UML no Desenvolvimento de Sistemas Interactivos

José Creissac Campos António Nestor Ribeiro
Departamento de Informática / CCTC, Universidade do Minho
Campus de Gualtar, 4710-057 Braga, Portugal
{jose.campos,anr}@di.uminho.pt

Resumo

Os processos típicos de análise e modelação de sistemas numa perspectiva de engenharia de software atribuem pouca importância à modelação da camada interactiva. O Unified Process e a linguagem que o suporta (UML) não são excepção. Este artigo propõe uma abordagem, que procura integrar a modelação da camada interactiva no processo de modelação típico baseado em UML, explorando as potencialidades fornecidas pela linguagem.

Palavras-Chave

Desenho baseado em modelos, UML, desenho centrado no utilizador, Unified Process.

1. INTRODUÇÃO

A *Unified Modelling Language* (UML) [Group 05] constituiu-se como a norma para o desenvolvimento de software orientado a objectos. A orientação a objectos é uma abordagem atractiva no que se refere ao desenvolvimento de sistemas interactivos, uma vez que este tipo de abordagem potencia a separação de camadas proposta em modelos como o MVC [Krasner 88]. No entanto, tanto a UML, enquanto linguagem de modelação, como o Unified Process [Jacobson 99], enquanto processo de desenvolvimento, são reconhecidamente deficientes no suporte ao desenvolvimento de sistemas interactivos [John 03]. Face às novidades introduzidas na versão 2.0 da UML torna-se interessante explorar qual a melhor forma de retirar o máximo benefício da linguagem no contexto do desenvolvimento de um sistema interactivo.

Neste artigo apresentamos uma proposta para a utilização da UML no contexto referido. Não propomos a inclusão de novos elementos na linguagem, visto que a nossa motivação se centra em utilizar a norma e procurar as melhores práticas que permitam a sua correcta utilização no desenho da camada interactiva. Este trabalho tem a sua génese na leccionação da linguagem a alunos de licenciatura e foi, em parte, motivado pelas dificuldades sentidas pelos mesmos na articulação entre camada interactiva e camada de lógica de negócio. Assim, exploramos até que ponto a UML pode ser utilizada na resolução dos problemas que um engenheiro de software enfrenta, quais os seus pontos fortes e fracos, bem como os compromissos que se impõem.

2 UML E SISTEMAS INTERACTIVOS

Tal como foi referido, têm surgido diversas propostas de extensão da UML para melhor a adaptar ao desenvolvimento de sistemas interactivos. Algumas destas propostas vão no sentido de adicionar novas linguagens ao conjunto disponibilizado pela UML [Patern? 00]. Outras no sentido de propôr perfis para a modelação de aspectos dos sistemas interactivos em UML [Nunes 01, Silva 02].

A adopção destas propostas por parte da comunidade tem-se mostrado lenta. Este facto dever-se-á, por um lado à natural resistência a aprender novas linguagens de modelação, e por outro, à resistência a apreender novos perfis face ao potencial de confusão que eles encerram (por se tratar de novas extensões à notação).

Acresce que em alguns casos os perfis se centram na modelação dos objectos gráficos presentes na interface quando esse aspecto começa a estar razoavelmente resolvido pelos IDEs correntes com a possibilidade de edição gráfica das mesmas. Na verdade, a questão coloca-se, não tanto na edição do aspecto gráfico da interface, mas mais na estruturação do diálogo entre utilizadores e aplicações, bem como na estruturação do código que o suporta.

Como a UML se encontra bem estabelecida e é utilizada e suportada por uma vasta comunidade de utilizadores, torna-se assim relevante, pelo menos num período transitório, explorar qual a melhor forma de retirar o máximo benefício do que a linguagem fornece de base, para a resolução dos problemas que um engenheiro de software enfrenta quando confrontado com a necessidade de desenvolver um sistema interactivo.

3. ETAPAS DO DESENVOLVIMENTO

O *Unified Process* (UP) não considera a interface como um sub-sistema com especificidades particulares, e genericamente incorpora-a nas fases de desenvolvimento. Deste modo, é criticado por não dar suficiente ênfase ao processo de descoberta dos requisitos de usabilidade e subsequente prototipagem e avaliação da interface.

Propomos, aqui, que a UML seja utilizada como mecanismo para a descrição da camada interactiva, e sugerimos alterações de processo ao delineado no UP, por forma a valorizarmos esta componente especial que merece um nível de detalhe razoavelmente mais complexo do que normalmente se lhe atribui.

3.1 Identificação das principais etapas

Por forma a utilizarmos a UML para a modelação da camada interactiva, e uma vez que não existe suporte nativo para este processo, é imperioso que se conjuguem os diagramas de forma a obter o efeito desejado. As alterações que se introduzem ao processo implicam que alguns diagramas são abordados mais cedo que o previsto no UP, visto que sendo assim auxiliam a descoberta de conhecimento sobre a camada interactiva. Apresentam-se agora as principais etapas e os diagramas utilizados em cada uma:

Análise de requisitos Embora no UP os requisitos derivados da camada interactiva estejam misturados com requisitos gerais de funcionamento, é possível registar todos os objectivos do utilizador que o sistema deverá suportar através de diagramas de Use Case. Torna-se necessário elencar, para cada um dos Use Cases, os cenários que o descrevem.

Análise de tarefas A descrição dos cenários, em que se conjugam requisitos e se estabelece uma ordem para a sua execução, pode ser utilizada como base para a modelação das tarefas. Alguns diagramas da UML, como os diagramas de actividade, são utilizados como ferramenta para a análise. Embora não exista uma mapeamento linear entre linguagens de modelação de tarefas típicas, por exemplo a linguagem ConcurTaskTrees (CTT) [Patern? 99], e os diagramas de actividade da UML, é possível utilizar estes últimos para descrever o comportamento que suportará um determinado Use Case.

Desenho do diálogo A estrutura do diálogo que deverá suportar o comportamento descrito pelos diagramas de actividade anteriormente mencionados, pode ser representado em UML à custa de um diagrama de estados [Horrocks 99]. É nesta fase que se começa a definir a API que a camada da lógica de negócio deverá fornecer à camada de interface.

Desenho da arquitectura Através da descoberta, durante a fase de descrição do diálogo, do comportamento associado às transições, é possível descrever no diagrama de classes quais são as entidades da camada de negócio e qual o seu estado e operações. A partir desta camada o processo é muito semelhante à utilização da UML para a descrição de um qualquer sistema de software.

Modelação de comportamento Nesta fase descrevem-se, recorrendo aos diagramas de comportamento da UML, os diversos aspectos comportamentais do sistema; em particular o modo como os objectos cooperam para implementar as operações previamente identificadas.

Desenvolvimento e Implementação Onde se desenvolvem em código final as funcionalidade do sistema e se abordam os aspectos típicos de *deployment*.

Como tal, a linguagem de modelação não necessita de a suportar, no ponto de vista de quem modela, embora seja importante que a distância semântica entre estes dois níveis não seja muito grande.

4. UML NO PROCESSO DE DESENVOLVIMENTO

Como atrás referido, o nosso objectivo é a incorporação das ferramentas diagramáticas da UML no processo de modelação e desenvolvimento da interface. Como não existe suporte explícito para estas tarefas na linguagem, nem existe especial interesse no processo de desenvolvimento associado (o UP) pela componente correspondente à parte interactiva, o que propomos resulta numa metodologia de utilização da linguagem por forma a fornecer um suporte "não oficial"às tarefas identificadas.

Uma alternativa seria a construção de extensões à linguagem ou mesmo a construção de ambientes dedicados à modelação e especificação da interface. Contudo, apesar de esse ser um caminho mais directo, levaria a que o processo fosse apenas possível em ferramentas próprias, logo não estando de forma alguma incorporado no natural processo de modelação. Isso constituiria um risco na medida em que tal significa que efectiva e implicitamente estariamos a reconhecer que de facto a camada interactiva e a camada de negócio constituem dois mundos separados.

Assim, as três fases da construção da camada interactiva que pretendemos incorporar são a *Análise de Requisitos*, a *Análise de Tarefas* e o *Desenho do Diálogo*. Para cada uma destas actividades, por esta ordem, é proposto um método de abordagem e utilização da UML, por forma a capturar a informação pretendida. Seguidamente descreve-se, de forma sucinta, o processo referido.

i) Análise de requisitos — utilizam-se os diagramas de Use Case para capturar os requisitos expressos pelos utilizadores do sistema. Este é o diagrama mais adequado para esta fase até pela definição de Use Case, que diz tratar-se de uma descrição informal dos requisitos funcionais, dos actores envolvidos e dos resultados produzidos.

Para melhorar a capacidade de expressividade da descrição associamos a cada um dos use case a descrição de um ou mais cenários, por forma a capturar a este nível toda a informação que se pode obter e que é importante para a camada interactiva. Requisitos de usabilidade, tal como outros requisitos não funcionais, podem ser incluídos, nesta fase, como notas ou restrições ao modelo.

ii) Análise de tarefas — utiliza-se a informação obtida anteriormente para extrair a informação necessária à especificação das tarefas.

Abordagens como as CTT capturam formalmente esta descrição, mas em UML podemos recorrer aos Diagramas de Actividade por forma a garantir um nível de expressividade semelhante.

Recentemente foi acrescentada aos diagramas a capacidade de delinear regiões interrompíveis (no que concerne ao controlo de fluxo), o que permite a especificação de operações de cancelamento da tarefa. Falta ainda, em

relação à expressividade oferecida pelo CTT, a capacidade de certos diálogos poderem ser temporariamente suspensos e posteriormente retomados (operador |>). Não é ainda linear a capacidade de efectuar estruturação e encapsulamento explícito de diagramas de actividade, pelo que a descrição da análise de tarefas obtida necessita por vezes de documentação de apoio.

iii) Desenho do diálogo — nesta fase, através da informação existente nos Diagramas de Use Case e de Actividade, identificam-se os écrans necessários à aplicação e, em função das descrições existentes nos Diagramas de Actividade desenvolvem-se Diagramas de Estados. Os diagramas de estados assim obtidos, não apresentam, a nível de fluxo de informação, alterações significativas em relação ao correspondente diagrama de actividades. No entanto, o foco de atenção passa das actividades realizadas por cada actor, para o impacto que essas actividades têm na interface.

Ao descrever as transições torna-se também possível decorá-las com os métodos que a camada de negócio deverá disponibilizar para suportar o comportamento descrito. Este passo metodológico permite que se já refinando o modelo e que sejam levantados os métodos da camada de negócio que implementarão as acções associadas à camada de apresentação.

Sendo este processo nitidamente iterativo, através da descrição dos diagramas de transição de estado vai-se completando a informação sobre o sistema e, além de se especificar a componente relativa ao controlo de diálogo, também se vai adquirindo informação sobre a arquitectura da camada de apresentação.

Descritas as fases da modelação da interface e sua correspondente incorporação na linguagem UML, importa referir algumas pré-condições que regem o método proposto: a) existe um diagrama de actividade para cada use case descoberto na análise de requisitos (cada diagrama de actividades deve conter toda a informação dos possíveis cenários que se podem obter a partir de um use case); b) deve existir um caminho no diagrama de estado para cada linha (fluxo) que se possa traçar no diagrama de actividades; c) o conjunto dos métodos obtidos na decoração dos diagramas de transição de estados para uma determinada entidade representa o comportamento exibido por essa entidade (logo, não existe comportamento numa entidade que não seja obtido a partir do processo de construção do diagrama de transição de estado).

A derivação da lógica de negócio é obtida através da completude do processo de construção dos diagramas de transição de estado. Apesar de este processo ser nitidamente iterativo e ser díficil determinar quando se chega a uma situação estável, o processo defendido neste artigo produz uma substancial parte do comportamento que constitui a lógica de negócio. Parece ser também a forma mais natural de determinar a lógica de negócio, uma vez que os métodos obtidos por este processo derivam totalmente da análise das tarefas que foram identificadas. Sendo assim, o comportamento obtido corresponde à totalidade (ou quase)

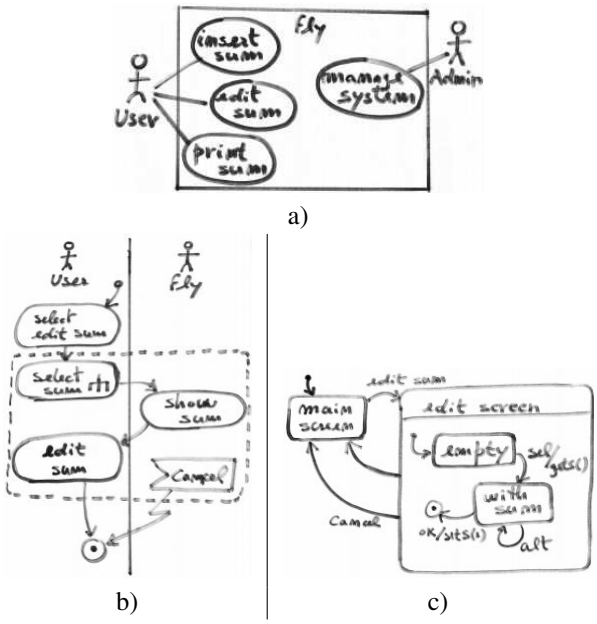

Figura 1. Diagramas UML

do que o sistema deve exibir. Existirão sempre métodos utilitários, ou resultantes de algum processo de *refactoring* ou re-ordenação do código, mas a componente principal que corresponde aos métodos da interface deverá ser encontrada através do processo defendido neste artigo.

5. UM EXEMPLO

Para cada fase mencionada na secção anterior, iremos agora apresentar (através de extractos de um exemplo — o sistema de gestão de sumários `Fly`) de que forma os diagramas UML podem ser utilizados numa perspectiva de desenvolvimento centrada no utilizador.

5.1 Análise de requisitos

Tal como referido, recorrendo a diagramas de Use Case, identificam-se os utilizadores do sistema, os seus objectivos na utilização do mesmo e os objectivos de usabilidade.

A figura 1.a) apresenta um extracto do modelo. Nela podem ser identificadas duas classes de utilizadores (actores): Users e Admins. A cada actor pode estar associada informação relativa a características consideradas relevantes, como nível de perícia ou outras. É importante realçar que, através do mecanismo de especialização/generalização, é possível realizar *user profiling*.

5.2 Análise de tarefas

Uma vez identificados os objectivos dos utilizadores, torna-se agora necessário descrever qual a interacção necessária para atingir cada um deles. É importante desenvolver um modelo de tarefas que permita, não só raciocinar sobre o sistema, como guiar a sua implementação. Na figura 2 apresenta-se o que poderia ser uma primeira abordagem à modelação da tarefa relativa ao objectivo Editar Sumário. O que o modelo descreve é um conjunto de comportamentos válidos.

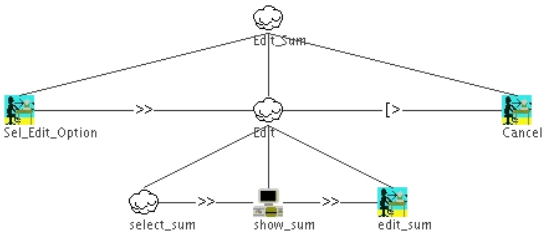

Figura 2. Modelo de tarefas em CTT

Na figura 1.b) apresenta-se o mesmo modelo de tarefas, desta feita descrito utilizando Diagramas de Actividade. Note-se a utilização de regiões interrompíveis para modelar o comportamento de cancelamento do diálogo e a utilização de pistas para associar as acções (actividades) aos diferentes actores. Neste aspecto, os Diagramas de Actividade apresentam alguma vantagem em relação às CTT, pois permite ilustrar a interacção do sistema com vários utilizadores em simultâneo de forma simples e directa. No entanto, a estruturação pretendida em sub-objectivos perde-se em grande medida. A única forma de a manter seria utilizar sub-actividades (ver actividade Select Sum) com o que isso implica de utilização de sub-diagramas, causando uma maior dificuldade na obtenção de uma visão global da tarefa.

5.3 Desenho do diálogo

Uma vez estabelecido um modelo de tarefas, torna-se necessário desenvolver uma interface que o suporte. Centrar-nos-emos na modelação do diálogo, uma vez que, cada vez mais, os IDEs fornecem suporte de qualidade para a modelação da componente gráfica. Assumindo uma interface típica organizada em écrans, o objectivo passa por identificar todos os écrans necessários a uma dada tarefa e qual o controlo de diálogo necessário para a suportar.

Na figura 1.c) apresenta-se o modelo para a tarefa anterior. O modelo identifica, não só os diferentes écrans da aplicação, como situações relevantes desses écrans. O controlo de diálogo é modelado através das transições entre estados. Estas transições permitem ainda definir quais os métodos da camada de negócio necessários à implementação da interface. O conjunto de todos os métodos derivados define a API que a lógica de negócio deverá implementar. Deste modo, esta fase constitui-se como a ponte para o desenvolvimento dessa camada aplicacional.

6. DISCUSSÃO E CONCLUSÕES

Apresentou-se neste artigo uma primeira abordagem à utilização da UML no desenvolvimento de um sistema interactivo. É proposta a utilização de diferentes diagramas da linguagem UML por forma a fornecer suporte a algumas das preocupações de um desenvolvimento centrado no utilizador e ilustra-se a abordagem com extractos de um exercício de modelação e desenvolvimento. A abordagem é complementar aos esforços de introdução de melhoramentos na linguagem, na medida em que procura explorar que o já existe por forma a avaliar as potencialidade e falhas da norma em vigor.

A abordagem apresentada é ainda preliminar, mas as novas caraterísticas da UML 2.0 apresentam um potencial interessante. Até à data a validação da abordagem tem sido efectuada através da sua aplicação à modelação/desenvolvimento de um número limitado de aplicações de pequeno/médio porte. Pretende-se continuar a análise das suas vantagens/desvantagens através da sua aplicação a sistemas de maior envergadura. Pretende-se ainda avaliar a possibilidade de realizar raciocínio sobre os modelos por forma a validar a correcção dos diferentes modelos que vão sendo desenvolvidos.

7. AGRADECIMENTOS

Este trabalho foi suportado pela FCT (Portugal) e FEDER (União Europeia) através do contrato POSC/EIA/56646/2004.

Referências

[Group 05] Object Management Group. Unified modeling language: Superstructure, v. 2.0. OMG specification formal/05-07-04, August 2005.

[Horrocks 99] I. Horrocks. *Constructing the User Interface with Statecharts*. Addison-Wesley Professional, 1999.

[Jacobson 99] I. Jacobson, G. Booch, e J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[John 03] B. E. John, L. Bass, e R. J. Adams. Communication across the HCI/SE divide: ISO 13407 and the Rational Unified Process. Em *Proceedings of HCI International 2003*, June 2003.

[Krasner 88] G. E. Krasner e S. T. Pope. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, August/September 1988.

[Nunes 01] N. J. Nunes e J. Falcão e Cunha. Wisdom — A UML based architecture for interactive systems. *Lecture Notes in Computer Science*, vol. 1946, 2001.

[Patern? 99] F. Patern? *Model Based Design and Evaluation of Interactive Applications*. Applied Computing. Springer Verlag, 1999.

[Patern? 00] F. Patern? ConcurTaskTrees and UML: how to marry them? Position paper at TUPIS'00 – a UML 2000 Workshop. York, UK, October 2000.

[Silva 02] P. P. Silva. *Object Modelling of Interactive Systems: The UMLi approach*. Tese de Doutoramento, Department of Computer Science - University of Manchester, 2002.