Especificação de Métricas para Linguagens de Modelação: estado da arte e sua aplicação no UML

Daniela Morais Fonte Ismael Vilas Boas
Universidade do Minho
danielamoraisfonte@gmail.com, ismael.vb@gmail.com

Resumo

A qualidade não passa apenas pela obtenção de software que respeita as diversas normas e padrões de qualidade, mas também pela garantia de que este cumpre todos os requisitos especificados. As medições quantitativas associadas a esta problemática revelam-se essenciais em qualquer ciência, o que não é excepção na Engenharia de Software: existe um esforço contínuo para encontrar novas propostas, adequadas ao desenvolvimento de software. Com este artigo pretendemos assim mostrar como tirar partido das Linguagens de Modelação existentes, e aplicar estas medições qualitativas numa fase mais inicial do projecto (aquando a sua modelação), de forma a não centrar a análise apenas no código fonte. Apresentamos assim as principais métricas existentes dentro da área da medição de modelos de UML e algumas das principais ferramentas para especificação de métricas sobre UML, que combinado com a aplicação de métricas também ao código fonte se revela um aliado fundamental para uma melhor gestão do software produzido.

Palavras-Chave

Métricas de Software, Medição, Linguagens de Modelação, UML, Qualidade do Software

1. INTRODUÇÃO

A produção de software segue, no seu geral, uma abordagem sistemática e organizada que procura a forma mais eficaz de obter um produto final com qualidade [HWY09]. A selecção do método mais apropriado para o seu desenvolvimento ou até da abordagem mais criativa são assim pontos fundamentais para atingir esta meta.

Esta *qualidade* não passa apenas pela obtenção de software desenvolvido de forma correcta (i.e., que respeite as diversas normas e padrões de qualidade), mas também pela garantia de que este cumpre todos os requisitos especificados [MP07]. Assim, esta busca pela qualidade implica um aperfeiçoamento de todo o processo de desenvolvimento do software, tendo sempre em conta a produtividade e os benefícios de utilizar ou não novos métodos e ferramentas.

Surge assim a questão de como avaliar essa mesma produtividade, benefícios ou até o impacto da variação de um ou mais requisitos impostos. Ou até, de como estimar dados quantitativos e qualitativos que nos permitam gerir todas as decisões de uma forma ponderada e eficaz, o que nos leva à necessidade de recolher dados que respondam a estas questões - números ou valores quantitativos que funcionarão como indicadores do estado actual do projecto [GPC00, Gua08]. Estas medições quantitativas revelam-se essenciais em qualquer ciência, o que não é excepção na Engenharia de Software: existe um esforço contínuo para encontrar propostas semelhantes adequadas ao desenvolvimento de software [LLL08, GZ07, GMP00, FP98, HWY09].

Estas medições representam uma forma objectiva e quantificável de estimar custos, esforço de desenvolvimento e de garantir qualidade através da optimização da performance, *debugging*, controlo de custos ou até através de testes [HWY09]. Permitem uma avaliação tanto do processo de desenvolvimento (com o objectivo de melhorá-lo de forma contínua), como do projecto em si para auxiliar na estimativa de qualidade e, consequentemente, num melhor controlo do projecto.

Uma *métrica* de software é assim qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada [Som07]. Deve ser facilmente calculada, compreendida e testada; passível de estudos estatísticos e de automação; e ainda sugerir uma estratégia de melhoria. Para além de quantificar o que pretendemos medir, é importante que o faça de forma fidedigna - i.e. que permita obter os mesmos resultados perante as mesmas condições.

Torna-se assim fulcral incorporar métricas no ciclo de vida do software, para que estas monitorizem a sua qualidade de produção de modo a facilitar a sua posterior manutenção. [HSB09, Mar98] Com o crescente desenvolvimento de software em larga escala, a sua complexidade aumenta rapidamente, o que dificulta o controlo de qualidade: é assim necessário que todo o processo de produção seja verificado com o auxílio das métricas correctas e adequadas para cada situação.

Muitas métricas foram já propostas especificamente para esta avaliação do planeamento de um sistema de software - porém, a maior parte das abordagens existentes para a medição dessas mesmas métricas implicam uma análise de código fonte [MP07, GZ07, LLL08, KMB04, FP98]. Como tirar então partido das Linguagens de Modelação existentes, e aplicar estas medições qualitativas a uma fase mais inicial do projecto - aquando a sua modelação?

De facto, nem sempre é simples aplicar métricas já existentes a fases iniciais do processo de desenvolvimento. Com o crescente uso de linguagens como o UML (*Unified Modeling Language*) [BRJ05] para modelar sistemas, é obrigatória uma pesquisa mais profunda no sentido de investigar como fazer medições a partir de modelos UML - e numa fase anterior à implementação do sistema. Deste modo, centramo-nos neste artigo na apresentação das principais métricas existentes sobre modelos UML e das respectivas ferramentas que permitem tanto a modelação como a especificação de métricas sobre UML. Para isso, este artigo possui a estrutura enunciada de seguida.

Na Secção 2 são introduzidos os principais conceitos relacionados com métricas de software, bem como o tipo de medição que estas permitem. Na Secção 3 são analisados os quatro modelos de UML mais utilizados e apresentadas as principais métricas associadas a cada um. Na Secção 4 são apresentadas algumas das principais ferramentas para modelação e especificação de métricas sobre UML. Na Secção 5 concluímos este artigo com algumas considerações finais.

Vamos assim começar por nos debruçar sobre a análise do tipo e formas de medição que as métricas nos permitem, para uma melhor compreensão de todos os conceitos envolvidos.

2. MÉTRICAS: COMO E O QUE MEDEM?

Uma *métrica* de software consiste numa medida quantitativa do grau em que um sistema se encontra em relação a um determinado atributo. Fornece medidas para o software e para o seu processo de desenvolvimento, associando a um atributo uma descrição quantitativa. Definem, recolhem e analisam os dados resultantes da medição, através dos quais é facilitada a compreensão,avaliação, controlo e optimização do produto final obtido.

Uma *medida* fornece uma indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo [KMB04]. Um *atributo* consiste numa propriedade física ou abstracta mensurável de uma entidade.

De acordo com *standard of software Quality Metrics* do IEEE, métricas de software são ”uma função cujo *input* consiste em dados sobre o software, e cujo *output* é um valor que pode ser decisivo sobre como o atributo em questão afecta o software em análise”.

2.1. Classes de Métricas de Software

As métricas de software dividem-se em três grandes classes: métricas de produto, do projecto e do processo [HWY09].

As primeiras, são utilizadas para compreender e controlar a qualidade do produto. Permitem prever e manipular a qualidade do software, centrando-se no nível de confiança, gestão, complexidade do software, portabilidade, documentação, etc.

As segundas, são essenciais para a compreensão e controlo do estado actual do projecto. São normalmente aplicadas para um projecto específico e medem aspectos como o custo associado, carga de trabalho, produtividade, risco, grau de satisfação dos clientes, etc.

As últimas, centram-se no processo de desenvolvimento do software, focando-se em aspectos como a duração do projecto, o seu custo total, eficiência dos métodos em uso, ciclo de vida do projecto, etc.

Representam assim padrões quantificativos de medição aplicáveis a vários aspectos dos projectos de software. Dentro destas classes encontramos

métricas orientadas ao tamanho dos atributos em análise; métricas orientadas à função, que efectuam a medição da complexidade de um software do ponto de vista do utilizador; métricas de produtividade, que se focam na saída dos vários processos. Existem ainda métricas técnicas, que se concentram nas próprias características do software e não na forma como este foi desenvolvido, e por último, métricas de qualidade que oferecem uma indicação de quanto o software se adequa aos requisitos iniciais e às exigências a isso implícitas.

3. APLICAÇÃO DE MÉTRICAS DE SOFTWARE A MODELOS UML

A modelação de Software consiste na construção de modelos que representam o comportamento de um sistema de software. Na sua construção, estes modelos são utilizados na identificação das características e funcionalidades (que o software deverá oferecer) recolhidas na análise de requisitos e no planeamento do seu desenvolvimento.

O desenho de modelos de software antes do início da implementação do projecto é já um standard dentro da Engenharia de Software. Implica a modelação dos atributos que representam cada componente do software a desenvolver, e as relações entre cada um deles.

Enquanto que a modelação de programas não orientada a objectos usa linguagens como os Fluxogramas ou as Redes Petri, a modelação de programas orientados a objectos usam normalmente a linguagem UML. Esta é uma linguagem de modelação visual e intuitiva usada como padrão pelo OMG [1], muito popular tanto a nível empresarial como académico.

Existem naturalmente inúmeras diferenças entre o desenvolvimento de software orientado a objectos e o método de desenvolvimento tradicional. Deste modo, os trabalhos existentes centram-se essencialmente nas características dos sistemas orientados a objectos para uma melhor adaptação das métricas a este tipo de sistemas [Mar98, GMP00, GZ07, HSB09, BDW99, BDW98].

Dentro dos diversos modelos UML apresentamos de seguida métricas sobre os quatro mais utilizados: os Diagramas de Classe, os Casos de Uso, os Diagramas de Estados e, por fim, os Diagramas de Actividade.

Os primeiros, modelam os objectos que integram directamente o sistema - consistem numa descrição formal da estrutura dos objectos no sistema. Para cada objecto, descrevem a sua entidade, os seus relacionamento com outros objectos, os seus atributos e respectivas operações.

Os modelos de objectos são assim representados graficamente por Diagramas de Classe, e a sua qualidade tem um impacto profundo na qualidade final do software posteriormente implementado pois estes descrevem o modelo geral de informação de um sistema.

Métricas de Classes

Métrica	Categoria	Descrição
NumAttr	Size	The number of attributes in the class.
NumOps	Size	The number of operations in a class.
Nesting		The nesting level of the class (for inner classes).
Dep_Out	Coupling (import)	The number of elements on which this class depends.

Tabela 1. Exemplos de Métricas sobre Diagramas de Classes

Na Tabela 1 podemos observar alguns exemplos de atributos que podemos medir com o uso de métricas sobre Diagramas de Classe. Outras métricas bastante comuns para este tipo de modelo contabilizam, por exemplo, o número de *gets* e *sets*, número de atributos herdados, ou até o número de interfaces que a classe implementa.

Os Casos de uso são utilizados para representar o levantamento de requisitos de um sistema. São muito úteis para garantir que o sistema final é útil para o utilizador [Cle06].

Os métodos públicos de cada classe orientam as acções modeladas pelos diagramas de Caso de Uso, e as respectivas acções disponibilizadas para cada actor afecto ao sistema. As métricas sobre este último tipo de de diagramas são assim úteis para validação do cumprimento dos requisitos impostos.

Na Tabela 2 podemos observar alguns exemplos de atributos que podemos medir com o uso de métricas sobre Diagramas de Caso de Uso. Outras métricas bastante comuns para este tipo de modelo contabilizam, por exemplo, o número de vezes que um Caso de Uso aparece num diagrama ou até o número de Casos de uso que estendem um determinado diagrama.

No caso dos Diagramas de Estado, estes são utilizados para descrever o comportamento de um objecto.

[1] Object Management Group

Métricas de Casos de uso

Métrica	Categoria	Descrição
NumAss		número de associações em que o caso de uso participa.
ExtPts		número de pontos de extensão do caso de uso.
Including	Ligação (import.)	número de casos de uso que este inclui.

Tabela 2. Exemplos de Métricas sobre Casos de Uso

Um estado representa uma situação de um objecto que se prolonga durante um determinado intervalo de tempo, durante o qual não sofre estímulos de objectos exteriores nem há qualquer alteração em nenhum dos seus atributos.

Métricas de Estados

Métrica	Categoria	Descrição
TEffects	Complexity	The number of transitions with an effect in the state machine.
TGuard	Complexity	The number of transitions with a guard in the state machine.
TTrigger	Complexity	The number of triggers of the transitions of the state machine.
States	Size	The number of states in the state machine.

Tabela 3. Exemplos de Métricas sobre Diagramas de Estados

As métricas associadas a este tipo de diagrama encontram-se associadas à complexidade e dimensão do problema [GMP02]. Na Tabela 3 podemos observar alguns exemplos de atributos que podemos medir com o uso de métricas sobre este tipo de diagramas. Outras métricas bastante comuns para este tipo de modelo contabilizam, por exemplo, o número de transições entre estados ou até o número de actividades definidas em cada estado.

Os Diagramas de Actividade descrevem fluxos de trabalho e são ainda úteis para detalhar operações de uma classe (incluindo comportamentos que possuam processamento paralelo).

Métricas de Actividade

Métrica	Categoria	Descrição
Actions	Size	The number of actions of the activity.
ObjectNodes	Size	The number of object nodes of the activity.
Pins	Size	The number of pins on nodes of the activity.
Guards	Complexity	The number of guards defined on object and control flows of the activity.

Tabela 4. Exemplos de Métricas sobre Diagramas de Actividade

Como podemos observar pelos exemplos da Tabela 4, existem também diversas métricas sobre Diagramas de Actividade. Para além destas, podem ainda ser medidos atributos representativos do número de grupos ou regiões de actividade num diagrama, o número de fluxos de objectos, ou até o número de excepções de cada diagrama.

A capacidade de efectuar medições precisas tanto de modelos UML, como de código fonte, é sem dúvida importante tanto para garantir uma avaliação da qualidade nas fases iniciais do sistema (quando o custo efectivo de alterações no projecto é muito menor), como para determinar em que pontos a implementação se pode desviar do seu projecto inicial - não cumprindo assim os seus requisitos iniciais. Para isto, a combinação destes dois tipos de métricas revela-se fundamental pois uma variação nos seus resultados poderá ajudar a identificar que partes da implementação se desviam do seu esboço inicial.

4. FERRAMENTAS PARA ESPECIFICAÇÃO DE MÉTRICAS SOBRE UML

Os ambientes de modelação em UML mais utilizados são o *Visual Paradigm for UML*[2] e o *Poseidon for UML*[3], que permitem formalizar sistemas em ambiente visual, de uma forma simples e intuitiva. Porém, estes sistemas não possuem suporte para a especificação de métricas. Para isso, é necessária a utilização de software especializado neste tipo de tarefa, dos quais destacamos o SDMetrics[4], que veremos em detalhe de seguida.

[2] http://www.visualparadigm.com/product/vpuml

[3] http://www.gentlewhere.com/products.html

[4] http://www.sparxsystems.com.au/products/ea

4.1. SDMetrics

Este sistema permite, como vimos, a especificação de métricas sobre modelos UML. Baseia a sua análise na importação de um modelo UML previamente definido e exportado para o formato XMi (suportado por exemplo tanto pelo *Visual Paradigm* como pelo *Poseidon*).

Desta forma, possibilita a análise de todas as métricas sobre modelos UML que vimos na secção anterior, entre muitas outras. Gera diversos gráficos, como por exemplo, organogramas, histogramas, matrizes de relações, entre outros, que permitem uma melhor análise dos resultados obtidos através da medição.

É um dos softwares desta área mais abrangente, que engloba uma grande variedade de métricas, e que oferece ainda um sistema de *benchmarking* baseado na análise estatística dos dados. Possui ainda a opção de ser corrido a partir de uma linha de comandos.

4.2. Sparx Systems Enterprise Architect

Este sistema permite a modelação formal em UML, baseada na normal UML 2.X. Disponibiliza uma ferramenta de avaliação de projectos, onde são calculadas métricas unicamente sobre Casos de Uso e actores. Permite assim o cálculo de méricas sobre configurações do projecto, para definir a complexidade do sistema em estudo.

Possui uma interface de configuração da avaliação de métricas, baseado em classificações pontuais associadas a cada Caso de Uso, permitindo um ajuste personalizado ao cálculo de complexidade associado a cada projecto. Permite ainda definir a complexidade associada a cada actor, permitindo variar o nível de complexidade de cada um, repercutindo assim de forma mais fidedigna a complexidade real associada a cada tipo de interacção (utilizador presente $\neq$ utilizador remoto).

4.3. IBM Rational

Consiste num sistema[5] apto para a modelação de software em UML, que permite também a sua posterior implementação e gestão. Possui aplicações específicas para avaliação da qualidade do produto, e suporta o desenvolvimento distribuído. Deste modo permite a especificação de métricas, não relacionadas directamente com os modelos UML, mas sim com todo o processo de desenvolvimento do sistema, e com o código fonte.

[5] `http://www.ibm.com/software/rational`

5. CONCLUSÃO

Como vimos, a capacidade de efectuar medições precisas tanto de modelos UML, como de código fonte, é sem dúvida importante tanto para garantir uma avaliação da qualidade nas fases iniciais do sistema, como para determinar em que pontos a implementação se pode desviar da sua modelação inicial. Esta *qualidade* não passa apenas pela obtenção de software desenvolvido de forma correcta (i.e., que respeite as diversas normas e padrões de qualidade), mas também pela garantia de que este cumpre todos os requisitos especificados.

É neste contexto que surge a necessidade de efectuar medições para avaliar pontos como a produtividade, os benefícios ou até o impacto da variação dos requisitos impostos. Estas medições quantitativas revelam-se essenciais em qualquer ciência, o que não é excepção na Engenharia de Software: existe um esforço contínuo para encontrar propostas semelhantes, adequadas ao desenvolvimento de software.

Com este artigo pretendemos mostramos como tirar partido das Linguagens de Modelação existentes, e aplicar estas medições qualitativas a uma fase mais inicial do projecto (aquando a sua modelação). Analisamos assim as principais métricas existentes dentro da área da medição de modelos de UML e algumas das principais ferramentas para especificação de métricas sobre UML, e consideramos ser fundamental a aplicação de métricas de software tanto a nível do código fonte, como à modelação do sistema - a própria combinação das diversas medições revela-se uma arma excepcional para controlo e avaliação do software final.

Referências

[BDW98] Lionel C. Briand, John W. Daly, and Jürgen Wüst. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, 3:65–117, July 1998.

[BDW99] Lionel C. Briand, John W. Daly, and Jürgen K. Wüst. A unified framework for coupling measurement in object-oriented systems. 25, 1999.

[BRJ05] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide, The (2nd Edition)*. Addison-Wesley Professional, 2 edition, May 2005.

[Cle06] Roy K. Clemmons. Project estimation with use case points. pages 18–22, 2006.

[FP98] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thompson Computer Press, 1998.

[GMP00] M. Genero, M. E. Manso, and M. Plattini. Early metrics for object oriented information systems, 2000.

[GMP02] Marcela Genero, David Miranda, and Mario Piattini. Empirical validation of metrics for uml statechart diagrams, 2002.

[GPC00] Marcela Genero, Mario Piattini, and Coral Calero. Early measures for uml class diagrams. 6, 2000.

[Gua08] Karina Guarizzo. Métricas de software, 2008.

[GZ07] Hui Gao and Li Zang. Research on software architecture level structure metrics. pages 19–23, 2007.

[HSB09] Xiao Han, Li Shang, and Wang Bo. A tool for the application of software metrics to uml class diagram. pages 181–184, 2009.

[HWY09] Tu Honglei, Sun Wei, and Zhang Yanan. The research on software metrics and software complexity. 2009.

[KMB04] Cem Kaner, Senior Member, and Walter P. Bond. Software engineering metrics: What do they measure and how do we know?, 2004.

[LLL08] Rüdiger Linckle, Jonas Lundberg, and Welf Löwe. Comparing software metrics tools. pages 20—24, 2008 2008.

[Mar98] M. Marchesi. Ooa metrics for the united modeling languages, 1998.

[MP07] Jaqueline A. McQuillan and James F. Power. On the application of software metrics to uml models. pages 217–226, 2007.

[Som07] Ian Sommerville. *Software Engineering*. Addison Wesley, 8 edition, 2007.