A Outra Face dos Modelos: Técnicas de visualização para explorar modelos.

Juan Manuel Jardim Mendes[1], Guilherme Gomes[2] e Leonel Nóbrega[3]

[1] Madeira-ITI, Licenciado em Engenharia Informática, Funchal, Portugal
juan.jardim.mendes@m-iti.org
[2] Madeira-ITI, MSc, Funchal, Portugal
guilhermegrg@m-iti.org
[3] Madeira-ITI, PhD, Funchal, Portugal
lnobrega@uma.pt

Resumo. Os modelos são tradicionalmente utilizados na engenharia de *software* para documentar aspectos do desenho e, em alguns casos, como base para a geração de parte ou a totalidade dos sistemas informáticos que descrevem. Embora subsista o debate sobre este tipo de abordagens e sobre o papel e qualidades que os modelos devem possuir nesta área de engenharia, a existência de modelos que estejam em conformidade com linguagens de modelação bem definidas permite outro tipo de utilizações que vão além das anteriormente referidas. Assente no conhecimento existente sobre a visualização de dados, neste trabalho demonstramos a utilização de técnicas de visualização que permitem extrair informação sobre os modelos numa perspectiva inovadora e que contribui favoravelmente para uma melhor compreensão, análise e validação dos mesmos.

1 Introdução

O desenvolvimento de *software* [1] não é uma tarefa fácil devido a vários factores, desde o planeamento, ao levantamento de requesitos, a escolha e implementação de padrões arquitecturais, de desenho e algoritmos, gestão de equipa, entre vários outros.

A aplicação de modelos no desenvolvimento de *software* [3] é uma abordagem bem conhecida ao desenho e compreensão de sistemas de *software* e que se tornou ainda mais popular com a introdução da *Unified Modeling Language* (UML).

Os modelos [5], são usados para ajudar aos engenheiros no desenho e na compreensão de sistemas de *software* complexos. Apesar do uso crescente de modelos [3], em termos gerais, eles são usados somente nas fases iniciais do desenvolvimento de *software* e quando existe a necessidade de produzir documentação para os sistemas já desenvolvidos. Em qualquer um dos casos e atendendo à quase constante mudança que os sistemas sofrem continuamente, o valor intrínseco dos modelos está estreitamente relacionado com a exactidão com que estes representam os sistemas que modelam. Assim, a contínua mudança do *software* implica também uma contínua actualização dos modelos, todavia tal esforço só se

possa justificar se os modelos suportarem e ajudarem a realização dessas alterações quer directa ou indirectamente.

A *Model Driven Architecture* (MDA) [2] é uma abordagem, proposta pelo *Object Management Group* (OMG), que reconhece a importância dos modelos no processo de desenvolvimento de *software*, tornando-os o ponto-chave no desenvolvimento. Assim, à medida que vão surgindo alterações, são actualizados primeiro os modelos e só depois são feitas as alterações no código, através de, por exemplo, geração de código.

Sendo os modelos [2] uma descrição ou especificação do sistema e o seu ambiente, existem várias linguagens de modelação que frequentemente são utilizados num projecto por forma a capturar informação relevante e complementar, diferentes perspectivas pertinentes à compreensão do sistema e do seu ambiente. Em última análise, estas diferentes vistas são essenciais ao desenvolvimento da solução adequada.

Existem vários exemplos de modelos, como o UML, *Human Activity Modeling* (HAM), Business Process Modeling Notation (BPMN), entre vários outros.

Devido a esta combinação de diagramas é muito difícil extrair informação relevante e compreender todo o sistema usando as vistas parciais que cada diagrama representa. Existe uma enormidade de relações e de informação quantitativa que os diferentes tipos de diagramas não capturam e que são fundamentais para uma efectiva compreensão dos sistemas em estudo.

A compreensão sobre o *software* [9] tem como objectivo responder a questões sobre os sistemas de software. Em algumas situações, envolve a recolha de informações sobre o sistema, em outras situações, envolve a compreensão sobre um aspecto particular do sistema, e ainda poderá envolver questões muito específicas, como por exemplo: por que existe esta chamada de uma rotina.

Uma abordagem possível com vista a uma melhor compreensão do *software* é aplicar os conceitos de visualização de dados. As visualizações [8] permitem ao utilizador uma melhor compreensão sobre os dados que estão sendo apresentados, com uma menor carga cognitiva, permitindo de uma forma simples e fácil extrair informação sobre os objectos que estão sendo visualizados.

O trabalho apresentado neste artigo descreve uma abordagem que tem sido seguida de foram a extrair informação dos modelos, através de diferentes visualizações, com o objectivo de dar ao utilizador a possibilidade de ter uma melhor compreensão, analise e validação dos modelos.

Este artigo está estruturado da seguinte forma: a secção 2 apresenta um conjunto de conceitos de forma a podermos perceber melhor o que se pretende. A secção 3 introduz a ferramenta que esta sendo desenvolvida para sustentar as ideias apesentadas no artigo. A secção 4 apresenta a análise de um caso de estudo que permite demonstrar a utilidade das técnicas de visualização para explorar modelos. Finalmente a secção 5 descreve as conclusões e um olhar sobre o trabalho futuro.

2 Visualização de Modelos

A visualização [8] é utilizada para melhorar a compreensão de informações através da redução da sobrecarga cognitiva. Com a utilização de metodologias e ferramentas de visualização, as pessoas são capazes de compreender as informações que são apresentadas num curto período de tempo. O termo "visualização" [4] pode ser definido como a representação gráfica da informação, com o objectivo de oferecer ao espectador uma compreensão qualitativa e quantitativa do conteúdo da informação. É também o processo de transformação de objectos, conceitos e números numa forma que é visível aos olhos humanos. Quando dizemos "informação", referimo-nos a dados, processos, relações, ou conceitos.

O nosso objectivo é a utilização de técnicas de visualização de forma a melhorar a compreensão sobre os sistemas que estão a ser desenvolvidos. Geralmente sistemas de grande dimensão possuem modelos de grande dimensão, e portanto, a utilização de técnicas de visualização é seguramente uma das formas mais eficazes de melhorar a compreensão dos engenheiros sobre o sistema que estão a desenvolver. No entanto, a existência de modelos só por si não é suficientes para podermos extrair essa informação, existe a necessidade de assegurar que estes modelos estão bem definidos e que a sua estrutura é conhecida. Os meta-modelos são uma forma de garantir estas propriedades e é aquelas que estamos interessados em considerar neste trabalho.

Um meta-modelo [5] é um modelo de uma linguagem de modelação, por exemplo, o meta-modelo da linguagem UML é definido à custa da OMG UML Superstructure [15], que por sua vez é um modelo definido a partir de Meta-Object Facility (MOF), que também é um meta-modelo. O *Meta-Object Facility* (MOF) [12] fornece uma estrutura *standard* para os modelos orientados a objectos e existem já alguns trabalhos [6] que tiram partido dos meta-modelos para definir e integrar as visualizações no contexto do *Model Driven Engineering* (MDE).

O *Model Driven Visualization* (MDV) [6] é uma abordagem para a concepção e geração de visualizações usando o meta-modelo e as transformações de modelos. Esta abordagem utiliza técnicas estabelecidas pelo MDE, o que permite suportar o rápido desenvolvimento de ferramentas de visualização de informação. O MDV inclui uma plataforma independente para a visualização de modelos comuns, sendo também uma técnica para gerar instâncias da plataforma destes modelos. O MDV contribui em diversas áreas tais como: a engenharia baseada em modelos, visualização de informação e engenharia de *software*.

No entanto, na nossa opinião, o MDV ainda não é suficiente para extrair informação específica dos modelos porque não permite realizar consultas sobre os modelos.

No caso dos modelos, as consultas são geralmente realizadas através de expressões na linguagem *Object Constraint Language* (OCL) proposta pela *Object Management Group* (OMG) [7]. O OCL é uma linguagem de expressões para especificar restrições sobre modelos orientados a objectos ou outros artefactos da linguagem UML e que facilmente pode ser adaptada à escrita de consultas sobre modelos. É uma linguagem precisa, textual e formal. Essa formalidade garante a não existência de interpretações ambíguas para as mesmas restrições, facto que ocorria antes da sua criação. Uma das suas principais características é que o seu uso não exige um forte conhecimento matemático para ser utilizada correctamente, como ocorre nos modelos Z e VDM.

Apesar [13], [14] do facto de o OCL ter sido originalmente concebido para expressar restrições sobre um modelo UML, a sua capacidade de navegação entre os modelos e as colecções de dados, fez com que esta linguagem torna-se numa linguagem de consulta de modelos. Mais recentemente, a OMG propôs uma evolução do OCL onde inclui-o a possibilidade de definir transformações entre modelos, para além de melhorar a utilização da linguagem OCL como uma linguagem de consulta. Esta nova linguagem foi designada por QVT [16].

Para além do OCL, outras linguagens têm sido usadas para realizar consultas em modelos, nomeadamente o *Prolog*. O *Prolog* [13], é uma linguagem declarativa de programação baseado no paradigma de programação lógica. Devido a sua abordagem, esta tem sido aplicada para implementar mecanismos de consultas para a linguagens de consulta baseadas em vários modelos de dados.

Em [13] tenta-se encontrar qual é a melhor linguagem, em termos de performance, para realizar consultas em modelos. Os autores em [13] concluem, em termos de resultados, que uma lista de representação de modelos, como as usadas em *Prolog*, é mais eficaz nas consultas em que se pretende obter elementos com base nas suas propriedades directas. Por outro lado, a utilização do OCL nas representações de modelos que reflectem a estrutura original do modelo, permite uma rápida avaliação de consultas com base nas propriedades das relações entre os elementos.

Portanto, tendo em conta esta análise, e tendo em conta também que o OCL pode ser facilmente integrado com o MOF, iremos utilizar, para a extracção de informação, a linguagem OCL. Assim, com a combinação dos modelos, das visualizações e do OCL temos uma forma para poder extrair, analisar e avaliar informações sobre os modelos.

3 Ferramenta

A fim de podermos por em pratica o conjunto de ideias que foram apresentadas anteriormente, neste momento no encontramos a desenvolver uma ferramenta que permita visualizar as informações resultantes das consultas realizadas com auxilio do OCL, sobre os modelos.

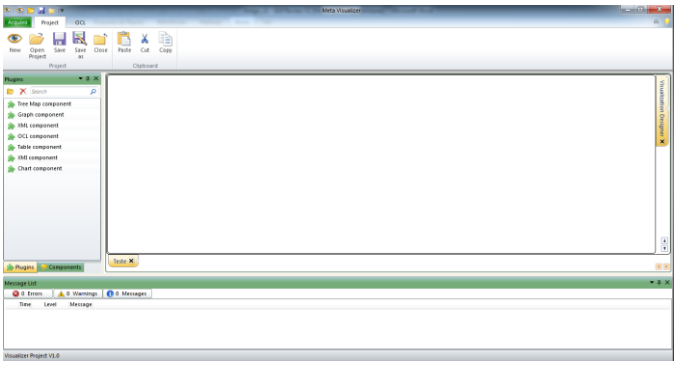

Fig. 1. Ferramenta

Esta ferramenta, Figura 1, caracteriza-se por possuir um conjunto de componentes que permitem que as visualizações sejam geradas. Cada componente pode ser vista como uma visualização dos dados a que tem acesso e por sua vez, cada componente desempenha uma função específica, por exemplo: carregar os dados do modelos que se encontram no formato XMI (*XML Metadata Interchange*), realizar consultas usando OCL ou visualizar um determinado gráfico. Estas componentes podem possuir portos de entrada e/ou de saídas cuja finalidade é a interligação de componentes.

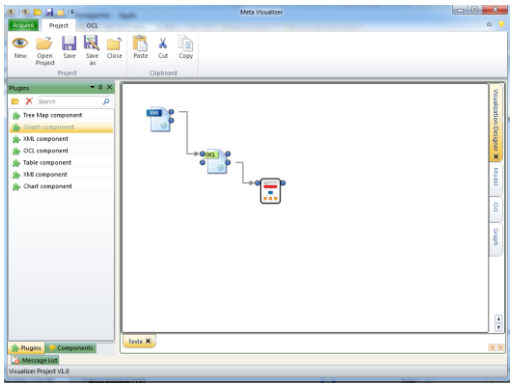

Fig. 2. Ligação dos portos das componentes

Ao interligar os portos das diferentes componentes, Figura 2, cada componente tem acesso as informações, processadas ou não, da componente de origem e por sua vez, com base no processamento dessas informações, gera uma determinada visualização. No caso da figura 2, podemos observar que a componente "OCL" tem acesso aos dados que se encontram na componente "XMI" que é usada para ler os modelos, e por sua vez, a componente "Graph" tem acesso ao resultado obtido através de uma consulta OCL realizada sobre o modelo.

Em termos das principais componentes, disponíveis neste momento na ferramenta, temos as seguintes: componente XMI, que permite visualizar o conteúdo dos modelos através de uma visualização em árvore, componente OCL que permite realizar consultas nos modelos com base no OCL, componente de gráficos que permite visualizar os dados através de diferentes gráficos, componente grafo que permite visualizar grafos e componente de mapa de calor que é caracterizados por permitir visualizar os elementos tendo em conta um segundo parâmetro que permite especificar o grau de importância.

Cada componente está disponível na ferramenta através do carregamento de um *plugins*, o que permite ainda uma maior flexibilidade no desenvolvimento de componentes específicos ou de visualizações específicas.

4 Caso de Estudo

De forma a podermos demonstrar a utilidade das técnicas de visualização para explorar modelos, usando a ferramenta descrita anteriormente, apresentamos nesta secção dois exemplos da utilização da ferramenta sobre o MOF.

Suponhamos a seguinte situação. Tendo por base o MOF, pretendemos saber os nomes de todas as classes que fazem parte do MOF. Para tal, realizamos a seguinte consulta em OCL:

```
Class.allInstances()->collect( c | c.name)
```

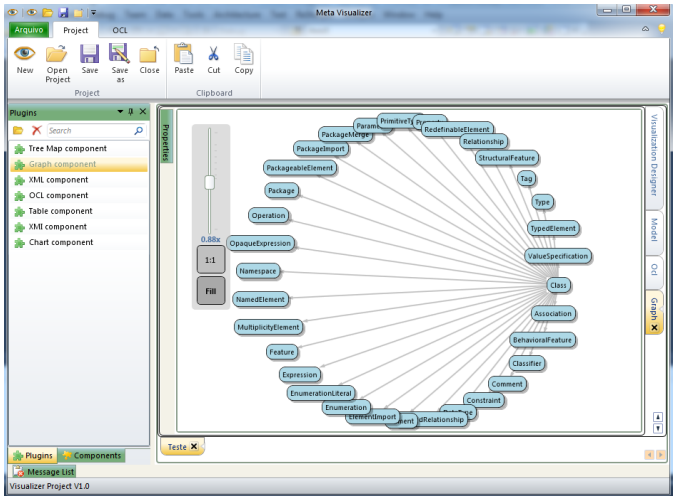

Fig. 3. Grafo resultante da 1ª consulta.

A partir do resultado obtido da consulta anterior, podemos utilizar diferentes visualizações para representar o resultado. Na figura 3, podemos observar o resultado da consulta e a sua visualização através de grafos. É necessário chamar a atenção de que o Meta-modelo do MOF é o próprio MOF.

Podemos também querer saber o número de atributos que cada elemento do "tipo" classe possui. Para tal realizamos a seguinte consulta:

```
Class.allInstances()->collect(c |
c.oclAsType(Class).ownedAttribute->size())
```

A partir da combinação destas duas consultas obtemos visualizações interessantes.

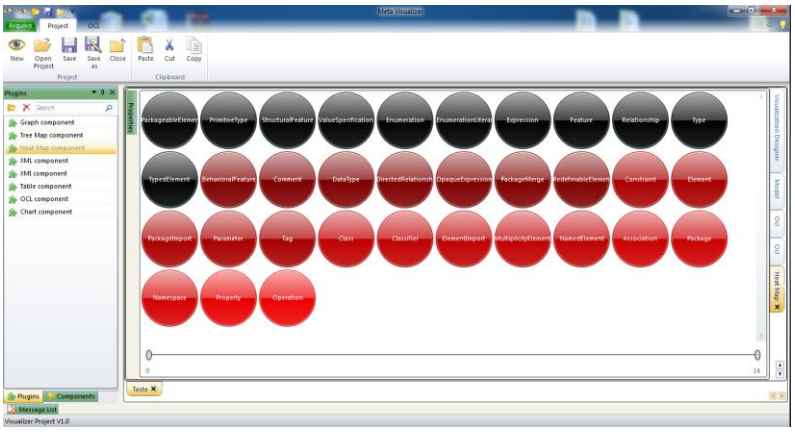

Fig. 4. Mapa de calor resultante da combinação da 1ª e 2ª consulta**.**

Na figura 3, podemos observar o mapa de calor (*Heat Map*) com a combinação das duas consultas. Rapidamente podemos observar que, por exemplo, os elementos "property" e "operation", são os elementos que possuem mais atributos e que os elementos "PackageableElement" e "PrimitiveType" são os elementos que possuem menos atributos. Esta visualização também possui um controlo adicional que nos permite escolher o intervalo dos elementos, ou seja, se estivermos somente interessados nos elementos que possuam mais do que 10 atributos, podemos seleccionar o intervalo [10, 14]. No entanto, numa primeira análise não podemos saber quantos atributos possui cada elemento.

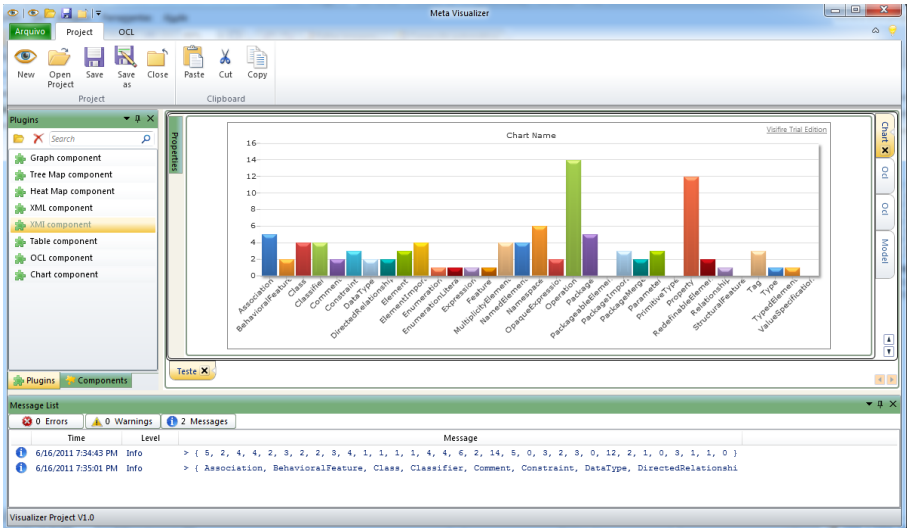

Fig. 5. Gráfico de Barras resultante da combinação da 1ª e 2ª consulta.

Por outro lado, na figura 4, podemos observar que os elementos "property" e "operation" possuem 14 e 12 atributos respectivamente, permitindo desta forma ter uma perspectiva quantitativa dos atributos. Para além do gráfico de barras mostrado na figura 4, também a ferramenta suporta outros tipos de gráficos, por exemplo, gráfico de linhas, circular, área, dispersão entre outros.

5 Conclusões

Com o desenvolvimento desta ferramenta pretendemos dar uma ajuda aos desenvolvedores de *software* de forma a poderem extrair, de uma forma mais fácil, informações sobre os modelos, através de consultas OCL. Também pretendemos ajudar a uma melhor compreensão e validação dos modelos através da utilização de diferentes visualizações.

Pretendemos futuramente desenvolver uma linguagem visual de OCL, para que o utilizador possa, de uma forma intuitiva e sem muitos conhecimentos de OCL, realizar as consultas que pretende. Também pretendemos dar as visualizações existentes e aquelas que poderão vir as ser construídas, um conjunto de propriedades que permitam, sem recorrer a realização de novas consultas, visualizar intervalos pretendidos, ou seleccionar um elemento ou conjunto de elemento específicos. Para além disso, pretendemos também dar suporte ao QVT e a novas visualizações que sejam interessantes para analisar os resultados. Por último, pretendemos dar suporte a visualização de grandes modelos e facilitar a visualização e compreensão desses modelos.

Referencias

1. Basili, V.R. Software development: A paradigm for the future. 1989 Proceedings of the Thirteenth Annual International Computer Software Applications Conference. 471-485 (1989).
2. Truyen, F. The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture, http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf, (2006).
3. Tekinerdoğan, D.B. CS 587 Model-Driven Software Development, http://www.cs.bilkent.edu.tr/~bedir/CS587-MDSD/.
4. Kaidi, Z. Data Visualization. , Singapore (2005).
5. Bull, R.I. Model Driven Visualization: Towards a Model Driven Engineering Approach for Information Visualization, http://webhome.cs.uvic.ca/~chisel/thesis/irbull_phd.pdf, (2008).
6. Computer Human Interaction & Software Engineering Lab, http://www.thechiselgroup.org/mdv.
7. Lima, D.H.A., Musial, R. Entendendo OCL Apresentação e utilização da Object Constraint Language, http://www.ic.unicamp.br/~eliane/Cursos/MC627/ocl_artigo.pdf, (2001).
8. ANAND RAO, A., MADHAVI, K. Framework for Visualizing Model-Driven Software Evolution and its Application. Journal of Computer Science. 7, 47-53 (2010).

9. P. Reiss, S. A Visual Query Language for Software Visualization. Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments. 80-82 (2002).
10. Visualization and Analytics Center for Enabling Technologies (VACET), http://www.vacet.org/vistools/query_driven_vis.html.
11. Visualization Group, http://www-vis.lbl.gov/Research/VariableInteractionQDV/index.html.
12. Hearnden, D., Raymond, K., Steel, J. MQL: a powerful extension to OCL for MOF queries. Seventh IEEE International Enterprise Distributed Object Computing Conference, 2003. Proceedings. 264-276 (2003).
13. Chimiak_Opoka, J., Felderer, M., Lenz, C., Lange, C. Querying UML Models using OCL and Prolog: A Performance Study. 2008 IEEE International Conference on Software Testing Verification and Validation Workshop. 81-88 (2008).
14. Akehurst, D.H., Bordbar, B. On Querying UML data models with OCL. UML 2001 The Unified Modeling Language Modeling Languages Concepts and Tools 4th International Conference Toronto Canada October 2001 Proceedings. 2185, 91-103 (2001).
15. Specification, O.M.G.A., Bars, C. OMG Unified Modeling Language (OMG UML),. Language. (2007).
16. Biehl, M. Literature Study on Model Transformations. Technology. 1-28 (2010).