2° Relatório da Disciplina Engenharia de Software

Desenvolvimento de Software Dirigido a Modelos (MDD, MDE, MDA)

Caique Salvador Noboa

Introdução

Esse relatório foi desenvolvido como trabalho para a disciplina de Engenharia de Software do professor Paulo Cézar Stadzisz, com o objetivo de estudar o desenvolvimento de software dirigido a modelos (MDD, MDE, MDA).

A ideia por trás do MDD é desenvolver *software* a partir de modelos, isto é, gerar código automaticamente a partir de modelos e diagramas. A intenção é simplificar o processo de desenvolvimento e aumentar a produtividade. Como é imprescindível a modelagem do *software* antes do desenvolvimento, a geração automática do código a partir do modelo em teoria apenas teria vantagens.

O grande motivador é a alta demanda de *software*, o alto custo e a dificuldade de desenvolvimento. Qualquer nova maneira de criar *software* que facilite e seja mais rápido, tem grande potencial de ser abraçado pela comunidade.

Atualmente existem problemas que impedem a ampla utilização desse modo de desenvolvimento, como a falta de maturidade nas implementações existentes, a falta de ferramentas disponíveis e a resistência de desenvolvedores, que não querem mudar a forma de trabalhar (gostam de programar).

Nesse relatório são descritos mais especificamente como o modelo é pensado, as qualidades e problemas do modelo, bem como qual é a expectativa para o futuro.

1- Entendendo o Modelo

Para começar a entender o modelo, devemos começar pela modelagem, afinal, o que é modelagem de *software*?

Fazendo um paralelo com a Engenharia Civil, a modelagem de *software* é equivalente a planta de uma casa. Pode ser uma planta com alto nível de detalhamento, ou uma planta mais superficial, abstraindo diversas informações. A modelagem de *software* também pode ser mais detalhada ou mais superficial, porém é raro ver empresas utilizando modelagens muito detalhadas. Muitos enxergam a modelagem como algo apenas para ilustração.

A principal vantagem de ter um modelo altamente detalhado, é a simplificação no momento de implementar, não há muita margem para erros por parte do programador. Quanto mais detalhado, mais manual é o trabalho do programador. Essa é uma informação importante e vamos voltar nela após o próximo argumento.

A história mostra que sempre podemos criar uma linguagem mais alto nível do que as existentes. Desde Assembly, FORTRAN, C até mais recentemente com C++, Java e outras. Linguagens de alto nível facilitam a vida do programador, fazem grandes abstrações, e fazem ser possível reutilizar boa parte dos códigos, podemos inclusive ver a quantidade de bibliotecas prontas existentes em linguagens como Python e Java Script que fazem reutilização de código.

Entendendo que a história nos leva a maiores níveis de abstração, fica mais fácil imaginar que em algum momento a programação não será mais como é hoje.

Juntando a informação de modelos bem detalhados que facilitam a vida do programador, e a informação que cada vez surge linguagens mais alto nível, podemos prever que em algum momento será comum fazer modelos se converterem em código automaticamente.

O Desenvolvimento de Software Dirigido a Modelo é exatamente isso. Detalhar o máximo possível os modelos e a partir disso criar código automaticamente.

A ideia seria criar modelos executáveis, o programador nem veria o código (assim como um programador em C não vê o código em Assembly). A partir de alguns cliques e definições nos modelos, seria possível executar o *software*.

2- Motivações

O desenvolvimento de *software* é difícil, custoso e demorado. Como já dito, qualquer maneira que facilite, ou abaixe o custo ou o tempo, pode ser altamente vantajoso.

Empresas grandes e sérias levam a Engenharia de Software a sério. E isso quer dizer que fazem planejamento e modelagem de *software.* Para elas, como a modelagem já está feita, teria apenas vantagens transformar a modelagem diretamente em código. E para novos projetos, seria possível desenvolver em nível de projeto.

E caso mude o projeto, simples alterações no modelo já seriam aplicadas, aumentando muito a velocidade de resolução de problemas relacionados ao planejamento.

O aumento de abstração seria muito maior, e seria mais fácil ensinar novas pessoas a “programar”, isso definitivamente melhoraria a alta demanda por programadores, pois seria mais rápido o processo de aprendizado.

3- Benefícios

As motivações e promessas para esse modelo são altas, mas quais seriam os benefícios reais dessas implementações?

Mais especificamente, o principal benefício é aumentar a velocidade de desenvolvimento para empresas que já fazem modelagem de *software*. O maior impacto é reduzir o tempo de programação a zero, sem aumentar o tempo de planejamento e modelagem.

Mas, mesmo para empresas que não fazem modelagem, ou que fazem de maneira mais simplificada, há inúmeros benefícios. Talvez o mais impactante seja obrigar a fazer uma modelagem e planejamento antes do desenvolvimento do *software,* pois já sabemos dos benefícios do planejamento e modelagem.

De maneira geral, aumentar a abstração na programação é algo que já trás aumento de produtividade, e se a nova maneira de programar for criar modelos e diagramas, será um salto muito importante em termos de abstração.

Como o foco seria criar modelos, existiria uma grande motivação da comunidade em criar melhores formar de fazer isso. Como por exemplo, importar bibliotecas que já fazem determinadas coisas, aumentando muito a reutilização de diagramas.

E não menos importante, desenvolver em tempo de projeto trás um aumento grande na produtividade da empresa, e a parte de planejamento seria tratada com muito mais cuidado, o que também é uma grande vantagem.

4- Problemas

Apesar de todos os benefícios, existem problemas no modelo, principalmente por ser algo que não foi investido o devido tempo pela comunidade, algo que pode mudar a qualquer instante.

Para começar a lista de problemas, podemos comentar o mais óbvio, a falta de maturidade no desenvolvimento atual. Ainda não foi criado nenhuma ferramenta que realmente faça algo que justifique alguma empresa utilizar.

E as ferramentas mais comuns que fazem modelos ainda não investiram fortemente em converter seus modelos em código, o que facilitaria, pois, a comunidade já está acostumada a utilizar essas ferramentas.

Outro problema, é que nem todas as empresas gastam o devido tempo com modelos e planejamento, e muitas vezes são vistos apenas como algo extra.

Porém, acredito que o maior problema seja os desenvolvedores, eles não querem essa mudança, gostam de desenvolver com código, e não tem muito interesse em diagramas e figuras. E existe o medo de perderem o

emprego, já que geralmente não são eles que fazem os modelos, e a parte que fazem seria feita automaticamente.

5- Conclusão

Apesar de grandes promessas e grande viabilidade de desenvolver *softwares* a partir de modelos, falta a aceitação por parte dos desenvolvedores.

Com o passar dos anos e o aumento no nível das linguagens, o modelo se tornará cada vez mais viável, e os problemas podem ser minimizados, até porque não existe nenhum problema que seja uma grande desvantagem em relação a maneira de desenvolver *softwares* atualmente.

A cada ano o modelo se tornará mais viável, até quem sabe um dia, seja de fato implementado e mude totalmente a maneira de programar.

Bibliografia:

[1]- https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/mdd-termos_v01.pdf

[2]- https://www.omg.org/mda/mda_files/Model-Driven_Architecture.pdf