Modelando Requisitos Especificados com Mapas conceituas através da UML-MC.

Genessa Robinson[1], Marcelo Scopel[2], Letícia Rafaela Rheinheimer[2], Junior Martins[2], Sérgio Crespo C S Pinto[2], Lúcia M. M. Giraffa[3], Crediné Menezes[4]

genessa.nho@terra.com.br marceloLscopel,@hotmail.com , leticiarafaela@yahoo.com.br juniormmartin@msn.com, crespo@exatas.unisinos.br, giraffa@inf.pucrs.br, credine@inf.ufes.br

Resumo – A especificação de requisitos funcionais de uma aplicação é um documento imprescindível para construção de sistemas de boa qualidade. A prática tem mostrado que a qualidade dessas especificações aumenta significativamente quando construída de forma cooperativa por desenvolvedores e usuários. A natureza dessa atividade é similar à utilizada por sujeitos quaisquer construindo conhecimento sobre um assunto de seu interesse. Sugerimos então que esse documento seja escrito com base em mapas conceituais, uma abordagem que tem se mostrado bastante eficaz no apoio à aprendizagem. A partir desses mapas o desenvolvedor pode produzir uma descrição mais rigorosa do sistema utilizando a linguagem UML. Esse passo pode ser realizado com mais precisão se a linguagem UML estiver sintonizada com o paradigma utilizado em mapas conceituais para representar conhecimento. Este artigo tem por objetivo apresentar uma proposta para estender a notação gráfica da UML para suportar mapas conceituais, utilizando os mecanismos de extensão da própria UML.

Palavras Chave – **UML, Mapa Conceitual, Mecanismo de Extensão.**

1. Introdução

A especificação de requisitos funcionais de uma aplicação é um documento imprescindível para construção de sistemas de boa qualidade. A prática tem mostrado que a qualidade dessas especificações aumenta significativamente quando construída de forma cooperativa por desenvolvedores e usuários.

Ao criar uma especificação de requisitos funcionais o desenvolvedor está, antes de tudo, construindo o seu entendimento sobre o mundo sendo modelado, através da definição de seus componentes, a articulação entre eles, e os comportamentos possíveis. A sua construção precisa ser registrada para que os usuários do sistema possam avaliar esse entendimento. Ao mesmo tempo os usuários também se comportam como parceiros, modificando ou acrescentando elementos.

Entendemos portanto que a especificação de requisitos funcionais pode ser realizada, com bastante ganhos, através do uso de mapas conceituais, uma abordagem que tem se mostrado bastante eficaz no apoio à aprendizagem.

A partir dos mapas conceituais que especificam o sistema, o desenvolvedor pode produzir uma descrição mais rigorosa do sistema utilizando uma linguagem mais precisa, orientada para engenharia de software, que facilite a realização das etapas seguintes. A UML (*Unified Modeling Language*) é bem aceita hoje pela comunidade de desenvolvimento de

[1] Universidade do Vale do Rio dos Sinos UNISINOS – Sistemas de Informação - Bacharel
[2] Mestrado em Computação Aplicada – PIPCA – Universidade do Vale do Rio dos Sinos - UNISINOS
[3] PUCRS/CTXML – TECNOPUC, Av.Ipiranga 6681-prédio 96b Porto Alegre – RS, + 55 51 3320-3672
[4] Universidade Federal do Espírito Santo- UFES – Programa de Pós-Graduação em Informática

software. Esse passo de tradução de mapas conceituais para UML pode ser realizado com mais precisão se a linguagem UML estiver sintonizada com o paradigma utilizado em mapas conceituais para representar conhecimento.

Um mapa conceitual é uma ferramenta conceitual, desenvolvida por Joseph Novak, que permite uma nova forma de representar e organizar conhecimento. Essa técnica é formada por conceitos e relações existentes entre os mesmos, sendo que os conceitos são organizados de forma hierárquica. Os conceitos mais gerais são colocados no topo da estrutura e os mais específicos vão sendo acrescentados em um nível inferior, de acordo com seu grau de inclusão. Utilizado primeiramente na educação, para registrar o conhecimento de um estudante tem sobre um determinado assunto, hoje tem sido aplicado em diversas áreas como, por exemplo, navegação na Web, e sistemas baseados no conhecimento.

Outra linguagem largamente utilizada para representar conhecimento é a UML (*Unified Modeling Language*). Essa linguagem pode ser aplicada em diferentes tipos de sistemas, domínios, métodos e processos. Ela permite a captura, comunicação e representação de conhecimento[2]. A UML fornece um conjunto de diagramas composto por elementos e relacionamentos, que permitem criar modelos gerais para um sistema de *software*. Porém, muitas vezes, não podemos comunicar de forma clara e precisa todos os detalhes de um sistema utilizando apenas sua notação básica. Para resolver esse problema, a UML dispõe de três mecanismos de extensão, que permitem estender a linguagem de forma controlada, para que a modelagem de um sistema possa ser mais detalhada [3, 4].

O objetivo deste artigo é apresentar o estudo realizado para estender da UML para suportar mapas conceituais, a UML-MC [5, 6]. Essa notação servirá como auxílio para que interessados em modelar sistemas baseados em mapas conceituais possam utilizar a notação aqui apresentada, a fim de não precisarem se preocupar com detalhes necessários para a representação de mapas conceituais. Na seção 2 descreveremos o paradigma mapas conceituais e na seção 3 faremos um *overview* da linguagem de modelagem UML. Por fim, na seção 4 explicaremos o processo de criação da UML-MC: escreveremos sobre a construção dos mapas conceituais utilizados para a criação dessa notação; falaremos sobre a construção de um diagrama de classes preliminar; por fim, detalharemos como se deu à extensão da UML para suportar mapas conceituais. Finalmente, na seção 5, apresentamos nossas considerações finais.

2. Mapas Conceituais

Mapa conceitual é uma ferramenta conceitual de grande valia para a representação e organização de conhecimento. Segundo o criador dessa metodologia, um mapa conceitual pode ser descrito como um conjunto de conceitos, representados através de nodos inter-relacionados, que expressam o conhecimento existente sobre um assunto. A ligação entre dois ou mais conceitos é feita através de uma linha tracejada entre eles. Para evidenciar o por quê de um relacionamento entre conceitos, palavras de ligação são colocadas nas linhas, formando assim proposições simples que mostram o significado do vínculo [7].

De acordo com esta definição, os elementos fundamentais de um mapa conceitual são: conceito, palavra de ligação e proposição. Em Novak [7], um conceito é uma regularidade percebida em eventos (qualquer coisa que acontece ou que pode ser ocasionada) ou objetos (qualquer coisa que existe e que pode ser observada) nomeados por algum termo. Uma palavra de ligação une os conceitos e indica o tipo de relação existente entre ambos. Uma proposição é a união entre dois ou mais conceitos unidos por palavras de ligação, formando uma unidade semântica.

Um mapa conceitual pode ser caracterizado como uma representação gráfica, organizada de forma hierárquica, onde os conceitos são dispostos por ordem de importância. Os conceitos mais profundos (gerais) são apresentados em um nível mais alto da estrutura hierárquica, e os conceitos mais específicos vão sendo acrescentados em um nível inferior [7]. Durante a construção de um mapa é preciso considerar que um conceito aparece uma única vez e que em algumas situações, linhas são finalizadas com uma seta (a fim de indicar um conceito derivado, quando ambos os conceitos encontram-se em um mesmo nível ou em casos de relações cruzadas) [8].

3. Unified Modeling Language - UML

A UML trata-se de uma linguagem-padrão cujo objetivo é visualizar, especificar, construir e documentar todas as informações necessárias para o desenvolvimento de modelos de sistemas complexos de software orientado a objeto [3, 4]. Definimos modelo como uma abstração que retrata a essência de um problema ou estrutura complexa, facilitando a compreensão do mesmo [20]. Um modelo pode abranger tanto um plano detalhado quanto um mais amplo, geralmente escritos em uma linguagem visual. Portanto, a maior parte de suas informações são expressas através de símbolos gráficos e conexões [22]. Conhecendo os diagramas e os relacionamentos disponibilizados pela UML, é possível o desenvolvimento de modelos complexos de software. Porém, existem situações onde, utilizando-se somente as ferramentas mencionadas acima, não é possível comunicar de forma clara e precisa todos os detalhes de um modelo. Por esse motivo, a UML oferece mecanismos de extensão, que permitem que a linguagem seja estendida ou adaptada para que a modelagem possa melhor representar a situação a que se destina [4]. Os mecanismos de extensão são apresentados a seguir.

3.1. Mecanismos de Extensão

Os mecanismos de extensão da UML são as ferramentas que possibilitam que a linguagem seja adaptada para diferentes tipos de sistemas, domínios, métodos e processos, permitindo definir e utilizar novos elementos, propriedades e semânticas [3].

Um *tagged value* define novas propriedades aos elementos da UML. Uma propriedade é usada para adicionar informações aos elementos do modelo, e estas podem ser usadas por humanos ou máquinas.

As *constraints* representam informações semânticas do modelo. São condições ou proposições descritas nos modelos, que determinam restrições que devem ser mantidas como verdadeiras[8].

Um *stereotype* representa um subconjunto de um elemento de modelagem existente, como por exemplo, uma classe, uma operação, ou um relacionamento [23]. Um *stereotype* pode ser visto como um tipo de especialização de um elemento de modelagem existente[4].

4. Criação da UML-MC

Apresentamos nesta seção a nossa proposta para descrição de mapas conceituais usando UML. A estratégia que usaremos é partir de um exemplo de mapa conceitual e a partir daí ir apresentando as necessidades e soluções encontradas.

A mapa da Figura 1 é uma especificação do ambiente AulaNet, construído a partir da leitura de documentos do sistema encontrado na literatura.

O mapa criado apresenta uma visão geral do ambiente AulaNet. Alguns ícones possuem links que levam a figuras ou textos explicativos que são úteis a todos os usuários do ambiente. Por exemplo, o conceito AulaNet possui dois ícones: uma figura (que leva a uma imagem de entrada do ambiente) e um ícone (que leva a uma página HTML, que dá uma introdução sobre o que é o ambiente AulaNet).

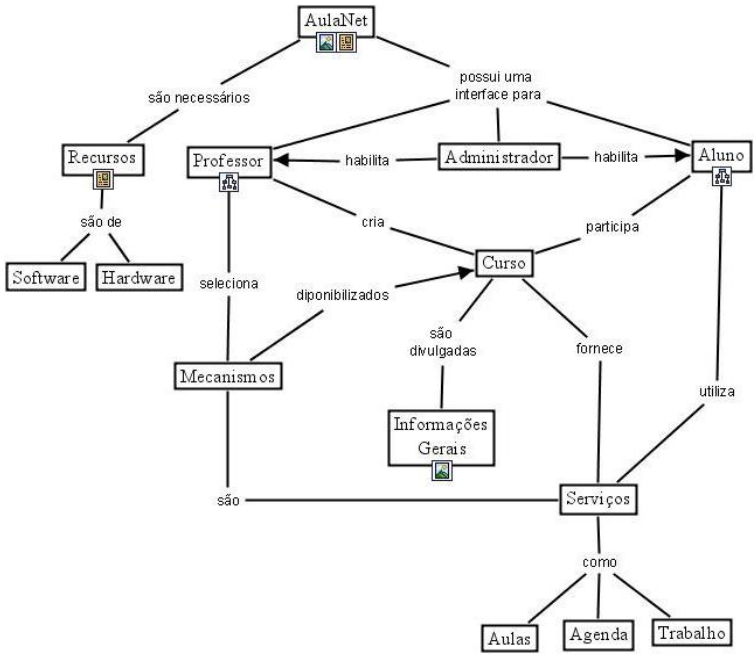

Figura 1 – Mapa Conceitual AulaNet

O conceito Recursos e Informações Gerais possuem um ícone que leva a uma página HTML. O primeiro, leva a uma página que esclarece informações sobre os recursos necessários para participar do ambiente e, o segundo, oferece um *link* a uma página que explica os tipos de informações que são disponibilizadas por um curso. Nos conceitos Professor e Aluno existe um ícone que leva a mapas conceituais secundários. Um professor ao acessar o *link* que leva ao mapa conceitual Guia do Professor, tem acesso a um mapa conceitual que contém informações relacionadas às atividades pertinentes ao professor no ambiente. Da mesma forma, um aluno ao acessar o mapa conceitual Guia do Aluno, recebe informações que são de seu interesse. Nos mapas conceituais secundários, possuem ícones que dão acesso à páginas HTML ou figuras, mas não a outros mapas conceituais.

4.2. Descrição de um Diagrama de Classes Preliminar

Para a representação de mapas conceituais através da notação UML utilizamos o diagrama de classes [4,20] . Em um diagrama de classes, cada conceito foi representado como uma classe e para mostrar que existe uma ligação entre conceitos, diferentes tipos de relacionamentos foram observados [5] e [6]. Na Figura 2, são mostrados dois diagramas de classes construídos para o mapa conceitual inicial AulaNet (Figura 1), sem o uso da UML-MC.

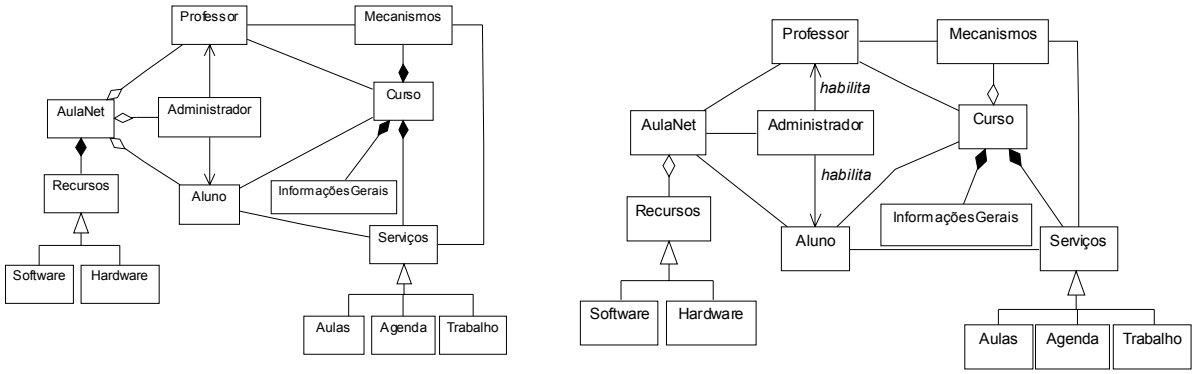

Figura 2 - Diagramas de classes do mapa conceitual inicial AulaNet sem o uso da UML-MC

Observando os diagramas da Figura 2, percebe-se que o mapa conceitual inicial (Figura 1), está sendo modelado de diferentes maneiras. Por exemplo, no primeiro diagrama, os conceitos Aluno, Administrador e Professor estão sendo relacionados ao conceito AulaNet por meio de um relacionamento de agregação. Já no segundo, essas relações estão sendo representadas através de uma associação. A UML permite diferentes interpretações no processo de escolha de como modelar uma aplicação, gerando dúvidas quanto a forma mais apropriada para representar mapas conceituais. Ou seja, não existe um padrão de modelagem que indique como mapas conceituais podem ser representados, cabendo ao modelador essa decisão. Ao propormos a extensão da UML (UML-MC) simplificamos a modelagem, pois oferecemos padrões que podem ser utilizados em modelagem de aplicações baseadas em mapas conceituais, ocasionando um ganho no tempo na tomada de decisão referente à representação das características pertencentes aos mapas conceituais.

Um único mapa conceitual pode não abranger todo um domínio de estudo, sendo necessária à criação de dois ou mais mapas para a representação completa do mesmo. Para a organização destes mapas conceituais cada mapa conceitual foi representado dentro de um pacote. Um pacote é um mecanismo que permite a distribuição de elementos em grupos de forma a organizá-los de acordo com particularidades existentes entre elementos [4, 25]. Um pacote que representa um mapa conceitual secundário é dependente de uma classe do mapa inicial. Assim, concluímos que existe uma relação de dependência entre os pacotes. Para indicar qual classe está referenciando um mapa conceitual, no pacote dependente, é feita uma referência à classe.

Depois de modelados os mapas conceituais utilizando a notação básica da UML, percebeu-se que esta não permite que todos os detalhes existentes em um mapa conceitual sejam expressos de forma clara. Por exemplo, com essa notação não foi possível determinarmos qual seria o conceito principal, intermediário ou o mais específico de um mapa. Também não foi possível representar, de forma precisa, a hierarquia entre os conceitos. Logo, foram utilizados os mecanismos de extensão disponibilizados pela UML.

4.3. Descrição dos Mecanismos de Extensão

A partir da análise dos mapas conceituais criados e dos diagramas preliminares, buscou-se identificar as particularidades existentes nos mapas conceituais e estudou-se formas de como poderiam ser representadas. Foram definidos os *stereotypes* utilizados na UML-MC. Depois foram determinadas quais propriedades (*tagged values*) e semânticas (*constraints*) deveriam ser descritas para modelar mapas conceituais na UML.

4.3.1. Definindo *Stereotypes*

Primeiramente, observamos que, de acordo com a disposição dos conceitos, os mesmos possuem comportamentos distintos. Para a definição dos elementos para representar os diferentes tipos de conceitos nos mapas conceituais foram identificados os seguintes *stereotypes*:

Nome: <<nodo_pai>>

Tipo de elemento: classe.

Descrição: um <<nodo_pai>> representa o conceito principal de um mapa conceitual, aparecendo sempre no topo da estrutura de cada mapa. Sempre possuirá dependentes que indicarão informações correspondentes a ele. Nos mapas conceituais utilizados como objeto de estudo, o mapa conceitual principal é usado para expressar o nome do ambiente e os mapas conceituais secundários representam usuários do sistema. Na Figura 3, o <<nodo_pai>> é AulaNet, pois ele representa o conceito geral do mapa conceitual inicial AulaNet.

Constraint: não aplicado.

Tagged value: não aplicado.

Nome: <<nodo_filho>>

Tipo de elemento: classe.

Descrição: um <<nodo_filho>> é a informação mais específica que existe em um mapa conceitual. Portanto, é usado em conceitos que não possuem nenhuma informação adicional a seu respeito. Sempre é subordinado a um <<nodo_pai>>, <<nodo_primo>> ou a um <<nodo_pai&filho>>, pois esclarece algo sobre o conceito superior, muitas vezes exemplificando o conceito a que está relacionado. Na Figura 3, os conceitos *Hardware* e *Software* são mais específicos do mapa conceitual AulaNet. Portanto, estes conceitos são considerados <<nodos_filho>>.

Constraint: não aplicado.

Tagged value: não aplicado.

Nome: <<nodo_pai&filho>>

Tipo de elemento: classe.

Descrição: o <<nodo_pai&filho>> ocupa um nível intermediário no mapa conceitual. Não representa o conceito principal do mapa conceitual, nem o conceito mais específico. Ele adiciona alguma informação a respeito de um conceito mais geral, que pode ser um <<nodo_pai>>, <<nodo_primo>> ou outro <<nodo_pai&filho>>. O <<nodo_pai&filho>> também possui conceitos dependentes (<<nodo_filho>>, <<nodo_primo>> ou outro <<nodo_pai&filho>>) que expressam informações a seu respeito. Um exemplo pode ser visto na Figura 3. Neste exemplo, o conceito Recursos é um <<nodo_pai&filho>>, pois especifica alguma informação sobre o conceito AulaNet e os conceitos *Software* e Hardware exemplificam tipos de recursos necessários para se utilizar o ambiente.

Constraint: não aplicado.

Tagged value: não aplicado.

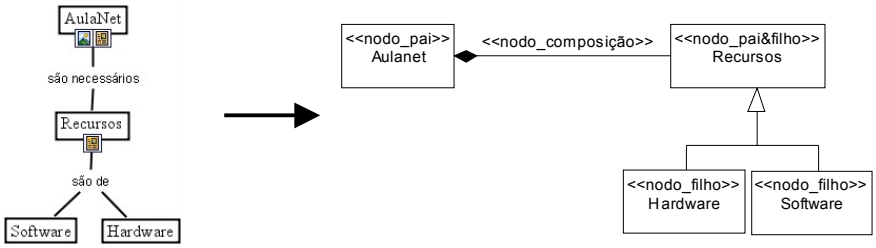

Figura 3 – Utilização dos *stereotypes* <<nodo_pai>>, <<nodo_filho>> e <<nodo_pai&filho>>.

Nome: <<nodo_primo>>

Tipo de elemento: classe.

Descrição: um <<nodo_primo>> ocorre somente quando conceitos encontram-se em um mesmo nível e relacionam-se entre si. Um <<nodo_primo>> sempre está ligado a outro <<nodo_primo>> por meio de uma associação unidirecional, pois um informa ou recebe alguma informação sobre outro. Um <<nodo_primo>> é uma especificação de um conceito (<<nodo_pai>> ou <<nodo_pai&filho>>) e pode ou não ser um conceito geral de outro, isto é, não necessariamente é fornecida alguma informação sobre o <<nodo_primo>>.

Depois de identificado como representar os conceitos de um mapa conceitual, foi analisado como tratar as relações existentes entre os conceitos. Basicamente, os conceitos específicos representam uma parte do conceito geral, gerando uma relação do tipo todo/parte. Portanto, foram definidos dois *stereotypes* para representar relações entre conceitos:

Nome: <<nodo_agregação>>

Tipo de elemento: relacionamento de agregação.

Descrição: o <<nodo_agregação>> ocorre quando um <<nodo_pai&filho>>, <<nodo_primo>> ou um <<nodo_filho>> informa algo sobre um conceito de nível superior, que pode ser um <<nodo_pai>>, <<nodo_pai&filho>>, ou <<nodo_primo>>. Nesse tipo de relação todo/parte, as informações não ficam armazenadas no conceito mais geral. Para essas ocorências, também seria possível utilizar um relacionamento de associação binária para mostrar que existe uma ligação entre conceitos, porém usamos o <<nodo_agregação>> para identificar qual conceito é hierarquicamente superior ao outro.

Constraint: não aplicado.

Tagged Value: não aplicado.

Nome: <<nodo_composição>>

Tipo de elemento: relacionamento de composição.

Descrição: o <<nodo_composição>> ocorre quando um <<nodo_pai&filho>>, <<nodo_primo>> ou um <<nodo_filho>> informa um material ou recurso que está contido no conceito mais geral a que está relacionado. Esse tipo de relacionamento só ocorre se o conceito específico possuir propriedades que fazem parte do conceito mais geral. Caso mais de um conceito esteja indicando alguma informação sobre um conceito e estes possuírem propriedades distintas ocorre um outro tipo de relacionamento: o de generalização/especialização.

Como já mencionado anteriormente, cada mapa conceitual fica armazenado dentro de um pacote. Para representar que um pacote contém um mapa conceitual, foi definido o seguinte *stereotype:*

Nome: <<mapa_conceitual>>

Tipo de elemento: pacote.

Descrição: um *stereotype* <<mapa_conceitual>> é um pacote que armazena um mapa conceitual. Um mapa conceitual é representado através de um diagrama de classes que fica armazenado dentro de um pacote <<mapa_conceitual>>. Por exemplo, o mapa conceitual inicial AulaNet é representado através de um diagrama de classes e este fica armazenado dentro do pacote <<mapa_conceitual>> AulaNet.

Os ícones presentes em alguns conceitos indicam que o conceito possui um *link* para uma mídia. Para representar as relações entre um conceito e uma mídia, foi preciso definir dois *stereotypes*:

Nome: <<nodo_link_mc>>

Tipo de elemento: relacionamento de dependência.

Descrição: representa um caminho navegacional entre pacotes <<mapa_conceitual>>. Esta relação ocorre entre o <<mapa_conceitual>> inicial e os <mapas_conceituais>> secundários a ele, pois para que os mapas conceituais subordinados sejam acessados, é preciso que eles façam referência a uma classe pertencente ao <<mapa_conceitual>> inicial.

Constraint: não aplicado.

Tagged Value: não aplicado.

Nome: <<nodo_link>>

Tipo de elemento: relacionamento de associação unidirecional.

Descrição: um <<nodo_link>> representa um caminho navegacional entre um conceito e uma mídia, que pode ser uma página HTML ou uma figura. Essa associação inicia de um conceito que aponta a uma página ou a uma imagem relacionada ao conceito selecionado, através de uma associação unidirecional. Trata-se de uma resposta a uma solicitação feita pelo sistema através da seleção do conceito. Por exemplo, na Figura 4 [6], o conceito AulaNet possui dois ícones: o da esquerda oferece um link para uma figura e o da direita um para uma página HTML. Para representar que AulaNet faz ligação a essas mídias, *stereotypes* <<nodo_link>> foram utilizados, juntamente com seus respectivos *tagged values*, descritos na próxima seção.

Constraint: não aplicado.

Tagged Value: parâmetros – é uma lista de nomes de parâmetros, que são passados na hora em que uma solicitação é feita, para informar qual mídia será acessada.

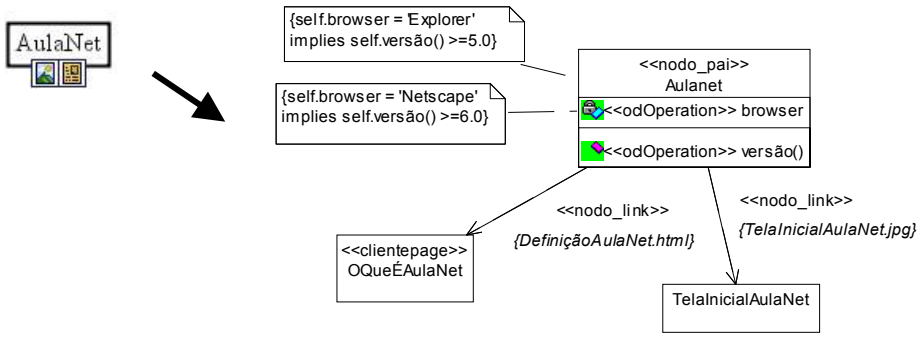

Figura 4 – Utilização de *stereotypes*, *constraints* e *tagged values* para conceitos que possuem ícones que levam a páginas HTML ou a figuras.

Com os *stereotypes* foi possível identificarmos os ícones que possuem ligação e uma mídia, porém não foi possível detalharmos as particularidades de acesso para cada mídia. Portanto, depois de definidos os *stereotypes* usados na UML-MC, foram estudados os outros mecanismos de extensão da UML.

3.4.2 Definindo *Tagged Values e Constraints*

Para os conceitos que possuem um ícone que representa um acesso a uma página HTML uma *constraint* foi definida. Dependendo do *browser* do usuário que acessa o mapa

conceitual, uma versão mínima é recomendada. Para um usuário que utiliza o *browser Explorer* a versão mínima deve ser 5.0; e para o *browser Netscape* é de 6.0. Na Figura 4 [6], mostramos essa condição, utilizando a linguagem OCL, que está representada em uma nota.

Para que possamos escrever essa expressão, definimos o atributo *browser* e a operação versão. Em OCL, criarmos propriedades para formarmos expressões é válido, tanto que o *stereotype* <<oclOperation>> é pré-definido para esse propósito [24]. A condição escrita na nota pode ser lida assim: se *browser* é *Netscape*, implica que a versão deve ser maior ou igual a seis. Da mesma forma, na outra nota, se o *browser* é *Explorer* implica que a versão do mesmo deve ser superior ou maior que cinco. Como já mencionado, o *stereotype* <<nodo_link>> é utilizado para mostrar a ligação entre um conceito a uma página HTML ou a uma figura. Um *tagged value* é usado como parâmetro para informar a página ou figura que pode ser acessada através do conceito selecionado. Quando um conceito acessa uma página HTML, utilizamos o *stereotype* <<*client page*>> definido por Conallen em [26]. Um <<*client page*>> representa uma página cliente onde sua instância é uma página web formatada em HTML, que contém propriedades que são executadas no *browser* cliente.

Para conceitos onde existe um ícone que leva a um mapa conceitual, uma *constraint* deve ser definida, uma vez que um mapa conceitual secundário só pode ser acessado se o mapa conceitual inicial for da mesma versão. Assim, foi definido o atributo versãoCMTool para criarmos a *constraint* indicada na nota da Figura 5 [6]. Neste exemplo, a *constraint* escrita na nota, restringe que, para, se ter acesso ao mapa conceitual Guia do Professor, a versão deve ser igual a 3.0.

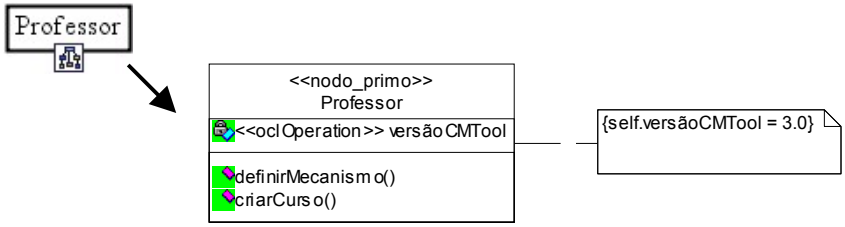

Figura 5 – Utilização de *constraint* para um conceito que possui um ícone para acesso a um mapa conceitual.

5. Conclusão

Neste artigo foi apresentada a notação UML-MC, uma proposta de extensão da notação da UML para modelar aplicações baseadas em mapas conceituais.

O uso da extensão aqui explicada dá mais semântica à modelagem, fornecendo uma simplificação da mesma, uma vez que interessados em modelar sistemas baseados em mapas conceituais, não precisam se preocupar em como representar detalhes existentes em mapas conceituais. Por exemplo, não é necessário identificar como mostrar a disposição dos conceitos, como organizar e relacionar diferentes mapas conceituais, nem como representar os conceitos que possuem ícones.

Na área de Informática e Educação está cada vez mais comum o uso de Mapas Conceituais como ferramenta de aprendizagem. Os problemas enfrentados na sua migração para a construção de sistemas computacionais residiam na forma como os Mapas poderiam ser representados. Isto permitia múltiplas interpretações e muitas vezes mudando radicalmente a semântica do trabalho.

Com a UML-MC vários padrões são sugeridos por meio da extensão da UML, oferecendo uma rica quantidade de estereotipos, vastamente comentados, permitindo que interpretações errôneas de tradução de um sistema baseado em Mapas Conceituais fosse cometida quando o mesmo tivesse necessidade de ser implementado em alguma linguagem orientada a objetos.

Referências Bibliográficas

[1] CAÑAS, A. J.; FORD, K. M.; COFFEY, J.; REICHHERZERT, T.; CARFF, R; et al.. **Herramientas para Construir y Compartir Modelos de Conocimiento Basados en Mapas Conceptuales**. Revista: Informática Educativa, Vol. 13, nro 2, 2000.

[2] ALHIR, Sinan Si. **Extending the Unified Modeling Language (UML)**. home.earthlink.net/~salhir/extendingtheuml.html , 1999.

[3] BOOCH, G.; JACOBSON, I.; RUMBAUGH, J.. **The Unified Modeling Language Reference Manual**. California: Addison-Wesley, 1999.

[4] BOOCH, G.; JACOBSON, I.; RUMBAUGH, J.. **Unified Modeling Language: Guia do Usuário**. Rio de Janeiro: Campus, 2000.

[5] ROBINSON, G. CRESPO, S.. **UML-MC: Estendendo a Notação Gráfica da UML para Suportar Mapas Conceituais de Sistemas de Auxílio em Ambientes de Ensino a Distância.** VII Simpósio de Informática e II Mostra Regional de *Software* Acadêmico. Uruguaiana: Hífen, 2002. v. 26. p. 111-115 .

[6] ROBINSON, G.. **UML-MC: Estendendo a Notação Gráfica da UML para Suportar Mapas Conceituais.** Trabalho de conclusão, UNISINOS 2003.

[7] NOVAK, D. J..**The Theory Underlying Concept Maps and How to Construct Them**.

[8] ONTORIA, A.; BALLESTEROS, A.; CUEVAS, C.; GIRALDO, L.; GÓMEZ, J. P.; et al. **Mapas Conceptuais - Uma Técnica para Aprender**. Lisboa: ASA, 1994, 1ª ed.

[9] KENNETH, F.; CAÑAS, A.; JONES, J.; STAHL, H.; NOVAK, J, et al. **ICONKAT: an Integrated Constructive Knowledge Acquisition tool**. Academic Press Limited, 1991.

[10] CAÑAS, A..; FORD, K. M.; COFFEY, J.; REICHHERZERT, T.; CARFF, R; HILL, G; et al.. **Colaboración en la Construcción de Conocimiento Mediante Mapas Conceptuales**. VIII Congreso Internacional sobre Tecnología y Educación a Distancia, San José, Costa Rica, nov. 1997.

[11] MOREIRA, M. A.; BUCHWEITZ, B. **Mapas Conceituais – Instrumentos Didáticos e Análise de Currículo**. São Paulo: Moraes, 1987.

[12] FARIA, Wilson de. **Mapas Conceituais: Aplicações ao Ensino, Currículo e Avaliação.** São Paulo: EPU, 1995.

[13] GAINES, B. R.; SHAW, M. L. G.. **Concept Maps as Hypermedia Components**. Knowledge Science Institute – University of Calgary, Canada, 1995.

[14] KREMER, R.; GAINES, B. R. **Embedded Interactive Language Concept Maps in Web Documents**. World Conference of the Web Society, out 1996.

[15] CARNOT, M. J.; DUNN, B. R.; CAÑAS, A. J.; GRAM, P.; MULDONN, J.. **Concept Maps vs. Web Pages for Information Searching and Browsing**. University of West Florida. Manuscript in preparation, 2001.

[16] FORD, K.M.; COFFEY, J. W.; TURNER, C. W.; ANDREWS, E. J. **Diagnosis and Explanation by a Nuclear Expert System**. International Journal of Expert Systems, 1996.

[17] CAÑAS, A.; LEAKE, D. B; WILSON, D. C.. **Managing, Mapping and Manipulating Conceptual Knowledge**. AAAI Workshop Technical Report WS-99-10: Exploring the Synergies of Knowledge Management & Case-Based Reasoning, AAAI Press, Menlo Calif, jul 1999.
[18] FURLAN, J. D. **Modelagem de Objetos Através da UML – The Unified Modeling Language**. São Paulo: Makron Books, 1998.
[19] KOBRYN, C. **UML 2001: a Standardization Odyssey**. Communications of de ACM. Vol 42, out 1999.
[20] QUATRINI, T. **Modelagem Visual com Rational Rose 2000 e UML.** Rio de Janeiro: Editora ciência Moderna Ltda, 2001.
[21] MATOS, A. Veloso. **UML: Prático e Descomplicado**. São Paulo: Érica, 2002.
[22] ERIKSSON, H. e PENKER, M. **UML Toolkit**. Nova York: Wiley, 1998.
[23] ÁLVAREZ, A. T.; ALEMÁN, J.L.F.. **Formally Modeling UML and its Evolution: a Holistic Approach**. Departamento de Informática: Linguagens e Sistemas, Universidade de Murcia (Espanha), 2000.
[24] WARMER, J., KLEPPE, A. **The Object Constraint Language: Precise Modeling with UML**. Massachusetts: Addison-Wesley, 1999.
[25] LARMAN, C. **Utilizando UML e Padrões: uma Introdução a Analise e ao Projeto Orientados a Objetos.** Porto Alegre: Bookman, 2000.
[26] CONALLEN, J.. Modeling Web Application Architectures with UML. Communications of the ACM. Vol. 42, out 1999.