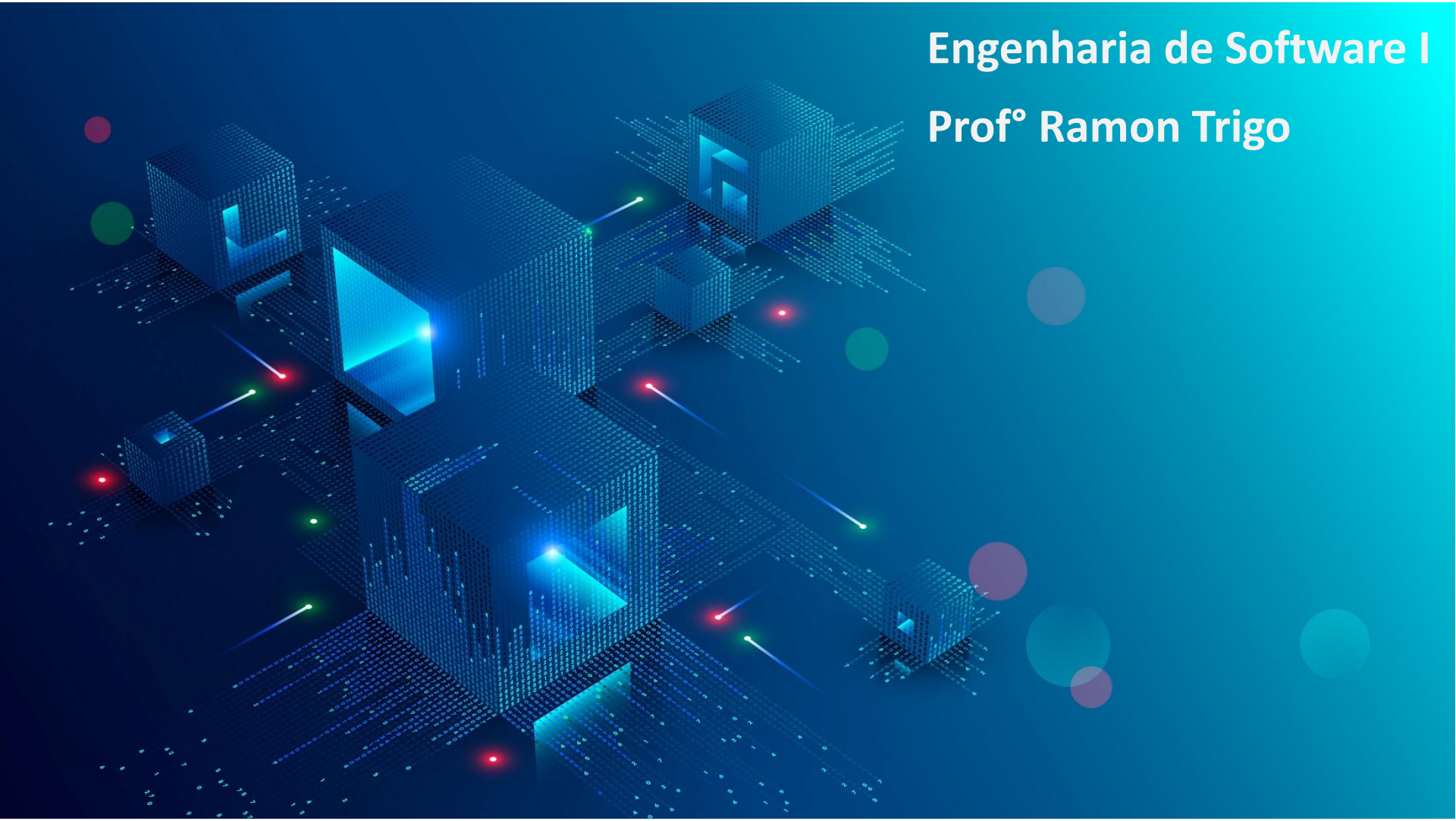
Fatec Registro

Centro Paula Souza

SÃO PAULO GOVERNO DO ESTADO

Engenharia de Software I

Prof° Ramon Trigo



Modelos de Processo de Software

Modelo de processo de software — às vezes chamado de ciclo de vida do desenvolvimento de software (ou modelo SDLC, do inglês Software Development Life Cycle) — é uma representação simplificada de um processo de software.

Cada modelo representa um processo a partir de uma perspectiva particular e. desse modo, fornece apenas informações parciais sobre esse processo.

Por exemplo, um modelo de atividades do processo mostra as atividades e sua sequência, mas não os papéis das pessoas envolvidas nelas.

Modelos Genéricos

Modelo em cascata.

Representa as atividades 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.

Modelos Genéricos

Desenvolvimento incremental.

Intercala as atividades de especificação, desenvolvimento e validação.

O sistema é desenvolvido como uma série de versões (incrementos), com cada uma delas acrescentando funcionalidade à versão anterior.

Modelos Genéricos

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 utilizados em um novo contexto, e na integração deles em um sistema.

Modelos Genéricos

O processo correto depende do cliente e dos requisitos que regulam o software, do ambiente em que esse software será utilizado e do tipo de software que está sendo desenvolvido.

Os sistemas críticos em segurança, por exemplo, são normalmente 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. Por sua vez. os produtos de software são. atualmente, desenvolvidos a partir de um modelo de processo incremental.

Os sistemas de negócio são desenvolvidos, cada vez mais, por meio da configuração dos sistemas preexistentes e da integração entre eles, a fim de criar um novo sistema com a funcionalidade exigida.

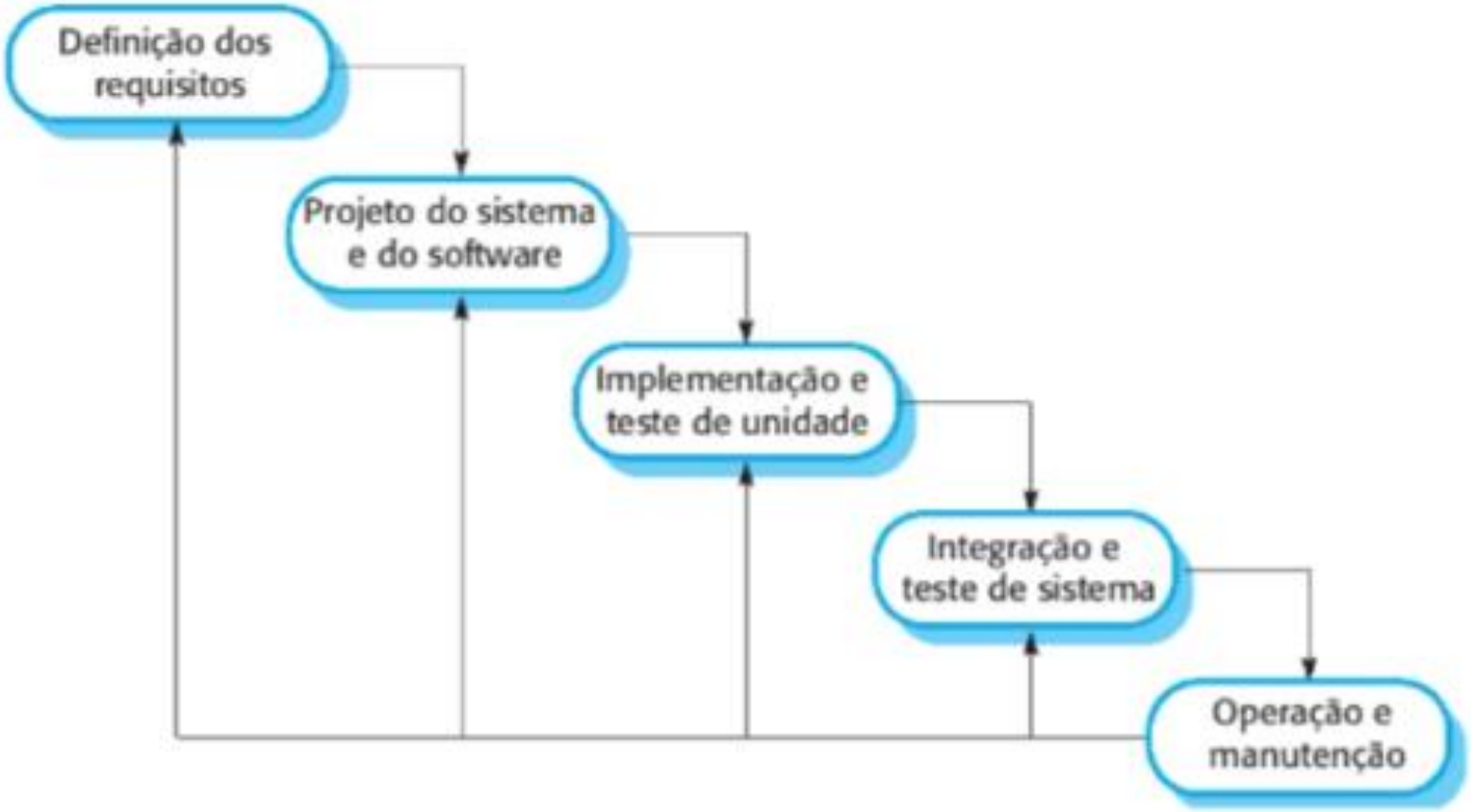
Modelo Cascata

O primeiro modelo de processo de desenvolvimento de software a ser publicado é derivado dos modelos utilizados na engenharia de grandes sistemas militares.

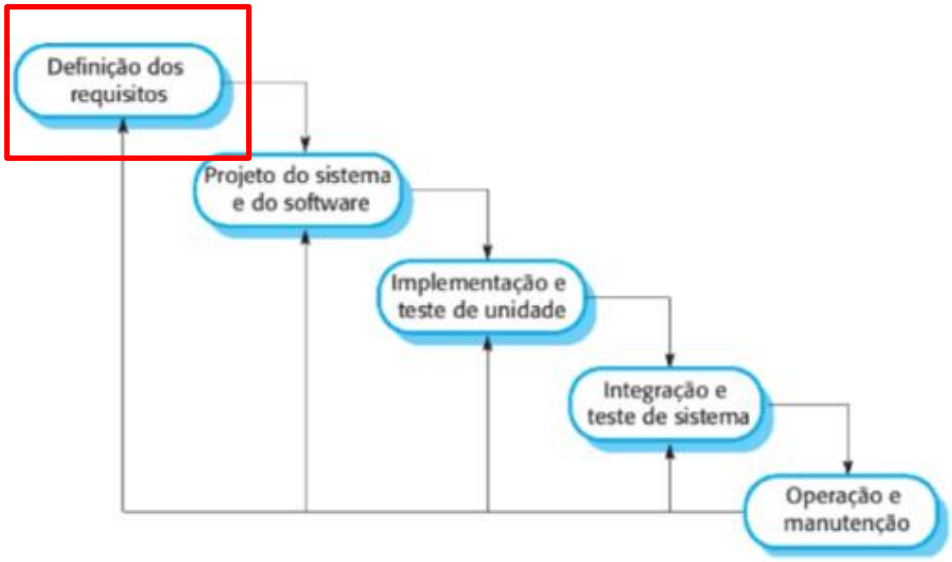
Ele apresenta o processo de desenvolvimento de software como uma série de estágios, conforme . Devido à cascata de uma fase para outra, esse modelo é conhecido como modelo em cascata ou ciclo de vida do software. O modelo em cascata é um exemplo de processo dirigido por plano.

A princípio, pelo menos, é necessário planejar e criar um cronograma de todas as atividades de processo antes de começar o desenvolvimento do software.

Modelo Cascata



Modelo Cascata

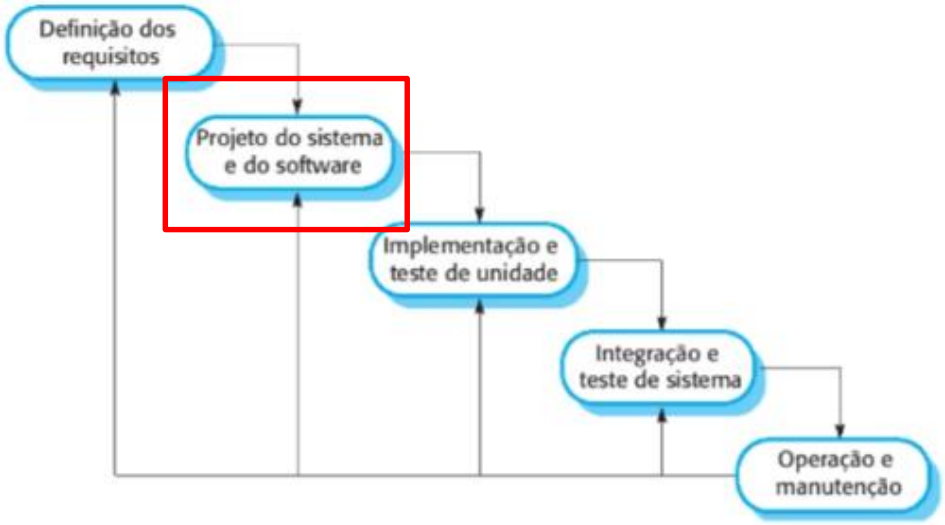


Análise e definição dos requisitos.

Os serviços, as restrições e as metas do sistema são estabelecidos por meio de consulta aos usuários.

Depois, eles são definidos em detalhes e servem como uma especificação do sistema

Modelo Cascata

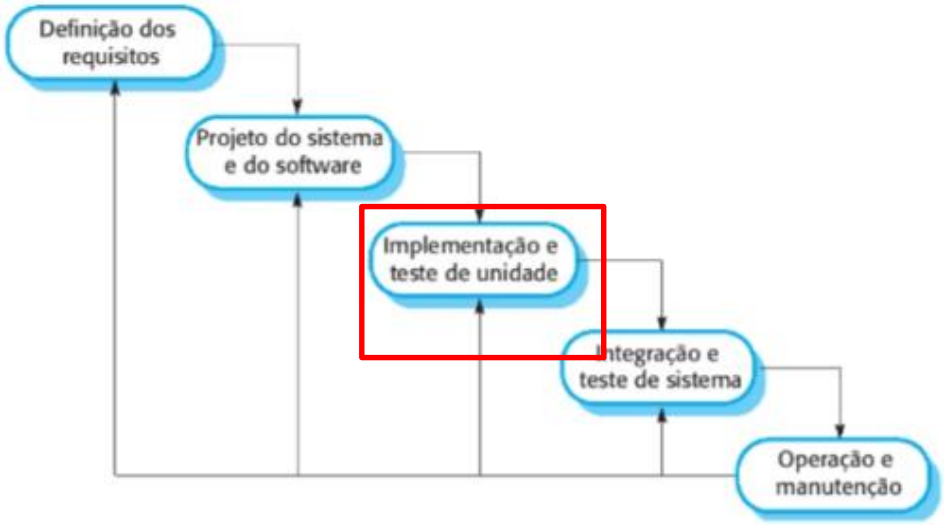


Projeto do sistema e do software.

O processo de projeto do sistema reparte os requisitos entre 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

Modelo Cascata

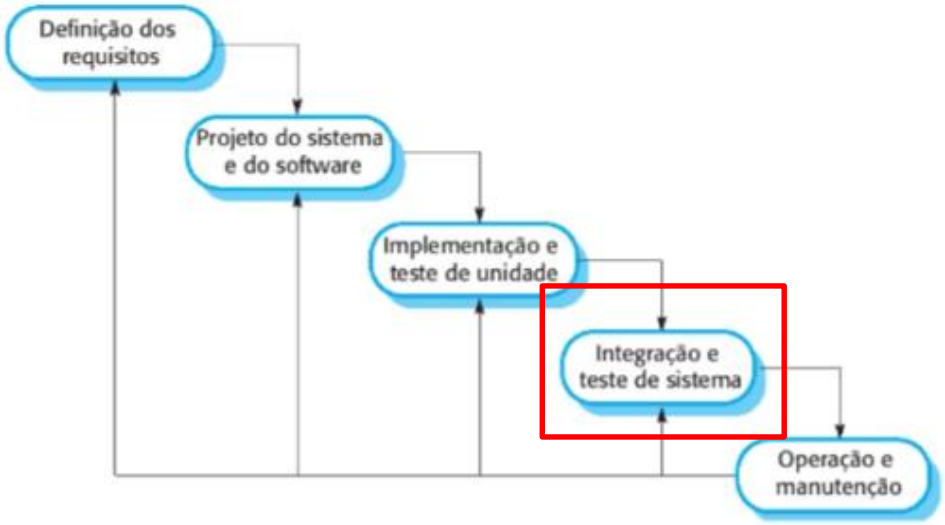


Implementação e teste de unidade.

Durante essa etapa, o projeto do software é realizado como um conjunto de programas ou unidades de programa.

O teste de unidade envolve a verificação de cada unidade, conferindo se satisfazem a sua especificação.

Modelo Cascata

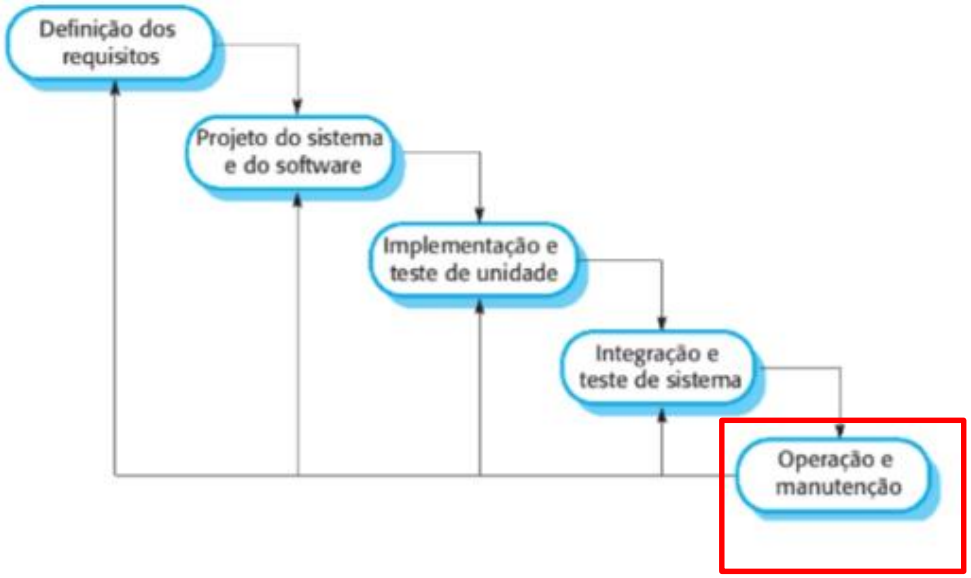


Integração e teste de sistema.

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 ao cliente.

Modelo Cascata



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.

Modelo Cascata

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

No desenvolvimento de hardware, que envolve altos custos de produção, isso faz sentido. No entanto, no desenvolvimento de software, esses estágios se sobrepõem e alimentam uns aos outros com informações. Durante o projeto (design), são identificados problemas com os requisitos; durante a codificação, são encontrados problemas com o projeto, e assim por diante.

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

Modelo Cascata

Durante a fase final do ciclo de vida (operação e manutenção), o software começa a ser utilizado.

Erros e omissões nos requisitos originais do software são descobertos, falhas de programação e de projeto emergem e a necessidade de novas funcionalidades é identificada.

Por isso, o sistema deve evoluir a fim de continuar sendo útil. Realizar essas mudanças (manutenção de software) pode envolver a repetição dos estágios de processo prévios

Modelo Cascata

A necessidade de comprometimento inicial e retrabaIho quando as mudanças são feitas significa que o modelo em cascata é adequado somente para alguns tipos de sistema, tais como:

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.

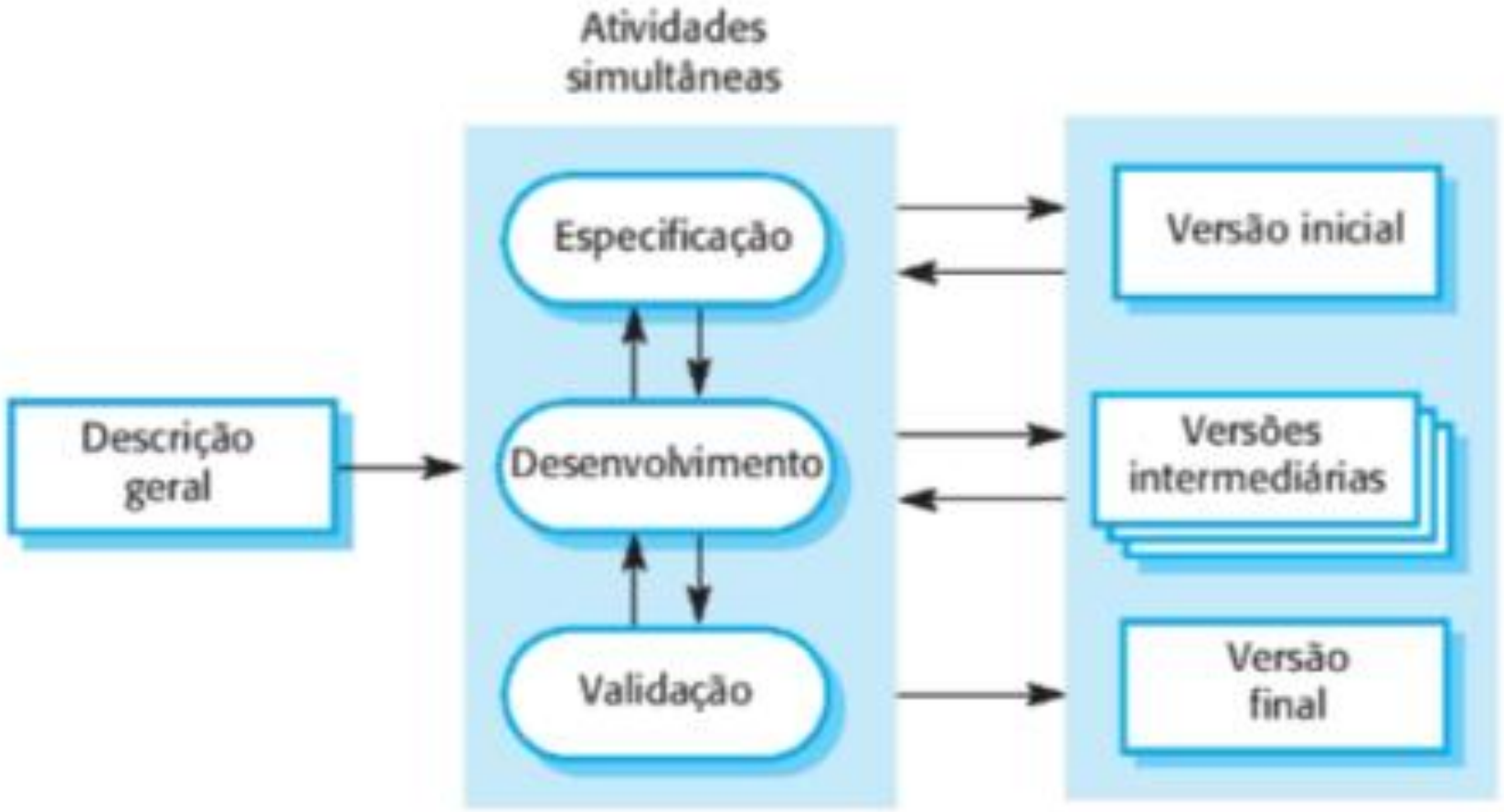
O modelo em cascata não é recomendado para situações em que a comunicação informal do time é possível e nas quais os requisitos de software mudam rapidamente

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 de várias versões, até alcançar o sistema necessário

As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, com feedback rápido ao longo de todas elas.

Desenvolvimento Incremental



Desenvolvimento Incremental

O desenvolvimento incremental, em alguma de suas formas, é atualmente a abordagem mais comum para o desenvolvimento de aplicações e produtos de software.

Cada incremento ou versão do sistema incorpora parte da funcionalidade necessária para o cliente. Geralmente, os incrementos iniciais incluem a funcionalidade mais importante ou a mais urgente.

Isso significa que o cliente ou usuário pode avaliar o sistema em um estágio relativamente precoce no desenvolvimento para ver se ele entrega o que é necessário.

Desenvolvimento Incremental

Se não entrega, então o incremento atual precisa ser alterado e, possivelmente. uma nova funcionalidade deve ser definida para os incrementos posteriores

Vantagens do Modelo Incremental

O desenvolvimento incremental tem três 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 refeita 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.

Vantagens do Modelo Incremental

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 cascata.

Problemas do 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

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 ao necessário

Elas procuram por esse código, modificam-no conforme a necessidade e integram-no ao novo código que desenvolveram.

Esse reuso informal ocorre independentemente do processo de desenvolvimento utilizada

Três tipos de componentes de software são reusados frequentemente:

1. Sistemas de aplicação 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.

Obs. Framework conjunto de código para desenvolvimento de software

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

Obs. Web Service disponível para comunicar diversas plataformas.

Software Baseado em Reuso

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.

3. Refinamento dos 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.

Software Baseado em Reuso