Universidade Tecnológica Federal do Paraná

Disciplina: Análise e Projeto Orientado a Objetos
Aluno: Jean Carlos Martins Miguel R.A: 1640593
Professor: Dr. Lucio Geronimo Valentin
UML Essencial: Um Breve Guia para Linguagem Padrão, FOWLER, Martin

1 Capítulo 2

1.1 Processo de Desenvolvimento

Logo que a UML foi concebida, os envolvidos em sua formação descobriram que, embora pudessem concordar com uma linguagem de modelagem, não iriam concordar com um processo. Como resultado, eles concordaram em deixar qualquer acordo sobre o processo para depois e restringir a UML a uma linguagem de modelagem. É importante destacar que a forma pelo qual você usa a UML depende muito do estilo de processo utilizado.

RUP (Rational Unified Process) é uma estrutura de processo que podemos utilizar com a UML, mas ele não tem nenhum relacionamento especial com a UML, já que a UML pode ser usada com qualquer processo, resumidamente o RUP é uma estratégia popular.

Processos iterativo e em cascata

Os dois processos são sempre confundidos e empregados de forma errada, o processo iterativo é visto como "elegante", já o processo em cascata é o contrário "parece vestir calça xadrez".Mas a essencial diferença é a forma como dividimos o projeto em segmentos menores. O autor cita exemplos de forma clara dos dois processos:

"O estilo em **cascata** subdivide um projeto com base nas atividades. Para construir software, você precisa realizar certas atividades: análise dos requisitos, projeto, codificação e teste. Assim, nosso projeto de um ano poderia ter uma fase de análise de dois meses, seguida de uma fase de projeto de quatro meses, após a qual viria uma fase de codificação de três meses, seguida de uma fase de teste de mais três meses".

"O estilo **iterativo** subdivide um projeto em subconjuntos de funcionalidade. Você poderia pegar um ano e dividi-lo em iterações de três meses. Na primeira iteração, você pegaria um quarto dos requisitos e faria o ciclo de vida do software completo para esse quarto: análise, projeto, código e teste. No final da primeira iteração, você teria um sistema que faria um quarto da funcionalidade necessária. Então, você faria uma segunda iteração tal que, no final de seis meses, tivesse um sistema que fizesse metade da funcionalidade.

É claro que o exposto é uma descrição simplificada, mas é a essência da diferença. Na prática, evidentemente, vazam algumas impurezas no processo".

O desenvolvimento iterativo pode ser encontrado com outros nomes: incremental, espiral, evolutivo e jacuzzi, por exemplo. Várias pessoas fazem distinções entre eles, mas não há um acordo generalizado a respeito.
A comunidade orientada a objetos é a favor do desenvolvimento iterativo e todos os envolvidos na construção da UML são a favor de pelo menos alguma forma de desenvolvimento iterativo.Mas pela percepção da prática do autor,ele afirma que o desenvolvimento em cascata ainda é a estratégia mais comum.

"Uma técnica comum no caso das iterações é usar quadro de tempo. Isso obriga uma iteração a ocorrer em um período de tempo fixo. Se você achar que não vai conseguir fazer tudo o que pretendia durante uma iteração, deve deslocar alguma funcionalidade dela; você não deve deslocar a data da iteração. A maioria dos projetos que utilizam desenvolvimento iterativo usam o mesmo período de iteração por todo o projeto; desse modo, você consegue um ritmo de construções regular".

Algumas práticas técnicas podem ajudar muito a refazer o trabalho de forma mais eficiente,o autor cita algumas, por exemplo:

Testes de regressão automatizados "ajudam, permitindo que você detecte rapidamente quaisquer defeitos que possam ter sido introduzidos na alteração de algo. Uma boa regra geral é a de que o tamanho do código de sua unidade em teste deve ser aproximadamente igual ao tamanho do seu código de produção".

Refactoring "é uma técnica disciplinada para alteração de software já existente.O refactoring trabalha usando uma série de pequenas transformações na base de código, que preservam o comportamento. Muitas dessas transformações podem ser automatizadas".

A integração contínua "mantém uma equipe em sincronismo para evitar ciclos de integração complicados . No centro disso reside um processo de construção totalmente automatizado que pode ser disparado automaticamente, quando qualquer membro da equipe verifica o código na base de código. Espera-se que os desenvolvedores façam verificações diariamente; portanto, as construções automatizadas são feitas muitas vezes por dia. O processo de construção inclui executar um grande grupo de testes de regressão a utomatizados, tal que quaisquer discrepâncias sejam capturadas rapidamente, e que possam ser corrigidas facilmente ".

Planejamentos preditivo e adaptativo

Uma estratégia preditiva procura fazer o trabalho antecipadamente no projeto, com o objetivo de gerar um maior entendimento do que precisa ser feito posteriormente.Assim,

você pode chegar a um ponto onde a última parte do projeto pode ser e stimada com um grau de precisão razoável. O planejamento a daptativo, "é o planejamento no qual a previsibilidade é vista como uma ilusão. Em vez de nos enganarmos com uma previsibilidade ilusória, devemos encarar a realidade da alteração constante e utilizar uma estratégia de planejamento que trata a alteração como uma constante em um projeto de software ".

Processos ágeis

Normalmente são processos muito adaptativos e voltados a pessoa e presume que o fator de sucesso de um projeto é a qualidade das pessoas envolvidas nesse projeto, o quão bem elas trabalham em equipe. Geralmente utilizam iterações curtas e com quadro de tempo estabelecidos entre um mês ou menos, essas metodologias não costumam associar peso aos documentos e desprezam o uso da UML no projeto, porém a utilizam no modo de esboço na maioria dos casos.Processos ágeis são caracterizados como leves já que tendem a ter pouca formalidade.

Rational Unified Process

Algumas vezes o RUP (Rational Unified Process) é mencionado junto da UML, embora sejam independentes um do outro. A primeira coisa que devemos fazer quando decidimos usar o RUP é escolher um caso de desenolvimento, ou seja, o processo que iremos utilizar em nosso projeto, sendo que casos de desenvolvimento são diferentes um do outro.
O autor cita quatro fases que todos os projetos RUP devem seguir, sendo elas:

- "A **concepção** faz uma avaliação inicial de um projeto. Normalmente, na c oncepção, você decide se vai comprometer fundos para realizar uma fase de elaboração".
- "A **elaboração** identifica os casos de uso principais do projeto e elabora o software em iterações para reorganizar a arquitetura do sistema. Ao final da elaboração, você deve ter uma boa idéia dos requisitos e um esqueleto funcional do s istema, que atue como semente de desenvolvimento. Em particular, você deve ter encontrado e resolvido os principais riscos do projeto".
- "A **construção**continua o processo, desenvolvendo funcionalidade suficiente para o lançamento".
- "A **transição** inclui várias atividades de último estágio que você não faz de forma iterativa. Isso pode incluir a distribuição para o centro de dados, treinamento dos

usuários e coisas parecidas".

Como adequar um processo a um projeto

Como os projetoss de software são distintos uns dos outros, a forma de proceder em um desenvolvimento depende de alguns fatores como, tamanho da equipe, a tecnologia utilizada no projeto, riscos, estilos de trabalho da equipe, etc, então é preciso examinar o projeto, considerar quais processos parecem ser mais próximos dos nossos objetivos e não devemos esperar que haja um processo único que funcione para todos os projetos, pois os fatores de cada empresa/projeto são diferentes.

Padrões

Os padrões focam em resultados do processo,ou seja, oferecem exemplos de projetos , ao contrário da UML que diz como expressar um projeto orientado a objetos.

Como encaixar a UML em um processo

Processo em cascata possui documentos que atuam como transferências entre fases de análise, projeto e codificação, a maioria das pessoas desenham diagramas UML em quadros em situações como reuniões por exemplo, para ajudar a transmitir suas ideias.

Análise de requisitos

Essa atividade é muito importante pois visa descobrir o quê os usuários e clientes de um produto de software querem que o sistema faça, o autor cita algumas técnicas de UML que são úteis nessa situação:

- "Casos de uso, que descrevem como as pessoas interagem com o sistema".
- "Um diagrama de classes desenhado a partir da perspectiva conceitual, o qual pode ser uma boa maneira de construir um vocabulário rigoroso do domínio".
- "Um diagrama de atividades, o qual pode exibir o fluxo de trabalho da o rganização, mostrando como o software e as atividades humanas interagem. Um diagra ma de atividades pode mostrar o contexto dos casos de uso e também os detalhes sobre como um caso de uso complicado funciona".

- "Um diagrama de estados, o qual pode ser útil, caso um conceito tenha um ciclo de vida interessante, com vários estados e eventos que mudam esses estados"

Mas o mais importante é a comunicação com os usuários e clientes quando estamos falando de análise de requisitos, ter uma comunicação clara e objetiva facilita bastante.

Projeto

Durante o processo de execução de nosso projeto, possa ser que tenham alguns diagramas técnicos, e precisamos então utilizar mais notações e sermos mais precisos a respeito dela, o autor cita algumas técnicas úteis:

- "Diagramas de classes a partir de uma perspectiva de software. Eles mostram as classes presentes no software e como elas se relacionam".
- "Diagramas de seqüência para cenários comuns. Uma estratégia valiosa é escolher o s cenários mais importantes e interessantes dos casos de uso e utilizar cartões CRC ou diagramas de seqüência para descobrir o que acontece no software".
- "Diagramas de pacotes para mostrar a organização em larga escala do software".
- "Diagramas de estados para classes com históricos de vida complexos".
- "Diagramas de distribuição para mostrar o layout físico do software".

  A implementação do código deve acompanhar os diagramas, no modo projeto.Caso precise ser feita alguma alteração, tem que ser feita uma revisão por parte dos projetista que elaboraram o projeto.

Documentação

Podemos utilizar a UML para nos ajudar a documentar o que foi feito quando tivermos construído um software, daí uma das importâncias dos diagramas UML úteis para se obter um entendimento amplo de um sistema.

Como entender o código legado

"A UML pode ajudá-lo a entender um emaranhado de códigos desconhecidos, de d uas maneiras. A construção de um esboço dos fatos principais pode funcionar como u m mecanismo gráfico de anotações, que o ajuda a capturar informações importantes, à medida

que você as aprende. Os esboços das classes principais de um pacote e suas interações mais importantes podem ajudar a esclarecer o que está acontecendo."
Cita o autor.

Como escolher um processo de desenvolvimento

O autor afirma que é a favor do processo de desenvolvimento iterativo,pois,trata-se de uma técnica fundamental, que você pode usar para expor cedo os riscos e para obter u m melhor controle sobre o desenvolvimento.

Onde encontrar mais informações

O autor cita onde encontrar mais informações para entendimento mais profundo dos assuntos mencionados anteriormente:
"Livros sobre processo de software sempre foram comuns e o surgimento do desenvolvimento ágil de software levou a muitas novas publicações. De modo geral, meu livro predileto sobre processos em geral é o [McConnell]. Ele fornece uma cobertura ampla e prática de muitos dos problemas envolvidos no desenvolvimento de software e uma longa lista de práticas úteis. Da comunidade que trabalha com desenvolvimento ágil de software, [Cockburn, agile] e [Highsmith] fornecem uma boa visão geral. Para boas orientações sobre a aplicação da UML num processo de maneira ágil, veja [Ambler]. Uma das metodologias ágeis mais populares é a Extreme Programming (XP), a XP gerou muitos livros e esse é o motivo pelo qual eu agora me refiro a ela como metodologia anteriormente leve. O ponto de partida usual é [Beck]. Embora seja escrito para XP, [Beck e Fowler] fornece mais detalhes sobre o p lanejamento de um projeto iterativo. Grande parte disso também é abordada por o utros livros sobre XP, mas se você estiver interessado apenas no aspecto do planejamento, essa seria uma boa escolha. Para obter mais informações sobre o Rational Unified Process, minha introdução predileta é [Kruchten]."

2 Capítulo 9

2.1 Casos de uso

É uma técnica com o propósito de captar os requisitos funcionais de um sistema,servem para descrever as interações comuns entre os usuários de um sistema e o próprio sistema.

"Um **cenário** é uma seqüência de passos que descreve uma interação entre um usuário e um sistema."
"Um **caso de uso** é um conjunto de cenários amarrados por um objetivo comum de usuário".
"No jargão dos casos de uso, os usuários são referidos como atores. Um **ator** é um papel que um usuário desempenha com relação ao sistema.Os atores podem ser o cliente, representante de serviço ao cliente, gerente de vendas e analista de produto. Os a tores realizam os casos de uso. Um único ator pode realizar muitos casos de uso; i nversamente, um caso de uso pode ter vários atores executando-o. Normalmente, você tem m uitos clientes; portanto, muitas pessoas podem ser o ator cliente".

Conteúdo de um caso de uso O autor afirma que "não existe nenhuma maneira padronizada para escrever o conteúdo de um caso de uso e diferentes formatos funcionam bem em diferentes casos, você começa escolhendo um dos cenários como sendo o cenário principal de sucesso (CPS). Dá início ao corpo do caso de uso escrevendo o cenário p rincipal de sucesso como uma seqüência de passos numerados. Então, pega os outros cenários e os escreve como extensões, descrevendo-os em termos de variações em relação ao cenário principal de sucesso."

Compra de um Produto

Nível do Objetivo: Nível do Mar

Cenário Principal de Sucesso:

1. O cliente navega pelo catálogo e seleciona itens para comprar
2. O cliente vai para o caixa
3. O cliente preenche o formulário da remessa (endereço de entrega; opção de entrega imediata ou em três dias)
4. O sistema apresenta a informação completa do faturamento, incluindo a remessa
5. O cliente preenche a informação de cartão de crédito
6. O sistema autoriza a compra
7. O sistema confirma imediatamente a venda
8. O sistema envia uma confirmação para o cliente por *e-mail*

Extensões:

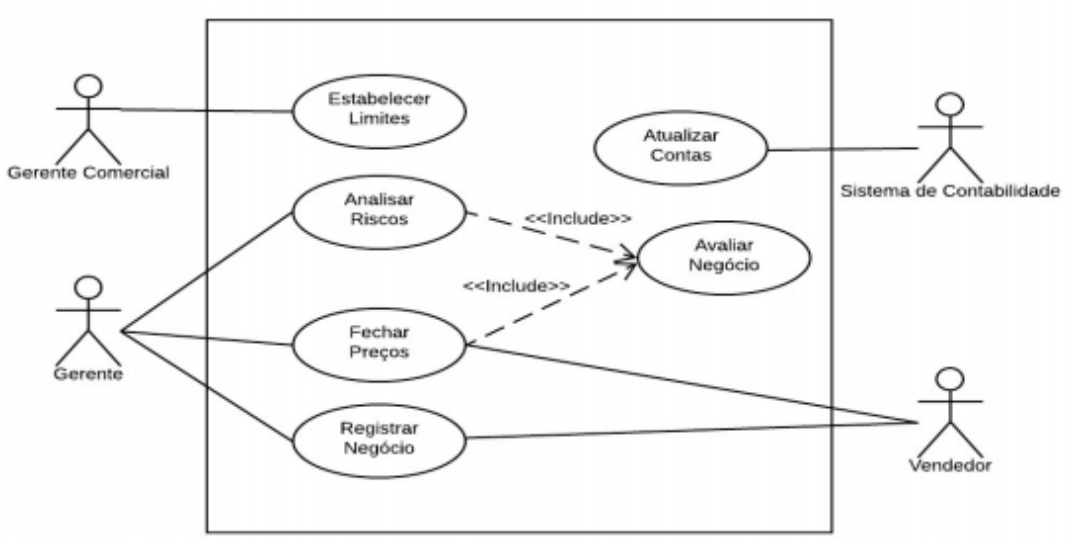
3a: Cliente regular
.1: O sistema mostra a informação atual da remessa, a informação de preço e a informação de cobrança
.2: O cliente pode aceitar ou escrever por cima desses padrões, retornando ao CPS, no passo 6

6a. O sistema falha na autorização da compra a crédito
.1: O cliente pode inserir novamente a informação do cartão de crédito ou cancelar

FIGURA 9.1 Exemplo de texto de caso de uso.

Exemplo de texto de caso de uso,(retirado do livro)

Diagramas de caso de uso



Exemplo de diagrama de caso de uso,(retirado da internet)

O diagrama de casos de uso mostra os atores, os casos de uso e os relacionamentos entre eles:

- Quais atores realizam quais casos de uso

- Quais casos de uso incluem outros casos de uso

Níveis de caso de uso

O autor cita as diferenças entre caso de uso de sistema e caso de uso de negócios: **"caso de uso de sistema** é uma interação com o software, enquanto um **caso de uso de negócio** examina como a aplicação responde ao cliente ou a um evento ".
Cita também os níveis de casos de uso com exemplos,"normalmente, os casos de uso no **nível do mar** representam uma interação distinta entre um ator principal e o sistema. Tais casos de uso transmitirão algo de valor para o ator principal e, normalmente, este levará de alguns minutos a meia hora para terminar. Os casos de uso existentes a penas porque foram incluídos pelos casos de uso de nível do mar estão em **nível de peixe**. Os casos de uso de nível mais alto (em **nível de pássaro**) mostram como os casos de uso de nível do mar se encaixam nas interações do negócio mais amplas. Os casos de uso em nível de pássaro normalmente são casos de uso de negócio, enquanto os casos de níveis do mar e de peixe são casos de uso de sistema ".

Casos de uso e funcionalidades (ou histórias)

Diversas estratégias usam as funcionalidades de um sistema – a Extreme Programming os chama de histórias de usuário – para ajudar a descrever requisitos.As funcionalidades constituem uma boa maneira de repartir um sistema para planejar um projeto iterativo, pelo qual cada iteração implementa várias funcionalidades. Os casos de uso fornecem uma narrativa de como os atores utilizam o sistema. Então, embora as duas técnicas descrevam requisitos, seus propósitos são diferentes.

Quando utilizar casos de uso

Os casos de uso são uma ferramenta de extrema importância para ajudar no entendimento dos requisitos funcionais de um sistema. Uma primeira passagem nos casos de uso

deve ser feita no início. Versões mais detalhadas dos casos de uso devem ser elaboradas apenas antes do desenvolvimento desse caso de uso. Mas devemos ter em mente que casos de uso representam uma visão externa do sistema. Então não espere quaisquer correlações entre eles e as classes dentro do sistema. Um grande perigo dos casos de uso é que as pessoas os tornam complicados demais e não conseguem prosseguir.

Onde encontrar mais informações

O autor cita onde encontrar mais informações para entendimento mais profundo dos assuntos mencionados anteriormente:
"Os casos de uso foram popularizados originalmente por Ivar Jacobson [Jacobson, OOSE]. Embora os casos de uso já existam há algum tempo, havia pouca padronização para seu uso. A UML nada diz sobre o importante conteúdo de um caso de uso e tem padronizado apenas os diagramas, muito menos importantes. Como resultado, você pode encontrar uma variedade de opiniões divergentes a respeito dos casos de uso. Nos últimos anos, entretanto, [Cockburn, use cases] tornou-se o livro padrão sobre o assunto ".