ODE – Um Ambiente de Desenvolvimento de Software Baseado em Ontologias

Gleidson Bertollo, Fabiano B. Ruy, Paula G. Mian, Juliana Pezzin,
Mellyssa Schwambach, Ana Candida C. Natali, Ricardo A. Falbo
Departamento de Informática da Universidade Federal do Espírito Santo
Av. Fernando Ferrari, CEP 29060-900, Vitória - ES – Brasil
falbo@inf.ufes.br

Resumo

Ambientes de Desenvolvimento de Software buscam integrar diversas ferramentas, com o objetivo de prover suporte a todo o processo de desenvolvimento de software. Contudo, para tal, é necessário que as ferramentas compartilhem uma mesma conceituação sobre processos de software e, portanto, ontologias podem ser utilizadas. Neste artigo, apresentamos ODE, um ambiente construído tendo por base uma ontologia de processo de software.

Abstract

Software Engineering Environments aims to integrate several tools, in order to provide support to the whole software development process. Although, it is necessary that the tools share the same conceptualization about software processes and thus, ontologies can be used. In this paper, we present ODE, an Ontology-based software Development Environment.

1 – Introdução

Um dos principais desafios da Engenharia de Software é prover mecanismos para que o processo de desenvolvimento de software se dê com qualidade e produtividade. Esse processo geralmente é complexo devido à grande quantidade de atividades a serem realizadas e informações envolvidas. Diversas ferramentas, conhecidas como ferramentas CASE, surgiram com o objetivo de facilitar o desenvolvimento e torná-lo mais produtivo.

Essas ferramentas muito ajudaram e, até certo ponto, cumpriram seus objetivos. Contudo, um problema detectado é que cada ferramenta apoiava apenas uma parcela do processo de software. Com o tempo verificou-se a necessidade de integração entre essas ferramentas para que se pudesse oferecer suporte a todo o processo de software.

A partir dessa necessidade, surgiu o conceito de Ambiente de Desenvolvimento de Software (ADS), cujo objetivo é prover um ambiente capaz de suportar todo o processo de desenvolvimento, com diversas ferramentas integradas trabalhando em conjunto.

Construir um ambiente que integre essa grande variedade de ferramentas também é uma tarefa complexa. Cada ferramenta pode funcionar independentemente das outras, porém todas agora estão inseridas em um mesmo processo de software, sendo necessário coordenar sua ativação e a comunicação entre elas (integração de processo e controle). Para que essa comunicação seja efetiva, os conceitos envolvidos no desenvolvimento de software precisam estar bem definidos e devem ser compartilhados pelas ferramentas. A conceituação necessária para que esse entendimento compartilhado possa ser alcançado em um ambiente pode ser obtida através do uso de ontologias. Neste artigo, é apresentado ODE (*Ontology-based software Development Environment*), um ADS construído tendo por base uma ontologia de processo de software [1].

2 – O ambiente ODE

ODE (*Ontology-based software Development Environment*) é um ADS centrado em processo, tendo sua fundamentação na ontologia de processo de software descrita em [1]. Um ADS centrado em processo é um ambiente que suporta a definição de processos de software [2] e se utiliza desta para estabelecer uma ligação explícita entre as ferramentas do ambiente e os processos definidos (integração de processo).

A principal característica que distingue ODE de outros ADSs é que ele é desenvolvido baseado em ontologias. Uma ontologia é uma representação do vocabulário de algum domínio ou problema. Mais precisamente, não é o vocabulário que qualifica uma ontologia, mas as conceituações que os termos do vocabulário pretendem capturar [3].

Se as ferramentas de um ADS são construídas baseadas em ontologias, a integração dessas ferramentas pode ser facilitada, pois os conceitos envolvidos estão bem definidos. A mesma ontologia pode ser usada para construir diferentes ferramentas, suportando atividades relacionadas à engenharia de software [4].

A arquitetura de ODE reflete sua base ontológica. Ela possui dois níveis: o Meta-Nível (*Pacote Conhecimento*) e o Nível Base (*Pacote Controle*). O meta-nível define classes que descrevem o conhecimento sobre os objetos no nível base. As classes do pacote conhecimento são diretamente derivadas de ontologias, utilizando a abordagem descrita em [5]. Todas as classes desse pacote têm seu nome precedido pela letra 'C', indicando que constituem o conhecimento de ODE. Seus objetos podem ser vistos como itens da instanciação de uma ontologia.

As classes do *Pacote Controle* não derivam diretamente de ontologias, mas são baseadas nelas. Algumas classes desse nível têm uma classe correspondente no meta-nível, preservando as mesmas restrições. Dessa maneira, o *Pacote Conhecimento* pode ser usado para descrever características dos objetos do *Pacote Controle*. Na figura 1, é apresentada parte do *Pacote Controle*. As classes que possuem um conhecimento correspondente têm essa associação representada na forma de atributos.

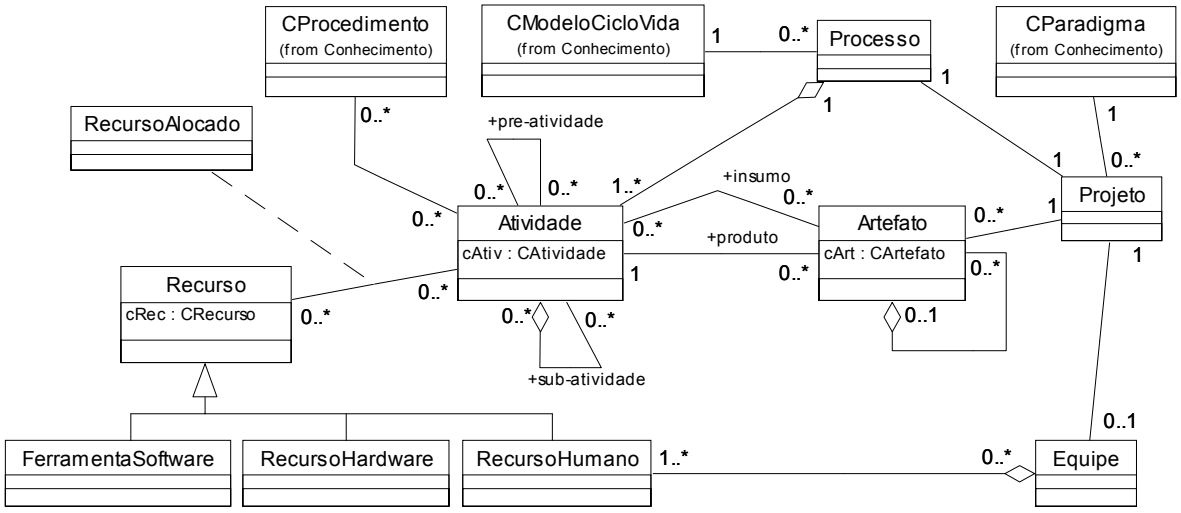

Figura 1 - Diagrama de Classes do Pacote Controle

A infra-estrutura de ODE permite controlar projetos de software e seus respectivos processos. Integradas a essa infra-estrutura existem diversas ferramentas que auxiliam as atividades do desenvolvimento de software. Dentre elas, pode-se citar: ferramenta de apoio a estimativas usando pontos de função, ferramenta de apoio ao planejamento e controle da qualidade, ferramentas de apoio à modelagem de análise e projeto, etc.

Uma importante característica de ODE é ser um ambiente configurável. Após a identificação do usuário, o ambiente configura-se segundo o seu perfil, disponibilizando apenas as funcionalidades às quais o usuário tem acesso e as informações do processo que são úteis a ele, de acordo com os projetos aos quais está alocado.

ODE está implementado em Java, possuindo, assim, a característica de ser multiplataforma. A persistência é feita em um banco de dados relacional, usando uma camada de persistência para fazer o mapeamento entre o mundo de objetos e o mundo relacional.

3 – Definição e Acompanhamento de Processo

Para que um desenvolvedor possa trabalhar no ambiente, deve estar alocado a um projeto de software que tenha um processo definido. ODE proporciona uma série de funcionalidades para tratar projetos. Assim, pode-se criar um novo projeto, abrir um projeto existente, caracterizar, ou excluir um projeto, definir o processo para o projeto, dentre outros. A caracterização consiste em definir as características relevantes do projeto, tais como o paradigma de desenvolvimento, tamanho, modularidade e equipe. A caracterização de um projeto é importante, pois se pode usar esses dados para fornecer ajuda na definição de uma série de informações sobre o projeto, tal como a sugestão de um modelo de ciclo de vida que melhor se adeque ao projeto em questão.

3.1 – Definição do Processo de Software

O primeiro passo na definição do processo de software é a escolha do seu modelo de ciclo de vida (MCV). De acordo com as características do projeto, são sugeridos os MCVs que melhor se ajustam ao projeto, filtrados a partir do repositório de conhecimento do ambiente. Contudo, o gerente de projeto fica livre para acatar a sugestão ou não.

O *Pacote Conhecimento* de ODE contém as macro-atividades que compõem a estrutura de cada MCV. Escolhido o modelo, essas macro-atividades são instanciadas para o processo, assim como a ordem de precedência das mesmas. A estrutura inicial pode ser redefinida, adicionando novas macro-atividades ou removendo aquelas não desejadas. A figura 2 mostra a definição de processos em ODE.

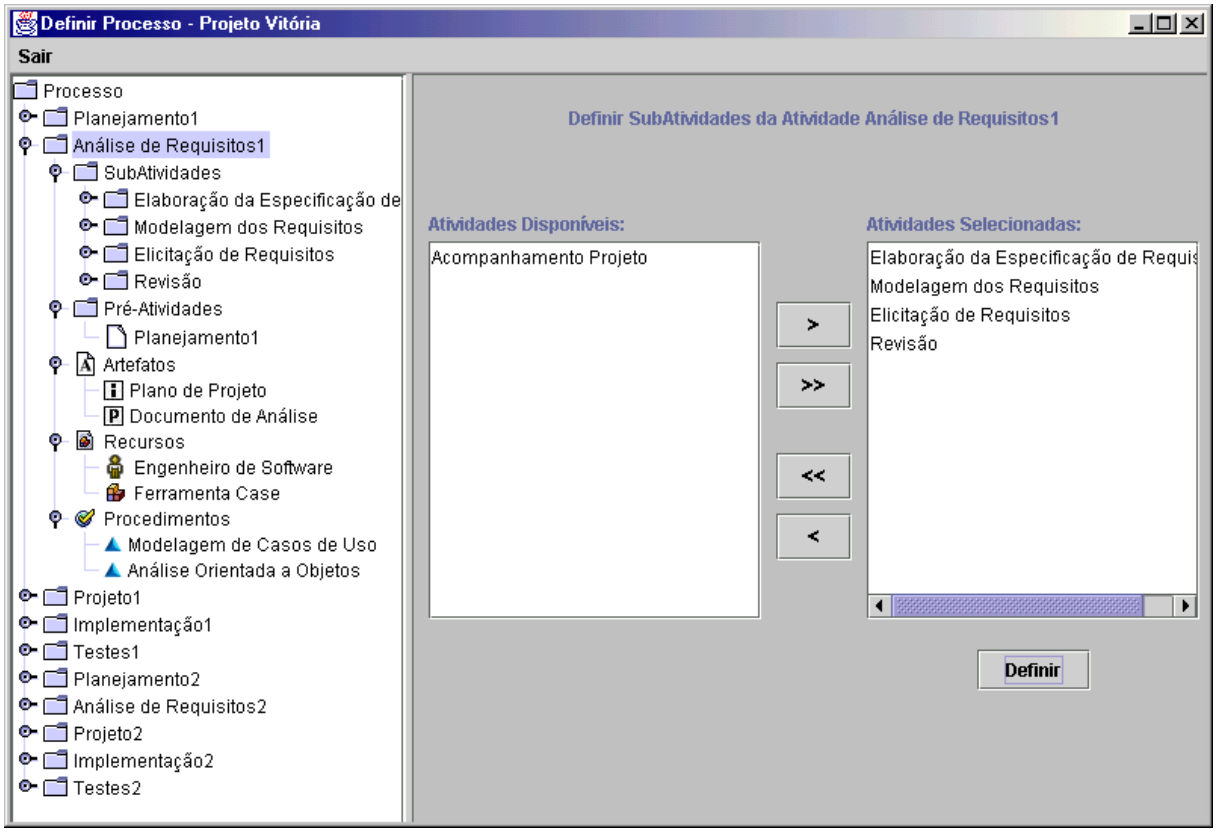

Figura 2 – Definição de Processo em ODE.

Conforme definido na ontologia de processo [1], um processo de software consiste basicamente de um conjunto de atividades e os elementos envolvidos na sua realização, a saber:

- **Sub-atividades**: uma atividade pode ser decomposta em outras atividades, ditas sub-atividades.
- **Pré-atividades**: uma atividade pode ter sua execução condicionada ao término de uma outra. A atividade antecedente é dita pré-atividade.
- **Artefatos**: uma atividade pode consumir ou produzir artefatos. Quando o artefato é consumido por uma atividade, ele é um insumo dessa atividade. Quando produzido, é um produto.
- **Recursos**: são elementos que auxiliam a execução das atividades. Podem ser recursos humanos, ferramentas de software ou recursos de hardware.
- **Procedimentos**: determinam como as atividades serão realizadas. Podem ser classificados em métodos, técnicas, roteiros e normas.

ODE possibilita a definição de cada um desses elementos de maneira simples e padronizada, como mostra a figura 2. Os elementos do processo já definidos podem ser facilmente visualizados através da árvore à esquerda. Para cada atividade instanciada, são exibidos suas sub-atividades, pré-atividades, artefatos, recursos e procedimentos.

No lado direito da figura 2, há duas listas que representam os elementos disponíveis, definidos com base no repositório de conhecimento, e os elementos instanciados, aqueles pertencentes ao *Pacote Controle* e que já fazem parte do processo. O gerente de projeto pode instanciar novos elementos ou remover elementos, movendo-os de um conjunto para o outro.

Depois de definido o processo, é possível alocar recursos humanos para suas atividades e definir seu cronograma de execução. A alocação de recursos e a definição do cronograma são importantes para a configuração do ambiente, pois definem quais ferramentas cada usuário poderá utilizar em um determinado momento, como mostra o exemplo da figura 3.

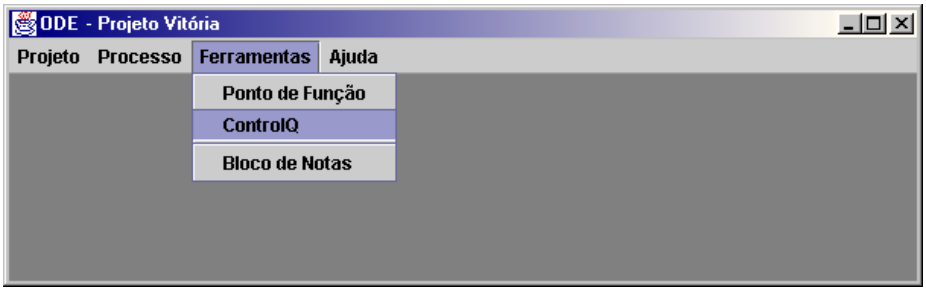

Figura 3 – O Ambiente Configurado.

3.2 – Acompanhamento de Projetos de Software

Uma vez que o projeto é iniciado, o ambiente permite que se faça o seu acompanhamento. Dessa maneira, é possível visualizar as atividades que compõem o processo de software, o estado em que se encontram, a ordem de precedência entre as mesmas e se são compostas ou não por outras atividades. O usuário pode, ainda, visualizar a quais atividades está alocado. Durante o andamento do projeto, os estados das atividades podem ser alterados.

O processo é exibido na forma de um grafo, como mostra a figura 4. As atividades são representadas pelos círculos e suas relações pelas setas, onde a seta fechada indica precedência e a seta aberta, composição. As atividades às quais o usuário está alocado aparecem em destaque, exibindo uma linha contínua ao redor do nó. As atividades que possuem alguma sub-atividade à qual o usuário está alocado são exibidas com uma linha tracejada ao seu redor.

As atividades podem se encontrar nos seguintes estados: Inativa, Aguardando Autorização, Em Execução, Suspensa, Cancelada e Finalizada.Ao iniciar um projeto, as macro-atividades que não possuem pré-atividades passam do estado *Inativa* para *Aguardando Autorização*, até que o Gerente de Projeto coloque a atividade *Em Execução*. Ela permanece nesse estado até o trabalho ser concluído, quando passa para o estado *Finalizada*. Toda atividade *Em Execução* pode ser provisoriamente *Suspensa*. Além disso, as atividades não finalizadas podem ser, a qualquer momento, canceladas.

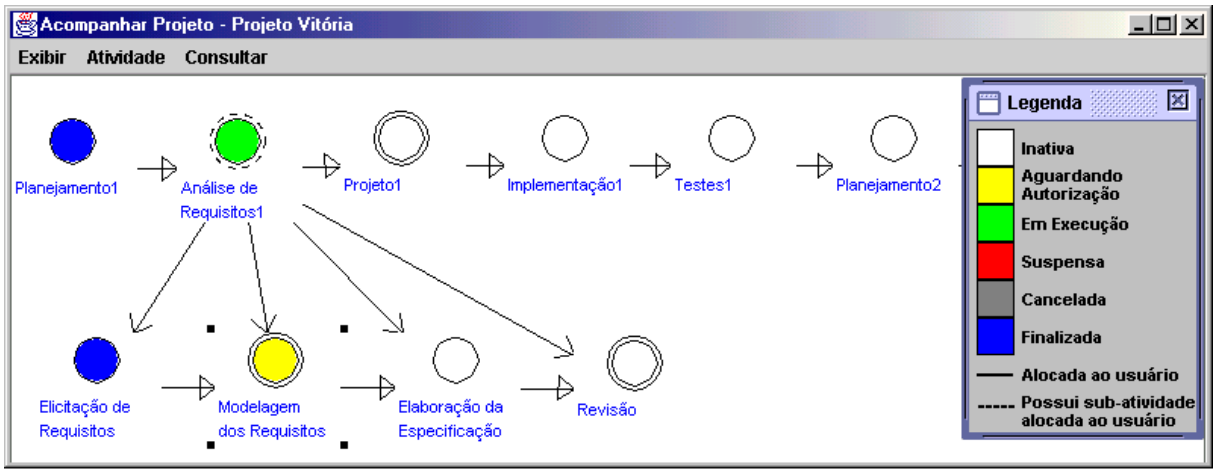

Figura 4 – Acompanhamento de Projeto em ODE.

4 – Integração de Ferramentas

Sabe-se que a integração de ferramentas é um dos maiores desafios para ADSs. Neste contexto, o uso de ontologias pode trazer muitos benefícios. Em ODE, todas as ferramentas são desenvolvidas baseadas na ontologia de processo de software ou em ontologias integradas a ela. Como exemplo, pode-se citar ControlQ – uma ferramenta de apoio ao planejamento e controle da qualidade, construída com base em uma ontologia de qualidade de software [6]. Uma grande vantagem dessa base ontológica é a facilidade encontrada na integração dessas ferramentas, uma vez que os conceitos das ontologias, tais como Processo, Atividade, Artefato, Recurso e Procedimento, estão bem definidos e são compartilhados pelas ferramentas do ambiente.

Para que uma ferramenta possa ser integrada a ODE, são necessários o aplicativo em si e uma representação para ele no ADS. Há dois tipos de ferramentas em ODE: as ferramentas internas, construídas no contexto do projeto, e as ferramentas externas, que funcionam independentemente do ambiente. A estrutura para integração de ferramentas em ODE é exibida na figura 5.

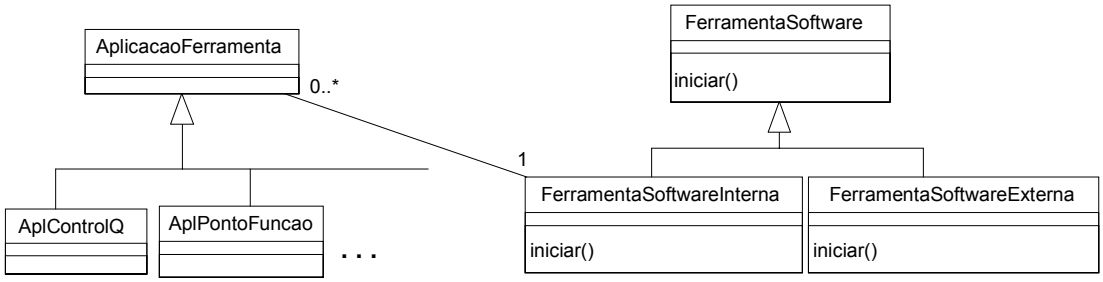

Figura 5 – Diagrama de Classes da Estrutura de Integração.

As ferramentas são recursos representados em ODE pela classe `FerramentaSoftware`. Na definição do processo, ao se escolher as ferramentas que vão apoiar as atividades, são definidas quais ferramentas poderão ser utilizadas no ambiente. A classe `FerramentaSoftware` possui o método `iniciar`, que é responsável pela inicialização das ferramentas no ambiente.

A inicialização de uma ferramenta externa é simples: o método `iniciar` chama o arquivo executável da ferramenta. Os objetos da classe `FerramentaSoftwareExterna` servem de fachada para as ferramentas externas, permitindo integrá-las ao processo de software. Contudo, a integração é apenas parcial, uma vez que os artefatos produzidos não são uniformemente tratados, não havendo integração de dados e de apresentação, como ocorre no caso das ferramentas internas.

As ferramentas internas devem pertencer à hierarquia de classes cuja superclasse é `AplicacaoFerramenta`, ou seja, a aplicação principal de cada ferramenta interna deve herdar de

`AplicacaoFerramenta`. Para propiciar a integração de processo, esta classe possui uma associação com a classe `FerramentaSoftwareInterna`. Assim, de maneira análoga às ferramentas externas, os objetos da classe `FerramentaSoftwareInterna` servem de fachada, permitindo integrar as ferramentas internas ao processo de software. Neste caso, porém, são tratadas as dimensões de integração de dados, controle, processo e apresentação [7].

Para exemplificar a estratégia de integração adotada, toma-se o caso da ferramenta ControlQ. Sua aplicação principal, `AplControlQ`, herda de `AplicacaoFerramenta`. Para que ControlQ fique disponível no ambiente, é necessário que esteja cadastrada como uma instância da classe `FerramentaSoftwareInterna`. Esta instância está associada a `AplControlQ`, que é executada quando o método `iniciar` da instância é evocado. Dessa forma, qualquer ferramenta desenvolvida no contexto de ODE pode ser facilmente integrada ao ambiente.

5 – Conclusão

Neste artigo foi apresentado o ambiente ODE, que vem sendo desenvolvido no Laboratório de Engenharia de Software – LabES – do Departamento de Informática da Universidade Federal do Espírito Santo (UFES). O objetivo do projeto é o desenvolvimento de um ambiente integrado que apóie todas as etapas do desenvolvimento de software.

Por ser um ambiente baseado em ontologias, ODE possui algumas vantagens como a criação de um repositório de conhecimento, que proporciona ao ambiente uma uniformidade de conceitos, primordial na integração de ferramentas.

A estrutura que compõe o núcleo do ambiente é capaz de abrigar inúmeras ferramentas. Atualmente ODE conta com algumas ferramentas integradas, tanto para apoiar o planejamento quanto o desenvolvimento propriamente dito. A intenção é povoar o ambiente com diversas ferramentas que vêm sendo desenvolvidas, tornando ODE um ambiente robusto capaz de apoiar todas as etapas do processo de software.

Agradecimentos

Os autores agradecem ao CNPq e à CAPES pelo apoio financeiro à pesquisa realizada.

Referências

[1] R. A. Falbo, “Integração de Conhecimento em um Ambiente de Desenvolvimento de Software”, Rio de Janeiro, RJ, Tese de Doutorado, COPPE/UFRJ, 1998.

[2] A. Fuggetta, “Software Process: A Roadmap”, in Proc. of The Future of Software Engineering, ICSE’2000, Limerick, Ireland, 2000.

[3] B. Chandrasekaran, John R. Josephson and V. Richard Benjamins, “What Are Ontologies, and Why Do We Need Them?”, IEEE Intelligent Systems, 1999.

[4] R. A. Falbo, G. Guizzardi, A. C. C. Natali, G. Bertollo, F. B. Ruy and P. G. Mian, “Towards Semantic Software Engineering Environments”, in Proc. of Int. Conference on Software Engineering and Knowledge Engineering, SEKE’2002, Ischia, Italy, 2002.

[5] G. Guizzardi, R. A. Falbo and J. G. Pereira Filho, “Using Objects and Patterns to Implement Domain Ontologies”, in Proc. of the 15th Brazilian Symposium on Software Engineering, Rio de Janeiro, Brazil, 2001.

[6] R. A. Falbo, G. Guizzardi, K. C. Duarte and A. C. C. Natali, “Developing Software for and with Reuse: An Ontological Approach”, in Proc. of the International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications CSITeA’02, Foz do Iguaçu, Brazil, 2002.

[7] S. L. Pfleeger, “Software Engineering: Theory and Practice”, 2nd ed, Prentice Hall, 2001.