



Gerenciamento de Projetos Iterativos e Metodologias Ágeis

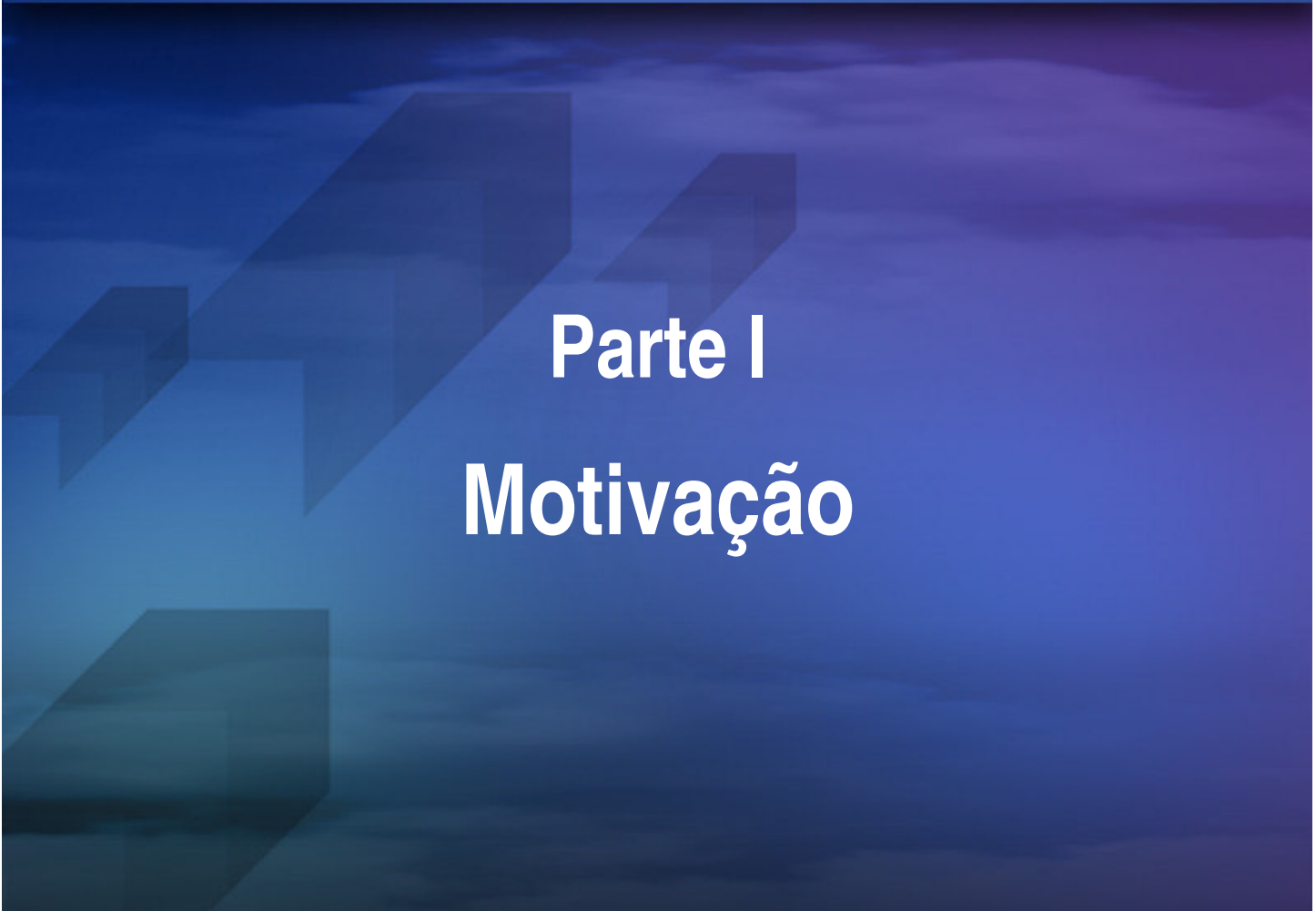
Marcio Tierno
Outubro 2006

O que NÃO esperar

- **Exposição sobre Gerenciamento de Projetos**
 - PMBOK, etc
- **Técnicas detalhadas sobre Gerenciamento Ágil**
- **Detalhes sobre metodologias ágeis**

O que esperar

- **Discussão (muita!)**
- **Motivação**
- **Introdução a processos iterativos**
- **Princípios gerais de agilidade**
- **Visão geral de gerenciamento de projetos nessas condições**

The background of the slide is a deep blue gradient with several semi-transparent, 3D rectangular blocks of varying sizes and orientations, creating a sense of depth and movement. The text is centered in the middle of the slide.

Parte I

Motivação

Por que software é diferente?

The background of the slide is a dark blue gradient. On the left side, there are several semi-transparent, 3D rectangular blocks of varying sizes and orientations, creating a sense of depth and architectural structure. The overall aesthetic is clean and modern.

Por que software é diferente?

- **~75% dos projetos de software falham**
 - Standish Group, 2004, Gartner Group, 2003
- **Dos projetos que falham, 71% falham devido aos requisitos**
 - Standish Group, 2004
- **42-64% dos defeitos se originam nos requisitos e 25-40% dos gastos totais são devidos a retrabalho causado por defeitos nos requisitos**
 - SEI Research Report, 2004

Sucesso do Projeto x Sucesso do Cliente

- **Definição de sucesso (projeto)**
 - Entregue dentro do prazo e do orçamento
- **Definição de sucesso (cliente)**
 - Negócio suportado a contento

Estas definições podem ser conflitantes ?

Caso hipotético

- **Em 1998, numa grande operadora de tel. Celular**
 - Não havia pré-pago (lançado depois)
 - Ativação de aparelho era feita por telefone
 - Demorada, sujeita a erros, irritante para o cliente
 - Projeto para ativação remota via web
 - 4 meses, US\$ 500K

Caso hipotético (cont.)

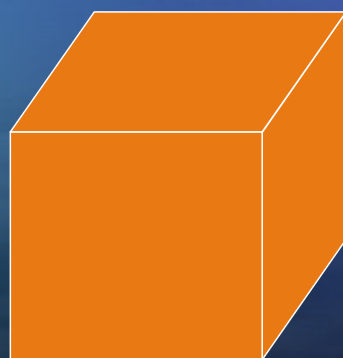
- **3,5 meses depois:**
 - Pré-pago é lançado; é preciso incorporar
 - Área comercial “se lembra” de que clientes geralmente querem escolher o número do seu celular
 - Área operacional “lembra” a equipe de TI que revendedores operam com PCs de 2a com internet em linha discada a 19200
 - Área financeira “descobre” que clientes com “nome sujo” deixam de pagar suas contas – consulta à Serasa deve ser incorporada

Caso hipotético (cont.)

- **Sucesso do projeto: entregar no prazo e no orçamento**
- **Sucesso do cliente: atender bem aos consumidores da empresa**
- **O que fazer?**

O Sucesso “Total”

- Dentro do prazo
- Dentro do orçamento
- Atende às reais necessidades do cliente



= SUCESSO!

Por que software é diferente?

- **Intangível**
- **Difícil saber o que pode e o que não pode ser feito**
- **Difícil saber o que se quer de fato**

- **(complete com sua experiência)**

Natureza dos requisitos de software

Ah, se nossos pacientes
soubessem que doença têm,
não precisaríamos de todos
estes testes!

É...e não erraríamos
tanto, também!



Natureza dos requisitos de software

- **Difíceis de levantar**
 - Usuário não sabe o que quer
 - Usuário não consegue expressar o que quer
 - Analista não consegue captar
- **Difíceis de comunicar**
- **Difíceis de validar, medir, quantificar, etc.**

Natureza dos requisitos de software

- **MUDAM!!!**
- “O Problema da Pedra”
 - Ed Yourdon
- “Síndrome do ‘Sim, mas...’”
 - Dean Lefingwell

Exercício

- **Você acabou de comprar um apartamento novo e deseja comprar uma cozinha planejada.**
 - **Especifique exatamente como você quer que sua cozinha planejada seja em um documento textual**
 - **Defina prazo e custo para entrega**
 - **Este documento será enviado à fábrica de cozinhas para confecção**
 - **A cozinha será instalada**
 - **Somente após concluídos todos os trabalhos você poderá ver a cozinha instalada**

Exercício

- **Quais as chances do resultado final ser o que você queria?**
- **Existe uma maneira alternativa, com maiores chances de satisfazê-lo(a) ?**

Situação atual

- Quais problemas você costuma enfrentar em projetos de software?

(relate aqui os seus)

Problemas com o modelo Waterfall

Requisitos



Análise & Design



Codificação

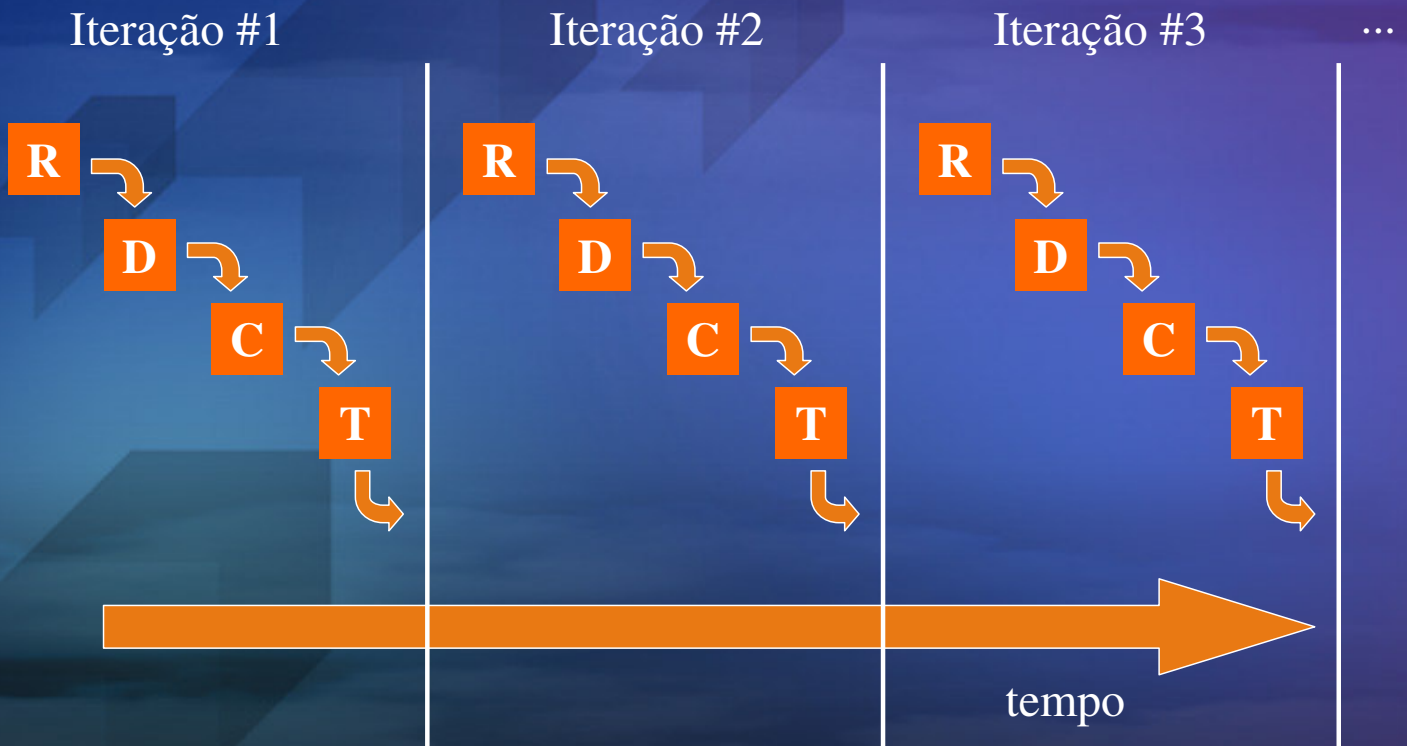


Testes



- Acompanhamento
- Riscos
- Mudança
- Testes
- Qualidade
- ...

O modelo iterativo controlado



Iteração

- **Conceito “Time Box”**
 - Prazo Fixo (não muito maior que 15 dias)
 - Custo Fixo
 - Lista de Funcionalidades que podem variar
- **Perca funcionalidades, não datas!**
- **Funcionalidades são priorizadas pelo cliente**
- **Cliente precisa de algo não previsto?**
 - Sem problemas, inclua. Mas exclua algo!

Processo Iterativo

- **Exemplo**

Mudança solicitada pelo usuário:

Alteração da forma de cálculo do Preço, incluindo variáveis de localização geográfica

Esforço: 90 horas

Lista priorizada de funcionalidades para próxima iteração

Prioridade	Funcionalidade	Esforço (horas)
1	Simular Preço em condições ideais	220
2	Simular Preço sem alguns dados	180
3	Autorizar Orçamento	80
4	Alteração da forma de cálculo do Preço, incluindo variáveis de localização geográfica	90

Sempre no prazo e no orçamento!

- Time box
- Cliente entende que funcionalidades no final da lista podem ficar de fora e/ou eles (os clientes) podem adicionar funcionalidades
- Flexibilidade está nas funcionalidades, não no prazo ou no custo
- **Perca funcionalidades, nunca datas!**

Pareto para software

- **80% das necessidades dos clientes são satisfeitas com 20% das funcionalidades de um sistema**

Exercício

- **Estime (prazo & custo), planeje e execute o seguinte projeto:**
 - **Levar uma peça de 750Kg de São Paulo a BH de carro**
 - **Prazo:** _____
 - **Custo:** _____

Exercício

- **Com menos de 1h de viagem, você recebe um telefonema:**
 - Há uma outra peça que precisa ser retirada no Rio e entregue em São Paulo o mais rápido possível
 - Esta peça tem prioridade absoluta
- **É possível executar esta “funcionalidade” no mesmo prazo e com o mesmo custo que a “funcionalidade” original?**

Funcionalidades podem variar

- No mesmo prazo
- No mesmo custo
- **Novas funcionalidades não são ADICIONADAS ao projeto, mas colocadas NO LUGAR de outras já existentes e que serão EXCLUÍDAS**

The background of the slide is a dark blue gradient with several semi-transparent, 3D rectangular blocks of varying sizes and orientations, creating a sense of depth and movement. The text is centered in the middle of the slide.

Parte II

Agilidade

O que NÃO é AGILIDADE?

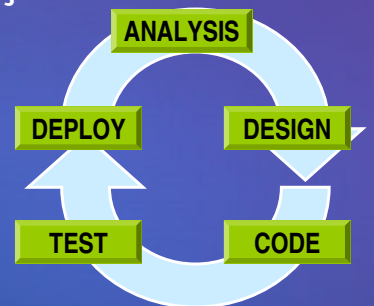


O que é AGILIDADE ?

- **Como software é feito?**
 - Pessoas
 - Processos e
 - Ferramentas
- **Nessa ordem,**
 - Pessoas boas vão compensar processos pobres ou não existentes
 - Pessoas boas vão criar ferramentas ou usar ferramentas de maneira efetiva
 - Processos e ferramentas não irão compensar inadequação do staff de desenvolvimento
 - Massa cinzenta é pré requisito

O que é AGILIDADE?

- Um estado mental, **não** um conjunto de passos, técnicas, etc
- “Encolher” o processo waterfall em pequenas iterações
- Garantir aplicações funcionais e progresso real
- Tornar simples abraçar a mudança
- Melhorar a comunicação entre negócios e TI
- Entregar valor para o negócio mais rápido e continuamente
- Software rodando – necessário mas **não** suficiente
- Qualidade e outras “-idades” desde o início



Agile Manifesto

- Indivíduos e interações *sobre...*
- Software rodando *sobre...*
- Colaboração do cliente *sobre...*
- Responder à mudança *sobre...*
- Processos e ferramentas
- Documentação extensiva
- Negociação contratual
- Seguir um plano

Conquanto damos valor às coisas da direita,
damos muito mais valor às da esquerda

<http://agilemanifesto.org>

Iteratividade como técnica de projetos


- **Interação na iteração: conhecendo-nos melhor**
 - Os usuários nunca têm uma real compreensão de seu problema, muito menos da amplitude da solução que TI pode prover-lhes
 - Ao interagir com o software entregue, os usuários passam a compreender melhor seus problemas e as possibilidades de solução
 - Ao interagir com usuários, analistas entendem melhor suas necessidades
 - Ao interagir com software, a confiança dos usuários aumenta – “Nosso projeto”

Iteratividade como técnica de projetos

- **Muitas vezes, o projeto entregue é totalmente diferente do projeto original**
- **Muitas funcionalidades não previstas são adicionadas**
- **Muitas funcionalidades originais são “dropadas”**
- **A satisfação é alta**

Para que somos pagos?

- Para ter, o mais rápida e eficientemente possível, software rodando que efetivamente responda às necessidades do negócio
- Não somos pagos para:
 - Escrever código
 - Produzir modelos
 - Produzir documentos
 - Produzir planos



Parte III

Gerenciamento Ágil de Projetos

Ger. de Projetos é Ortogonal a Agilidade?

- Gerenciamento de projetos “clássico”
 - Adora tarefas detalhadas, durações, serial/paralelo, recursos
 - Aprecia muito planejamento de longo prazo com granularidade fina
 - (a propósito, o que você estará fazendo em 18 de dezembro deste ano?)
 - Acompanha (de forma acurada) a execução das tarefas (de forma tediosa e dolorosa)
- Ágil
 - Adora planejamento de curto prazo no detalhe suficiente
 - Procura prover feedback das metas verdadeiras
 - Gosta de fazer as coisas mais simples que funcionam
- Solução?
 - Projetos de software precisam de ambos
 - Acompanhamento das funcionalidades priorizadas – **90% NÃO PERMITIDO!!!**
 - Planejamento e acompanhamento tradicionais de
 - Grandes marcos (releases, iterações)
 - Atividades que não relacionadas à produção efetiva de software
 - Dependências intra-projetos

Gerencie as coisas certas!

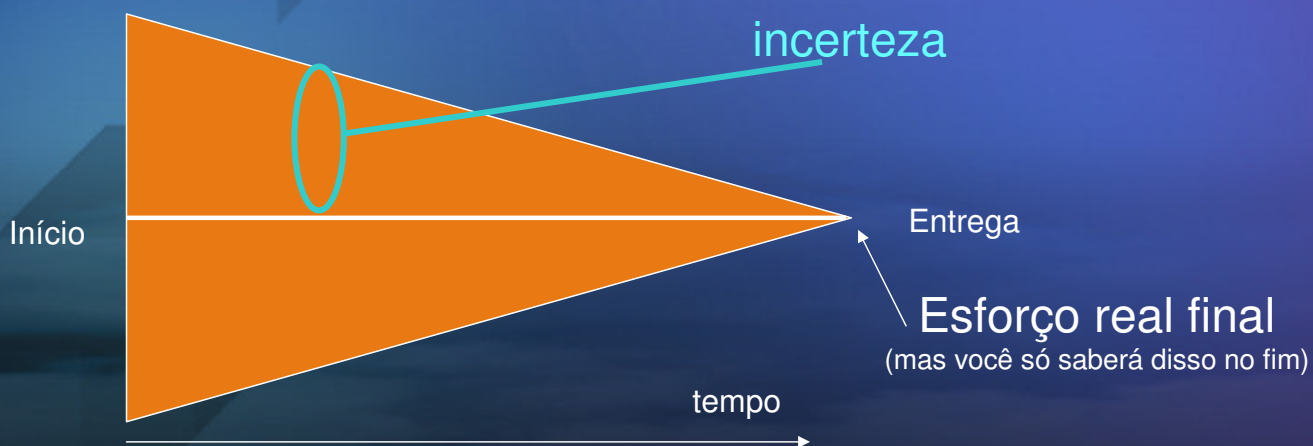
- **Software é “people intensive”**
 - Muita atividade intelectual
 - Não deve ser tratado como linha de montagem
- **Cronografe e meça progresso real**
 - Conclusão de funcionalidades
 - Tarefas necessárias para completá-las
 - Não confunda atividade com progresso
- **Dê mais poder às equipes**
 - Não micro-gerencie, remova barreiras
 - Peça por resultados, ouça a estimativa
 - Mantenha o pessoal engajado e falando
- **Isso tudo é sobre o cliente obter as funcionalidades que quer!**

Estimativa

- **Estimativas são pouco acuradas no início do projeto**
- **A variação é maior devido à mudança de requisitos do que qualquer outro fator**
- **Requisitos vão mudar**

Estimativa

- Há muita incerteza no início, que diminui conforme o final do projeto se aproxima
- É impossível eliminar essa incerteza
- Seja “honesto” com o cliente em relação a isso
 - Compartilhe o risco – reduz custo



Estimativa Ágil

- **“Calcule” apenas o necessário**
- **Heurística funciona tão bem quanto algoritmo**
- **Confie na equipe**
- **Seja transparente com o cliente**
- **Objetivo é definir um “time frame” razoável para o projeto**

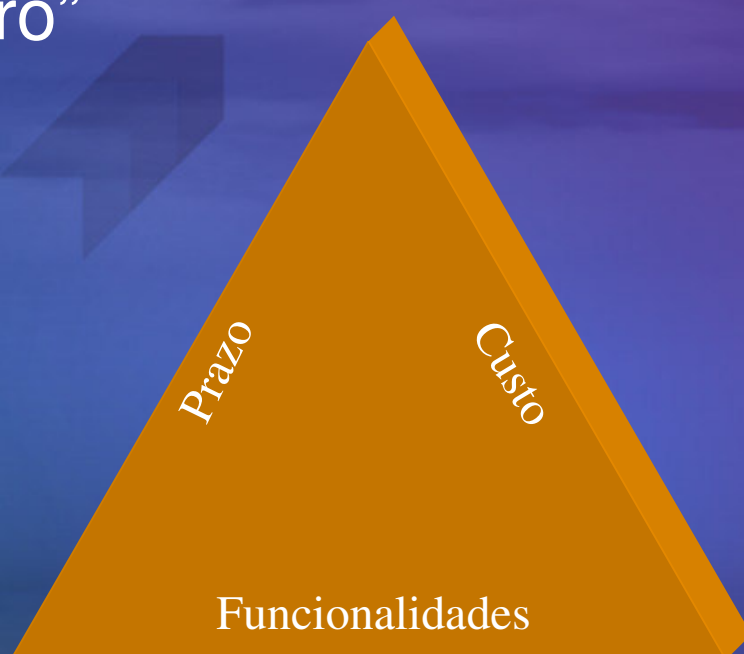
- **Modelos prescritivos demandam mais informação do que se tem à mão**
- **Mesmo que sejam acurados, as mudanças nos requisitos invalidarão as estimativas**

Planejamento

- **Planeje com foco em objetivos concretos, mensuráveis, i.e., funcionalidades**
- **Planejamento tradicional para grandes marcos (releases, iterações)**
- **Não faça micro-planejamento, granularidade mais grosseira**
- **Planejamento em níveis**

O “triângulo de ferro”

- **Cliente**
 - Pode especificar 2, mas não 3
- **Equipe de projeto dita o esforço de uma funcionalidade**
- **Gerenciamento não pode de uma vez só:**
 - Definir funcionalidades
 - Fixar os recursos
 - Estabelecer data de término
- **Planejamento é esforço coletivo**
- **A meta é ser o mais “honesto” possível com o plano**



Planejamento em níveis

- **Projeto**
 - Grandes marcos
 - Fases de maturidade (check points para go/no go) ou Releases
- **Fase / Release**
 - Iterações
- **Iteração**
 - Tarefas
 - Recursos

Plano de Release

- **Discuta / Determine estimativas de projeto**
- **Em alto nível, defina quais funcionalidades são “obrigatórias” e de “alta prioridade”**
- **(Re)Priorize as funcionalidades**
- **Considere protótipos arquiteturais para redução de risco**
- **Considere questões como integração com outros sistemas, entrega, rigor para QA, etc**
- **Meta: obter estimativa de prazo, custo e escopo**

Plano de Release – artefatos

- **Roadmap geral com o release 1 em detalhe suficiente para guiar as iterações iniciais**
 - **Plano de release**
 - Lista de funcionalidades em ordem de prioridade
 - Plano “grosseiro” de iterações – pelo menos para as primeiras
 - Plano de integração com outros sistemas
 - **Plano de testes “grosseiro”, se necessário**
 - **Plano de deployment**
 - **Estimativas de orçamento**
 - **Estimativas de cronograma e recursos**
 - **Crie um plano de projeto conforme o necessário**

Plano de Iteração

- Lista de funcionalidades, em ordem de prioridade
- Recursos
- Atividades / tarefas
- Plano de testes da iteração
- Critério de sucesso

Acompanhamento

- **Para iteração**
 - **Medição é baseada estritamente em termos de funcionalidades entregues**
 - Desenvolvidas
 - Testadas
 - Aceitas
- **Para fase / release**

Acompanhamento

- Funcionalidades “perdidas” numa iteração ficam para a próxima
- Mudanças são bem vindas e incorporadas com base na prioridade do cliente
- Time box – nunca perca datas!
- Correções de erros podem ficar para próxima iteração
- Constante adequação do plano à realidade
 - Gerenciamento de projetos!

Iteração – Propósito

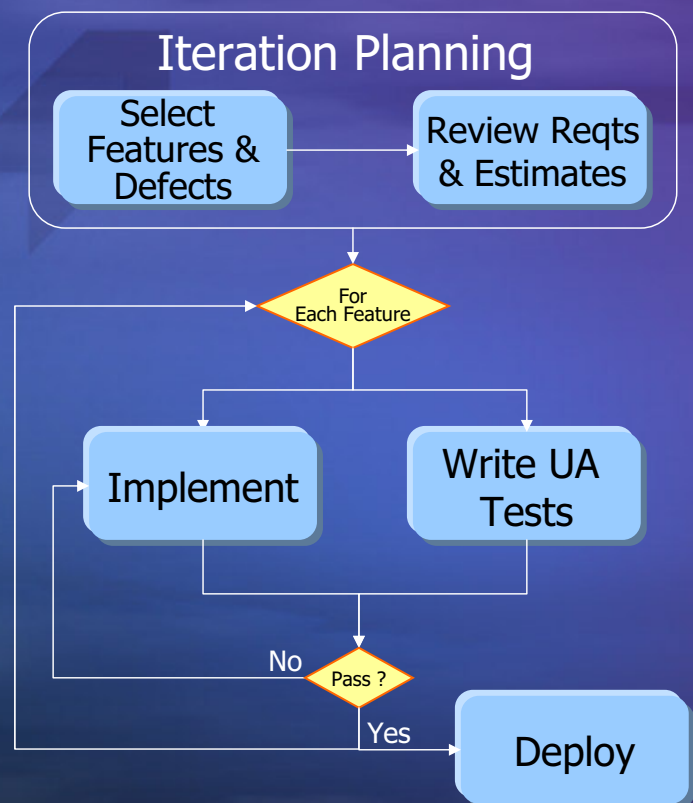
- Cada iteração resulta num conjunto de funcionalidades prontas com qualidade de produção
- Define o “quando” para entrega das funcionalidades
- Duração em torno de 2 semanas (malheável)
- Entrega funcionalidades prontas
- Demonstra progresso tangível
- Obtém feedback precoce do cliente / gerente de Produto

“Tasks” de Desenvolvimento para Planejamento e Acompanhamento

- **Funcionalidade**
 - Possui valor para o cliente
 - Pode ser melhorada, modificada, potencialmente retirada
 - “Definida precisamente” De forma a que se possa saber quando está pronta
 - Pode ser planejada, assinalada e estimada
- **Tarefa**
 - Parcela de trabalho a ser feita, geralmente uma única vez
 - Pode ser planejada, assinalada e estimada
- **Defeito**
 - Alum tipo de erro
 - É relacionado a alguma funcionalidade
 - Pode ser planejado, assinalado e estimado

Processo de Des/to – Uma iteração

- Uma visão geral
- Planeje uma iteração
 - A partir do topo da lista de funcionalidades/bugs priorizados
 - Selecione apenas o quanto você consegue fazer
- Implemente
 - Mod., Código, QA
- Funciona?
 - Passe pelos testes locais, faça check-in
 - Passe pelos testes na máquina de build
 - Feche essa issue!



Iterações – o processo

- A cada iteração o cliente obtém mais funcionalidades
- A prioridade é flexível para acomodar mudanças na demanda
- Funcionalidades são implementadas de acordo com a lista de prioridades
- A equipe, “just in time”, ouve novamente o briefing das funcionalidades escolhidas para a iteração
- Desenvolvedores garantem uma estimativa acurada, de granularidade fina
- Testadores garantem que podem “chegar” ao teste de aceite, para provar que a funcionalidade existe e está correta
- Entre 4-8hrs no início da iteração para o planejamento da mesma, para uma equipe de 5 a 10 profissionais

Iterações – o processo

- **Estimativas devem considerar o seguinte:**
 - Preencha apenas ~50% do tempo da equipe quando alocando funcionalidades – permita tempo para design, discussão, retrabalho, alimentação, etc
 - Eventuais “folgas” são preenchidas com a implementação de funcionalidades previstas para próxima iteração
- **Reveja/defina estimativas finas**
- **Discuta/determine como testar as funcionalidades desenvolvidas**
- **Não chame nenhum segmento de trabalho de “iteração” se o mesmo não incluir o build e a entrega de pelo menos uma funcionalidade**

Gerenciamento de Risco

- **Muito do que fazemos é sobre gerenciamento de risco**
 - Requisitos incorretos
 - Código não funcionar direito
 - Ultrapassar prazos
 - Questões de integração / entrega
- **Ágil – um estado mental – é também sobre gerenciamento de risco**

Ambiente apropriado

- Ferramentas
- Condições de trabalho
- Ambiente

Envolvimento continuado do cliente final

- **No mínimo, a cada fim de iteração**
- **Cliente deve ser parte da equipe**
- **Mudanças são bem vindas**
- **A prioridade é do cliente**

O produto final

- **Pode diferir grandemente do pedido original**
- **Pode conter muitas funcionalidades não pedidas originalmente**
- **Pode não conter muitas funcionalidades pedidas originalmente**
- **Atende às necessidades do cliente**

Conclusão

Para que somos pagos?

- Para entregar, o mais rápida e eficientemente possível, software que suporte o negócio de forma conveniente
- Não somos pagos para
 - Fazer planos
 - Acompanhar tarefas
 - Escrever código
 - Testar

Hábitos de projetos bem sucedidos

- **Acolha e dê poderes a profissionais de qualidade**
- **Torne simples abraçar a mudança**
- **Utilize um processo iterativo com envolvimento dos stakeholders**
- **Use ferramentas para automatizar o máximo possível**
- **Meça progresso via código funcionando**
- **Tenha sempre um estado mental ágil**
- **Reconheça o que você não sabe, mitigue riscos, questione e melhore continuamente**
- **Perca funcionalidades, nunca datas!**

Para onde ir daqui?

- **Técnicas de gerenciamento ágil de projetos**
- **Gerenciamento ágil de requisitos**
- **Metodologias Ágeis**
 - XP
 - Agile Modeling
 - Agile Unified Process
 - ICONIX
 - SCRUMs
 - Crystal



COMPUWARE®

www.compuware.com

Obrigado!

marcio.tierno@compuware.com