ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

CONFIGURAÇÃO DINÂMICA DE INTERFACE COM O USUÁRIO

Trabalho de Conclusão de Curso

Engenharia da Computação

Fernando Antônio Farias Rocha
Orientador: Prof. Sergio Castelo Branco Soares

Recife, maio de 2007

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

CONFIGURAÇÃO DINÂMICA DE INTERFACE COM O USUÁRIO

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Fernando Antônio Farias Rocha
Orientador: Prof. Sergio Castelo Branco Soares

Recife, maio de 2007

UNIVERSIDADE
DE PERNAMBUCO

Fernando Antônio Farias Rocha

CONFIGURAÇÃO DINÂMICA DE INTERFACE COM O USUÁRIO

Àquele que sempre esteve ao meu lado sem nunca pedir nada em troca.

Akim

✶20/01/1995 11/06/2005✞

“Não importa quão estreito o portão,
Quão carregada a sentença de castigos,
Sou eu o mestre do meu destino,
Sou eu o capitão da minha alma.”

William Henley "Invictus"

Resumo

O desenvolvimento de certos sistemas é uma tarefa que pode acarretar grandes perdas, que vão desde econômicas até de vidas humanas, caso aconteça uma falha durante o processo de desenvolvimento. O processo de simulação é uma maneira de realizar análises destes sistemas sem a necessidade de concretizar a implementação destes, contudo, o desenvolvimento de simuladores pode ser uma tarefa tão, ou mais complexa do que o desenvolvimento do próprio sistema. O ambiente MPhyScaS busca uma maneira de diminuir a complexidade do desenvolvimento de simuladores, oferecendo uma linha de produtos para a geração de simuladores. Uma grande dificuldade enfrentada por este ambiente é a utilização de componentes para serem integrados a esta linha de produtos, os quais realizam tarefas semelhantes, mas necessitam de dados diferentes para a sua correta configuração. Neste contexto, o processo de desenvolvimento de interface gráfica é extremamente custoso, podendo chegar à metade do tempo gasto em toda a atividade de implementação. Desta forma, é inviável que o desenvolvedor de um componente necessite implementar também a interface com o usuário deste componente. Este trabalho propõe uma abordagem para a geração automática de interfaces com o usuário para ser integrado ao ambiente MPhyScas, desta forma abstraindo completamente a implementação da interface para o desenvolvedor de um novo componente.

Abstract

Some systems development can lead to huge losses, from economics to human life, if some failure occurs during the development process. Simulation is one way to perform some analyses of those systems without implementing them; however, simulators development might be as complex as or even more complex than the system development itself. The MPhyScaS environment addresses the issue of decreasing the simulator development complexity, through a product line to generate simulators. One difficulty in such environment is integrating components into this product line. Such components address similar goals, but might need different data to their configuration. In this context, the graphic user interface (GUI) development process is very expensive, in some cases almost half the implementation effort. This work proposes an approach to automatize the GUI generation in the MPhyScaS environment, abstracting its implementation from the components developers.

Sumário

Índice de Figuras

Tabela de Símbolos e Siglas

(Dispostos por ordem de aparição no texto)

MPhyScaS – *Multi-Physics and Multi-Scales Solver Enviroment*

SPL – *Software Products Line* (Linha de Produtos de Software)

GUI – *Graphic User Interface* (Interface Gráfica do Usuário)

UML – *Unified Modeling Language*

SWT – *Standard Widget Toolkit*

XML – *Extensible Markup Language*

FEM – *Finite Element Method* (Método do Elemento Finito)

DEMEC – Departamento de Mecânica Computacional

UFPE – Universidade Federal de Pernambuco

DSC – Departamento de Sistemas Computacionais

UPE – Universidade de Pernambuco

IDE – *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

API – *Application Program Interface* (Interface do Programa de Aplicação)

SO – Sistema Operacional

JNI – *Java Native Interface* (Interface Nativa de Java)

W3C – *World Wide Web Consortium*

HTML – *Hyper Text Markup Language*

JDOM – *Java-based Document Object Model*

CAD – *Computer Aided Design* (Desenho Auxiliado por Computador)

DLL – *Dinamic-Link Library* (Biblioteca de Ligação Dinâmica)

Agradecimentos

São muitas pessoas que eu tenho que agradecer por ter chegado onde estou, grandes amigos que não pretendo esquecer nunca. Muitos compartilharam este momento sofrendo junto a agonia de entregar a monografia agora, e outros participaram me dando total apoio quando necessário. Sei que esquecerei de citar alguns, mas tenho que deixar pra história os nomes das pessoas que marcaram estes cinco anos de curso, como: Tássia, Renata, Juliane, Andreza (primeiro as damas, lógico), Flávio (mestre Boina), Fred, Leandro, Paulo (grande PC), Leopoldo Teixeira, Leopoldo Rabelo, Thiago, Sandro, Samuel, Fernando Rocha (não sou eu, é Jet Lee), Judne, Ricardo Ulisses, entre outros.

Meus pais e irmãos, não sei como me aturaram durante tanto tempo... Vinte três anos suportando um chato feito eu, não é fácil... Obrigado.

Aos professores, a estes um agradecimento especial, para a grande maioria eu tenho que agradecer por me mostrarem atitudes das quais me orgulho e que irão me guiar para ser além de um bom profissional, uma pessoa de caráter.

Enfim, obrigado a todos que conviveram comigo durante estes cinco longos anos, e...

Até o mestrado... =D

Capítulo 1

Introdução

O desenvolvimento de alguns sistemas pode ser algo extremamente custoso como também arriscado, podendo envolver grande perda econômica e até de vidas humanas caso ocorra alguma falha durante o seu desenvolvimento. Simuladores são uma possibilidade para realizar análises sobre estes sistemas sem a necessidade de serem desenvolvidos previamente, contudo, o desenvolvimento destes simuladores podem ser tão, ou mais complexos do que o próprio sistema a ser simulado.

O ambiente de geração de simuladores, MPhyScaS (*Multi-Physics and Multi-Scales Solver Enviroment*), propõe uma maneira de realizar a geração de diversos simuladores, sem a necessidade de implementar completamente cada simulador, para isto utiliza técnicas de reuso de software e o conceito de Linhas de Produtos de Software (SPL).

Pelo fato de ser uma ferramenta tão versátil, capaz de gerar simuladores para a resolução de problemas completamente diferentes, acarreta em componentes de uma mesma categoria, que realizam tarefas parecidas, mas necessitam de dados diferentes. Com isso, há a necessidade de interfaces com o usuário diferentes, apesar de serem componentes de uma mesma categoria. Segundo Myers e Rosson [1], o tempo gasto na implementação da Interface Gráfica com o Usuário (GUI – *Graphic User Interface*) de um sistema pode chegar a ser mais da metade de todo tempo gasto durante o processo de implementação, com isto, caso o usuário que implementa o componente também tenha a tarefa de desenvolver a GUI deste componente, ocasionará, em no mínimo, a duplicação do tempo necessário para liberar este componente, além de requerer um conhecimento mais aprofundado da interface do sistema no qual este componente fará parte.

A geração automática de interface facilita este processo de integrar um componente a uma linha de produtos, pois o desenvolvedor do componente, bastará informar as características da GUI que este componente necessita. Este processo de geração automática é divido em três etapas: **(i)** a definição das tarefas que o usuário deverá realizar na GUI gerada; **(ii)** a compilação destes dados para uma forma de representação abstrata; e **(iii)** a montagem desta GUI, de acordo com a sua representação abstrata.

No ambiente MphyScaS existe uma categoria capaz de modelar situações do mundo real, denominada Fenômeno. Esta categoria pode ser vista como o ponto de maior variância entre os componentes existentes, já que cada um destes componentes necessita de uma GUI própria.

Esta monografia apresenta uma abordagem para a geração da GUI para componentes desta categoria Fenômeno, por ser uma categoria em que cada um de seus componentes necessita de dados de entrada diferentes, dificultando a geração de uma interface padrão para a categoria. Também não sendo viável requerer que o desenvolvedor de um Fenômeno implemente a sua GUI, já que é desejável uma abstração deste desenvolvedor em relação aos simuladores gerados pelo MPhyScaS.

1.1 Principais Contribuições

As principais contribuições deste trabalho estão listadas a seguir:

- A ferramenta procura abstrair o desenvolvimento da GUI de novos componentes desenvolvidos para o ambiente MPhyScaS, agilizando o processo de criação destes novos componentes.
- A GUI gerada procura implementar restrições aos parâmetros cadastrados, não necessitando uma programação adicional por parte do desenvolvedor do componente.
- Incentiva a adoção do ambiente MphyScaS para o desenvolvimento de simuladores, fornecendo maior suporte para o desenvolvimento e reutilização de componentes.

1.2 Organização da Monografia

Esta monografia está organizada em Capítulos e Anexos. A seguir será detalhado o conteúdo de cada parte:

O Capítulo 2 aborda as principais tecnologias utilizadas durante o desenvolvimento deste trabalho. Detalhando o ambiente em que a ferramenta proposta será integrada, como também as bibliotecas e técnicas utilizadas.

O Capítulo 3 entra em detalhes sobre a ferramenta desenvolvida, detalhando seu funcionamento, casos de uso e diagramas UML (*Unified Modeling Language*).

O Capítulo 4 aborda dois estudos de caso, onde temos o cadastro de novos Fenômenos no ambiente MPhyScaS, desta forma mostrando a ferramenta em atividade, além de realizar uma análise comparando os dois estudos de caso.

O Capítulo 5 concluí o trabalho, mostrando os resultados obtidos, as principais contribuições referentes a este trabalho e possíveis melhorias que podem ser realizadas na ferramenta proposta.

O Anexo A contém o documento de requisitos referentes à ferramenta desenvolvida.

O Anexo B contém os documentos detalhados de cada caso de uso correspondente aos requisitos catalogados.

Capítulo 2

Principais Tecnologias Utilizadas

Este trabalho propõe uma geração automatizada de interfaces gráficas para a entrada de dados dos simuladores gerados pelo ambiente MPhyScaS. Este ambiente pode ser considerado uma Linha de Produtos de Software, onde seu produto final são os simuladores gerados. Na abordagem adotada, são utilizadas algumas técnicas de Engenharia de Software como reuso e componentização, tendo toda a implementação desta ferramenta baseada no pacote gráfico SWT (*Standard Widget Toolkit*) de Java e para prover persistência dos dados é utilizada a linguagem XML (*Extensible Markup Language*). Nesta seção é descrita em mais detalhes as tecnologias utilizadas.

2.1 MPhyScaS (*Multi Physics and Multi Scales Solver Environment*)

O desenvolvimento de alguns sistemas é extremamente custoso e arriscado, principalmente se não houver um estudo a priori. Sistemas como a realização de manutenção preventiva de dutos de petróleo e gás pode envolver a vida de muitos seres humanos, caso ocorra um erro durante o processo de manutenção ou erros nos cálculos de quando esta manutenção deva ocorrer. Um atraso na manutenção pode vir a ser fatal.

Cálculos como este, para realizar a analise do desgaste da tubulação de petróleo e gás, são muito complexos, sendo um sistema que envolve diversos fenômenos atuando em uma geometria contínua [2], tornando extremamente difícil a resolução precisa deste problema. A utilização de simuladores é uma maneira econômica, rápida e segura para realizar análises em sistemas reais sem a necessidade de executar tais processos manualmente. No entanto, o desenvolvimento de simuladores pose ser algo tão complexo quanto o desenvolvimento do próprio sistema a ser analisado [3].

Simuladores que contêm diversos fenômenos atuando em uma geometria contínua impõem dificuldades que tornam o desenvolvimento destes ainda mais complexo. Fatos como as relações dos dados entre os fenômenos e a existência de especialidades distintas para cada fenômeno [2] formam um ambiente propício para o desenvolvimento de simuladores ineficientes, com custo e tempo de desenvolvimento relativamente altos. A aplicação de técnicas de Engenharia de

Software[4], como reuso [5] e modularização dos componentes desenvolvidos [4], tornariam o desenvolvimento mais eficaz.

O MPhyScaS (*Multi-Physics and Multi-Scales Solver Environment*) consiste em um ambiente para a geração de simuladores multi-físicos baseado no Método do Elemento Finito (FEM – *Finite Element Method* [6]). O ambiente MPhyScaS trata-se de um projeto em desenvolvimento pelo Departamento de Mecânica Computacional da Universidade Federal de Pernambuco (DEMEC – UFPE) em parceria com o Departamento de Sistemas Computacionais da Universidade de Pernambuco (DSC – UPE) e trata-se de uma extensão do trabalho proposto por Lencastre [7].

O MPhyScaS tenta ser uma solução para as dificuldades apresentadas no decorrer do desenvolvimento de simuladores. O ambiente pode ser considerado como uma Linha de Produto de Software, tendo como resultado, simuladores multi-físicos. Esta versatilidade, um ambiente capaz de gerar diversos simuladores diferentes, só é possível pelo fato destes simuladores gerados serem definidos como uma estrutura computacional suportada por uma linguagem bastante ampla de padrões [8].

O ambiente MPhyScaS pode ser dividido em três partes: **(i)** o Gerador / Configurador de simuladores, responsável por juntar todos os Componentes de Software recebido como entrada e agregá-los ao núcleo do Simulador e assim, gerar um Simulador; **(ii)** o Repositório de Componentes de Software, sistema capaz de armazenar e organizar todos os componentes possíveis de serem agregados a um Simulador; **(iii)** e o produto final, considerado o Simulador, propriamente dito. A organização do ambiente pode ser visualizada na Figura 2-1.

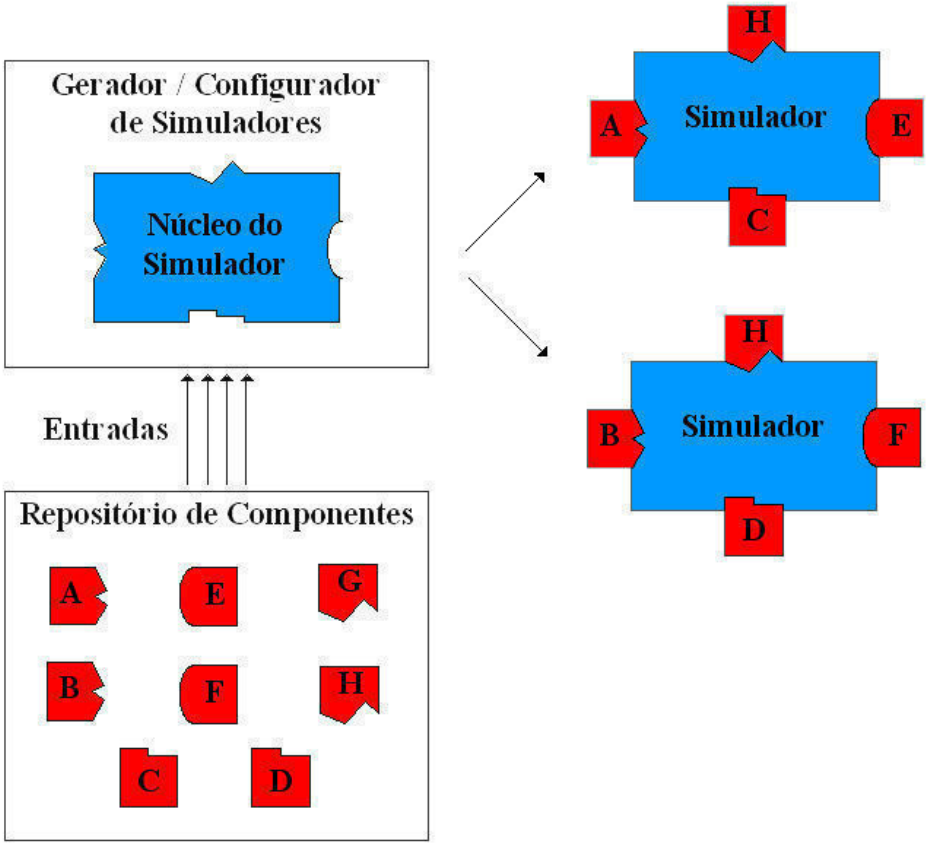

Figura 2-1 Organização do ambiente MPhyScaS.

Assim, a utilização do ambiente MPhyScaS pode ser dividida em duas etapas: **(i)** a utilização do Gerador / Configurador de Simuladores, onde um usuário com um maior nível de conhecimento utiliza o ambiente para informar quais Componentes de Software serão utilizados para a geração de um simulador especialista em resolver problemas de uma certa categoria; e **(ii)** o simulador, produto do Gerador / Configurador, sendo disponibilizado para usuários em geral, sendo sua utilização limitada a resolver problemas de uma categoria específica.

Os Componentes de Software cadastrados no Repositório de Componentes do MPhyScaS podem ser agrupados em categorias, como: Geradores de Malha, para a geometria em que o Simulador irá atuar; Algoritmos de resolução, componentes capazes de realizar o cálculo dos sistemas algébricos envolvidos na simulação; Fenômenos, componentes que representam os fatos a serem simulados, entre outras categorias. Componentes de uma mesma categoria realizam tarefas semelhantes e geralmente possuem dados de entrada para a sua correta configuração padronizados. O usuário especialista, que manipula com o Gerador / Configurador de Simuladores, pode ainda realizar o cadastro de novos componentes de software, caso deseje utilizar um componente não existente no Repositório, para isto basta realizar a implementação de acordo com o padrão de desenvolvimento imposto pelo ambiente MPhyScaS.

Pelo fato de tentar ser um Gerador de Simuladores completamente genérico, o desenvolvimento de novos componentes da categoria Fenômeno fica limitado, porque nem todos os Fenômenos necessitam dos mesmos dados de entrada para serem configurados para a sua utilização na simulação. Sendo necessária uma Interface Gráfica com o Usuário (GUI) específica para cada Fenômeno integrado ao ambiente MPhyScaS.

Aplicação no Trabalho

O desenvolvimento de um componente de software científico, o caso dos componentes utilizados pelo ambiente MPhyScaS, não é uma tarefa trivial, acrescentar a este desenvolvimento a especificação e implementação da GUI deste componente tornaria o processo extremamente custoso. Além do aumento do tempo gasto durante a implementação de um novo componente, exigiria que o desenvolvedor tivesse conhecimento da completa estrutura da GUI dos simuladores que teriam este componente.

Este trabalho propõe uma solução para a criação da GUI de Fenômenos que serão agregados ao MPhyScaS, realizando uma geração automática de interface gráfica para estes componentes.

2.2 Linhas de Produtos

A idéia de linha de produtos já é amplamente utilizada em processo de fabricação em diversas áreas, sendo basicamente a utilização de um mesmo núcleo para realizar a montagem de diversos produtos semelhantes. Por exemplo, uma indústria automotiva pode produzir milhares de carros de um mesmo modelo contendo pequenas variações, para isto necessitando apenas de partes que sofreram um processo de arquitetura cuidadoso e de um núcleo que esteja preparado para realizar a montagem destas partes.

Esta idéia de linha de produtos foi transplantada para a área de desenvolvimento de software sendo denominada Linhas de Produto de Software (*Software Product Lines* – SPL [9]). O conceito de uma SPL é semelhante ao conceito geral sendo que aplicado à área específica. Basicamente seria a produção de um conjunto de sistemas de software com algumas características em comum, produzidos por um mesmo núcleo que atenda a uma necessidade especifica do mercado ou missão [10].

Desta forma, uma SPL visa a produção de diversos produtos, sendo todos de um mesmo tipo. Como por exemplo, o MphyScaS onde sua SPL tem como foco o desenvolvimento de diversos simuladores, onde cada um destes que são gerados possuem pequenas diferenças entre eles.

Apesar da idéia de criar software a partir de partes reutilizáveis já ser discutida há décadas, não se conseguia um bom aproveitamento destas técnicas. Apenas recentemente, com os avanços na área de linha de produtos, é que pode ser considerado que o conceito de SPL's pode de fato alcançar os objetivos aos quais se propôs [11].

Uma SPL pode ser descrita por quatro conceitos básicos: **(i)** entradas de software, como requisitos, componentes de software, casos de testes e documentação – que podem ser configurados de modo a gerar todos os produtos da SPL; **(ii)** decisões do modelo, definindo opções e variações dos produtos gerados; **(iii)** mecanismo de produção, que é responsável em juntar as entradas de software e utilizar as decisões tomadas para resultar no produto de saída; e **(iv)** softwares de saída, que são o conjunto de produtos finais da SPL. Este processo é ilustrado na Figura 2-2.

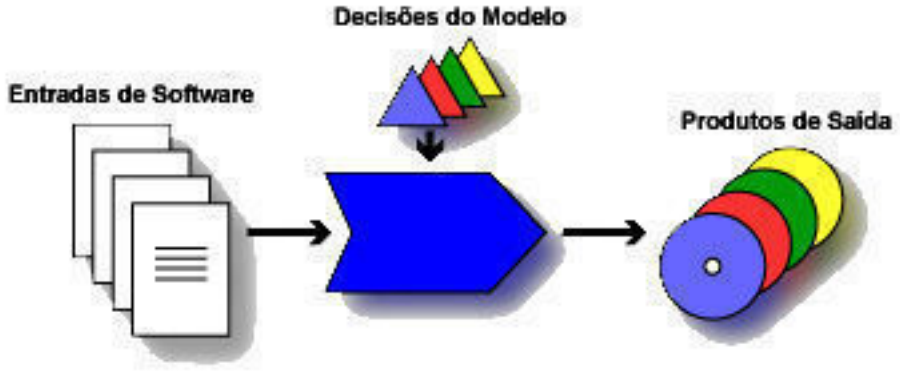

Figura 2-2 Modelo de uma Linha de Produto de Software.

Estes conceitos ilustram os principais objetivos de uma SPL, que são juntar as semelhanças e gerir as variações de forma a buscar sempre melhoria da qualidade do produto final, tempo de desenvolvimento, custo e complexidade de criar e manter uma linha de produtos de software similares.

No entanto é necessário observar que a reutilização de software para diminuir os custos de desenvolvimento e aumentar a qualidade já é uma técnica utilizada há bastante tempo. A diferença da reutilização já existente e de SPL é que a reutilização é um processo oportunista, que dá ênfase ao reuso de pequenas partes do código que já foram desenvolvidas anteriormente, enquanto que as SPLs reutilizam código de forma planejada. Assim, o processo de

desenvolvimento de cada módulo de uma SPL é desenhado e desenvolvido visando à reutilização, modificação e otimização de forma que este módulo consiga ser implantado nos diversos produtos produzidos pela SPL.

Aplicação no Trabalho

A integração de novos componentes a uma SPL pode ser um processo custoso, principalmente caso este processo necessite da implementação da GUI do componente a ser integrado. Desta forma, este projeto visa fazer com que os componentes de software da categoria Fenômeno possam ser integrados a SPL do MPhyScaS mais facilmente, sem a necessidade do desenvolvedor de um novo Fenômeno se preocupe em implementar a GUI de configuração deste.

2.3 SWT (*Standard Widget Toolkit*)

A SWT (*Standard Widget Toolkit* [12]) é uma biblioteca Java para a criação de GUI. Inicialmente desenvolvida pela *IBM Corporation* como parte integrante do projeto *Eclipse*, foi posteriormente separada a implementação da interface do restante do projeto da IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) e atualmente é disponibilizada como uma biblioteca Java para criação de GUI, sendo mantida pela *Fundação Eclipse*.

Um dos principais objetivos no desenvolvimento da biblioteca SWT foi em prover um alto desempenho, para isto a biblioteca utiliza um recurso de Java para acessar a API (*Aplication Program Interface* – Interface do Programa de Aplicação) do Sistema Operacional (SO) no qual a aplicação está executando. Este recurso, *Java Native Interface* (JNI), possibilita a comunicação de aplicações Java com outras linguagens, entre elas C, uma linguagem reconhecida pelo seu alto desempenho e por ser a linguagem utilizada para o desenvolvimento dos principais Sistemas Operacionais [13].

A tarefa de desenhar os componentes da GUI é algo bastante custosa para a aplicação. Assim a SWT se beneficia da sua comunicação direta com o SO e repassa para este a solicitação de desenhar os componentes de interface. Desta forma, o desenho dos componentes tem seu desempenho melhorado pelo fato do SO ter acesso direto aos recursos da máquina.

Devido a esta comunicação entre a biblioteca SWT e o SO, faz com que o estilo dos componentes, utilizados por uma aplicação desenvolvida sobre SWT, reflita o estilo dos componentes de interface do sistema operacional, como por exemplo, alterando o padrão de cores que o SO utiliza em seus botões, vai alterar o padrão de cores utilizados na aplicação SWT, exemplificado na Figura 2-3.

Sempre que a SWT recebe uma solicitação de desenhar um componente de interface gráfica, esta solicitação é repassada para o SO, a SWT mapeia cada um dos seus componentes de sua interface gráfica a um componente do SO, criando assim, uma camada de componentes sobre os componentes do SO.

Aplicação no Trabalho

O ambiente MPhyScaS se aproveita das características da biblioteca SWT, e tem toda a sua interface desenvolvida sobre esta biblioteca, oferecendo um bom desempenho e abstração dos

componentes gráficos. Desta forma, a abordagem de geração de interface é implementada também utilizando a biblioteca SWT.

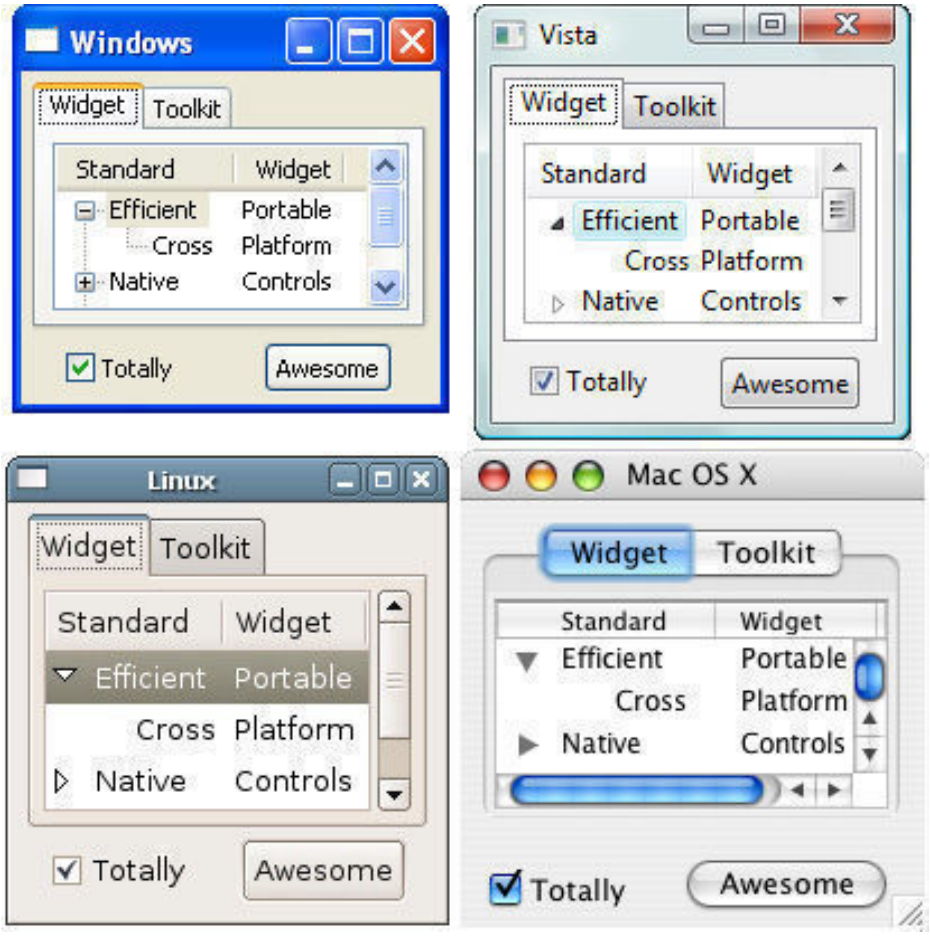

Figura 2-3 Comportamento dos componentes gráficos de uma aplicação SWT ao ser executada em diversos sistemas operacionais.

2.4 XML (*Extensible Markup Language*)

A linguagem XML (*Extensible Markup Language*) é uma especificação técnica desenvolvida pela W3C (*World Wide Web Consortium* [14]) para superar as limitações da linguagem utilizada como padrão para o desenvolvimento para a internet. Atualmente a linguagem XML passou a ser um padrão para a definição e comunicação de dados estruturados que trafegam pela internet. A linguagem XML especifica um conjunto de regras e convenções para a estruturação do arquivo, facilitando a geração, leitura e o entendimento do seu conteúdo pela máquina.

Na Figura 2-4 pode ser visualizado um exemplo de um documento XML simples, mostrando a relação de funcionários e seus respectivos salários. Pode ser reparado o cabeçalho XML na primeira linha do exemplo.

Apesar da grande semelhança dos arquivos escritos em XML e HTML (*Hyper Text Markup Language*), existem grandes diferenças no objetivo de cada linguagem de marcação. A linguagem HTML é voltada para a especificação da estrutura visual do conteúdo da internet, enquanto que a linguagem XML procura estruturar os dados contidos em seu arquivo. Ambas as linguagens utilizam *tags* como marcação do documento, mas existe uma grande diferença na utilização destas *tags* em cada linguagem. No HTML cada *tag* possui um significado fixo, enquanto que em XML a *tag* é criada a cargo do desenvolvedor e tem seu sentido interpretado por quem estiver lendo o documento.

```
<?xml version="1.0" ?>
<funcionarios>
        <funcionario>
                <nome>Alessandra</nome>
                <salario>1000</salario>
        </funcionario>
        <funcionario>
                <nome>Paulo</nome>
                <salario>700</salario>
        </funcionario>
</funcionarios>
```

Figura 2-4 Exemplo de código XML

Aplicação no Trabalho

O processo de geração de interfaces é composto por uma etapa em que é necessário representar o que deve ser gerado de uma forma abstrata em que não tenha uma correspondência direta em como esta interface deverá ser gerada. Neste trabalho, esta fase de representação abstrata é representada em um arquivo de dados utilizando o padrão XML. Para a manipulação dos documentos XML, é utilizado a biblioteca JDOM (*Java-based Document Object Model*) [15].

2.5 Padrão JavaBeans

Com o intuito de definir um padrão de componentes que possam ser reutilizados, a *Sun Microsystem* em conjunto com outras empresas, entre elas a *Inprise* (antiga *Borland*) [16], criaram uma especificação detalhando um modelo de componente Java, estes componentes são conhecidos como JavaBeans [17]. Uma grande vantagem do padrão JavaBeans é que ele não altera a linguagem Java, sendo apenas uma especificação de desenvolvimento de componentes [18].

Como ocorre em alguns ambientes de desenvolvimento, os componentes desenvolvidos sobre esta especificação podem ser manipulados visualmente, como por exemplo, para adicionar um *bean* para um projeto, basta arrastar o *bean* específico para a área de trabalho do projeto. E, modificando as propriedades do *bean* são alteradas a sua aparência e funcionalidade.

Os *beans* utilizam eventos para realizar a comunicação com outros *beans*. Eventos são objetos criados por um gerador de uma ação e divulgados para todos os outros objetos que estiverem cadastrados para receber esta ação.

As propriedades de um *bean* definem uma característica ou um atributo, ao alterar uma destas propriedades é modificado seu comportamento ou sua aparência e esta modificação pode gerar um evento. Estas propriedades são referenciadas por um nome e podem ter qualquer tipo, mas só podem ser acessadas através de métodos padronizados para leitura ou escrita, as assinaturas destes métodos são:

```
public <tipo> get<NomePropriedade> ();

public void set<NomePropriedade> (<tipo> valor);
```

Quando é utilizado um *bean* em uma ferramenta de desenvolvimento visual, esta ferramenta necessita expor as propriedades e eventos do *bean*. Para isto, ela utiliza a API de *reflection* de Java, que permite descobrir características de um objeto via padrões de projeto. Para que este mecanismo funcione é necessário que obedeça algumas convenções de nomenclatura, como por exemplo, devem existir os métodos de acesso *get / is* e *set* de cada atributo da classe, como já descrito anteriormente.

Grande parte dos componentes de software necessitam manter informações sobre seu estado durante a sua execução, para uma posterior exibição destes dados. Para isto, a informação dos estados dos componentes deve ser salva em um meio persistente de armazenamento. O padrão JavaBeans utiliza o mecanismo de serialização de Java para persistência. Tudo que um componente JavaBean deve fazer é implementar a *interface java.io.Serializable*.

Diversos *beans* podem ser combinados com o intuito de formar aplicações inteiras, como interfaces gráficas com o usuário e até mesmo, criar outros *beans*, que podem ser compostos por uma ou mais classes Java [19].

Aplicação no Trabalho

A divisão em componentes de um sistema é uma forma de facilitar o seu desenvolvimento, manutenção e uma possível expansão deste[4]. Por este fato, o padrão de componentes JavaBeans é utilizado neste trabalho tentando aprimorar o seu desenvolvimento.

Capítulo *3*

MPhyScaS GUI Generator

Neste capítulo são apresentados os principais conceitos do *MphyScaS GUI Generator*, além de entrar em detalhes sobre o planejamento da ferramenta, como os casos de uso e diagramas de classe, e de sua implementação, como detalhar cada etapa do processo da geração automática de interface referenciando os módulos implementados.

3.1 Motivação

O processo de definição e desenvolvimento de uma interface gráfica entre o computador e o ser humano é extremamente importante para a qualidade do sistema em desenvolvimento, além de ser uma das etapas que consome maior tempo, chegando a ser mais de 50% do tempo total gasto no desenvolvimento do sistema [1].

Numa SPL como o MPhyScaS, cada componente que pode ser integrado ao núcleo do simulador necessita de uma GUI própria, o que leva a atrasos de desenvolvimento e dificuldade de uso deste componente.

Os componentes que podem ser utilizados pelo ambiente MPhyScaS podem ser divididos em categorias, como: **(i)** Carregador de Geometria, categoria de componentes responsáveis por carregar a definição de uma geometria, que será utilizada na simulação, criada em um programa de desenho auxiliado por computador (CAD - *Computer Aided Design*); **(ii)** Fenômenos, categoria de componentes que possuem a capacidade de modelar situações do mundo real; e **(iii)** Geradores de Malha, a simulação não pode ser executada em uma geometria continua, que é como é definida a geometria em um CAD, estes componentes são responsáveis em discretizar a geometria a ser utilizada na simulação; entre outras categorias existentes. Desta forma, componentes de uma mesma categoria, quando necessitam de dados do usuário, geralmente apresentam uma GUI padrão desta categoria. Mas, no caso da categoria de componentes Fenômeno, existe uma particularidade, cada componente desta categoria necessita de uma entrada de dados diferente. Impossibilitando que exista uma GUI padrão para esta categoria de componentes.

Dado que cada Fenômeno possui uma GUI diferente, é necessário, por parte do desenvolvedor de um componente desta categoria, o conhecimento referente a GUI do simulador, já que este necessita desenvolver a GUI do Fenômeno integrada ao ambiente do simulador.

Neste trabalho é apresentada uma solução para este problema apresentado, fornecendo uma interface de comunicação entre a GUI necessária para cada Fenômeno e o módulo do simulador (Figura 3-1), abstraindo completamente o desenvolvimento da interface para o usuário do MPhyScaS que irá implementar um novo Fenômeno. Na Figura 3-1 pode ser visto três componentes (A, C e F) que são de categorias que possuem uma GUI padrão, enquanto que a ferramenta proposta está facilitando a integração de novos Fenômenos ao Simulador, já que está possibilitando uma maior variedade de pontos de integração aos novos Fenômenos serem integrados ao Núcleo do Simulador.

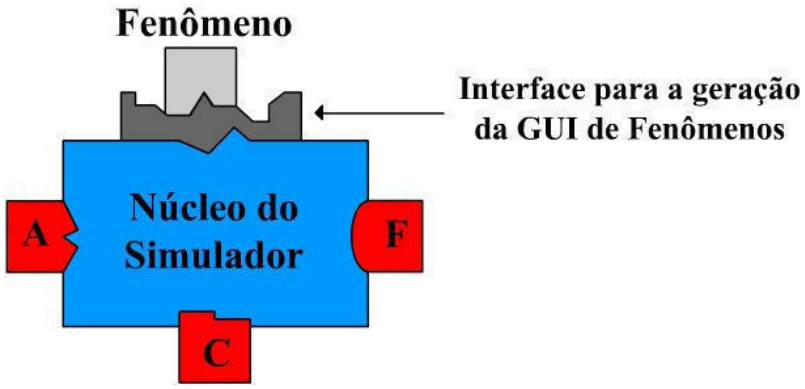

Figura 3-1 Modelo da ferramenta desenvolvida aplicada ao ambiente MPhyScaS.

3.2 A proposta

As abordagens utilizadas no processo de geração automática de interface, basicamente procuram realizar o mapeamento entre modelos que representam tarefas que deverão ser realizadas pelo usuário e o modelo da GUI. Geralmente, existe uma terceira etapa entre estas duas, a etapa que representa de forma abstrata a interface a ser gerada [20].

A abordagem adotada neste trabalho possui estas três etapas, **(i)** o cadastro da interface a ser gerada, definindo as tarefas em que o usuário deverá realizar no uso da interface; **(ii)** a compilação destes dados para um formato de arquivo, tornando esta definição da interface abstrata, permitindo qualquer sistema capaz de ler e compreender o arquivo montar a interface; e **(iii)** a etapa final, em que a GUI é montada a partir deste arquivo que representa, de forma abstrata, a interface.

O desenvolvimento deste trabalho visa resolver a dificuldade enfrentada no ambiente MPhyScaS, onde cada componente de software da categoria de Fenômenos necessita de uma GUI própria para a entrada dos dados referentes ao Fenômeno especifico. O ambiente de desenvolvimento de simuladores MPhyScaS consiste em uma SPL que possui um repositório de Componentes expansível, possibilitando assim, o usuário desenvolver novos componentes para serem utilizados em sua SPL. Este ambiente pode ser separado em três módulos: **(i)** o Gerador / Configurador de Simuladores, responsável por juntar todos os Componentes de Software recebido como entrada e agregá-los ao núcleo do simulador e assim, gerar um simulador; **(ii)** o Repositório de Componentes de Software, sistema capaz de armazenar e organizar todos os componentes possíveis de serem agregados a um simulador; **(iii)** e o produto final, considerado o simulador, propriamente dito.

Ao ser desenvolvido um novo componente, este necessita ser cadastrado no Repositório de Componentes para que fique disponível para o Gerador / Configurador de Simuladores utilizá-lo e gerar um simulador contendo este novo componente. Neste momento, realiza-se a primeira etapa da geração automática da interface gráfica, em que o usuário detalha a GUI que deve ser gerada quando este novo componente for utilizado em um simulador, informando quantos parâmetros devem ser solicitados ao usuário do simulador, como também o tipo de cada parâmetro e suas restrições.

A segunda etapa da geração ocorre quando este componente é selecionado para integrar um simulador específico, neste momento em que o Gerador / Configurador de Simuladores recebe a solicitação para gerar o simulador, os dados referentes à interface gráfica a ser gerada é compilado e salvo em um arquivo contendo toda a descrição desta interface.

Com o simulador gerado, ocorre a terceira etapa da geração da GUI, no momento da utilização deste simulador, quando o usuário selecionar a opção de informar os dados de configuração do Fenômeno que foi cadastrado, o simulador irá ler a descrição da GUI que possui e realizará a montagem desta interface, de forma transparente para o usuário que realizou o cadastro do novo Fenômeno.

3.3 A Ferramenta

Nesta seção é apresentada a ferramenta desenvolvida no âmbito desta monografia. Será detalhada a arquitetura, diagramas UML e diagrama de casos de uso da ferramenta, como também será exposto os detalhes da utilização e do funcionamento da ferramenta, enfatizando as três fases existentes em um processo de geração automática de GUI.

3.3.1 Arquitetura Proposta

A ferramenta proposta foi dividida em dois módulos, o módulo *Creator* e o módulo *Builder*. O módulo *Creator* é responsável pelo cadastro dos parâmetros e por compilar esta definição para uma forma abstrata de representação da interface. Ficando o módulo *Builder* responsável por ler esta representação abstrata e montar a GUI.

Ambos os módulos foram desenvolvidos com o intuito de serem integrados a um outro sistema, no caso específico desta monografia, os módulos serão integrados ao ambiente MPhyScaS, como mostram as Figura 3-2 e Figura 3-3 que detalham a arquitetura da ferramenta desenvolvida integrada com o ambiente MPhyScaS.

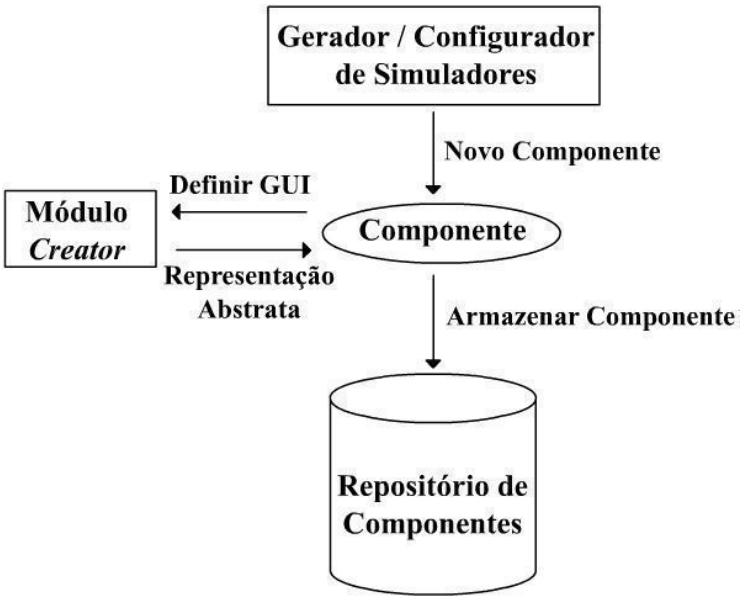

Figura 3-2 Arquitetura da integração do módulo *Creator* ao ambiente MPhyScaS.

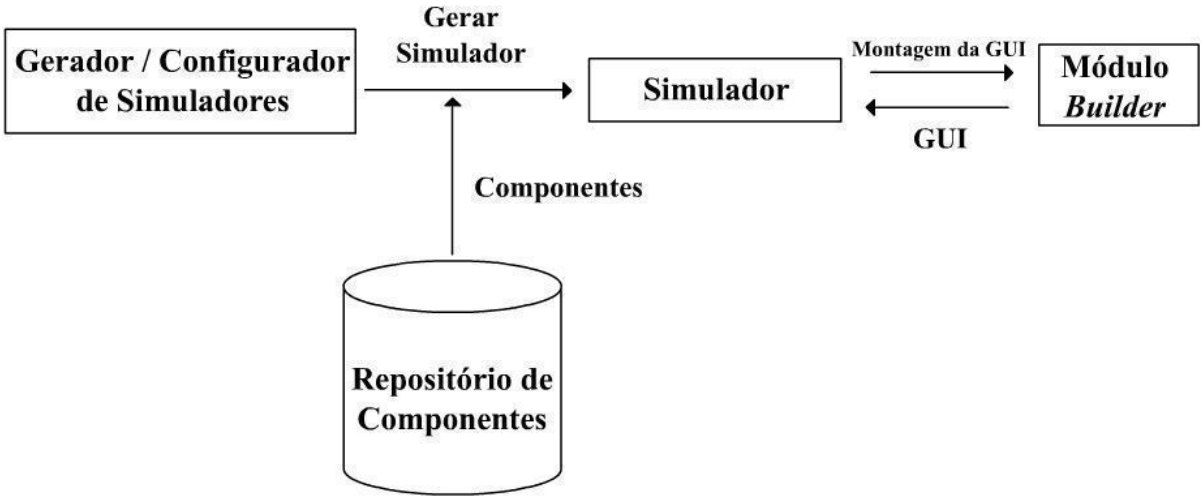

Figura 3-3 Arquitetura da integração do módulo *Builder* ao ambiente MPhyScaS.

Especificamente, a Figura 3-2 ilustra o funcionamento do módulo *Creator*. No momento em que o Gerador / Configurador de Simuladores recebe uma solicitação para cadastro de um novo componente da categoria Fenômeno, este solicita ao módulo *Creator* que gerencie o cadastro dos dados de configuração deste novo Fenômeno. Após ter sido realizada toda a definição destes dados de entrada, o módulo *Creator* compila estes dados para uma representação abstrata, guardando esta representação em um arquivo XML junto com o componente no Repositório de Componentes do MPhyScaS.

O segundo módulo, ilustrado na Figura 3-3, entra em funcionamento apenas após a geração de um simulador. No momento em que o usuário final do simulador gerado solicita que apareça a interface do Fenômeno para poder configurá-lo, o simulador repassa esta solicitação para o módulo *Builder* para que este leia o arquivo que contem a representação abstrata da GUI a ser gerada e realize a montagem desta. Após o processo de montagem, o módulo retorna para o simulador a GUI montada para ser exibida e utilizada pelo usuário do simulador.

O módulo *Creator* interage com o usuário denominado *Usuário Criador*, desta iteração o ator *Módulo Creator* fica responsável por um caso de uso, sendo a criação do arquivo que representa de forma abstrata a interface a ser gerada posteriormente. Já o ator *Usuário Criador* é responsável por quatro casos de uso: **(i)** criar parâmetro para ser exibido na GUI; **(ii)** editar um parâmetro já gerado; **(iii)** excluir um dos parâmetros gerados; e **(iv)** definir as restrições que os parâmetros deverão cumprir no momento em que o usuário for entrar com os dados. Estes casos de uso estão representados no diagrama de Casos de Uso do módulo *Creator*, ilustrado na Figura 3-4.

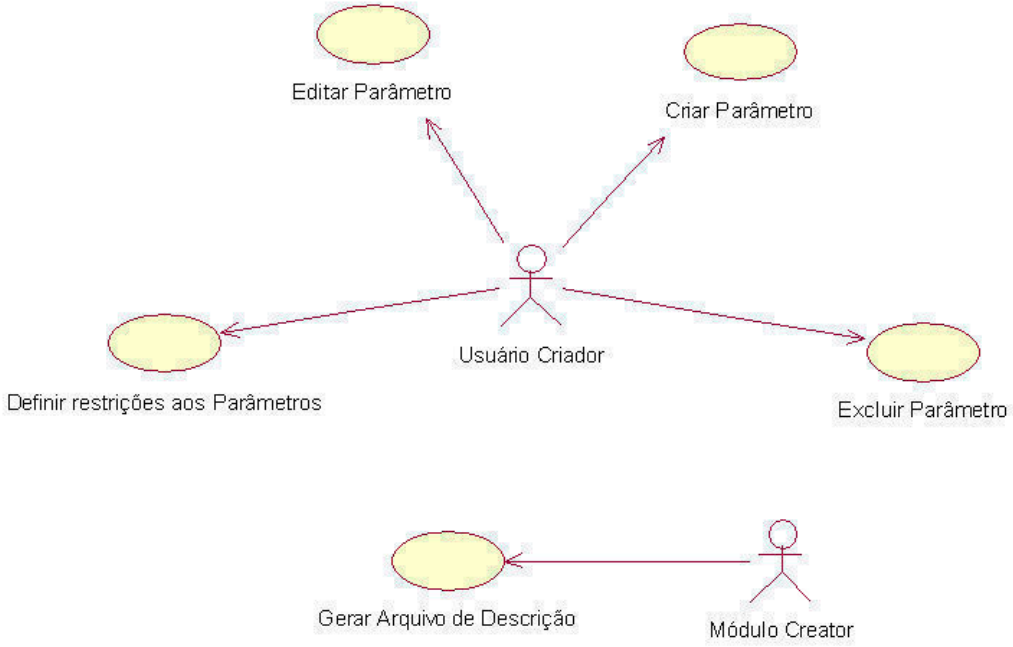

Figura 3-4 Diagrama de caso de uso do módulo *Creator*.

Já o módulo *Builder* interage com um outro usuário, o ator *Usuário Final*. Este usuário fica responsável por interagir com a interface gerada em um Simulador, enquanto que o ator *Módulo Builder* é o responsável por **(i)** ler e **(ii)** gerar a interface da representação abstrata gerada pelo módulo *Creator*, **(iii)** garantir que as restrições de um parâmetro sejam atendidas; e **(iv)** notificar ao sistema em que está integrado, que houve alguma alteração nos valores do parâmetro. Estes casos de uso podem ser vistos na Figura 3-5.

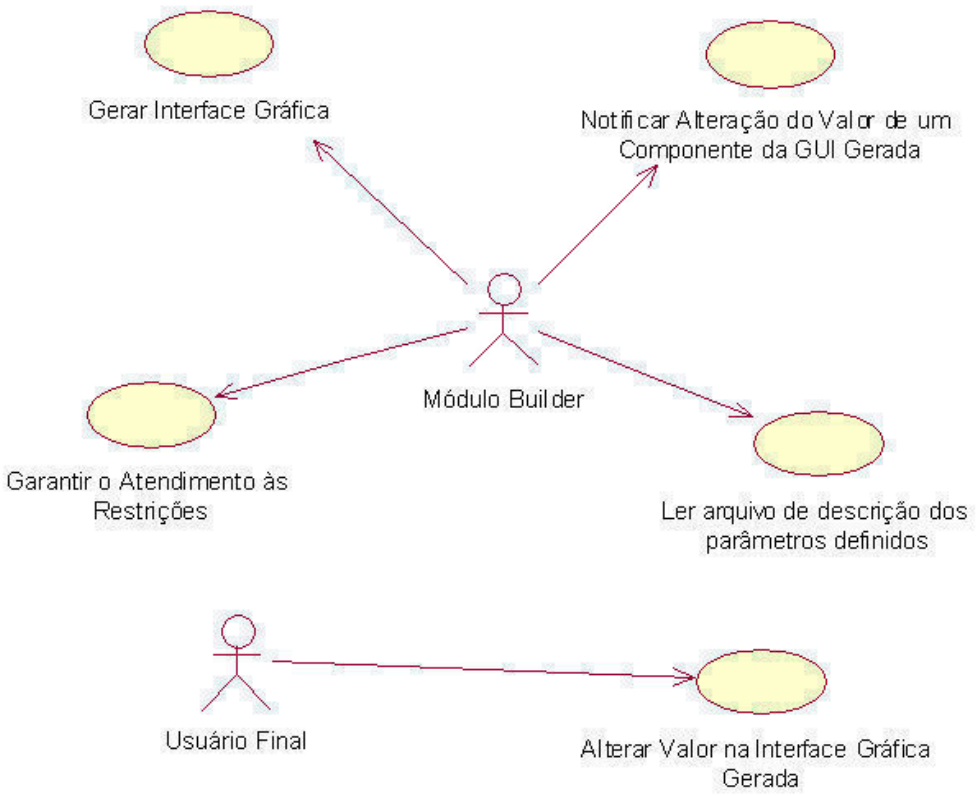

Figura 3-5 Diagrama de casos de uso do módulo *Builder*.

Os documentos detalhados de cada requisito cadastrado e os respectivos caso de uso estão documentados como Anexo A e Anexo B, respectivamente, desta monografia.

A implementação da ferramenta foi dividida em quatro pacotes: **(i)** um pacote responsável por gerenciar os tipos de cada parâmetro inserido pelo usuário e seu componente gráfico correspondente no momento em que for gerada a GUI; **(ii)** um pacote contendo os tipos de restrições que cada parâmetro pode conter; **(iii)** um pacote responsável pela implementação do módulo *Creator*; e **(iv)** um pacote responsável pela implementação do módulo *Builder*. A Figura 3-6 ilustra o diagrama de pacotes da ferramenta.

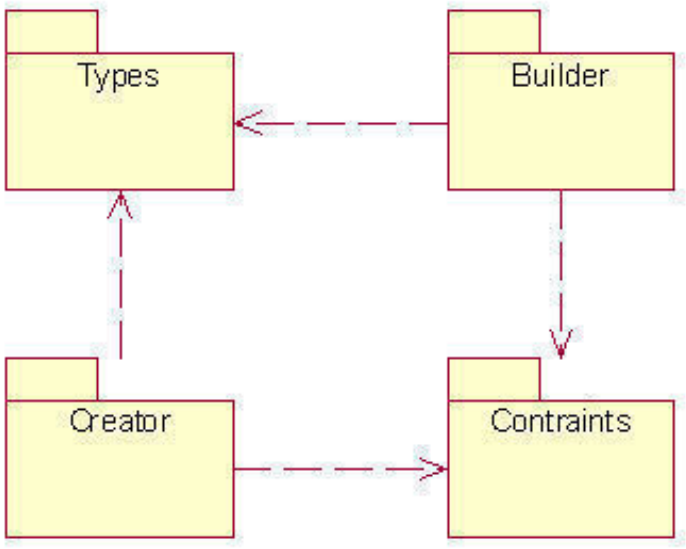

Figura 3-6 Diagrama de pacotes da ferramenta proposta.

A Figura 3-7 ilustra o diagrama de classe do pacote de tipos – *Types*. Este pacote é responsável por armazenar os tipos possíveis de cada parâmetro que o usuário poderá inserir e os tipos de cada componente gráfico que pode ser utilizado na montagem da GUI gerada. A classe *ParameterComponentTypeAssociation* é responsável por realizar a associação entre cada tipo existente. Já a Figura 3-8 representa o diagrama de classe do pacote de restrições – *Constraints*.

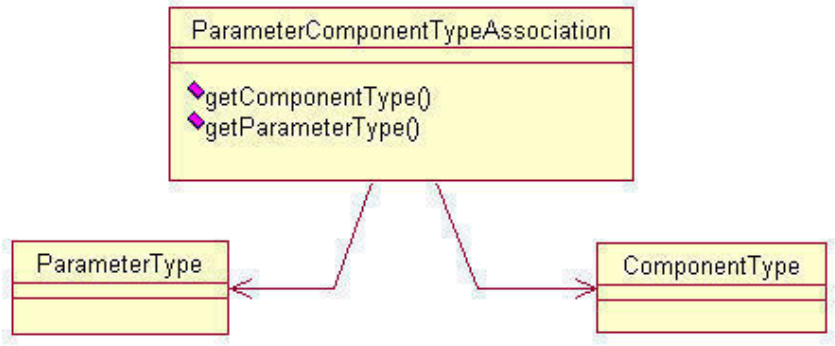

Figura 3-7 Diagrama de classes do pacote de tipos – *Types*.

Figura 3-8 Diagrama de classes do pacote de restrições – *Constraints*.

Como pode ser visto na Figura 3-9, a classe que gerencia o módulo *Creator* (classe *Creator*) mantêm uma lista de parâmetros cadastrados pelo Usuário Criador (objetos da classe *Parameter*), além de interagir com três outras classes: **(i)** *NewParameterGUI*, responsável por gerenciar a inserção de novos parâmetros; **(ii)** *XML_Writer*, responsável pela compilação dos

dados para a representação abstrata da interface; e **(iii)** *XML_Reader*, classe responsável por recuperar os dados dos parâmetros, para uma posterior alteração destes dados pelo *Usuário Criador*. A classe *Parameter* é uma generalização para todos os parâmetros existentes na ferramenta. Esta classe possui um *ParameterListenerSupport* para poder informar à classe *Creator* a modificação do nome de algum parâmetro existente. Um parâmetro possui um único tipo (*ParameterType*), e pode possuir uma lista de restrições (*Constraint*).

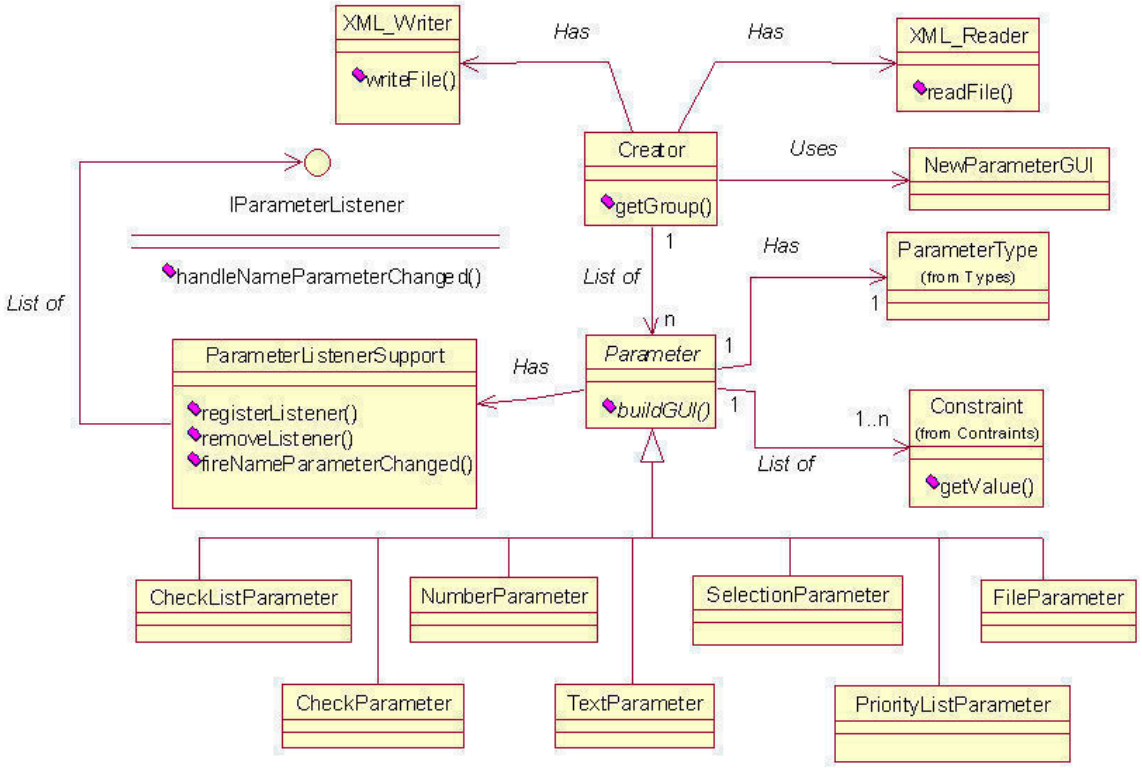

Figura 3-9 Diagrama de classes do pacote responsável pelo módulo *Creator* – *Creator*.

A Figura 3-10 representa o diagrama de classes do pacote responsável pelo módulo *Builder* – pacote *Builder*. A classe responsável por gerenciar toda a geração da GUI é a classe *Builder*. Esta classe é responsável, além da montagem da GUI, por realizar a leitura e interpretação dos dados presentes na representação abstrata e gerenciar os componentes (classe *Component*) presentes nesta GUI gerada.

A classe *Component* é uma generalização para todos os componentes possíveis de serem utilizados na montagem da GUI. Esta classe possui um único tipo (*ComponentType*) e pode possuir uma lista de restrições (*Constraint*). No momento em que existe uma alteração do valor de um componente, o objeto desta classe dispara um evento (*ComponentEvent*) e quem estiver registrado como observador deste objeto será notificado.

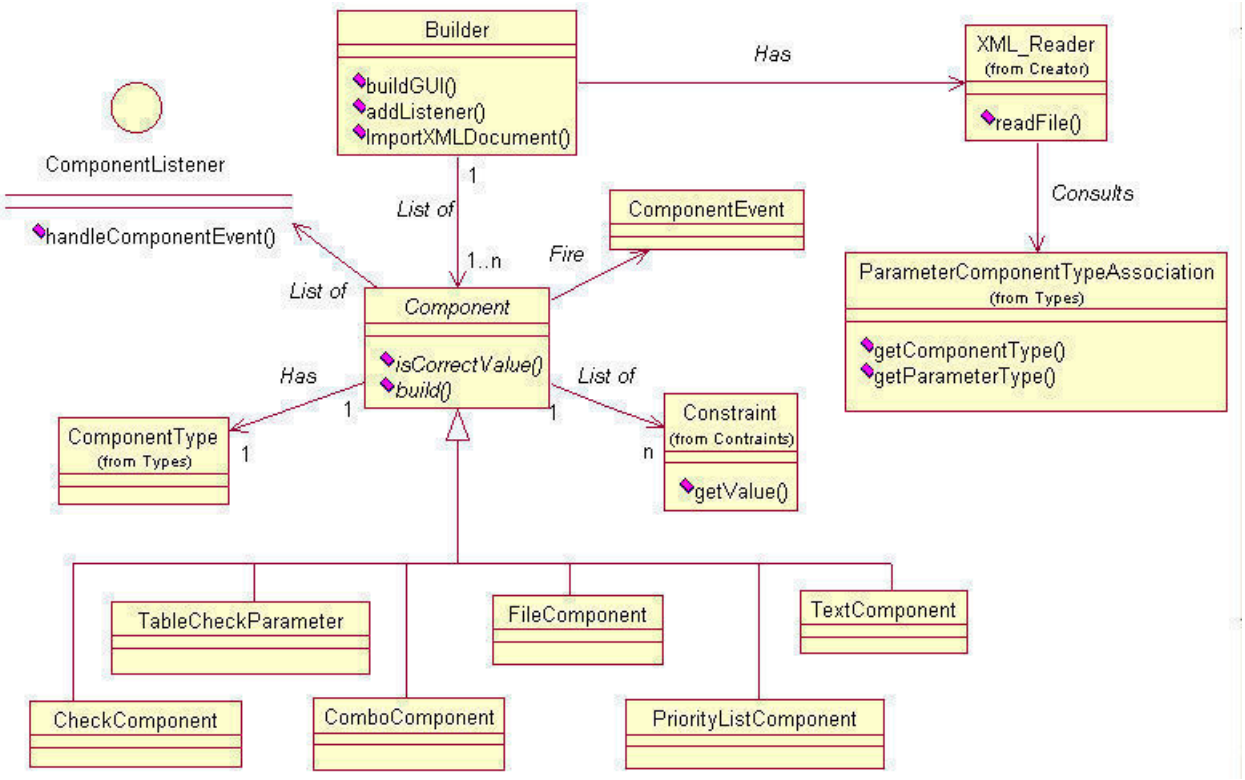

Figura 3-10 Diagrama de classes responsável pelo módulo *Builder – Builder*.

O correto planejamento e definição da arquitetura da ferramenta irá facilitar o desenvolvimento de um software que atenda a todos os requisitos catalogados, além de ser uma ferramenta que facilite a sua futura manutenção [21].

3.3.2 Primeira fase do processo – Módulo *Creator*

A primeira fase do processo de geração automática da GUI proposto neste trabalho é implementado pelo módulo *Creator*. Este módulo é responsável por recolher informações do *Usuário Criador* sobre quais dados devem ser mostrados na GUI a ser gerada.

Este módulo é integrado ao Gerador / Configurador de Simuladores do ambiente MPhyScaS, sendo ativo no momento em que o usuário deseja cadastrar um novo componente da categoria Fenômeno. Neste momento o Gerador / Configurador de Simuladores realiza uma chamada ao método *getGroup(Composite parent)* de um objeto da classe *Creator*. Este método tem como parâmetro de entrada um SWT *Composite* e tem como saída um SWT *Group* representado a tela onde o Usuário Criador entrará com os dados referentes aos parâmetros do Fenômeno a ser criado, como ilustra a Figura 3-11. Esta tela contem as opções gerais da definição dos parâmetros de entrada de um Fenômeno. Opções como a inserção de um novo parâmetro, remoção de um parâmetro já existente, seleção de um parâmetro para alterar as suas definições e a exibição de todos os parâmetros já existentes em uma tabela, mostrando seu código, tipo e nome.

No momento em que o usuário deseja inserir um novo parâmetro no Fenômeno, este seleciona a opção "*New parameter*", sendo então exibida uma nova janela pelo sistema solicitando que indique o tipo do parâmetro a ser criado e seu nome (Figura 3-12), ressaltando

que este nome não é obrigatoriamente o nome final do parâmetro, o sistema possibilita que, caso deseje, este nome seja alterado futuramente, durante a definição das características deste parâmetro.

Figura 3-11 SWT *Group* retornado pela chamada ao método *getGroup(Composite parent)* do módulo *Creator*.

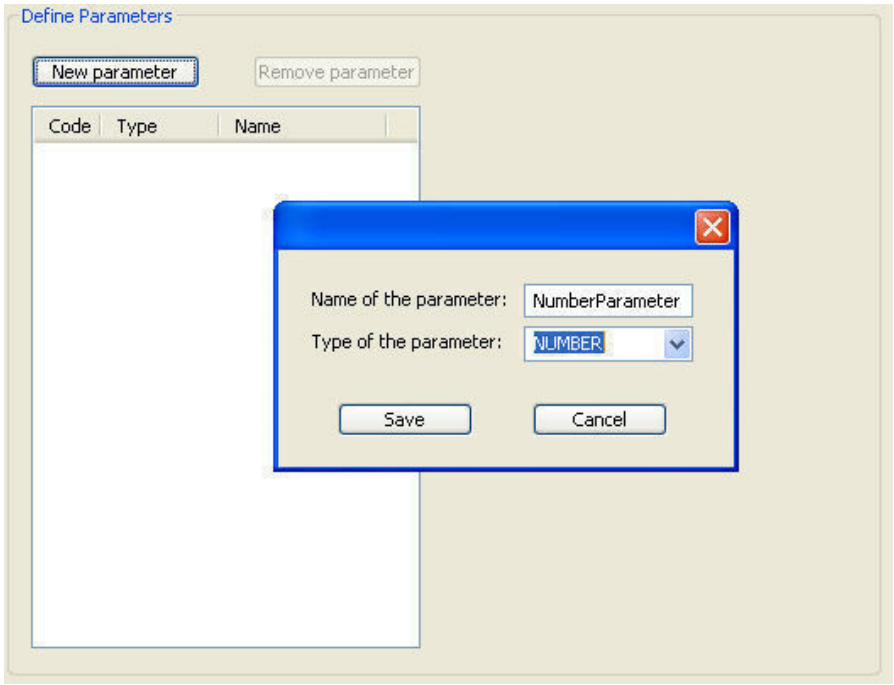

Figura 3-12 Inserção de um novo parâmetro, solicitando ao usuário que indique o nome e o tipo do parâmetro.

A qualquer momento, enquanto este módulo estiver ativo, o *Usuário Criador* pode selecionar um parâmetro criado e verificar suas características, como nome, descrição, valor inicial e suas restrições. Um exemplo de como estas características são mostradas para o usuário esta representada na Figura 3-13, onde mostra as características de um parâmetro numérico.

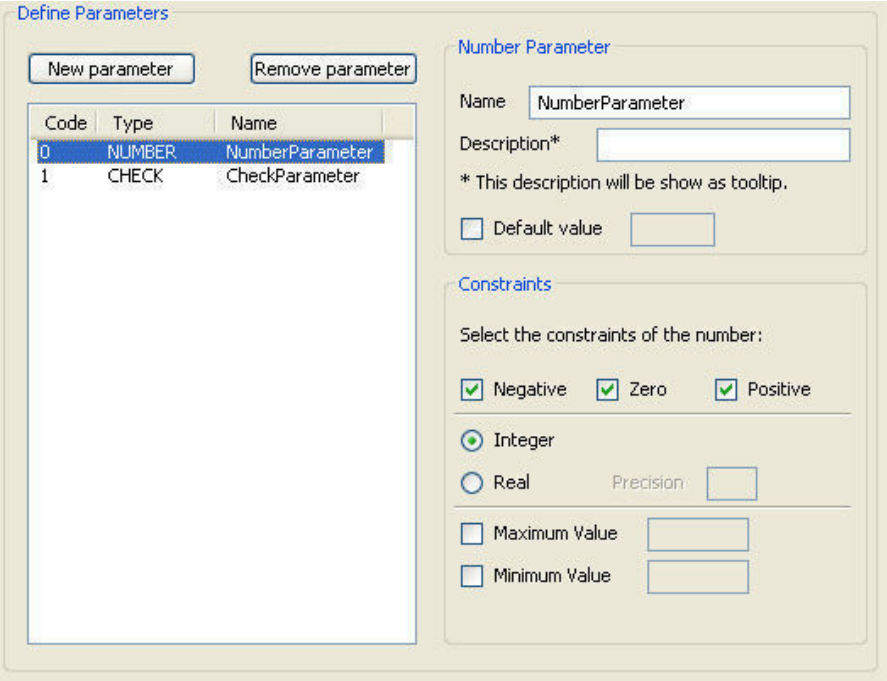

Figura 3-13 Exemplo de como são exibidas as características de um parâmetro para o usuário. Nesta imagem, estão representadas as características de um parâmetro numérico.

Características como nome, descrição e valor inicial estão presentes em todos os parâmetros, podendo ser alterado enquanto o módulo *Creator* estiver ativo. A descrição presente nos parâmetros é uma característica opcional, com o intuito de facilitar o entendimento pelo o *Usuário Final* quando for visualizar a GUI gerada. Esta descrição será apresentada como dica, ou seja, no momento em que o *Usuário Final* deixar o cursor do mouse sobre o parâmetro irá aparecer esta pequena descrição, caso esta exista. Já a parte de restrições nem todos os parâmetros possuem e cada um possui suas próprias restrições.

3.3.3 Segunda fase do processo – Representação abstrata

Após a completa definição das características de todos os parâmetros de um novo componente, este componente ficará disponibilizado no Repositório de Componentes do ambiente MPhyScaS para ser utilizado na geração de um simulador.

No momento em que o usuário define os componentes de software que farão parte de um simulador e solicita que o este seja gerado, o Gerador / Configurador de Simuladores gera um arquivo de configuração contendo as especificações de todos os componentes presentes neste simulador. Neste arquivo, estará presente também a definição das interfaces que deverão ser geradas.

A ferramenta proposta está preparada tanto para gerar o arquivo de definição para ser integrado em algum outro arquivo de definições, como no caso do MPhyScaS, ou gerar um

arquivo de definições com apenas os dados da ferramenta. No caso de ser utilizado para gerar apenas a estrutura do documento XML para ser integrado a um outro arquivo, contendo mais definições, o caso do MPhyScaS, a ferramenta gera toda a estrutura da definição da GUI em um *JDOM Element.*

A estrutura do documento XML gerado para a representação abstrata da interface a ser gerada pode ser vista no exemplo da Figura 3-14. Neste exemplo, está a representação de uma interface que contem três parâmetros: **(i)** um parâmetro numérico (linhas 2 a 7), contendo as restrições que permitirão que o usuário insira um valor inteiro (linha 5) positivo e não-nulo (linha 4) e com o valor inicial *5* (linha 2); **(ii)** um parâmetro do tipo arquivo (linhas 8 a 10), possibilitando apenas a leitura de arquivos do formato *Microsoft Word* (linha 8); e **(iii)** um parâmetro de seleção (linhas 11 a 16), necessitando que o usuário escolha apenas um item da lista que contem os valores: *v1, v2 e v3* (linhas 13 e 14).

```
1:  <parameters>
2:     <param id="0" name="Number Parameter" type="NUMBER"
           default="5" description="">
3:        <constraints>
4:           <positiveConstraint></positiveConstraint>
5:           <integerConstraint></integerConstraint>
6:        </constraints>
7:     </param>
8:     <param id="1" name="FileParameter" type="FILE" default="MS
           Word $ *.doc" description="">
9:        <constraints></constraints>
10:    </param>
11:    <param id="2" name="SelectionParameter" type="SELECTION"
           default="" description="">
12:       <constraints>
13:          <selectionConstraint value="v1 # v2 # v3">
14:          </selectionConstraint>
15:       </constraints>
16:    </param>
17:</parameters>
```

Figura 3-14 Exemplo de representação abstrata de uma GUI a ser gerada pela ferramenta.

Com esta representação abstrata em um arquivo XML, este pode ser levado para um outro ambiente que contenha apenas o módulo *Builder* da ferramenta proposta, e a GUI será gerada contendo todas as descrições inseridas pelo *Usuário Criador*.

3.3.4 Terceira fase do processo – Módulo *Builder*

A fase final do processo de geração automática de interface gráfica consiste na leitura e interpretação da descrição abstrata e a montagem desta interface. Esta ultima etapa é realizada pelo módulo *Builder*. A GUI gerada pelo módulo *Builder* do exemplo da Figura 3-14 está ilustrado na Figura 3-15.

O módulo *Builder* é responsável de garantir que o Usuário Final insira apenas valores dentro das restrições inseridas pelo Usuário Criador. Desta forma, caso o Usuário Final insira um valor incorreto, a interface gerada mostra uma mensagem de erro para notificar o usuário do valor incorreto (Figura 3-16).

Outra responsabilidade do módulo *Builder* é de notificar o sistema ao qual está integrado, para informar que o valor de algum parâmetro foi alterado e que o sistema atualize o seu valor correspondente a este parâmetro.

Figura 3-15 GUI gerada pelo módulo Builder do exemplo da Figura 3-14.

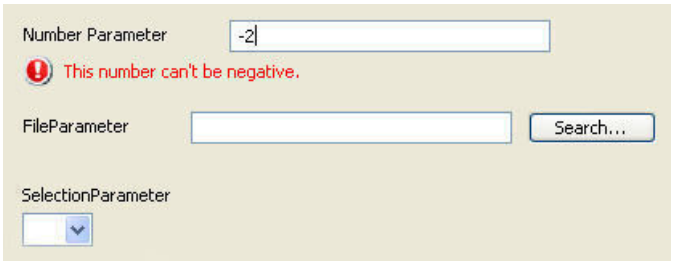

Figura 3-16 Inserção de valor incorreto pelo Usuário Final e notificação pelo módulo Builder a este usuário.

Este módulo é integrado aos simuladores gerados pelo ambiente MPhyScaS que contenham um componente que necessite que a sua GUI seja gerada automaticamente pela ferramenta. Para realizar a geração automática basta instânciar um objeto da classe *Builder* e solicitar a leitura da representação abstrata, ou através de um arquivo XML que contenha apenas esta descrição ou passando como parâmetro para a leitura o *JDOM Element* raiz da estrutura que contenha esta representação.

Após a leitura da representação abstrata, o sistema pode solicitar que a GUI seja gerada com a chamada ao método *buildGUI(Composite parent)* ou *buildGUI(Composite parent, String name)*, em ambos os métodos é passado como parâmetro de entrada um *SWT Composite* onde a GUI será montada, mas no primeiro método o retorno é também um *SWT Composite* enquanto no segundo que necessita de uma *String* como parâmetro, é retornado um *SWT Group* tendo como título do grupo a *String* passada. Em ambos os casos, o retorno contem toda a GUI gerada.

Para ser notificado da alteração do valor dos parâmetros presentes na GUI gerada, o sistema necessita se registrar como observador do módulo *Builder*, realizando a chamada ao método *addListener(IComponentListener listener)*. Este método tem como parâmetro de entrada a implementação de um *IComponentListener*, indicando o procedimento que o sistema deve executar no momento em que o parâmetro sofrer uma alteração.

No momento em que ocorre uma alteração de um parâmetro da GUI gerada, um evento é gerado e todos os observadores desta instância do módulo *Builder* são notificados. Este evento possui encapsulado o objeto que gerou o evento, desta forma o observador quando notificado, pode verificar se necessita executar algum procedimento quando este parâmetro for alterado.

Capítulo 4

Estudos de Caso

Para exemplificar a ferramenta desenvolvida, são realizados dois estudos de caso. Em cada estudo é levado em consideração um componente Fenômeno distinto, que pode ser cadastrado no Repositório de Componentes do ambiente MPhyScaS. No primeiro estudo, é analisado como seria a GUI necessária para um Fenômeno Elástico Isotrópico que atua em uma geometria de duas dimensões, já no segundo estudo, é analisado um Fenômeno de Difusão que também atua em uma geometria de duas dimensões.

4.1 Fenômeno Elástico Isotrópico – 2D

A elasticidade em um material seria a capacidade deste sofrer deformações ao ser submetido a forças externas ou internas (por exemplo, o próprio peso), mas retornam a sua forma original quando a tensão é removida [22].

Esta propriedade dos materiais, pode ser dividida em três categorias: **(i)** material elástico isotrópico, onde o material se comporta igualmente em todas as três dimensões; **(ii)** material elástico ortotrópico, materiais que se comportam de forma igual em duas de suas dimensões e tendo a terceira dimensão se comportando de maneira diferente das outras; e **(iii)** anisotrópico, sendo materiais que se comporta diferentemente em cada uma das suas três dimensões [22].

Existe no MPhyScaS, um componente de software que atua sobre a geometria correspondente, fazendo com que este material se comporte como um material elástico. Para o estudo de caso aqui presente, o Fenômeno levado em consideração atua sobre uma geometria de duas dimensões (por exemplo: um plano) de forma que esta geometria se comporte como um material elástico isotrópico.

Para a correta utilização deste Fenômeno, é necessário a inserção de diversos dados para sua configuração. Como a quantidade de dados necessários é extensa, será explicada apenas uma parte destes dados, considerados como parâmetros constitutivos e dados referentes ao material, como: coeficiente de dilatação térmica, densidade e temperatura de referência, além do método de Carregamento de Corpo.

4.1.1 Geração da GUI

Após o usuário responsável pelo desenvolvimento do componente de software, capaz de atuar como um fenômeno elástico isotrópico, terminar toda a implementação deste componente seguindo os padrões do ambiente MPhyScaS, será necessário agora que o usuário cadastre este componente no Repositório de Componentes do MPhyScaS.

No momento do cadastro, o usuário necessitará informar quais os parâmetros que o usuário final deverá inserir quando for utilizar este novo Fenômeno. A inserção destes dados é realizada interagindo com o módulo *Creator* da ferramenta proposta nesta monografia.

Desta forma, é necessário que o usuário criador do Fenômeno, detalhe cada parâmetro necessário na configuração deste Fenômeno. Estes parâmetros são: **(i)** parâmetros constitutivos, composto por dois valores (*E* – módulo de Elasticidade; e *ni* – módulo de Poisson); **(ii)** coeficiente de dilatação térmica (ρ - *Rho*); **(iii)** densidade (α - *Alpha*); **(iv)** temperatura de referência (T_0); e **(v)** método de Carregamento de Corpo (*Body Load*). Com exceção do parâmetro (v), que seria a entrada de um módulo DLL (*Dynamic-Link Library*) que implementa o método desejado e o nome do método que deverá ser executado desta biblioteca, todos os outros parâmetros são compostos por números reais com precisão de cinco casas decimais.

Na inserção destes dados no módulo *Creator* são inseridos cinco parâmetros numéricos, um para a entrada de texto e um para a entrada de um arquivo contendo o filtro permitindo apenas arquivos com a extensão: **.dll*. O resultado da inserção destes dados no módulo *Creator* pode ser visto na Figura 4-1.

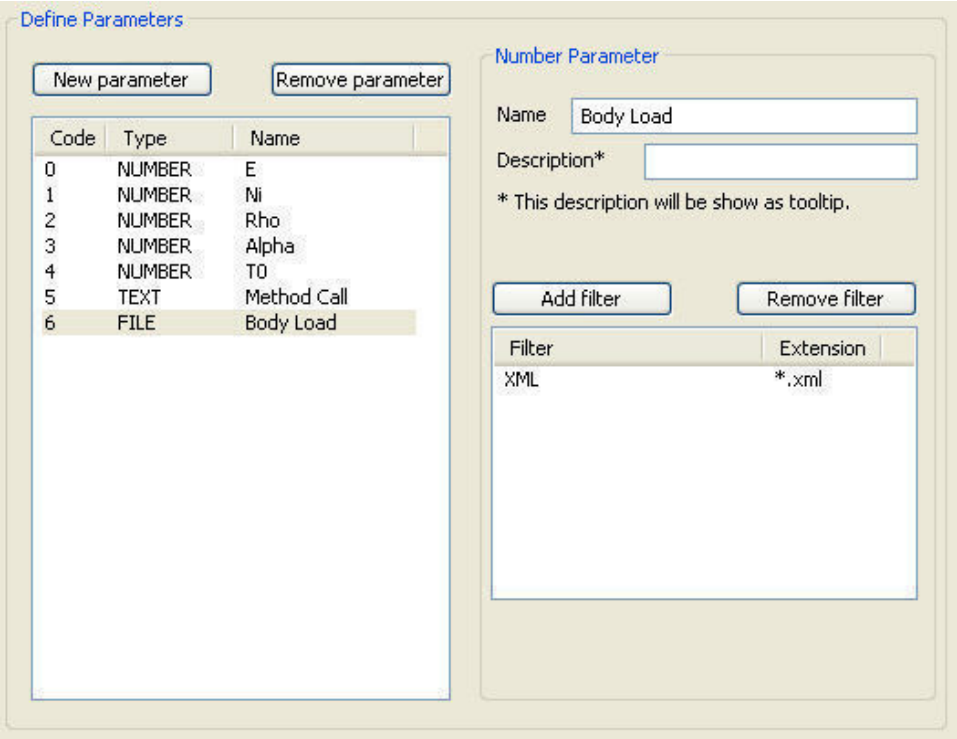

Figura 4-1 Cadastro dos parâmetros de um Fenômeno Elástico Isotrópico que atua em uma geometria com duas dimensões.

Com este cadastro realizado, o novo Fenômeno está disponível para ser utilizado como parte integrante de um simulador. Quando for executado um simulador que contenha este

Fenômeno, o segundo módulo da ferramenta é ativado, módulo *Builder*, realizando a montagem da GUI deste Fenômeno. O resultado da montagem desta GUI pode ser visualizado na Figura 4-2.

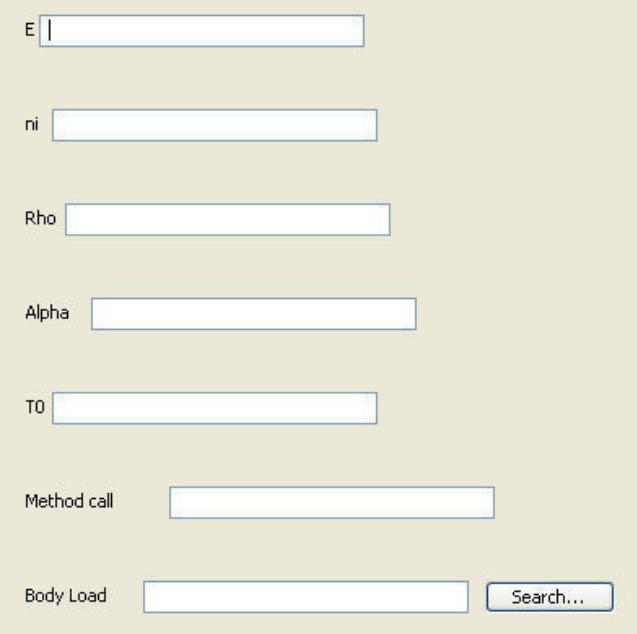

Figura 4-2 GUI, definida na Figura 4-1, do Fenômeno Elástico Isotrópico gerada pela ferramenta.

Esta GUI gerada irá obedecer às restrições impostas pelo *Usuário Criador* da mesma. Por exemplo, obrigará ao *Usuário Final* inserir um número real com precisão de cinco casas decimais, caso o usuário insira um número com menor precisão, a ferramenta completará este valor e caso este número tenha mais casas decimais do que o permitido pelo *Usuário Criador*, a ferramenta irá truncar o número para permanecer de acordo com a restrição estabelecida. E para a entrada de arquivo, só está disponível a visualização de arquivos do tipo XML.

4.2 Fenômeno de Difusão – 2D

O processo de difusão em uma material consiste no movimento de partículas de uma área de maior concentração para uma área de menor concentração, buscando o equilíbrio [23]. A difusão não ocorre unicamente com partículas, pode ocorrer também em todos os fenômenos físicos de termodinâmica sob a influência de variações térmicas [23].

Existe no MphyScaS um componente de software capaz de modelar o processo de difusão em um material (geometria). Neste estudo de caso será analisado o processo de geração de GUI para um Fenômeno de Difusão que atua em duas dimensões.

Como no estudo de caso da seção 4.1, a correta configuração deste componente para a sua utilização é bastante complexa e extensa, por isso será utilizado uma quantidade diminuta de dados. Serão levados em consideração apenas os dados de Capacitância Específica, Difusividade e também será necessário para o usuário escolher qual o método de estimativa de erro em que deseja avaliar o resultado deste Fenômeno.

4.2.1 Geração da GUI

Após o correto desenvolvimento do componente de software capaz de modelar um fenômeno de difusão, este componente deve ser cadastrado no MphyScaS. Como no estudo de caso da seção 4.1, o cadastro dos parâmetros necessários para a sua configuração é realizado utilizando a ferramenta proposta.

Levando em consideração apenas uma parte dos dados necessários, para não tornar o estudo de caso extenso, teríamos a inserção de três parâmetros, estes seriam: **(i)** Capacitância Específica (*Cp*); **(ii)** Difusividade (*K11, K12, K21* e *K22*); e **(iii)** a escolha de qual o método de estimativa de erro (*Error Estimate Approach*) em que deseja avaliar o resultado deste Fenômeno (Discretização Espacial – *Spatial Discretization*; Discretização Temporal – *Temporal Discretization*; Ordem de Aproximação Polinomial - *Polynomial Order of Approach*; ou Modelo – *Model*).

A definição destes parâmetros pode ser vista na Figura 4-3, onde pode ser notado que com exceção do método de estimativa de erro, onde teríamos um parâmetro de seleção, todos os outros parâmetros são numéricos.

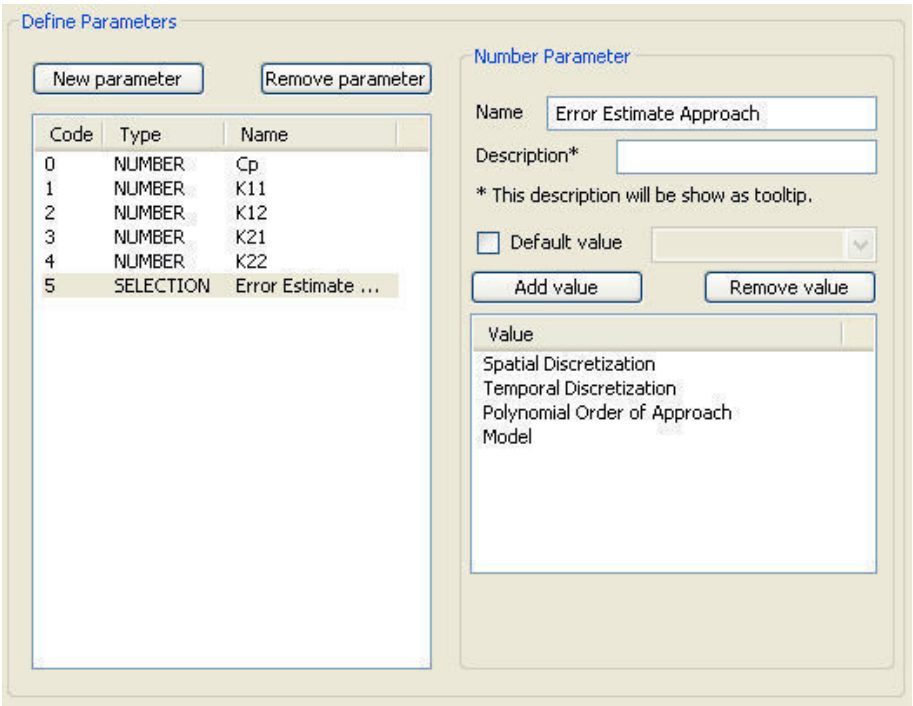

Figura 4-3 Cadastro dos parâmetros de um Fenômeno Difusão que atua em uma geometria com duas dimensões.

Com esta correta definição dos parâmetros do Fenômeno Difusão, este pode ser então utilizado em um Simulador do MphyScaS. Este componente terá sua interface gerada automaticamente, com o auxilio da ferramenta desenvolvida nesta monografia, tendo como resultado a Figura 4-4.

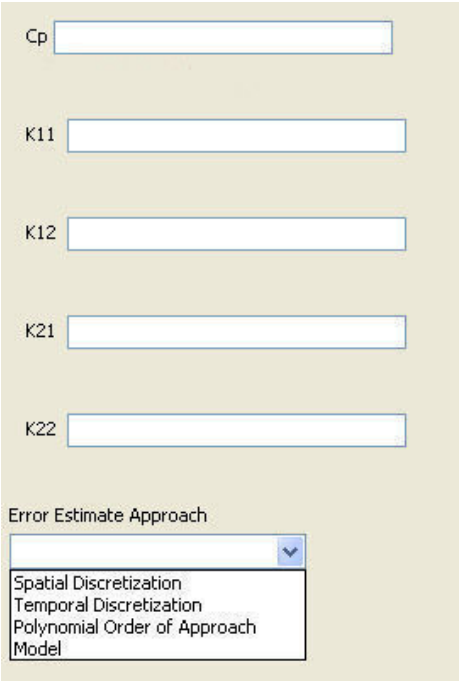

Figura 4-4 GUI, definida na Figura 4-3, do Fenômeno Difusão gerada pela ferramenta.

Assim como a GUI gerada para o estudo de caso 4.1, esta GUI também irá impor restrições para o *Usuário Final* que irá utilizá-la, exigindo que os dados seja inseridos corretamente como definido pelo *Usuário Criador*.

4.3 Resultados

Como pode ser visto nos estudos de caso 4.1 e 4.2, onde são levados em consideração dois componentes de software da categoria Fenômeno, que apesar de serem de uma mesma categoria, necessitam de dados de entrada diferentes. Mesmo levando em consideração uma pequena parte dos dados necessários para a correta configuração de ambos os Fenômenos, a GUI necessária é completamente distinta.

Desta forma, pode ser analisado o quanto do tempo de desenvolvimento seria gasto realizando a implementação desta interface gráfica. Sendo isso o grande objetivo da ferramenta proposta, abstrair o desenvolvimento da GUI do restante da implementação do componente.

Para esta monografia não foi levado em consideração o design da GUI gerada, apesar de ser de extrema importância para o planejamento de uma GUI eficiente [24], assim a GUI gerada não foi projetada por um especialista em design amigável, mas a ferramenta realiza bem o seu papel com relação a diminuir o esforço por parte do desenvolvedor dos componentes. De fato, há uma diminuição de esforço até mesmo em relação a restrições que os parâmetros devam obedecer.

Desta forma, a ferramenta se mostra útil no que esta se propõe, que seria facilitar o cadastro de novos componentes no ambiente MphyScaS.

Capítulo 5

Conclusões e Trabalhos Futuros

O desenvolvimento deste trabalho resultou em uma ferramenta capaz de gerar interfaces gráficas com o usuário a partir da definição das tarefas que o usuário deverá realizar nesta GUI gerada. Especificamente, este trabalho é abordado no ambiente de geração de Simuladores MPhyScaS.

O ambiente MPhyScaS pode ser considerado uma linha de produto de software, onde o seu produto final é gerado a partir da seleção de um conjunto de componentes de software. Estes componentes podem ser divididos em categorias e com isto, componentes de uma mesma categoria geralmente apresentam um mesmo padrão no tipo e na quantidade dos dados de entrada, mas existe uma categoria de componentes, denominada Fenômeno, em que cada componente pode requisitar uma quantidade de dados de entrada diferente, como o tipo destes, também pode ser diferente.

A ferramenta desenvolvida é capaz de agilizar a implementação de novos Fenômenos para serem integrados ao ambiente MPhyScaS, pois abstrai o desenvolvimento da GUI para a configuração destes, necessitando apenas que o desenvolvedor deste novo Fenômeno detalhe como são os parâmetros necessários que estarão presentes na GUI a ser gerada. A ferramenta fica então responsável por compilar os dados informados pelo usuário da ferramenta, gerando uma representação abstrata da GUI, e após este processo, realizar a interpretação desta representação e fazer a montagem desta.

A implementação orientada a objeto da ferramenta facilitará uma possível expansão desta, com a inserção de novos tipos de parâmetros e também o fato de poder integrá-la não somente ao ambiente MPhyScaS, mas em qualquer outro sistema que necessite de uma geração automática de interface.

5.1 Contribuições

As principais contribuições deste trabalho estão listadas a seguir:

- A ferramenta abstrai o desenvolvimento da GUI de novos Fenômenos desenvolvidos para o ambiente MPhyScaS, agilizando o processo de criação destes novos componentes.

- A GUI gerada implementa restrições aos parâmetros cadastrados, não necessitando de uma programação adicional ao desenvolvedor do Fenômeno.

- Incentiva a adoção do ambiente MphyScaS para o desenvolvimento de simuladores, fornecendo maior suporte para o desenvolvimento e reutilização de componentes.

5.2 Trabalhos Futuros

Os trabalhos realizados durante este projeto dão margens a melhorias na ferramenta, seja na estrutura, fazendo com que a inserção de novos tipos de parâmetros não necessitem de uma grande reprogramação, como melhorias em sua integração ao MPhyScaS, possibilitando que a ferramenta gere a interface de outros componentes ou até mesmo toda a interface do simulador gerado pelo ambiente.

Ainda existe a possibilidade de realizar a organização dos componentes gráficos presentes na GUI gerada, visando um melhor design. Podendo realizar tal tarefa de forma automática, ou à vontade do Usuário Criador, solicitando que indique como os componentes devem aparecer na GUI. Outra melhoria possível referente ao design da GUI a ser gerada, é a possibilidade de usar alguma ferramenta visual de edição de componentes JavaBeans, aproveitando que este foi o padrão de componentização utilizado.

Outra melhoria capaz de ser realizada é a generalização desta ferramenta, permitindo sua integração em diversos outros sistemas que possuam a mesma dificuldade que possui o ambiente MPhyScaS, a existência de componentes com quantidade e tipos de parâmetros de entrada variados.

Para a realização destas melhorias seriam necessários novos estudos de caso, possibilitando uma melhor validação da ferramenta e desta forma evoluir a sua implementação.

Bibliografia

[1] MYERS, B.A.; ROSSON, M.B. *Survey on user interface programming*. In: Proceedings: CHI 1992. May 3-7, Monterey, CA.

[2] SANTOS, F. C. G. et al. *Transfer and sharing of data between coupled multi-physics phenomena during simulations: the Phenomenon Computational Pattern*, in World Conference on Computer Science – International Conference on Modeling, Simulation & Visualization Methods, pp. 49-55, CSREA Press, Las Vegas, Estados Unidos, 26-29 de Junho de 2006.

[3] BOWDEN, R. O., BATEMAN, R., GOGG, T. G. *Simulação Otimizando os Sistemas.* 1ª edição. Ed. São Paulo: IMAM, 2005. 142 p.

[4] PRESSMAN, R. S. *Engenharia de Software*. Makron Books, 1995, 1056p.

[5] JACOBSON, I., GRISS, M., JONSSON, P. *Software Reuse: Architecture, Process and Organization for Business Success*. 1ª edição. Ed. Boston, USA: Addison-Wesley Professional, 1997. 497p.

[6] LOGAN, D. L. *A First Course in the Finite Element Method.* Thomson-Engineering: 4 edition, 2006, 832p.

[7] LENCASTRE, M. *The Conceptualization of an Environment for the Development of FEM Simulators*. 2004. Tese de Doutorado, Universidade Federal de Pernambuco, UFPE, 2004.

[8] SANTOS, F. C. G., BARBOSA, J. M. A., BRITO Jr., E. R. R. *Simulação do problema de evolução do dano em barras elastoviscoplásticas empregando o Grafo de Interface Genérica (GIG).* In: III Congresso Nacional de Engenharia Mecânica, 2004, Belém, Brasil.

[9] CLEMENTS, P., NORTHROP, L. *Software Product Lines: Practices and Patterns.* Addison Wesley Professional, 2001, 608p.

[10] MCGREGOR, J. D. *Software Product Lines*, in Journal of Object Technology, vol. 3 no. 3, Março-Abril, 2004, pp. 65–74.

[11] KRUEGER, C. W. *Introduction to the Emerging Practice of Software Product Line Development.* Methods & Tools, Outono 2006.

[12] *SWT: The Standard Widget Toolkit*. Disponível em <http://www.eclipse.org/swt/>. Acesso em 24 de maio de 2007.

[13] TANENBAUM, A. S. *Sistemas Operacionais Modernos*. Prentice Hall, 2003, 695p.

[14] *World Wide Web Consortium XML*. Disponível em <http://www.w3.org/XML>. Acesso em 24 de maio de 2007.

[15] *JDOM*. Disponível em <http://www.jdom.org>. Acesso em 24 de maio de 2007.

[16] *Application Development Trends Articles: Can Borland ride the life cycle?* Disponível em <http://www.adtmag.com/article.aspx?id=7732>. Acesso em 24 de maio de 2007.

[17] *Sun Developer Network: JavaBeans*. Disponível em <http://www.sun.com/beans>. Acesso em 24 de maio de 2007.

[18] SOUTO, L. M. *Especificação e Implementação de Componentes Para Modelar Redes Locais sem Fio Ad Hoc Padrão IEEE 802.11*. Tese de Mestrado, Universidade Federal de Campina Grande, UFCG, 2005, 81p.

[19] ARAÚJO, J. A. ELIAS, G. *Um Ambiente de Execução de Componentes JavaBeans.*

[20] LIBÓRIO, A. et al. *Geração de interfaces através de sistemas baseados em conhecimento: uma abordagem centrada em explicações de modelos de resolução de problemas.* In: In Proceedings of the Latin American conference on Human-computer interaction, vol.1. New York: ACM Press, 2005.

[21] BASS, L., CLEMENTS, P., KAZMAN, R. *Software Architecture in Practice.* Second Edition. Addison Wesley, 1998, 560p

[22] OGDEN, W. E. *Non-Linear Elastic Deformations*. New Ed edition, 1997, 544p.

[23] CUSSLER, E. L. *Diffusion: Mass Transfer in Fluid Systems. Second Edition*. Cambridge University Press, 1997, 580p.

[24] GALITZ, W. O. *The essential guide to user interface design: an introduction to GUI design principles and techniques.* Ed. Wiley Computer Publishing, 1996, 626p.

Anexo A

Documento de Requisitos

Neste anexo é apresentado detalhadamente, os requisitos catalogados na fase de pré-desenvolvimento da ferramenta proposta.

Documento de Requisitos

MPhyScaS GUI Generator

Versão 0.2 | 27/04/2007

Responsável: Fernando Rocha

Histórico de Alterações

Data	Versão	Descrição	Autor
27/04/2007	0.2	Realização das correções solicitadas.	Fernando Rocha
24/04/2007	0.1	Versão inicial do documento	Fernando Rocha

Conteúdo

1. Introdução

Este documento define e especifica o sistema *MPhyScaS GUI Creator* a ser desenvolvido. Seu propósito é colecionar, analisar e definir as necessidades do cliente e as características de alto nível que o sistema deve prover, focando nos requisitos técnicos identificados e no motivo destes existirem.

1.1 Visão geral deste documento

Esta introdução fornece as informações necessárias para fazer um bom uso deste documento, explicitando seus objetivos e as convenções que foram adotadas no texto. As demais seções apresentam a especificação do sistema *MPhyScaS GUI Creator* e estão organizadas como descrito abaixo.

- **Seção 2 – Descrição Geral do Produto**: descreve, de forma geral, o produto.
- **Seção 3 – Requisitos Funcionais**: lista os requisitos funcionais do sistema, especificando seus objetivos e prioridades.
- **Seção 4 – Requisitos Não Funcionais**: especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de hardware e software.
- **Seção 5 – Escopo Negativo:** especifica as funcionalidades que estão relacionadas com o sistema, mas que não fazem parte do escopo do projeto e, portanto, não serão implementadas. Convenções, termos e abreviações

Esta seção explica o conceito de alguns termos importantes que serão mencionados no decorrer deste documento. Estes termos são descritos na tabela a seguir, estando apresentados por ordem alfabética.

Termo	Descrição
Requisitos funcionais	Requisitos técnicos do software que compõe o sistema, que descrevem ações que o sistema deve estar apto a executar, ou seja, o que o sistema deve fazer.
Requisitos não funcionais	Requisitos técnicos do software que compõe o sistema, que descrevem atributos que o sistema deve possuir ou restrições sob as quais ele deve operar.
Requisitos não técnicos	Requisitos não relacionados ao software. Requisitos não técnicos estão fora do escopo deste documento, devendo, se necessário, serem incluídos apenas no Plano do Projeto.
Módulo creator	Sistema usuário responsável por controlar a definição dos parâmetros que aparecerão na GUI gerada posteriormente.
Módulo builder	Sistema usuário responsável por gerar automaticamente a GUI e realizar a comunicação com o outro sistema no qual foi integrado.
Usuário criador	Usuário que utiliza a ferramenta para o cadastro dos parâmetros presentes na interface gerada.
Usuário final	Usuário que utiliza a interface gráfica gerada pela ferramenta como produto final.

1.1.1 Prioridades dos requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”, “importante” e “desejável”. A prioridade dos requisitos é utilizada no gerenciamento do escopo das etapas do projeto e na definição das prioridades durante o desenvolvimento do sistema.

- **Essencial:** requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, devem ser implementados desde as primeiras implantações do sistema.
- **Importante:** requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implantados o mais rápido possível, mas, se não forem, parte do sistema poderá ser implantada mesmo assim.
- **Desejável:** requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser implantados por último, sem comprometer o funcionamento do sistema.

2. Visão Geral do Produto

O *MPhyScaS GUI Creator* consiste em uma ferramenta para a geração automática de Interface Gráfica para o Usuário (GUI) para ser integrada ao ambiente MPhyScaS. O ambiente MPhyScaS (*Multi Physics and Multi Scales Solver Enviroment*) é um projeto em desenvolvimento pelo DSC – UPE em parceria com o DEMEC – UFPE, com a finalidade de gerar e/ou configurar simuladores multi-físicos.

O ambiente MPhyScaS pode ser definido como uma linha de produtos, onde a sua saída seria os simuladores gerados. Desta forma, foi identificada uma dificuldade no desenvolvimento desta linha de produtos, já que nem todos os simuladores possuem a mesma quantidade de dados de entrada. A ferramenta propõe uma solução capaz gerar automaticamente a GUI para a entrada de dados, atendendo assim a todos os simuladores gerados.

3. Requisitos Funcionais

A ferramenta é composta por dois módulos, *Creator* e *Builder*. O módulo *Creator* é utilizado para o cadastro dos dados (parâmetros) que serão inseridos pelo usuário final em um simulador gerado pelo MPhyScaS. Já o módulo *Builder* é o responsável por gerar a interface gráfica dos parâmetros cadastrados.

3.1 Requisitos do módulo '*Creator*'

[RF001] Cadastrar parâmetros

Este caso de uso permite o cadastro de parâmetros para os quais será gerada uma interface gráfica, esta interface será gerada automaticamente no módulo *Builder*.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF002] Excluir parâmetro

Este caso de uso permite a exclusão de parâmetros já existentes.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF003] Editar parâmetro

Este caso de uso permite a modificação dos dados de parâmetros inseridos.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF004] Definir restrições aos parâmetros

Este caso de uso permite a definição de restrições aos parâmetros inseridos, como por exemplo, definir que um parâmetro do tipo numérico aceite apenas valores positivos.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF005] Gerar arquivo de descrição

Este caso de uso permite exportar os dados dos parâmetros para um arquivo. Possibilitando transportar este arquivo para o módulo '*Builder*'.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

3.2 Requisitos do módulo '*Builder*'

[RF001] Gerar automaticamente a GUI para os parâmetros cadastrados

Este caso de uso permite que a ferramenta gere automaticamente a interface gráfica dos parâmetros cadastrados no módulo '*Creator*' *[RF001]* .

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF002] Prover um mecanismo de notificação

Este caso de uso permite que a ferramenta gere eventos para o sistema no qual está integrado *[NF005]*, possibilitando assim uma comunicação, informando o estado dos parâmetros mostrados na GUI.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF003] Implementar restrições aos parâmetros

Este caso de uso permite que a GUI gerada restrinja os tipos de parâmetros de acordo com definido pelo usuário no módulo '*Creator' [RF004]*, como por exemplo: garantir que um número negativo não seja aceito por um parâmetro do tipo numérico que aceite apenas valores positivos e não nulos.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF004] Alterar dados presentes na GUI gerada

Este caso de uso permite que o usuário final altere os dados presentes na GUI gerada pelo módulo '*Builder*'.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[RF005] Ler descrição dos parâmetros definidos

Este caso de uso permite que o módulo '*Builder*' leia a descrição dos parâmetros definidos no módulo '*Creator*' *[RF005]*.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

4. Requisitos Não Funcionais

[NF001] Usabilidade

O sistema terá uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[NF002] Portabilidade

O código-fonte deve ser portável, de maneira que possa ser compilado para plataformas alternativas.

Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

[NF003] Escalável

A ferramenta deve prover a capacidade de integrar novos componentes gráficos ou novos tipos de parâmetros.

Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

[NF004] Documentação

A ferramenta deve ser bem documentada. Isto inclui documentação do código-fonte, diagramas UML, especificação da arquitetura etc.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

[NF005] Ser integrável

A ferramenta deverá ser integrada a algum outro sistema, mostrando neste sistema a sua interface gerada.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

5. Escopo Negativo

Não faz parte do escopo do projeto o design da interface gráfica gerada.

Anexo B

Documentos de Casos de Uso

A seguir estão listados os documentos detalhando os casos de uso referentes aos requisitos funcionais listados no Anexo A.

MPhyScaS GUI Generator

Caso de Uso
UC001 – Cadastrar parâmetros

Versão 2.0

Autor: Fernando Rocha
27/04/2007

UC001	
Nome do Caso de Uso	*Cadastrar parâmetros.*
Descrição	Permite o cadastro de parâmetros para os quais será gerada uma interface gráfica, esta interface será gerada automaticamente no módulo *Builder*.
Requisito	Creator, RF001
Responsável	Fernando Rocha.
Módulo	*Creator.*
Nível	Requisito.
Ator Primário	Usuário criador.
Stakeholders e Interesses	Usuário criador deseja definir os parâmetros que serão exibidos na interface gráfica.
Pré-condições	O módulo *creator* da ferramenta deve estar acoplado com o sistema que definirá a interface a ser gerada.
Pós-condições	Um novo parâmetro será cadastrado na interface.
Cenário Principal	1. Este caso de uso inicia quando o usuário seleciona a opção "*New parameter*". 2. O sistema exibe uma nova janela pedindo que seja informado o nome e o tipo do parâmetro a ser criado. 3. O usuário seleciona o tipo e o nome do parâmetro e seleciona a opção "*Add parameter*". 4. O sistema atualiza a tabela contendo os parâmetros criados.
Subfluxos e Fluxos Alternativos	*Nenhum.*
Exceções	1. (E003) Parâmetro duplicado. 1. Caso o nome inserido pelo usuário já esteja sendo utilizado por um outro parâmetro, o usuário receberá um aviso de duplicidade.
Requisitos Não-Funcionais	NF001 e NF003.
Variações Esperadas	Existência de novos parâmetros possíveis de serem criados.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC002 – Excluir parâmetro

Versão 1.0

Autor: Fernando Rocha
27/04/2007

UC002	
Nome do Caso de Uso	*Excluir parâmetro.*
Descrição	Permite a exclusão de parâmetros já existentes.
Requisito	Creator, RF002
Responsável	Fernando Rocha.
Módulo	*Creator.*
Nível	Requisito.
Ator Primário	Usuário criador.
Stakeholders e Interesses	Usuário criador deseja poder remover um parâmetro indesejado.
Pré-condições	Algum parâmetro já deve estar definido.
Pós-condições	Um parâmetro será removido na interface.
Cenário Principal	1. Este caso de uso inicia quando o usuário seleciona um parâmetro especifico na tabela dos parâmetros existentes. 2. O sistema habilita então o botão "*Remove parameter*". 3. O usuário aciona o botão de remoção. 4. O sistema remove o parâmetro e atualiza a tabela contendo os parâmetros criados.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF001 e NF003.
Variações Esperadas	Existência de novos parâmetros possíveis de serem removidos.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC003 – Editar parâmetro

Versão 2.0

Autor: Fernando Rocha
27/04/2007

UC003	
Nome do Caso de Uso	*Editar parâmetro.*
Descrição	Permite a modificação dos dados de parâmetros inseridos.
Requisito	Creator, RF003.
Responsável	Fernando Rocha.
Módulo	*Creator.*
Nível	Requisito.
Ator Primário	Usuário criador.
Stakeholders e Interesses	Usuário criador deseja editar os dados de parâmetros inseridos.
Pré-condições	Algum parâmetro já deve estar definido.
Pós-condições	O parâmetro terá seus dados alterados.
Cenário Principal	1. Este caso de uso inicia quando o usuário seleciona um parâmetro especifico na tabela dos parâmetros existentes. 2. O sistema exibe os dados do parâmetro que são possíveis de serem alterados. 3. O usuário realiza as modificações desejadas. 4. O sistema atualiza o parâmetro.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF001 e NF003.
Variações Esperadas	Existência de novos parâmetros possíveis de serem editados.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC004 – Definir restrições aos parâmetros

Versão 1.0

Autor: Fernando Rocha
28/04/2007

UC004	
Nome do Caso de Uso	Definir restrições aos parâmetros.
Descrição	*Permite a definição de restrições aos parâmetros inseridos, como por exemplo, definir que um parâmetro do tipo numérico aceite apenas valores positivos*.
Requisito	*Creator*, RF004
Responsável	Fernando Rocha.
Módulo	*Creator*.
Nível	Requisito.
Ator Primário	Usuário criador.
Stakeholders e Interesses	Usuário criador impor restrições aos dados que serão inseridos à interface que será gerada.
Pré-condições	Ao menos um parâmetro já deve estar definido.
Pós-condições	Serão inseridas restrições ao parâmetro.
Cenário Principal	1. Este caso de uso inicia quando o usuário seleciona um parâmetro existente. 2. O sistema exibe os dados que podem ser modificados no parâmetro (UC003), entre estes dados estão as restrições possíveis que o parâmetro pode assumir. 3. O usuário seleciona as restrições desejadas. 4. O sistema atualiza o parâmetro, permitindo que este assuma apenas valores coerentes com suas restrições.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	1. (E004) Valor default não permitido pela restrição inserida. 1. Caso o parâmetro tenha um valor default e este não seja mais válido após inserida uma restrição, o sistema informa ao usuário e altera o valor default para um aceitável.
Requisitos Não-Funcionais	NF001.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC005 – Gerar arquivo de descrição

Versão 2.0

Autor: Fernando Rocha
28/04/2007

UC005	
Nome do Caso de Uso	*Gerar arquivo de descrição.*
Descrição	Permite exportar os dados dos parâmetros para um arquivo, possibilitando transportar este arquivo para o módulo '*Builder*'.
Requisito	*Creator*, RF005
Responsável	Fernando Rocha.
Módulo	*Creator.*
Nível	Requisito.
Ator Primário	Módulo creator.
Stakeholders e Interesses	As descrições dos parâmetros criados devem possibilitar seu uso em outro sistema.
Pré-condições	Todos os parâmetros desejados pelo usuário criador já devem estar criados.
Pós-condições	O arquivo de descrição será gerado.
Cenário Principal	1. Este caso de uso inicia quando o sistema integrado com o módulo '*Creator*' (Gerador e/ou Configurador de Simuladores do ambiente MPhyScaS) solicita exportar os dados dos parâmetros definidos. 2. O sistema retorna uma estrutura contendo todos os dados dos parâmetros definidos. 3. O Gerador e/ou Configurador de Simuladores exporta estes dados para um meio persistente de dados.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF005.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC006 – Gerar automaticamente a GUI para os parâmetros cadastrados

Versão 2.0

Autor: Fernando Rocha
28/04/2007

UC006	
Nome do Caso de Uso	*Gerar automaticamente a GUI para os parâmetros cadastrados.*
Descrição	permite que a ferramenta gere automaticamente a interface gráfica dos parâmetros cadastrados no módulo 'Creator'.
Requisito	*Builder*, RF001.
Responsável	Fernando Rocha.
Módulo	*Builder.*
Nível	Requisito.
Ator Primário	Módulo Builder.
Stakeholders e Interesses	A interface gráfica dos parâmetros definidos deverá ser gerada automaticamente.
Pré-condições	A descrição dos parâmetros já devem ter sido lida.
Pós-condições	A GUI gerada será apresentada.
Cenário Principal	1. Este caso de uso inicia quando o módulo '*Builder*' recebe uma solicitação para exibir uma GUI. 2. O módulo monta a GUI referente a solicitação recebida. 3. O sistema exibe a GUI gerada.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	1. Nenhuma.
Requisitos Não-Funcionais	NF005.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC007 – Prover um mecanismo de notificação

Versão 1.0

Autor: Fernando Rocha
28/04/2007

UC007	
Nome do Caso de Uso	*Prover um mecanismo de notificação.*
Descrição	Permite que a ferramenta gere eventos para o sistema no qual está integrado, possibilitando assim uma comunicação, informando o estado dos parâmetros mostrados na GUI.
Requisito	*Builder*, RF002
Responsável	Fernando Rocha.
Módulo	*Builder*
Nível	Requisito.
Ator Primário	Módulo Builder
Stakeholders e Interesses	O módulo '*Builder*' deverá gerar eventos para notificar ao sistema no qual está integrado (Simuladores) que um dos seus parâmetros contidos na GUI gerada foi alterado.
Pré-condições	A GUI já foi gerada.
Pós-condições	O Simulador receberá a notificação e atualizará seus dados.
Cenário Principal	1. Este caso de uso inicia o usuário final altera algum dado presente na GUI gerada. 2. O módulo '*Builder*' gera um evento notificando o Simulador que um dado foi alterado pelo usuário.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF005.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC008 – Implementar restrições aos parâmetros

Versão 1.0

Autor: Fernando Rocha
28/04/2007

UC008	
Nome do Caso de Uso	*Implementar restrições aos parâmetros*
Descrição	Permite que a GUI gerada restrinja os tipos de parâmetros de acordo com definido pelo usuário no módulo 'Creator', como por exemplo: garantir que um número negativo não seja aceito por um parâmetro do tipo numérico que aceite apenas valores positivos e não nulos.
Requisito	*Builder*, RF003
Responsável	Fernando Rocha.
Módulo	*Builder*.
Nível	Requisito.
Ator Primário	Módulo Builder
Stakeholders e Interesses	Os parâmetros exibidos na GUI gerada pelo módulo '*Builder*' necessitam ter suas restrições atendidas.
Pré-condições	A interface deve ter sido gerada.
Pós-condições	O parâmetro terá seu valor garantido dentro das restrições impostas.
Cenário Principal	1. Este caso de uso inicia quando o usuário final altera algum parâmetro mostrado na GUI gerada. 2. O módulo '*Builder*' verifica se o valor inserido está de acordo com as restrições existentes no parâmetro. 3. O módulo 'Builder' atualiza o valor do parâmetro.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	1. (E002) Valor inserido não atende as restrições O sistema exibe uma mensagem de erro para o usuário.
Requisitos Não-Funcionais	Nenhum.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC009 – Alterar dados presentes na GUI gerada

Versão 2.0

Autor: Fernando Rocha
28/04/2007

UC009	
Nome do Caso de Uso	*Alterar dados presentes na GUI gerada*
Descrição	Permite que o usuário final altere os dados presentes na GUI gerada pelo módulo '*Builder*'.
Requisito	*Builder*, RF004
Responsável	Fernando Rocha.
Módulo	*Builder*.
Nível	Requisito.
Ator Primário	Usuário final.
Stakeholders e Interesses	O usuário deseja alterar um dado dos parâmetros presentes na interface gerada.
Pré-condições	A interface já tem que estar gerada.
Pós-condições	O sistema em que o módulo "*Builder*" está integrado será notificado da alteração.
Cenário Principal	1. Este caso de uso inicia quando o usuário altera o dado de um parâmetro contido na interface gerada. 2. A ferramenta gera um evento de alteração de dados e notifica os assinantes do mesmo. de alteração de dados e notifica os assinantes do mesmo. 3. Os assinantes (Os assinantes (o sistema em que o módulo "*Builder*" está acoplado) executam a ação requerida.) executam a ação requerida.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF005.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.

MPhyScaS GUI Generator

Caso de Uso
UC010 – Ler descrição dos parâmetros definidos

Versão 2.0

Autor: Fernando Rocha
28/04/2007

UC010	
Nome do Caso de Uso	*Ler descrição dos parâmetros definidos*
Descrição	Permite que o módulo '*Builder'* leia a descrição dos parâmetros definidos no módulo '*Creator''*.
Requisito	*Builder*, RF005
Responsável	Fernando Rocha.
Módulo	*Builder*.
Nível	Requisito.
Ator Primário	Módulo Builder.
Stakeholders e Interesses	O módulo '*Builder*' necessita ler a descrição dos parâmetros para poder gerar a interface.
Pré-condições	O arquivo de descrição dos parâmetros deve existir.
Pós-condições	O módulo '*Builder*' poderá gerar a GUI.
Cenário Principal	1. Este caso de uso inicia quando o sistema ao qual o módulo '*Builder*' está integrado (Simulador) instancia um objeto do módulo. 2. A ferramenta ler o arquivo de descrição dos parâmetros. 3. O Simulador fica habilitado em solicitar a geração da interface.
Subfluxos e Fluxos Alternativos	Nenhum.
Exceções	Nenhuma.
Requisitos Não-Funcionais	NF005.
Variações Esperadas	Nenhuma.
Observações	Nenhuma.