Modelo Cascata ou Clássico

INTRODUÇÃO

O modelo clássico ou cascata, que também é conhecido por abordagem "top-down", foi proposto por Royce em 1970. Até meados da década de 1980 foi o único modelo com aceitação geral. Esse modelo foi derivado de modelos de actividade de engenharia com o fim de estabelecer ordem no desenvolvimento de grandes produtos de software. Comparado com outros modelos de desenvolvimento de software, este é mais rígido e menos administrativo.

O modelo cascata é um dos mais importantes modelos, e é referência para muitos outros modelos, servindo de base para muitos projectos modernos. A versão original deste modelo foi melhorada e retocada ao longo do tempo e continua sendo muito utilizado hoje em dia.

Grande parte do sucesso do modelo cascata está no facto dele ser orientado para documentação. No entanto deve salientar-se que a documentação abrange mais do que arquivo de texto, abrange representações gráficas ou mesmo simulação.

Uma abordagem incorporando processos, métodos e ferramentas deve ser utilizada pelos criadores de software. Esta abordagem é muitas vezes designada de Abordagem do Processo de Desenvolvimento. Existem três abordagens de modelos de processo de desenvolvimento de software. Elas tentem colocar ordem numa actividade inerentemente caótica.

Uma vez definido o modelo de ciclo de desenvolvimento, existem três abordagens para implementá-lo:

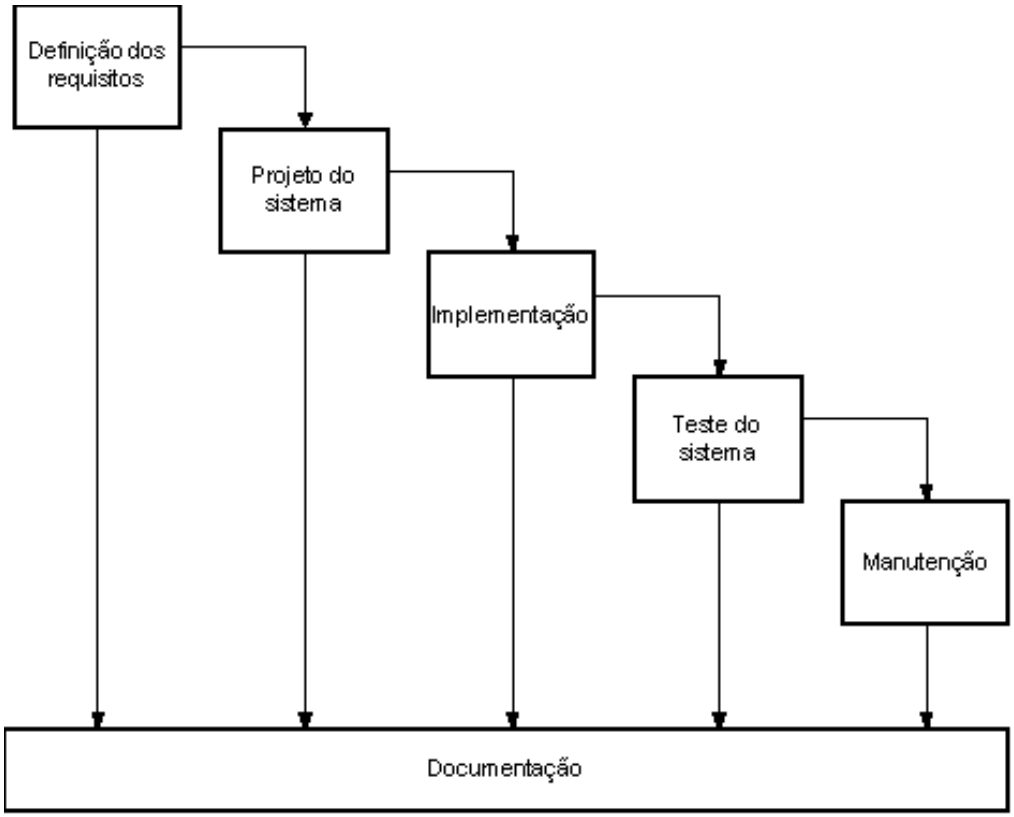
Cascata pura;

Incremental;

Evolucionária.

Toda esta secção constitui uma interpretação do disposto na referência [FAI96].

Descrição do modelo



O modelo Cascata é um modelo de engenharia projectado para ser aplicado no desenvolvimento do software. A ideia principal que o dirige é que as diferentes etapas de desenvolvimento seguem uma sequência:

a saída da primeira etapa "fluí" para a segunda etapa e a saída da segunda etapa "fluí" para a terceira e assim por diante. As actividades a executar são agrupadas em tarefas, executadas sequencialmente, de forma que uma tarefa só poderá ter início quando a anterior tiver terminado.

O modelo em cascata tem a vantagem que só avança para a tarefa seguinte quando o cliente valida e aceita os produtos finais da tarefa actual. O modelo pressupõe que o cliente participa activamente no projecto e que sabe muito bem o que quer. Este modelo minimiza o impacto da compreensão adquirida no decurso de um projecto, uma vez que se um processo não pode voltar atrás de modo a alterar os modelos e as conclusões das tarefas anteriores, é normal que as novas ideias sobre o sistema não sejam aproveitadas. Numa tentativa de resolver este tipo de problema foi definido um novo tipo de processo baseado no clássico em cascata, designado por *modelo em cascata revisto*, cuja principal diferença consiste em prever a possibilidade de a partir de qualquer tarefa do ciclo se poder regressar a uma tarefa anterior de forma a contemplar alterações funcionais e/ou técnicas que entretanto tenham surgido, em virtude de um maior conhecimento que entretanto se tenha obtido. O risco desta abordagem é que, na ausência de um processo de gestão do projecto e de controlo das alterações bem definido, podemos passar o tempo num ciclo infinito, sem nunca se atingir o objectivo final, ou seja disponibilizar o sistema a funcionar.

As Diferentes Etapas de Desenvolvimento

Análise e definição dos requisitos

Nesta etapa, estabelecem-se os requisitos do produto que se deseja desenvolver, o que consiste usualmente nos serviços que se devem fornecer, limitações e objetivos do software. Sendo isso estabelecido, os requisitos devem ser definidos de uma maneira apropriada para que sejam úteis na etapa seguinte. Esta etapa inclui também a documentação e o estudo da facilidade e da viabilidade do projecto com o fim de determinar o processo de início de desenvolvimento do projecto do sistema; pode ser vista como uma concepção de um produto de software e também como o início do seu ciclo de vida.

Projecto do sistema

O projecto do sistema é um processo de vários passos que se centraliza em quatro atributos diferentes do sistema: estrutura de dados, arquitectura do software, detalhes procedais e caracterização das interfaces. O processo de projecto representa os requisitos de uma forma que permita a codificação do produto (é uma prévia etapa de codificação). Da mesma maneira que a análise dos requisitos, o projecto é documentado e transforma-se em uma parte do software.

Implementação

Esta é a etapa em que são criados os programas. Se o projecto possui um nível de detalhe elevado, a etapa de codificação pode implementar-se automaticamente. A princípio, sugere-se incluir um teste unitário dos módulos nesta etapa; nesse caso, as unidades de código produzidas são testadas individualmente antes de passar a etapa de integração e teste global.

Teste do sistema

Concluída a codificação, começa a fase de teste do sistema. O processo de teste centraliza-se em dois pontos principais: as lógicas internas do software e as funcionalidades externas. Esta fase decide se foram solucionados erros de "comportamento" do software e assegura que as entradas definidas produzam resultados reais que coincidam com os requisitos especificados.

Manutenção

Essa etapa consiste na correcção de erros que não foram previamente detectados, em melhorias funcionais e de preferência e outros tipos de suporte. A etapa de manutenção à parte do ciclo de vida do produto de software e não pertence estritamente ao seu desenvolvimento.

Melhorias e correcções podem ser consideradas como parte do desenvolvimento.

As etapas descritas são as principais, porém existem sub-etapas dentro de cada etapa, as quais diferem muito de um projecto para outro. Também é possível que certos projectos de software exijam a incorporação de uma etapa extra ou a separação de uma etapa em outras etapas.

Com certeza, todas essas variações do modelo Cascata possuem o mesmo conceito básico: a ideia de que uma etapa fornece saída que serão usadas como entradas para a etapa seguinte. Portanto, o processo de desenvolvimento de um produto de software de acordo com o modelo Cascata é simples de conhecer e controlar.

Outras actividades que também são levadas em consideração em cada uma das etapas de desenvolvimento do software: a documentação, a verificação e a administração das etapas serem documentos. A verificação, por sua vez, é necessária para que uma etapa forneça os dados correctos para a etapa seguinte. Já a administração, efectua a gestão e o controle da etapa.

Problemas

O ciclo de vida Cascata é o paradigma mais visto e mais amplamente empregue na engenharia de software, porém sua aplicabilidade, em muitos campos, tem sido questionada.
Entre os problemas que surgem quando se aplica o modelo são:

- Na realidade, os projectos raramente seguem o fluxo sequencial que o modelo propõe. A interacção é sempre necessária e está presente, criando problemas na aplicação do modelo;
- Em princípio, é difícil para o cliente especificar os requisitos explicitamente, o que acarreta a incerteza natural do início de qualquer projecto;
- O cliente deve ser paciente, pois uma versão funcional não estará disponível até o final do desenvolvimento. Qualquer erro ou mal-entendido, se não for detectado até que o software seja revisado, pode ser desastroso.

Apesar desses problemas, o modelo Cascata tem um lugar bem definido e importante nos trabalhos de engenharia de software. Ele fornece um padrão do qual se encaixam métodos para a análise, projecto, codificação e manutenção.

Domínio de aplicações

O modelo Cascata aplica-se bem em situações em que o software a ser desenvolvido é simples, os requisitos são bem conhecidos, a tecnologia

usada é bem acessível e os recursos para o desenvolvimento estão disponíveis.

CONCLUSÃO

Vantagens do modelo

• Torna o processo de desenvolvimento estruturado. Tem uma ordem sequencial de fases. Cada fase cai em cascata na próxima e cada fase deve estar terminada antes do início da seguinte;
• Todas as actividades identificadas nas fases do modelo são fundamentais e estão na ordem certa;
• Esta abordagem é actualmente a norma e provavelmente permanecerá como tal nos próximos tempos.

Desvantagens do modelo

• Não fornece feedback entre as fases e não permite a actualização ou redefinição das fases anteriores;
• Não suporta modificações nos requisitos;
• Não prevê a manutenção;
• Não permite a reutilização;
• É excessivamente sincronizado;
• Se ocorrer um atraso todo o processo é afectado;
• Faz aparecer o software muito tarde.

- O modelo conduz a uma rígida divisão de trabalho (analistas, arquitectos, programadores, controladores de qualidade, programadores de manutenção);
- Só o chefe do projecto tem uma visão global do problema;
- Quando algo corre mal, a culpa é dos outros... Ninguém se sente realmente responsável.