Escola Superior de Tecnologia e Gestão
Politécnico do Porto

Author
Cristóvão Sousa
Carla Pereira

Version
V2

Metodologias de Desenvolvimento de SW

Metodologias Ágeis

Summary | *Agile Mindset*; Manifesto Agile; Princípios e valores; Metodologias XP e SCRUM

Intro

> "Agile is an iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner with "just enough" ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders"
>
> *(http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm*

Introdução

- Origem dos métodos ágeis: **Manifesto for Agile Software Development**, este originou um movimento na indústria de software conhecido como agile software development (Manifesto publicado em 2001)

- Em 2005, foi publicada uma adenda conhecida como **PM Declaration of Interdependence** – que é um conjunto de 6 princípios de gestão inicialmente planeados para projectos de desenvolvimento de software ágeis

  - Este nome foi alterado mais tarde para **The declaration of interdependence for modern management**, pois estes princípios poderiam ser aplicados a outras situações de gestão

- Valores nos quais se focam os métodos ágeis:
  - **Indivíduos** e **Interacções** em vez de processos e ferramentas
  - **Software funcional** em vez de documentação
  - **Colaboração do cliente** em vez de negociação de contratos
  - **Resposta rápida às alterações** em vez de seguir planos

Characteristics

- Modularidade
- Iteratividade
- Calendarizado com ciclos de vida de iteração de uma a seis semanas
- Economia no processo de desenvolvimento
- Abordagem convergente (incremental)
- “People-Oriented”
- trabalho colaborativo e comunicativo

Princípios 1/12

- A nossa prioridade mais alta é satisfazer o cliente através de entregas de software funcional cedo (rapidamente) e continuamente
  - Tendem a evitar desenvolvimento de requisitos e actividades do projecto infrutuosos
  - Procuram desenvolver software funcional tão rápido quanto possível
  - Vêem muito melhor projectos de prototipagem (com protótipos) mas com a diferença que o produto resultante é entregue para ser usado (ou seja, são protótipos funcionais)
  - A maioria dos métodos ágeis permitem que os clientes definam o que é mais valioso e atribuem ao cliente um papel importante no estabelecimento da ordem pela qual as funcionalidades são entregues

Princípios 2/12

- Alterações/mudanças nos requisitos são bem vindas, até mesmo tarde no desenvolvimento. Os processos ágeis aproveitam a mudança para vantagem competitiva do cliente
  - Este é o princípio principal no qual os métodos ágeis são construídos e a razão pela qual foi adoptado o nome “Ágil”
- **Defendem que as alterações são inevitáveis**
- **Encorajam a mudança** ao longo da vida do projecto

Princípios 3/12

- Entregas de software funcional frequentemente, de semanas a meses, preferencialmente na escala temporal mais curta
- Entregas contínuas
- É explícito que os incrementos num projecto ágil devem ser tão curtos quanto possível

Princípios 4/12

- Os “experts” do negócio e "developers" têm que trabalhar juntos diariamente ao longo do projecto
- Todos os métodos ágeis requerem **colaboração** entre a equipa de desenvolvimento e os outros *stakeholders* do projecto

Princípios 5/12

- Os projectos surgem através de indivíduos motivados. Dando-lhes o ambiente e apoio que eles precisam e deve existir uma relação de confiança

- Os métodos ágeis são todos construídos na suposição que os membros da equipa de desenvolvimento são todos competentes, motivados, apoiados pela organização, e capacitados para as suas tarefas

Princípios 6/12

- O método mais eficiente e eficaz de obter informação de e dentro de uma equipa de desenvolvimento é a conversação cara-a-cara

- Este é o princípio pelo qual os métodos ágeis tendem a ter poucos documentos escritos

- Todos eles valorizam a comunicação cara-a-cara como o meio principal de partilha de informação

Princípios 7/12

- Software funcional é a medida principal de progresso do projecto

- É uma outra forma pela qual os métodos ágeis fazem greve à documentação

Princípios 8/12

- Processos ágeis promovem desenvolvimento sustentável. Os sponsors (patrocinadores), "developers", e utilizadores devem ser capazes de manter um ritmo constante indefinidamente

Princípios 9/12

- Atenção contínua para a excelência técnica e bom projecto aumenta a agilidade
  - Este é talvez o princípio mais valioso dos métodos ágeis
  - Focam-se em assegurar que a qualidade de trabalho técnico é mantida a um nível alto
- As equipas de desenvolvimento tem a responsabilidade principal de assegurar que o que se entrega está correcto e que o software satisfaz as necessidades dos clientes (o que não significa que não seja necessários meios para validação e verificação independente)

Princípios 10/12

- Simplicidade é essencial

Princípios 11/12

- As melhores arquitecturas, requisitos e projectos surgem de equipas "self-organizing"
  - "Self-management "é um carimbo dos métodos ágeis. Contrariamente aos métodos de comando e controlo usados para gerir muitos projectos de software. Equipas "self-managed " são de facto muito eficazes
  - Este tipo de alterações são difíceis de terem efeitos e requerem que muitas pessoas adquiram novas competências e comportamento, mas podem ter resultados positivos significativos

Princípios 12/12

- Em intervalos regulares, a equipa reflecte em como se tornar mais eficaz e ajusta o seu comportamento adequadamente
- "**retrospective ou postmortem analysis**" é uma actividade importante adoptada pelos métodos ágeis como meio principal para melhorar os seus processos

Tópicos comuns

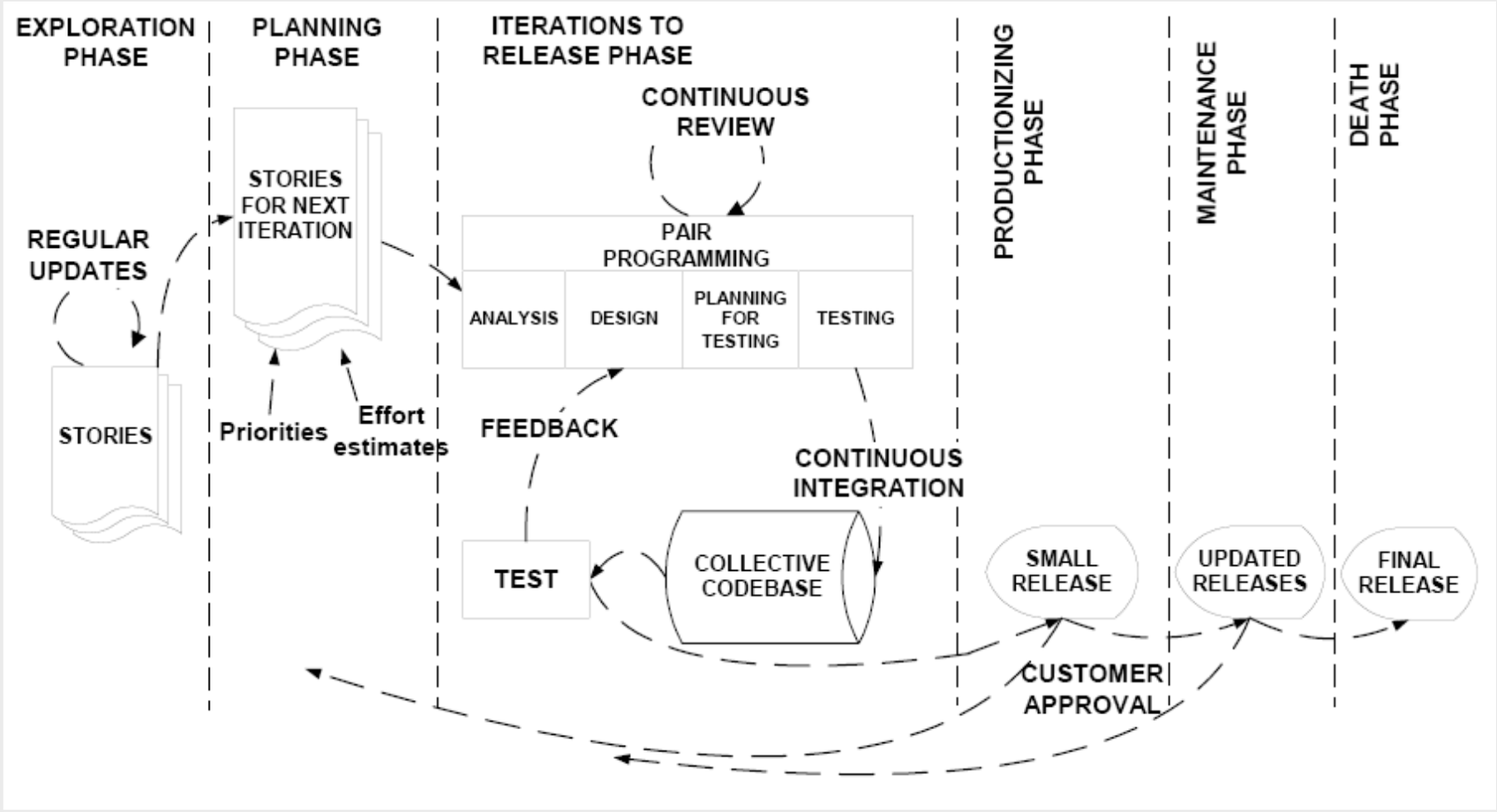
- Agilidade
- Alteração/mudança
- Planeamento
- Comunicação
- Aprendizagem

Principais metodologias

- **Extreme Programming (XP)**
- **Scrum**
- Dynamic Systems Development Method (DSDM)
- Adaptative Software Development (ASD)
- **Feature Driven Development (FDD)**
- Crystal Family of Methodologies
- RUP
- Open Source Software Development
- ...

Extreme Programming (XP)

- O ciclo de vida tem 5 fases (exploração, planeamento, iterações e lançamento, productionizing - SW em produção), manutenção e término)



Extreme Programming (XP)

- Exploração:
  - Os clientes escrevem em cartões o que desejam ver incluído na primeira versão
  - Cada cartão (story card) descreve uma característica a ser adicionada ao programa
  - Ao mesmo tempo a equipa de projecto familiariza-se com as ferramentas, tecnologias e práticas que serão usadas no projecto
  - A tecnologia a ser usada é testada e as possibilidades de arquitectura para o sistema são exploradas construindo um protótipo do sistema
  - Duração: entre algumas semanas e alguns meses, dependendo muito do conhecimento que os programadores tem da tecnologia a usar

Extreme Programming (XP): Story Card

Customer Story and Task Card — B/W Development \ COLA

DATE: 3/19/98 — TYPE OF ACTIVITY: NEW: X FIX: ___ ENHANCE: ___ FUNC. TEST

STORY NUMBER: 1275 — PRIORITY: USER: ___ TECH: ___

PRIOR REFERENCE: ___ — RISK: ___ TECH ESTIMATE: ___

TASK DESCRIPTION:
SPLIT COLA: When the COLA rate chgs. in the middle of the B/W Pay Period you will want to pay the 1ST week of the pay period at the OLD COLA rate and the 2ND week of the Pay Period at the NEW COLA rate. Should occur automatically based on system design.

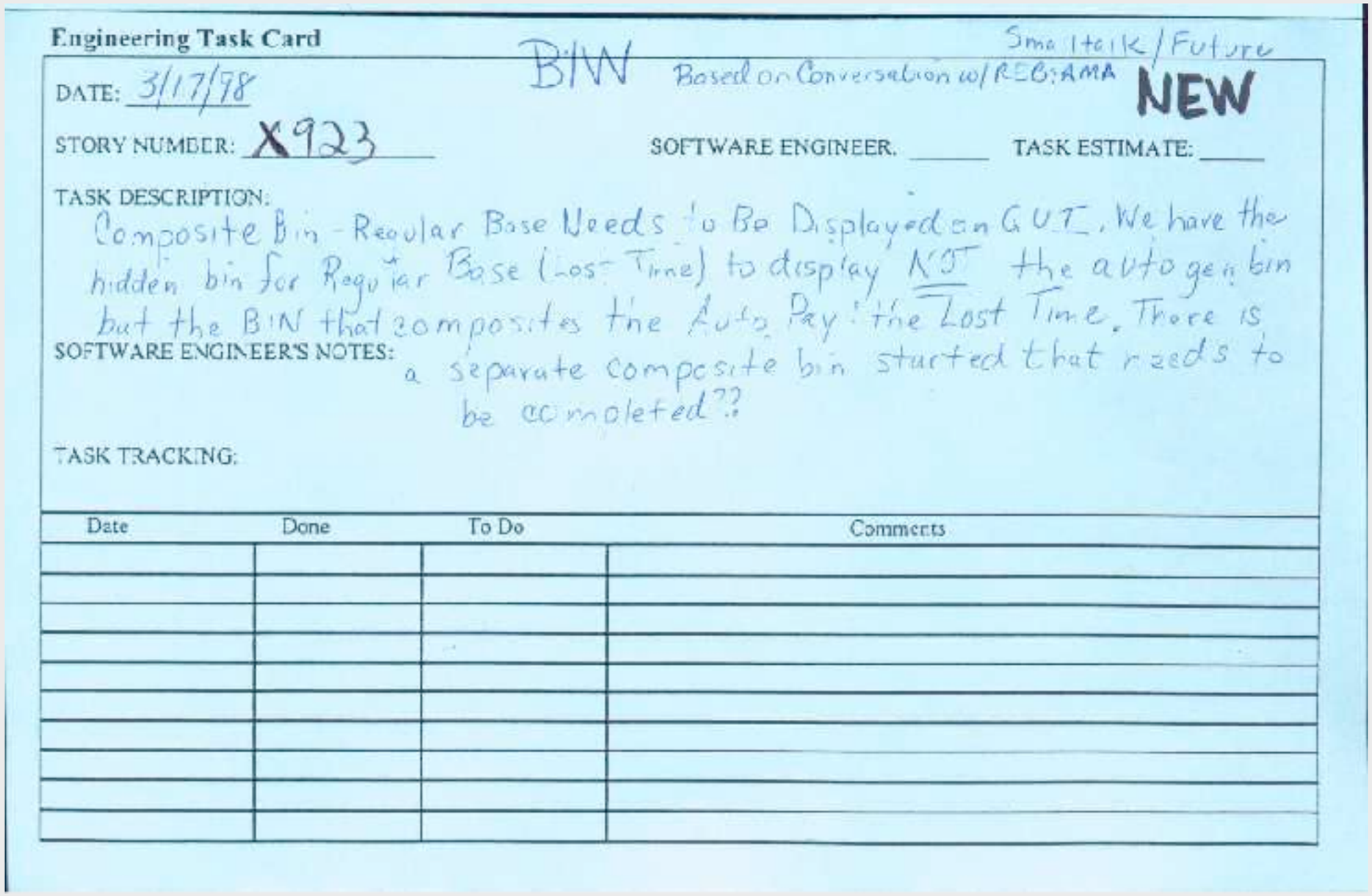
NOTES:
For the OT, we will run a m/frame program that will pay or calc the COLA on the 2ND week of OT. The plant currently retransmits the hours data for the 2ND week exclusively so that we can calc COLA. This will come into the Model as a "2.44" COLA.

TASK TRACKING: Gross Pay Adjustment. Create RM Boundary and Place in DEE=Excess COLA BIN

Date	Status	To Do	Comments

http://xprogramming.com/xpmag/story_and_task_cards

Extreme Programming (XP): Task Card



Engineering Task Card

DATE: 3/17/78

B/W Based on Conversation w/REBECCA

Small talk / Future NEW

STORY NUMBER: X923

SOFTWARE ENGINEER. _____ TASK ESTIMATE: _____

TASK DESCRIPTION:

Composite Bin - Regular Base Needs to Be Displayed on GUI. We have the hidden bin for Regular Base (Lost Time) to display NOT the auto gen bin but the BIN that composites the Auto Pay + the Lost Time. There is a separate composite bin started that needs to be completed??

SOFTWARE ENGINEER'S NOTES:

TASK TRACKING:

Date	Done	To Do	Comments

http://xprogramming.com/xpmag/story_and_task_cards

Extreme Programming (XP): Cards

http://blog.livingroomarchitect.com/2008/08/story-card-matrix.html

Extreme Programming (XP)

- Planeamento:
  - Fixa a ordem de prioridade para as story cards e um acordo dos conteúdos da primeira pequena versão
  - Os programadores primeiro estimam o esforço que cada story card requer e o escalonamento é acordado
  - O tempo para apresentação da primeira versão não excede normalmente 2 meses. A própria fase de planeamento leva alguns dias

Extreme Programming (XP)

- Iterations to release:
  - Inclui várias iterações do sistema antes da primeira versão
  - A primeira iteração cria um sistema com a arquitectura de todo o sistema, isto é alcançado seleccionando as story cards que influenciam a construção da estrutura do sistema
  - O cliente decide que story cadrs são seleccionados para cada iteração
  - Os testes funcionais criados pelos clientes são realizados no final de cada iteração
  - No fim da última iteração o sistema está pronto para produção

Extreme Programming (XP)

- Productionizing ("Software em produção"):
  - **Requer testes extras** e análise do desempenho do sistema antes do sistema poder ser entregue ao cliente
  - Novas alterações podem ser encontradas e é decidido se estas serão incluídas na release actual
  - Durante esta fase as iterações podem precisar de ser aceleradas de 3 semanas a 1 semana
  - As ideias e sugestões são documentadas para implementação posterior, durante por exemplo a fase de manutenção

Extreme Programming (XP)

- Manutenção:
  - Depois da primeira entrega do produto ao cliente, deve manter-se o sistema em produção enquanto também se produzem novas iterações
  - A fase de manutenção requer um esforço para tarefas de suporte ao cliente
  - A velocidade de desenvolvimento poderá diminuir depois do sistema estar em produção
  - Pode requer novas pessoas para a equipa e alterações na estrutura da equipa

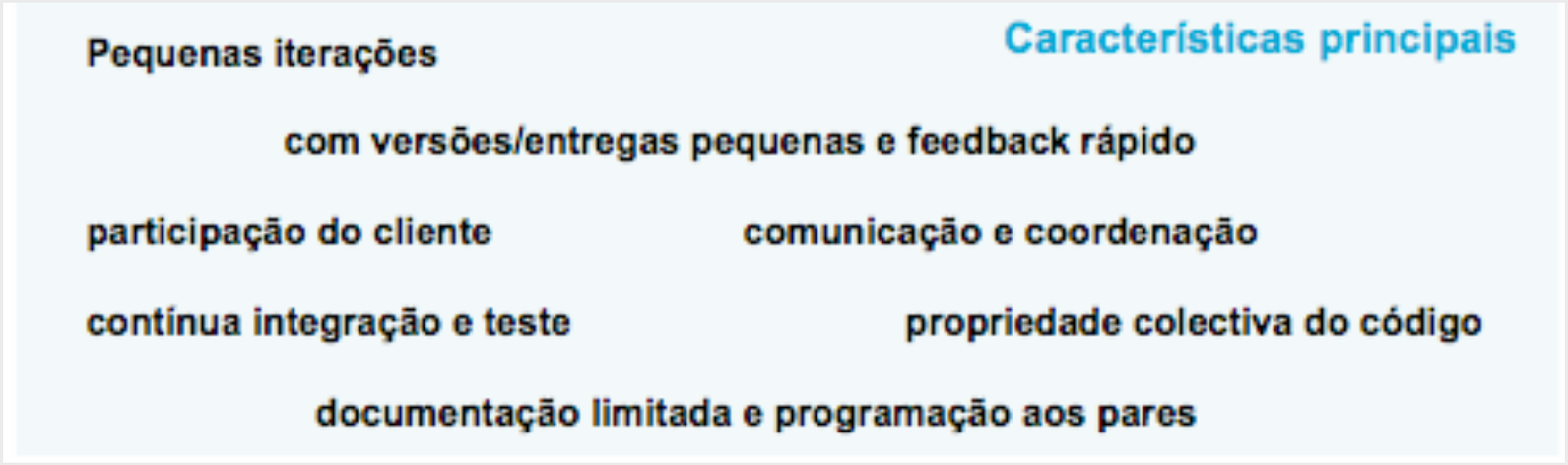
Extreme Programming (XP): *Roles*

- **Programmer**: escreve testes e mantém o código do programa tão simples e definido quanto possível
- **Customer**: escreve as story cards e testes funcionais, e decide quando cada requisito é satisfeito. Os clientes definem as prioridades de implementação para os requisitos
- **Tester**: ajuda os clientes a escrever testes funcionais. Executa testes funcionais regularmente, ...
- **Tracker**: avalia o processo, dá feedback no XP. Localiza as estimativas feitas pela equipa (por exemplo, esforço estimado) e dá feedback no sentido de melhorar estimativas futuras. Analisa o progresso de cada iteração e avalia se o objectivo é alcançável com os recursos e restrições de tempo ou se qualquer alteração é necessária no processo
- **Coach**: responsável pelo processo como um todo. Guia os outros membros da equipa durante o processo
- **Consultant**: é um membro externo que possui o conhecimento técnico específico necessário. Guia a equipa na resolução dos seus problemas específicos.
- **Manager (big Boss)**: toma as decisões. Para isto, comunica com a equipa de projecto para determinar a situação actual e distinguir qualquer dificuldades ou deficiências no processo

Extreme Programming (XP): Práticas

- Os **clientes tomam as decisões** de negócio enquanto os **programadores decidem** os **aspectos técnicos**.

- O XP permite o desenvolvimento de software de sucesso, apesar dos vagos requisitos e alterações constantes nos requisitos, com equipas de tamanho pequeno a médio.



Extreme Programming (XP): Práticas

- **Planning game** - interacção próxima entre clientes e programadores. os programadores estimam o esforço necessário para a implementação das stories dos clientes e os clientes então decidem sobre o âmbito e tempo das releases

- **Small/short releases** - um sistema simples é produzido rapidamente - pelo menos uma vez em cada 2 a 3 meses. Novas versões são pelo menos entregues mensalmente

- **Metaphor** - o sistema é definido por uma metáfora/conjunto de metáforas entre os clientes e os programadores. Esta "shared story" guia todo o desenvolvimento descrevendo como o sistema trabalha

- **Simple design** - o foco é em projectar uma solução o mais simplista possível que seja implementável no momento. Código extra e complexidade desnecessária são removidos imediatamente

Extreme Programming (XP): Práticas

- **Testing** - são feitos testes continuamente, os clientes escrevem os testes funcionais
- **Refactoring** - restruturação do sistema removendo duplicação, melhorando a comunicação, simplificando e adicionando flexibilidade
- **Pair Programming** - duas pessoas escrevem o código num computador
- **Collective ownership** - qualquer um pode alterar qualquer parte do código a qualquer altura
- **Continuous integration** - um pedaço de código novo é integrado no código base assim que esteja pronto, assim o sistema é integrado e construído muitas vezes por dia
- **40-hour week** - no máximo a semana de trabalho tem 40 horas. Se tiver mais é tratado como um problema a ser resolvido

Extreme Programming (XP): Práticas

- **On-site costumer** -o cliente tem que estar disponível a tempo inteiro para a equipa

- **Coding standards** - existem regras de codificação, são seguidas pelos programadores

- **Open Workspace** - uma sala grande com cubículos pequenos é preferida. Os programadores devem estar colocados no centro da sala

- **Just rules** - a equipa tem as suas regras, que são seguidas, mas a qualquer altura podem ser alteradas. As alterações devem ser acordadas e o seu impacto avaliado

Extreme Programming (XP): Âmbito

- Definida para equipas de tamanho pequeno a médio - limitado entre 3 e um máximo de 20 membros por projecto

- Comunicação e coordenação entre membros do projecto deve ser possível a qualquer momento

- Cultura organizacional

- Aspectos tecnológicos – tecnologias que não permitem alterações ou que exigem grandes períodos de feedback não são apropriadas para o XP.

SCRUM

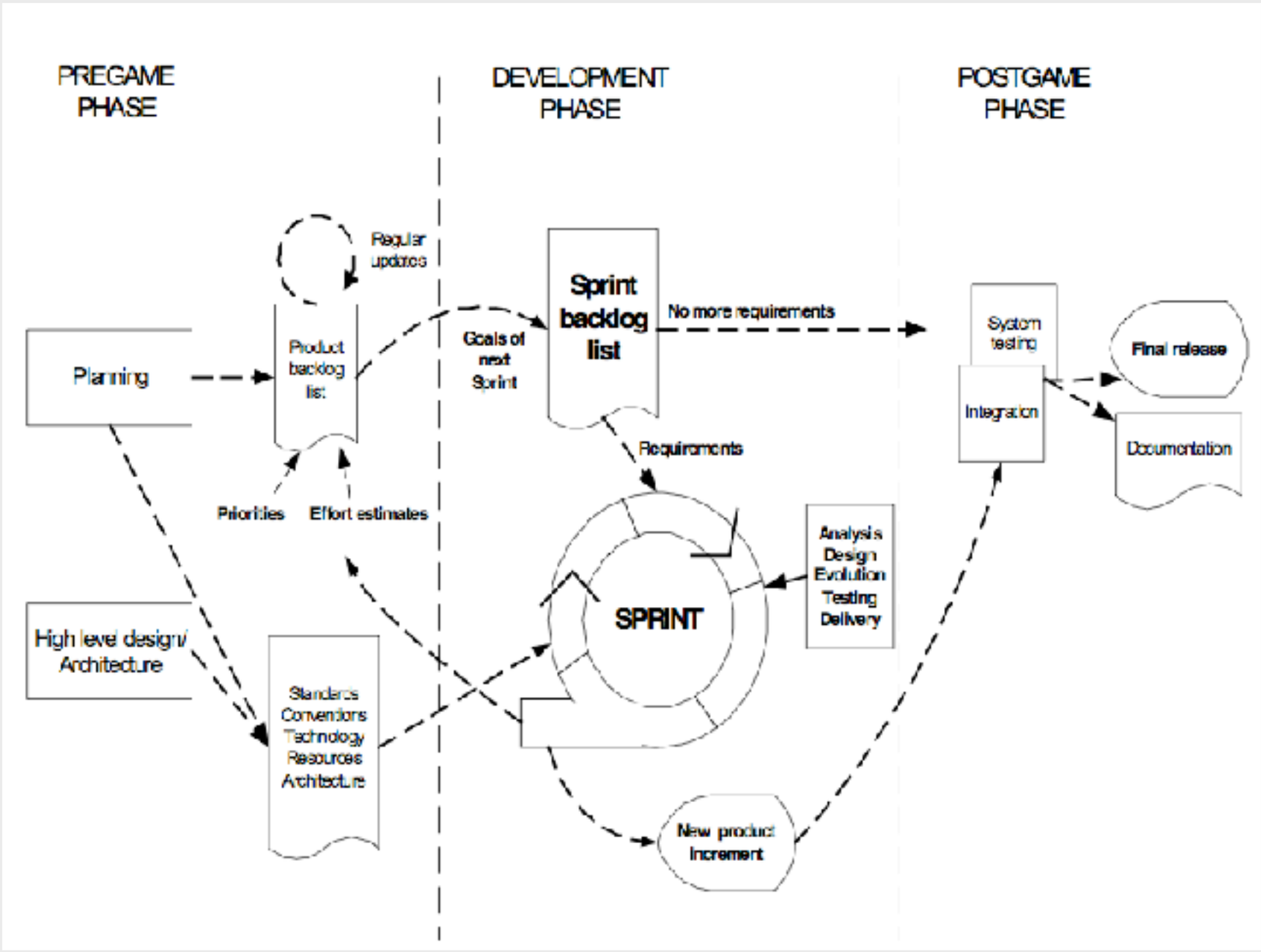
SCRUM

- Desenvolvida para gestão do processo de desenvolvimento de sistemas
- Abordagem empírica que aplica ideias da teoria de controlo de processos industrias ao desenvolvimento de sistemas
  - resultando numa abordagem que re-introduz a ideia de flexibilidade, adaptabilidade e produtividade
- Não define nenhumas técnicas específicas de desenvolvimento de software para a fase de implementação
- Concentra-se na forma como os membros da equipa devem funcionar para produzir um sistema flexível num ambiente constantemente variável

SCRUM

- A ideia principal é que o desenvolvimento do sistema envolve várias variáveis ambientais e técnicas que estão provavelmente a alterar-se durante o processo
  - O que torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para ser capaz de responder às alterações
- Ajuda a melhorar as práticas de engenharia existentes numa organização
  - para isto envolve actividades de gestão frequentes visando consistentemente a identificação de qualquer deficiências ou impedimentos no processo de desenvolvimento bem como das práticas que são usadas

SCRUM: O Processo



SCRUM: O Processo

- **“Pre-Game” - Inclui 2 sub-fases: Planeamento e Arquitectura/Projecto de alto nível**
  - **Planeamento**
    - Inclui a definição do sistema a ser desenvolvido
    - É criada uma Product Backlog list contendo todos os requisitos que são actualmente conhecidos
    - **Os requisitos são priorizados e o esforço necessário para a sua implementação é estimado**
    - A Product backlog list é constantemente actualizada com novos ou mais detalhados itens, bem como com estimativas mais precisas e novas ordens de prioridades
    - Definição da equipa de projecto, ferramentas e outros recursos
    - Avaliação do risco e aspectos de controlo, necessidades de formação e aprovação da gestão de verificação
    - Em todas as iterações, o produto actualizado é revisto pela equipa Scrum para definir o compromisso para a próxima iteração

SCRUM: O Processo

- **“Pre-Game” - Arquitectura/Projecto de alto nível**
  - O projecto de alto nível do sistema inclui o planeamento da arquitectura com os itens actuais na Product BacKlog
  - No caso de melhorias num sistema existente, as alterações necessárias para implementar os Backlog itens são identificadas junto com os problemas que podem causar
  - Uma reunião de revisão do projecto é realizada para tratar das propostas para a implementação e são tomadas decisões com base nesta revisão
  - Em adição, planos preliminares para os conteúdos das releases são preparados

SCRUM: O Processo

- **“Development or Game Phase”**
  - É a parte ágil da abordagem Scrum
  - Esta fase é tratada como uma "black box" onde o imprevisível é esperado
  - As diferentes variáveis técnicas e ambientais - tais como prazo, qualidade, requisitos, recursos, tecnologias de implementação e ferramentas e métodos de desenvolvimento
    - Podem variar durante o processo
    - São observadas e controladas através de várias práticas durante os Sprints da fase de desenvolvimento
  - Scrum visa este controlo constantemente - não apenas no início do projecto de desenvolvimento de software - de forma a ser capaz de se adaptar com flexibilidade às alterações
  - Esta fase é desenvolvida em Sprints - são ciclos iterativos onde a funcionalidade é desenvolvida ou melhorada produzindo novos incrementos

SCRUM: O Processo

- **“Development or Game Phase”**
  - Cada Sprint inclui as fases tradicionais do desenvolvimento de software: análise, requisitos, projecto, evolução e entrega
  - A arquitectura e projecto do sistema evoluem durante o desenvolvimento do Sprint
  - Um Sprint é planeado para durar entre uma semana a um mês
  - Podem existir, p.e., 3 a 8 Sprints num processo de desenvolvimento de sistemas antes do sistema estar pronto para distribuição
  - Pode haver mais do que uma equipa a construir o incremento

SCRUM: O Processo

- **“Post-Game Phase”**
  - Contém o encerramento das entregas
    - O sistema está pronto para ser entregue, os requisitos estão concluídos, não há mais itens para acrescentar
    - Inclui tarefas tais como integração, testes do sistema e documentação

SCRUM: *Roles*

- **Scrum Master** - é um novo papel de gestão introduzido pela Scrum
  - É responsável por assegurar que o projecto é executado de acordo com as práticas, valores e regras do Scrum e que progridem como planeado.
  - Interage com a equipa de projecto, bem como com o cliente e gestão durante o projecto
  - É responsável por assegurar que qualquer impedimento é removido e alterado no processo para manter a equipa a trabalhar tão produtivamente quanto possível
- **Product Owner** - é oficialmente responsável pelo projecto, gestão, controlo e tornar visível o Product Backlog list
  - É seleccionado pelo Scrum Master, cliente e gestão
  - Toma as decisões finais das tarefas relacionadas com o product Backlog
  - Participa na estimativa dos esforços de desenvolvimento para os backlog itens e torna os aspectos da Backlog em características a serem desenvolvidas

SCRUM: *Roles*

- **Scrum Team** - é a equipa de projecto que tem a autoridade para decidir as acções necessárias e organizar-se no sentido de alcançar os objectivos de cada Sprint.
  - Está envolvida, por exemplo, na estimativa de esforços, criação do Sprint Backlog, revisão do Product Backlog list e sugerir restrições que necessitam de ser removidas do projecto
- **Costumer** - O cliente participa nas tarefas relacionadas com product Backlog itens para o sistema a ser desenvolvido ou melhorado.
- **Management** - Participa na definição de objectivos e requisitos.
  - Por exemplo, a gestão está envolvida na selecção do Product Owner, medição do progresso e redução do backlog com o Scrum Master.

SCRUM: *Práticas/Artefactos*

- **Product Backlog**
  - Define tudo que é necessário no produto final baseado no conhecimento actual
  - Define o trabalho a ser feito no projecto
  - Inclui uma lista do negócio e dos requisitos técnicos para o sistema a ser construído ou melhorado priorizada e constantemente actualizada
  - Múltiplos actores podem participar na geração dos Product Backlog itens
    - clientes, equipa de projecto, marketing e vendas, gestão e apoio ao cliente
- **Effort Estimation**
  - É um processo iterativo
  - À medida que mais informação é disponibilizada dos itens do Product Baclog deixa de ser necessária
  - O Product Owner com a equipa(s) Scrum são responsáveis por realizar as estimativas de esforços

SCRUM: Priorizar e Estimar

- É difícil estimar! As estimativas são sempre uma “mentira”, resta-nos tentar “mentir” o menos possível;

- Uma forma de reduzir o erro das estimativas é fazê-las em equipa. Por um lado reduz-se o risco das estimativas dadas, e a troca de ideias que se gera durante a estimação ajuda a equipa a perceber melhor as tarefas;

- O “**planning pocker**” é uma boa técnica para estimar em conjunto;

- As estimativas permitem-nos calcular uma “velocidade”, que é útil para prever a quantidade de trabalho da estimativa seguinte, mas que não é uma boa medida da produtividade da equipa.

Planning Poker: Como?

- Selecionar as User Stories ou Tarefas onde exist a necessidade de estimar esforço

- A equipa discute uma User Story (ativa) ou tarefa e todos selecionam uma carta

- A determinada altura após a breve discussão cada um mostra a sua carta

- Uma média é calculada em função dos calores de cada carta. O valor da média é aceite e segue-se para a User story seguinte, ou procedee-se a uma nova jogada. Existe ainda a possibilidade dos elementos mais experientes ajustarem o valor da média e seguir-se para a “jogada” seguinte.

Planning Poker: Cartas

- Considere-se um conjunto de cartas, numeradas. A forma de numerar as cartas pode variar:
  - Sequência de Fibonacci: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
  - Alguns SW usam a série: 0, ½, 1, 2, 3, 5, 8, 13, 21, 34, 100, e ? (dúvida)
  - Outros, exemplo T-SHIRT: XS, S, M, L, XL, XXL

Planning Poker

- Considere-se a seguinte sequência: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, em que a interpretação poderá ser:
  - 1 - “considere-se feito”
  - 2- *Piece of Cake*
  - *3 - It ain’t rocket science*
  - *5* - ornitorrinco
  - 8 - Começa a ser duro para uma pessoa só
  - 13 - Demasiado grande e necessário dividir, mas posso arriscar!
  - 20 - Definitivamente, esta tarefa necessita de ser sub dividida
  - 40 - Vou meter-me numa “alhada”
  - 100 - Tarefa Monstruosa

SCRUM: *Práticas/Artefactos*

- **Sprint**
  - É o procedimento de adaptação as alterações ambientais variáveis (requisitos, tempo, recursos, conhecimento, tecnologia, etc)
  - A equipa Scrum organiza-se para produzir um novo incremento no produto executável num Sprint que dura aproximadamente 30 dias
  - As ferramentas de trabalho da equipa são Sprint Planning Meetings, Sprint Backlog e Daily Scrum Meetings
- **Sprint Planning Meeting**
  - É uma reunião com 2 partes organizada pelo Scrum Master
    - Os clientes, utilizadores, gestão, Product Owner e Scrum Team participam na primeira parte da reunião para decidir os objectivos e funcionalidades do próximo Sprint.
    - A segunda parte da reunião é assegurada pelo Scrum Master e a Scrum Team focando-se em como o incremento do produto é implementado durante o Sprint.

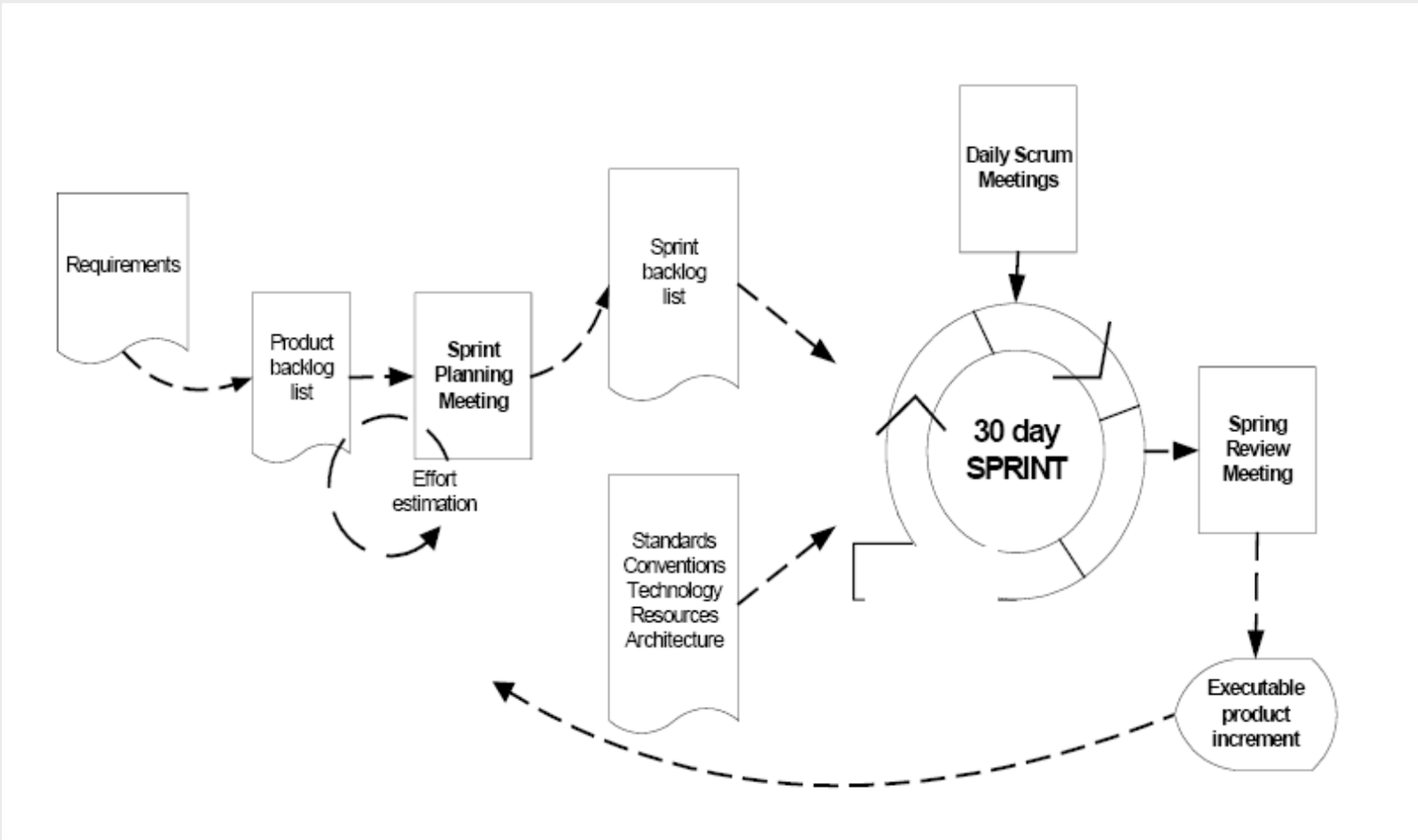
SCRUM: *Práticas/Artefactos*

- **Sprint Backlog**
  - É o ponto de partida de cada Sprint
  - É uma lista de Product Backlog itens seleccionados para serem implementados no próximo Sprint
  - Quando todos os itens no Sprint Backlog estão concluídos, uma nova iteração do sistema é entregue
- **Daily Scrum Meeting**
  - Para acompanhar o progresso continuamente e também servem como planeamento de reuniões: o que foi feito depois da última reunião e o que será feito antes da próxima.
  - São discutidos problemas e outros assuntos variáveis e são controlados, nesta reunião diária de aprox. 15 minutos
  - Qualquer deficiências ou restrições no processo de desenvolvimento do sistema são procuradas, identificadas e removidas para melhorar o processo
  - O Scrum Master conduz as Scrum meetings
  - Além da Scrum team, também a gestão, por exemplo, pode participar na reunião

SCRUM: *Práticas/Artefactos*

- **Sprint Review Meeting**
  - No último dia do Sprint, a Scrum Team e o Scrum Master apresentam os resultados - o incremento do produto - do Sprint à gestão, clientes, utilizadores e ao Product Owner numa reunião informal
  - Os participantes avaliam o incremento e tomam a decisão sobre as actividades seguintes
  - A reunião de revisão pode originar novos Backlog Itens e até mesmo mudar a direcção do sistema que está a ser construído

SCRUM: *Práticas/Artefactos*



O PO de fine a **visão** com base nas informações obtidas junto dos utilizadores, da equipa, stakeholders e gestores.

O PO e o Srum Master criam o **Product Backlog** - lista inicial de necessidades que necessitam de ser produzidas para que a visão do projecto seja bem sucedida

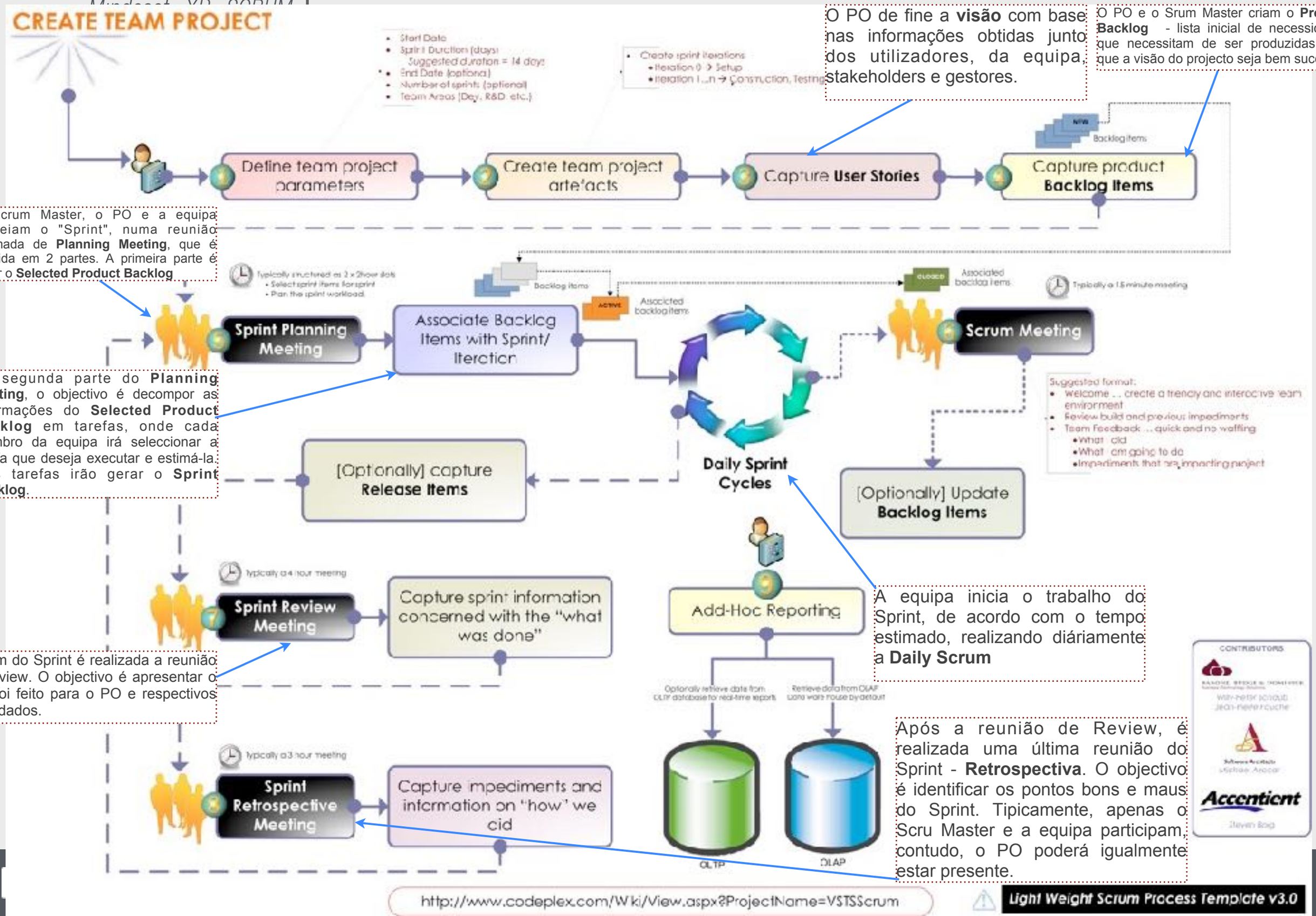
O Scrum Master, o PO e a equipa planeiam o "Sprint", numa reunião chamada de **Planning Meeting**, que é dividida em 2 partes. A primeira parte é gerar o **Selected Product Backlog**

Na segunda parte do **Planning Meeting**, o objectivo é decompor as informações do **Selected Product Backlog** em tarefas, onde cada membro da equipa irá seleccionar a tarefa que deseja executar e estimá-la. Tais tarefas irão gerar o **Sprint Backlog**.

A equipa inicia o trabalho do Sprint, de acordo com o tempo estimado, realizando diáriamente a **Daily Scrum**

No fim do Sprint é realizada a reunião de review. O objectivo é apresentar o que foi feito para o PO e respectivos convidados.

Após a reunião de Review, é realizada uma última reunião do Sprint - **Retrospectiva**. O objectivo é identificar os pontos bons e maus do Sprint. Tipicamente, apenas o Scru Master e a equipa participam, contudo, o PO poderá igualmente estar presente.



Scrum in a Nutshell

PO
Product Owner

PBI 1
PBI 2
PBI 3
PBI 4
PBI 5
PBI 6
product backlog

sprint planning

The Sprint
1 - 4 Weeks

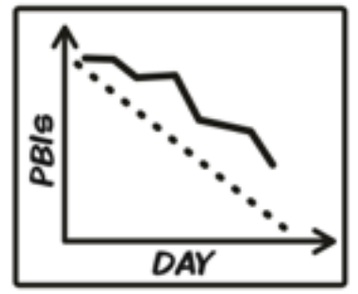
PBI	TASK TO DO	
1	1a	1b
2	2a	2b
3	3a	3b

sprint backlog

daily scrum
15min.

PBI	TO DO	DOING	DONE
1		1a	1b
2		2a 2b	
3	3a 3b		

visible progress



STAKEHOLDERS

PBI 4
PBI 5
PBI 6

PBI 5.1
PBI 4.2
PBI 4.1
PBI 6.1
PBI 5.2
PBI 6.3
PBI 6.2

backlog refinement

retrospective

PO

STAKEHOLDERS
sprint review

Development Team

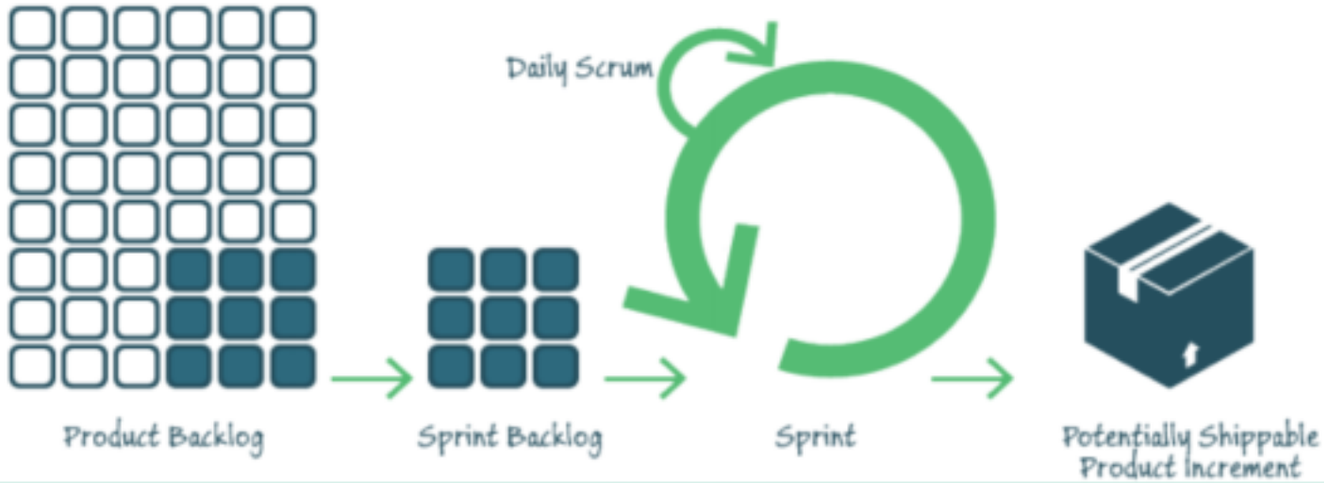
ScrumMaster

product increment

QUALITY
Definition of Done

the beginner's Scrum CHEAT SHEET

beLithe



ROLES

Develops product vision
Owns product backlog
Prioritizes backlog items

Product Owner

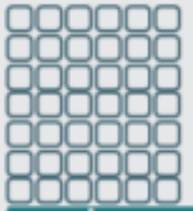
Process coach
Removes impediments
Facilitates team meetings

Scrum Master

Cross-functional
Self-organizing
Completes all the work

The Team

ARTIFACTS



Lists all work on a product
Never complete, always changing
Higher order = higher priority

Product Backlog

Lists work to be done in the current sprint
Pulled from product backlog
Items are broken into tasks

Sprint Backlog

Shows amount of work remaining/complete
Provides visual of current status
Shows likelihood of on-time completion

Burndown Chart

ACTIVITIES

Plan created for what is to be delivered in the upcoming sprint

Sprint Planning

Consistent iteration of time where the team completes work

Sprint

Daily 15 minute meeting where team mates report: 1. what did you do yesterday, 2. what are you doing today, 3. list any impediments

Daily Scrum

Team shows off completed work to the Product Owner

Sprint Review

Team discusses how the sprint went (except Product Owner)

Sprint Retro

- http://srummy.com
- SrumPad (Trial)
- ScrumWorks
- SkinnyBoard
- Agilo
- ...

- WhiteBoard