Processo e Metodologia

A T2Ti

Apresentação

A T2Ti nasce do sonho de três colegas que trabalhavam no maior banco da américa latina.

Tudo começa em 2007 com o lançamento do curso Java Starter. Logo depois veio o Siscom Java Desktop seguido de outros treinamentos.

Desde então a Equipe T2Ti se esforça para produzir material de qualidade que possa formar profissionais para o mercado, ensinando como desenvolver sistemas de pequeno, médio e grande porte.

Um dos maiores sucessos da Equipe T2Ti foi o Projeto T2Ti ERP que reuniu milhares de profissionais num treinamento dinâmico onde o participante aprendia na prática como desenvolver um ERP desde o levantamento de requisitos. Foi através desse treinamento que centenas de desenvolvedores iniciaram seu negócio próprio e/ou entraram no mercado de trabalho.

Em 2010 a T2Ti lança sua primeira aplicação para produção, o Controle Financeiro Pessoal. O sucesso foi tanto que saiu até em matéria no site Exame, ficando entre os 10 aplicativos mais baixados da semana.

Começa então a era de desenvolvimento de sistemas para alguns clientes exclusivos, pois o foco ainda era em desenvolvimento de treinamentos. A T2Ti desenvolve sistemas para o mercado nacional e internacional.

Atualmente a T2Ti se concentra nas duas vertentes: desenvolver sistemas e produzir treinamentos.

_____

Sumário

Processo e Metodologia

Processo e Metodologia

Introdução; Metodologia; Processo: Anos 70; Anos 80; Anos 90; Passos ou Atividades.

Processo | Abordagens

Modelo em Cascata; Prototipação; Desenvolvimento Iterativo e Incremental; Ciclo de Vida; RAD; Desenvolvimento Ágil de Software: ASD; DSDM; FDD; LSD; AUP; Adaptativo versus Preditivo.

Processo e Metodologia

Introdução

Desenvolver sistemas é uma arte. Um usuário de software não tem a menor noção do que é preciso para desenvolver um sistema. Infelizmente, muitos programadores e desenvolvedores também não. Como assim?

Existem muitos colegas, graduados ou não, que não sabem direito o que estão fazendo. Para desenvolver um sistema é preciso estudar e ler muito. Certa vez um professor me falou que na Europa, em um projeto de 1 ano, os profissionais passam 7 meses planejando e 3 meses executando. No Brasil, num mesmo projeto, passamos 3 semanas planejando e o restante do tempo executando.

Tem um ditado que diz: "nem oito nem oitenta". 70% do tempo planejando? Parece demais. Mas não dar atenção ao planejamento pode trazer sérias consequências.

Você precisa se esforçar para criar softwares elegantes. E o que seria isso?

Vou começar descrevendo o que é um software deselegante:

- É aquele software feito sem uma estrutura clara.
- É aquele do qual não se consegue reusar partes e que não se consegue entender como funciona sem uma boa carga de documentação (e muitas vezes nem assim).
- É aquele no qual uma pequena modificação em uma de suas características pode causar um não funcionamento generalizado.

Já um software elegante:

- Sua estrutura é intrinsecamente mais fácil de compreender.
- É auto documentado e pode ser compreendido em nível macro ou em detalhes.
- Ele é mais fácil de modificar: quando alguma de suas características é mudada, ele continua funcionando.

Processo e Metodologia

Introdução

Um dos maiores problemas que enfrentamos quando desenvolvemos software é a pressa. O cliente tem pressa. O usuário tem pressa. O desenvolvedor tem pressa. No final das contas o sistema fica uma porcaria.

As atividades de desenvolvimento de um sistema podem ser divididas em quatro:

- Análise
- Projeto
- Implementação
- Teste

Como é que as pessoas costumam fazer? A análise é uma entrevista com o dono da empesa e o estudo de alguns documentos utilizados ali. O projeto é um contrato que traz o preço do sistema e a data de entrega. Tudo isso ocorre em uma semana. A implementação é a fase mais pesada e o teste será feito pelo usuário quando o sistema for instalado.

Introdução

E assim o desenvolvedor entrega o sistema em um período curto e já fala do tal contrato de manutenção, onde ele irá, de fato, "terminar" o sistema, pois, como não houve planejamento adequado, quando os usuários começarem a usar o sistema (de fato estão testando), o desenvolvedor terá que implementar uma série que requisitos que ele não levantou, pois ele nem passou direito por essa fase.

Mas pra ele não tem problema, afinal, ele tem um contrato de manutenção. Com o tempo, o sistema virará uma "colcha de retalhos" e o desenvolvedor vai falar para a empresa que é preciso fazer outro, afinal, as coisas evoluem, temos novas tecnologias, etc. Já ouviu ou viu algo similar?

Tudo isso é causado pela falta de planejamento, pela pressa. E muitas vezes a pressa é do próprio cliente: "preciso desse sistema pro mês que vem. Consegue fazer?"

Se o desenvolvedor diz que não consegue, vem outro e diz que vai fazer. Entenda que é um problema cultural. A maioria das pessoas no nosso país pensa e age dessa forma. Eles querem chegar rápido e muitas vezes não sabem nem pra onde querem ir.

Alguns defendem que, regulando o mercado, o problema será resolvido. Não tem nada a ver. Regular o mercado só prejudica o consumidor. Por que o consumidor tem que pagar mais caro para uns profissionais que estão num "clube"? Quanto mais aberto o mercado, melhor.

Eu já mencionei que muitos colegas aprendem a programar em uma linguagem e já começam a produzir software para o mercado sem nunca passar numa faculdade. Eu não sou contra isso, de forma alguma. Na realidade, muitos que nunca pisaram numa faculdade tem mais conhecimento que os que estão ensinando por lá. Tem muito profissional autodidata.

Introdução

Mas, afinal, como deveriam funcionar as atividades de desenvolvimento de um sistema?

Análise

A análise enfatiza a investigação do problema. O objetivo da análise é levar o analista a investigar e a descobrir. Para que esta etapa seja realizada em menos tempo e de forma mais precisa, deve-se ter um bom método de trabalho. Alguns seguem à risca um método inventado na academia. Outros adotam os melhores aspectos de diversos métodos e criam uma maneira única de trabalho.

Pode-se dizer que o resultado da análise é o enunciado de um problema, e que o projeto será a sua resolução. Problemas mal enunciados podem até ser resolvidos, mas a solução não corresponderá às expectativas.

A qualidade do processo de análise é importante porque um erro de concepção resolvido na fase de análise tem um determinado custo. Na fase de projeto terá um custo maior. Na fase de implementação, o curso será maior ainda. E na fase de implantação do sistema tem um custo relativamente astronômico.

Eu já ouvi um relato que ocorreu numa grande empresa. Imaginaram por lá que seria bom fazer um sistema onde as pessoas pudessem comprar produtos através de um balcão internacional. O pagamento seria feito exclusivamente por cartão de crédito. Então montaram a equipe, levantaram os requisitos, desenvolveram a solução em ambiente de desenvolvimento (dados e respostas fictícios) e quando foram para o ambiente de homologação foi preciso entrar em contato com as operadoras de cartão para realizar um teste online. Surpresa: a operadoras, na época, não forneciam esse serviço.

Processo e Metodologia

Introdução

Projeto

A fase de projeto enfatiza a proposta de uma solução que atenda os requisitos da análise. Então, se a analise é uma investigação para tentar descobrir o que o cliente quer, o projeto consiste em propor uma solução com base no conhecimento adquirido na análise.

Implementação

A utilização de técnicas sistemáticas nas fases de análise e projeto faz com que o processo de geração de código possa ser automatizado.

Neste caso, cabe ao programador dominar as características específicas das linguagens, ferramentas, frameworks e estruturas de dados para adaptar o código gerado aos requisitos indicados quando necessário.

Testes

A fase de testes pode envolver uma série de procedimentos que vão desde os testes unitários, feitos pelo programador, para verificar se os componentes gerados atendem à especificação do projetista, até os testes de caso de uso, normalmente efetuados por um analista experiente, que visam verificar a adequação do sistema aos requisitos inicialmente levantados.

Enfim, são essas as atividades principais quando desenvolvemos um software. Não existe uma forma única e certa de fazer. Existem vários processos e várias metodologias.

Às vezes os termos "processo" e "metodologia" se confundem. Alguns autores se referem a algo como "processo" e outros se referem como "metodologia". E ainda temos o termo "método"!

Processo e Metodologia

Metodologia

No estudo da Engenharia de Software e no Gerenciamento de Projetos, uma metodologia é um conjunto estruturado de práticas que pode ser repetível durante o processo de produção de software. Exemplos desse conjunto estruturado de práticas: material de treinamento, programas de educação formais, planilhas, diagramas, etc.

Metodologias de Engenharia de Software abrangem muitas disciplinas, incluindo Gerenciamento de Projetos, e as suas fases como: análise, projeto, codificação, teste, e mesmo a garantia da qualidade.

Existem uma ampla discussão científica a respeito das palavras: metodologia e método. Várias pessoas utilizam-nas como sinônimos.

No entanto, é importante destacar a diferença entre as duas. O método tem relação direta com o processo (como fazer algo?), e a metodologia é a área de estudo de um ou vários métodos. Esta definição está embasada na etimologia de ambas as palavras. Metodologia e Método possuem como radical Grego, méthodos (caminho para chegar a um fim) e logia (estudo de).

Processo e Metodologia

Metodologia

Na Engenharia de Software esta discussão assume um caráter contínuo. Alguns autores assumem que o método caracteriza o processo com uma série de atividades, utilizado na construção de um software, enquanto que uma metodologia se traduz na codificação de um conjunto de boas práticas recomendadas, acompanhada de material de treinamento, programas de educação formal, planilhas e diagramas.

Dentro deste contexto, o método de Engenharia de Software pode ser considerado como parte da metodologia.

Outros autores direcionam a metodologia com base em uma abordagem filosófica do problema. Dentro deste contexto é possível afirmar que a Engenharia de Software é rica em métodos e pobre em metodologias.

Dentro desta abordagem temos:

- Metodologia Estruturada.
- Metodologia Orientada a Objetos.
- Metodologias de Desenvolvimento Ágil.

Processo e Metodologia

Processo

Um processo de desenvolvimento de software é um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto de software. É estudado dentro da área de Engenharia de Software, sendo considerado um dos principais mecanismos para se obter software de qualidade e cumprir corretamente os contratos de desenvolvimento, sendo uma das respostas técnicas adequadas para resolver a Crise do Software.

PRESSMAN (2011) chama a tal crise de aflição crônica, visto que nos acompanha há 30 anos.

Quais os problemas que causam tal aflição? PRESSMAN (2011) menciona três:

- As estimativas de prazo e de custo frequentemente são imprecisas.
- A produtividade das pessoas da área de software não tem acompanhado a demanda por seus serviços.
- A qualidade software muitas vezes é menor que a adequada.

O autor descreve as possíveis causas e mitos para essa aflição "sem fim".

Processo e Metodologia

Processo

E qual seria a solução? Como ter certeza do prazo e do custo de um software? Como entregar aquilo que o cliente deseja? Segundo PRESMANN (2011), não existe uma abordagem em particular que seja a melhor. No entanto, ao combinarmos métodos abrangentes para todas as fases de desenvolvimento do software, melhores ferramentas para automatizar esses métodos, melhores técnicas para a garantia da qualidade e uma filosofia de coordenação predominante, podemos conseguir uma disciplina para o desenvolvimento do software: a Engenharia de Software.

Foi o estudo e constante evolução dessa disciplina que nos levou às metodologias, métodos e processos que temos hoje.

Os passos ou atividades do processo podem ser listados da seguinte forma: Análise Econômica; Análise de Requisitos; Especificação; Arquitetura; Implementação (ou codificação); Testes; Documentação; Suporte e Treinamento; Manutenção.

Antes de detalharmos esses passos, veremos um breve histórico do processo de desenvolvimento de software.

Processo e Metodologia

Processo | Anos 70

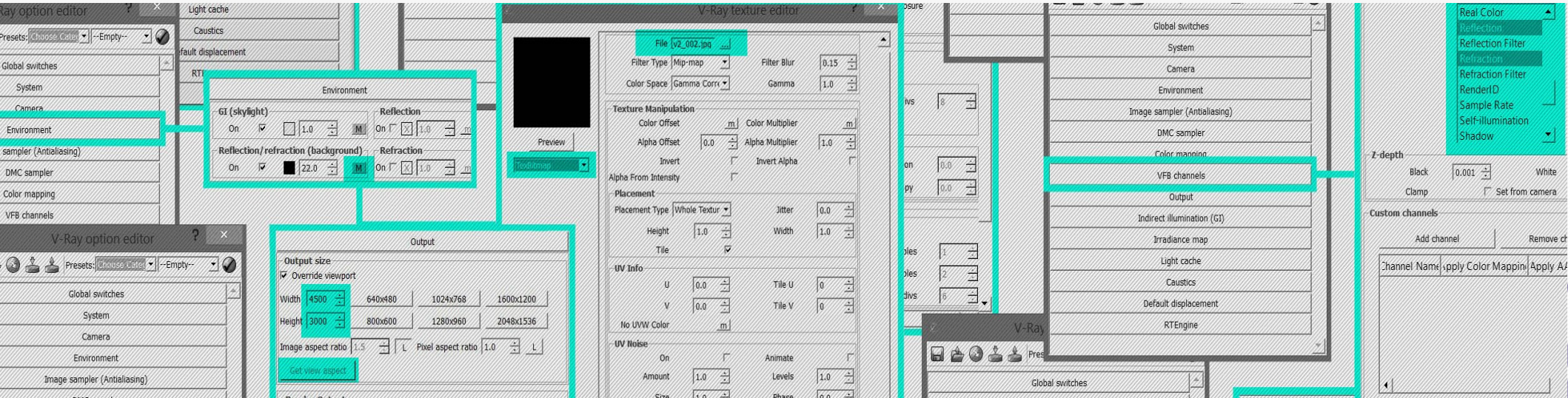
Análise Estruturada

A análise estruturada é uma atividade de construção de modelos. Utiliza uma notação com a finalidade de retratar o fluxo e o conteúdo das informações utilizadas pelo sistema, dividir o sistema em partições funcionais e comportamentais e descrever a essência daquilo que será construído. Utiliza os modelos ambiental e comportamental.

O modelo ambiental descreve o ambiente no qual o sistema se insere, ou seja, descreve o contexto do sistema.

O modelo comportamental descreve as ações que o sistema deve realizar para responder da melhor forma aos eventos definidos no modelo ambiental. São utilizadas várias técnicas:

- Diagrama de fluxos de dados (DFD).
- Dicionário de dados (DD).
- Diagrama de entidades e associações (ou relacionamentos) (Diagrama entidade relacionamento [DER] ou Modelo de entidades e relacionamentos [MER]).
- Especificação de processos (EP) – (DESENHO).
- Diagrama de transição de estados (DTE).



Processo e Metodologia

Processo | Anos 80

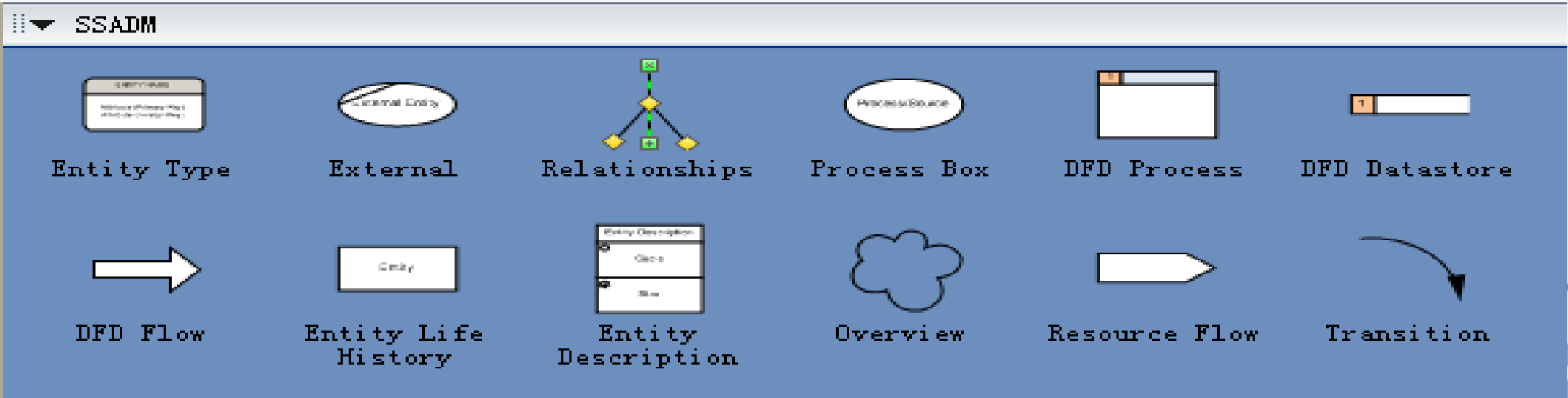
SSADM

SSADM é um acrônimo para Structured Systems Analysis and Design Methodology (Estrutura de Análise de Sistemas e Metodologia de Projeto). É uma metodologia de análise e desenho estruturado proposta por Learmonth em 1981.

Muito usada durante as décadas de 1980 e 1990. Caiu em desuso com o desenvolvimento da UML.

SSM

SSM é um acrônimo para Soft System Methodology. Foi desenvolvida no departamento de sistemas de administração de informações da Universidade de Lancaster, por uma equipe liderada por Peter Checkland, com o intuito de tentar resolver problemas em empresas, de maneira sistêmica e em situações onde os mesmos se mostravam pouco estruturados ou mesmo obscuros. É um processo baseado em modelos sistêmicos, levando à escolha de uma ação propositada.



Processo e Metodologia

Processo | Anos 90

OOP

Surge a famosa programação orientada a objetos.

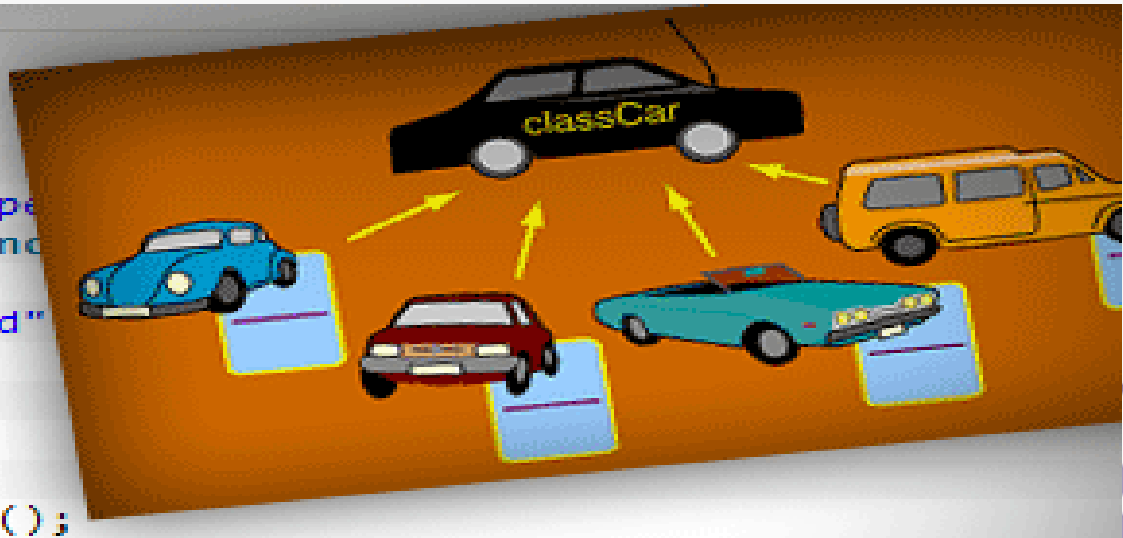
A orientação a objetos é um modelo de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

RAD

RAD é um acrônimo para Rapid Application Development (Desenvolvimento Rápido de Aplicação). É um modelo de processo de desenvolvimento de software interativo e incremental que enfatiza um ciclo de desenvolvimento extremamente curto (entre 60 e 90 dias).

O termo foi registrado por James Martin em 1991 e tem substituído gradativamente o termo de prototipação rápida que já foi muito utilizada no passado.

```
view plain  copy to clipboard  print  ?
01.  //Define the Car class
02.  function Car() { }
03.  Car.prototype.speed= 'Car Spe
04.  Car.prototype.setSpeed = func
05.      this.speed = speed;
06.      alert("Car speed changed"
07.  }
08.
09.  //Define the Ferrari class
10.  function Ferrari() { }
11.  Ferrari.prototype = new Car();
```



Processo | Anos 90

DSDM

DSDM é um acrônimo para Dynamic Systems Development Method (Metodologia de Desenvolvimento de Sistemas Dinâmicos).

é uma abordagem originalmente baseada em "Desenvolvimento Rápido de Aplicação" (RAD).

É um processo iterativo que a cada iteração (ou incremento) possui somente a quantidade de trabalho suficiente. Isto facilita o movimento para o próximo incremento.

Seu objetivo é entregar softwares no tempo e com custo estimados através do controle e ajuste de requisitos ao longo do desenvolvimento.

É mais uma abordagem de desenvolvimento de software ágil que oferece uma metodologia para construir e manter sistemas que atendem restrições de prazo apertado através do uso da prototipagem incremental.

Seu formato é propriedade da Agile Alliance.

DSDM®
CONSORTiUM

Processo e Metodologia

Processo | Anos 90

SCRUM

É um método de desenvolvimento ágil de software criado por Jeff Sutherland e a sua equipe de desenvolvimento de software. O método foi criado no início dos anos 90 e recentemente ganhou incrementos nos métodos gráficos através de Schwaber e Beedle.

Nos aprofundaremos na Parte II deste livro.

TSP

O Team Software Process tenciona ajudar equipes a desenvolver produtos de software de modo eficaz, guia equipes no lançamento e execução de projetos, criando um ambiente de trabalho em grupo onde o trabalho individual é disciplinado e voltado à equipe.

Objetiva aumentar a produtividade e o desempenho, reduzir custos de desenvolvimento total e também de manutenção.

Processo e Metodologia

Processo | Anos 90

RUP

O RUP, abreviação de Rational Unified Process (ou Processo Unificado Rational), é um processo proprietário de Engenharia de Software criado pela Rational Software Corporation, adquirida pela IBM.

Fornece técnicas a serem seguidas pelos membros da equipe de desenvolvimento com o objetivo de aumentar a sua produtividade no processo de desenvolvimento.

XP

XP é um acrônimo para eXtreme Programming (Programação Extrema). É mais uma abordagem de desenvolvimento de software ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software.

Processo e Metodologia

Processo | Passos ou Atividades

Análise Econômica

Visa a estabelecer se o projeto de software gerará lucro, e se a receita gerada será o suficiente para cobrir os custos.

Análise de Requisitos

A extração dos requisitos de um cliente.

Especificação

Tarefa de descrever precisamente o software que será escrito.

Arquitetura

A arquitetura de um sistema de software remete a uma representação abstrata daquele sistema. A aquitetura é concernente à garantia de que o sistema de software irá ao encontro de requisitos do produto, como também assegurar que futuros requisitos possam ser atendidos. A etapa da arquitetura também direciona as interfaces entre os sistemas de software e outros produtos de software, como também com o hardware básico ou com o sistema operacional.

Processo | Passos ou Atividades

Implementação (ou codificação)

A transformação de um projeto para um código deve ser a parte mais evidente do trabalho da engenharia de software, mas não necessariamente a sua maior porção.

Testes

A fim de validar o produto de software, cada funcionalidade de cada módulo de ser testada, levando em consideração a especificação feita na fase de projeto. O principal resultado é o relatório de testes, que contém as informações relevantes sobre erros encontrados no sistema, e seu comportamento em vários aspectos.

Documentação

Uma importante tarefa é a documentação do projeto interno do software para propósitos de futuras manutenções e aprimoramentos.

Suporte e Treinamento

Como parte da fase de desenvolvimento, é muito importante o treinamento para os usuários de software mais entusiasmados, alternando o treinamento entre usuários neutros e usuários favoráveis ao software.

Manutenção

A manutenção e melhoria de software lidam com a descoberta de novos problemas e requisitos. Ela pode tomar mais tempo que o gasto no desenvolvimento inicial do mesmo.

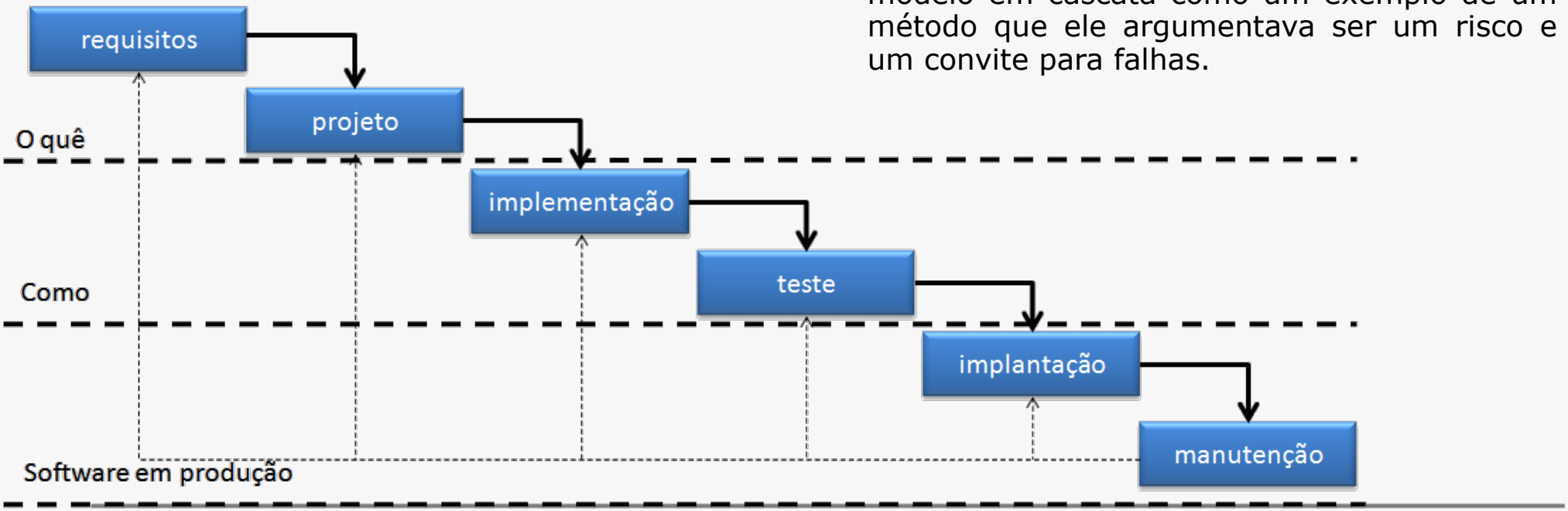
Não somente pode ser necessário adicionar códigos que combinem com o projeto original, mas determinar como o software trabalhará em algum ponto depois da manutenção estar completa, pode requerer um significativo esforço por parte de um engenheiro de software.

Processo | Abordagens

Modelo em Cascata

É um modelo de desenvolvimento de software sequencial no qual o desenvolvimento é visto como um fluir constante para frente (como uma cascata) através das fases de análise de requisitos, projeto, implementação, testes (validação), implantação e manutenção de software.

A origem do termo cascata é frequentemente citado como sendo de um artigo publicado em 1970 por W. W. Royce. Ironicamente, Royce defendia um abordagem iterativa para o desenvolvimento de software e nem mesmo usou o termo cascata. Royce originalmente descreve o que é hoje conhecido como o modelo em cascata como um exemplo de um método que ele argumentava ser um risco e um convite para falhas.



Processo | Abordagens

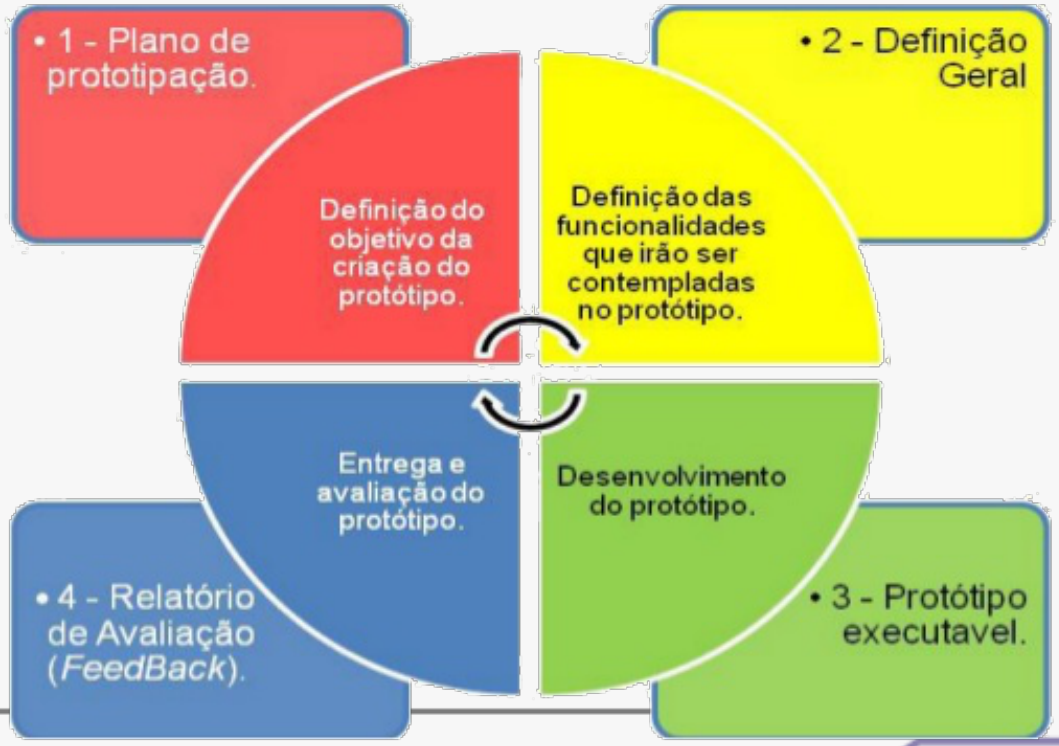
Prototipação

É uma abordagem baseada em uma visão evolutiva do desenvolvimento de software, afetando o processo como um todo. Envolve a produção de versões iniciais - protótipos (análogo a maquetes para a arquitetura) - de um sistema futuro com o qual é possível realizar verificações e experimentos, com o intuito de avaliar algumas de suas características antes que o sistema venha realmente a ser construído, de forma definitiva.

Em muitos casos o cliente define somente um conjunto de objetivos gerais para o Sistema (Software), mas não foi capaz de gerar requisitos bem definidos. Essa seria uma ocasião oportuna para criar um protótipo.

O protótipo pode ser feito com a mesma ferramenta que o sistema será desenvolvido. Por exemplo, pode-se criar rapidamente os modelos de tela no Delphi para uma aplicação que será posteriormente feita no Delphi.

Uma desvantagem é que o cliente vê o protótipo e pode achar que o sistema está quase pronto. Quando você explicar que se trata apenas de uma amostra, de um protótipo, ele pode achar que foi perda de tempo.



Processo | Abordagens

Desenvolvimento Iterativo e Incremental

O Desenvolvimento Iterativo e Incremental é um dos clássicos modelos de processo de desenvolvimento de software criado em resposta às fraquezas do modelo em cascata, o mais tradicional. Os dois padrões mais conhecidos de sistemas iterativos de desenvolvimento são o RUP (Processo Unificado da Rational) e o desenvolvimento ágil de software. Por isso o desenvolvimento iterativo e incremental é também uma parte essencial da Programação Extrema e outros.

Desenvolvimento Incremental é uma estratégia de planejamento estagiado em que várias partes do sistema são desenvolvidas em paralelo, e integradas quando completas. Não implica, requer ou pressupõe desenvolvimento interativo ou em cascata – ambos são estratégias de retrabalho.

A alternativa ao desenvolvimento incremental é desenvolver todo o sistema com uma integração única.

Desenvolvimento Iterativo e Incremental

Desenvolvimento iterativo é uma estratégia de planejamento de retrabalho em que o tempo de revisão e melhorias de partes do sistema é pré-definido.

Isto não pressupõe desenvolvimento incremental, mas funciona muito bem com ele.

Uma diferença típica é que a saída de um incremento não é necessariamente assunto de um refinamento futuro e seu teste ou retorno do usuário não é utilizado como entrada para planos de revisão ou especificações para incrementos sucessivos.

Ao contrario, a saída de uma iteração é examinada para modificação, e especialmente para revisão dos objetivos das iterações sucessivas.

Os termos Iterativo e Interativo costumam se confundir e muitos não sabem como utilizá-los. A Wikipedia nos dá uma pista dessa confusão:

"O nome iterativo (fazer de novo; REITERAR; REPETIR) e interativo (atividade ou trabalho compartilhado, em que existem trocas e influências recíprocas) se confundem em função até dos seus significados. Se imaginarmos o desenvolvimento de sistemas como puramente a criação de programas dentro da área de TI, sem a intervenção dos usuários, até que eles estejam coerentes com os requisitos de software, o correto é estarmos usando a palavra ITERATIVO, devido a repetição dessa ação.

No entanto, se imaginarmos que a cada repetição deve-se interagir com os usuários para que o sistema cada vez mais corresponda as expectativa deles, o correto seria usar, a rigor, a palavra INTERATIVO. Mas, por convenção (ou por repetição), a grande maioria dos autores de Engenharia de Software adotam a palavra ITERATIVO para identificar este tipo de desenvolvimento."

Desenvolvimento Iterativo e Incremental | Ciclo de Vida

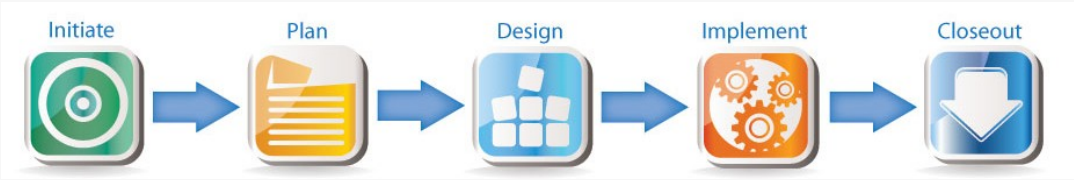
A ideia básica por trás da abordagem iterativa é desenvolver um sistema de software incremental, permitindo ao desenvolvedor tirar vantagem daquilo que foi aprendido durante a fase inicial de desenvolvimento de uma versão do sistema. O aprendizado ocorre simultaneamente tanto para o desenvolvedor, quanto para o usuário do sistema.

Os passos fundamentais do processo estão em iniciar o desenvolvimento com um subconjunto simples de Requisitos de Software e iterativamente alcançar evoluções subsequentes das versões até o sistema todo estar implementado. A cada iteração, as modificações de projeto são feitas e novas funcionalidades são adicionadas.

O projeto em si consiste da etapa de inicialização, iteração e da lista de controle do projeto.

A etapa de inicialização cria uma versão base do sistema. O objetivo desta implementação inicial é criar um produto para que o usuário possa avaliar. Ele deve oferecer um exemplo dos aspectos chave do problema e prover uma solução que seja simples o bastante para que possa ser compreendida e implementada facilmente.

Para guiar o processo iterativo, uma lista de controle de projeto é criada. Ela conterá um registro de todas as tarefas que necessitam ser realizadas. Isto inclui itens tais como novas características a serem implementadas e áreas para serem projeto na solução atual. A lista de controle deve ser continuamente revisada como um resultado da fase de análise.



Desenvolvimento Iterativo e Incremental | Ciclo de Vida

A etapa iterativa envolve o reprojeto e a implementação das tarefas da lista de controle do projeto, além da análise da versão corrente do sistema.

O objetivo para o projeto de implementação de qualquer iteração é ser simples, direto e modular, preparado para suportar reprojeto neste estágio ou como uma tarefa a ser adicionada na lista de controle do projeto.

O código pode, em alguns casos, representar uma fonte maior da documentação do sistema. A análise de uma interação é baseada no *feedback* do usuário e facilidades da análise do programa disponíveis.

As estruturas de análise envolvidas são a modularidade, a usabilidade, a reusabilidade e a eficiência e obtenção dos objetivos. A lista de controle do projeto é modificada à luz dos resultados da análise.

É bom frisar que o Ciclo de Vida não é um aspecto pertencente apenas ao Desenvolvimento Iterativo e Incremental. Os modelos de ciclo de vida são o esqueleto, ou as estruturas pré-definidas nas quais encaixamos as fases do processo.

O modelo de ciclo de vida é a primeira escolha a ser feita no processo de software. A partir desta escolha definir-se-á desde a maneira mais adequada de obter as necessidades do cliente, até quando e como o cliente receberá sua primeira versão operacional do sistema.

Muitas vezes são adotados vários tipos de ciclo de vida durante o desenvolvimento de um software, mesmo que se utilize apenas um processo. Os ciclos de vida se comportam de maneira sequencial (fases seguem determinada ordem) e/ou incremental (divisão de escopo) e/ou iterativa (retroalimentação de fases) e/ou evolutiva (software é aprimorado).

Processo | Abordagens

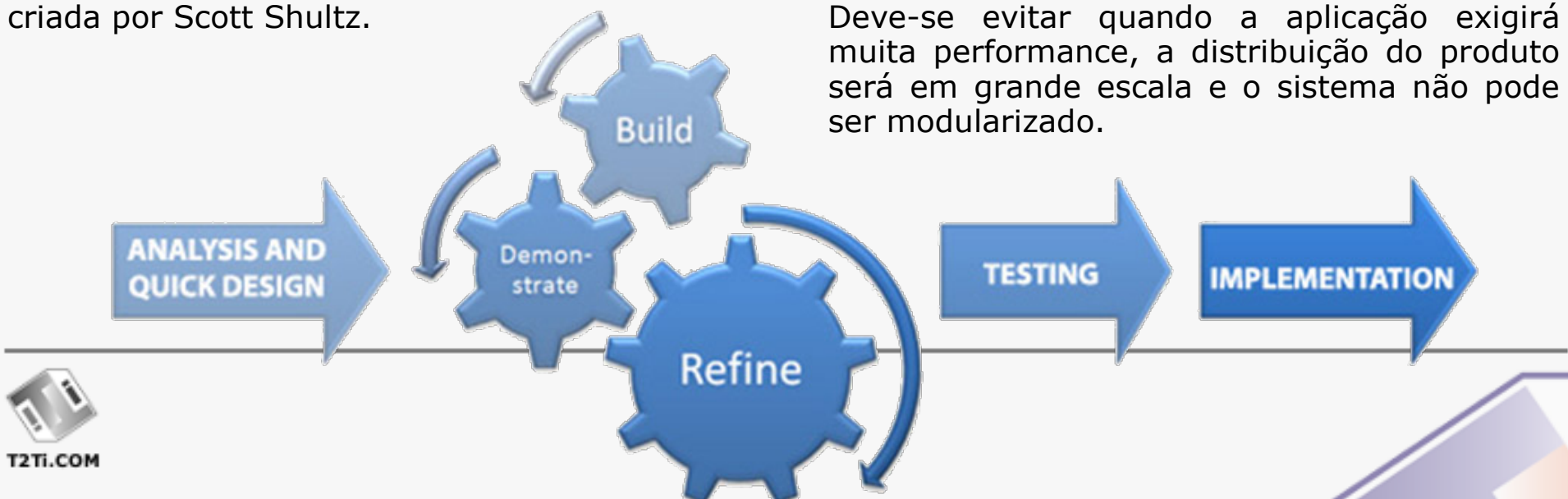
RAD

Rapid Application Development (RAD) ou Desenvolvimento Rápido de Aplicação (em português), é um modelo de processo de desenvolvimento de software interativo e incremental que enfatiza um ciclo de desenvolvimento extremamente curto (entre 60 e 90 dias).

Nos anos 80 os trabalhos de Barry Boehm (modelo de processo em espiral) e Tom Gilb (modelo de processo evolucionário) serviram de base para uma metodologia chamada de Rapid Iterative Production Prototyping (RIPP) criada por Scott Shultz.

James Martin estendeu o RIPP agregando valores de outros processos tornando-o maior e mais formal sendo assim denominado de RAD. O RAD foi finalmente formalizado em 1991 com a publicação de um livro.

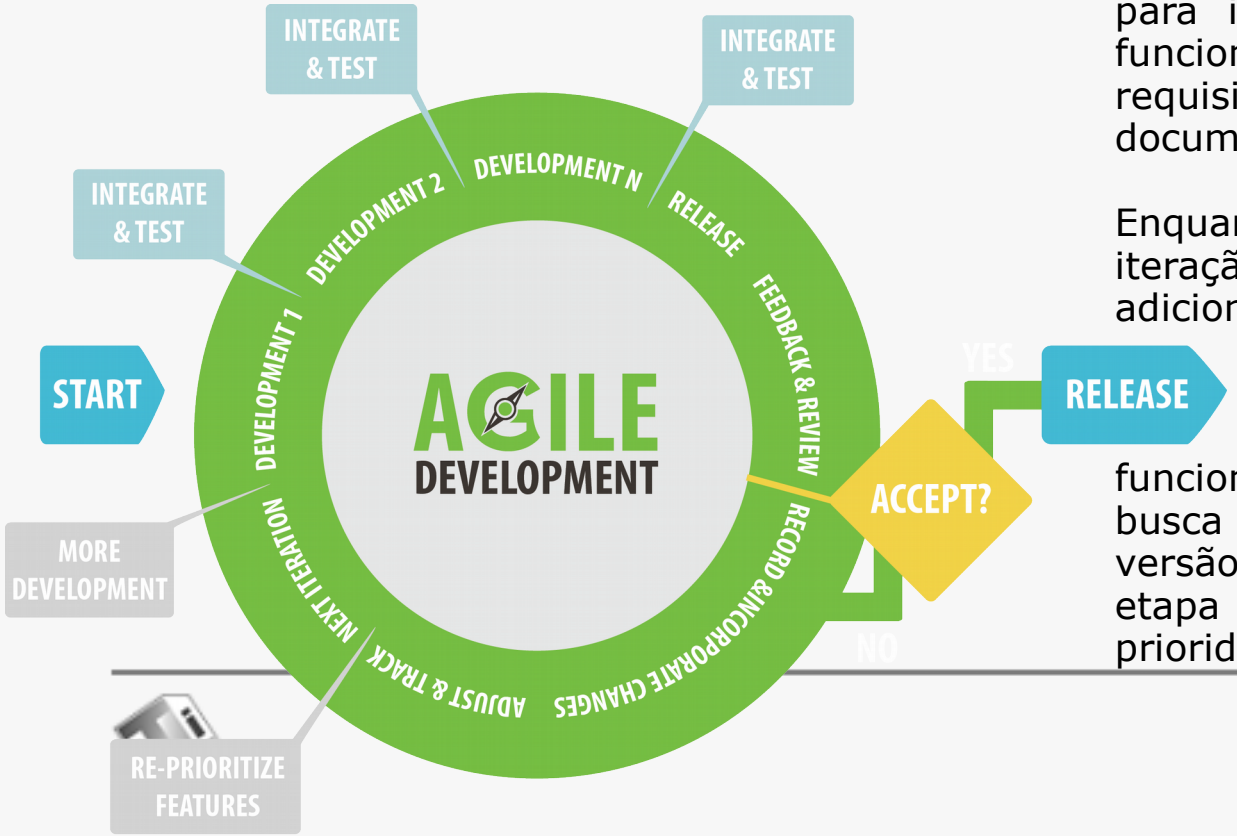
A maior vantagem do processo é o fato de a aplicação ficar pronta muito rápido. É apropriado para o desenvolvimento de aplicações "stand alone", onde a performance não é tão importante, a distribuição do produto é pequena e onde o sistema pode ser dividido em vários módulos independentes. Deve-se evitar quando a aplicação exigirá muita performance, a distribuição do produto será em grande escala e o sistema não pode ser modularizado.



Processo | Abordagens

Desenvolvimento Ágil de Software

Agile Software Development (Desenvolvimento Ágil de Software) ou Método Ágil é um conjunto de metodologias de desenvolvimento de software. O desenvolvimento ágil, tal como qualquer metodologia de software, providencia uma estrutura conceitual para gerenciar projetos de engenharia de software.



A maioria dos métodos ágeis tenta minimizar o risco pelo desenvolvimento do software em curtos períodos, chamados de iteração, os quais gastam tipicamente menos de uma semana a até quatro. Cada iteração é como um projeto de software em miniatura de seu próprio, e inclui todas as tarefas necessárias para implantar o mini incremento da nova funcionalidade: planejamento, análise de requisitos, projeto, codificação, teste e documentação.

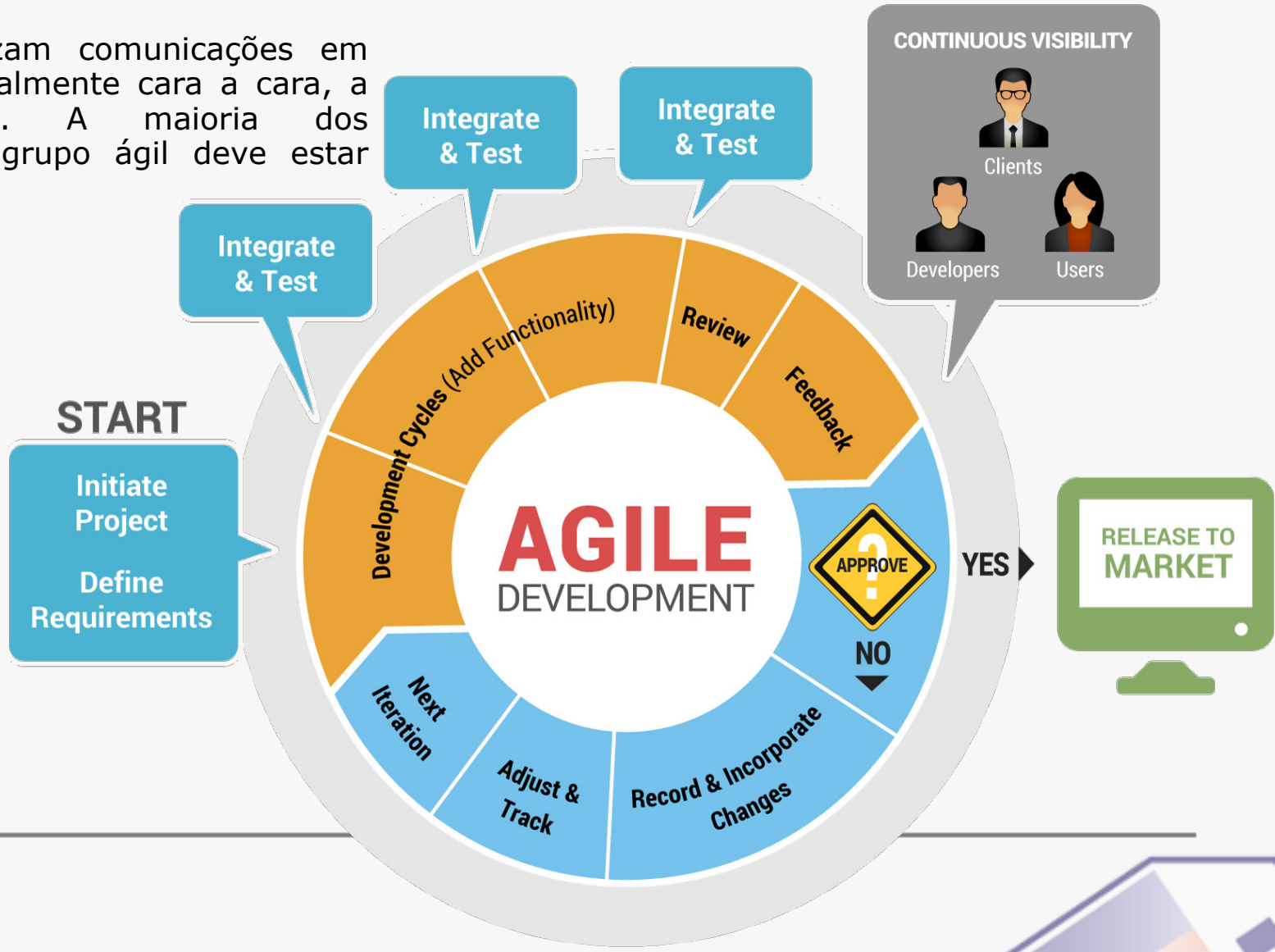
Enquanto em um processo convencional, cada iteração não está necessariamente focada em adicionar um novo conjunto significativo de funcionalidades, um projeto de software ágil busca a capacidade de implantar uma nova versão do software ao fim de cada iteração, etapa a qual a equipe responsável reavalia as prioridades do projeto.

Processo | Abordagens

Desenvolvimento Ágil de Software

Métodos ágeis enfatizam comunicações em tempo real, preferencialmente cara a cara, a documentos escritos. A maioria dos componentes de um grupo ágil deve estar agrupada em uma sala.

Isso inclui todas as pessoas necessárias para terminar o software: no mínimo, os programadores e seus clientes (clientes são as pessoas que definem o produto, eles podem ser os gerentes, analistas de negócio, ou realmente os clientes). Nesta sala devem também se encontrar os testadores, projetistas de iteração, redatores técnicos e gerentes.



Desenvolvimento Ágil de Software

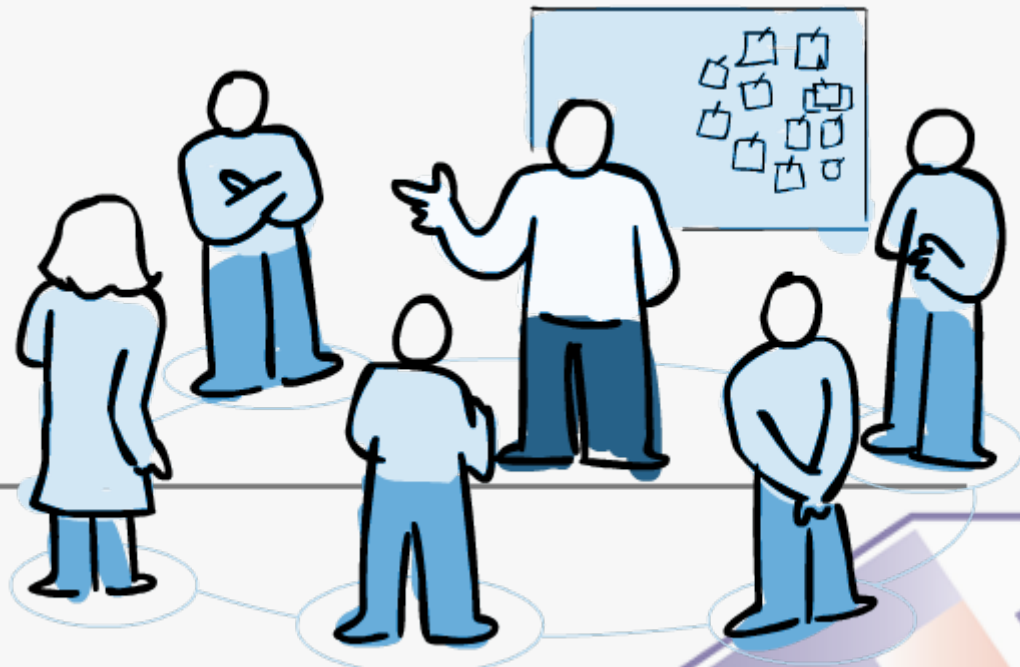
Métodos ágeis também enfatizam trabalho no software como uma medida primária de progresso. Combinado com a comunicação cara a cara, métodos ágeis produzem pouca documentação em relação a outros métodos, sendo este um dos pontos que podem ser considerados negativos. É recomendada a produção de documentação que realmente será útil.

A página do manifesto ágil (**Agile Manifest**) nos informa os valores do desenvolvimento ágil de software:

- Indivíduos e interações sobre processos e ferramentas.
- Software funcional sobre documentação abrangente.
- Colaboração do cliente sobre negociação de contratos.
- Responder a mudanças sobre seguindo um plano.

As definições modernas de desenvolvimento de software ágil evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados", caracterizados por uma pesada regulamentação, regimentação e micro gerenciamento usado no modelo em cascata para desenvolvimento.

O processo originou-se da visão de que o modelo em cascata era burocrático, lento e contraditório.



Desenvolvimento Ágil de Software

Uma visão que levou ao desenvolvimento de métodos ágeis e iterativos era retorno às práticas de desenvolvimento vistas nos primórdios da história do desenvolvimento de software.

No início os métodos ágeis eram conhecidos como "métodos leves". Em 2001, membros proeminentes da comunidade se reuniram em Snowbird e adotaram o nome métodos ágeis, tendo publicado o Manifesto Ágil, documento que reúne os princípios e práticas desta metodologia de desenvolvimento.

Mais tarde, algumas pessoas formaram a Agile Alliance, uma organização não lucrativa que promove o desenvolvimento ágil.

Os métodos ágeis iniciais incluíam Scrum (1986), Crystal Clear, Programação extrema (1996), Adaptive Software Development, Feature Driven Development e Dynamic Systems Development Method (1995).

Estudaremos em outro material os métodos SCRUM e XP.

Faremos agora um resumo de alguns métodos ágeis conhecidos e utilizados no mercado.

Processo | Abordagens

Desenvolvimento Ágil de Software | ASD

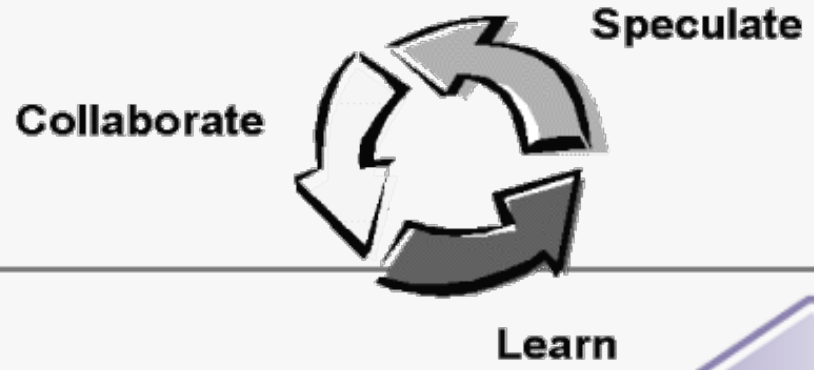
Jim Highsmith propôs o Desenvolvimento de Software Adaptativo (Adaptative Software Development - ASD) como uma técnica para construção de software e sistemas altamente complexos. Esse modelo se concentra na colaboração e auto-organização das equipes.

O criador do Modelo Adaptativo define um ciclo de vida para o modelo baseando-se em três fases: especulação, colaboração e aprendizagem.

Na fase de especulação o projeto é iniciado e os ciclos adaptáveis são planejados. Tal planejamento utiliza as informações contidas no inicio do projeto como: a missão do cliente, restrições do projeto e os requisitos básicos. Os requisitos básicos serão utilizados para definir o conjunto de ciclos da versão, ou seja, os incrementos de software operacional. Vale salientar que esse plano de ciclos sofrerá mudanças.

A colaboração envolve confiança, críticas sem animosidade, auxílio, trabalho árduo, comunicação dos problemas ou preocupações de forma a conduzir a ações efetivas. Assim sendo, a colaboração ajuda bastante no levantamento de necessidades e especificações.

O aprendizado é um elemento-chave para que se possa conseguir uma equipe auto-organizada. O aprendizado irá ajudar a todos os desenvolvedores a aumentar os níveis reais de entendimento. Com base nisso, as equipes ASD aprendem através de três maneiras: grupos focados, revisões técnicas e autópsias de projetos.



Processo | Abordagens

Desenvolvimento Ágil de Software | DSDM

O DSDM - Dynamic System Development Method (Método de Desenvolvimento de Sistemas Dinâmicos) é mais uma abordagem de desenvolvimento de software ágil que oferece uma metodologia para construir e manter sistemas que atendem restrições de prazo apertado através do uso da prototipagem incremental.

Este método é mantido pela DSDM Consortium (www.dsdm.org). Este grupo é mundial e conta com diversas empresas que são membros do grupo que mantém o DSDM.

É um processo iterativo onde cada iteração (ou incremento) possui somente a quantidade de trabalho suficiente. Isto facilita o movimento para o próximo incremento.

Detalhes remanescentes são completados depois quando outros requisitos forem conhecidos ou alterações tiverem sido solicitadas pelo cliente.

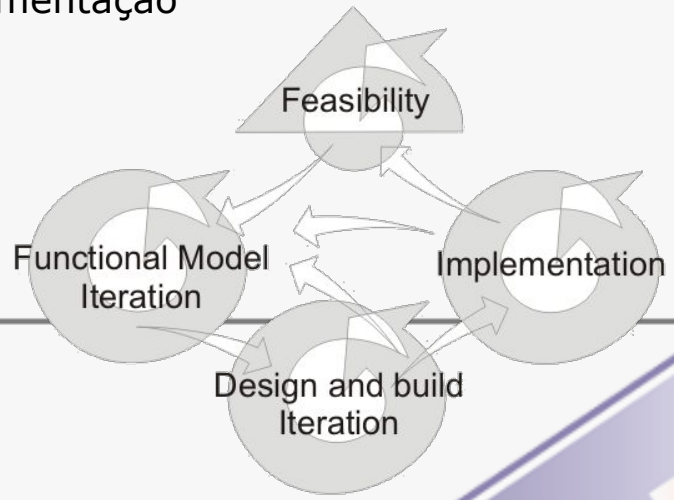
No DSDM temos a definição de um modelo de processos ágeis, chamado ciclo de vida DSDM, que define três ciclos iterativos diferentes que são precedidos por duas atividades de ciclo de vida adicionais.

Atividades

- Estudo da Viabilidade
- Estudo do Negócio

Ciclos Iterativos

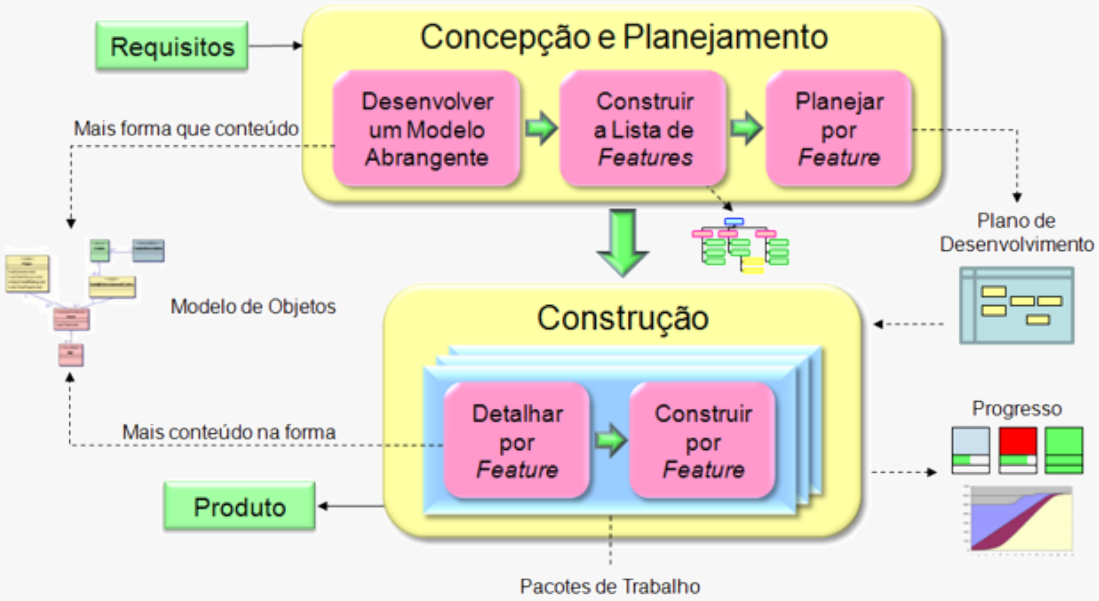
- Iteração de Modelos Funcionais
- Iteração de Projeto e Desenvolvimento
- Implementação



Processo | Abordagens

Desenvolvimento Ágil de Software | FDD

O Desenvolvimento Dirigido a Funcionalidade ou Feature Driven Development (FDD) foi criado por Peter Coad e seus colegas. O trabalho foi aperfeiçoado por Stephen Palmer e John Felsing.



O FDD evidencia a colaboração entre as pessoas da equipe, gerencia os problemas e complexidades de projetos utilizando a decomposição baseada em funcionalidades que é seguida pela integração dos incrementos de software e por fim enfatiza a comunicação de detalhes técnicos usando formas verbais, gráficos e texto.

Além disso, o FDD enfatiza as atividades de garantia da qualidade por meio da estratégia de desenvolvimento incremental, inspeções de código e de projeto, auditorias, coleta de métricas e o uso de padrões para análise, projeto e construção.

O FDD define cinco processos: desenvolver um modelo geral, construir uma lista de funcionalidades, planejar por funcionalidades, projetar por funcionalidade e desenvolver por funcionalidade.

Processo | Abordagens

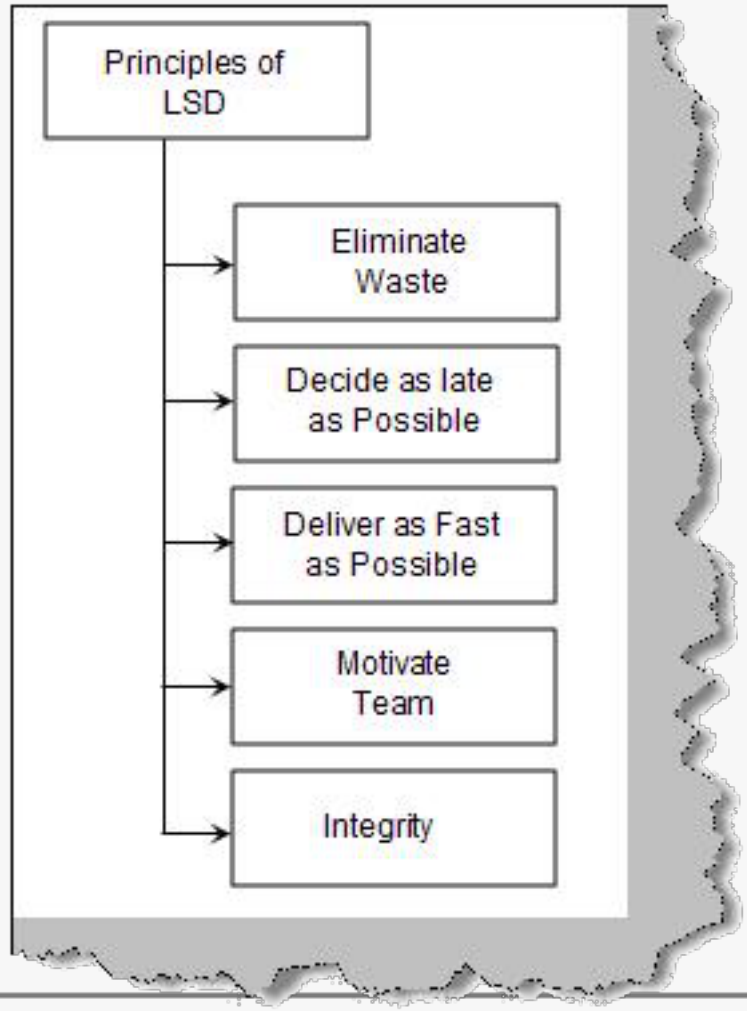
Desenvolvimento Ágil de Software | LSD

O Lean Software Development – LSD (Desenvolvimento de Software Enxuto) adaptou os princípios da fabricação enxuta da indústria para o mundo da engenharia de software.

Entre os princípios do desenvolvimento enxuto tem-se: eliminar desperdícios, decidir o mais tarde possível, entregar o mais rápido possível, motivar as pessoas e integridade.

Cada um desses princípios foi adaptado para o contexto do processo de software.

Por exemplo: "eliminar desperdício". No contexto de um projeto de software ágil é interpretado como não adicionar recursos ou funções obscuras, avaliar possíveis impactos do custo e cronograma de qualquer requisito que seja solicitado, eliminar etapas do processo que sejam supérfluos, etc.



Processo | Abordagens

Desenvolvimento Ágil de Software | AUP

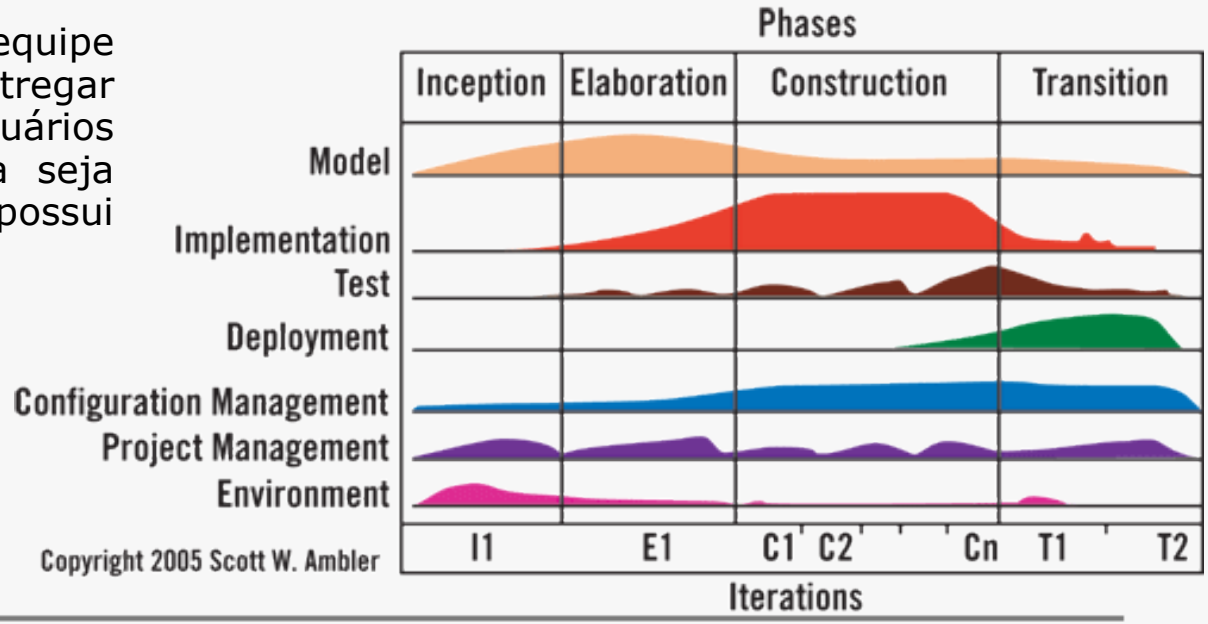
O Agile Unified Process – AUP (Processo Unificado Ágil) adota uma filosofia serial (sequência linear de atividades) para o que é amplo e iterativo.

Este processo possui atividades nas fases clássicas adotadas pelo Processo Unificado: Início, Elaboração, Construção e Transição.

Dentro de cada uma das atividades a equipe itera para alcançar a agilidade e entregar incrementos de software para os usuários finais. É importante que essa entrega seja mais rápida quanto possível. A AUP possui sete atividades:

- Modelagem
- Implementação
- Teste
- Desenvolvimento
- Gerenciamento de Configuração
- Gerenciamento de Projeto
- Ambiente

Portanto, se o leitor gostar do RUP, mas achar que ele é muito pesado e quiser implementar seus sistemas de uma forma mais leve, pode usar o AUP, que é, por assim dizer, uma versão simplificada do RUP.



Processo | Abordagens

Desenvolvimento Ágil de Software | Adaptativo versus Preditivo

Métodos Ágeis são algumas vezes caracterizados como o oposto de metodologias guiadas pelo planejamento ou disciplinadas. Uma distinção mais acurada é dizer que os métodos existem em um contínuo do adaptativo até o preditivo.

Métodos ágeis existem do lado adaptativo deste contínuo. Métodos adaptativos buscam a adaptação rápida a mudanças da realidade. Quando uma necessidade de um projeto muda, uma equipe adaptativa mudará também. Um time adaptativo terá dificuldade em descrever o que irá acontecer no futuro.

Uma equipe adaptativa pode relatar quais tarefas se iniciarão na próxima semana. Quando perguntado acerca de uma implantação que ocorrerá daqui a seis meses, uma equipe adaptativa deve ser capaz somente de relatar a instrução de missão para a implantação, ou uma expectativa de valor versus custo.

Métodos preditivos, em contraste, colocam o planejamento do futuro em detalhes. Uma equipe preditiva pode reportar exatamente quais aspectos e tarefas estão planejados para toda a linha do processo de desenvolvimento.

Elas porém tem dificuldades de mudar de direção. O plano é tipicamente otimizado para o objetivo original e mudanças de direção podem causar a perda de todo o trabalho e determinar que seja feito tudo novamente.

Equipes preditivas frequentemente instituem um comitê de controle de mudança para assegurar que somente as mudanças mais importantes sejam consideradas.

Não existe o certo ou errado. Existe a necessidade e a busca pela solução adequada a cada necessidade. Algumas empresas utilização um ou outro processo ou vários deles ao mesmo tempo.

Referências

PRESSMAN, Roger S. Engenharia de Software. São Paulo: Makron Books, 2011.
SOMMERVILLE, Ian. Engenharia de Software. 8. ed. São Paulo: Pearson Addison Wesley, 2007.
TONSIG, Sergio Luiz. Engenharia de Software: Análise e Projeto de Sistemas. 2. ed. Rio de Janeiro: Ciência Moderna, 2008.
Ken Schwaber e Jeff Sutherland. Scrum Guide. Disponível em http://www.scrum.org
BOEHM, B. Balancing Agility and Discipline: A Guide for the Perplexed. 2 ed. Boston,MA: Addison-Wesley, 2004
Manifesto for Agile Software Development. Disponível em http:www.agilemanifesto.org
MARTINS, J. C. C. (2007) Gerenciando Projetos de Desenvolvimento de Software com PMI, RUP e UML. 4 ed. Rio de Janeiro: Brasport.
KROLL, P. e Kruchten P. (2003) The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP. Addison Wesley.
BOENTE, A. N. P. (2003) Gerenciamento & Controle de Projetos. Rio de Janeiro: Axcel Books.
ROCHA, A. R. C.; Maldonado, J. C.; Weber, K. C. (2001) Qualidade de Software: Teoria e Prática. São Paulo: Prentice Hall.