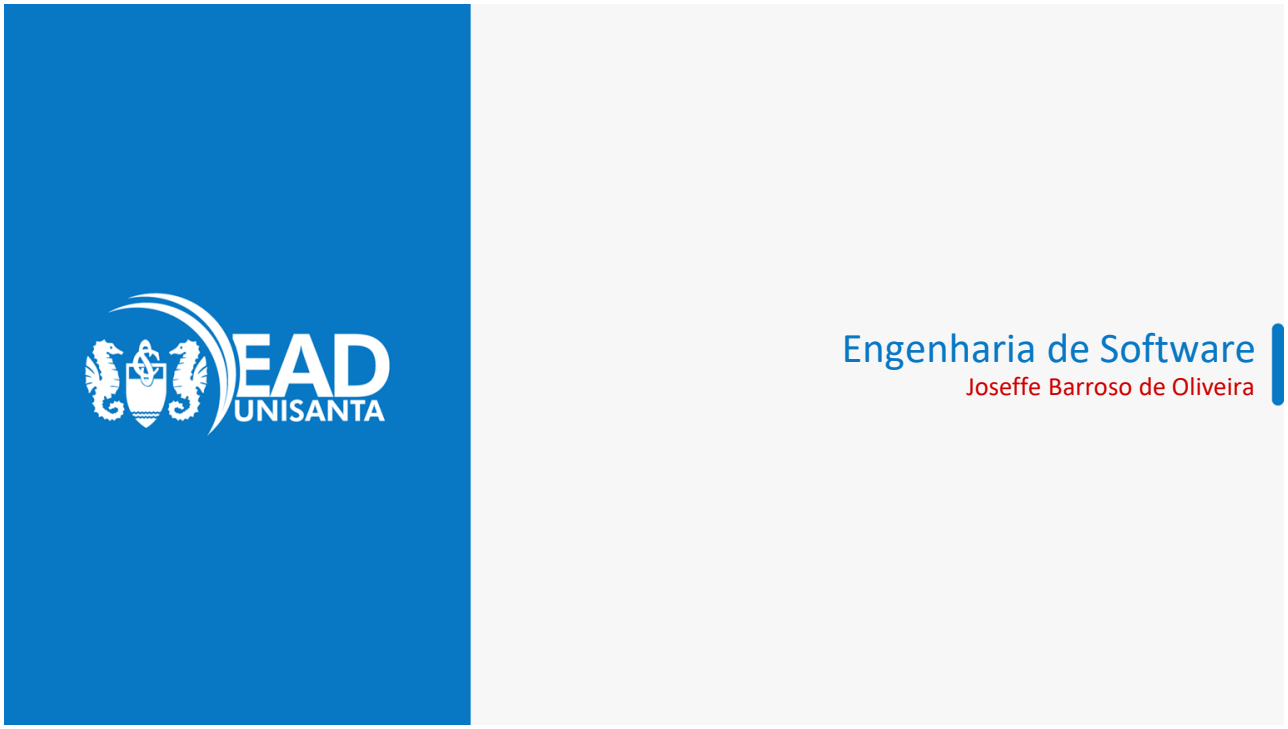

EAD
UNISANTA

1


EAD
UNISANTA
Engenharia de Software
Joseffe Barroso de Oliveira

2

AULA
Ciclos de vida

3

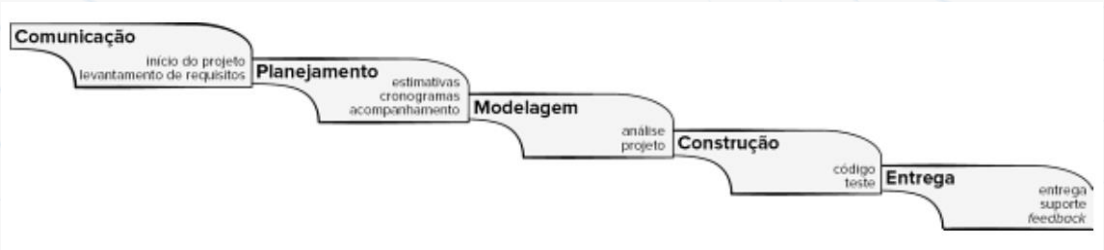
Introdução

Os ciclos de vida definem um conjunto prescrito de elementos de processo e um fluxo de trabalho de processo previsível. Um ciclo de vida concentra-se em estruturar e ordenar o desenvolvimento de software. As atividades e tarefas ocorrem sequencialmente, com diretrizes de progresso definidas. A seguir será apresentado uma visão geral da abordagem dos processos prescritivos, Cada modelo de processo também prescreve um fluxo de processo (também denominado fluxo de trabalho) – ou seja, a forma pela qual os elementos do processo estão relacionados. Todos os modelos de processo de software podem acomodar as atividades metodológicas genéricas, porém cada um deles dá uma ênfase diferente a essas atividades e define um fluxo de processo que invoca cada atividade metodológica (bem como tarefas e ações de engenharia de software) de forma diversa.

4

Modelo cascata

O modelo cascata, algumas vezes chamado de modelo sequencial linear, sugere uma abordagem sequencial e sistemática para o desenvolvimento de software, começando com a especificação dos requisitos do cliente, avançando pelas fases de planejamento, modelagem, construção e entrega, e culminando no suporte contínuo do software concluído.



5

Modelo cascata

O modelo cascata é o paradigma mais antigo da engenharia de software. Entretanto, ao longo das últimas cinco décadas, as críticas a esse modelo de processo aumentaram muito. Entre os problemas às vezes encontrados quando se aplica o modelo cascata, temos:

1. Projetos reais raramente seguem o fluxo sequencial proposto pelo modelo.

2. Com frequência, é difícil para o cliente estabelecer explicitamente todas as necessidades no início da maioria dos projetos.

3. O cliente deve ter paciência. Uma versão operacional do(s) programa(s) não estará disponível antes de estarmos próximos ao final do projeto.

4. Erros graves podem não ser detectados até o programa operacional ser revisto.

6

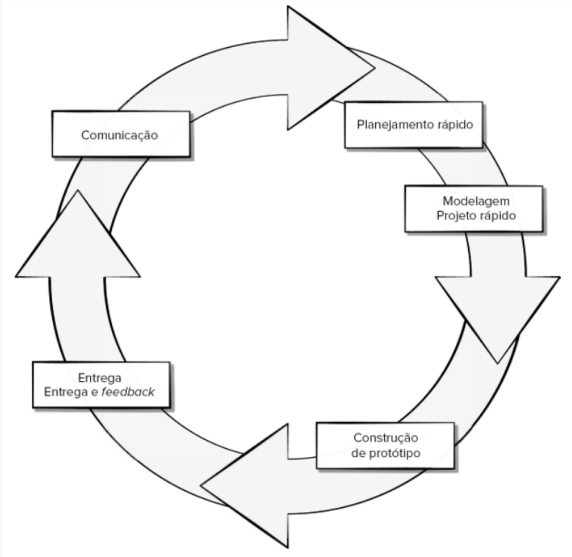
Modelo de prototipação

Frequentemente, o cliente define uma série de objetivos gerais para o software, **mas não identifica, com detalhes**, os requisitos para funções e recursos. Em outros casos, **o desenvolvedor se encontra inseguro quanto à eficiência de um algoritmo**, quanto à adaptabilidade de um sistema operacional ou quanto à forma em que deve ocorrer a interação homem-máquina. Em situações como essas, e em muitas outras, o **paradigma da prototipação pode ser a melhor abordagem**. Embora a prototipação possa ser utilizada como um modelo de processo isolado (stand-alone process), ela é mais comumente utilizada como uma técnica a ser implementada no contexto de qualquer um dos modelos de processo. Independentemente da forma como é aplicado, quando os requisitos estão obscuros, o paradigma da prototipação auxilia os envolvidos a compreender melhor o que está para ser construído.

7

Modelo de prototipação

Na sua forma ideal, o protótipo atua como um mecanismo para identificar os requisitos do software. Caso seja necessário desenvolver um protótipo operacional, pode-se utilizar partes de programas existentes ou aplicar ferramentas que possibilitem gerar rapidamente tais programas operacionais. Tanto os envolvidos quanto os engenheiros de software gostam do paradigma da prototipação. Os usuários podem ter uma ideia prévia do sistema final.



8

Modelo de prototipação

Entretanto, a prototipação pode ser problemática pelas seguintes razões:

1. Os envolvidos enxergam o que parece ser uma versão operacional do software. Eles podem desconhecer que a arquitetura do protótipo (a estrutura do programa) também está evoluindo. Isso significa que os desenvolvedores podem não ter considerado a qualidade global do software, nem sua manutenção em longo prazo.

2. O engenheiro de software pode ficar tentado a fazer concessões na implementação para conseguir que o protótipo entre em operação rapidamente. Se não tomar cuidado, essas escolhas longe de ideais acabam se tornando uma parte fundamental do sistema.

9

Modelo evolucionário/espiral

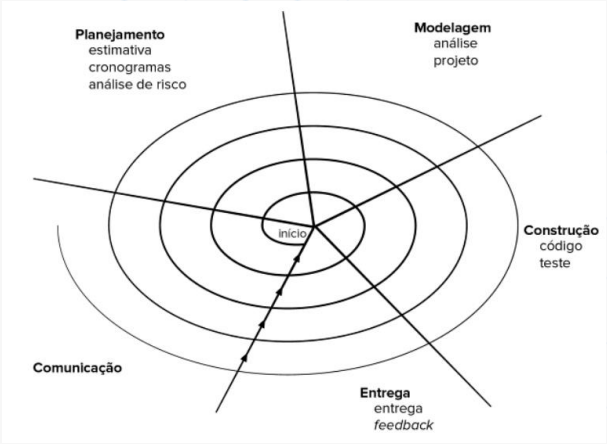
Como todos os sistemas complexos, o software evolui ao longo do tempo. Conforme o desenvolvimento do **projeto avança**, **os requisitos do negócio e do produto frequentemente mudam**, tornando inadequado seguir um planejamento em linha reta de um produto final.

O modelo espiral é um modelo de processo de software evolucionário que une a natureza iterativa da prototipação aos aspectos sistemáticos e controlados do modelo cascata. Tem potencial para o rápido desenvolvimento de versões cada vez mais completas do software.

10

Modelo evolucionário/espiral

O modelo espiral é **dividido em um conjunto de atividades metodológicas** definidas pela equipe de engenharia de software. Assim que esse processo evolucionário começa, a equipe de software realiza **atividades indicadas por um circuito em torno da espiral**, no sentido horário, começando pelo seu centro. **Os riscos são levados** em conta à medida que cada evolução é realizada.



11

Modelo de processo unificado

Sob certos aspectos, o Processo Unificado **é uma tentativa de aproveitar os melhores recursos e características dos modelos tradicionais** de processo de software, mas **caracterizando-os de modo a implementar muitos dos melhores princípios do desenvolvimento ágil de software**. O Processo Unificado reconhece a importância da comunicação com o cliente e de métodos racionalizados para descrever a visão do cliente sobre um sistema (os casos de uso).

Ele enfatiza o importante papel da arquitetura de software e "ajuda o arquiteto a manter o foco nas metas corretas, como compreensibilidade, confiança em mudanças futuras e reutilização". Ele sugere um fluxo de processo iterativo e incremental, proporcionando a sensação evolucionária que é essencial no desenvolvimento de software moderno.

12

Modelo de processo unificado

É provável que, ao mesmo tempo em que as fases de construção, transição e produção estejam sendo conduzidas, já se tenha iniciado o incremento de software seguinte. Isso significa que as cinco fases do PU não ocorrem em sequência, mas de forma concomitante e escalonada.