Um Estudo Empírico Utilizando Z e UML para a Especificação de um Sistema de Informação

Luiz Eduardo Galvão Martins
lgmartin@unimep.br
UNIMEP - Universidade Metodista de Piracicaba

Resumo

Este artigo apresenta um relato de experiência em que procurou-se articular alguns diagramas da UML (modelagem semi-formal) e a linguagem de especificação Z (modelagem formal) para realizar a especificação de um sistema de informação gerencial. Durante o experimento ficou evidenciado os benefícios que a modelagem formal pode trazer para a melhoria da qualidade na especificação dos requisitos de um sistema de informação. Também pôde-se perceber a necessidade de uma clara correspondência entre as duas formas de modelagem, para que a especificação ocorresse de forma coerente . No experimento realizado tal correspondência foi estabelecida, garantindo que as especificações produzidas pudessem ser articuladas corretamente e conduzissem à confecção de bons modelos do sistema.

Palavras-Chave

Especificação de Sistemas, Modelagem Formal, Modelagem Semi-Formal, Modelagem de Sistemas de Informação, UML, Z

1. Introdução

A modelagem dos diversos aspectos de um sistema de informação tem se tornado uma atividade cada vez mais comum no processo de desenvolvimento de software, principalmente nas fases iniciais onde se dá especial atenção para a definição dos requisitos [Hui, 2003]. Os requisitos normalmente são elicitados, modelados, discutidos e posteriormente especificados, de tal forma que um documento que especifica detalhadamente o que o sistema deve fazer é produzido. Este documento é utilizado como guia para as fases posteriores do desenvolvimento do software [Breitman *et al.,* 1999].

A maioria dos documentos de especificação de sistemas de informação contém especificações descritas em linguagem natural e especificação semi-formal com diagramas, em particular esta segunda modalidade tem sido amplamente utilizada. Porém, a especificação semi-formal com diagramas, tipicamente diagramas como DER[1], DFD[2] e mais recentemente os diagramas da UML[3] [Booch *et al.,* 1999][OMG, 2001], freqüentemente apresenta ambigüidades e imprecisões, que podem levar a problemas na implementação do software.

[1] Diagrama Entidade-Relacionamento
[2] Diagrama de Fluxo de Dados
[3] Unified Modeling Language

Vários métodos formais de especificação de sistema têm sido propostos nos últimos 20 anos [Vienneau, 1997][Wing, 1990], tendo como um dos objetivos principais oferecer notações e semânticas precisas para a especificação de software, de tal forma que ambigüidades e imprecisões possam ser efetivamente evitadas na especificação, e portanto diminuída a incidência de problemas na implementação do software [Potter, 1996].

Porém os métodos formais têm sido pouco usados, comparativamente aos métodos semi-formais de especificação [Sommerville, 2001]. Alguns motivos que podem ser apontados para o pouco uso de métodos formais dentro da comunidade de desenvolvedores de software são:

- Pouca divulgação dos métodos formais perante os desenvolvedores de software em formação (graduandos em cursos de computação);
- As notações formais são difíceis de serem entendidas, pois normalmente requerem conhecimento matemático prévio;
- Não há ferramentas "maduras" de software que gerem código fonte efetivo para uso, a partir da notação formal, ficando uma lacuna entre a especificação formal e a implementação dos programas.

Este artigo apresenta um relato de experiência sobre o uso de métodos formais, em particular a linguagem de especificação Z, articulada com o uso de métodos semi-formais, em particular a UML, na modelagem e especificação de um sistema de informação gerencial. A escolha de um sistema de informação gerencial para a realização do experimento foi proposital, pois o objetivo foi experimentar a utilização de um método formal para a modelagem de um sistema cuja natureza (gerenciamento de informações) é muito encontrada no mercado de desenvolvimento de software [Palshikar, 2001][Saiedian, 1997].

Espera-se que este trabalho possa contribuir para a divulgação de métodos formais no uso cotidiano de desenvolvimento de sistemas de informação, aproximando as práticas de modelagem semi-formais (com diagramas) da modelagem formal, aproveitando a potencialidade de ambas as abordagens dentro de um mesmo processo de especificação de sistemas.

2. O Desenvolvimento do Experimento

O sistema de informação escolhido compreendia um subconjunto do sistema de informação de uma cooperativa médica[4]. Este subconjunto tinha por objetivo gerenciar as informações pertinentes ao atendimento domiciliar dos pacientes da cooperativa médica. Esta modalidade de atendimento abrangia um universo de 377 pacientes e 12 profissionais da área médica, com uma média de 1.700 atendimentos mensais. Os *stakeholders* do sistema constituíam-se de três usuários operativos do sistema, uma gerente de seção e dois engenheiros de requisitos (um sênior e um iniciante). A figura 1 apresenta uma macro-visão das funcionalidades esperadas do sistema de informação a ser implementado.

[4] A cooperativa médica em questão foi a UNIMED da cidade de Piracicaba-SP.

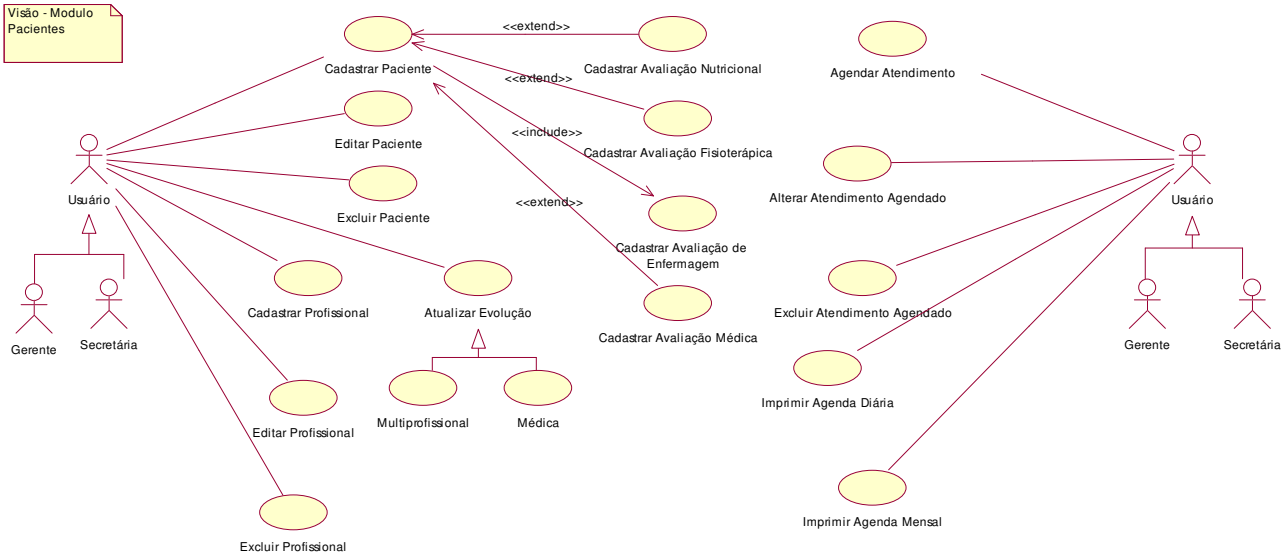

Figura 1 - Funcionalidades esperadas para o sistema de informação de atendimento domiciliar de pacientes.

A Figura 1 apresenta um diagrama de casos de uso, que mostra as funcionalidades esperadas do sistema de informação a ser implementado. Neste diagrama cada elipse representa um requisito funcional a ser implementado no sistema [Cockburn, 2000].

2.1 Especificação Semi-Formal

A especificação dos requisitos iniciou-se com a modelagem semi-formal das informações pertinentes ao sistema de informação em análise. As informações foram obtidas com várias seções de entrevistas junto aos futuros usuários do sistema, pessoas que atuavam diretamente dentro do sistema de informação (profissionais da área médica, auxiliar de escritório e gerente de seção). Após o levantamento de informações, o primeiro modelo produzido foi o diagrama de casos de uso apresentado na Figura 1. Após uma validação inicial dos requisitos expressos no diagrama de casos de uso partiu-se para a modelagem dos dados da aplicação, utilizando-se o diagrama de classes para tal modelagem (conforme mostrado na Figura 2).

Na figura 2 o diagrama de classes apresentado expressa a organização das informações presentes no módulo paciente[5] do sistema de informação. Neste modelo a ênfase da modelagem foi para os dados pertinentes ao módulo paciente, mas sem se preocupar com a distribuição das operações que deveriam atuar sobre tais dados, pois os casos de uso estavam balizando a visão funcional da modelagem inicial. As classes "avaliação fisioterápica" e "avaliação de enfermagem" possuiam uma grande quantidade de atributos, que não foram mostrados na figura apenas por uma questão de otimização de espaço ocupado pelo diagrama. Após a elaboração do modelo de classes o mesmo passou por um processo de validação junto aos *stakeholders* do sistema [Lemoine, 1998]. A modelagem semi-formal elaborada pôde facilmente ser compreendida pelos *stakeholders,*

[5] O sistema de informação analisado foi dividido em três módulos: pacientes, estoque e custo.

permitindo que os mesmos pudessem validar os documentos produzidos sem maiores dificuldades.

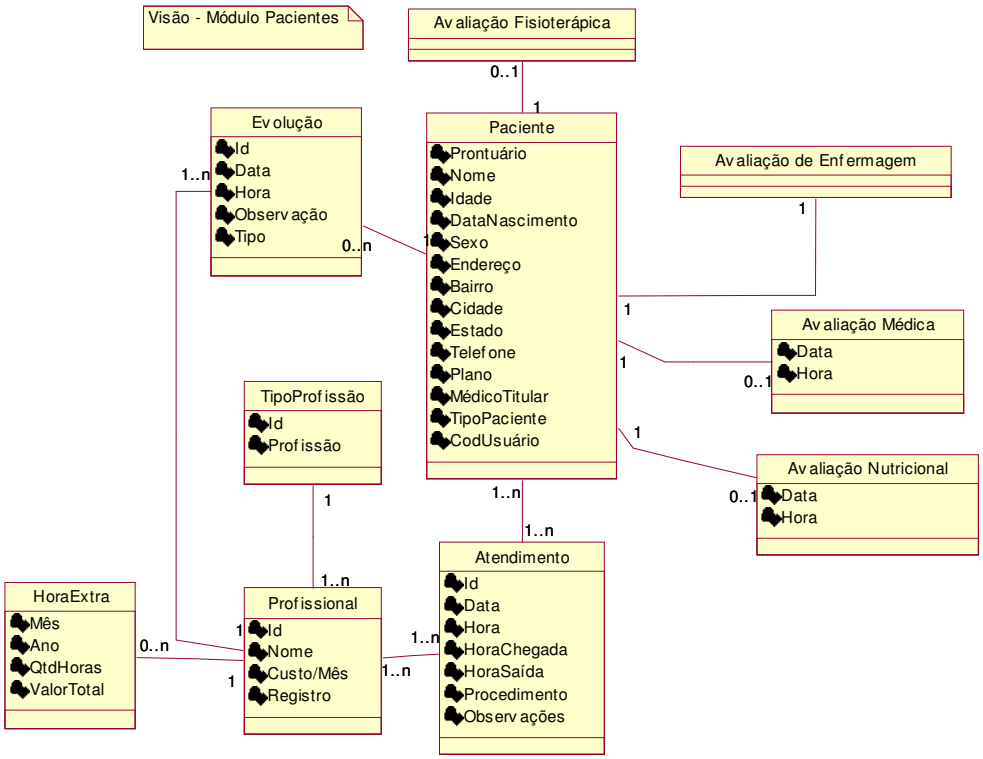

Figura 2 - Modelo de classes organizando as informações presentes no sistema de informação analisado.

2.2 Especificação Formal

Após a modelagem semi-formal ter sido validada pelos *stakeholders,* partiu-se para a modelagem formal das informações presentes nos modelos semi-formais (diagramas de casos de uso e de classes). A notação adotada para a modelagem formal foi a linguagem Z [Spivey, 1998][Moura, 2001]. No processo de especificação formal a seguinte estratégia foi adotada:

1) Especificar todas as classes presentes no diagrama de classes como tipos em Z, usando a noção de esquemas (sem a parte predicativa);
2) Especificar todos os relacionamentos existentes entre as classes como sendo relações e funções em Z;
3) Especificar todos os casos de uso como sendo esquemas em Z, demonstrando na parte predicativa as restrições e operações necessárias para o efetivo funcionamento dos casos de uso.

Os passos arrolados acima estão exemplificados nas Figuras 3, 4 e 5. A Figura 3 mostra a classe paciente especificada como um tipo em Z. A classe paciente possui vários atributos que exigiram a criação de tipos adicionais para que a especificação da classe ficasse completa, tais como

SEXO, DATA, FONE e ESTADO. Estes tipos também foram aproveitados para especificar atributos de outras classes.

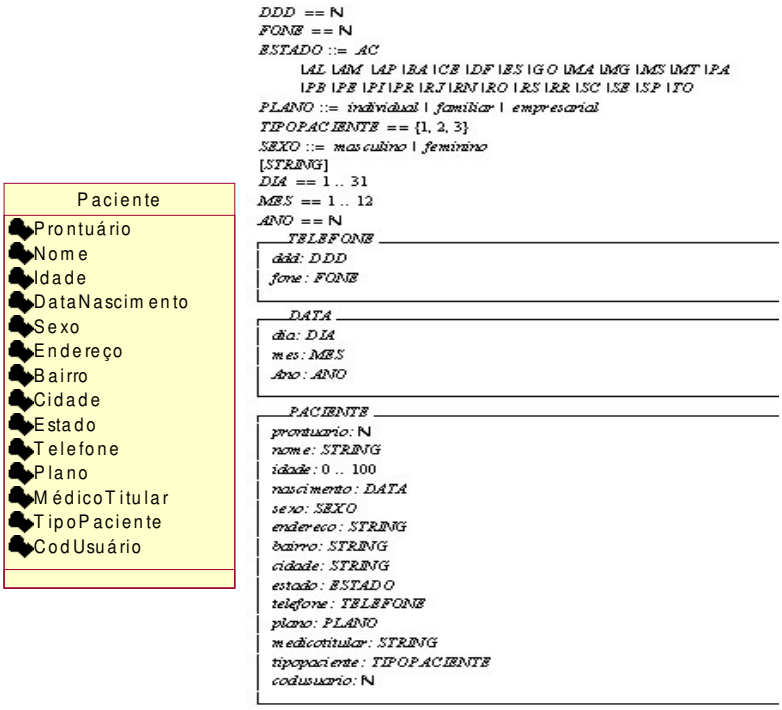

Figura 3 - Especificação em Z da classe Paciente.

A Figura 4 mostra todos os relacionamentos presentes no diagrama de classes, especificados como relações em Z (a maioria como funções). Os tipos EVOLUCAO, PACIENTE, ATENDIMENTO, PROFISSIONAL, PROFISSAO, HORAEXTRA, AVALMEDICA, AVALNUTRICIONAL, AVALENFERMAGEM e AVALFISIOTERAPICA foram previamente especificados usando a noção de esquemas.

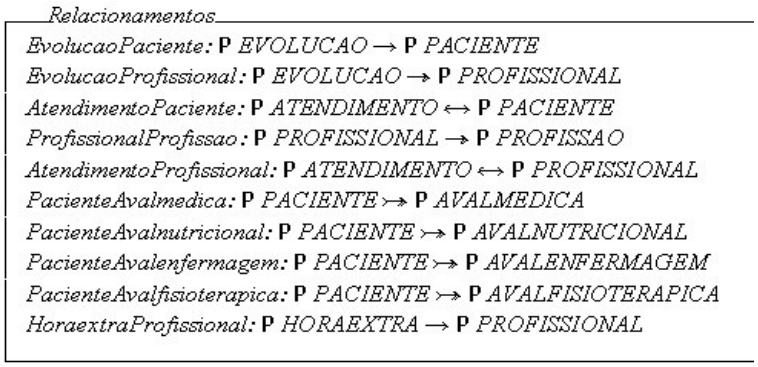

Figura 4 - Especificação em Z dos relacionamentos presentes no diagrama de classes.

A especificação dos relacionamentos entre as classes é um aspecto importante da modelagem de sistemas. Durante o experimento identificamos as correspondências entre a modelagem dos relacionamentos das classes em UML e a modelagem dos relacionamentos dos tipos em Z. A Tabela 1 mostra estas correspondências.

Tabela 1 - Correspondências entre Z e UML quanto à modelagem de relacionamentos.

Relacionamentos no diagrama de classes da UML[6]	**Relacionamentos em Z**
Associação 0..1 - 0..1	Função injetora parcial ($\rightarrowtail$)
Associação 1 - 0..1	Função injetora total ($\rightarrowtail$)
Associação 0..1 - 1	Função bijetora ($\rightarrowtail\!\!\!\!\rightarrow$)
Associação 1 - 1	Função bijetora ($\rightarrowtail\!\!\!\!\rightarrow$)
Associação 0..n - 0..n	Relação ($\leftrightarrow$)
Associação 1..n - 1..n	Relação ($\leftrightarrow$)
Associação 0..n - 1..n	Relação ($\leftrightarrow$)
Associação 0..1 - 1..n	Função sobrejetora parcial ($\rightarrow\!\!\!\!\rightarrow$)
Associação 1 - 1..n	Função sobrejetora total ($\rightarrow\!\!\!\!\rightarrow$)
Associação 1 - 0..n	Função total ($\rightarrow$)
Associação 0..1 - 0..n	Função parcial ($\rightarrow$)

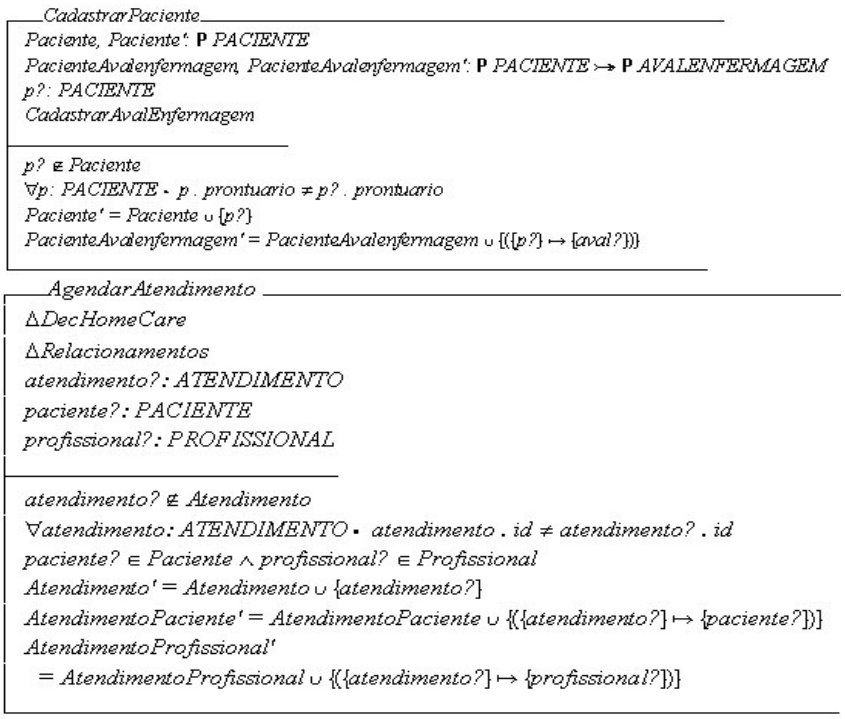

Figura 5 - Especificação em Z dos casos de uso "Cadastrar Paciente" e Agendar Atendimento".

A Figura 5 mostra os casos de uso "Cadastrar Paciente" e "Agendar Atendimento" especificados como esquemas em Z. Esta é uma articulação importante que pode ser feita entre os casos de uso e a especificação em Z. Os casos de uso são muito úteis para a especificação inicial dos

[6] A notação de multiplicidade de objetos no relacionamento foi indicada por *m..n,* onde *m* indica o número mínimo de objetos envolvidos no relacionamento e *n* indica o número máximo de objetos envolvidos. Como o relacionamento é uma via de "mão-dupla", há necessidade de indicar a multiplicidade mínima e máxima nos dois lados de um relacionamento, quando este envolve duas classes diferentes (situação mais comum na modelagem de classes). A notação "1" é uma forma resumida da notação "1..1". O *n* indica a multiplicidade "muitos".

requisitos do software, apresentando os requisitos funcionais num alto nível de abstração [Cockburn, 2000], o que facilita a comunicação entre os *stakeholders* do sistema. Porém, quando se torna necessária uma especificação detalhada de como os casos de uso devem funcionar, os engenheiros de requisitos precisam utilizar formas de especificação mais precisas, neste caso a especificação formal em Z pode ser útil, descrevendo os casos de uso como esquemas, oferecendo precisão e rigor na especificação.

3. Discussão sobre o Experimento Realizado

No experimento realizado a modelagem semi-formal foi adotada como primeira abordagem de modelagem do sistema. Os modelos diagramáticos produzidos constituíram a primeira especificação do sistema, com um alto nível de abstração, oferecendo boas condições de comunicação entre desenvolvedores, usuários e clientes do sistema. Após executada a modelagem semi-formal, com os diagramas da UML, o próximo passo foi a modelagem formal do sistema. O documento que serviu como base para produzir os tipos em Z foi o diagrama de classes. A definição de tipos em Z foi um pré-requisito para as especificações seguintes. Para cada classe do diagrama de classes foi criado um tipo em Z, as classes possuíam vários atributos, dos quais alguns exigiram a definição de tipos iniciais (o tipo STRING foi um exemplo desta situação). Como classes representam estruturas de dados, os tipos correspondentes definidos em Z foram especificados usando-se a noção de esquema [Moura, 2001], o que facilitou a correspondência entre as classes da UML e os tipos em Z.

O próximo passo foi especificar formalmente os relacionamentos entre as classes. Para isso foi utilizado o conceito de relações entre conjuntos, pois Z oferece bons recursos para a modelagem de tal conceito. Para o módulo "Paciente" foram identificadas duas relações, duas funções totais, duas funções sobrejetoras totais e quatro funções bijetoras (Figura 4). Estabelecer a correspondência entre a modelagem de relacionamentos em UML e em Z foi uma atividade que consumiu bastante tempo, pois os engenheiros de requisitos não tinham muita experiência na produção de especificações em Z, e esta correspondência não estava evidente. Assim, a produção das correspondências entre as formas de relacionamentos, conforme apresentadas nas Tabelas 1 e 2, foi de grande importância para a correta especificação dos relacionamentos entre os tipos definidos em Z, que precisavam estar totalmente equivalentes à modelagem do diagrama de classes.

Para uma discussão inicial dos requisitos funcionais do sistema os diagramas de casos de usos foram muito úteis, pois abstraíram detalhes desnecessários para o nível de elicitação dos requisitos, e foram de fácil compreensão para os usuários. Porém, uma vez que os requisitos funcionais foram identificados eles precisavam ser discutidos com mais detalhes, pois era necessário explorar os relacionamentos e as dependências entre eles, analisar quais dados eram relevantes no contexto de cada requisito e as restrições existentes. Durante o experimento a especificação formal dos casos de uso, utilizando-se a linguagem Z, mostrou-se muito profícua para se obter o detalhamento desejado, pois o rigor da especificação possibilitou a identificação de novas informações presentes nos requisitos e um melhor entendimento sobre os mesmos. A estratégia adotada foi a de especificar cada caso de uso como um esquema em Z. A parte declarativa dos esquemas permitiu uma boa visualização dos dados necessários para o funcionamento dos casos de uso correspondentes, bem como os relacionamentos dos dados

envolvidos. A parte predicativa possibilitou identificar claramente quais eram as pré-condições necessárias para o correto funcionamento dos casos de uso, e também permitiu especificar de forma precisa as principais operações envolvendo os dados do contexto de cada caso de uso. Para uma efetiva articulação entre os modelos semi-formais e formais, foi necessário estabelecer claramente as correspondências entre as técnicas de modelagem adotadas. Durante o experimento tal correspondência foi realizada, traçando-se um paralelo entre os elementos de modelagem utilizados nos diagramas de caso de uso e de classes (UML), e os elementos de modelagem utilizados em Z. A Tabela 2 apresenta as correspondências identificadas durante o experimento.

Tabela 2 - Correspondência entre os elementos de modelagem da UML e Z

Elementos de Modelagem da UML	**Elementos de Modelagem em Z**
Classe	Tipo (definido utilizando-se a noção de esquema, sem a parte predicativa, para se definir uma estrutura de dados)
Objeto	Variável declarada como sendo de um tipo estruturado (um esquema definido previamente)
Atributo de objeto	Variável simples
Associação entre duas classes	Relação ou função, dependendo das multiplicidades envolvidas na associação das classes (os detalhes desta correspondência estão na Tabela 1)
Caso de uso	Esquema
Relacionamento de inclusão entre casos de uso	Invocação de esquema dentro de esquema
Operação de um objeto	Esquema

Conclusão

A experiência relatada mostrou que para a especificação inicial de um sistema os modelos semi-formais são adequados, pois ajudam a abstrair detalhes desnecessários para a etapa inicial da especificação dos requisitos, facilitando a comunicação entre usuários e engenheiros de requisitos, e possibilitando que o processo de análise de requisitos fosse fluente e produtivo. Porém, num segundo momento, onde se buscou uma especificação mais detalhada dos requisitos, os modelos semi-formais, em particular os modelos de casos de uso e de classes, demonstraram-se inadequados, pois faltavam-lhes recursos para uma especificação mais precisa. Foi justamente neste ponto da especificação dos requisitos que a modelagem formal se mostrou útil e pôde oferecer importante contribuições para a melhoria da qualidade dos requisitos especificados, e consequentemente contribuir para um processo de desenvolvimento de software de melhor qualidade. Com o experimento realizado pôde-se evidenciar que a modelagem formal teve um papel importante na especificação do sistema de informação, ajudando a melhorar a qualidade dos atributos da especificação de requisitos, dentro os quais podemos destacar: precisão, completude, corretude e não-ambiguidade [Davis, 1997]. Articulando-se a modelagem semi-formal com a modelagem formal, conseguiu-se uma especificação de requisitos de melhor qualidade, onde os modelos se completaram, constituindo-se uma documentação robusta do sistema de informação a ser implementado. Como decorrência do experimento realizado, pretende-se explorar mais as correspondências entre os elementos de modelagem da UML e Z, pois evidenciou-se que tal correspondência é crucial para a utilização articulada destas técnicas de modelagem.

Referências Bibliográficas

Booch, G. *et al.* (1999) "The Unified Modeling Language User Guide", Addison Wesley.

Breitman, K. *et al.* (1999) "The World's a Stage: A Survey on Requirements Engineering using a Real-Life Case Study", Journal of the Brazilian Computer Society, N. 1, Vol. 6, pp. 13-37.

Cockburn, A., "Writing Effective Use Cases", Addison-Weley, October/2000.

Davis *et al.* (1997) "Identifying and Measuring Quality in a Software Requirements Specification", in Software Requirements Engineering, 2nd Ed., IEEE CS Press, pp 164-175.

Hui, B., *et al.* (2003) "Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework", 11th IEEE Conference on Requirements Engineering.

Lemoine, M. *et al.* (1998) "Validating Requirements: The Evolutionary Approach", Proceedings of the COMPSAC'98.

Moura, A. V. (2001) "Especificações em Z: uma Introdução", Editora da Unicamp.

OMG (2001) "Unified Modeling Language Specification", Version 1.4.

Palshikar, G. K. (2001) "Applying Formal Specifications to Real-World Software Development", IEEE Software.

Potter, B., *et al.* (1996) "Introduction to Formal Specification and Z", 2nd Edition, Prentice Hall.

Saiedian, H. (1997) "Formal Methods in Information Systems Engineering" in Software Requirements Engineering, 2nd Edition, IEEE-CS Press.

Sommerville, I. (2001) "Software Engineering", Pearson Education.

Spivey, J. M. (1998) "The Z Notation: A Reference Manual", 2nd Edition Published by J. M. Spivey. http://spivey.oriel.ox.ac.uk/~mike/zrm/zrm.pdf.

Vienneau, R. (1997) "A Review of Formal Methods" in Software Requirements Engineering, 2nd Edition, IEEE-CS Press.

Wing, J. M. (1990) "A Specifier's Introduction to Formal Methods", IEEE Computer, Vol. 23, No. 9, pp. 8-24.