

Processos de Software

O objetivo deste capítulo é introduzir os conceitos do processo de software - um conjunto coerente de atividades para produção de software.

• Um processo de software é um conjunto de atividades relacionadas que levam à produção de um sistema de software.

↳ Embora existam muitos processos de software diferentes, todos eles devem incluir, de alguma forma, as quatro atividades fundamentais da engenharia de software.

1 - Especificação: A funcionalidade do software e as restrições sobre a sua operação devem ser definidas.

2 - Desenvolvimento: O software deve ser produzido para atender à especificação.

3 - Validação: O software deve ser validado para garantir que atenda ao que o cliente deseja.

4 - Evolução: O software deve evoluir para atender às mudanças nas necessidades dos clientes.

Essas atividades são complexas por si só e incluem subatividades como validação dos requisitos, projeto da arquitetura e teste de unidade.

1 - Uma atividade de processo resulta em produtos ou em entregas. Por exemplo, o resultado da atividade de projeto da arquitetura pode ser um modelo da arquitetura de software.

2 - Os papéis refletem as responsabilidades das pessoas envolvidas no processo. Entre os exemplos de papéis, temos os de gerente de projeto, do gerente de

configuração e do programador.

3- Há condições que devem ser mantidas antes ou depois de uma actividade do processo ter sido aprovada ou um produto ter sido produzido. Antes de começar o projeto de arquitetura, por exemplo, uma precondição poderia ser a de que o consumidor tenha aprovado todos os requisitos; depois que essa actividade for concluída, uma pos-condição poderia ser a de que os modelos em UML descrevendo a arquitetura fossem revisados.

- o projeto é parte inerente a todos os processos. Os processos dirigidos por planos são aqueles em que todas as actividades são planejadas antecipadamente e progredem em relação ao que foi planejado.
- / —

Modelos de Processo de Software

Modelos de processo genérico

1- Modelo em cascata: Representa as actividades fundamentais do processo, como especificação, desenvolvimento, validação e evolução, na forma de fases de processo distintas, como especificação de requisitos, projeto de software, implementação e testes.

2- Desenvolvimento incremental: Intercala as actividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), com cada uma delas acrescentando funcionalidade à versão anterior.

3 - Integração e configuração: Baseia-se na disponibilidade de componentes ou sistemas reusáveis. O processo de desenvolvimento de sistemas se concentra na configuração desses componentes, para que sejam utilizadas em um novo contexto, e na integração delas em um sistema.

→ Os sistemas críticos em segurança em, por exemplo, geralmente são desenvolvidos a partir de um processo em cascata, já que é necessária uma grande quantidade de análise e de documentação antes de começar sua implementação.

Várias tentativas têm sido feitas para desenvolver modelos de processo genéricos universais baseados em todos esses modelos genéricos. Um dos mais conhecidos é o Rational Unified Process - RUP, que foi desenvolvido pela Rational. O RUP é um modelo flexível, que pode ser instanciado de diferentes maneiras para criar processos que se assemelhem a qualquer um dos modelos de processo genéricos discutidos aqui.

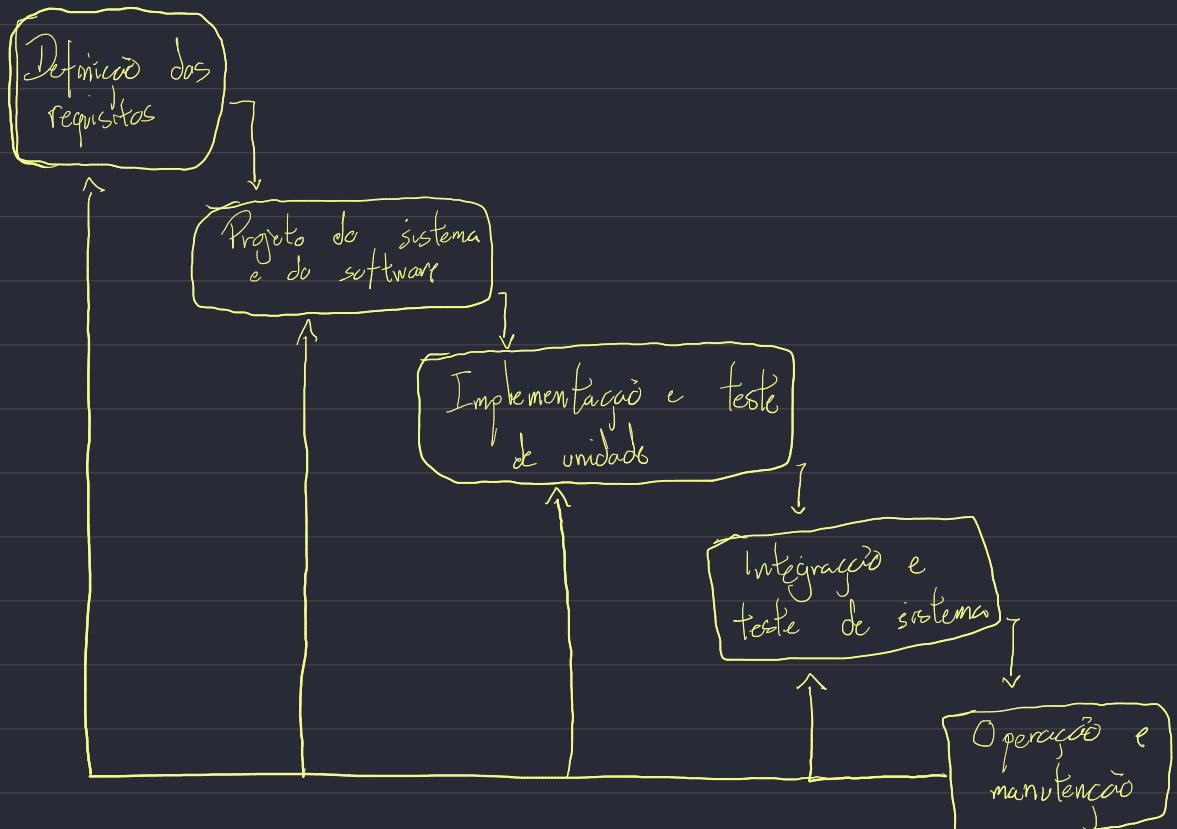
O RUP é descrito normalmente a partir de três perspectivas: uma dinâmica, que mostra as fases do modelo no tempo; uma estática, que mostra as atividades do processo; e uma prática, que sugere práticas a serem utilizadas no processo. As fases do RUP são a concepção, em que se estabelece um business case para o sistema; a elaboração, em que são desenvolvidos os requisitos e a arquitetura; a construção, em que o software é implementado; e a transição, em que o sistema é implementado.

Modelo em Cadeia

O primeiro modelo de processo de desenvolvimento de software a ser

publicado é derivado dos modelos utilizados na engenharia de grandes sistemas militares. Ele apresenta o processo de desenvolvimento de software como uma série de estágios.

Devido à cascata de uma fase para outra, esse modelo é conhecido como modelo em cascata ou ciclo de vida do software. O modelo em cascata é um exemplo de processo dirigido por plano.



1 - Análise e definição de requisitos: Os serviços, as restrições e as metas do sistema são estabelecidos por meio da consulta aos usuários. Depois, eles são definidos em detalhes e servem como uma especificação do sistema.

2 - Projeto do sistema e do software: O processo de projeto do sistema reporta os requisitos de sistemas de hardware e de software, e estabelece uma arquitetura global do sistema. O projeto de software envolve a

Identificação e a descrição das abstrações fundamentais do sistema de software e seus relacionamentos.

3 - Implementação e teste de unidade: Durante essa etapa, o projeto de software é realizado como um conjunto de programas ou unidades de programa. O teste de uma unidade envolve a verificação de cada unidade, conferindo se satisfazem a sua especificação.

4 - Integração e teste de sistema: As unidades de programa ou os programas são integrados e testados como um sistema completo a fim de garantir que os requisitos de software tenham sido cumpridos. Após os testes, o sistema de software é entregue aos clientes.

5 - Operação e manutenção: Normalmente, essa é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve corrigir os erros que não foram descobertos nas primeiras fases do ciclo de vida, melhorar a implementação das unidades do sistema e aperfeiçoar os serviços do sistema à medida que novos requisitos são descobertos.

A princípio, o resultado de cada fase no modelo cascata consiste em um ou mais documentos que são aprovados. A fase seguinte não deve começar até que a fase anterior tenha começado.

- No desenvolvimento de hardware, que envolve altos custos de produção, isso faz sentido.

- No entanto, no desenvolvimento de software, esses estágios se sobreponem e alimentam uns aos outros com informações. O processo de software,

na prática, nunca é um modelo linear simples, pois envolve feedback entre as fases.

- O modelo cascata é adequado somente para alguns tipos de sistema, tais como:

1 - Sistemas embarcados, nos quais o software deve interagir com sistemas de hardware. Em virtude da inflexibilidade do hardware, normalmente não é possível postergar as decisões sobre a funcionalidade do software até que ele seja implementado.

2 - Sistemas críticos, nos quais há necessidade de ampla análise da segurança (safety) e segurança da informação (security) da especificação e do projeto do software. Nesses sistemas, os documentos de especificações e de projeto devem estar completos para que a análise seja possível. Geralmente, é muito caro corrigir, durante a fase de implementação, os problemas relacionados à segurança na especificação e no projeto.

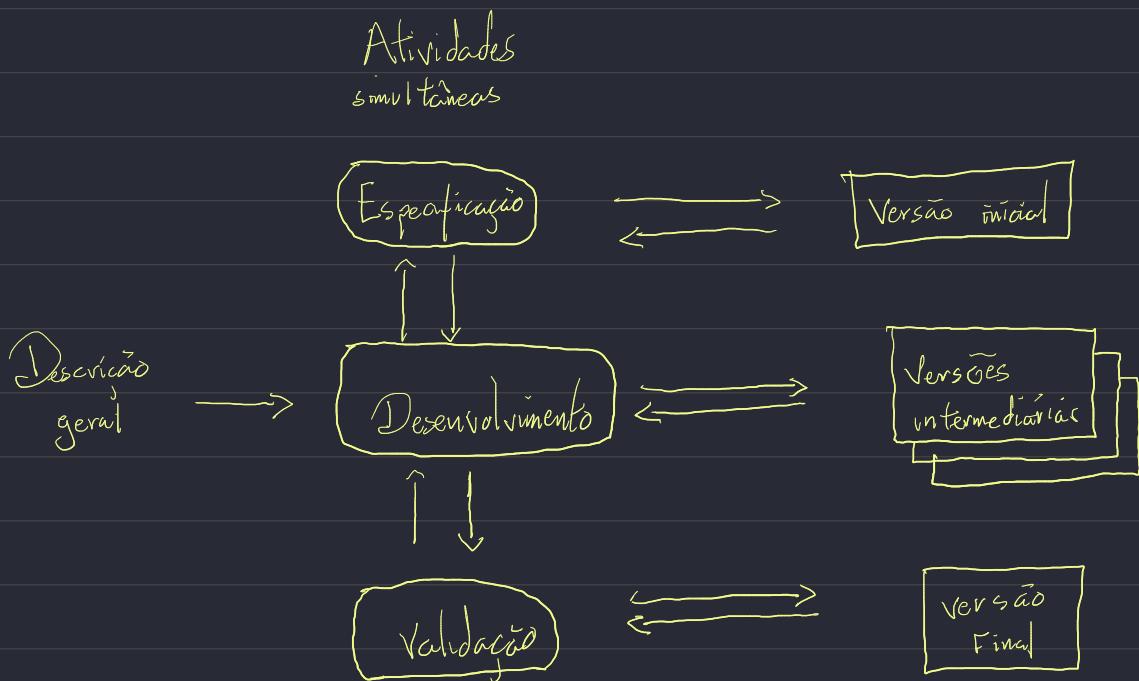
3 - Grandes sistemas de software, que fazem parte de sistemas de engenharia mais amplos, desenvolvidos por várias empresas parceiras. O hardware nos sistemas pode ser desenvolvido a partir de um modelo similar, e as empresas preferem usar um modelo comum para o hardware e o software. Além disso, quando várias empresas estão envolvidas, podem ser necessárias especificações completas para permitir o desenvolvimento independente dos diferentes subsistemas.

- O modelo em cascata não é recomendado para situações em que a comunicação informal do time é possível e nas quais os requisitos de software mudam rapidamente.



1 Desenvolvimento Incremental

O desenvolvimento incremental se baseia na ideia de desenvolver uma implementação inicial, obter feedback dos usuários ou terceiros e fazer o software evoluir através das várias versões, até alcançar o sistema necessário.



O modelo incremental, em alguma de suas formas, é atualmente a abordagem mais comum para o desenvolvimento de aplicações e produtos de software. Essa abordagem pode ser dirigida por plano ou ágil; na maioria das vezes, uma mistura de ambas.

O modelo incremental tem grandes vantagens em relação ao modelo em cascata:

1 - O custo de implementação das mudanças nos requisitos é reduzido. A quantidade de análise e documentação que precisa ser referida é significativamente menor do que a necessária ao modelo em cascata.

2 - É mais fácil obter feedback do cliente sobre o trabalho de

desenvolvimento. Os clientes podem comentar as demonstrações de software e ver o quanto foi implementado. Para eles, é mais difícil julgar o progresso a partir dos documentos do projeto (design) de software.

3- A entrega e a implantação antecipadas de um software útil para o cliente são possíveis, mesmo se toda a funcionalidade não tiver sido incluída. Os clientes são capazes de usar o software e de obter valor a partir dele mais cedo do que com um processo em cascada.

Problemas no desenvolvimento incremental

Embora o desenvolvimento incremental tenha muitas vantagens, ele não está livre de problemas. A principal dificuldade é o fato de que as grandes organizações têm procedimentos burocráticos que evoluíram ao longo do tempo, o que pode levar a uma incompatibilidade entre esses procedimentos e um processo iterativo ou ágil mais informal.

Do ponto de vista da gestão, a abordagem incremental tem dois problemas:

1. O processo não é visível. Os gerentes precisam de resultados regulares para medir o progresso. Se os sistemas forem desenvolvidos rapidamente, não é econômico produzir documentos que reflitam cada versão do sistema.

2. A estrutura do sistema tende a se degradar à medida que novos incrementos são adicionados. Mudanças regulares deixam o código bagunçado

Uma vez que novas funcionalidades são adicionadas de qualquer maneira possível. Fica cada vez mais difícil e caro adicionar novas características a um sistema. Para reduzir a degradação estrutural e a bagunça generalizada no código, os métodos ágeis sugerem que se refatore (melhore e reestruture) o software regularmente.

Integração e configuração

Na maioria dos projetos há algum reuso de software. Com frequência, isso acontece informalmente quando as pessoas que trabalham no projeto conhecem ou procuram algum código similar ou necessário. Eles procuram por esse código, modificam-no conforme a necessidade e integram-no ao novo código que desenvolveram.

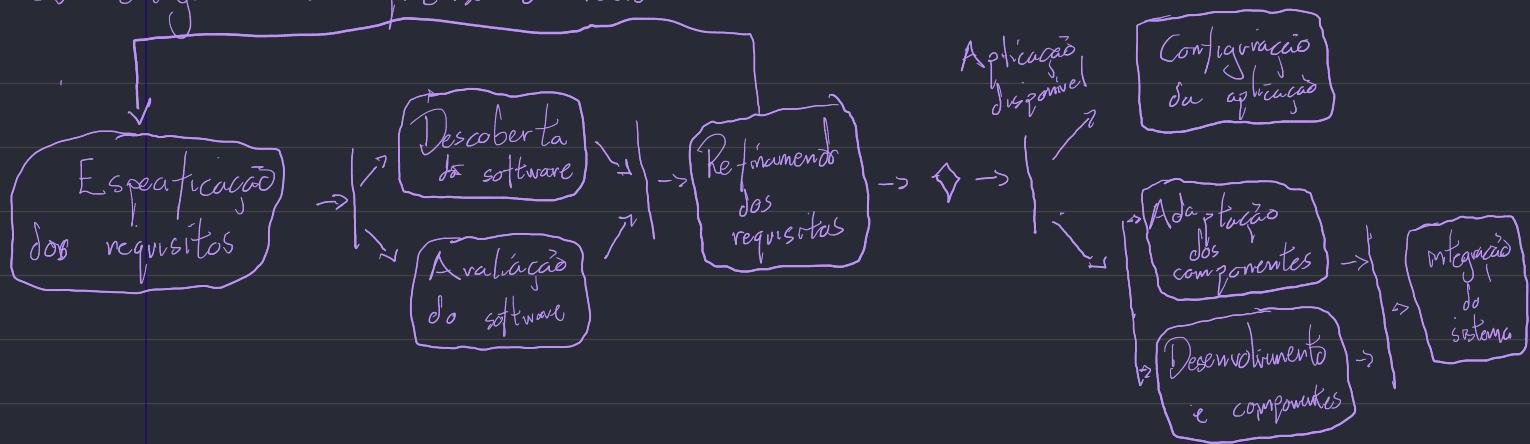
Três tipos de componentes de software são reusados frequentemente:

1. Sistemas de aplicações stand-alone configurados para utilização em um ambiente particular. Esses sistemas são de uso geral e possuem muitas características, mas precisam ser adaptados para uso em uma aplicação específica.

2. Coleções de objetos desenvolvidos como um componente ou como um pacote a ser integrado a um framework de componentes, como Java Spring framework.

3. Web services desenvolvidos de acordo com os padrões de serviço e que estão disponíveis para uso remoto na internet.

Os estágios nesse processo de reuso são:



1- Especificação dos requisitos: Os requisitos iniciais do sistema são propostos. Eles não precisam ser elaborados em detalhes, mas devem incluir descrições breves dos requisitos essenciais e das características de sistema desejáveis.

2- Descoberta e avaliação do software: Com base em uma descrição dos requisitos de software, é feita uma busca pelos componentes e sistemas que fornecem a funcionalidade necessária. Os candidatos são avaliados para ver se satisfazem os requisitos essenciais e se são genericamente adequados ao uso no sistema.

3- Refinamento de requisitos: Nesse estágio, os requisitos são definidos com base nas informações dos componentes reusáveis e das aplicações que foram descobertas. Os requisitos são modificados para refletir os componentes disponíveis e a especificação do sistema é redefinida. Onde as modificações forem impossíveis, a atividade da análise de componentes pode ser reintroduzida para procurar soluções alternativas.

4- Configuração da aplicação: Se estiver disponível uma aplicação de prateleira que satisfaça os requisitos, ela pode ser configurada para utilização a fim de criar o novo sistema.

5- Adaptação e integração dos componentes: Se não houver uma aplicação

de prateleira, componentes reusáveis podem ser modificados ou novos componentes podem ser desenvolvidos, visando a integração posterior ao sistema.

— / —

Atividades do Processo

Os processos de software reais são sequências intercaladas de atividades técnicas, colaborativas e gerenciais, cujo objetivo global é especificar, projetar, implementar e testar um sistema de software.

As quatro atividades de processo básica - a especificação, o desenvolvimento, a validação e a evolução - são organizadas de modo distinto em diferentes processos de desenvolvimento. No modelo em cascata, elas são organizadas em sequência; no desenvolvimento incremental, são intercaladas.