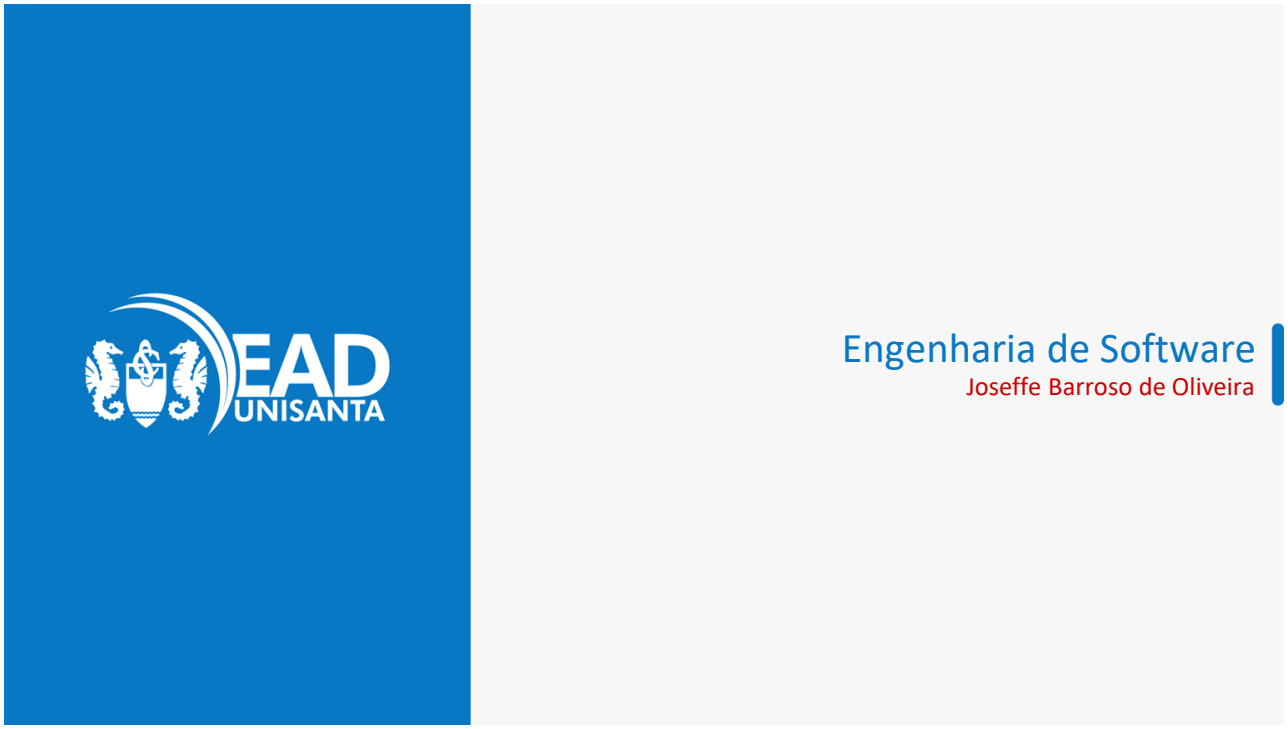

EAD
UNISANTA

1


EAD
UNISANTA
Engenharia de Software
Joseffe Barroso de Oliveira

2

AULA

Processo de desenvolvimento de software e suas melhores práticas

3

Introdução

Cada projeto é diferente e cada equipe é diferente. Nenhuma metodologia de engenharia de software é apropriada para todo artefato de software possível. Neste capítulo abordaremos o uso de um processo adaptável que pode ser ajustado para atender às necessidades de desenvolvedores de software que trabalham em produtos dos mais diversos tipos.

4

Definição dos requisitos

Todo projeto de software começa com a equipe tentando entender o problema a ser resolvido e determinando quais resultados são importantes para os envolvidos. Isso inclui **entender as necessidades de negócios que motivam o projeto e as questões técnicas que o limitam**. O processo é chamado de engenharia de requisitos. As equipes que não dedicam uma quantidade de tempo razoável a essa tarefa logo descobrem que o seu projeto contém retrabalho caro, orçamentos estourados, artefatos de baixa qualidade, entregas atrasadas, clientes insatisfeitos e equipes desmotivadas. A engenharia de requisitos não pode ser ignorada e não pode ser deixada em ciclos intermináveis antes que a construção do artefato possa começar.

5

Projeto de arquitetura preliminar

As decisões de projeto preliminares muitas vezes precisam ocorrer quando os requisitos são definidos. Em algum momento, as decisões sobre arquitetura precisarão ser alocadas a incrementos de produtos. De acordo com Bellomo e seus colegas, um **entendimento inicial sobre as escolhas de arquitetura e requisitos é essencial** para gerenciar o desenvolvimento de artefatos de software grandes e complexos.

6

Estimativa de recursos

Um dos aspectos mais controversos do uso da prototipação ágil ou espiral é **estimar o tempo necessário para completar um projeto** quando este não pode ser definido completamente. É importante entender, antes de começar e de aceitar o projeto, se há ou não uma probabilidade razoável de entregar artefatos de software a tempo e com custos aceitáveis. As primeiras estimativas correm o risco de estarem incorretas porque o escopo do projeto não está bem definido e tende a mudar após o início do desenvolvimento. As estimativas produzidas quando o projeto está quase terminado não servem para orientar o gerenciamento do projeto. **O truque é estimar o tempo de desenvolvimento de software no início, com base no que é conhecido naquele momento**, e revisar suas estimativas regularmente à medida que requisitos são adicionados ou após incrementos de software serem entregues.

7

Construção do primeiro protótipo

Os desenvolvedores podem **usar o primeiro protótipo para provar que o seu projeto de arquitetura inicial é uma abordagem viável** para a entrega da funcionalidade exigida ao mesmo tempo que atende às restrições de desempenho do cliente. Criar um protótipo operacional sugere que a engenharia de requisitos, o projeto de software e a construção procedem todos em paralelo.

As seguintes etapas deve ser seguidas nessa fase:

1. Faça a transição do protótipo de papel para o projeto de software.
2. Crie um protótipo da interface do usuário.
3. Crie um protótipo virtual.
4. Adicione entradas e saídas ao seu protótipo.
5. Desenvolva os seus algoritmos.
6. Teste o seu protótipo.
7. Mantenha a entrega em mente enquanto cria o protótipo.

8

Avaliação do protótipo

O teste é conduzido pelos desenvolvedores enquanto o protótipo é construído e se torna uma parte importante da sua avaliação. **O teste demonstra que os componentes do protótipo são operacionais**, mas é improvável que os casos de teste identifiquem todos os defeitos.

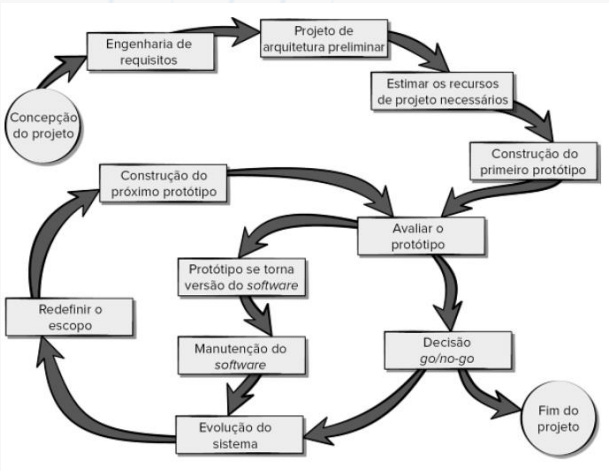
Dicas de melhores práticas para coletar feedback sobre o seu protótipo:

1. Forneça scaffolding quando solicitar feedback sobre o protótipo.
2. Teste o seu protótipo com as pessoas certas.
3. Faça as perguntas certas.
4. Seja neutro quando apresentar alternativas aos usuários.
5. Adapte durante o teste.
6. Permita que os usuários contribuam com ideias.

9

Decisão go/no-go

Após o protótipo ser avaliado, os envolvidos do projeto **decidem se devem continuar ou não o desenvolvimento do artefato de software**. No método que estamos propondo aqui, **cada volta em torno da espiral desenvolve um incremento** significativo do artefato de software final. Você pode trabalhar na história de usuário do projeto ou backlog de recursos para identificar um subconjunto importante do artefato final a ser incluído no primeiro protótipo e repetir esse ciclo para cada protótipo subsequente.



10

Evolução do protótipo

Após o protótipo ter sido desenvolvido e revisado pela equipe de desenvolvimento e por outros envolvidos, **é o momento de considerar o desenvolvimento do próximo protótipo**. O **primeiro passo é coletar todo o feedback** e os dados da avaliação do protótipo atual. Então, os desenvolvedores e envolvidos começam as negociações para planejar a **criação de mais um protótipo**. Após chegarem a um acordo sobre as características do novo protótipo, consideram-se as restrições temporais e orçamentárias conhecidas, além da viabilidade técnica de se implementar o protótipo. Se os riscos de desenvolvimento são considerados aceitáveis, o trabalho continua.

11

Disponibilização do protótipo

Quando um processo de prototipação evolucionário é aplicado, os desenvolvedores podem ter dificuldade em saber quando o artefato está acabado e pode ser disponibilizado para os clientes. Os desenvolvedores não querem lançar software cheio de bugs para os usuários, que então decidiriam que o software é de má qualidade. Um protótipo considerado candidato a lançamento **deve ser submetido a testes de aceitação do usuário além dos testes funcionais e não funcionais (de desempenho)** que seriam conduzidos durante a construção do protótipo.

12

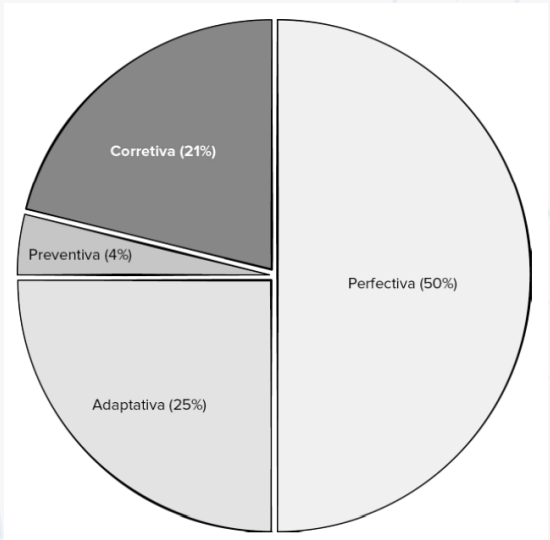
Manutenção do software

A manutenção é definida como as **atividades necessárias para manter o software operacional após ele ser aceito e entregue (lançado)** no ambiente do usuário.

- **Manutenção corretiva:** Consertar problemas descobertos;
- **Manutenção adaptativa:** Modificação reativa do software após a entrega para mantê-lo utilizável;
- **Manutenção perfectiva:** Modificação pró ativa do software após a entrega para adicionar novos recursos;
- **Manutenção preventiva:** Modificação pró ativa do software após a entrega para detectar e corrigir falhas;

13

Manutenção do software



14