Formalização das Regras de Mapeamento i * para Diagrama de Classes

Abstract: A fase de levantamento de requisitos de um projeto é extremamente essencial, pois identifica todas as características que o projeto deve ter. Após essa fase, eles devem ser modelados para serem melhor compreendidos. Para modelar soluções, UML (Unified Modeling Language) é uma das linguagens mais utilizadas, mas não foi desenvolvida para capturar requisitos de domínio para qualidade. Para capturar esses requisitos, são usados modelos baseados em Engenharia de Requisitos Orientada a Objetivos (GORE), como i * (iStar). Este artigo apresenta uma formalização de regras de mapeamento i * para diagrama de classes no contexto do Model-Driven Development (MDD), com o objetivo de criar diagramas de classes mais completos, onde os requisitos de qualidade são capturados.

1. Introdução

As empresas precisam responder rapidamente às novas demandas do mercado, construindo novas soluções ou realizando manutenção em sistemas existentes. Portanto, deve atualizar seus processos e funcionar corretamente, sem descuidar dos requisitos de qualidade [1]. É necessário que ao final da fase de especificação de requisitos, todas as partes interessadas estejam perfeitamente cientes das características e do comportamento do sistema. Para isso, são propostos e utilizados diversos modelos, principalmente modelos de Unified Modeling Language (UML).

A UML é eficiente para especificar "o que" um sistema faz e "como" ele faz algo, mas não é para descrever o "por que" ele faz [2]. Ele não foi projetado para capturar os requisitos de domínio (requisitos iniciais) [3]. Para minimizar esses problemas, surgiu a Engenharia de Requisitos Orientada a Objetivos (GORE) [4]. Nas abordagens orientadas a objetivos, a engenharia de requisitos é responsável por descobrir, formular e analisar o problema a ser resolvido, bem como concluir porque o problema deve ser resolvido e quem é o responsável por resolvê-lo. [5]. A necessidade de se ter especificações de requisitos mais precisas que considerem os motivos, motivações e intenções capturados pela abordagem GORE levou à proposta inicial de modelos de regras de mapeamento i * (orientadas a objetivos) para diagramas de classes [6] em UML, que posteriormente foram estendido [7]. Desta vez, pode-se pensar em contribuir para uma possível transformação automática entre modelos. A formalização das regras de transformação entre esses modelos foi inicializada em [8] e tornou-se necessário formalizar e testar todas as regras, para permitir transformações automáticas entre os modelos (i * para diagramas de classes).

Este artigo tem como objetivo demonstrar uma transformação entre modelos no contexto de Model Driven Development (MDD) [9], que é obtida através da formalização das regras de mapeamento descritas acima. Para tanto, o presente artigo está organizado da seguinte forma: a seção 2 define resumidamente os objetivos da pesquisa; A seção 3 discutimos as contribuições científicas; A seção 4 fornece as conclusões e a seção 5 apresenta os trabalhos em andamento e futuros.

2. Objetivos da pesquisa

Este trabalho tem como objetivo demonstrar uma transformação entre modelos no contexto do MDD. Isso será alcançado por meio da formalização das diretrizes propostas por [6] e estendidas por [7]. Essas diretrizes foram criadas para mapear i * no diagrama de classes UML. O objetivo desta transformação é manter a consistência entre o sistema de software desejado e os objetivos da organização, bem como estabelecer o impacto que qualquer mudança de objetivos poderá causar no sistema e vice-versa.

3. Contribuições científicas

Usar modelos para projetar sistemas complexos é padrão nas disciplinas tradicionais da engenharia. Não podemos imaginar a construção de um edifício, uma ponte ou um carro, sem primeiro construir uma variedade de projetos e simulá-los. Os modelos nos ajudam a entender um problema complexo (e possíveis soluções) por meio da abstração.

Atualmente, o Model-Driven Development (MDD) [9] tem se mostrado uma tendência de alta reputação [10]. Na verdade, o MDD visa acelerar o desenvolvimento de software, automatizando o desenvolvimento de produtos e empregando modelos reutilizáveis ou abstrações para visualizar o código (ou o domínio do problema). Usando os modelos, ou abstrações, podemos descrever conceitos complexos de forma mais legível do que as linguagens de computador. Isso melhora a comunicação entre as partes interessadas, porque os modelos costumam ser mais fáceis de entender do que o código [11].

A contribuição mais importante deste trabalho é o desenvolvimento de uma transformação entre modelos que abrange o MDD. Essa transformação será responsável por criar os diagramas de classe mais completos, que cobrem melhores requisitos do usuário.

Essa transformação será alcançada por meio da formalização das diretrizes propostas por [6] e estendidas por [7]. Essas diretrizes são apresentadas na Tabela 1. Não faz parte do escopo deste estudo discutir essas regras, mas sim a formalização das mesmas.

O processo de formalização é mostrado na Figura 1. O processo inicia com a entrada dos dados obtidos pela ferramenta iStarTool [12]. No iStarTool, o elemento i

* é projetado e a ferramenta gera um arquivo XMI correspondente. Na segunda etapa ("transformação entre modelos"), este arquivo XMI é importado e as regras descritas na linguagem ATL [13] são aplicadas. Na última etapa, um modelo de saída é gerado contendo os elementos do diagrama de classes gerado. Este modelo de saída é outro arquivo XMI, que deve ser importado por uma ferramenta CASE para que o diagrama de classes possa ser visualizado.

4. Conclusões

A elicitação de requisitos é essencial para que um sistema seja desenvolvido com todos os recursos e funcionalidades necessários, e os modelos de aplicativos podem nos ajudar a visualizar o sistema antes de sua construção começar. Existem vários modelos, o UML é mais usado. No entanto, a UML não captura todos os requisitos do sistema, indicando "como" um sistema deve ser feito e não "por que" deve ser feito. Dentre as abordagens que atendem às necessidades do sistema, destaca-se i *.

Este trabalho apresentou a formalização do mapeamento de regras i * para o diagrama de classes, a fim de criar diagramas de classes que atendessem aos requisitos do usuário de forma mais completa.

5. Trabalho em andamento e futuro

O objetivo deste trabalho é criar diagramas de classes mais completos utilizando os recursos do MDD, abrangendo os recursos i * e com base em regras previamente criadas. Para isso, essas regras estão sendo formalizadas na linguagem ATL. Além disso, estamos realizando uma comparação entre a linguagem ATL e outras três linguagens de transformação (QVT, ETL e MOFScript).

Como trabalho futuro, está prevista a finalização da formalização das regras, bem como a criação de uma ferramenta para automatizar todo o processo. Também está prevista a adaptação da ferramenta Xgood [14]. Esta ferramenta decide quais elementos i * devem ser mapeados.