nossa ferramenta de mapeamento.

- **Application Programming Interface (API):** Essa API permite a criação de *plugins* para a ferramenta, possibilitando a adição de novos menus. Através dessa funcionalidade foi criada a opção de menu permitindo o acesso direto à ferramenta **SysMLToSystemC**.

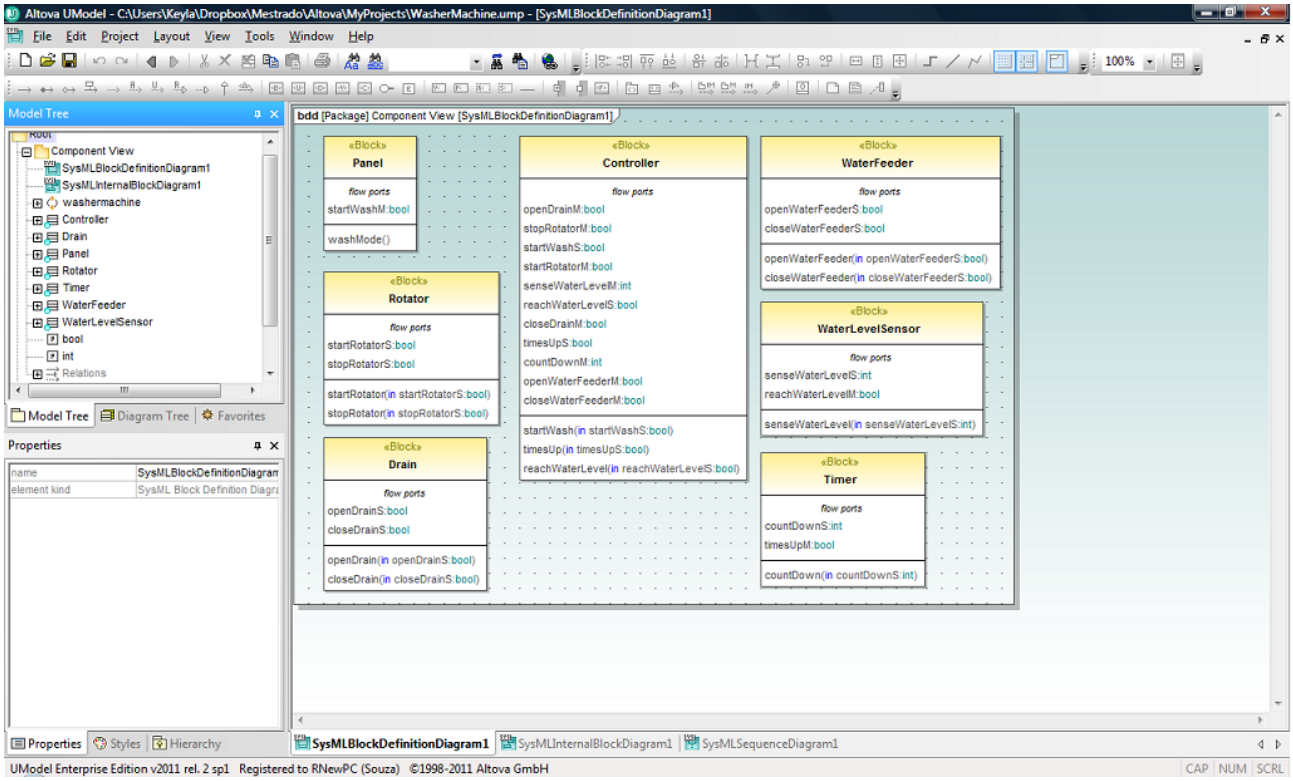


Figura 3.2: Ferramenta Altova UModel.

3.2 Metodologia Detalhada

O objetivo é permitir a modelagem estrutural e comportamental para geração final de uma especificação executável SystemC. A metodologia de transformação utilizada para gerar código fonte SystemC a partir de diagramas SysML está descrita nas próximas subseções. Cada etapa está descrita na ordem em que ocorre, com ênfase para as etapas três e cinco, que são as de maior importância para o processo de tradução.

3.2.1 1ª Etapa: Modelagem em Diagramas SysML

O projetista deve modelar o SoC em diagramas SysML utilizando a ferramenta Altova UModel. Na Figura 3.2 pode ser vista a tela da ferramenta com seus menus e propriedades.

Foi decidido que somente dois diagramas estruturais SysML - Diagrama de Definição de Bloco e Diagrama Interno de Bloco - e o Diagrama de Transição de Estados serão mapeados para SystemC. O Diagrama de Sequência será utilizado internamente somente para conferência das operações existentes em cada bloco e terá suas informações exportadas no arquivo XML de saída para futuros trabalhos. Os diagramas podem ser modelados com todas as características que o projetista achar necessárias, mas devido ao alto nível de abstração da especificação executável gerada, somente algumas informações serão utilizadas na geração do código fonte em SystemC. Entre as informações utilizadas na geração do código fonte, podem ser citadas:

- Bloco: Nome, portas, conectores e operações;
- Porta: Nome, tipos e direção;
- Conector: Tipo, blocos e portas conectados por ele;
- Operação: Nome, nomes e tipos dos parâmetros de entrada e saída.

Os tipos de dados utilizados na configuração de portas, conectores e parâmetros de operações serão convertidos na sua forma de texto. Podem ser utilizados tipos já fornecidos pela ferramenta UModel ou o projetista pode adicionar tipos primitivos SystemC ao projeto.

3.2.2 2ª Etapa: Exportação de Diagramas SysML como arquivo XMI

Esta etapa também é realizada utilizando a ferramenta Altova UModel. O projetista deve exportar o projeto com diagramas SysML como um arquivo no formato XML-based Metadata Interchange (XMI) através no menu da ferramenta, conforme a Figura 3.3.

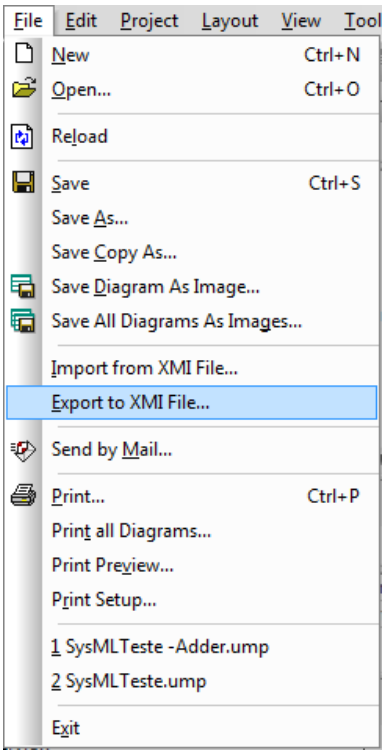


Figura 3.3: Menu para exportação XMI.

XMI [OMG, 2011] é um formato de troca de metadados definida nos termos do padrão Meta-Object Facility (MOF)[1] [OMG, 2010]. XMI suporta a troca de modelos completos entre ferramentas de modelagem e é um repositório textual da descrição completa do SoC.

[1]MOF é um padrão OMG originado na UML que é utilizado na criação de metamodelos. Sua arquitetura é baseada em quatro camadas: Camada de informação, Camada de modelos, camada de metamodelos e camada de meta-metamodelos.

3.2.3 3ª Etapa: Importação de arquivo XMI para ferramenta SysMLToSystemC

Esta etapa e as seguintes são realizadas pelo plugin de tradução **SysMLToSystemC**. Um plugin para geração de código fonte SystemC foi desenvolvido e está integrado a ferramenta Altova UModel no menu *Project/Generate SystemC Code* conforme pode ser visto na Figura 3.4. Detalhes de como foi desenvolvido o plugin e como ele foi integrado na ferramenta podem ser vistos no Apêndice A.

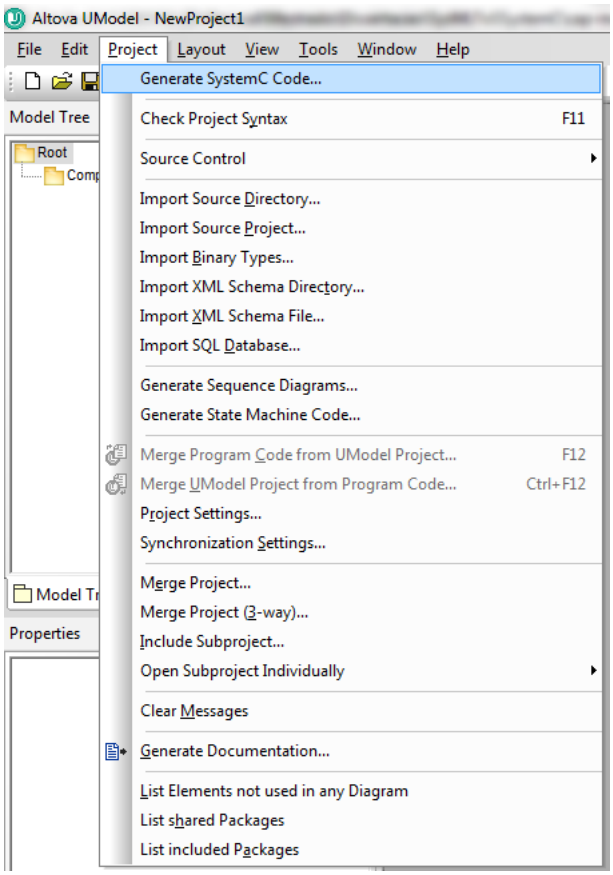


Figura 3.4: Menu do plugin **SysMLToSystemC**.

Ao acessar o menu do plugin, conforme a Figura 3.4, a ferramenta de geração de código fonte SystemC é aberta. A tela da ferramenta **SysMLToSystemC** pode ser vista na Figura 3.5.

A representação XMI do modelo SysML tem muitos detalhes relativos a elementos de representação gráfica, referências múltiplas entre os diferentes diagramas e outras informações específicas da ferramenta de modelagem. Como este arquivo é gerado de acordo com o metamodelo de estrutura MOF, as informações associadas com os elementos não são de fácil acesso. Devido a isso, este formato não seria o ideal para

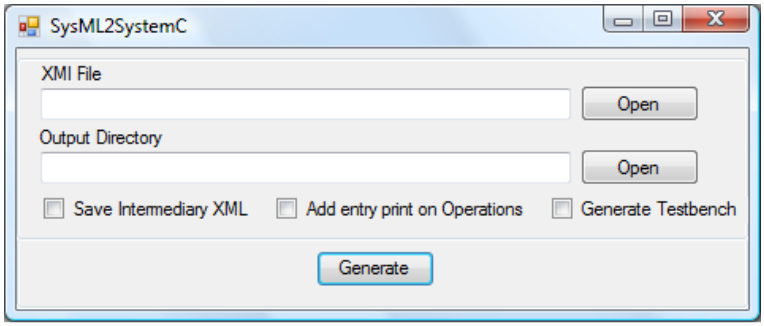


Figura 3.5: Plugin **SysMLToSystemC**.

começar a tradução do código fonte. Então foi tomada a decisão de processar o arquivo XMI para extrair somente as informações relevantes e os detalhes necessários no processamento das etapas seguintes.

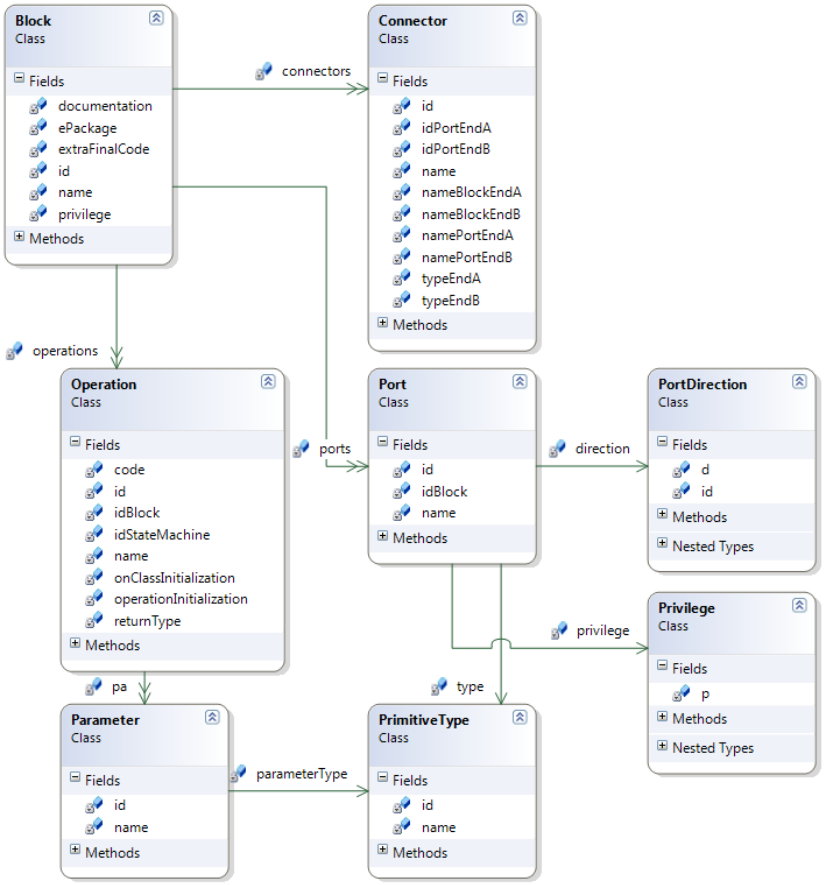


Figura 3.6: Diagrama de Classes representando os Blocos e suas informações.

Com todos os dados coletados, um modelo de classes C♯ é gerado capaz de descrever o SoC modelado em SysML. Esse grupo de classes é responsável por descrever os blocos, suas portas e operações, e também os diagramas de Transição de Estado.

As Figuras 3.6 e 3.7 demonstram os diagramas de classes detalhando como as classes em C♯ foram criadas para descrever as informações.

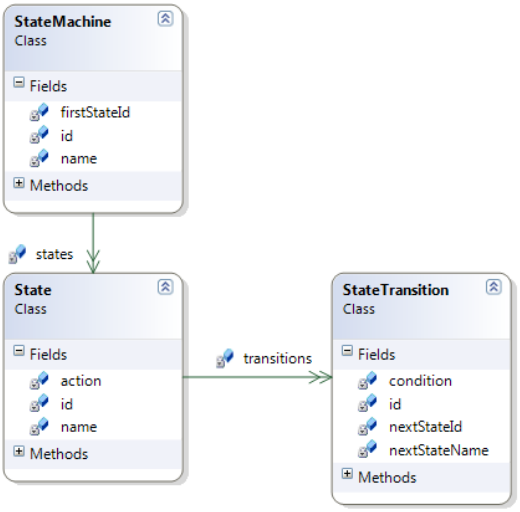


Figura 3.7: Diagrama de Classes representando os Diagramas de Transição de Estados.

Os Diagramas de Sequência também são processados e tem suas informações armazenadas em classes C♯, conforme pode ser visto na Figura 3.8, para processamento posterior.

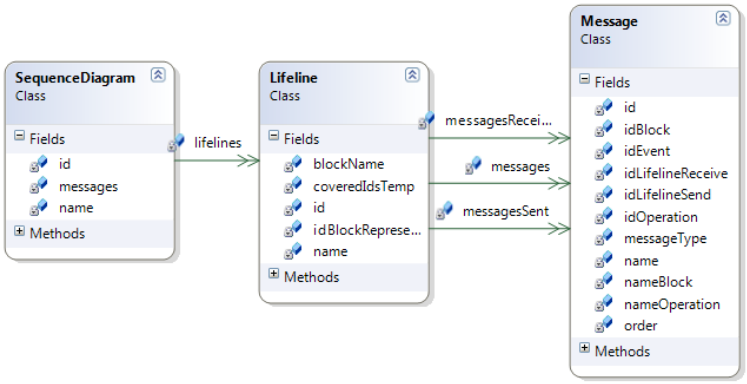


Figura 3.8: Diagrama de Classes representando os Diagramas de Sequência.

3.2.4 4ª Etapa: Escolha de opções relativas ao SystemC a ser gerado

Durante esta etapa é possível escolher opções na tela que serão utilizadas durante a geração do código fonte SystemC. A tela da ferramenta oferece opções diferentes de acordo com os diagramas presentes no sistema que está sendo importado. A Figura 4.9 mostra uma tela com todas as opções possíveis para uma geração. As opções existentes são:

- **Output Directory:** Nesta caixa de texto deve ser escolhido o diretório de saída aonde os arquivos devem ser gerados. Caso o campo esteja vazio, todos os arquivo serão salvos no diretório aonde os arquivos do *plugin* **SysMLToSystemC** estão salvos.

- **Save Intermediary XML:** Ao marcar esta opção a ferramenta irá salvar o arquivo XML intermediário utilizado internamento pela ferramenta em um arquivo de nome *BlocksModel.xml* no diretório de saída escolhido. Este arquivo contém a descrição completa do sistema e pode ser usado por outras ferramentas sem a necessidade que estas façam qualquer integração direta com o Altova UModel.

- **Add entry print on Operations:** Esta opção adiciona uma impressão com o nome do módulo e o nome da operação no começo de todas as operações do sistema. Mesmo que o usuário adicione código próprio na operação - a ser detalhado adiante - na geração final dos arquivos essa impressão irá anteceder esses códigos inseridos sem comprometer o funcionamento das operações. Esta opção auxilia na verificação do comportamento do sistema e garantia da execução das operações na ordem esperada.

- **Generate Testbench:** Esta opção gera um *Testbench* para validação do sistema. A estrutura do *testbench* consiste em um módulo **driver**, que gera entradas para o sistema a ser testado (DUV –Device Under Validation), e um módulo **monitor**, que receberá as saídas geradas pelo DUV, além do próprio DUV. A Figura 3.9 descreve o modelo do *testbench* gerado.

  O módulo **driver** é criado com as portas de saída equivalentes a todas as portas de entrada do DUT que não estiverem conectadas a nenhuma outra porta. As portas de saída criadas são conectadas a essas portas de entrada. Um processo com o nome **generate ()** também é criado para que posteriormente o projetista insira código fonte para geração de dados.

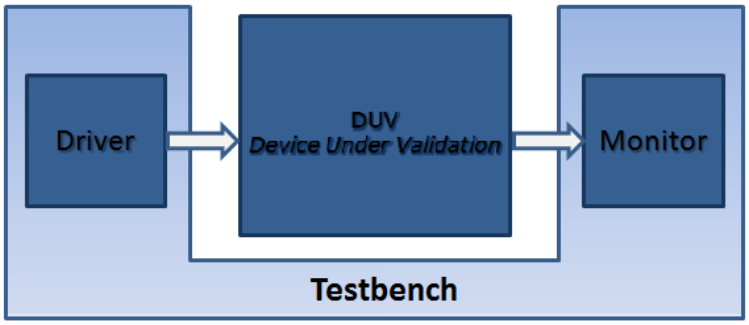


Figura 3.9: Modelo de Testbench.

O módulo **monitor** é criado com as portas de entrada equivalentes a todas as portas de saída do DUT que não estiverem conectadas a nenhuma outra porta. As portas de entrada criadas neste módulo são conectadas as portas de saída do DUT. Um processo com o nome **show ()** também é criado para que posteriormente o projetista insira código fonte para impressão de dados. Esse processo é gerado com sensibilidade a todas as portas de entrada do módulo **monitor**.

Os dois módulos são adicionados ao sistema, salvos no XML intermediário e ligados ao sistema completo no arquivo *main.cpp*. O uso de testbenches reduz a quantidade de erros em projetos e facilitará a verificação do comportamento do sistema caso o DUT passe pelo processo de refinamento sucessivo no futuro, através da comparação das saídas obtidas em cada nível de refinamento.

Caso o projetista não tenha deixado nenhuma porta desconectada, ou porque criou o *testbench* manualmente na modelagem SysML ou por falha na modelagem, os módulos extras não serão criados mesmo com a opção de geração marcada.

- **Use Sequence Diagram:** Esta opção só aparecerá para o usuário da ferramenta no caso da existência de pelo menos um Diagrama de Sequência no sistema. A primeira opção do menu é a não utilização das informações de nenhum diagrama para a geração da especificação executável. As opções seguintes mostram os nomes dos diagramas de sequência do sistema. Somente um diagrama pode ser escolhido para ser utilizado na geração dos arquivos. O diagrama de sequência escolhido irá impactar somente na conferência das operações de cada módulo. Independente da opção escolhida neste ponto, as informações de todos os diagramas de sequência serão inseridas no XML intermediário.

- **Operation x Source Code (State Machine, File):** Nesta tabela são listadas todas as operações existentes no sistema, com o nome do módulo e o nome da operação propriamente dito. Para cada operação o usuário terá as opções de:

Não adicionar nenhum código extra na operação (None), Adicionar código fonte manualmente ou a partir de um arquivo (Entry Source Code) e utilizar o comportamento de uma das máquinas de transição de estados - caso não tenha nenhuma no sistema, essa opção não estará disponível no menu. Ao escolher iniciar a geração do código fonte, a ferramenta abrirá uma tela para cada operação onde a inclusão do código fonte foi selecionada. Nas Figuras 3.10 e 3.11 foi escolhido um código já pronto de um arquivo texto para incluir na operação **generatedata ()** de um bloco de nome Driver. Na tela são listadas informações básicas que podem ser utilizadas na escrita do código: nome do bloco ao qual a operação faz parte, nome da operação, as portas de entrada e saída disponíveis neste bloco.

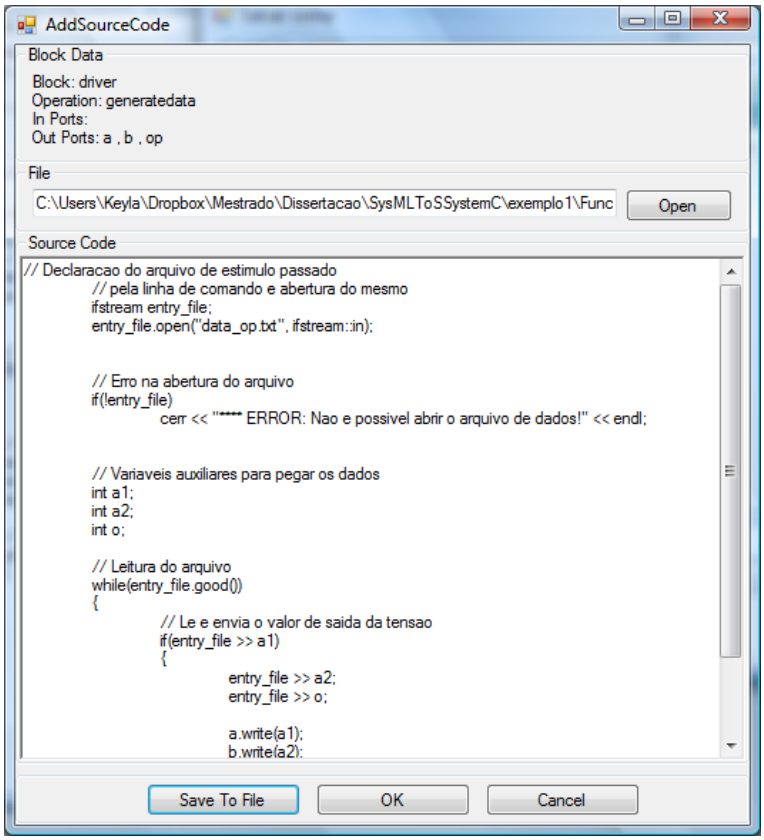


Figura 3.10: Adicionando código fonte para operação a partir de arquivo.

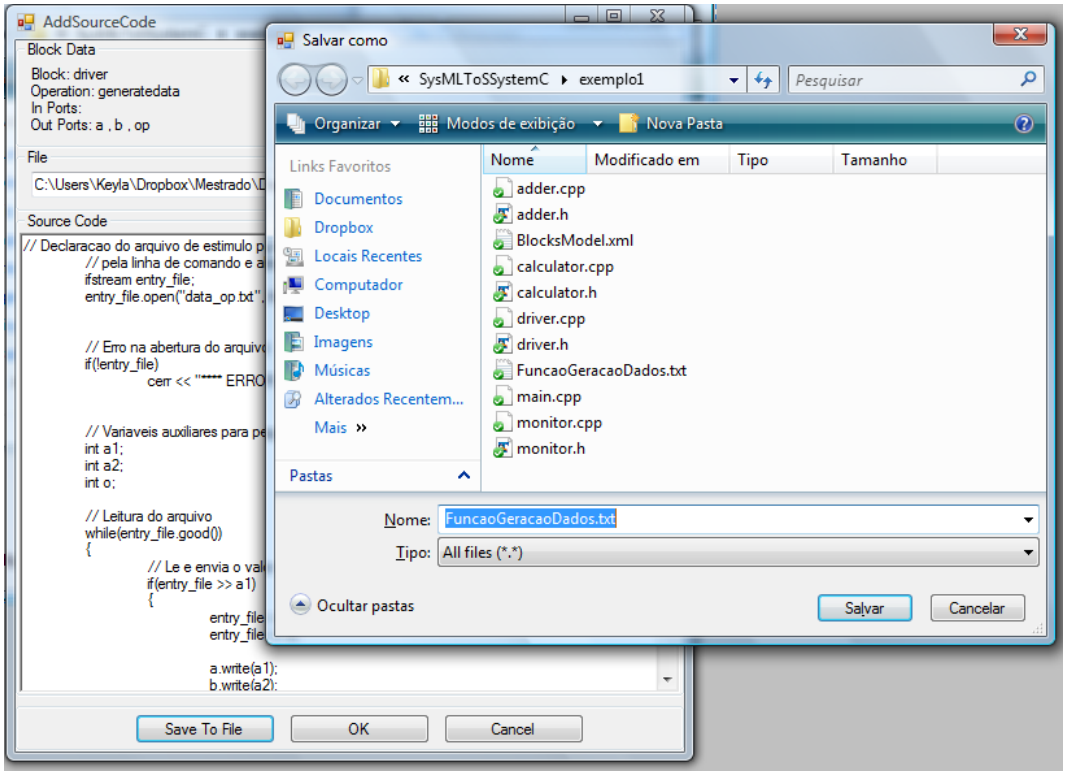


Figura 3.11: Salvando arquivo texto com operação de leitura de dados.

3.2.5 5ª Etapa: Geração de Especificação Executável em SystemC

Utilizando as informações agora descritas como um grupo de classes C♯ e com base nas opções escolhidas na etapa anterior, relativas ao relacionamento entre Diagramas de Transição de Estados e operações de blocos, e a escolha da utilização de um diagrama de sequência, um arquivo XML é gerado como um passo intermediário para geração da especificação executável em SystemC. Desta forma a leitura das informações pelos algoritmos seguintes e a troca de informações com outras ferramentas pode ser realizada de forma mais simples. Para gerar o arquivo XML a ferramenta recupera as instâncias de todas as classes, as associações entre todas elas e gera um documento XML de saída com todas as informações. Na Figura 3.12 pode ser vista a estrutura do arquivo XML intermediário gerado.

O arquivo XML intermediário pode ser facilmente transformado em código fonte SystemC utilizando arquivos *template*. A tecnologia escolhida para realizar esse passo foi a W3C XSLT (Extensible Stylesheet Language Transformations) [Kay, 2002], que transforma arquivos XML baseado em folhas de estilo. Os arquivos XSLTs utilizados na transformação dos arquivos estão no Apêndice B.

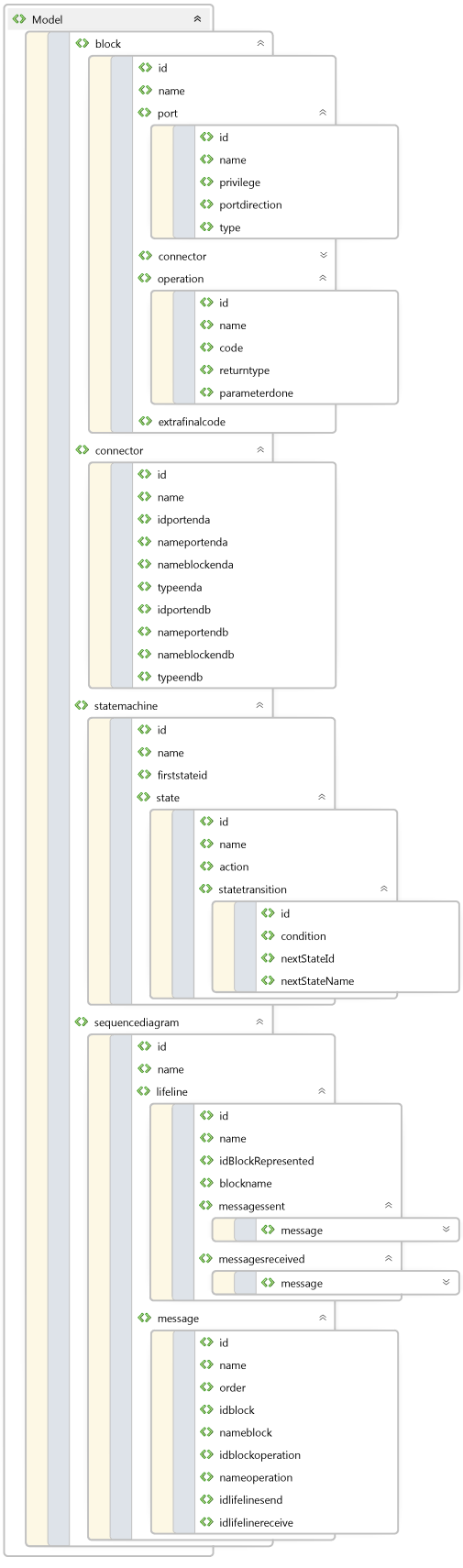


Figura 3.12: Estrutura do Arquivo XML Intermediário.

A ferramenta irá transformar o arquivo XML intermediário em uma especificação executável SystemC utilizando diferentes arquivos XSLT para cada tipo de arquivo a ser gerado (.h, .cpp). Na Seção 3.2.5.1 estão detalhadas as regras utilizadas para o mapeamento dos diagramas SysML para o código fonte SystemC. Essas regras são utilizadas em parte na geração do XML e parte na transformação do XML para a especificação executável em SystemC.

3.2.5.1 Mapeamento do SysML para SystemC

O mapeamento entre a modelagem do sistema em SysML e a especificação executável em SystemC foi feito através de várias regras. Nas seções seguintes serão detalhadas as regras para o mapeamento de cada parte do sistema.

Blocos → Módulos

Cada bloco modelado em SysML dará origem a um módulo em SystemC. Os módulos são gerados em dois arquivos, como nas Listagens 3.1 e 3.2:

- Arquivo .h que terá a criação do módulo, declaração das portas e declaração dos processos.
- Arquivo .cpp tem o código de cada processo do módulo.

Para facilitar a explicação das linhas principais de ambos os arquivos, os comentários foram separados nas duas próximas seções.

O arquivo .h O modelo de formação dos arquivos .h de cada módulo está na Listagem 3.1. As linhas principais do arquivo são:

- **Linha 4:** O arquivo é iniciado com a inclusão do arquivo de cabeçalho da biblioteca SystemC.
- **Linha 5:** Declaração do módulo com a macro *SC_MODULE*.
- **Linhas 8 e 9:** Declaração das portas do módulo. As portas de entrada e saída são declaradas *sc_fifo_in<tipo >* e *sc_fifo_out<tipo >* respectivamente. O tipo utilizado será o tipo escolhido para a porta de fluxo no SysML. Será utilizado o texto descritivo deste tipo no arquivo.
- **Linhas 12 e 13:** Declaração de todos os processos deste módulo, sempre no padrão ***void*** *NomeDoProcesso ()*. Detalhes da criação e da inicialização dos

processos - feita mais adiante - estarão na seção seguinte sobre mapeamento de Operações → Processos.

- **Linha 15:** Construtor do módulo com a macro *SC_CTOR*.
- **Linhas 18-21:** A inicialização dos processos é feita com a macro *SC_THREAD*. O detalhamento da inicialização é feito na próxima seção. Todos os processos do módulo são inicializados nesta parte do arquivo.

```
1  //This file is automatically generated
2  //by SysML2SystemC Project
3
4  #include "systemc.h"
5  SC_MODULE(modulo)
6  {
7  //Declaracao das portas de entrada e saida
8  sc_fifo_in< int > a;
9  sc_fifo_out< int > b;
10
11 //Declaracao dos processos
12 void process1( );
13 void process2( );
14
15 SC_CTOR(modulo)
16 {
17
18 //Inicializacao dos processos
19 SC_THREAD(process1);
20 SC_THREAD(process2);
21 sensitive << a.data_written();
22
23 //Inicializacao dos estados inicial atual
24 //da Maquina de transicao de estados, caso
25 //tenha sido escolhida para algum processo
26 InitialState = FirstState;
27 CurrentState = InitialState ;
28 }
29
30 //Enumeracao dos estados da Maquina de
31 //transicao de estados, caso
32 //tenha sido escolhida para algum processo
33 private:
34 enum typeState={FirstState,S_OP,S_Add,
35 S_Sub,S_Mul,S_Div,FinalState,S_Chk,S_ILG};
36 typeState CurrentState,InitialState,
37 NextState;
38 };
```

Listagem 3.1: modulo.h.

- **Linhas 23-27:** Se o usuário decidir utilizar um diagrama de transição de estados para representar o comportamento de algum processo deste módulo, neste ponto

serão inicializados os estados **inicial** e **atual** que controlará a movimentação dentro da máquina de estados.

- **Linhas 30-37:** Nestas linhas são criados os estados da máquina de estados. Também só serão geradas caso o usuário decida utilizar um diagrama de transição de estados para algum processo do módulo atual.

```
1  //This file is automatically generated
2  //by SysML2SystemC Project
3
4  #include "systemc.h"
5  #include "modulo.h"
6  void modulo::process1( )
7  {
8  //Imprimindo entrada do modulo,
9  //caso a opcao tenha sido selecionada
10 cout << "Entering modulo::process1" << endl;
11
12 //Restante do código que o usuário
13 //decida incluir.
14 }
15
16 void modulo::process2( )
17 {
18         while(true)
19         {
20                 wait();
21                 //Imprimindo entrada do modulo,
22                 //caso a opcao tenha sido selecionada
23                 cout << "Entering modulo::process2" << endl;
24
25                 //Restante do código que o usuario
26                 //decida incluir.
27         }
28 }
```

Listagem 3.2: modulo.cpp.

O arquivo .cpp O modelo de formação dos arquivos .cpp de cada módulo está na Listagem 3.2. Neste arquivo ficam os códigos fontes de todos os processos pertencentes aos módulos. As linhas principais do arquivo são:

- **Linha 5:** Nesta linha há a referência ao arquivo de cabeçalho .h do módulo.

- **Linhas 6 e 16:** Nestas linhas estão os cabeçalhos dos processos. O cabeçalho sempre tem o nome do módulo e o nome da função, sem parâmetros, conforme será explicado adiante.

- **Linhas 10 e 23:** Caso o usuário tenha escolhido a opção de imprimir a entrada de cada processo, a primeira linha será sempre o comando para impressão informando entrada no módulo e processo.
- **Linhas 18,19 e 20:** Como este processo *process2* é um processo do tipo *SC_THREAD*, o escalonador SystemC iria executá-lo uma única vez. Para que ele execute mais vezes, foi introduzido um comando **while(true)** para que este processo seja executado infinitamente. O comando **wait()** fará com que o processo espere os eventos de escrita nas portas da lista de sensibilidade para que continue a execução.

As linhas seguintes terão o código inserido pelo próprio usuário ou ficarão em branco, caso o usuário decida deixar assim.

Operações → Processos

As operações dos blocos em SysML são transformadas em processos de módulos em SystemC. A criação dos processos é simples, sendo criados sempre no módulo ao qual fazem parte, com inicialização dos mesmos no arquivo .h do módulo e código fonte no arquivo .cpp. Existem duas formas de inicializar os processos diferenciados pelos parâmetros de entrada da operação em SysML:

- **Operações sem parâmetros de entrada:** Este tipo de operação é inicializado com a macro *SC_THREAD*, como pode ser visto na linha 19 da Listagem 3.1. Esta escolha foi feita, pois como não temos um parâmetro que influencia a operação e, por consequência, o processo a ser criado em SystemC, com esta macro o processo é executado uma só vez sem entrar em repetições ilimitadas.
- **Operações com parâmetros de entrada:** Este tipo de operação é inicializado com a macro *SC_THREAD*. Os comandos seguintes são as configurações das portas às quais este processo será sensível com base nas portas utilizadas como parâmetros de entrada da operação. Essa sequência de comandos pode ser vista nas linhas 20 e 21 da Listagem 3.1. Na Listagem 3.2 vemos o código fonte do processo, com comando para execução em *loop* infinito. Esta construção pode ser utilizada neste caso pois, como o processo será executado sensível aos parâmetros de entrada, não corre o risco de haver repetições infinitas e permite que o processo seja executado várias vezes, dando mais liberdade ao projetista do sistema.

Em ambos os casos os processos serão gerados em SystemC sem parâmetros de entrada nas suas declarações. O parâmetro de saída pode ser configurado na modelagem SysML e será usado na geração.

Arquivo principal *main.cpp*

O arquivo *main.cpp* é responsável pela instanciação de todos os módulos, os canais que interligam cada módulo, relacionar as portas de entrada e saída de cada módulo ao seu respectivo canal e de iniciar a simulação. O arquivo é gerado no padrão da Listagem 3.3. Os detalhes de cada parte do arquivo a seguir:

- **Linha 1:** Nas primeiras linhas do arquivo estarão as referências a todos os arquivos .h de todos os módulos que compõem o sistema.
- **Linha 7:** Dentro da rotina *sc_main ()*, as primeiras linhas são destinadas a criação dos canais, que são como os "fios", que conectam cada módulo do sistema. Os canais são gerados com nomes padronizados, *{NomeDoMóduloInicial}_{NomeDoMóduloFinal}_{NomeDaPortaDoMóduloInicial}*. Todos os canais são do tipo *sc_fifo <tipo >*, onde o tipo será o tipo de dados da porta que será conectada a esse canal.
- **Linha 11:** Após a criação dos canais são criadas as instâncias de todos os módulos do sistema.
- **Linha 13:** Após a criação dos canais e dos módulos, as portas de entrada e saída de cada módulo são conectadas aos seus respectivos canais.
- **Linha 16:** No final do arquivos a simulação é iniciada com execução infinita.

```
1  #include "*.h"  //Inclui o .h de todos os módulos
2  #include "systemc.h"
3
4  int sc_main(int argc, char* argv[])
5  {
6  //Criação de todos os canais no padrão
7  sc_fifo<double> NomeDoMóduloInicial_NomeDoMóduloFinal_NomeDaPortaDoMóduloInicial
8  ("NomeDoMóduloInicial_NomeDoMóduloFinal_NomeDaPortaDoMóduloInicial");
9
10 //Instanciação dos módulos
11 NomeDoModulo my_NomeDoModulo("my_NomeDoModulo");
12
13 //Conexão das portas de todos os módulos aos seus respectivos canais
14
15 //Inicio a execução infinitamente
16 sc_start();
17 return 0;
18 }
```

Listagem 3.3: Formação do arquivo *main.cpp*.

Um exemplo de um arquivo *main.cpp* completo pode ser visto nos casos de uso da ferramenta, no exemplo na Listagem 4.1.

Capítulo 4

Estudos de Caso

As seções apresentadas neste capítulo demonstram a utilização da ferramenta **SysMLToSystemC**. Devido à dificuldade de modelar um sistema único que demonstrasse bem as principais características da ferramenta, foram desenvolvidos três pequenos projetos com características e naturezas diferentes, o que resultou em modelagem com diagramas diferentes. O produto final nos três casos é uma especificação executável em SystemC da modelagem SysML. A Seção 4.1 apresenta o projeto de um *pipeline* simples e deve demonstrar a criação dos diversos canais que ligam cada módulo e seus relacionamentos no arquivo principal *main.cpp*. A Seção 4.2 apresenta a modelagem de uma calculadora que realiza as principais operações matemáticas e demonstra a geração de código SystemC para um processo a partir de um Diagrama de Transição de Estados. Na Seção 4.3 há a modelagem de um sistema de uma Máquina de Lavar, com todo o processo de lavagem. Os dois primeiros são exemplos de sistemas orientados a processamento de dados. O sistema da Máquina de Lavar é um exemplo de sistema orientado a controle.

4.1 Pipeline

O primeiro exemplo a ser desenvolvido foi feito com base em um *pipeline* composto de três processos que formam estágios individuais. Este sistema de exemplo foi desenvolvido com base em um dos exemplos da própria biblioteca SystemC [Iniciative, 2010]. Desta forma, após a geração automática da especificação executável foi possível comparar os resultados encontrados com os resultados do sistema original, para confirmar que ambos os sistemas tem o mesmo comportamento. O Diagrama de Definição de Bloco da Figura 4.1 mostra os blocos que compõe o sistema assim como suas conexões.

O primeiro estágio do *pipeline* aceita duas entradas e realiza a soma e a diferença entre elas. O segundo estágio recebe como entrada os resultados do primeiro estágio e calcula o produto e o quociente entre eles. No terceiro estágio as entradas são os resultados calculados no estágio anterior, **prod** e **quot**, e este estágio calcula o valor de $prod^{quot}$.

Após a modelagem do diagrama de definição de bloco na ferramenta Altova UModel, foi acessada a ferramenta **SysMLToSystemC** para geração do código SystemC. Na Figura 4.2 podem ser observadas as opções escolhidas para a geração. Uma opção a ser ressaltada foi a escolha de adicionar uma impressão na entrada de cada método, afim de verificar a entrada em cada um deles.

Outra opção escolhida foi a de escrever código fonte para todas as operações do sistema. Vale ressaltar que o código fonte escrito nesse ponto não é verificado quanto a sua sintaxe SystemC, sendo somente uma tela para facilitar a entrada de informações para as funções. Como o sistema foi desenvolvido com base em um já existente, os cálculos matemáticos das funções foram inseridos como os do sistema original. Na Figura 4.3 pode ser visto o processo de inclusão de código fonte da operação **numgen ()** do módulo responsável por gerar os dados de entrada para o primeiro estágio. Esse passo foi repetido para todas as operações do sistema.

A opção de geração do *testbench* não foi selecionada, pois os módulos de geração de dados de entrada e de leitura e impressão da saída foram modelados diretamente no SysML. A opção de incluir os módulos do *testbench* diretamente na modelagem do sistema fica a cargo do projetista, mas a utilização da opção de geração automática é recomendada pois reduz o tempo gasto na modelagem de dois blocos, na conferência de todas as portas de entrada e saída do sistema, e acelera o tempo até a simulação nos casos de mudança de informações das portas, tais como: tipo, nome, direção e módulos aos quais pertencem.

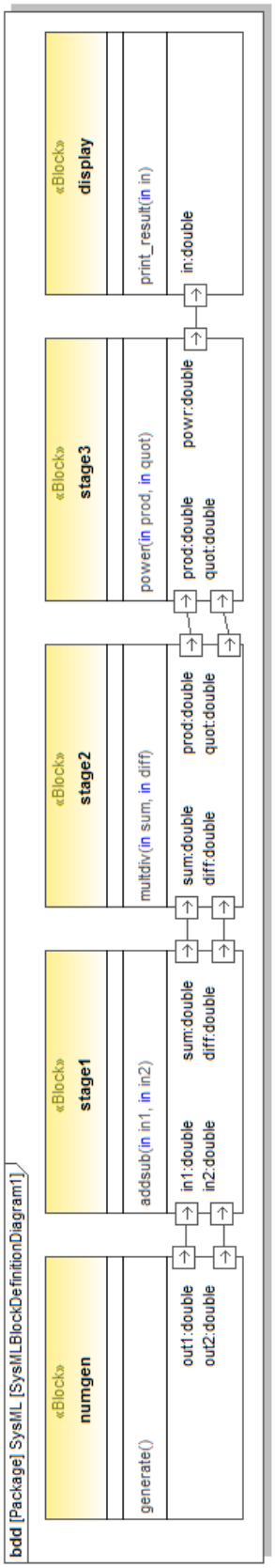


Figura 4.1: Diagrama de Definição de Bloco Pipeline.

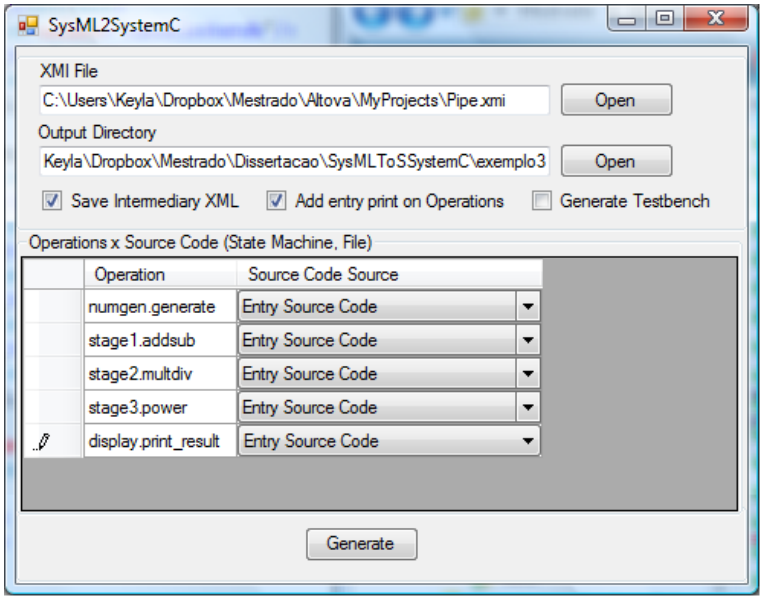


Figura 4.2: Opções escolhidas para geração da Especificação Executável.

Com todas as opções escolhidas o código foi gerado. O arquivo principal do sistema está na Listagem 4.1. Nesse arquivo podem ser vistas as seções geradas, da forma explicada na Seção 3.2.5.1.

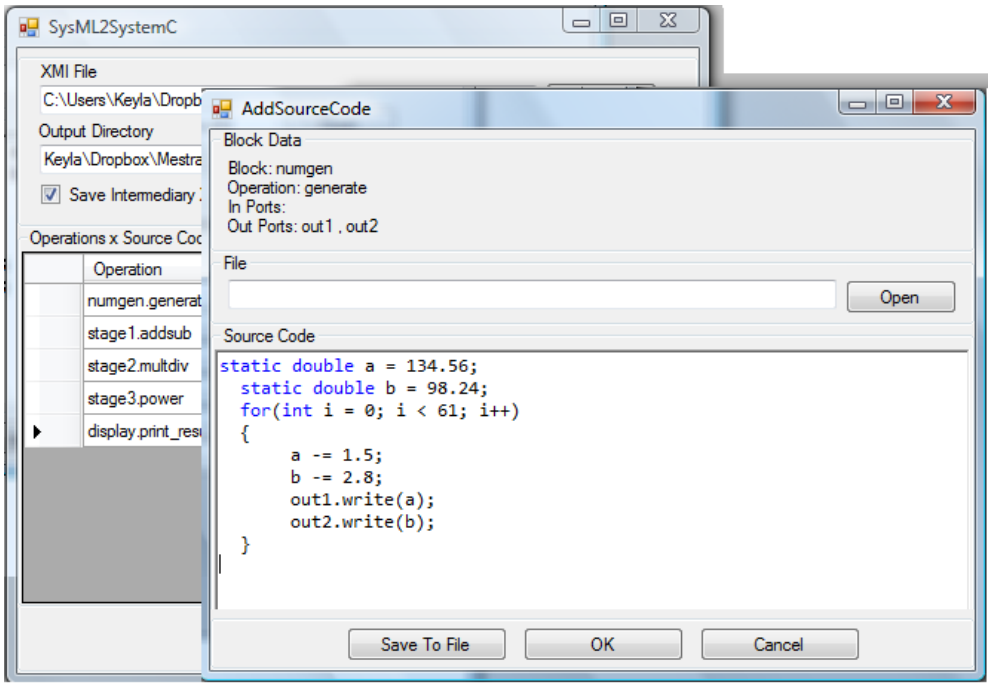


Figura 4.3: Adicionando código para função **numgen()**.

A geração automática do arquivo *main.cpp* é de grande auxílio ao projetista. No arquivo criado por este exemplo, pode ser observada a grande quantidade de linhas e comandos gastos para fazer as conexões das portas aos seus respectivos canais. O sistema utilizado é simples, contendo apenas cinco blocos e poucas portas, e gerou quatorze linhas somente para realizar as conexões, além das instanciações dos sinais e dos módulos com nomes padronizados.

```
#include "numgen.h"
#include "stage1.h"
#include "stage2.h"
#include "stage3.h"
#include "display.h"
#include "systemc.h"
#include "time.h"
int sc_main(int argc, char* argv[])
{
        sc_fifo<double> numgen_stage1_out1("numgen_stage1_out1");
        sc_fifo<double> numgen_stage1_out2("numgen_stage1_out2");
        sc_fifo<double> stage1_stage2_sum("stage1_stage2_sum");
        sc_fifo<double> stage1_stage2_diff("stage1_stage2_diff");
        sc_fifo<double> stage2_stage3_prod("stage2_stage3_prod");
        sc_fifo<double> stage2_stage3_quot("stage2_stage3_quot");
        sc_fifo<double> stage3_display_powr("stage3_display_powr");
        numgen my_numgen("my_numgen");
        stage1 my_stage1("my_stage1");
        stage2 my_stage2("my_stage2");
        stage3 my_stage3("my_stage3");
        display my_display("my_display");
        my_numgen.out1(numgen_stage1_out1);
        my_stage1.in1(numgen_stage1_out1);
        my_numgen.out2(numgen_stage1_out2);
        my_stage1.in2(numgen_stage1_out2);
        my_stage1.sum(stage1_stage2_sum);
        my_stage2.sum(stage1_stage2_sum);
        my_stage1.diff(stage1_stage2_diff);
        my_stage2.diff(stage1_stage2_diff);
        my_stage2.prod(stage2_stage3_prod);
        my_stage3.prod(stage2_stage3_prod);
        my_stage2.quot(stage2_stage3_quot);
        my_stage3.quot(stage2_stage3_quot);
        my_stage3.powr(stage3_display_powr);
        my_display.in(stage3_display_powr);

        sc_start();
        return 0;
}
```

Listagem 4.1: main.cpp.

Devido à opção de impressão na entrada de cada operação, pode ser visto na Listagem 4.2 que a linha com a impressão fica no início da operação sem atrapalhar o código fonte que foi inserido manualmente. Nesta listagem pode ser visto o arquivo

numgen.cpp gerado a partir do código inserido na Figura 4.3. A especificação executável completa deste sistema está listada no Anexo A.

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "numgen.h"
void numgen::generate( )
{
        cout << "Entering numgen::generate" << endl;
        static double a = 134.56;
        static double b = 98.24;
        for(int i = 0; i < 46; i++)
        {
                a -= 1.5;
                b -= 2.8;
                out1.write(a);
                out2.write(b);
                wait(0,SC_NS);
        }
}
```

Listagem 4.2: numgen.cpp.

4.1.1 Comparação com o sistema original

Para que o tempo de execução fosse medido de forma igual, todas as linhas que imprimiam entradas em processos foram comentadas no código gerado automaticamente. Essas linhas, no entanto, foram contadas como linhas de código na comparação da Tabela 4.1 a seguir.

	Sistema Original	Especificação Executável
Linhas de Código	175	186
Tempo Execução (seg)	0.016000	0.017000
Número de Arquivos	11	11

Tabela 4.1: Comparação Pipeline Original x Especificação Executável.

A contagem das linhas de código não levou em conta linhas comentadas e em branco. Durante a contagem, os comandos que estavam numa mesma linha - a título de redução de espaço gasto na impressão do trabalho - foram separados, de forma que cada linha tivesse somente um comando a ser executado e contabilizado. Devido à simplicidade, alto nível de abstração da especificação executável e impressão de entrada em cada um dos processos, o código gerado automaticamente tem mais linhas no total do que o sistema original. O número de arquivos dos dois sistemas é o mesmo, já que ambos tem a mesma quantidade de módulos e arquivo *main.cpp*.

O tempo de execução do sistema original e a especificação executável foi semelhante. O sistema original gastou tempo extra na impressão dos resultados zerados. Em uma execução sem nenhuma impressão sendo feita, o sistema original executou em tempo bem inferior ao da especificação executável gerada automaticamente. A maior diferença entre os dois sistemas, que contribuiu para o aumento no tempo de execução da especificação executável, está na criação dos processos, que foram criados através da macro *SC_METHOD* no sistema original e através da macro *SC_THREAD* na especificação executável[1].

O resultado da execução dos dois sistemas pode ser visto nas Listagens 4.3 e 4.4. O sistema original gera alguns resultados iniciais zerados pois é sensível aos ciclos de um *clock*, fazendo com que o primeiro valor só chegue no último módulo após quatro ciclos do *clock* - pois passa por um módulo em cada ciclo. O sistema gerado automaticamente é sensível aos valores das entradas e por isso não gera valores zerados na saída. Os resultados dos cálculos feitos pela especificação executável deram os mesmos valores dos resultados calculados pelo sistema original, pois, apesar de inicialmente o sistema original gerar alguns resultados zerados, após o início do processamento de dados os dois sistemas retornam a mesma sequência de valores. Isso confirma que todas as portas e conexões foram configuradas de forma correta e que o comportamento do sistema da especificação executável foi o mesmo do sistema original.

[1]Processos do tipo *SC_METHOD* são simulados mais rapidamente do que qualquer outro tipo de processo. Os processos do tipo *SC_THREAD* simulam de forma mais lenta devido a uma sobrecarga na troca de contextos entre o processo e o *kernel* de simulação SystemC, devido a possibilidade de inclusão de **wait()** dentro do processo, o *kernel* SystemC deve reativar o processo assim que a condição ao qual é sensível fica verdadeira. No caso dos processos *SC_METHOD* isso não acontece pois o não é possível a inclusão de **wait()** dentro do código, fazendo com que os processos sejam executados inteiramente a cada execução [Bailey et al., 2007]

```
  SystemC 2.2.0 --- Sep 26 2011 20:21:56
Copyright (c) 1996-2006 by all Contributors
          ALL RIGHTS RESERVED
Result = 0.000000
Result = 0.000000
Result = 0.000000
Result = 0.000000
Result = 788066329449454920000000.000000
Result = 502530815646186170000000.000000
Result = 378550458821278640000000.000000
Result = 331876704603786050000.000000
Result = 33420444988898120000.000000
Result = 3820952118187877900.000000
Result = 490858804069859970.000000
Result = 70202627078491096.000000
Result = 11085844451580030.000000
Result = 1918564708588393.200000
Result = 361468937900785.370000
Result = 73693454684249.922000
Result = 16168706434671.576000
Result = 3798873238107.017600
Result = 951496336681.912960
Result = 253014396064.442320
Result = 71160144395.979309
Result = 21095409040.568798
Result = 6570959918.812003
Result = 2144377535.813800
Result = 731219410.505769
Result = 259897217.280326
Result = 96068017.366532
Result = 36852754.874646
Result = 14643125.538828
Result = 6015770.001653
Result = 2551056.252337
Result = 1114936.454100
Result = 501486.558247
Result = 231829.990700
Result = 110013.001139
Result = 53528.291220
Result = 26676.202538
Result = 13603.019339
Result = 7091.170580
Result = 3775.739764
Result = 2051.852446
Result = 1137.197387
Result = 642.367483
Result = 369.596546
Result = 216.488204
Result = 129.031386
Result = 78.222784
Result = 48.217197
Result = 30.212864
Result = 19.241416
Run time: 0.016000
```

Listagem 4.3: Resultado Original.

```
  SystemC 2.2.0 --- Sep 29 2011 13:28:04
Copyright (c) 1996-2006 by all Contributors
          ALL RIGHTS RESERVED
Result = 788066329449454920000000.000000
Result = 502530815646186170000000.000000
Result = 378550458821278640000000.000000
Result = 331876704603786050000.000000
Result = 33420444988898120000.000000
Result = 3820952118187877900.000000
Result = 490858804069859970.000000
Result = 70202627078491096.000000
Result = 11085844451580030.000000
Result = 1918564708588393.200000
Result = 361468937900785.370000
Result = 73693454684249.922000
Result = 16168706434671.576000
Result = 3798873238107.017600
Result = 951496336681.912960
Result = 253014396064.442320
Result = 71160144395.979309
Result = 21095409040.568798
Result = 6570959918.812003
Result = 2144377535.813800
Result = 731219410.505769
Result = 259897217.280326
Result = 96068017.366532
Result = 36852754.874646
Result = 14643125.538828
Result = 6015770.001653
Result = 2551056.252337
Result = 1114936.454100
Result = 501486.558247
Result = 231829.990700
Result = 110013.001139
Result = 53528.291220
Result = 26676.202538
Result = 13603.019339
Result = 7091.170580
Result = 3775.739764
Result = 2051.852446
Result = 1137.197387
Result = 642.367483
Result = 369.596546
Result = 216.488204
Result = 129.031386
Result = 78.222784
Result = 48.217197
Result = 30.212864
Result = 19.241416
Run time: 0.017000
```

Listagem 4.4: Especificação Executável.

4.2 Calculadora

Neste exemplo pode ser visto um sistema que representa uma calculadora capaz de realizar as operações matemáticas mais comuns: adição, subtração, multiplicação e divisão. Para demonstrar a geração automática do *testbench*, foi criado somente um Diagrama Interno de Bloco com o módulo principal que irá fazer os cálculos. Os diagramas SysML das figuras a seguir foram modelados utilizando a ferramenta Altova UModel. Na Figura 4.4 pode ser visto o Diagrama Interno de Bloco da Calculadora.

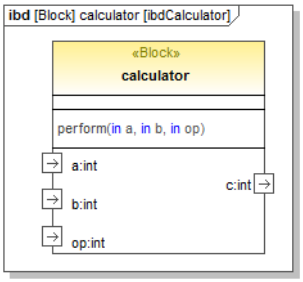


Figura 4.4: Diagrama Interno de Bloco Calculator.

O Bloco Calculator é o principal do sistema e será responsável por realizar as operações de acordo com a entrada **op** nas entradas **a** e **b**, e retornar o resultado na saída **c**. Para descrever o comportamento da sua principal operação **perform ()** foi modelado o Diagrama de Transição de Estados da Figura 4.5.

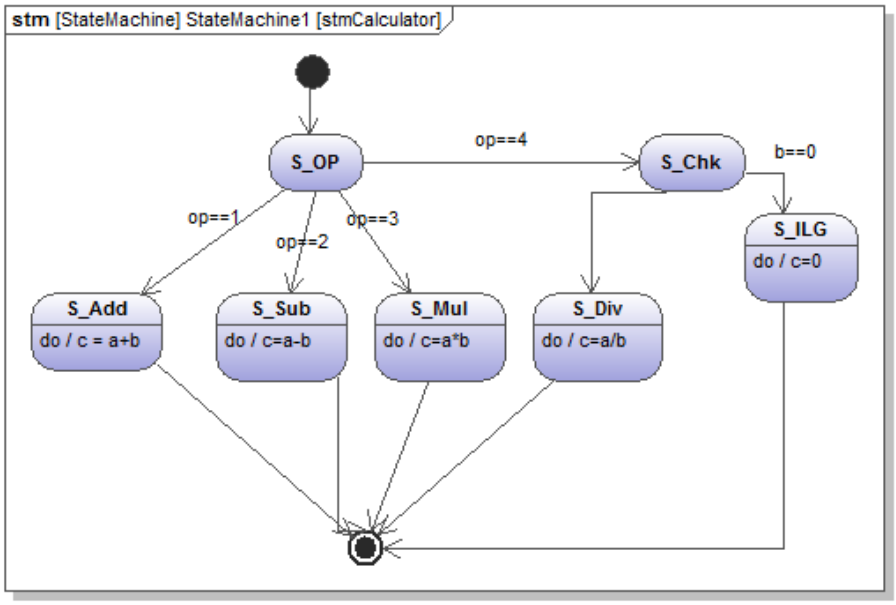


Figura 4.5: Diagrama de Transição de Estados Calculator.

Após a modelagem do sistema estar completa na ferramenta Altova UModel, a ferramenta **SysMLToSystemC** foi executada para a geração da especificação executável em SystemC. A tela inicial permite a seleção das opções a serem utilizadas durante a geração do código em SystemC. Na Figura 4.6 foi feita a escolha do arquivo XMI de entrada e do diretório de saída. Foram escolhidas também as opções de geração do arquivo XML intermediário, impressão na tela da entrada de cada operação e da geração automática do *testbench*. A operação **perform** () do Bloco Calculator foi associada com o comportamento descrito no Diagrama de Transição de Estados.

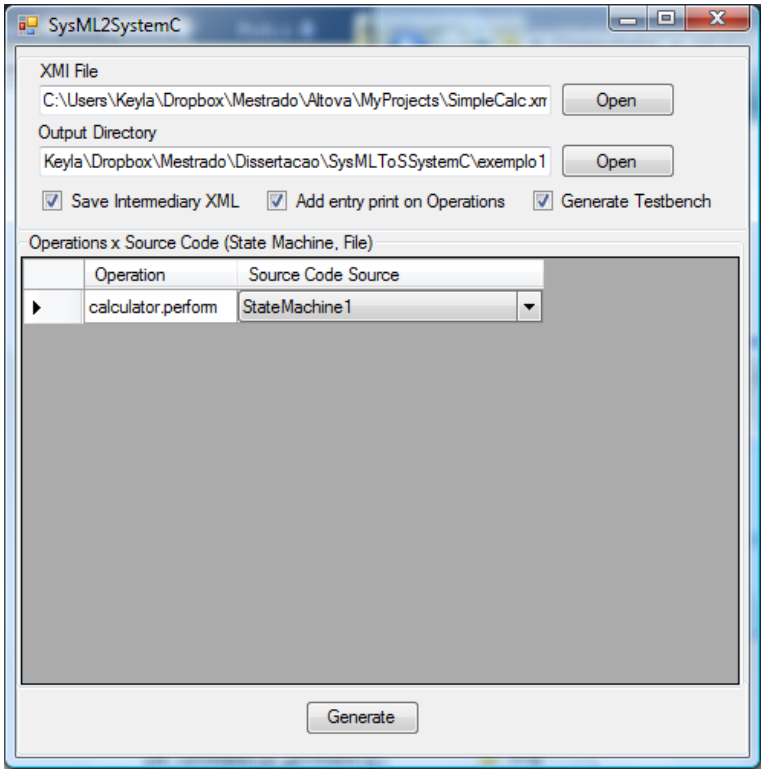


Figura 4.6: Seleção de opções para geração da Especificação Executável SystemC.

Como saída, os arquivos .cpp e .h de cada módulo são gerados, assim também como o arquivo main.cpp e o arquivo XML com a descrição completa do sistema. Nas Listagens 4.5 e 4.6 estão os detalhes de como foi gerada a função **perform** () com base no Diagrama de Transição de Estados.

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(calculator)
{
        sc_fifo_in< int > op;
        sc_fifo_in< int > a;
        sc_fifo_in< int > b;
        sc_fifo_out< int > c;
```

```
        void perform( );
        SC_CTOR(calculator)
        {
                SC_THREAD(perform);
                sensitive << a.data_written();
                sensitive << b.data_written();
                sensitive << op.data_written();
        }
    private:
        enum typeState {FirstState,S_OP,S_Add,S_Sub,S_Mul,S_Div,FinalState,S_Chk,S_ILG};
        typeState CurrentState,InitialState, NextState;
};
```

Listagem 4.5: Código fonte calculator.h.

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "calculator.h"
void calculator::perform( )
{
        while(true)
        {
                wait();
                cout << "Entering calculator::perform" << endl;
                int op_;
                op.nb_read(op_);
                int a_;
                a.nb_read(a_);
                int b_;
                b.nb_read(b_);
                int c_;
                c_ = NULL;
                InitialState = FirstState;
                CurrentState = InitialState;
                NextState = InitialState;
                while (CurrentState != FinalState) {
                        switch (CurrentState)
                        {
                        case FirstState:
                                NextState=S_OP;
                                break;

                        case S_OP:
                                if(op_==1)NextState=S_Add;
                                else if(op_==2)NextState=S_Sub;
                                else if(op_==3)NextState=S_Mul;
                                else if(op_==4)NextState=S_Chk;
                                break;

                        case S_Add:
                                c_ = a_+b_;
                                NextState=FinalState;
                                break;
```

```
                        case S_Sub:
                                c_=a_-b_;
                                NextState=FinalState;
                                break;

                        case S_Mul:
                                c_=a_*b_;
                                NextState=FinalState;
                                break;

                        case S_Div:
                                c_=a_/b_;
                                NextState=FinalState;
                                break;

                        case FinalState:
                                break;

                        case S_Chk:
                                if(b_==0)NextState=S_ILG;
                                else NextState=S_Div;
                                break;

                        case S_ILG:
                                c_=0;
                                NextState=FinalState;
                                break;

                        default:
                                NextState = FinalState ;
                        } //switch
                        //update Current State

                        CurrentState = NextState ;
                }
                c.write(c_);
        }
}
```

Listagem 4.6: Código fonte calculator.cpp.

Nas Listagens 4.7 e 4.8 podem ser vistos os arquivos de cabeçalho dos dois módulos do *testbench* gerado automaticamente. Neles pode ser visto que as portas de entrada e saída foram criadas de acordo com as portas livres do sistema. Na Listagem 4.9 pode ser verificado que esses novos módulos tiveram suas portas corretamente associadas às portas de entrada e saída do módulo principal da Calculadora. O Anexo B contém todos os arquivos gerados, incluindo os dois descritos acima.

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(driver)
{
```

```
        sc_fifo_out< int > op;
        sc_fifo_out< int > a;
        sc_fifo_out< int > b;

        void generate( );
        SC_CTOR(driver)
        {
                SC_THREAD(generate);
        }
};
```

Listagem 4.7: driver.h.

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(monitor)
{
        sc_fifo_in< int > c;

        void show( );
        SC_CTOR(monitor)
        {
                SC_THREAD(show);
                sensitive << c.data_written();
        }
};
```

Listagem 4.8: monitor.h.

```
#include "calculator.h"
#include "driver.h"
#include "monitor.h"
#include "systemc.h"
int sc_main(int argc, char* argv[])
{
sc_fifo<int> driver_calculator_op("driver_calculator_op");
sc_fifo<int> driver_calculator_a("driver_calculator_a");
sc_fifo<int> driver_calculator_b("driver_calculator_b");
sc_fifo<int> calculator_monitor_c("calculator_monitor_c");
calculator my_calculator("my_calculator");
driver my_driver("my_driver");
monitor my_monitor("my_monitor");

my_driver.op(driver_calculator_op); my_calculator.op(driver_calculator_op);
my_driver.a(driver_calculator_a); my_calculator.a(driver_calculator_a);
my_driver.b(driver_calculator_b); my_calculator.b(driver_calculator_b);
my_calculator.c(calculator_monitor_c); my_monitor.c(calculator_monitor_c);

sc_start();
return 0;
}
```

Listagem 4.9: main.cpp.

4.2.1 Resultado da Execução

Após a geração automática da especificação executável, foi inserido código em SystemC nos processos do *testbench*. A geração de dados foi feita a partir de um arquivo com os valores de **a**, **b** e **op** separados por espaço em cada linha. Na Listagem 4.10 estão os dados do arquivo de entrada para a geração de dados. Na Listagem 4.11 pode ser vista a saída gerada, com as impressões de entradas nos processos, e o resultado obtido nas linhas com o valor de **c:**. Pelos resultados pode ser confirmado que todas as operações foram executadas corretamente.

```
#a b op
3 4 1
2 3 2
5 7 3
20 4 4
5 2 3
24 35 3
71 83 2
67 98 1
```

Listagem 4.10: Data_Op.txt.

```
   SystemC 2.2.0 --- Sep 29 2011 13:28:04
Copyright (c) 1996-2006 by all Contributors
            ALL RIGHTS RESERVED
Entering driver::generate
Entering calculator::perform
Entering monitor::show
c:7
Entering calculator::perform
Entering monitor::show
c:-1
Entering calculator::perform
Entering monitor::show
c:35
Entering calculator::perform
Entering monitor::show
c:5
Entering calculator::perform
Entering monitor::show
c:10
Entering calculator::perform
Entering monitor::show
c:840
Entering calculator::perform
Entering monitor::show
c:-12
Entering calculator::perform
Entering monitor::show
c:165
```

Listagem 4.11: Resultados.

4.3 Máquina de Lavar

Este sistema modela uma máquina de lavar comum com a etapa de lavagem. O sistema foi inspirado no exemplo utilizado no trabalho de Shian-Jiun Chiou [Chiou, 2008]. Os blocos que compõem o sistema podem ser vistos na Figura 4.7. As conexões existentes entre todos os blocos podem ser vistas na Figura 4.8.

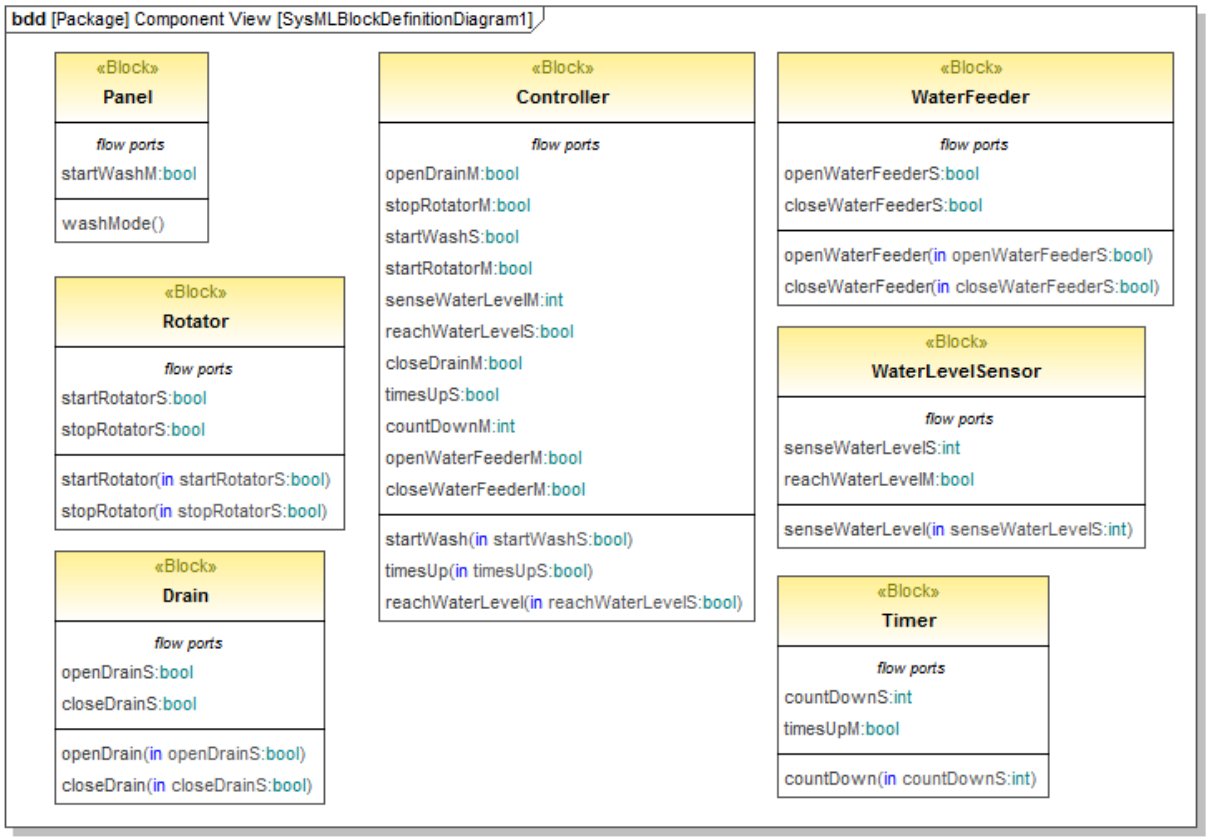


Figura 4.7: Diagrama de Definição de Bloco Máquina de Lavar.

O bloco principal de todo o sistema é o **Controller**, responsável por controlar as etapas a serem executadas durante o processo de lavagem das roupas. O bloco **Panel** representa o painel da máquina onde o processo de lavagem é escolhido e então iniciado. Os demais blocos são responsáveis por encher a máquina de água e controlar o nível - **WaterFeeder** e **WaterLevelSensor** - e módulos para bater a roupa - **Rotator** - e drenar a água - **Drain**.

O sistema tem o mesmo comportamento de uma máquina de lavar comum:

1. Inicia o processo de lavagem;

2. Enche a máquina com água até o nível correto;

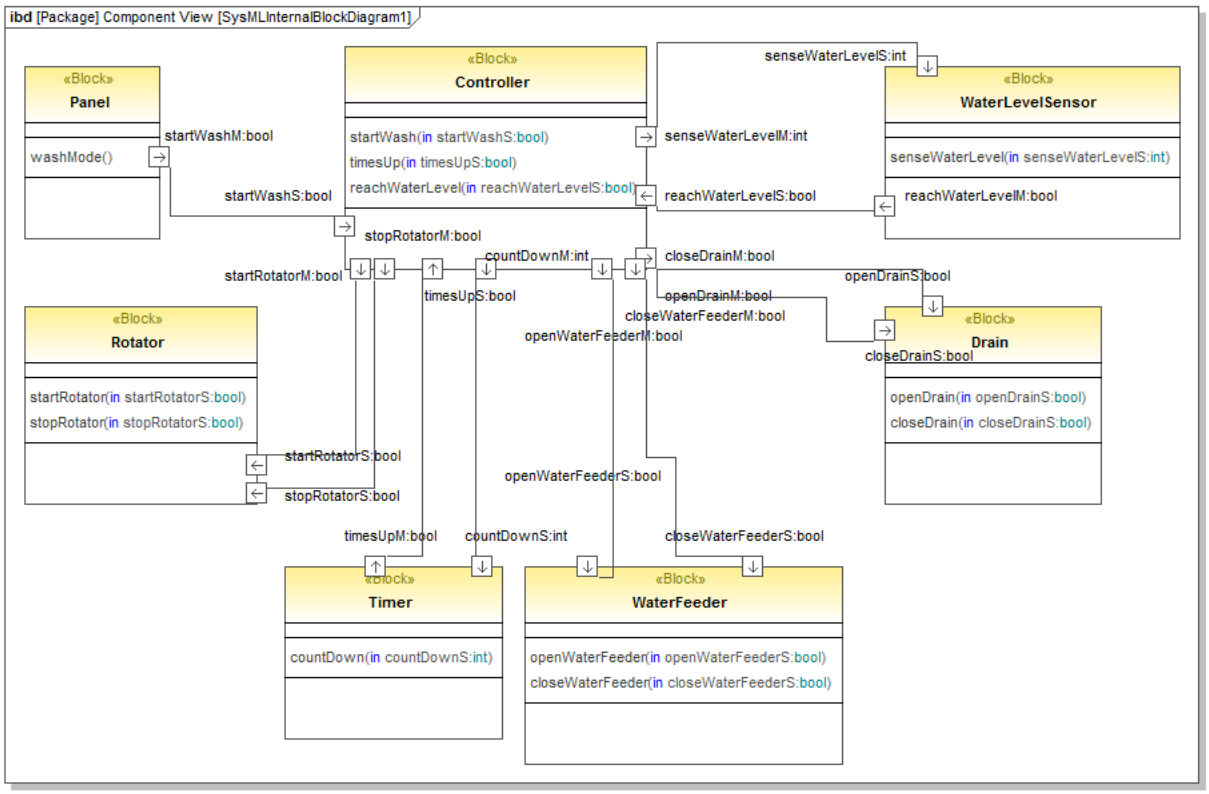


Figura 4.8: Diagrama Interno de Bloco Máquina de Lavar.

3. Bate a roupa para que seja lavada;

4. Escoa a água da máquina.

Esse comportamento do sistema foi modelado utilizando um diagrama de sequência que pode ser visto na Figura 4.10. Nesse diagrama a ordem da execução das operações que são realizadas em cada módulo do sistema pode ser vista com detalhes. O diagrama de sequência foi utilizado na ferramenta apenas para conferência das operações disponíveis em cada bloco. Todas as informações relativas a ele também são exportadas no arquivo XML para que sejam utilizadas por um próximo trabalho que fará o refinamento sucessivo com base nestas informações.

Após a modelagem completa do sistema da máquina de lavar, a ferramenta **SysMLToSystemC** foi executada e as opções para gerar o XML intermediário e impressão na entrada de cada operação foram escolhidas. Devido à existência de algum Diagrama de Sequência no sistema, uma nova opção é mostrada permitindo a escolha de um dos diagramas de sequência como comportamento do sistema. Essa opção fará somente com que o diagrama seja utilizado internamente para conferência dos processos existentes em cada módulo. O comportamento final do sistema não é alterado a partir do diagrama de sequência escolhido. Foi escolhida a opção de introduzir código

fonte para algumas das funções do sistema. Todas essas opções escolhidas podem ser vistas na Figura 4.9.

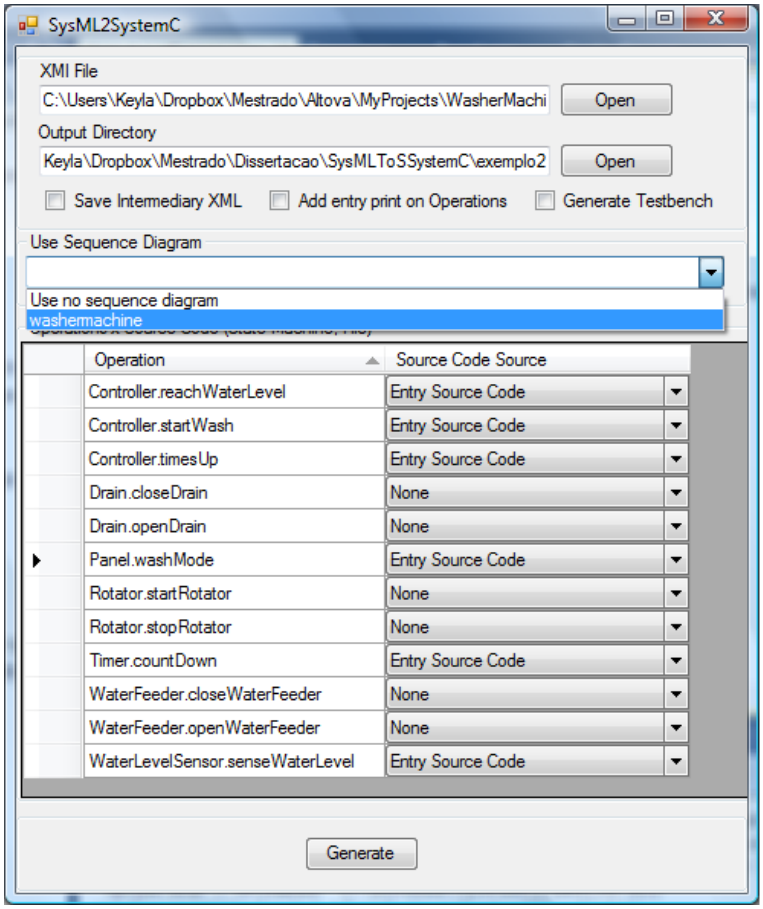


Figura 4.9: Opções escolhidas para geração da Especificação Executável da Máquina de Lavar.

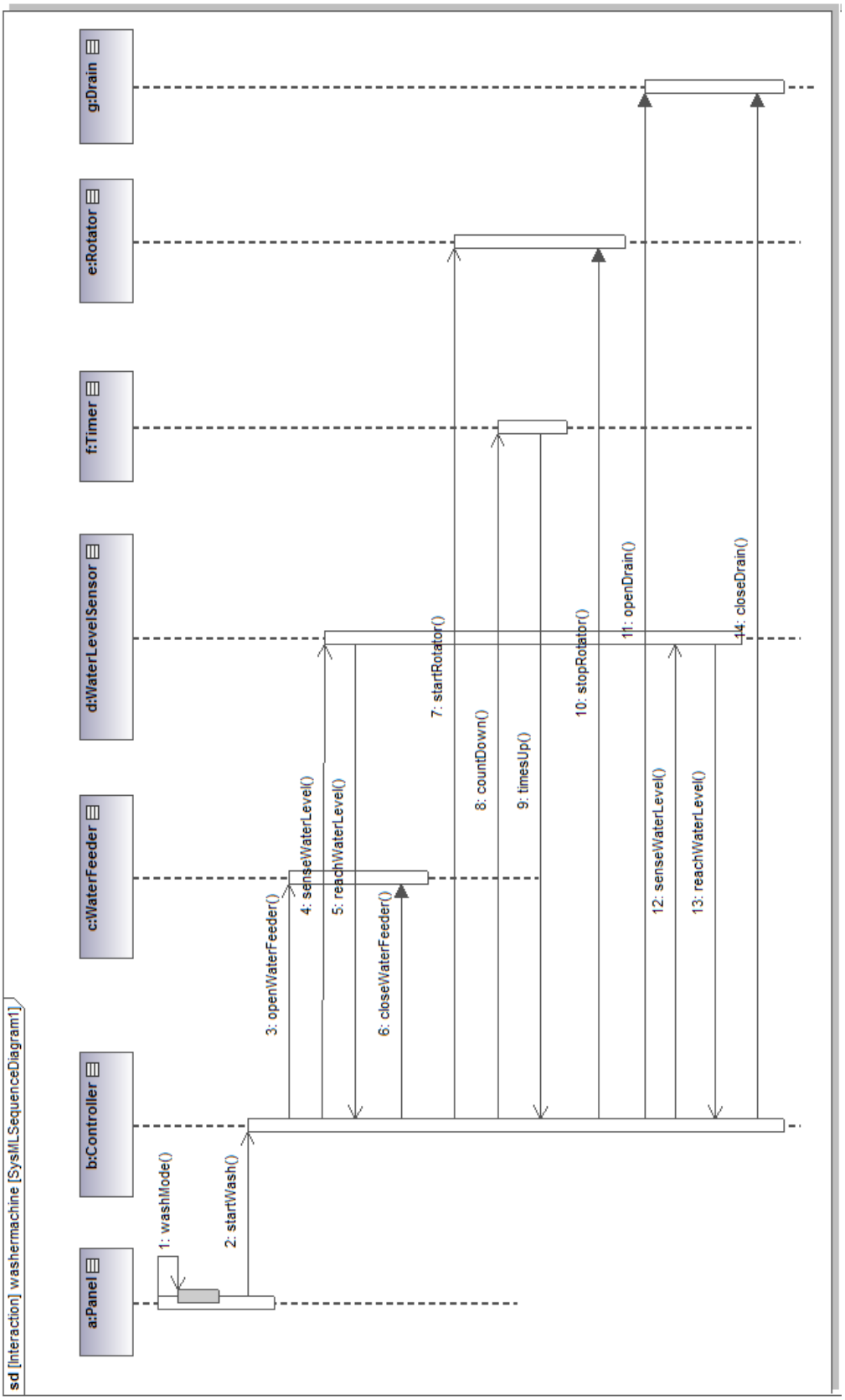


Figura 4.10: Diagrama de Sequência Máquina de Lavar.

Nas Listagens 4.12, 4.13 e 4.14 podem ser vistos o código fonte gerado para o arquivo principal do sistema e os dois arquivos gerados para o módulo Controller. O Anexo C contém a listagem da especificação executável completa gerada pela ferramenta para o sistema da máquina de lavar.

```
#include "Controller.h"
#include "WaterFeeder.h"
#include "WaterLevelSensor.h"
#include "Panel.h"
#include "Rotator.h"
#include "Drain.h"
#include "Timer.h"
#include "systemc.h"

int sc_main(int argc, char* argv[])
{
sc_fifo<int> Controller_WaterLevelSensor_senseWaterLevelM
("Controller_WaterLevelSensor_senseWaterLevelM");
sc_fifo<bool> Controller_WaterLevelSensor_reachWaterLevelS
("Controller_WaterLevelSensor_reachWaterLevelS");
sc_fifo<bool> Controller_Drain_openDrainM
("Controller_Drain_openDrainM");
sc_fifo<bool> Controller_Drain_closeDrainM
("Controller_Drain_closeDrainM");
sc_fifo<bool> Controller_Timer_timesUpS
("Controller_Timer_timesUpS");
sc_fifo<int> Controller_Timer_countDownM
("Controller_Timer_countDownM");
sc_fifo<bool> Controller_WaterFeeder_openWaterFeederM
("Controller_WaterFeeder_openWaterFeederM");
sc_fifo<bool> Controller_WaterFeeder_closeWaterFeederM
("Controller_WaterFeeder_closeWaterFeederM");
sc_fifo<bool> Panel_Controller_startWashM
("Panel_Controller_startWashM");
sc_fifo<bool> Rotator_Controller_startRotatorS
("Rotator_Controller_startRotatorS");
sc_fifo<bool> Rotator_Controller_stopRotatorS
("Rotator_Controller_stopRotatorS");

Controller my_Controller("my_Controller");
WaterFeeder my_WaterFeeder("my_WaterFeeder");
WaterLevelSensor my_WaterLevelSensor("my_WaterLevelSensor");
Panel my_Panel("my_Panel");
Rotator my_Rotator("my_Rotator");
Drain my_Drain("my_Drain");
Timer my_Timer("my_Timer");

my_Controller.senseWaterLevelM(Controller_WaterLevelSensor_senseWaterLevelM);
my_WaterLevelSensor.senseWaterLevelS(Controller_WaterLevelSensor_senseWaterLevelM);
my_Controller.reachWaterLevelS(Controller_WaterLevelSensor_reachWaterLevelS);
my_WaterLevelSensor.reachWaterLevelM(Controller_WaterLevelSensor_reachWaterLevelS);
my_Controller.openDrainM(Controller_Drain_openDrainM);
my_Drain.openDrainS(Controller_Drain_openDrainM);
my_Controller.closeDrainM(Controller_Drain_closeDrainM);
```

```
my_Drain.closeDrainS(Controller_Drain_closeDrainM);
my_Controller.timesUpS(Controller_Timer_timesUpS);
my_Timer.timesUpM(Controller_Timer_timesUpS);
my_Controller.countDownM(Controller_Timer_countDownM);
my_Timer.countDownS(Controller_Timer_countDownM);
my_Controller.openWaterFeederM(Controller_WaterFeeder_openWaterFeederM);
my_WaterFeeder.openWaterFeederS(Controller_WaterFeeder_openWaterFeederM);
my_Controller.closeWaterFeederM(Controller_WaterFeeder_closeWaterFeederM);
my_WaterFeeder.closeWaterFeederS(Controller_WaterFeeder_closeWaterFeederM);
my_Panel.startWashM(Panel_Controller_startWashM);
my_Controller.startWashS(Panel_Controller_startWashM);
my_Rotator.startRotatorS(Rotator_Controller_startRotatorS);
my_Controller.startRotatorM(Rotator_Controller_startRotatorS);
my_Rotator.stopRotatorS(Rotator_Controller_stopRotatorS);
my_Controller.stopRotatorM(Rotator_Controller_stopRotatorS);

sc_start();
return 0;
}
```

Listagem 4.12: main.cpp.

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Controller)
{
        sc_fifo_out< bool > openDrainM;
        sc_fifo_out< bool > stopRotatorM;
        sc_fifo_in< bool > startWashS;
        sc_fifo_out< bool > startRotatorM;
        sc_fifo_out< int > senseWaterLevelM;
        sc_fifo_in< bool > reachWaterLevelS;
        sc_fifo_out< bool > closeDrainM;
        sc_fifo_in< bool > timesUpS;
        sc_fifo_out< int > countDownM;
        sc_fifo_out< bool > openWaterFeederM;
        sc_fifo_out< bool > closeWaterFeederM;

        void startWash( );
        void reachWaterLevel( );
        void timesUp( );
        SC_CTOR(Controller)
        {
                SC_THREAD(startWash);
                sensitive << startWashS.data_written();
                SC_THREAD(reachWaterLevel);
                sensitive << reachWaterLevelS.data_written();
                SC_THREAD(timesUp);
                sensitive << timesUpS.data_written();
        }
};
```

Listagem 4.13: controller.h.

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project

#include "systemc.h"
#include "Controller.h"
void Controller::startWash( )
{
        while(true)
        {
                wait();
                cout << "Entering Controller::startWash" << endl;
                openWaterFeederM.write(true);
                senseWaterLevelM.write(1);
        }
}
void Controller::reachWaterLevel( )
{
        while(true)
        {
                wait();
                cout << "Entering Controller::reachWaterLevel" << endl;
                bool t;
                if(timesUpS.nb_read(t))
                {
                        closeDrainM.write(true);
                }
                else
                {
                        closeWaterFeederM.write(true);
                        startRotatorM.write(true);
                        countDownM.write(1);
                }
        }
}
void Controller::timesUp( )
{
        while(true)
        {
                wait();
                cout << "Entering Controller::timesUp" << endl;
                stopRotatorM.write(true);
                openDrainM.write(true);
                senseWaterLevelM.write(2);
        }
}
```

Listagem 4.14: controller.cpp.

4.3.1 Resultado da Execução

Após a geração do código fonte em SystemC a especificação executável foi testada e gerou a saída da Listagem 4.15. A ordem das operações foi exatamente a esperada e modelada nos diagramas SysML, confirmando que a especificação executável gerada automaticamente tem o comportamento esperado do sistema.

```
   SystemC 2.2.0 --- Sep 29 2011 13:28:04
Copyright (c) 1996-2006 by all Contributors
           ALL RIGHTS RESERVED

Entering Panel::washMode
Entering Controller::startWash
Entering WaterFeeder::openWaterFeeder
Entering WaterLevelSensor::senseWaterLevel
Entering Controller::reachWaterLevel
Entering WaterFeeder::closeWaterFeeder
Entering Rotator::startRotator
Entering Timer::countDown
Entering Controller::timesUp
Entering Rotator::stopRotator
Entering Drain::openDrain
Entering WaterLevelSensor::senseWaterLevel
Entering Controller::reachWaterLevel
Entering Drain::closeDrain
```

Listagem 4.15: Resultado da execução da Máquina de Lavar.

Capítulo 5

Conclusão

Neste trabalho foi desenvolvida a ferramenta **SysMLToSystemC** de apoio a uma metodologia de desenvolvimento de SoCs para reduzir o esforço gasto pelo projetista durante as etapas iniciais do projeto. A ferramenta desenvolvida gera especificações executáveis em SystemC automaticamente a partir de diagramas SysML. O trabalho também apresentou uma visão geral dos trabalhos já desenvolvidos relativos ao uso combinado de diagramas UML/SysML e a transformação destes para código fonte SystemC.

A transformação dos diagramas estruturais SysML e dos diagramas comportamentais de Máquina de Transição de Estados e de Sequência para código fonte SystemC é baseada em arquivos XMI, XML e modelos XSLT. Um arquivo XML intermediário de saída foi criado de forma a facilitar a troca de informações com outros aplicativos e permitir a extensão das funcionalidades já desenvolvidas por este trabalho. A utilização de modelos XSLT também foi feita para permitir mudanças na forma de geração dos arquivos sem a necessidade de mudança na ferramenta criada.

O funcionamento da ferramenta **SysMLToSystemC** foi validado através da aplicação da ferramenta apresentada em três sistemas de naturezas diferentes. O processo de geração das especificações executáveis dos três foi detalhado no Capítulo 4. Em todos os três casos, o comportamento da especificação executável gerada automaticamente com o uso da ferramenta desenvolvida foi igual ao comportamento do sistema original ou de acordo com o esperado - como no caso do sistema da máquina de lavar, no qual não havia um sistema original sendo executado para comparação.

5.1 Contribuições do Trabalho

A contribuição do trabalho está no fato de reduzir o tempo gasto no processo do desenvolvimento de sistemas e de reduzir a possibilidade de erros em partes do código que serão geradas automaticamente.

Uma contribuição indireta do trabalho está na criação de uma documentação visual dos sistemas a serem desenvolvidos. Devido a necessidade da modelagem inicial do sistema em SysML, o projetista poderá utilizar os diagramas modelados como parte da documentação do sistema.

Este trabalho também contribui cientificamente na pesquisa do tópico relativo a transformação entre SysML e SystemC pois vários trabalhos já foram desenvolvidos no tópico da transformação de modelos UML para SystemC mas ainda existem poucos trabalhos com base em diagramas SysML, sendo assim um tópico ainda pouco explorado.

- **Geração da Especificação Executável:** A especificação executável em SystemC é gerada automaticamente podendo ser utilizada tanto para simulação quanto para síntese. A tradução entre modelos é tradicionalmente feita de forma manual, correndo risco de erros humanos como falta de partes do sistema, mudanças de tipos de portas e também erros de sintaxe em SystemC. A utilização de uma ferramenta de automatização acelera as fases iniciais do projeto e elimina erros não intencionais que ocorrem eventualmente na escrita manual de códigos fontes e que consomem tempo de depuração posteriormente.

- **Geração Automática do *main.cpp*:** O arquivo *main.cpp* contém as inicializações dos módulos e as ligações entre eles. Essa tarefa precisa de muita precisão e envolve uma grande quantidade de linhas. Em sistemas maiores e com muitos módulos, é comum os módulos serem desenvolvidos por pessoas diferentes. A geração automática deste arquivo com as ligações existentes entre os módulos do sistema reduz a possibilidade de erros e o tempo gasto pelo responsável por unir todos os módulos - que, sem a ferramenta, teria que ler manualmente cada um dos módulos, os nomes das portas e quais teriam conexão com portas de outros módulos. Este é um ponto onde ocorrem falhas no código, gerando problemas de comunicação entre os módulos e que são resolvidos com a geração do arquivo através da ferramenta **SysMLToSystemC**.

- **Geração Automática do *Testbench*:** A geração automática do *testbench*, com a inclusão do módulo de geração de dados de entrada (*driver*) e um monitor de dados de saída (*monitor*), reduz o tempo gasto da modelagem até a primeira simulação do sistema. A criação das portas de entrada e saída dos módulos do *testbench* automaticamente a partir da modelagem do sistema também reduz o custo gasto com mudanças no projeto, agilizando a implementação.

- **Padronização de nomes:** A padronização dos nomes dos módulos e sinais facilita a compreensão e manutenção dos sistemas desenvolvidos. Tendo em vista que várias pessoas podem trabalhar no desenvolvimento de um mesmo sistema, corrigindo e evoluindo, é de extrema importância para compreensão de todos o que existe nos sistemas existentes. O padrão criado para os nomes dos sinais também auxilia na validação do sistema, já que ele indica, através da nomenclatura, os módulos e portas que estão ligados àquele sinal, reduzindo o tempo gasto na procura de um possível erro do sistema.

- **Documentação Visual do Sistema:** Um documento visual descrevendo o sistema serve de ponte para a comunicação entre os diversos interessados do sistema,uma vez que define seus elementos e relações que o compõem. O documento também pode ser utilizado para uma análise do sistema. A documentação também reduz a possibilidade de um sistema diferente do esperado pelo cliente, pois na, falta de um documento, a especificação pode ser imprecisa e podem existir pontos de discrepâncias na especificação executável por não existir uma documentação explícita.

5.2 Trabalhos Futuros

A ferramenta desenvolvida gera um arquivo XML com a descrição do sistema na expectativa de que outras ferramentas possam utilizar a descrição do sistema sem a necessidade de nova integração com a ferramenta Altova UModel. Outros desenvolvimentos serão necessários para complementar os resultados apresentados neste trabalho.

- **Utilização de outros diagramas SysML e importação de mais informações disponíveis:** Devido a geração da especificação executável estar num nível de abstração alto, não foi necessária a importação de todas as informações disponibilizadas pelos diagramas SysML utilizados. A importação de novos dados contidos nos diagramas já utilizados e a importação das informações contidas nos demais diagramas SysML irá gerar um arquivo XML de saída mais completo,

com informações que podem ser utilizadas na geração de código SystemC em níveis de abstração mais baixos. O Diagrama de Atividades pode ser utilizado em conjunto com o Diagrama de Sequência para extração de informações relativas a comunicação. A utilização do Diagrama de Requisitos pode ser útil em um trabalho futuro de verificação formal e validação dos requisitos atendidos pelo sistema.

- **Refinamentos Sucessivos:** A ferramenta desenvolvida gera uma especificação executável do sistema modelado. Ainda há a necessidade do desenvolvimento futuro de uma ferramenta que faça o refinamento sucessivo automatizado dessa especificação, reduzindo o tempo gasto pelo projetista na tarefa de analisar o código em busca de estruturas que devem passar pelo refinamento.

- **Integração com Núcleo de Simulação e *Parser* SystemC:** A ferramenta **SysMLToSystemC** não efetua a conferência da sintaxe SystemC introduzida manualmente pelo usuário como código fonte de processos e nem efetua a simulação do sistema. A possibilidade de realizar essas ações na própria ferramenta reduz ainda mais o tempo gasto pelo projetista, já que ele poderia desenvolver o sistema completamente num mesmo ambiente.

- **Validação da ferramenta em sistemas mais complexos:** A ferramenta desenvolvida foi validada através de três sistemas simples. Devido ao nível de abstração alto da especificação executável, um sistema mais complexo iria ser diferente somente no número de módulos criados e num arquivo *main.cpp* com mais linhas. Ainda assim é interessante que no futuro a ferramenta seja validada com um sistema maior e mais complexo afim de verificar o comportamento e tempo gasto pela ferramenta para a geração automática da especificação executável.

- **Inclusão de marcadores no código fonte:**Uma funcionalidade interessante a ser inserida numa futura versão da ferramenta é a possibilidade do projetista adicionar marcadores no código fonte escrito manualmente, de forma que novas execuções da ferramenta não reutilizem o código marcado e não sobrescreva com código novo. Essa funcionalidade facilitará ainda mais com que mudanças feitas na modelagem sejam atualizadas na especificação executável sem perda de parte do código já validado.

- **Comentários nos Diagramas SysML que sejam salvos na especificação executável em SystemC :** A possibilidade de inclusão de comentários na modelagem dos diagramas SysML, que possam ser salvos na especificação executável nos respectivos módulos, portas, processos e sinais, aumenta a qualidade do código SystemC gerado, que será mais documentado e permitirá um maior entendimento do sistema pelo desenvolvedor que fará manutenções no código.

- **Engenharia Reversa:** A geração do arquivo XMI com a descrição do sistema em diagramas SysML, que possa ser importado pela ferramenta Altova UModel, auxilia na criação do documento visual de sistemas já existentes. A importação através da ferramenta seria feita sem a necessidade inicial do aprendizado da modelagem em SysML e permitiria o aproveitamento dos demais benefícios que a ferramenta já tem e terá após novas melhorias.

- **Repetição de Instâncias:** Atualmente para a geração de várias instâncias de um mesmo módulo (exemplo, memória) o projetista deveria modelar diagramas SysML idênticos, o que geraria vários módulos idênticos e mas com nomes diferentes. Uma futura melhoria seria o tratamento de possíveis heranças de blocos, de forma a criar instâncias com nomes diferentes de um mesmo módulo.

Referências Bibliográficas

Andersson, P. & Höst, M. (2008). UML and SystemC–A Comparison and Mapping Rules for Automatic Code Generation. *Embedded Systems Specification and Design Languages*, pp. 199--209.

Badawy, W. & Jullien, G. (2003). *System-on-chip for Real-time Applications*. Kluwer Academic Pub.

Bailey, B.; Grant, M. & Anderson, T. (2005). *Taxonomies for the development and verification of digital systems*. Springer Verlag.

Bailey, B.; Martin, G. & Piziali, A. (2007). *ESL design and verification: a prescription for electronic system-level methodology*. Morgan Kaufmann Pub.

Benini, L. & De Micheli, G. (2002). Networks on chips: A new soc paradigm. *COMPUTER,*, pp. 70--78.

Black, D.; Donovan, J.; Bunton, B. & Keist, A. (2009). *SystemC: from the ground up*. Springer Verlag.

Boutekkouk, F. (2010). Automatic SystemC Code Generation from UML Models at Early Stages of Systems on Chip Design. *International Journal of Computer Applications*, 8(6):10--17.

Cai, L. & Gajski, D. (2003). Transaction level modeling: An overview. In *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '03, pp. 19--24, New York, NY, USA. ACM.

Chiou, S.-J. (2008). Translating SysML Specification to SystemC Transaction-Level Modeling. Master's thesis, Department of Computer Science and Engineering Tatung University, Taipei, Taiwan(R.O.C.).

DIAS, S. (2007). Srd: Uma ferramenta de apoio ao projetista de sistemas de hardware utilizando a linguagem systemc. *UFMG, Belo Horizonte*.

Forum, S. (2010). `http://www.sysmlforum.com/`.

Friedenthal, S.; Moore, A. & Steiner, R. (2008). *A practical guide to SysML: The systems modeling language*. Elsevier.

Ghenassia, F. (2006). *Transaction-Level Modeling with Systemc: Tlm Concepts and Applications for Embedded Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Grötker, T.; Liao, S.; Martin, G. & Swan, S. (2002). *System design with SystemC*. Springer Netherlands.

Iniciative, O. S. (2010). `http://www.systemc.org/home`.

Initiative, O. (2002). Systemc version 2.0 user's guide. *Update for SystemC 2.0*, 1.

Initiative, O. (2005). IEEE 1666: SystemC Language Reference Manual, 2005.

Kay, M. (2002). Xsl transformations (xslt) version 2.0. *W3C Working Draft*, 16.

Keutzer, K.; Malik, S.; Member, S.; Newton, A. R.; Rabaey, J. M. & Sangiovanni-vincentelli, A. (2000). System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19:1523--1543.

Kreku, J.; Hoppari, M.; Tiensyrjä, K. & Andersson, P. (2007). SystemC Work load Model Generation from UML for Performance Simulation. In *Proceedings of the International Conference Forum on Specification and Design Languages (FDL) 2007*.

Language, O. S. M. (2010). `http://www.omgsysml.org`.

Miller, J.; Mukerji, J. et al. (2001). Model driven architecture (mda). *Object Management Group, Draft Specification ormsc/2001-07-01*.

Miller, J.; Mukerji, J. et al. (2003). MDA Guide Version 1.0.1. *Object Management Group*, 234.

Object Management Group, I. (2010). `www.omg.org`.

OMG (2003). UML for systems engineering RFP, OMG Document: ad/03-03-41.

OMG, M. (2007). XMI Mapping Specification, v2. 1.

OMG, M. (2010). Meta Object Facility (MOF) Core Specification, Version 2.4.

OMG, X. (2011). XML Metadata Interchange (XMI®).

Papa, R. (2006). Geracao de especificacoes executaveis para o projeto de modulos para sistemas em chips. *UFMG, Belo Horizonte*.

Pasricha, S. (2002). Transaction level modeling of SoC with SystemC 2.0. In *Synopsys User Group Conference (SNUG)*, volume 3, p. 3. Citeseer.

Pauwels, M.; Vanderperren, Y.; Sonck, G.; Oostende, P.; Dehaene, W. & Moore, T. (2004). A design methodology for the development of a complex system-on-chip using uml and executable system models. *System Specification & Design Languages*, pp. 129--141.

Prevostini, M. & Zamsa, E. (2007). SysML Profile for SoC Design and SystemC Transformation. *ALaRI, Faculty of Informatics University of Lugano via G. Buffi*, 13.

Raslan, W. & Sameh, A. (2007a). Mapping SysML to SystemC. In *Proc. Forum spec. & Design Lang.(FDL)*.

Raslan, W. & Sameh, A. (2007b). System-Level Modeling and Design Using SysML and SystemC. In *Integrated Circuits, 2007. ISIC'07. International Symposium on*, pp. 504--507.

Riccobene, E. (2005). A HW/SW Co-design environment based on UML and SystemC. In *Proceedings of the Forum on Specification and Design Languages (FDL'05)*.

Riccobene, E.; Scandurra, P.; Rosti, A. & Bocchio, S. (2005). A SoC design methodology involving a UML 2.0 profile for SystemC. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pp. 704--709. IEEE Computer Society.

Rose, A.; Swan, S.; Pierce, J. & Fernandez, J. (2005). Transaction level modeling in SystemC. *Open SystemC Initiative*.

SysML, O. (2006). The OMG systems modeling language.

SysML, O. (2008). OMG Systems Modeling Language (OMG SysML) version 1.1.

Tutorial, X. (2011). `http://www.w3schools.com/xsl/default.asp`.

UModel, A. (2011). `http://www.altova.com/umodel.html`.

Weilkiens, T. (2007). *Systems engineering with SysML/UML: modeling, analysis, design.* Morgan Kaufmann.

Xi, C.; JianHua, L.; ZuCheng, Z. & YaoHui, S. (2005). Modeling SystemC design in UML and automatic code generation. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, p. 935. ACM.

Apêndice A

Desenvolvimento do *Plugin* para ferramenta Altova UModel

A ferramenta de modelagem Altova UModel permite a criação de *plugins* que podem ser integrados na própria ferramenta. Essa extensão da ferramenta permite, entre outras modificações, a criação de menus novos e execução de operações relacionadas ao acesso aos menus criados.

A extensão para a ferramenta Altova UModel deve ser criada na forma de uma DLL desenvolvida em C♯. A DLL criada deve referenciar e extender a biblioteca **UModelPlugInLib.dll**, disponibilizada pela própria ferramenta Altova UModel, conforme a Figura A.1.

```
using System;
using System.Text;
using UModelPlugInLib;

namespace GenerateSystemCPlugin
{
    public class GenerateSystemCPlugin : IUModelPlugIn
    {
        IUModelPlugIn Members
    }

}
```

Figura A.1: Criação do Plugin para Altova UModel

Para facilitar o desenvolvimento e teste do nosso *plugin* ele foi desenvolvido em uma segunda DLL que é chamada pela DLL que implementa a biblioteca de *plugins* disponibilizada pela ferramenta Altova UModel. No momento da chamada do menu criado a janela do *plugin* de geração do código fonte em SystemC é criada e centralizada no monitor. Esta parte do código fonte pode ser vista na Figura A.2.

```
public void OnCommand(int nID, object pUModel)
{
    SysML2SystemC c = new SysML2SystemC();
    c.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
    c.Show();
}
```

Figura A.2: Código fonte para abertura da janela do Plugin

O código fonte completo da DLL que cria o *plugin* e executa a ferramenta de geração de código fonte pode ser visto na Listagem A.1.

```
ï»¿using System;
using System.Text;
using UModelPlugInLib;

namespace GenerateSystemCPlugin
{
    public class GenerateSystemCPlugin : IUModelPlugIn
    {
        #region IUModelPlugIn Members

        public string GetDescription()
        {
            return "GenerateSystemCPlugin;GenerateSystemCPlugin generates SystemC source code from XMI
        }

        public string GetUIModifications()
        {
            return "<ConfigurationData>" +
                        "<Modifications>" +
                            "<Modification>" +
                                "<Action>Add</Action>" +
                                "<UIElement type=\"MenuItem\">" +
                                    "<ID>1</ID>" +
                                    "<Name>Generate SystemC Code...</Name>" +
                                    "<Info>Generate SystemC Code</Info>" +
                                    "<Place>0</Place>" +
                                    "<MenuID>101</MenuID>" +
                                    "<Parent>:Project</Parent>" +
                                "</UIElement>" +
                            "</Modification>" +
                        // add Seperator to Edit menu
                            "<Modification>" +
                                "<Action>Add</Action>" +
                                "<UIElement type=\"MenuItem\">" +
                                    "<ID>0</ID>" +
                                    "<Place>1</Place>" +
                                    "<MenuID>101</MenuID>" +
                                    "<Parent>:Project</Parent>" +
                                "</UIElement>" +
                            "</Modification>" +
                       // finish modification description
                        "</Modifications>" +
                    "</ConfigurationData>";
```

```
        }

        public void OnCommand(int nID, object pUModel)
        {
            SysML2SystemC c = new SysML2SystemC();
            c.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            c.Show();
        }

        public void OnInitialize(object pUModel)
        {
            // before processing DDE or batch commands
        }

        public void OnRunning(object pUModel)
        {
            // DDE or batch commands are processed; application is fully initialized
        }

        public void OnShutdown(object pUModel)
        {
            // application will shutdown; release all unused objects
        }

        public UModelUpdateAction OnUpdateCommand(int nID, object pUModel)
        {
            if (nID == 1)
               return UModelUpdateAction.UModelUpdateAction_Enable;
            return UModelUpdateAction.UModelUpdateAction_Disable;
        }

      #endregion
    }

}
```

Listagem A.1: GenerateSystemC.cs

A.1 Instalação do *Plugin* na ferramenta Altova UModel

Para a instalação e funcionamento correto do *plugin* **SysMLToSystemC** é necessário criar um diretório com os seguintes arquivos dentro:

- SysML2SystemC.dll
- SysML2SystemCPlugin.dll
- SysML2SystemCPlugin.tlb

- Interop.UModelPlugInLib.dll
- xml2systemccpp.xslt
- xml2systemchdata.xslt
- xml2systemcmain.xslt

Pelo menu **Tools|Customize** da ferramenta Altova UModel escolher a aba de **Plug-Ins** e o botão **Add Plug-in...**. Nesse momento escolher o arquivo *SysML2SystemCPlugin.dll*, conforme Figura A.3.

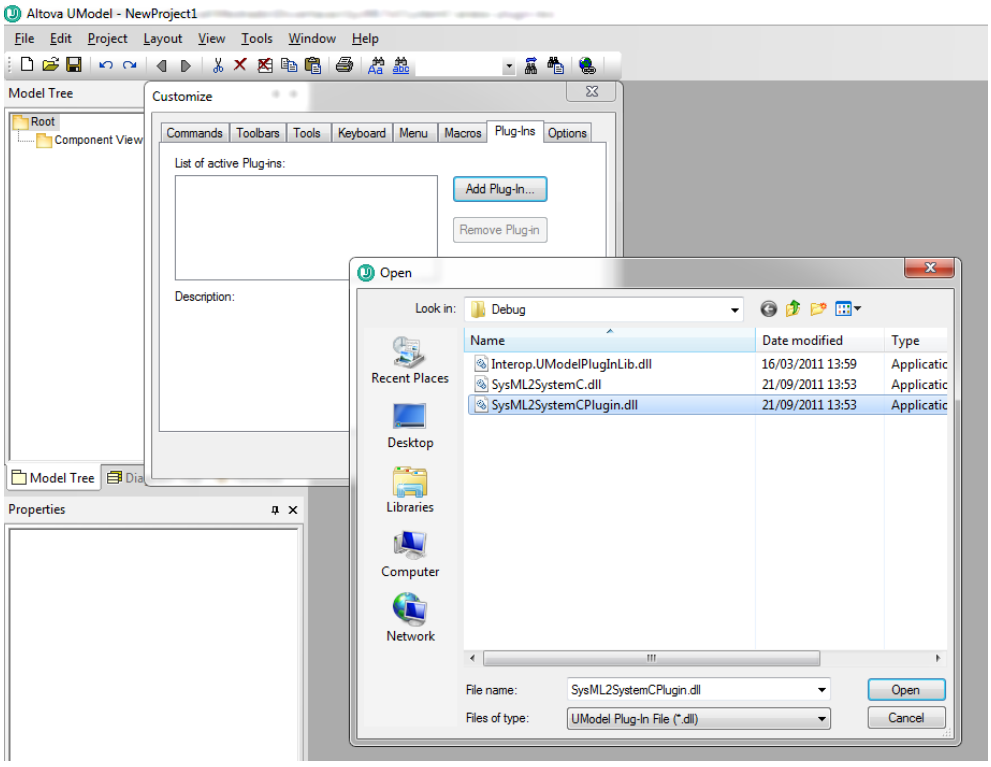


Figura A.3: Processo de instalação do Plugin

Após a instalação o novo *plugin* está disponível entre os demais instalados, com a descrição relativa abaixo. A confirmação da instalação pode ser vista na Figura A.4.

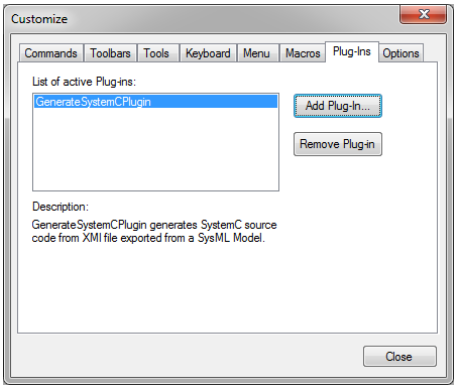


Figura A.4: Confirmação de Plugin instalado com sucesso

Apêndice B

Arquivos de Estilo XSLT

Os arquivos XLSTs foram criados de forma a deixar a possibilidade de algumas mudanças no padrão de geração dos arquivos em SystemC sem a necessidade de mudança na ferramenta. Nos três casos abaixo o arquivo intermediário XML é utilizado como arquivo de origem a ser modificado pelos arquivos de estilo XSLT.

Nas Listagens B.1 e B.2 podem ser vistos os dois arquivos XSLT utilizados para a geração dos arquivos .cpp e .h respectivamente dos módulos SystemC. Na Listagem B.3 temos o arquivo XSLT utilizado para a geração do arquivo principal *main.cpp*.

Na Seção temos um tutorial de arquivos de estilo XSLT e como utilizá-los para a personalização da especificação executável a ser gerada.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method='text'/>
        <xsl:template match="/">
                <xsl:for-each select="Model/block">
      <xsl:text>//This file is automatically generated</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>//by SysML2SystemC Project</xsl:text>
      <xsl:text>&#xa;</xsl:text>
                        <xsl:text>&#xa;</xsl:text>
                        <xsl:text>#include "systemc.h"</xsl:text>
                        <xsl:text>&#xa;</xsl:text>
                        <xsl:text>#include "</xsl:text>
                        <xsl:value-of select="name"/>
                        <xsl:text>.h"</xsl:text>
                        <xsl:text>&#xa;</xsl:text>
                        <xsl:variable name="blockname">
                                <xsl:value-of select="name"/>
                        </xsl:variable>
                        <xsl:for-each select="operation">
                                <xsl:text>void </xsl:text>
                                <xsl:copy-of select="$blockname" />
                                <xsl:text>::</xsl:text>
                                <xsl:value-of select="name"/>
```

```
                                <xsl:text>( </xsl:text>
                                <xsl:value-of select="parameterdone"/>
                                <xsl:text>)</xsl:text><xsl:text>&#xa;</xsl:text>
                                <xsl:text>{</xsl:text><xsl:text>&#xa;</xsl:text>
        <xsl:value-of select="code"/>
        <xsl:text>&#xa;</xsl:text>
                                <xsl:text>}</xsl:text>
                                <xsl:text>&#xa;</xsl:text>
                            </xsl:for-each>
                        </xsl:for-each>
        </xsl:template>
</xsl:stylesheet>
```

Listagem B.1: xml2systemccpp.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method='text'/>
  <xsl:template match="/">
    <xsl:for-each select="Model/block">
      <xsl:text>//This file is automatically generated</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>//by SysML2SystemC Project</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>#include "systemc.h"</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>SC_MODULE(</xsl:text>
      <xsl:value-of select="name"/>
      <xsl:text>)</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>{</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:for-each select="port">
        <xsl:if test="portdirection='none'">
          <xsl:text>sc signal inout if&lt; </xsl:text>
        </xsl:if>
        <xsl:if test="portdirection = 'in'">
          <xsl:text>sc_fifo_in&lt; </xsl:text>
        </xsl:if>
        <xsl:if test="portdirection = 'out'">
          <xsl:text>sc_fifo_out&lt; </xsl:text>
        </xsl:if>
        <xsl:if test="portdirection = 'inout'">
          <xsl:text>sc_fifo_inout&lt; </xsl:text>
        </xsl:if>
        <xsl:value-of select="type"/>
        <xsl:text> &gt; </xsl:text>
        <xsl:value-of select="name"/>
        <xsl:text>;</xsl:text>
        <xsl:text>&#xa;</xsl:text>
      </xsl:for-each>
      <xsl:text>&#xa;</xsl:text>
      <xsl:for-each select="operation">
        <xsl:value-of select="returntype"/>
        <xsl:text> </xsl:text>
        <xsl:value-of select="name"/>
```

```
        <xsl:text>( </xsl:text>
        <xsl:value-of select="parameterdone"/>
        <xsl:text>);</xsl:text>
        <xsl:text>&#xa;</xsl:text>
      </xsl:for-each>
      <xsl:text>SC_CTOR(</xsl:text>
      <xsl:value-of select="name"/>
      <xsl:text>)</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:text>{</xsl:text>
      <xsl:text>&#xa;</xsl:text>
      <xsl:for-each select="operation">
                <xsl:text>SC_THREAD(</xsl:text>
        <xsl:value-of select="name"/>
        <xsl:text>);</xsl:text>
        <xsl:text>&#xa;</xsl:text>
                  <xsl:value-of select="operationinitialization"/>
                  <xsl:text>&#xa;</xsl:text>
                <xsl:value-of select="onclassinitialization"/>
                <xsl:text>&#xa;</xsl:text>
          </xsl:for-each>
      <xsl:text>}</xsl:text>
      <xsl:text>&#xa;</xsl:text>
                <xsl:value-of select="extrafinalcode"/>
                <xsl:text>&#xa;</xsl:text>

                <xsl:text>};</xsl:text>
      <xsl:text>&#xa;</xsl:text>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Listagem B.2: xml2systemchdata.xslt

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method='text'/>
        <xsl:template match="/">
                <xsl:for-each select="Model/block">
                        <xsl:text>#include "</xsl:text>
                        <xsl:value-of select="name"/>
                        <xsl:text>.h"</xsl:text>
                        <xsl:text>&#xa;</xsl:text>
                </xsl:for-each>
                <xsl:text>#include "systemc.h"</xsl:text>
                <xsl:text>&#xa;</xsl:text>
                <xsl:text>&#xa;</xsl:text>
                <xsl:text>int sc_main(int argc, char* argv[])</xsl:text>
                <xsl:text>&#xa;</xsl:text>
                <xsl:text>{</xsl:text><xsl:text>&#xa;</xsl:text>
                <xsl:for-each select="Model/connector">
                        <xsl:text>sc_fifo&lt;</xsl:text>
                        <xsl:value-of select="typeenda"/><xsl:text>&gt; </xsl:text>
                        <xsl:value-of select="nameblockenda"/>
                        <xsl:text>_</xsl:text><xsl:value-of select="nameblockendb"/>
                        <xsl:text>_</xsl:text><xsl:value-of select="nameportenda"/>
```

```
                <xsl:text>("</xsl:text><xsl:value-of select="nameblockenda"/>
                <xsl:text>_</xsl:text><xsl:value-of select="nameblockendb"/>
            <xsl:text>_</xsl:text><xsl:value-of select="nameportenda"/>
            <xsl:text>");</xsl:text>
                <xsl:text>&#xa;</xsl:text>
            </xsl:for-each>
            <xsl:text>&#xa;</xsl:text>
            <xsl:for-each select="Model/block">
                <xsl:value-of select="name"/>
                <xsl:text> my_</xsl:text><xsl:value-of select="name"/>
                <xsl:text>("my_</xsl:text><xsl:value-of select="name"/>
                <xsl:text>");</xsl:text>
                <xsl:text>&#xa;</xsl:text>
            </xsl:for-each>
            <xsl:text>&#xa;</xsl:text>
            <xsl:for-each select="Model/connector">
                <xsl:text>my_</xsl:text>
                <xsl:value-of select="nameblockenda"/>
                <xsl:text>.</xsl:text>
                <xsl:value-of select="nameportenda"/>
                <xsl:text>(</xsl:text>
                <xsl:value-of select="nameblockenda"/>
                <xsl:text>_</xsl:text>
                <xsl:value-of select="nameblockendb"/>
                <xsl:text>_</xsl:text>
                <xsl:value-of select="nameportenda"/>
                <xsl:text>); </xsl:text>

                <xsl:text>my_</xsl:text>
                <xsl:value-of select="nameblockendb"/>
                <xsl:text>.</xsl:text>
                <xsl:value-of select="nameportendb"/>
                <xsl:text>(</xsl:text>
                <xsl:value-of select="nameblockenda"/>
                <xsl:text>_</xsl:text>
                <xsl:value-of select="nameblockendb"/>
                <xsl:text>_</xsl:text>
                <xsl:value-of select="nameportenda"/>
                <xsl:text>); </xsl:text>
                <xsl:text>&#xa;</xsl:text>
            </xsl:for-each>
            <xsl:text>&#xa;</xsl:text>
            <xsl:text>sc_start();</xsl:text><xsl:text>&#xa;</xsl:text>
            <xsl:text>return 0;</xsl:text><xsl:text>&#xa;</xsl:text>
            <xsl:text>}</xsl:text><xsl:text>&#xa;</xsl:text>
        </xsl:template>
</xsl:stylesheet>
```

Listagem B.3: xml2systemcmain.xslt

B.1 Tutorial XSLT

XSLT significa *EXtensible Stylesheet Language Transformation*, é uma linguagem para transformação de arquivos XML em um outro arquivo texto - de formato específico (como um outro arquivo XML, ou HTML) ou um arquivo texto qualquer, como no caso deste trabalho em que arquivos SytemC são criados.

Um arquivo de estilo XSL consiste em um ou mais conjuntos de regras que podem ser chamados de *templates*. Um *template* contém regras que serão aplicadas quando um nó XML específico for encontrado. No processo de transformação, XSLT define partes do documento fonte que devem comparadas aos *templates* criados. Quando uma das comparações é atendida, XSLT irá transformar a parte compatível do documento fonte no texto do documento resultante.

Para facilitar o entendimento de como é feita a transformação será utilizado um exemplo simples com o arquivo XML da Listagem B.4 e o arquivo XSLT da Listagem B.5.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>CD1</title>
    <artist>Artista1</artist>
    <country>Pais1</country>
  </cd>
  <cd>
    <title>CD2</title>
    <artist>Artista2</artist>
    <country>Pais2</country>
  </cd>
  <cd>
    <title>CD3</title>
    <artist>Artista3</artist>
    <country>Pais3</country>
  </cd>
  <cd>
    <title>CD4</title>
    <artist>Artista4</artist>
    <country>Pais4</country>
  </cd>
</catalog>
```

Listagem B.4: XML Exemplo

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3 <xsl:template match="/">
4
5   <xsl:text>Artistas</xsl:text>
6   <xsl:text>&#xa;</xsl:text>
7   <xsl:for-each select="catalog/cd">
```

```
8       <xsl:value-of select="artist"/>
9       <xsl:text>&#xa;</xsl:text>
10    </xsl:for-each>
11
12    <xsl:text>&#xa;</xsl:text>
13
14    <xsl:text>Paises</xsl:text>
15    <xsl:text>&#xa;</xsl:text>
16    <xsl:for-each select="catalog/cd">
17       <xsl:value-of select="country"/>
18       <xsl:text>&#xa;</xsl:text>
19    </xsl:for-each>
20
21  </xsl:template>
22  </xsl:stylesheet>
```

Listagem B.5: XSLT Exemplo

O arquivo XML exemplo tem um catálogo básico de CDs, cada um com título, artista e país. No arquivo de estilo XSLT observamos as *tags* principais e que também são utilizadas neste trabalho:

- **Linha 1:** Como o arquivo XSLT é um documento XML, sempre deve começar com a declaração XML $<?xml version = "1.0" encoding = "ISO-8859-1"?>$.
- **Linhas 2 e 22:** Nestas linhas estão as *tags* de abertura e fechamento $< xsl : stylesheet >$, que definem que este documento é um arquivo de estilo XSLT.
- **Linhas 3 e 21:** O elemento $< xsl : template >$ define um *template* e o atributo $match = "/"$ associa este *template* com o nó raiz do documento XML a ser transformado.

Essas três configurações iniciais são padrão para todos os arquivos de estilo XSLT. Os comandos vistos nas demais linhas são utilizados para a transformação do arquivo propriamente dito. Os comandos básicos utilizados nos arquivos XSLT deste trabalho são:

- $< xsl : text >$**:** Este elemento é utilizado para escrita de texto literal no arquivo de saída. Exemplos da utilização dele podem ser vistos na linha 5 - onde foi escrito o texto **Artistas**; linhas 6, 9, 12, 15 e 18 - onde foram incluídos os códigos relativos a quebra de linha; e na linha 14 - onde foi escrito o texto **Países**.
- $< xsl : for - each >$**:** Este elemento é utilizado para percorrer os nós de um determinado conjunto nós, especificado na expressão do atributo **select**.

- $< xsl : value - of >$**:** Este elemento é utilizado para extrair o valor de um nó do XML de entrada e incluí-lo no arquivo de saída da transformação. No atributo **select** é especificada a expressão para chegar no nó escolhido.

- $< xsl : if >$**:** Este elemento é utilizado para executar um teste condicional em algum conteúdo do arquivo XML de entrada. No atributo **test** é incluída a expressão do teste a ser realizado.

Por ser um padrão aberto, existem vários sites e documentos com tutoriais e exemplos de utilização. Entre alguns, pode ser utilizado o Tutorial XSLT da W3 Schools [Tutorial, 2011], com diversos exemplos e mais detalhes da transformação XSLT.

Na Listagem B.6 pode ser visualizado o arquivo texto de saída da transformação realizada sobre o arquivo XML de entrada.

```
Artistas
Artista1
Artista2
Artista3
Artista4

Paises
Pais1
Pais2
Pais3
Pais4
```

Listagem B.6: Resultado da transformação do XML de entrada e com o XSLT

Anexo A

Arquivos de saída do Pipeline

Abaixo podem ser vistas as Listagens de todos os arquivos gerados automaticamente para a Especificação Executável do Pipeline.

```
#include "numgen.h"
#include "stage1.h"
#include "stage2.h"
#include "stage3.h"
#include "display.h"
#include "systemc.h"
#include "time.h"
int sc_main(int argc, char* argv[])
{
        sc_fifo<double> numgen_stage1_out1("numgen_stage1_out1");
        sc_fifo<double> numgen_stage1_out2("numgen_stage1_out2");
        sc_fifo<double> stage1_stage2_sum("stage1_stage2_sum");
        sc_fifo<double> stage1_stage2_diff("stage1_stage2_diff");
        sc_fifo<double> stage2_stage3_prod("stage2_stage3_prod");
        sc_fifo<double> stage2_stage3_quot("stage2_stage3_quot");
        sc_fifo<double> stage3_display_powr("stage3_display_powr");
        numgen my_numgen("my_numgen");
        stage1 my_stage1("my_stage1");
        stage2 my_stage2("my_stage2");
        stage3 my_stage3("my_stage3");
        display my_display("my_display");
        my_numgen.out1(numgen_stage1_out1);
        my_stage1.in1(numgen_stage1_out1);
        my_numgen.out2(numgen_stage1_out2);
        my_stage1.in2(numgen_stage1_out2);
        my_stage1.sum(stage1_stage2_sum);
        my_stage2.sum(stage1_stage2_sum);
        my_stage1.diff(stage1_stage2_diff);
        my_stage2.diff(stage1_stage2_diff);
        my_stage2.prod(stage2_stage3_prod);
        my_stage3.prod(stage2_stage3_prod);
        my_stage2.quot(stage2_stage3_quot);
        my_stage3.quot(stage2_stage3_quot);
        my_stage3.powr(stage3_display_powr);
        my_display.in(stage3_display_powr);
```

```
        sc_start();
        return 0;
}
```

Listagem A.1: main.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(display)
{
        sc_fifo_in< double > in;

        void print_result( );
        SC_CTOR(display)
        {
                SC_THREAD(print_result);
                sensitive << in.data_written();
        }
};
```

Listagem A.2: Display.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "display.h"
void display::print_result( )
{

        while(true)
        {
                wait();
                cout << "Entering display::print_result" << endl;
                printf("Result = %f\n", in.read());
        }
}
```

Listagem A.3: Display.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(numgen)
{
        sc_fifo_out< double > out1;
        sc_fifo_out< double > out2;

        void generate( );
        SC_CTOR(numgen)
        {
                SC_THREAD(generate);
        }
};
```

Listagem A.4: NumGen.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "numgen.h"
void numgen::generate( )
{
        cout << "Entering numgen::generate" << endl;
        static double a = 134.56;
        static double b = 98.24;
        for(int i = 0; i < 46; i++)
        {
                a -= 1.5;
                b -= 2.8;
                out1.write(a);
                out2.write(b);
                wait(0,SC_NS);
        }
}
```

Listagem A.5: NumGen.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(stage1)
{
        sc_fifo_in< double > in1;
        sc_fifo_in< double > in2;
        sc_fifo_out< double > sum;
        sc_fifo_out< double > diff;

        void addsub( );
        SC_CTOR(stage1)
        {
                SC_THREAD(addsub);
                sensitive << in1.data_written();
                sensitive << in2.data_written();
        }
};
```

Listagem A.6: Stage1.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "stage1.h"
void stage1::addsub( )
{
        while(true)
        {
                wait();
```

```
                cout << "Entering stage1::addsub" << endl;
                double a;
                double b;

                a = in1.read();
                b = in2.read();
                sum.write(a+b);
                diff.write(a-b);
        }
}
```

Listagem A.7: Stage1.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(stage2)
{
        sc_fifo_in< double > sum;
        sc_fifo_in< double > diff;
        sc_fifo_out< double > prod;
        sc_fifo_out< double > quot;

        void multdiv( );
        SC_CTOR(stage2)
        {
                SC_THREAD(multdiv);
                sensitive << sum.data_written();
                sensitive << diff.data_written();
        }
};
```

Listagem A.8: Stage2.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "stage2.h"
void stage2::multdiv( )
{
        while(true)
        {
                wait();
                cout << "Entering stage2::multdiv" << endl;
                double a;
                double b;

                a = sum.read();
                b = diff.read();
                if( b == 0 ) b = 5.0;

                prod.write(a*b);
                quot.write(a/b);
        }
}
```

Listagem A.9: Stage2.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(stage3)
{
        sc_fifo_in< double > prod;
        sc_fifo_in< double > quot;
        sc_fifo_out< double > powr;

        void power( );
        SC_CTOR(stage3)
        {
                SC_THREAD(power);
                sensitive << prod.data_written();
                sensitive << quot.data_written();
        }
};
```

Listagem A.10: Stage3.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "stage3.h"
void stage3::power( )
{

        while(true)
        {
                wait();
                cout << "Entering stage3::power" << endl;
                double a;
                double b;
                double c;

                a = prod.read();
                b = quot.read();
                c = (a>0 && b>0)? pow(a, b) : 0.;
                powr.write(c);
        }
}
```

Listagem A.11: Stage3.cpp

```
<Model>
<block>
<id>U7a10c139-0218-4c8c-86c6-3da31628c851</id>
<name>numgen</name>
<package>
<id></id>
<name></name>
```

```
</package>
<port>
<id>U769a8ada-a8ae-4ab6-a4c5-0af3d6a0bf32</id>
<name>out1</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<port>
<id>U1ac14a9b-b1cd-495c-9cb9-c790b7f0c934</id>
<name>out2</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<connector>
<id>U6663e0bd-bd1b-493b-81cb-906b56925a54</id>
<name></name>
<idportenda>U769a8ada-a8ae-4ab6-a4c5-0af3d6a0bf32</idportenda>
<idportendb>U0a973046-5c9b-455b-ace3-b2e8b51c6e94</idportendb>
</connector>
<connector>
<id>U1fff9181-e8a7-409e-8323-930ec5a99c2f</id>
<name></name>
<idportenda>U1ac14a9b-b1cd-495c-9cb9-c790b7f0c934</idportenda>
<idportendb>U47bed89f-b1f9-4b9d-9e56-7f103cbced02</idportendb>
</connector>
<operation>
<id>Ua71f1a5d-e13f-4038-b1cc-5ed59f1d29b1</id>
<name>generate</name>
<code><![CDATA[cout << "Entering numgen::generate" << endl;
static double a = 134.56;
  static double b = 98.24;
  for(int i = 0; i < 61; i++)
  {
        a -= 1.5;
        b -= 2.8;
        out1.write(a);
        out2.write(b);
  }
]]></code>
<returntype>void</returntype>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>Uc3a9c4a0-cd44-471f-a7a7-2fec9b91d998</id>
<name>stage1</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U0a973046-5c9b-455b-ace3-b2e8b51c6e94</id>
```

```
<name>in1</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>U47bed89f-b1f9-4b9d-9e56-7f103cbced02</id>
<name>in2</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>U36647b8d-ca88-4a7f-93d9-8ff8f1b3e3a5</id>
<name>sum</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<port>
<id>U479323be-6097-409e-9c04-ea79c02fc48b</id>
<name>diff</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<connector>
<id>Ufd558e05-5079-4bf6-b700-96b52b468083</id>
<name></name>
<idportenda>U36647b8d-ca88-4a7f-93d9-8ff8f1b3e3a5</idportenda>
<idportendb>Uacb09fc2-40b2-41df-b3f0-769cf5f2e13e</idportendb>
</connector>
<connector>
<id>U975219b7-5bdf-4d7c-a651-cda57d00dacd</id>
<name></name>
<idportenda>U479323be-6097-409e-9c04-ea79c02fc48b</idportenda>
<idportendb>U63c4969c-5c57-41d1-9c7d-670a7e9f95d6</idportendb>
</connector>
<operation>
<id>Ubf8755d4-018d-4e4a-9e34-03c272d76e33</id>
<name>addsub</name>
<code><![CDATA[cout << "Entering stage1::addsub" << endl;
double a;
  double b;

  a = in1.read();
  b = in2.read();
  sum.write(a+b);
  diff.write(a-b);
]]></code>
<returntype>void</returntype>
<parameter>
<id>Ua4b7c125-3dcf-4f3a-a706-4e17386f5859</id>
<name>in1</name>
<type>bool</type>
</parameter>
<parameter>
<id>Uf1a1e334-fe7c-4c32-886d-c2a37c2d4b6e</id>
<name>in2</name>
```

```
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << in1.data_written();
sensitive << in2.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>Uc880a960-41be-4a94-89cf-0086dd58d301</id>
<name>stage2</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>Uacb09fc2-40b2-41df-b3f0-769cf5f2e13e</id>
<name>sum</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>U63c4969c-5c57-41d1-9c7d-670a7e9f95d6</id>
<name>diff</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>U4c39b7de-1c80-4b7e-abb0-009166639d38</id>
<name>prod</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<port>
<id>Ud01e2f7c-0df5-4572-8b04-3a071070c6fa</id>
<name>quot</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>double</type>
</port>
<connector>
<id>U0256fa25-0788-48a1-953b-176be1d3fe70</id>
<name></name>
<idportenda>U4c39b7de-1c80-4b7e-abb0-009166639d38</idportenda>
<idportendb>Ud6609f81-e54f-44ef-b7fd-41b1420adc18</idportendb>
</connector>
<connector>
<id>Uc520e1fd-28a2-4bb6-92e8-3de9c72cccd2</id>
<name></name>
<idportenda>Ud01e2f7c-0df5-4572-8b04-3a071070c6fa</idportenda>
<idportendb>U9a5cf5f4-dc3e-4a22-8e70-b9e9ab845fdc</idportendb>
</connector>
<operation>
<id>U439fc934-ee84-4eca-83f8-ed8f908abbf9</id>
```

```
<name>multdiv</name>
<code><![CDATA[cout << "Entering stage2::multdiv" << endl;
double a;
  double b;

  a = sum.read();
  b = diff.read();
  if( b == 0 )
      b = 5.0;

  prod.write(a*b);
  quot.write(a/b);
]]></code>
<returntype>void</returntype>
<parameter>
<id>U2f032987-d7ff-44fa-aaa9-6d2b5b73f894</id>
<name>sum</name>
<type>bool</type>
</parameter>
<parameter>
<id>U94ce79b0-3c7b-4719-8c45-887e6e47e5ac</id>
<name>diff</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << sum.data_written();
sensitive << diff.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U5ba2739c-ae69-41b5-bfba-c76725717e8e</id>
<name>stage3</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>Ud6609f81-e54f-44ef-b7fd-41b1420adc18</id>
<name>prod</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>U9a5cf5f4-dc3e-4a22-8e70-b9e9ab845fdc</id>
<name>quot</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<port>
<id>Ue83ed474-2c6f-4296-acde-5acc098ebbec</id>
<name>powr</name>
<privilege>protected</privilege>
```

```
<portdirection>out</portdirection><type>double</type>
</port>
<connector>
<id>Ubdb6e340-74d3-4335-8546-d1ab9b62be2c</id>
<name></name>
<idportenda>Ue83ed474-2c6f-4296-acde-5acc098ebbec</idportenda>
<idportendb>U72c065bf-1561-4135-b8ff-5da4faf481a6</idportendb>
</connector>
<operation>
<id>U7ed4e428-ede8-4a48-adf0-c073c91e416c</id>
<name>power</name>
<code><![CDATA[cout << "Entering stage3::power" << endl;
double a;
  double b;
  double c;

  a = prod.read();
  b = quot.read();
  c = (a>0 && b>0)? pow(a, b) : 0.;
  powr.write(c);
]]></code>
<returntype>void</returntype>
<parameter>
<id>U636863f3-4edb-4442-b87c-937135146e9f</id>
<name>prod</name>
<type>bool</type>
</parameter>
<parameter>
<id>U88e0f816-a190-4442-9142-bfb6de84d546</id>
<name>quot</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << prod.data_written();
sensitive << quot.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U66ddabf5-acad-4d87-bb12-3b2057bfae0a</id>
<name>display</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U72c065bf-1561-4135-b8ff-5da4faf481a6</id>
<name>in</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>double</type>
</port>
<operation>
<id>Ubde30b38-36e3-4407-b286-51fab7adcd8d</id>
```

```
<name>print_result</name>
<code><![CDATA[cout << "Entering display::print_result" << endl;
printf("Result = %f\n", in.read());]]></code>
<returntype>void</returntype>
<parameter>
<id>Ued2a5f28-c124-45fe-8286-91e342e630c4</id>
<name>in</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << in.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<connector>
<id>U6663e0bd-bd1b-493b-81cb-906b56925a54</id>
<name></name>
<idportenda>U769a8ada-a8ae-4ab6-a4c5-0af3d6a0bf32</idportenda>
<nameportenda>out1</nameportenda>
<nameblockenda>numgen</nameblockenda>
<typeenda>double</typeenda>
<idportendb>U0a973046-5c9b-455b-ace3-b2e8b51c6e94</idportendb>
<nameportendb>in1</nameportendb>
<nameblockendb>stage1</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>U1fff9181-e8a7-409e-8323-930ec5a99c2f</id>
<name></name>
<idportenda>U1ac14a9b-b1cd-495c-9cb9-c790b7f0c934</idportenda>
<nameportenda>out2</nameportenda>
<nameblockenda>numgen</nameblockenda>
<typeenda>double</typeenda>
<idportendb>U47bed89f-b1f9-4b9d-9e56-7f103cbced02</idportendb>
<nameportendb>in2</nameportendb>
<nameblockendb>stage1</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>Ufd558e05-5079-4bf6-b700-96b52b468083</id>
<name></name>
<idportenda>U36647b8d-ca88-4a7f-93d9-8ff8f1b3e3a5</idportenda>
<nameportenda>sum</nameportenda>
<nameblockenda>stage1</nameblockenda>
<typeenda>double</typeenda>
<idportendb>Uacb09fc2-40b2-41df-b3f0-769cf5f2e13e</idportendb>
<nameportendb>sum</nameportendb>
<nameblockendb>stage2</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>U975219b7-5bdf-4d7c-a651-cda57d00dacd</id>
<name></name>
```

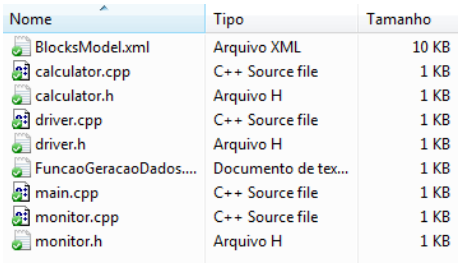
```
<idportenda>U479323be-6097-409e-9c04-ea79c02fc48b</idportenda>
<nameportenda>diff</nameportenda>
<nameblockenda>stage1</nameblockenda>
<typeenda>double</typeenda>
<idportendb>U63c4969c-5c57-41d1-9c7d-670a7e9f95d6</idportendb>
<nameportendb>diff</nameportendb>
<nameblockendb>stage2</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>U0256fa25-0788-48a1-953b-176be1d3fe70</id>
<name></name>
<idportenda>U4c39b7de-1c80-4b7e-abb0-009166639d38</idportenda>
<nameportenda>prod</nameportenda>
<nameblockenda>stage2</nameblockenda>
<typeenda>double</typeenda>
<idportendb>Ud6609f81-e54f-44ef-b7fd-41b1420adc18</idportendb>
<nameportendb>prod</nameportendb>
<nameblockendb>stage3</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>Uc520e1fd-28a2-4bb6-92e8-3de9c72cccd2</id>
<name></name>
<idportenda>Ud01e2f7c-0df5-4572-8b04-3a071070c6fa</idportenda>
<nameportenda>quot</nameportenda>
<nameblockenda>stage2</nameblockenda>
<typeenda>double</typeenda>
<idportendb>U9a5cf5f4-dc3e-4a22-8e70-b9e9ab845fdc</idportendb>
<nameportendb>quot</nameportendb>
<nameblockendb>stage3</nameblockendb>
<typeendb>double</typeendb>
</connector>
<connector>
<id>Ubdb6e340-74d3-4335-8546-d1ab9b62be2c</id>
<name></name>
<idportenda>Ue83ed474-2c6f-4296-acde-5acc098ebbec</idportenda>
<nameportenda>powr</nameportenda>
<nameblockenda>stage3</nameblockenda>
<typeenda>double</typeenda>
<idportendb>U72c065bf-1561-4135-b8ff-5da4faf481a6</idportendb>
<nameportendb>in</nameportendb>
<nameblockendb>display</nameblockendb>
<typeendb>double</typeendb>
</connector>
</Model>
```

Listagem A.12: BlocksModel.xml

Anexo B

Arquivos de saída da Calculadora

Para o sistema Calculator foram gerados diversos arquivos conforme pode ser visto na Figura B.1. Abaixo podem ser vistas as Listagens de todos os arquivos gerados automaticamente para a Especificação Executável da Calculadora.



Nome	Tipo	Tamanho
BlocksModel.xml	Arquivo XML	10 KB
calculator.cpp	C++ Source file	1 KB
calculator.h	Arquivo H	1 KB
driver.cpp	C++ Source file	1 KB
driver.h	Arquivo H	1 KB
FuncaoGeracaoDados....	Documento de tex...	1 KB
main.cpp	C++ Source file	1 KB
monitor.cpp	C++ Source file	1 KB
monitor.h	Arquivo H	1 KB

Figura B.1: Arquivos de saída gerados para o sistema Calculator

```
#include "calculator.h"
#include "driver.h"
#include "monitor.h"
#include "systemc.h"
int sc_main(int argc, char* argv[])
{
sc_fifo<int> driver_calculator_op("driver_calculator_op");
sc_fifo<int> driver_calculator_a("driver_calculator_a");
sc_fifo<int> driver_calculator_b("driver_calculator_b");
sc_fifo<int> calculator_monitor_c("calculator_monitor_c");
calculator my_calculator("my_calculator");
driver my_driver("my_driver");
monitor my_monitor("my_monitor");

my_driver.op(driver_calculator_op); my_calculator.op(driver_calculator_op);
my_driver.a(driver_calculator_a); my_calculator.a(driver_calculator_a);
my_driver.b(driver_calculator_b); my_calculator.b(driver_calculator_b);
my_calculator.c(calculator_monitor_c); my_monitor.c(calculator_monitor_c);

sc_start();
return 0;
```

```
}
```

Listagem B.1: main.cpp

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(calculator)
{
        sc_fifo_in< int > op;
        sc_fifo_in< int > a;
        sc_fifo_in< int > b;
        sc_fifo_out< int > c;

        void perform( );
        SC_CTOR(calculator)
        {
                SC_THREAD(perform);
                sensitive << a.data_written();
                sensitive << b.data_written();
                sensitive << op.data_written();
        }
    private:
        enum typeState {FirstState,S_OP,S_Add,S_Sub,S_Mul,S_Div,FinalState,S_Chk,S_ILG};
        typeState CurrentState,InitialState, NextState;
};
```

Listagem B.2: Calculator.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "calculator.h"
void calculator::perform( )
{
        while(true)
        {
                wait();
                cout << "Entering calculator::perform" << endl;
                int op_;
                op.nb_read(op_);
                int a_;
                a.nb_read(a_);
                int b_;
                b.nb_read(b_);
                int c_;
                c_ = NULL;
                InitialState = FirstState;
                CurrentState = InitialState;
                NextState = InitialState;
                while (CurrentState != FinalState) {
                        switch (CurrentState)
                        {
                        case FirstState:
                                NextState=S_OP;
```

```
                    break;

                case S_OP:
                    if(op_==1)NextState=S_Add;
                    else if(op_==2)NextState=S_Sub;
                    else if(op_==3)NextState=S_Mul;
                    else if(op_==4)NextState=S_Chk;
                    break;

                case S_Add:
                    c_ = a_+b_;
                    NextState=FinalState;
                    break;

                case S_Sub:
                    c_=a_-b_;
                    NextState=FinalState;
                    break;

                case S_Mul:
                    c_=a_*b_;
                    NextState=FinalState;
                    break;

                case S_Div:
                    c_=a_/b_;
                    NextState=FinalState;
                    break;

                case FinalState:
                    break;

                case S_Chk:
                    if(b_==0)NextState=S_ILG;
                    else NextState=S_Div;
                    break;

                case S_ILG:
                    c_=0;
                    NextState=FinalState;
                    break;

                default:
                    NextState = FinalState ;
                } //switch
                //update Current State

                CurrentState = NextState ;
            }
            c.write(c_);
        }
}
```

Listagem B.3: Calculator.cpp

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(driver)
{
        sc_fifo_out< int > op;
        sc_fifo_out< int > a;
        sc_fifo_out< int > b;

        void generate( );
        SC_CTOR(driver)
        {
                SC_THREAD(generate);
        }
};
```

Listagem B.4: Driver.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "driver.h"
void driver::generate( )
{
        cout << "Entering driver::generate" << endl;
        // Declaracao do arquivo de estimulo passado
        // pela linha de comando e abertura do mesmo
        ifstream entry_file;
        entry_file.open("data_op.txt", ifstream::in);

        // Erro na abertura do arquivo
        if(!entry_file)
                cerr << "**** ERROR: Nao e possivel abrir o arquivo de dados!" << endl;

        // Variaveis auxiliares para pegar os dados
        static int a1, a2, o;

        // Leitura do arquivo
        while(entry_file.good())
        {
                // Le e envia o valor de saida da tensao
                if(entry_file >> a1)
                {
                        entry_file >> a2;
                        entry_file >> o;

                        a.write(a1);
                        b.write(a2);
                        op.write(o);

                        wait(10,SC_NS);
                }
        }
}
```

Listagem B.5: Driver.cpp

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(monitor)
{
        sc_fifo_in< int > c;

        void show( );
        SC_CTOR(monitor)
        {
                SC_THREAD(show);
                sensitive << c.data_written();
        }
};
```

Listagem B.6: Monitor.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "monitor.h"
void monitor::show( )
{
        while(true)
        {
                wait();
                cout << "Entering monitor::show" << endl;
                cout << "c:" << c.read() <<"\n";
        }
}
```

Listagem B.7: Monitor.cpp

```
<Model>
<block>
<id>U4ee1e3c3-fa56-47ff-ba20-d0ce6dd73af4</id>
<name>calculator</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U23d5995a-2a5e-4ef9-915e-3d8bd8694667</id>
<name>op</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>int</type>
</port>
<port>
<id>U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</id>
<name>a</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>int</type>
</port>
<port>
```

```
<id>U73eb67ad-f2da-47a1-9690-bda2e141323b</id>
<name>b</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>int</type>
</port>
<port>
<id>Uff671bcf-67a1-419a-8983-b101d7c99f6b</id>
<name>c</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>int</type>
</port>
<connector>
<id>99-Uff671bcf-67a1-419a-8983-b101d7c99f6b</id>
<name>ConectorTestbenchc</name>
<idportenda>Uff671bcf-67a1-419a-8983-b101d7c99f6b</idportenda>
<idportendb>999-Uff671bcf-67a1-419a-8983-b101d7c99f6b</idportendb>
</connector>
<operation>
<id>Ubb77cc91-20d6-41eb-8dde-15d17d09eb00</id>
<name>perform</name>
<code><![CDATA[cout << "Entering calculator::perform" << endl;

int op_;
op.nb_read(op_);
int a_;
a.nb_read(a_);
int b_;
b.nb_read(b_);
int c_;
c_ = NULL;
InitialState = FirstState;
 CurrentState = InitialState;
 NextState = InitialState;
 while (CurrentState != FinalState) {
 switch (CurrentState)
{
 case FirstState:

NextState=S_OP;
break;


 case S_OP:

if(op_==1)NextState=S_Add;
 else if(op_==2)NextState=S_Sub;
 else if(op_==3)NextState=S_Mul;
 else if(op_==4)NextState=S_Chk;
break;


 case S_Add:
 c_ = a_+b_;
NextState=FinalState;
break;
```

```
 case S_Sub:
 c_=a_-b_;
NextState=FinalState;
break;


 case S_Mul:
 c_=a_*b_;
NextState=FinalState;
break;


 case S_Div:
 c_=a_/b_;
NextState=FinalState;
break;


 case FinalState:

break;


 case S_Chk:

if(b_==0)NextState=S_ILG;
 else NextState=S_Div;
break;


 case S_ILG:
 c_=0;
NextState=FinalState;
break;


default:
NextState = FinalState ;
} //switch
//update Current State

CurrentState = NextState ;}
 if(c_ != NULL) c.write(c_);]]></code>
<returntype>void</returntype>
<parameter>
<id>U27e34bb9-5c46-43c0-8a17-0952c71b1d14</id>
<name>a</name>
<type>bool</type>
</parameter>
<parameter>
<id>Uac6fa3c4-2023-4306-9b63-dbc7d5cfdae5</id>
<name>b</name>
<type>bool</type>
```

```
</parameter>
<parameter>
<id>U2ebd0f26-299b-4c84-b25f-0af4d94b4526</id>
<name>op</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << a.data_written();
sensitive << b.data_written();
sensitive << op.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode>
private:
enum typeState {FirstState,S_OP,S_Add,S_Sub,S_Mul,S_Div,FinalState,S_Chk,S_ILG};
typeState CurrentState,InitialState, NextState;</extrafinalcode>
</block>
<block>
<id>9999</id>
<name>driver</name>
<port>
<id>999-U23d5995a-2a5e-4ef9-915e-3d8bd8694667</id>
<name>op</name>
<portdirection>out</portdirection><type>int</type>
</port>
<port>
<id>999-U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</id>
<name>a</name>
<portdirection>out</portdirection><type>int</type>
</port>
<port>
<id>999-U73eb67ad-f2da-47a1-9690-bda2e141323b</id>
<name>b</name>
<portdirection>out</portdirection><type>int</type>
</port>
<connector>
<id>99-U23d5995a-2a5e-4ef9-915e-3d8bd8694667</id>
<name>ConectorTestbenchop</name>
<idportenda>999-U23d5995a-2a5e-4ef9-915e-3d8bd8694667</idportenda>
<idportendb>U23d5995a-2a5e-4ef9-915e-3d8bd8694667</idportendb>
</connector>
<connector>
<id>99-U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</id>
<name>ConectorTestbencha</name>
<idportenda>999-U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</idportenda>
<idportendb>U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</idportendb>
</connector>
<connector>
<id>99-U73eb67ad-f2da-47a1-9690-bda2e141323b</id>
<name>ConectorTestbenchb</name>
<idportenda>999-U73eb67ad-f2da-47a1-9690-bda2e141323b</idportenda>
<idportendb>U73eb67ad-f2da-47a1-9690-bda2e141323b</idportendb>
</connector>
<operation>
```

```
<id>99999</id>
<name>generate</name>
<code><![CDATA[cout << "Entering driver::generate" << endl;
]]></code>
<returntype>void</returntype>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>9998</id>
<name>monitor</name>
<port>
<id>999-Uff671bcf-67a1-419a-8983-b101d7c99f6b</id>
<name>c</name>
<portdirection>in</portdirection><type>int</type>
</port>
<operation>
<id>99998</id>
<name>show</name>
<code><![CDATA[cout << "Entering monitor::show" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>99998-999-Uff671bcf-67a1-419a-8983-b101d7c99f6b</id>
<name>c</name>
<type>int</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << c.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<connector>
<id>99-U23d5995a-2a5e-4ef9-915e-3d8bd8694667</id>
<name>ConectorTestbenchop</name>
<idportenda>999-U23d5995a-2a5e-4ef9-915e-3d8bd8694667</idportenda>
<nameportenda>op</nameportenda>
<nameblockenda>driver</nameblockenda>
<typeenda>int</typeenda>
<idportendb>U23d5995a-2a5e-4ef9-915e-3d8bd8694667</idportendb>
<nameportendb>op</nameportendb>
<nameblockendb>calculator</nameblockendb>
<typeendb>int</typeendb>
</connector>
<connector>
<id>99-U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</id>
<name>ConectorTestbencha</name>
<idportenda>999-U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</idportenda>
<nameportenda>a</nameportenda>
<nameblockenda>driver</nameblockenda>
```

```
<typeenda>int</typeenda>
<idportendb>U75b1e2c7-c3a8-419e-8993-f50b1cccb02e</idportendb>
<nameportendb>a</nameportendb>
<nameblockendb>calculator</nameblockendb>
<typeendb>int</typeendb>
</connector>
<connector>
<id>99-U73eb67ad-f2da-47a1-9690-bda2e141323b</id>
<name>ConectorTestbenchb</name>
<idportenda>999-U73eb67ad-f2da-47a1-9690-bda2e141323b</idportenda>
<nameportenda>b</nameportenda>
<nameblockenda>driver</nameblockenda>
<typeenda>int</typeenda>
<idportendb>U73eb67ad-f2da-47a1-9690-bda2e141323b</idportendb>
<nameportendb>b</nameportendb>
<nameblockendb>calculator</nameblockendb>
<typeendb>int</typeendb>
</connector>
<connector>
<id>99-Uff671bcf-67a1-419a-8983-b101d7c99f6b</id>
<name>ConectorTestbenchc</name>
<idportenda>Uff671bcf-67a1-419a-8983-b101d7c99f6b</idportenda>
<nameportenda>c</nameportenda>
<nameblockenda>calculator</nameblockenda>
<typeenda>int</typeenda>
<idportendb>999-Uff671bcf-67a1-419a-8983-b101d7c99f6b</idportendb>
<nameportendb>c</nameportendb>
<nameblockendb>monitor</nameblockendb>
<typeendb>int</typeendb>
</connector>
<statemachine>
<id>U59bdbc00-29ae-4b40-ae0a-50154607e9c3</id>
<name>StateMachine1</name>
<firststateid>Uc4373f3a-f41f-4805-bd4c-80b004f3c871</firststateid>
<state>
<id>Uc4373f3a-f41f-4805-bd4c-80b004f3c871</id>
<name>FirstState</name>
<action></action>
<statetransition>
<id>U22e7065c-37a7-4808-9dbf-f635ecc77748</id>
<condition></condition>
<nextStateId>U359a9bce-6e39-4ddd-a3d5-2be7433a3f15</nextStateId>
<nextStateName>S_OP</nextStateName>
</statetransition>
</state>
<state>
<id>U359a9bce-6e39-4ddd-a3d5-2be7433a3f15</id>
<name>S_OP</name>
<action></action>
<statetransition>
<id>U0f4de06e-d506-482d-af8c-ed964e33c462</id>
<condition>op==1</condition>
<nextStateId>U7a208f9e-af88-4b70-a400-4a9dfd5150fb</nextStateId>
<nextStateName>S_Add</nextStateName>
</statetransition>
```

```
<statetransition>
<id>Ua950035b-4f87-44dd-9270-9fddbe79ca67</id>
<condition>op==2</condition>
<nextStateId>U4efbf7b9-f0d9-4171-82cc-83f944678acc</nextStateId>
<nextStateName>S_Sub</nextStateName>
</statetransition>
<statetransition>
<id>U4f6bea80-62d7-4540-b655-66bee0d636df</id>
<condition>op==3</condition>
<nextStateId>U5189f842-9354-4a34-9700-cca57050629f</nextStateId>
<nextStateName>S_Mul</nextStateName>
</statetransition>
<statetransition>
<id>Uc1abfbdb-ef44-4daf-9268-d5a2ead0dfd8</id>
<condition>op==4</condition>
<nextStateId>U59c8a7d0-6a46-4214-91b8-19bb509ce60d</nextStateId>
<nextStateName>S_Chk</nextStateName>
</statetransition>
</state>
<state>
<id>U7a208f9e-af88-4b70-a400-4a9dfd5150fb</id>
<name>S_Add</name>
<action> c = a+b;</action>
<statetransition>
<id>U6a256b88-ce01-47e8-9c36-12c8885902fe</id>
<condition></condition>
<nextStateId>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</nextStateId>
<nextStateName>FinalState</nextStateName>
</statetransition>
</state>
<state>
<id>U4efbf7b9-f0d9-4171-82cc-83f944678acc</id>
<name>S_Sub</name>
<action> c=a-b;</action>
<statetransition>
<id>Uf3847646-ec93-41e1-a865-b98ba567fb48</id>
<condition></condition>
<nextStateId>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</nextStateId>
<nextStateName>FinalState</nextStateName>
</statetransition>
</state>
<state>
<id>U5189f842-9354-4a34-9700-cca57050629f</id>
<name>S_Mul</name>
<action> c=a*b;</action>
<statetransition>
<id>U84be9d8f-0b03-471c-8079-cfee9b51dd7c</id>
<condition></condition>
<nextStateId>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</nextStateId>
<nextStateName>FinalState</nextStateName>
</statetransition>
</state>
<state>
<id>Uce40d3f1-87bd-46a9-a09f-e13bc695ddd3</id>
<name>S_Div</name>
```

```
<action> c=a/b;</action>
<statetransition>
<id>Uc27d154a-5e62-484d-8ae8-5736f7628fa0</id>
<condition></condition>
<nextStateId>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</nextStateId>
<nextStateName>FinalState</nextStateName>
</statetransition>
</state>
<state>
<id>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</id>
<name>FinalState</name>
<action></action>
</state>
<state>
<id>U59c8a7d0-6a46-4214-91b8-19bb509ce60d</id>
<name>S_Chk</name>
<action></action>
<statetransition>
<id>Ud941fc05-2375-4f35-9721-8fdc23cd644c</id>
<condition></condition>
<nextStateId>Uce40d3f1-87bd-46a9-a09f-e13bc695ddd3</nextStateId>
<nextStateName>S_Div</nextStateName>
</statetransition>
<statetransition>
<id>Ua12638f9-cf2d-4aa2-9bb3-12ea7d671511</id>
<condition>b==0</condition>
<nextStateId>Ua8366125-d378-4042-93cd-c7c7b55d5802</nextStateId>
<nextStateName>S_ILG</nextStateName>
</statetransition>
</state>
<state>
<id>Ua8366125-d378-4042-93cd-c7c7b55d5802</id>
<name>S_ILG</name>
<action> c=0;</action>
<statetransition>
<id>U60848d3d-b601-411b-9e2d-9dff7b96068f</id>
<condition></condition>
<nextStateId>U49c52f53-4ec3-4a0d-94da-9f9bc00472f3</nextStateId>
<nextStateName>FinalState</nextStateName>
</statetransition>
</state>
</statemachine>
</Model>
```

Listagem B.8: BlocksModel.xml

Anexo C

Arquivos de saída da Máquina de Lavar

Abaixo podem ser vistas as Listagens de todos os arquivos gerados automaticamente para a Especificação Executável da Máquina de Lavar.

```
#include "Controller.h"
#include "WaterFeeder.h"
#include "WaterLevelSensor.h"
#include "Panel.h"
#include "Rotator.h"
#include "Drain.h"
#include "Timer.h"
#include "systemc.h"

int sc_main(int argc, char* argv[])
{
sc_fifo<int> Controller_WaterLevelSensor_senseWaterLevelM
("Controller_WaterLevelSensor_senseWaterLevelM");
sc_fifo<bool> Controller_WaterLevelSensor_reachWaterLevelS
("Controller_WaterLevelSensor_reachWaterLevelS");
sc_fifo<bool> Controller_Drain_openDrainM
("Controller_Drain_openDrainM");
sc_fifo<bool> Controller_Drain_closeDrainM
("Controller_Drain_closeDrainM");
sc_fifo<bool> Controller_Timer_timesUpS
("Controller_Timer_timesUpS");
sc_fifo<int> Controller_Timer_countDownM
("Controller_Timer_countDownM");
sc_fifo<bool> Controller_WaterFeeder_openWaterFeederM
("Controller_WaterFeeder_openWaterFeederM");
sc_fifo<bool> Controller_WaterFeeder_closeWaterFeederM
("Controller_WaterFeeder_closeWaterFeederM");
sc_fifo<bool> Panel_Controller_startWashM
("Panel_Controller_startWashM");
sc_fifo<bool> Rotator_Controller_startRotatorS
("Rotator_Controller_startRotatorS");
```

```
sc_fifo<bool> Rotator_Controller_stopRotatorS
("Rotator_Controller_stopRotatorS");

Controller my_Controller("my_Controller");
WaterFeeder my_WaterFeeder("my_WaterFeeder");
WaterLevelSensor my_WaterLevelSensor("my_WaterLevelSensor");
Panel my_Panel("my_Panel");
Rotator my_Rotator("my_Rotator");
Drain my_Drain("my_Drain");
Timer my_Timer("my_Timer");

my_Controller.senseWaterLevelM(Controller_WaterLevelSensor_senseWaterLevelM);
my_WaterLevelSensor.senseWaterLevelS(Controller_WaterLevelSensor_senseWaterLevelM);
my_Controller.reachWaterLevelS(Controller_WaterLevelSensor_reachWaterLevelS);
my_WaterLevelSensor.reachWaterLevelM(Controller_WaterLevelSensor_reachWaterLevelS);
my_Controller.openDrainM(Controller_Drain_openDrainM);
my_Drain.openDrainS(Controller_Drain_openDrainM);
my_Controller.closeDrainM(Controller_Drain_closeDrainM);
my_Drain.closeDrainS(Controller_Drain_closeDrainM);
my_Controller.timesUpS(Controller_Timer_timesUpS);
my_Timer.timesUpM(Controller_Timer_timesUpS);
my_Controller.countDownM(Controller_Timer_countDownM);
my_Timer.countDownS(Controller_Timer_countDownM);
my_Controller.openWaterFeederM(Controller_WaterFeeder_openWaterFeederM);
my_WaterFeeder.openWaterFeederS(Controller_WaterFeeder_openWaterFeederM);
my_Controller.closeWaterFeederM(Controller_WaterFeeder_closeWaterFeederM);
my_WaterFeeder.closeWaterFeederS(Controller_WaterFeeder_closeWaterFeederM);
my_Panel.startWashM(Panel_Controller_startWashM);
my_Controller.startWashS(Panel_Controller_startWashM);
my_Rotator.startRotatorS(Rotator_Controller_startRotatorS);
my_Controller.startRotatorM(Rotator_Controller_startRotatorS);
my_Rotator.stopRotatorS(Rotator_Controller_stopRotatorS);
my_Controller.stopRotatorM(Rotator_Controller_stopRotatorS);

sc_start();
return 0;
}
```

Listagem C.1: main.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Controller)
{
        sc_fifo_out< bool > openDrainM;
        sc_fifo_out< bool > stopRotatorM;
        sc_fifo_in< bool > startWashS;
        sc_fifo_out< bool > startRotatorM;
        sc_fifo_out< int > senseWaterLevelM;
        sc_fifo_in< bool > reachWaterLevelS;
        sc_fifo_out< bool > closeDrainM;
        sc_fifo_in< bool > timesUpS;
        sc_fifo_out< int > countDownM;
        sc_fifo_out< bool > openWaterFeederM;
        sc_fifo_out< bool > closeWaterFeederM;
```

```
        void startWash( );
        void reachWaterLevel( );
        void timesUp( );
        SC_CTOR(Controller)
        {
                SC_THREAD(startWash);
                sensitive << startWashS.data_written();
                SC_THREAD(reachWaterLevel);
                sensitive << reachWaterLevelS.data_written();
                SC_THREAD(timesUp);
                sensitive << timesUpS.data_written();
        }
};
```

Listagem C.2: Controller.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project

#include "systemc.h"
#include "Controller.h"
void Controller::startWash( )
{
        while(true)
        {
                wait();
                cout << "Entering Controller::startWash" << endl;
                openWaterFeederM.write(true);
                senseWaterLevelM.write(1);
        }
}
void Controller::reachWaterLevel( )
{
        while(true)
        {
                wait();
                cout << "Entering Controller::reachWaterLevel" << endl;
                bool t;
                if(timesUpS.nb_read(t))
                {
                        closeDrainM.write(true);
                }
                else
                {
                        closeWaterFeederM.write(true);
                        startRotatorM.write(true);
                        countDownM.write(1);
                }
        }
}
void Controller::timesUp( )
{
        while(true)
        {
                wait();
```

```
                cout << "Entering Controller::timesUp" << endl;
                stopRotatorM.write(true);
                openDrainM.write(true);
                senseWaterLevelM.write(2);
        }
}
```

Listagem C.3: Controller.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Drain)
{
        sc_fifo_in< bool > openDrainS;
        sc_fifo_in< bool > closeDrainS;

        void openDrain( );
        void closeDrain( );
        SC_CTOR(Drain)
        {
                SC_THREAD(openDrain);
                sensitive << openDrainS.data_written();
                SC_THREAD(closeDrain);
                sensitive << closeDrainS.data_written();
        }
};
```

Listagem C.4: Drain.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project

#include "systemc.h"
#include "Drain.h"
void Drain::openDrain( )
{
        while(true)
        {
                wait();
                cout << "Entering Drain::openDrain" << endl;
        }
}
void Drain::closeDrain( )
{
        while(true)
        {
                wait();
                cout << "Entering Drain::closeDrain" << endl;
        }
}
```

Listagem C.5: Drain.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Panel)
{
        sc_fifo_out< bool > startWashM;

        void washMode( );
        SC_CTOR(Panel)
        {
                SC_THREAD(washMode);
        }
};
```

Listagem C.6: Panel.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "Panel.h"
void Panel::washMode( )
{
        cout << "Entering Panel::washMode" << endl;
        startWashM.write(true);
}
```

Listagem C.7: Panel.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Rotator)
{
        sc_fifo_in< bool > startRotatorS;
        sc_fifo_in< bool > stopRotatorS;

        void startRotator( );
        void stopRotator( );
        SC_CTOR(Rotator)
        {
                SC_THREAD(startRotator);
                 sensitive << startRotatorS.data_written();
                SC_THREAD(stopRotator);
                 sensitive << stopRotatorS.data_written();
        }
};
```

Listagem C.8: Rotator.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "Rotator.h"
void Rotator::startRotator( )
```

```
{
        while(true)
        {
                wait();
                cout << "Entering Rotator::startRotator" << endl;
        }
}
void Rotator::stopRotator( )
{
        while(true)
        {
                wait();
                cout << "Entering Rotator::stopRotator" << endl;
        }
}
```

Listagem C.9: Rotator.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(Timer)
{
        sc_fifo_in< int > countDownS;
        sc_fifo_out< bool > timesUpM;

        void countDown( );
        SC_CTOR(Timer)
        {
                SC_THREAD(countDown);
                sensitive << countDownS.data_written();
        }
};
```

Listagem C.10: Timer.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "Timer.h"
void Timer::countDown( )
{
        while(true)
        {
                wait();
                cout << "Entering Timer::countDown" << endl;
                timesUpM.write(true);
        }
}
```

Listagem C.11: Timer.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
```

```
SC_MODULE(WaterFeeder)
{
        sc_fifo_in< bool > openWaterFeederS;
        sc_fifo_in< bool > closeWaterFeederS;

        void openWaterFeeder( );
        void closeWaterFeeder( );
        SC_CTOR(WaterFeeder)
        {
                SC_THREAD(openWaterFeeder);
                sensitive << openWaterFeederS.data_written();
                SC_THREAD(closeWaterFeeder);
                sensitive << closeWaterFeederS.data_written();
        }
};
```

Listagem C.12: WaterFeeder.h

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "WaterFeeder.h"
void WaterFeeder::openWaterFeeder( )
{
        while(true)
        {
                wait();
                cout << "Entering WaterFeeder::openWaterFeeder" << endl;
        }
}
void WaterFeeder::closeWaterFeeder( )
{
        while(true)
        {
                wait();
                cout << "Entering WaterFeeder::closeWaterFeeder" << endl;
        }
}
```

Listagem C.13: WaterFeeder.cpp

```
ï»¿//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
SC_MODULE(WaterLevelSensor)
{
        sc_fifo_in< int > senseWaterLevelS;
        sc_fifo_out< bool > reachWaterLevelM;

        void senseWaterLevel( );
        SC_CTOR(WaterLevelSensor)
        {
                SC_THREAD(senseWaterLevel);
                sensitive << senseWaterLevelS.data_written();
        }
```

```
};
```

Listagem C.14: WaterLevelSensor.h

```
//This file is automatically generated
//by SysML2SystemC Project
#include "systemc.h"
#include "WaterLevelSensor.h"
void WaterLevelSensor::senseWaterLevel( )
{
        while(true)
        {
                wait();
                cout << "Entering WaterLevelSensor::senseWaterLevel" << endl;
                reachWaterLevelM.write(true);
        }
}
```

Listagem C.15: WaterLevelSensor.cpp

```
<Model>
<block>
<id>U08bb5613-2612-4490-ba15-3e81c46166a3</id>
<name>Controller</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>Ufd21d4d9-ed28-4638-b809-9e20f3806746</id>
<name>openDrainM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<port>
<id>U667db552-d62e-4af4-8268-8f18ca278692</id>
<name>stopRotatorM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<port>
<id>Ue5462c4c-0355-43b0-bccd-632caea53047</id>
<name>startWashS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>U9275a130-2aad-476a-9173-14fbfea8b00b</id>
<name>startRotatorM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<port>
<id>Ud8edd603-050f-454a-b6e2-404f4f140731</id>
<name>senseWaterLevelM</name>
```

```
<privilege>protected</privilege>
<portdirection>out</portdirection><type>int</type>
</port>
<port>
<id>U38471786-ab9a-4df6-b3a2-ad4ac0a623df</id>
<name>reachWaterLevelS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>Ueff08daa-56ee-4b1e-a1b2-7fe9598595d8</id>
<name>closeDrainM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<port>
<id>U3ae6b538-2fdb-4502-bf0b-22f3d68b853a</id>
<name>timesUpS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>Ufc831d1f-cb90-44ca-b651-bef5c58806cb</id>
<name>countDownM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>int</type>
</port>
<port>
<id>Ua3af25ea-b543-4e12-84c8-16960bdacc20</id>
<name>openWaterFeederM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<port>
<id>Uf726f353-4eb9-4f14-ba43-dc1e4741dad9</id>
<name>closeWaterFeederM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<connector>
<id>U999404a0-a498-4be2-9eee-b6138d62042c</id>
<name></name>
<idportenda>Ud8edd603-050f-454a-b6e2-404f4f140731</idportenda>
<idportendb>U71ce35ac-6a15-40fc-b99e-061989015260</idportendb>
</connector>
<connector>
<id>U8d01e9a0-d054-4d06-ad8c-14991a966a56</id>
<name></name>
<idportenda>U38471786-ab9a-4df6-b3a2-ad4ac0a623df</idportenda>
<idportendb>U2eb85f7f-70e6-4af4-92c5-47354cdd38ec</idportendb>
</connector>
<connector>
<id>U34ae9d1a-e97f-4a41-b380-12a9cef7b5bd</id>
<name></name>
<idportenda>Ufd21d4d9-ed28-4638-b809-9e20f3806746</idportenda>
```

```
<idportendb>U9d8a0949-8531-44f6-a172-7903a13a649c</idportendb>
</connector>
<connector>
<id>U0f583329-d3d2-4270-84e5-7693f92ac3c8</id>
<name></name>
<idportenda>Ueff08daa-56ee-4b1e-a1b2-7fe9598595d8</idportenda>
<idportendb>Ue1e70112-3722-436d-8933-df1413096a36</idportendb>
</connector>
<connector>
<id>Ud801d238-ce5a-4270-b17b-5f99512a5bc5</id>
<name></name>
<idportenda>U3ae6b538-2fdb-4502-bf0b-22f3d68b853a</idportenda>
<idportendb>Ufaf109de-1d63-49e4-b5d0-3dff3b5172b9</idportendb>
</connector>
<connector>
<id>U0452493e-648e-4bfe-a313-5f445b61aebf</id>
<name></name>
<idportenda>Ufc831d1f-cb90-44ca-b651-bef5c58806cb</idportenda>
<idportendb>Ufa7b923c-9f2d-4b33-9aea-80ec37460fdc</idportendb>
</connector>
<connector>
<id>U08f0ebea-e82e-4829-b0d8-b4f8eb4ab7e7</id>
<name></name>
<idportenda>Ua3af25ea-b543-4e12-84c8-16960bdacc20</idportenda>
<idportendb>U26e4180e-435c-4189-ba6d-d682296592ae</idportendb>
</connector>
<connector>
<id>U83695798-78e5-4d35-9877-3b619f7821ca</id>
<name></name>
<idportenda>Uf726f353-4eb9-4f14-ba43-dc1e4741dad9</idportenda>
<idportendb>Ua2d79149-bc0e-461b-9f84-fadd50e6cd93</idportendb>
</connector>
<operation>
<id>U008c9ca4-b07f-4667-8d2a-0114c4fee333</id>
<name>startWash</name>
<code><![CDATA[cout << "Entering Controller::startWash" << endl;
openWaterFeederM.write(true);
senseWaterLevelM.write(1);]]></code>
<returntype>void</returntype>
<parameter>
<id>U0536e205-ac75-4ffe-9e6a-cf9fdbe88e71</id>
<name>startWashS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << startWashS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<operation>
<id>Udefa143a-9b2b-474e-853d-6cca4dc237db</id>
<name>reachWaterLevel</name>
<code><![CDATA[cout << "Entering Controller::reachWaterLevel" << endl;
bool t;
if(timesUpS.nb_read(t))
```

```
{
        closeDrainM.write(true);
}
else
{
closeWaterFeederM.write(true);
startRotatorM.write(true);
countDownM.write(1);]]></code>
<returntype>void</returntype>
<parameter>
<id>U2fb4a2d8-ae34-4c73-8334-06e5ec9e41f2</id>
<name>reachWaterLevelS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << reachWaterLevelS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<operation>
<id>U190a7ebb-bc8a-405d-a801-431c98f690ce</id>
<name>timesUp</name>
<code><![CDATA[cout << "Entering Controller::timesUp" << endl;
stopRotatorM.write(true);
openDrainM.write(true);
senseWaterLevelM.write(2);]]></code>
<returntype>void</returntype>
<parameter>
<id>U34f7dcdc-ee79-4951-84d6-2d34b1a7c782</id>
<name>timesUpS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << timesUpS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</id>
<name>WaterFeeder</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U26e4180e-435c-4189-ba6d-d682296592ae</id>
<name>openWaterFeederS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>Ua2d79149-bc0e-461b-9f84-fadd50e6cd93</id>
<name>closeWaterFeederS</name>
```

```
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<operation>
<id>Ucfe64e0c-c6aa-441e-afa4-20ad52e2c2c6</id>
<name>openWaterFeeder</name>
<code><![CDATA[cout << "Entering WaterFeeder::openWaterFeeder" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>U14e3c660-a203-457c-a906-1892d3ef55ef</id>
<name>openWaterFeederS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << openWaterFeederS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<operation>
<id>U1afbaa9f-9c3f-43f1-ae80-298229eb955f</id>
<name>closeWaterFeeder</name>
<code><![CDATA[cout << "Entering WaterFeeder::closeWaterFeeder" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>U8b7e642e-9948-48fa-b585-01255c9dbe51</id>
<name>closeWaterFeederS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << closeWaterFeederS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U11575726-b867-42b6-bc52-3a381e573049</id>
<name>WaterLevelSensor</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U71ce35ac-6a15-40fc-b99e-061989015260</id>
<name>senseWaterLevelS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>int</type>
</port>
<port>
<id>U2eb85f7f-70e6-4af4-92c5-47354cdd38ec</id>
<name>reachWaterLevelM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
```

```
</port>
<operation>
<id>U2290bd01-2226-414c-9553-200712271cbd</id>
<name>senseWaterLevel</name>
<code><![CDATA[cout << "Entering WaterLevelSensor::senseWaterLevel" << endl;
reachWaterLevelM.write(true);]]></code>
<returntype>void</returntype>
<parameter>
<id>Uf9220eaf-4d9c-4213-bc39-ce3844a18a62</id>
<name>senseWaterLevelS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << senseWaterLevelS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>Ubbf24451-158e-42b1-a6f2-2556e108a88a</id>
<name>Panel</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U8f490ca6-34b0-4faf-96e4-ec6eeb79b0df</id>
<name>startWashM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<connector>
<id>Ud84f578e-bdf7-4cad-8a6a-f3d53a00639b</id>
<name></name>
<idportenda>U8f490ca6-34b0-4faf-96e4-ec6eeb79b0df</idportenda>
<idportendb>Ue5462c4c-0355-43b0-bccd-632caea53047</idportendb>
</connector>
<operation>
<id>Uc4cf2d18-58c0-4347-a33b-413abf85348e</id>
<name>washMode</name>
<code><![CDATA[cout << "Entering Panel::washMode" << endl;
startWashM.write(true);]]></code>
<returntype>void</returntype>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U14e1466e-f316-4387-849e-1a753c987c39</id>
<name>Rotator</name>
<package>
<id></id>
```

```
<name></name>
</package>
<port>
<id>U8889cff3-8aea-455b-bf2c-d1370736b885</id>
<name>startRotatorS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>U18090c4e-d879-4f96-b739-208db198047f</id>
<name>stopRotatorS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<connector>
<id>U43864a5f-5220-4cfd-b04e-1e81773e67fe</id>
<name></name>
<idportenda>U8889cff3-8aea-455b-bf2c-d1370736b885</idportenda>
<idportendb>U9275a130-2aad-476a-9173-14fbfea8b00b</idportendb>
</connector>
<connector>
<id>Ub00065aa-d4c9-4a3d-b26a-7cecfa726b60</id>
<name></name>
<idportenda>U18090c4e-d879-4f96-b739-208db198047f</idportenda>
<idportendb>U667db552-d62e-4af4-8268-8f18ca278692</idportendb>
</connector>
<operation>
<id>U747280e4-c5d8-4137-bca2-f4933304ae22</id>
<name>startRotator</name>
<code><![CDATA[cout << "Entering Rotator::startRotator" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>Uf9855f74-4979-462e-b0b1-483217fdb55a</id>
<name>startRotatorS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << startRotatorS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<operation>
<id>U81870389-f40c-4cbb-99a5-776347cebe07</id>
<name>stopRotator</name>
<code><![CDATA[cout << "Entering Rotator::stopRotator" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>Ub31f02b9-38e7-4227-a312-e873453672f7</id>
<name>stopRotatorS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
```

```
sensitive << stopRotatorS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U7c74f7b4-32da-4e85-9498-df94271d6569</id>
<name>Drain</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>U9d8a0949-8531-44f6-a172-7903a13a649c</id>
<name>openDrainS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<port>
<id>Ue1e70112-3722-436d-8933-df1413096a36</id>
<name>closeDrainS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>bool</type>
</port>
<operation>
<id>U7694c54e-f23b-47ee-bcec-2410a4cd4d1d</id>
<name>openDrain</name>
<code><![CDATA[cout << "Entering Drain::openDrain" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>U0b6f5827-6766-43e6-b551-1edb8be0b36c</id>
<name>openDrainS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << openDrainS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<operation>
<id>U4d2f46ce-9efa-4863-858f-3be4514fef34</id>
<name>closeDrain</name>
<code><![CDATA[cout << "Entering Drain::closeDrain" << endl;
]]></code>
<returntype>void</returntype>
<parameter>
<id>U09c7dcfd-56e7-4428-8583-da407c243625</id>
<name>closeDrainS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << closeDrainS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
```

```
</operation>
<extrafinalcode></extrafinalcode>
</block>
<block>
<id>U2d1f4d48-67a9-401b-a461-84b3bddc0944</id>
<name>Timer</name>
<package>
<id></id>
<name></name>
</package>
<port>
<id>Ufa7b923c-9f2d-4b33-9aea-80ec37460fdc</id>
<name>countDownS</name>
<privilege>protected</privilege>
<portdirection>in</portdirection><type>int</type>
</port>
<port>
<id>Ufaf109de-1d63-49e4-b5d0-3dff3b5172b9</id>
<name>timesUpM</name>
<privilege>protected</privilege>
<portdirection>out</portdirection><type>bool</type>
</port>
<operation>
<id>Uef63be07-c57e-440c-8d44-488315337a5c</id>
<name>countDown</name>
<code><![CDATA[cout << "Entering Timer::countDown" << endl;
timesUpM.write(true);]]></code>
<returntype>void</returntype>
<parameter>
<id>U3286ad25-fa9b-4ee1-a531-abff3757b419</id>
<name>countDownS</name>
<type>bool</type>
</parameter>
<parameterdone></parameterdone>
<operationinitialization><![CDATA[ dont_initialize();
sensitive << countDownS.data_written();]]></operationinitialization>
<onclassinitialization></onclassinitialization>
</operation>
<extrafinalcode></extrafinalcode>
</block>
<connector>
<id>U999404a0-a498-4be2-9eee-b6138d62042c</id>
<name></name>
<idportenda>Ud8edd603-050f-454a-b6e2-404f4f140731</idportenda>
<nameportenda>senseWaterLevelM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>int</typeenda>
<idportendb>U71ce35ac-6a15-40fc-b99e-061989015260</idportendb>
<nameportendb>senseWaterLevelS</nameportendb>
<nameblockendb>WaterLevelSensor</nameblockendb>
<typeendb>int</typeendb>
</connector>
<connector>
<id>U8d01e9a0-d054-4d06-ad8c-14991a966a56</id>
<name></name>
```

```
<idportenda>U38471786-ab9a-4df6-b3a2-ad4ac0a623df</idportenda>
<nameportenda>reachWaterLevelS</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>U2eb85f7f-70e6-4af4-92c5-47354cdd38ec</idportendb>
<nameportendb>reachWaterLevelM</nameportendb>
<nameblockendb>WaterLevelSensor</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>U34ae9d1a-e97f-4a41-b380-12a9cef7b5bd</id>
<name></name>
<idportenda>Ufd21d4d9-ed28-4638-b809-9e20f3806746</idportenda>
<nameportenda>openDrainM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>U9d8a0949-8531-44f6-a172-7903a13a649c</idportendb>
<nameportendb>openDrainS</nameportendb>
<nameblockendb>Drain</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>U0f583329-d3d2-4270-84e5-7693f92ac3c8</id>
<name></name>
<idportenda>Ueff08daa-56ee-4b1e-a1b2-7fe9598595d8</idportenda>
<nameportenda>closeDrainM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>Ue1e70112-3722-436d-8933-df1413096a36</idportendb>
<nameportendb>closeDrainS</nameportendb>
<nameblockendb>Drain</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>Ud801d238-ce5a-4270-b17b-5f99512a5bc5</id>
<name></name>
<idportenda>U3ae6b538-2fdb-4502-bf0b-22f3d68b853a</idportenda>
<nameportenda>timesUpS</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>Ufaf109de-1d63-49e4-b5d0-3dff3b5172b9</idportendb>
<nameportendb>timesUpM</nameportendb>
<nameblockendb>Timer</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>U0452493e-648e-4bfe-a313-5f445b61aebf</id>
<name></name>
<idportenda>Ufc831d1f-cb90-44ca-b651-bef5c58806cb</idportenda>
<nameportenda>countDownM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>int</typeenda>
<idportendb>Ufa7b923c-9f2d-4b33-9aea-80ec37460fdc</idportendb>
<nameportendb>countDownS</nameportendb>
<nameblockendb>Timer</nameblockendb>
```

```
<typeendb>int</typeendb>
</connector>
<connector>
<id>U08f0ebea-e82e-4829-b0d8-b4f8eb4ab7e7</id>
<name></name>
<idportenda>Ua3af25ea-b543-4e12-84c8-16960bdacc20</idportenda>
<nameportenda>openWaterFeederM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>U26e4180e-435c-4189-ba6d-d682296592ae</idportendb>
<nameportendb>openWaterFeederS</nameportendb>
<nameblockendb>WaterFeeder</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>U83695798-78e5-4d35-9877-3b619f7821ca</id>
<name></name>
<idportenda>Uf726f353-4eb9-4f14-ba43-dc1e4741dad9</idportenda>
<nameportenda>closeWaterFeederM</nameportenda>
<nameblockenda>Controller</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>Ua2d79149-bc0e-461b-9f84-fadd50e6cd93</idportendb>
<nameportendb>closeWaterFeederS</nameportendb>
<nameblockendb>WaterFeeder</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>Ud84f578e-bdf7-4cad-8a6a-f3d53a00639b</id>
<name></name>
<idportenda>U8f490ca6-34b0-4faf-96e4-ec6eeb79b0df</idportenda>
<nameportenda>startWashM</nameportenda>
<nameblockenda>Panel</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>Ue5462c4c-0355-43b0-bccd-632caea53047</idportendb>
<nameportendb>startWashS</nameportendb>
<nameblockendb>Controller</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>U43864a5f-5220-4cfd-b04e-1e81773e67fe</id>
<name></name>
<idportenda>U8889cff3-8aea-455b-bf2c-d1370736b885</idportenda>
<nameportenda>startRotatorS</nameportenda>
<nameblockenda>Rotator</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>U9275a130-2aad-476a-9173-14fbfea8b00b</idportendb>
<nameportendb>startRotatorM</nameportendb>
<nameblockendb>Controller</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<connector>
<id>Ub00065aa-d4c9-4a3d-b26a-7cecfa726b60</id>
<name></name>
<idportenda>U18090c4e-d879-4f96-b739-208db198047f</idportenda>
<nameportenda>stopRotatorS</nameportenda>
```

```
<nameblockenda>Rotator</nameblockenda>
<typeenda>bool</typeenda>
<idportendb>U667db552-d62e-4af4-8268-8f18ca278692</idportendb>
<nameportendb>stopRotatorM</nameportendb>
<nameblockendb>Controller</nameblockendb>
<typeendb>bool</typeendb>
</connector>
<sequencediagram>
<id>U28042a55-af09-430f-807e-0d94df8a78d7</id>
<name>washermachine</name>
<lifeline>
<id>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</id>
<name>Lifeline1</name>
<idBlockRepresented>Ubbf24451-158e-42b1-a6f2-2556e108a88a</idBlockRepresented>
<blockname>Panel</blockname>
<messagessent>
<message>
<id>Ub698ff60-0b3b-44f9-9fcd-0e3352effc8a</id>
<name></name>
<order>1</order>
<idblock>Ubbf24451-158e-42b1-a6f2-2556e108a88a</idblock>
<nameblock>Panel</nameblock>
<idblockoperation>Uc4cf2d18-58c0-4347-a33b-413abf85348e</idblockoperation>
<nameoperation>washMode</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U70b8b21d-9b80-49f7-84fa-e12eee91e3cd</id>
<name></name>
<order>2</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U008c9ca4-b07f-4667-8d2a-0114c4fee333</idblockoperation>
<nameoperation>startWash</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagessent>
<messagesreceived>
<message>
<id>Ub698ff60-0b3b-44f9-9fcd-0e3352effc8a</id>
<name></name>
<order>1</order>
<idblock>Ubbf24451-158e-42b1-a6f2-2556e108a88a</idblock>
<nameblock>Panel</nameblock>
<idblockoperation>Uc4cf2d18-58c0-4347-a33b-413abf85348e</idblockoperation>
<nameoperation>washMode</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagesreceived>
```

```
</lifeline>
<lifeline>
<id>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</id>
<name>Lifeline2</name>
<idBlockRepresented>U08bb5613-2612-4490-ba15-3e81c46166a3</idBlockRepresented>
<blockname>Controller</blockname>
<messagessent>
<message>
<id>U39b133ad-057c-4553-af2b-3cf2fee8a5f0</id>
<name></name>
<order>3</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>Ucfe64e0c-c6aa-441e-afa4-20ad52e2c2c6</idblockoperation>
<nameoperation>openWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U34a7be9f-380c-4cf3-adca-f1f2eb921e63</id>
<name></name>
<order>4</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U3e629aec-bf9d-4168-8536-289a9eb19411</id>
<name></name>
<order>6</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>U1afbaa9f-9c3f-43f1-ae80-298229eb955f</idblockoperation>
<nameoperation>closeWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>U3d4b7f3a-f6a4-4f35-87c4-f74b768b67b2</id>
<name></name>
<order>7</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U747280e4-c5d8-4137-bca2-f4933304ae22</idblockoperation>
<nameoperation>startRotator</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
```

```
<message>
<id>Ubdfaea30-0c05-4bb3-bd20-dca77d9541c2</id>
<name></name>
<order>8</order>
<idblock>U2d1f4d48-67a9-401b-a461-84b3bddc0944</idblock>
<nameblock>Timer</nameblock>
<idblockoperation>Uef63be07-c57e-440c-8d44-488315337a5c</idblockoperation>
<nameoperation>countDown</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U157f010d-96df-4146-a3e5-07ee35b87ba7</id>
<name></name>
<order>10</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U81870389-f40c-4cbb-99a5-776347cebe07</idblockoperation>
<nameoperation>stopRotator</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>Uf70a1606-10b1-467f-86a5-00db5b5b6318</id>
<name></name>
<order>11</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U7694c54e-f23b-47ee-bcec-2410a4cd4d1d</idblockoperation>
<nameoperation>openDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>U9b559247-a13c-4cc2-99d8-040760b38521</id>
<name></name>
<order>12</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U20ed75a1-ac05-4169-bf7a-3461b4c718d5</id>
<name></name>
<order>14</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U4d2f46ce-9efa-4863-858f-3be4514fef34</idblockoperation>
```

```
<nameoperation>closeDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
</messagessent>
<messagesreceived>
<message>
<id>U70b8b21d-9b80-49f7-84fa-e12eee91e3cd</id>
<name></name>
<order>2</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U008c9ca4-b07f-4667-8d2a-0114c4fee333</idblockoperation>
<nameoperation>startWash</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>Uef4cbd08-4767-4e1f-ae5e-3d22c29cc7a7</id>
<name></name>
<order>5</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U127c5f63-0d06-4d2e-b673-853cd109f4dc</id>
<name></name>
<order>9</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U190a7ebb-bc8a-405d-a801-431c98f690ce</idblockoperation>
<nameoperation>timesUp</nameoperation>
<idlifelinesend>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U1a4caf37-9a04-4038-aeee-f8a7e932c91d</id>
<name></name>
<order>13</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
```

```
</messagesreceived>
</lifeline>
<lifeline>
<id>U41d4802b-87be-493d-867e-e8e022ec18c0</id>
<name>Lifeline3</name>
<idBlockRepresented>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idBlockRepresented>
<blockname>WaterFeeder</blockname>
<messagessent>
</messagessent>
<messagesreceived>
<message>
<id>U39b133ad-057c-4553-af2b-3cf2fee8a5f0</id>
<name></name>
<order>3</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>Ucfe64e0c-c6aa-441e-afa4-20ad52e2c2c6</idblockoperation>
<nameoperation>openWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U3e629aec-bf9d-4168-8536-289a9eb19411</id>
<name></name>
<order>6</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>U1afbaa9f-9c3f-43f1-ae80-298229eb955f</idblockoperation>
<nameoperation>closeWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
</messagesreceived>
</lifeline>
<lifeline>
<id>U8e4e21e9-9978-4942-817e-38b17de93fbd</id>
<name>Lifeline4</name>
<idBlockRepresented>U11575726-b867-42b6-bc52-3a381e573049</idBlockRepresented>
<blockname>WaterLevelSensor</blockname>
<messagessent>
<message>
<id>Uef4cbd08-4767-4e1f-ae5e-3d22c29cc7a7</id>
<name></name>
<order>5</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
```

```
<id>U1a4caf37-9a04-4038-aeee-f8a7e932c91d</id>
<name></name>
<order>13</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagessent>
<messagesreceived>
<message>
<id>U34a7be9f-380c-4cf3-adca-f1f2eb921e63</id>
<name></name>
<order>4</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U9b559247-a13c-4cc2-99d8-040760b38521</id>
<name></name>
<order>12</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagesreceived>
</lifeline>
<lifeline>
<id>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</id>
<name>Lifeline5</name>
<idBlockRepresented>U14e1466e-f316-4387-849e-1a753c987c39</idBlockRepresented>
<blockname>Rotator</blockname>
<messagessent>
</messagessent>
<messagesreceived>
<message>
<id>U3d4b7f3a-f6a4-4f35-87c4-f74b768b67b2</id>
<name></name>
<order>7</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U747280e4-c5d8-4137-bca2-f4933304ae22</idblockoperation>
<nameoperation>startRotator</nameoperation>
```

```
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U157f010d-96df-4146-a3e5-07ee35b87ba7</id>
<name></name>
<order>10</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U81870389-f40c-4cbb-99a5-776347cebe07</idblockoperation>
<nameoperation>stopRotator</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
</messagesreceived>
</lifeline>
<lifeline>
<id>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</id>
<name>Lifeline6</name>
<idBlockRepresented>U2d1f4d48-67a9-401b-a461-84b3bddc0944</idBlockRepresented>
<blockname>Timer</blockname>
<messagessent>
<message>
<id>U127c5f63-0d06-4d2e-b673-853cd109f4dc</id>
<name></name>
<order>9</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U190a7ebb-bc8a-405d-a801-431c98f690ce</idblockoperation>
<nameoperation>timesUp</nameoperation>
<idlifelinesend>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagessent>
<messagesreceived>
<message>
<id>Ubdfaea30-0c05-4bb3-bd20-dca77d9541c2</id>
<name></name>
<order>8</order>
<idblock>U2d1f4d48-67a9-401b-a461-84b3bddc0944</idblock>
<nameblock>Timer</nameblock>
<idblockoperation>Uef63be07-c57e-440c-8d44-488315337a5c</idblockoperation>
<nameoperation>countDown</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
</messagesreceived>
</lifeline>
<lifeline>
<id>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</id>
<name>Lifeline7</name>
```

```
<idBlockRepresented>U7c74f7b4-32da-4e85-9498-df94271d6569</idBlockRepresented>
<blockname>Drain</blockname>
<messagessent>
</messagessent>
<messagesreceived>
<message>
<id>Uf70a1606-10b1-467f-86a5-00db5b5b6318</id>
<name></name>
<order>11</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U7694c54e-f23b-47ee-bcec-2410a4cd4d1d</idblockoperation>
<nameoperation>openDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>U20ed75a1-ac05-4169-bf7a-3461b4c718d5</id>
<name></name>
<order>14</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U4d2f46ce-9efa-4863-858f-3be4514fef34</idblockoperation>
<nameoperation>closeDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
</messagesreceived>
</lifeline>
<message>
<id>Ub698ff60-0b3b-44f9-9fcd-0e3352effc8a</id>
<name></name>
<order>1</order>
<idblock>Ubbf24451-158e-42b1-a6f2-2556e108a88a</idblock>
<nameblock>Panel</nameblock>
<idblockoperation>Uc4cf2d18-58c0-4347-a33b-413abf85348e</idblockoperation>
<nameoperation>washMode</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U70b8b21d-9b80-49f7-84fa-e12eee91e3cd</id>
<name></name>
<order>2</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U008c9ca4-b07f-4667-8d2a-0114c4fee333</idblockoperation>
<nameoperation>startWash</nameoperation>
<idlifelinesend>U4c348bea-360b-4c99-ae3d-f4b95c17b5f9</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
```

```
<message>
<id>U39b133ad-057c-4553-af2b-3cf2fee8a5f0</id>
<name></name>
<order>3</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>Ucfe64e0c-c6aa-441e-afa4-20ad52e2c2c6</idblockoperation>
<nameoperation>openWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U34a7be9f-380c-4cf3-adca-f1f2eb921e63</id>
<name></name>
<order>4</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>Uef4cbd08-4767-4e1f-ae5e-3d22c29cc7a7</id>
<name></name>
<order>5</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U3e629aec-bf9d-4168-8536-289a9eb19411</id>
<name></name>
<order>6</order>
<idblock>Ub32e2b03-0aa3-4d7e-8695-257def3acad1</idblock>
<nameblock>WaterFeeder</nameblock>
<idblockoperation>U1afbaa9f-9c3f-43f1-ae80-298229eb955f</idblockoperation>
<nameoperation>closeWaterFeeder</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U41d4802b-87be-493d-867e-e8e022ec18c0</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>U3d4b7f3a-f6a4-4f35-87c4-f74b768b67b2</id>
<name></name>
<order>7</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U747280e4-c5d8-4137-bca2-f4933304ae22</idblockoperation>
```

```
<nameoperation>startRotator</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>Ubdfaea30-0c05-4bb3-bd20-dca77d9541c2</id>
<name></name>
<order>8</order>
<idblock>U2d1f4d48-67a9-401b-a461-84b3bddc0944</idblock>
<nameblock>Timer</nameblock>
<idblockoperation>Uef63be07-c57e-440c-8d44-488315337a5c</idblockoperation>
<nameoperation>countDown</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U127c5f63-0d06-4d2e-b673-853cd109f4dc</id>
<name></name>
<order>9</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>U190a7ebb-bc8a-405d-a801-431c98f690ce</idblockoperation>
<nameoperation>timesUp</nameoperation>
<idlifelinesend>Ub57e04e5-2976-434c-b1af-ea2f4259bcd1</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U157f010d-96df-4146-a3e5-07ee35b87ba7</id>
<name></name>
<order>10</order>
<idblock>U14e1466e-f316-4387-849e-1a753c987c39</idblock>
<nameblock>Rotator</nameblock>
<idblockoperation>U81870389-f40c-4cbb-99a5-776347cebe07</idblockoperation>
<nameoperation>stopRotator</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U1b578f2b-ec1d-4ef0-8eee-57f443d7ef6c</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>Uf70a1606-10b1-467f-86a5-00db5b5b6318</id>
<name></name>
<order>11</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U7694c54e-f23b-47ee-bcec-2410a4cd4d1d</idblockoperation>
<nameoperation>openDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
<message>
<id>U9b559247-a13c-4cc2-99d8-040760b38521</id>
```

```
<name></name>
<order>12</order>
<idblock>U11575726-b867-42b6-bc52-3a381e573049</idblock>
<nameblock>WaterLevelSensor</nameblock>
<idblockoperation>U2290bd01-2226-414c-9553-200712271cbd</idblockoperation>
<nameoperation>senseWaterLevel</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U1a4caf37-9a04-4038-aeee-f8a7e932c91d</id>
<name></name>
<order>13</order>
<idblock>U08bb5613-2612-4490-ba15-3e81c46166a3</idblock>
<nameblock>Controller</nameblock>
<idblockoperation>Udefa143a-9b2b-474e-853d-6cca4dc237db</idblockoperation>
<nameoperation>reachWaterLevel</nameoperation>
<idlifelinesend>U8e4e21e9-9978-4942-817e-38b17de93fbd</idlifelinesend>
<idlifelinereceive>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinereceive>
<messagetype>asynchCall</messagetype>
</message>
<message>
<id>U20ed75a1-ac05-4169-bf7a-3461b4c718d5</id>
<name></name>
<order>14</order>
<idblock>U7c74f7b4-32da-4e85-9498-df94271d6569</idblock>
<nameblock>Drain</nameblock>
<idblockoperation>U4d2f46ce-9efa-4863-858f-3be4514fef34</idblockoperation>
<nameoperation>closeDrain</nameoperation>
<idlifelinesend>Ue7c84bc1-32b8-40b2-b34e-a65767c3b14c</idlifelinesend>
<idlifelinereceive>U40ec7e76-c9b9-496d-b2d9-457b125ad1db</idlifelinereceive>
<messagetype>syncCall</messagetype>
</message>
</sequencediagram>
</Model>
```

Listagem C.16: BlocksModel.xml