XP : Uma Visão Geral

Profa. Karin Becker

Instituto de Informática - UFRGS

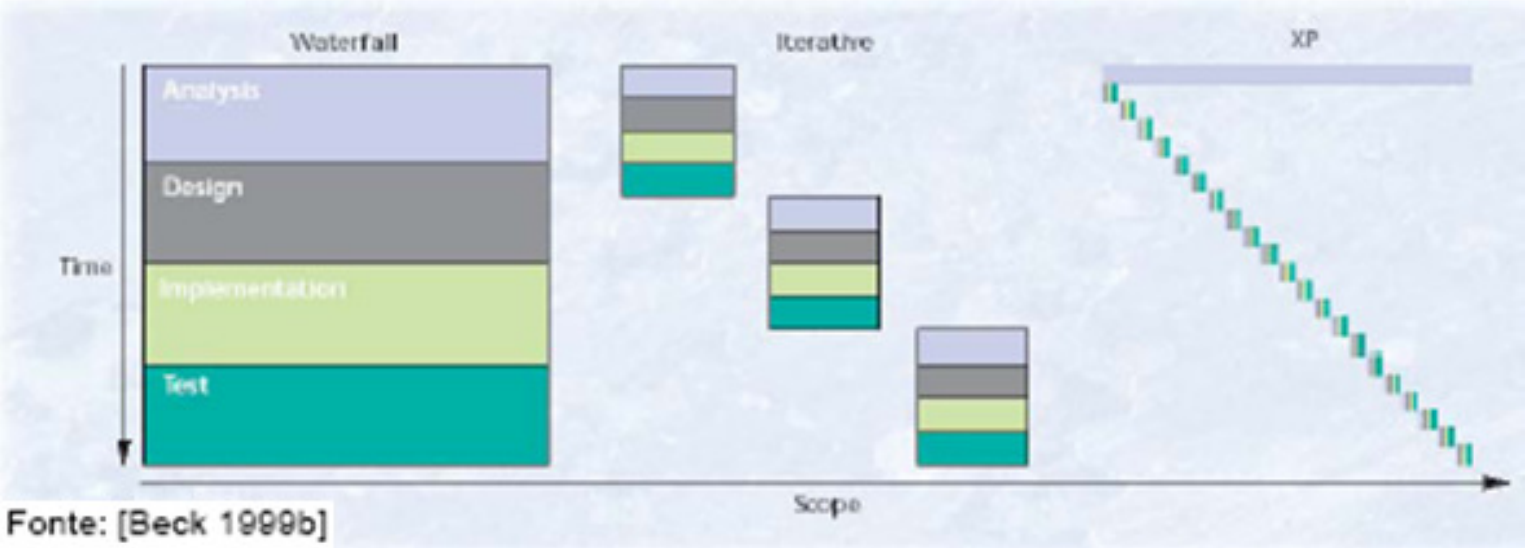
Extreme Programming não é ...

O método

O time

Extreme Programming (XP)

- desenvolvimento e na entrega de incrementos de funcionalidade muito pequenos.

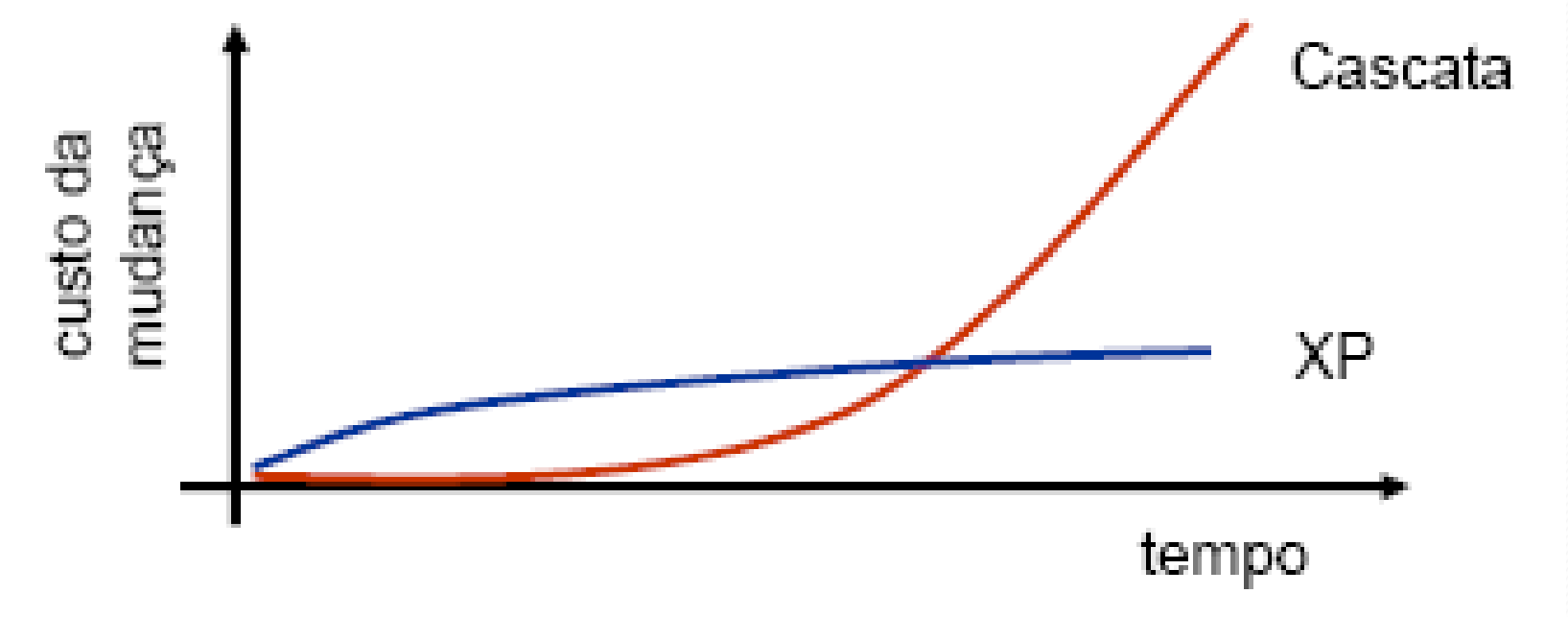


Fonte: [Beck 1999b]

- Baseia-se no aprimoramento constante do código, no envolvimento do usuário na equipe e no desenvolvimento e programação em pares
- “light-weight”

XP : Abrace a mudança!

- Desafiar a curva do custo da mudança



XP e Mudanças

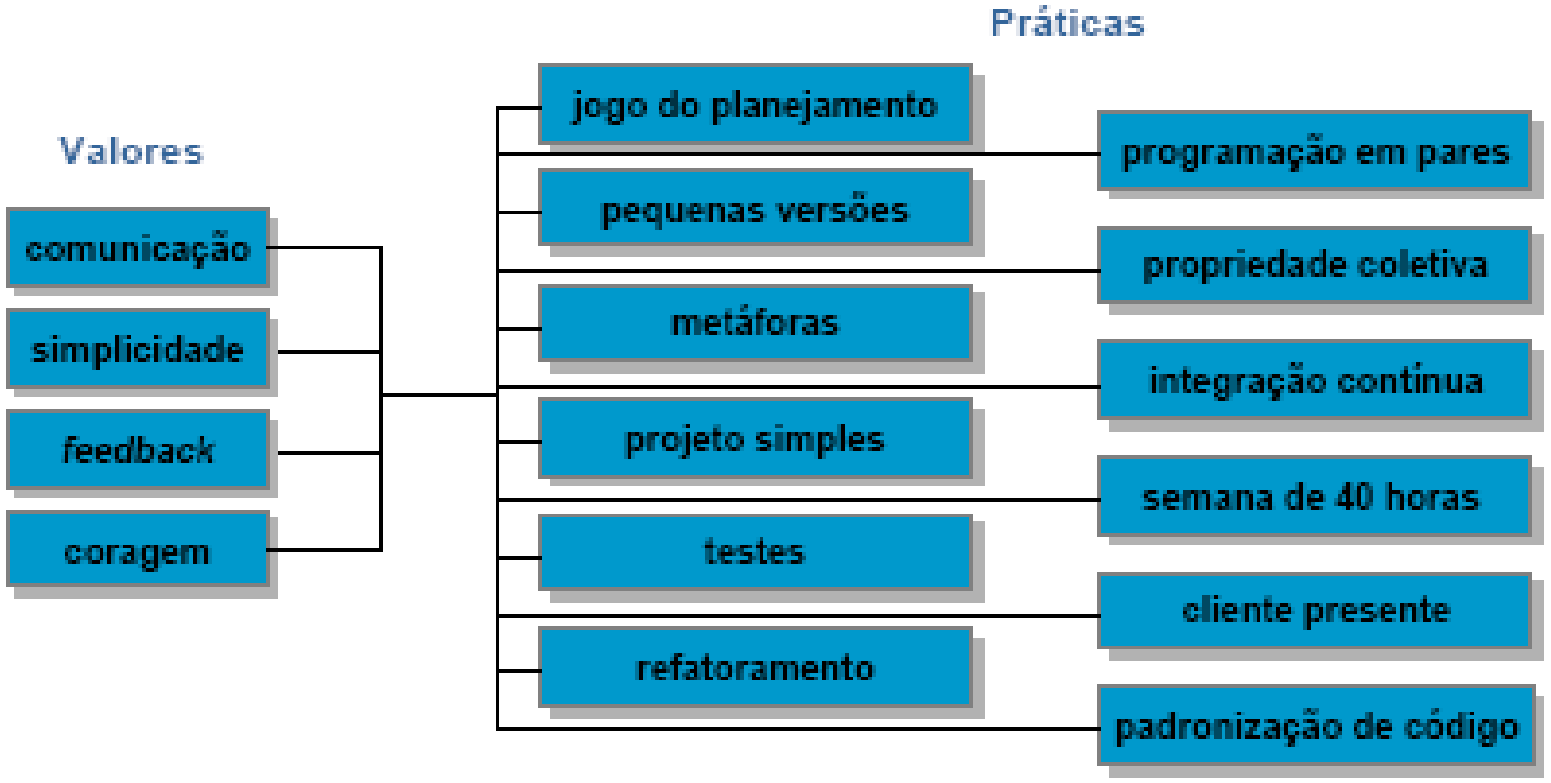
- Tradicionalmente, a engenharia de software recomenda projetar para mudança
  - Vale despender tempo e esforço antecipando mudanças quando isso reduz custos posteriores no ciclo de vida
- XP assume que este esforço não vale a pena quando as mudanças não podem ser confiavelmente previstas
- XP preconiza
  - ciclos curtos : previsibilidade e redução de incertezas/riscos
  - Simplicidade e melhorias constantes de código (*refactoring*) : facilitar a mudança
  - Testes automatizados e integração contínua : aumentar a confiança

O Mantra do Desenvolvedor XP

- **Escute**, para que saiba qual é o problema a resolver
- **Planeje**, para que você sempre faça a coisa mais importante ainda a fazer
- **Codifique**, senão o software não sai
- **Teste**, senão você não sabe se está funcionando
- **Refatore**, senão o código vai ficar tão ruim que será impossível dar manutenção

XP

- Equipes pequenas e médias (2 a 10 pessoas)
- Iterações curtas (2 a 4 semanas)
- Reúne práticas de implementação em um conjunto coerente, acrescentando idéias de processo.



XP: algumas práticas

Tabela 17.2 Práticas da *extreme programming*

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são registrados em cartões de histórias e as histórias a serem incluídas em um release são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores dividem essas histórias em 'tarefas'. Veja as figuras 17.4 e 17.5.
Pequenos releases	O conjunto mínimo útil de funcionalidade que agrega valor ao negócio é desenvolvido primeiro. Releases do sistema são freqüentes e adicionam funcionalidade incrementalmente ao primeiro release.
Projeto simples	É realizado um projeto suficiente para atender aos requisitos atuais e nada mais.
Desenvolvimento *test-first*	Um framework automatizado de teste unitário é usado para escrever os testes para uma nova parte da funcionalidade antes que esta seja implementada.
Refactoring	Espera-se que todos os desenvolvedores recriem o código continuamente tão logo os aprimoramentos do código forem encontrados. Isso torna o código simples e fácil de manter.
Programação em pares	Os desenvolvedores trabalham em pares, um verificando o trabalho do outro e fornecendo apoio para realizar sempre um bom trabalho.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de tal maneira que não se formem ilhas de conhecimento, com todos os desenvolvedores de posse de todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Tão logo o trabalho em uma tarefa seja concluído, este é integrado ao sistema como um todo. Depois de qualquer integração, todos os testes unitários do sistema devem ser realizados.
Ritmo sustentável	Grandes quantidades de horas extras não são consideradas aceitáveis, pois, no médio prazo, há uma redução na qualidade do código e na produtividade.
Cliente *on-site*	Um representante do usuário final do sistema (o cliente) deve estar disponível em tempo integral para apoiar a equipe de XP. No processo da *extreme programming*, o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema à equipe para implementação.

Sommerville

XP : Valores

- Comunicação
- Simplicidade
- Feedback
- Coragem

- *Coach*
  - pessoa responsável por garantir a aderência a estes valores nas práticas

Documento = Código

- Codificação é a atividade central do projeto
- Testes (que também são código) servem de especificação
- Comunicação **oral** entre desenvolvedores, baseada no código (testes e funcionalidade) que descreve o sistema
- Isto não quer dizer que equipe XP não se valham de modelos
  - Documento não é incremento
  - Modelos servem para entender e se comunicar

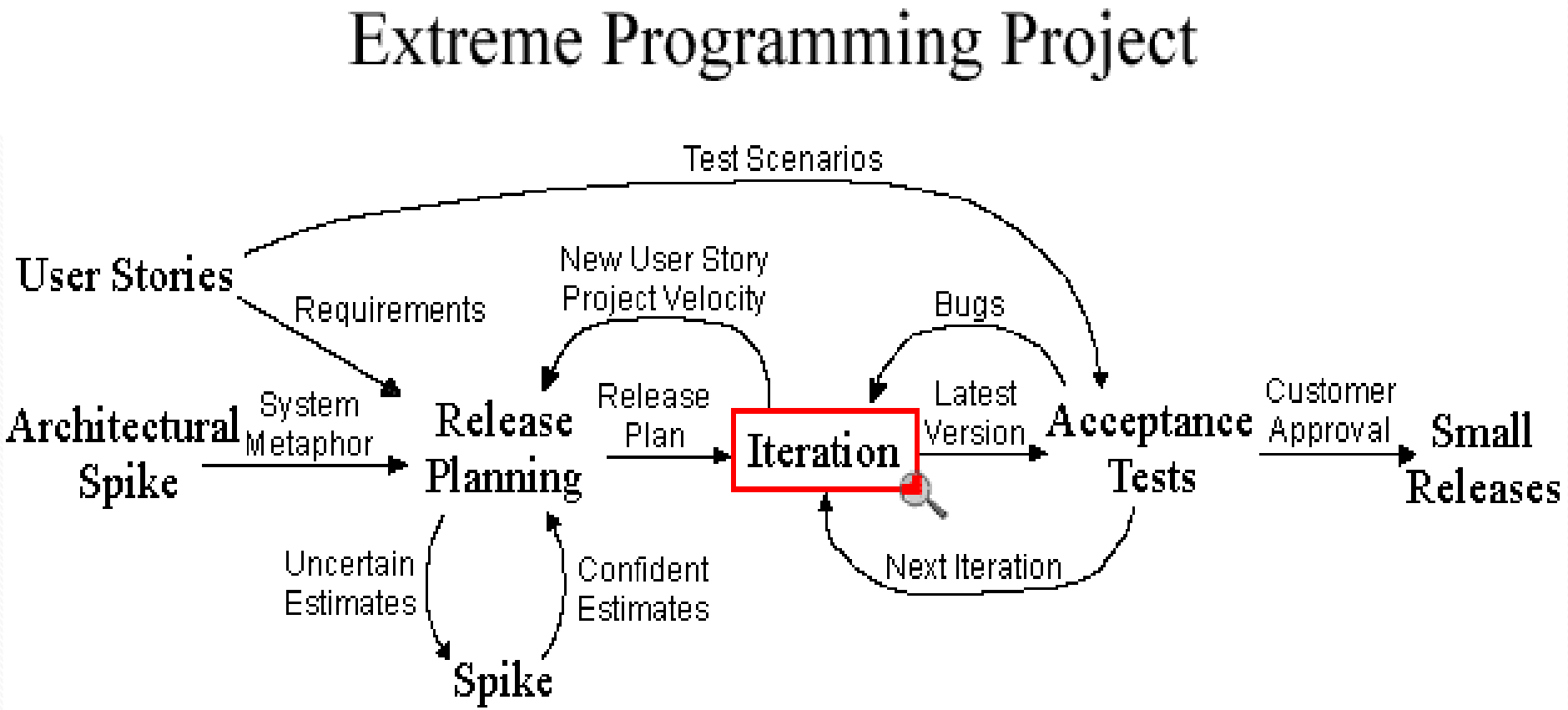
Unidades de Trabalho

- Releases
  - pequenas e freqüentes (a cada 2-3 meses)
- Iterações
  - Uma iteração alcança algum objetivo (tipicamente a adição de nova funcionalidade)
  - Nada é feito que não seja imediatamente útil e necessário para não afetar os prazos de desenvolvimento
  - divididas em tarefas
- Tarefas
  - Tarefas são a menor quantidade de trabalho que pode ser feita até que todos os testes voltem a funcionar
  - Tarefas não levam mais que um dia
  - Uma vez concluídas, tarefas são integradas imediatamente

XP: Ciclo de desenvolvimento

- O desenvolvimento incremental é apoiado em releases de sistema pequenos e freqüentes
- O envolvimento do cliente significa o seu engajamento em tempo integral com a equipe
- Pessoas, e não processos na programação em pares, propriedade coletiva e um processo que evita longas horas de trabalho
- As mudanças são apoiadas em releases de sistema regulares
- Manutenção da simplicidade por de meio de *refactoring* constante do código

XP : Ciclo de Desenvolvimento



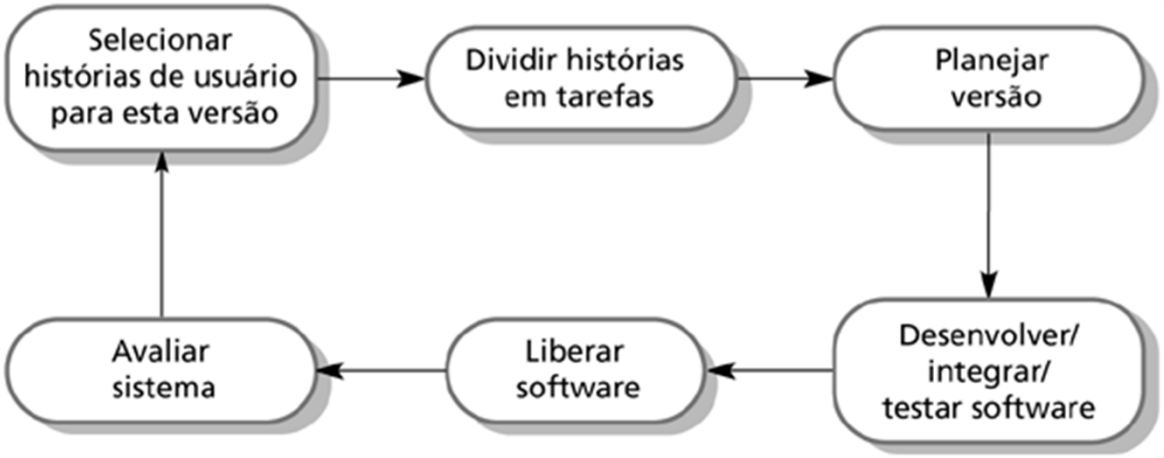
Fonte: www.extremeprogramming.org

Ciclo de Release

- Um release é um conjunto de histórias que são disponibilizados simultaneamente
  - As histórias mais importantes e/ou mais difíceis têm prioridade
  - Cliente define a prioridade

Figura 17.3

Ciclo de um release em *extreme programming*



Cenário de Requisitos

- os requisitos de usuários são expressos como cenários ou histórias de usuários
  - História, cenário, épico
- Sem formato definido
  - “Como usuário, desejo baixar e imprimir um artigo”
  - **Figura 17.4**

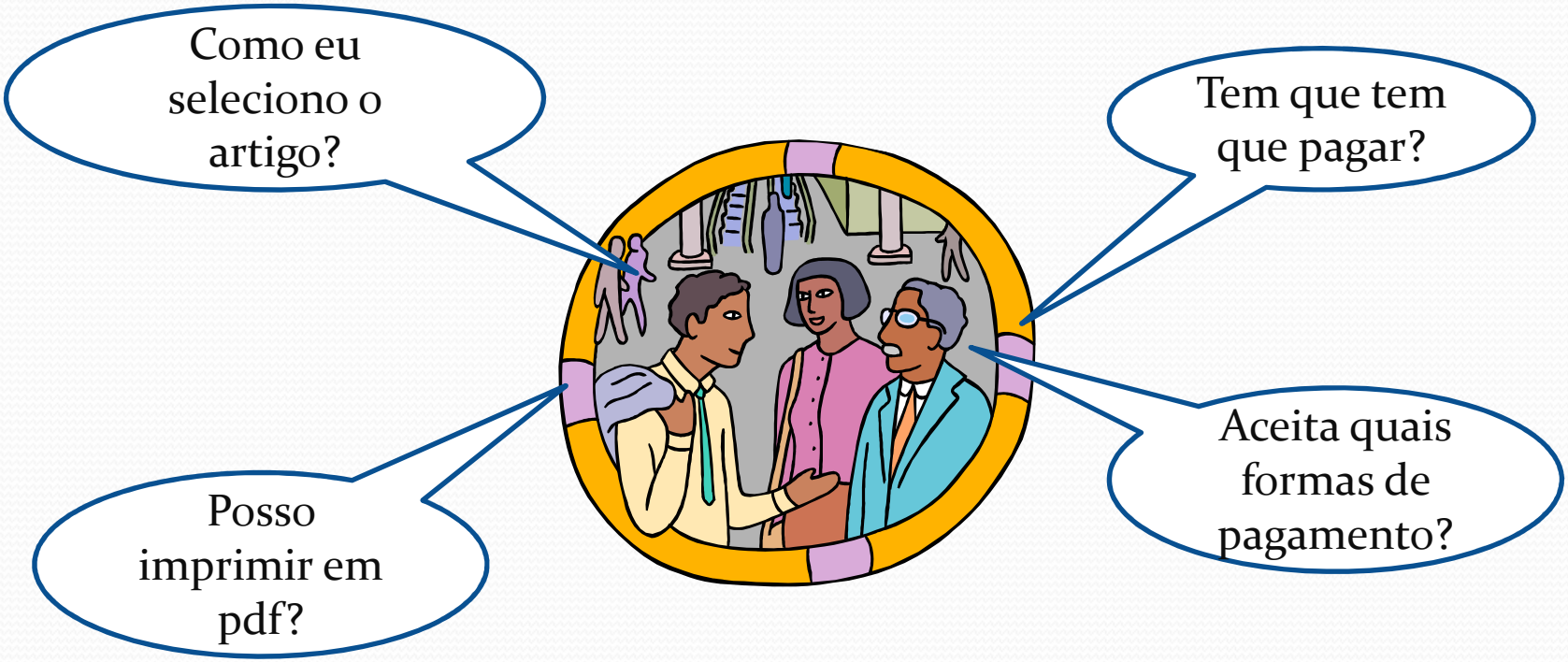
    Cartão de histórias para baixar documentos.

Baixando e imprimindo um artigo
Primeiro, você seleciona o artigo que deseja em uma lista. Depois você precisa informar ao sistema como quer pagar pelo artigo — isso pode ser feito por meio de uma assinatura, de uma conta empresarial ou por cartão de crédito. Após, você obtém do sistema um formulário de direitos autorais para preenchimento. Após enviar o formulário, o artigo desejado é baixado para seu computador. Em seguida, você escolhe uma impressora para imprimir uma cópia do artigo. Você informa ao sistema que a impressão foi bem-sucedida. Se o artigo for somente para impressão, você não poderá manter uma versão em PDF, de modo que o artigo será excluído automaticamente de seu computador.

Sommerville

- CCC – Card, Conversation, Confirmation

Requisitos: Conversation



Requisitos : Confirmation

Figura 17.6

Descrição de caso de teste para validação de cartão de crédito.

Baixando e imprimindo um artigo
Entrada: Uma string que representa o número de cartão de crédito e dois inteiros que representam o mês e o ano de expiração do cartão. **Testes:** Verificar se todos os bytes na string são dígitos. Verificar se o mês varia entre 1 e 12 e o ano é posterior ou igual ao ano atual. Usando os primeiros 4 dígitos do número de cartão de crédito, verificar se o emissor do cartão é válido consultando a tabela de emissores de cartões. Verificar a validade do cartão de crédito enviando o número do cartão e a data de expiração para o emissor de cartões. **Saída:** OK ou mensagem de erro indicando que o cartão é inválido.

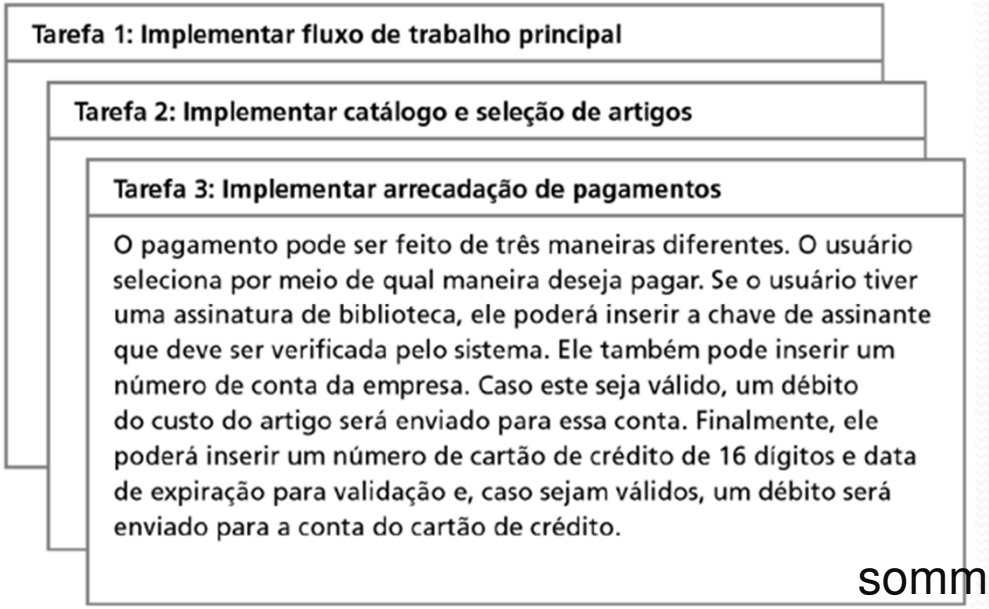
sommerville

Cenário de Requisitos

- a equipe de desenvolvimento detalha tarefas de implementação

Figura 17.5

Cartões de tarefa para baixar documentos.



sommerville

- equipe usa as tarefas para estimativas
  - Todos são responsáveis pelas estimativas

Prática: Testar

- Envolvimento do cliente no desenvolvimento e validação de testes
- Testes se tornam as especificações da programação
  - Escrevendo testes **antes** da funcionalidade, esclarecem-se as dúvidas sobre o que o software deve fazer
- Testar um pouco, codificar um pouco
  - "Test-first programming"
  - se você não automatizou o teste, sua programação não está concluída
- Ferramentas de teste automatizadas são usadas para executar todos os testes de componentes cada vez que uma nova release é construída
- Testes impõem confiança ao sistema
- Testes dão coragem para alterar o sistema

Prática : Projeto Simples

- Projetos flexíveis são uma defesa contra mudanças imprevistas no software
- Porém, projetos flexíveis também têm custos
  - Tempo para desenvolvimento e manutenção
  - O código fica mais complexo
  - Muita vezes a flexibilidade não é utilizada
- Como mudança é barata em XP, mantém-se o projeto o mais simples possível, modificando-o quando for necessário suportar mais funcionalidade

Prática: Projeto Simples

- O melhor projeto é aquele que:
  - Passa em todos os testes
  - Não contém duplicação de funcionalidade
  - Salienta as decisões de projeto importantes
  - Tem o menor número possível de classes e métodos

Prática: Refatorar

- Refatorar é melhorar o código sem alterar sua funcionalidade
- Antes de uma mudança, você refatora o código para que facilitar a realização de mudanças
- Refatoração contínua possibilita manter um bom projeto, apesar das mudanças freqüentes
- Projeto é uma atividade diária, de responsabilidade de todos
- Aumento contínuo de qualidade do código
- Teste e Pair Programing dão a coragem de mudar

Para saber mais ...

- A Gentle Introduction to Extreme Programming.
  - Um tutorial bem acessível sobre XP.
  - www.extremeprogramming.org/
- Alguns blogs
  - Jeffries: xprogramming.com/index.php
  - Fowler: http://martinfowler.com/
  - Wells : www.extremeprogramming.org/
- Quer conhecer Kent Beck?
  - http://www.youtube.com/watch?v=4Awqwtyll8I

Referências

- Beck, K. *Extreme Programming Explained – Embrace Change*. Addison-Wesley. (2nd Ed.)*
- Beck, K. *Test-Driven Design*. Addison-Wesley.
- Cohn, M. User Stories Applied: For Agile Software Development. Pearson.
- Anderson, A., Beattie, Ralph, Beck, Kent et al. Chrysler goes to "extremes", *Distributed Computing* (October 1998), 24-28.
- Williams, L. and Kessler, R., "All I Really Need to Know about," Communications of the ACM (May 2000)
- Abrahamsson, P. et al. *Agile software development methods: reviews and analysis*. Espoo: VTT Publications, 2002. Disponível em:http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf.