Processo Unificado

Universidade Federal do Maranhão – UFMA
Pós Graduação de Engenharia de Eletricidade
Grupo de Computação
Assunto: Introdução

Autoria:Aristófanes Corrêa Silva
Adaptação: Alexandre César M de Oliveira

1 Processo Unificado (PU)

1.1 Bibliografia

- Liliana Favre. UML and the Unified Process. IRM Press, 2003.
- Graig Larman. Applying UML and Patterns: an introduction to object-oriented analysis and design and the unified process.2003. 2 Edition
- Philippe Kruchten. The Rational Unified Process na Introduction. 2000. 2 Edition

1.2 Processos

- É um conjunto de passos que define *quem* está fazendo o *que*, *quando* e *como* para alcançar determinado objetivo.
- Na engenharia de software este objetivo é entregar de maneira eficiente e previsível, um produto capaz de atender às necessidades de seu negócio.

1.3 UML e PU

- A UML por ser apenas uma linguagem para modelagem orientada a objeto e amplamente independente de processo, indica apenas como criar e ler os modelos bem-formados, mas não aponta quais modelos serão criados nem quando deverão ser criados. Essa tarefa cabe ao processo de desenvolvimento de sistemas.

- O Processo Unificado captura algumas das melhores práticas atuais de desenvolvimento de software de forma que pode ser adaptada a uma ampla variedade de projetos e empresas.

- O Processo Unificado não pode ser considerado apenas como a união das melhores e principais características das metodologias mais populares, ele também possui características específicas, como ser orientado a casos de uso, ser centrado na arquitetura, ser interativo e incremental.

1.4 Características

  - ser orientado a casos de uso
  - ser centrado na arquitetura
  - ser interativo e incremental

- A arquitetura fornece a estrutura que guia o trabalho de cada interação, os casos de uso definem as metas e dirigem o trabalho de cada interação.

1.5 Processo orientado por casos de uso

- Um diagrama de casos de uso define a funcionalidade de um sistema para cada usuário do mesmo.
- O objetivo que é que o sistema seja desenvolvido sob a perspectiva de atender, especificamente, às necessidades de cada usuário que interage com o sistema, evitando desta forma, que o sistema possa ser desenvolvido a ponto de evitar funcionalidades desnecessárias.
- Casos de uso não são ferramentas ligadas apenas à especificação de requisitos do sistema, mas também ao projeto, implementação e testes do mesmo, ou seja, o processo de desenvolvimento do sistema é orientado por casos de uso.
- Ser orientado por casos de uso significa que o processo de desenvolvimento segue um fluxo, ou seja, o processo passa por uma série de fluxos de trabalho que derivam dos casos de uso.
- Os modelos de análise, projeto e implementação são criados pelos desenvolvedores com base no modelo de casos de uso. Este processo é conhecido como realização de casos de uso e é evidenciado durante todo o ciclo de vida do Processo Unificado.
- Casos de uso são desenvolvidos de acordo com a arquitetura do sistema. Assim, eles definem a arquitetura do sistema, e esta, por sua vez, influencia a seleção dos casos de uso. Conseqüentemente, tanto a arquitetura do sistema quanto os casos de uso, evoluem durante o ciclo de vida do sistema.

1.6 Processo centrado na arquitetura

- A arquitetura é a visão de todos os modelos que juntos representam o sistema como um todo
- O conceito de arquitetura de software engloba os aspectos estáticos e dinâmicos mais significantes do sistema. A arquitetura cresce além das necessidades do empreendimento, como percebido pelos usuários e suporte, e refletido nos casos de uso.
- "Significante" em um sistema depende do ponto de vista de cada desenvolvedor
- A arquitetura também é influenciada por muitos outros fatores, como a plataforma na qual o software será implantado, os blocos de construção reutilizáveis, requisitos de desenvolvimento e requisitos não-funcionais.
- A arquitetura é a visão do projeto do sistema como um todo, destacando suas características mais importantes, mas sem entrar em detalhes.
- O processo ajuda o arquiteto a concentrar-se nas metas corretas, como inteligibilidade, poder de recuperação para mudanças futuras e reutilização.
- A relação existente entre casos de uso e a arquitetura é que os casos de uso estão ligados a funcionalidade de um sistema e a arquitetura, por sua vez está ligada à forma deste.

- Funcionalidade e forma devem estar balanceadas para se alcançar um produto final de qualidade, ou seja, casos de uso e arquitetura devem estar ligados a tal ponto que o primeiro seja desenvolvido de acordo com a arquitetura, e esta por sua vez, forneça um ambiente para a realização de todos os requisitos dos casos de uso.

- A arquitetura de um sistema deve ser projetada a ponto de permitir que o sistema evolua, não apenas durante o início de seu desenvolvimento, mas através de gerações futuras.

- Para alcançar este objetivo, os arquitetos devem trabalhar com as funções chaves de um sistema, ou seja, os casos de uso chaves de um sistema

1.7 Processo Iterativo e Incremental

- Durante o desenvolvimento de um sistema, é prático dividi-lo em mini-projetos.

- Cada mini-projeto é uma iteração que resulta em um incremento

- Iterações referem-se às etapas do fluxo de trabalho, e incrementos, ao avanço no desenvolvimento do produto.

- Para serem mais efetivas, as iterações devem ser controladas, ou seja, devem ser executadas de modo planejado.

- Os desenvolvedores baseiam suas escolhas, a respeito do que será implementado em uma interação:

- A iteração lida com um grupo de casos de uso que juntos representam o funcionamento do sistema em desenvolvimento.
- A iteração lida com os riscos mais significativos do empreendimento.

- Iterações sucessivas constroem um conjunto de artefatos a partir do estado em que estes foram deixados ao término da iteração passada.
- Partindo dos casos de uso, o mini-projeto prossegue através dos fluxos de trabalho subseqüentes (análise, projeto, implementação e teste) até alcançar a forma de código executável.
- Em cada iteração, os desenvolvedores identificam e especificam os casos de uso relevantes, criam o projeto de acordo com a arquitetura escolhida, implementam o projeto em componentes e verificam se os componentes satisfazem os casos de uso
- Se uma iteração alcançar seu objetivo, o que geralmente ocorre, o desenvolvimento prossegue com a próxima iteração.
- Se uma iteração não alcançar seu objetivo, os desenvolvedores devem revisar as decisões tomada anteriormente e tentar uma nova abordagem.
- Para conseguir maior economia no desenvolvimento, a equipe de projeto tenta selecionar apenas as iterações necessárias para alcançar as metas predefinidas.
- Um projeto bem sucedido avança por um curso com o mínimo de desvios do curso planejado inicialmente pelos desenvolvedores.

- Entretanto, o surgimento de problemas imprevistos acarreta no aumento do número de iterações ou alterações na seqüência da mesmas, fazendo com que o processo de desenvolvimento tome mais tempo e dedicação.
- Minimizar a possibilidade de surgimento de problemas imprevistos é uma das metas da redução de riscos
- Um processo iterativo controlado tem várias vantagens:
  - Reduzem o risco de custo para as despesas em um único incremento. Se os desenvolvedores precisarem repetir a iteração, será perdido apenas o esforço dedicado a um iteração, não ao produto como um todo
  - Reduzem o risco de violação de prazos. O fato de identificar problemas no início do desenvolvimento, permite que os desenvolvedores aloquem tempo para resolve-los sem extrapolar prazos
  - Aceleram o tempo de desenvolvimento do projeto como um todo, pelo fato de desenvolvedores trabalharem de maneira mais eficiente em cima de resultados de escopo pequeno e claro.
- Um processo iterativo controlado tem várias vantagens:
  - Aceleram o tempo de desenvolvimento do projeto como um todo, pelo fato de desenvolvedores trabalharem de maneira mais eficiente em cima de resultados de escopo pequeno e claro.

- Reconhecem uma realidade muitas vezes ignorada, que a necessidade dos usuários e requisitos correspondentes não podem ser completamente definidos no início do desenvolvimento. Eles devem ser refinados em sucessivas iterações. Este modo de operação torna mais fácil a adaptação do sistema a mudanças dos requisitos.