Engenharia de Software

Processos de Desenvolvimento Ágil

Luís Morgado

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Desenvolvimento Ágil de Software

- O desenvolvimento ágil de software é uma abordagem que tem por objectivo uma **maior flexibilidade e capacidade de adaptação à mudança**, em contraste com processos mais formais e, por isso, menos flexíveis
- É uma forma de **desenvolvimento cíclico**, mas caracterizado por ciclos de curta duração (iterações), com versões executáveis disponibilizadas em cada iteração
- Surgiu na década de 1990, como **alternativa aos métodos de maior formalidade** que prevaleciam nesse período, propondo abordagens menos formais e mais adaptativas, como é o caso dos métodos de *desenvolvimento rápido de aplicações* (RAD) e o método *extreme programming* (XP)
- Em 2001 um conjunto de profissionais de desenvolvimento de software, como é o caso de Kent Beck (processo XP) ou Jeff Sutherland (processo *Scrum*), publicaram o que ficou conhecido como ***manifesto ágil*** de desenvolvimento de software

Desenvolvimento Ágil de Software

- O desenvolvimento ágil de software, tal como proposto no *manifesto ágil* de desenvolvimento de software (Kent Beck, 2001), é baseado em quatro valores principais, no qual se valoriza mais:
  - **Indivíduos e interacções** em vez de **processos e ferramentas**
  - **Software funcional** em vez de **documentação exaustiva**
  - **Colaboração com o cliente** em vez da **negociação de contratos**
  - **Responder à mudança** em vez de **seguir um plano**
- No *manifesto ágil* de desenvolvimento de software, apesar dos autores reconhecerem o valor da adequada definição e utilização de processos e ferramentas, ou da importância da documentação, da negociação de contratos e do planeamento, valorizam mais o foco nos indivíduos e respectiva comunicação, em software funcional, na colaboração com o cliente e na capacidade de adaptação à mudança de forma flexível e ágil

Desenvolvimento Ágil de Software

Princípios considerados base numa metodologia ágil:

1. **A prioridade máxima é satisfazer o cliente** através da entrega antecipada e contínua de software de valor
2. **Acolher a mudança dos requisitos**, mesmo em etapas tardias do desenvolvimento
3. **Entregar software funcional com frequência**, de algumas semanas a alguns meses, com preferência pelos prazos mais curtos
4. **Colaboração próxima e frequente entre clientes e equipa de desenvolvimento**
5. **Construir projectos em torno de indivíduos motivados**
6. **Conversação cara-a-cara** é a melhor forma de comunicação
7. **Software funcional** é a principal medida de progresso
8. **Desenvolvimento sustentado**, capaz de manter um ritmo constante
9. **Atenção contínua à excelência técnica** e à boa concepção do software
10. **Simplicidade** - a arte de maximizar a quantidade de trabalho não efectuado - é essencial
11. **Equipas auto-organizadas** produzem as melhores arquitecturas, requisitos e soluções
12. **Introspeção** - regularmente, as equipas de desenvolvimento reflectem sobre a forma de se tornar mais eficazes e ajustam-se em conformidade

[Kent Beck, 2001]

Desenvolvimento Ágil

Agilidade significa eficiência e capacidade de lidar com a mudança

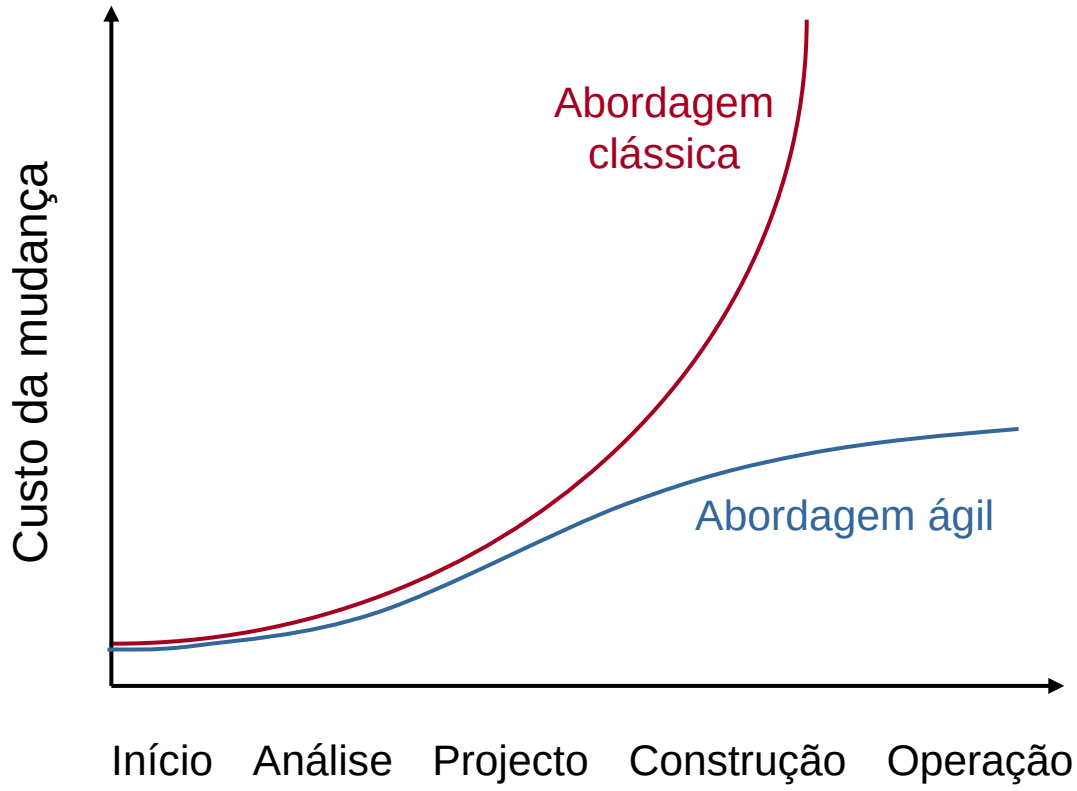
Não cair no erro de assumir que agilidade significa ausência de disciplina ou desenvolvimento informal

É necessário um processo organizado e sistemático!

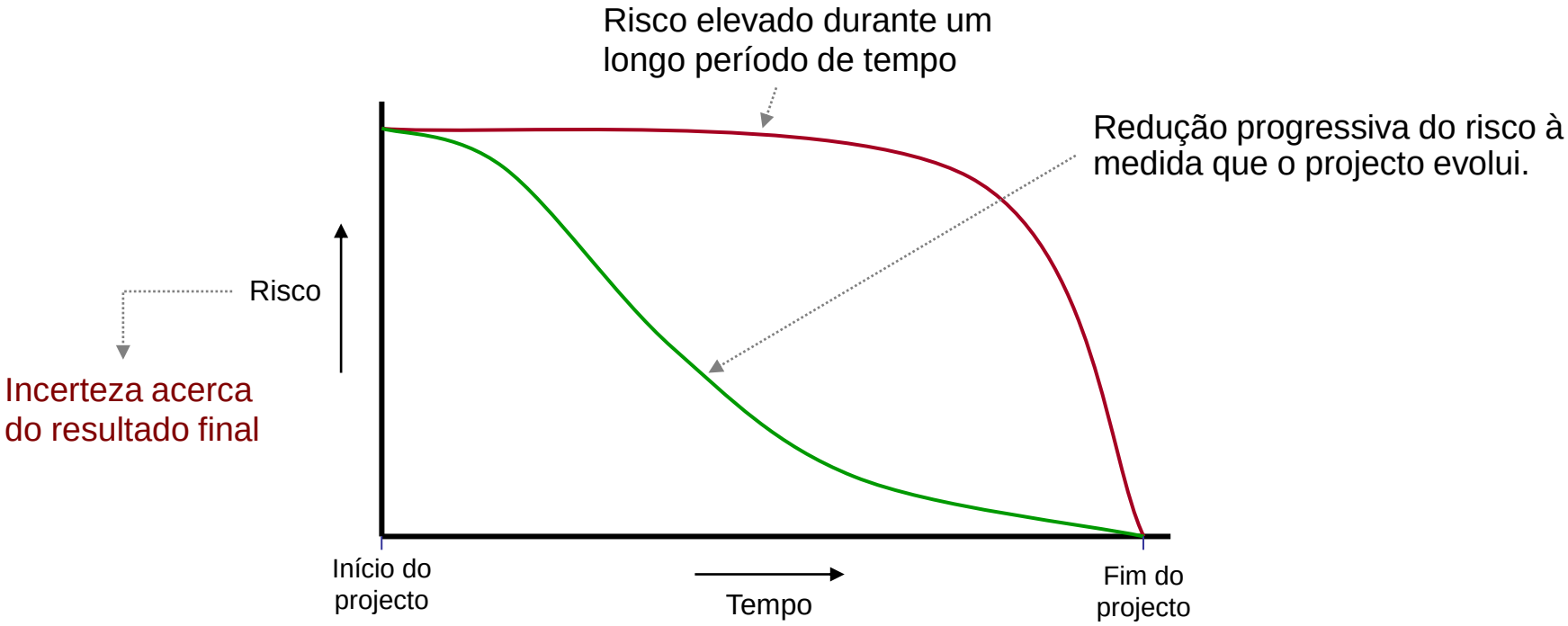
Processos de Desenvolvimento Ágil

Mudança como parte integrante do processo

Rapidez e eficácia do desenvolvimento



Redução do risco de desenvolvimento



Redução de incerteza ← **Aumento do conhecimento**

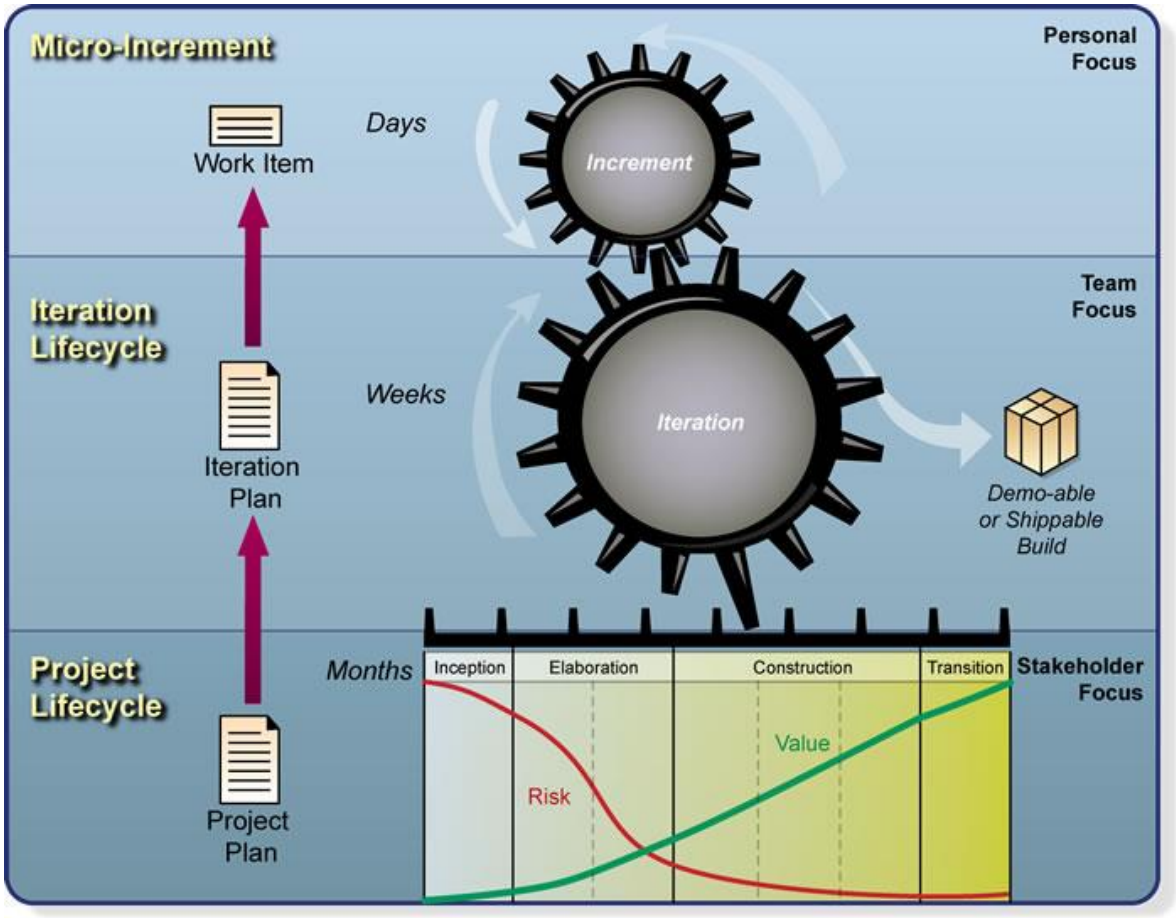
Metodologia de Desenvolvimento Ágil

- Numa metodologia ágil, o desenvolvimento é organizado em **incrementos de pequena dimensão** (*versões* ou *entregas*) no sentido de minimizar a incerteza do planeamento e projecto desses incrementos
- O desenvolvimento desses incrementos é organizado em **iterações curtas** (normalmente de uma a quatro semanas)
- Cada iteração envolve uma **equipa multifuncional** que trabalha em todas as actividades (planeamento, análise, concepção, construção, teste)
- No final de cada iteração, **uma versão funcional é demonstrada** às partes interessadas (cliente, utilizadores)
  - **Tem por objectivo a minimização da incerteza e do risco associado,** através da obtenção de informação (*feedback*) das partes interessadas no sistema, permitindo a detecção precoce de erros ou problemas e a adaptação rápida do produto em desenvolvimento a alterações necessárias
- O desenvolvimento de um produto é organizado em **múltiplas iterações**
- **Software funcional** é a principal medida de aferição do progresso de um projecto

Metodologia de Desenvolvimento

No âmbito do desenvolvimento ágil o desenvolvimento de um projecto (*ciclo de de vida do projecto*) é organizado em iterações, com duração no âmbito de semanas, as quais produzem versões incrementais executáveis e demonstráveis, com o objectivo de reduzir o risco de desenvolvimento e maximizar o valor para as partes interessadas (*stakeholders*)

As iterações organizam-se em micro-incrementos que consistem na realização dos itens de trabalho referentes à concretização dos requisitos da solução a produzir



[OpenUP, 2020]

Processos de Desenvolvimento Ágil

Diferentes processos de desenvolvimento ágil de software têm sido propostos, nomeadamente:

- *Extreme Programming* (XP)
- *Dynamic Systems Development Method* (DSDM)
- *Crystal Methodology*
- *Feature-Driven Development* (FDD)
- *Lean Development*
- *Scrum Development Process*

Processo *Extreme Programming* (XP)

- O processo *Extreme Programming* (XP) é um processo de desenvolvimento de software ágil que tem por objectivo **produzir software de elevada qualidade de forma eficiente e adaptável à incerteza e mudança** das soluções a produzir
- É **centrado na satisfação do cliente,** com a **produção de versões executáveis, em ciclos de desenvolvimento curtos**, no sentido de agilizar o processo de desenvolvimento e de introduzir pontos de controlo do desenvolvimento que permitam a inclusão de novos requisitos dos clientes de forma sistemática mas flexível
- **Enfatiza o trabalho de equipa** em equipas auto-organizadas
- O processo XP **integrou diferentes práticas de desenvolvimento de software, levando-as em alguns casos a níveis extremos**, como é o caso da utilização intensiva de testes automatizados que validam o funcionamento mesmo de pequenas partes do sistema a produzir, em vez de testar apenas características de âmbito mais abrangente
- Tem a sua origem no projecto C3 (*Chrysler Comprehensive Compensation System*), no âmbito do desenvolvimento liderado por Kent Beck a partir de Março de 1996

Processo *Extreme Programming* (XP)

O processo *Extreme Programming* (XP) define um conjunto de práticas principais, organizadas em quatro áreas:

- **Informação (*feedback*) atempada e detalhada**
  - Programação em pares
  - Desenvolvimento guiado por testes
  - Equipa integral (inclusão do cliente no processo de desenvolvimento)
- **Processo contínuo**
  - Integração contínua
  - Refactorização (melhoria contínua da arquitectura e implementação)
  - Entregas de pequena dimensão
- **Conhecimento partilhado**
  - Normas de escrita de código
  - Propriedade colectiva do código
  - Arquitectura simples
  - Metáfora do sistema
- **Bem-estar do programador**
  - Ritmo sustentado (desenvolvimento capaz de ser mantido a um ritmo constante)
  - Semana de 40 horas

Processo *Extreme Programming* (XP)

O processo *Extreme Programming* (XP) integrou e dinamizou diferentes práticas de desenvolvimento de software, levando-as em alguns casos a níveis extremos, como é o caso do *desenvolvimento guiado por testes*, nomeadamente:

- **Histórias de utilizador** (*User Stories*)
  - Descrições informais, em linguagem natural, da forma como os utilizadores podem concretizar os seus objectivos de utilização do sistema
- **Protótipos exploratórios** (*Spikes*)
  - Protótipos de arquitectura e de implementação elaborados de forma expedita, num tempo curto, com o objectivo de reduzir a incerteza acerca das soluções a desenvolver
- **Programação em pares** (*Pair programming*)
  - Dois programadores participam em simultâneo num desenvolvimento conjunto
- **Desenvolvimento guiado por testes** (*Test-driven development*)
  - Técnica de programação que implica escrever primeiro casos de teste e depois implementar o código necessário para passar nos testes
- **Refactorização** (*Refactoring*)
  - Alteração da estrutura interna de uma parte de software para melhorar a sua arquitectura, compreensibilidade e adaptabilidade, sem alterar o seu comportamento observável

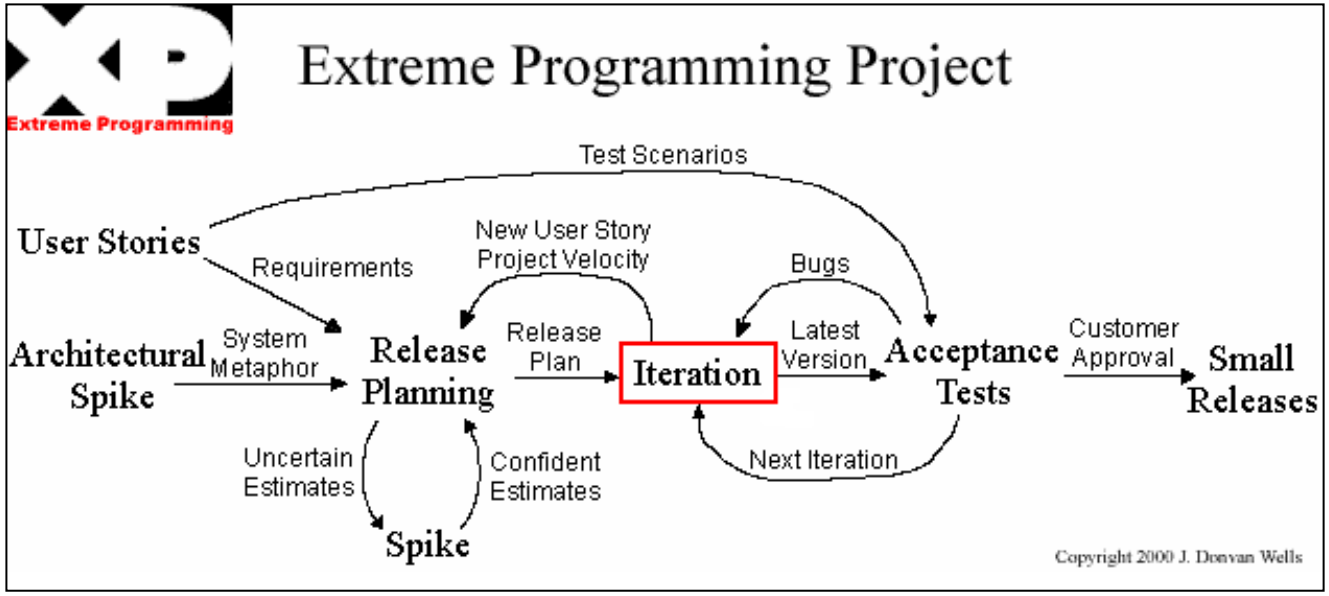
Processo *Extreme Programming* (XP)

No processo *Extreme Programming* (XP) é enfatizada a **importância do código** produzido – o artefacto principal para que exista um produto funcional

Nesse sentido, o ciclo de desenvolvimento de um projecto é **centrado em actividades orientadas para a produção de código funcional**, em entregas incrementais de pequena dimensão

Os requisitos são representados sob a forma de histórias de utilização (*user stories*), os quais servem de base ao planeamento das entregas (*release planning*)

No início do desenvolvimento, para redução de incerteza e aumento da confiança das estimativas, são elaborados protótipos exploratórios *(spike*) de arquitectura e implementação, em particular, é definido o conceito *metáfora do sistema* que consiste numa arquitectura exploratória baseada na analogia com um domínio real, a qual serve de base para a actividade de planeamento das entregas



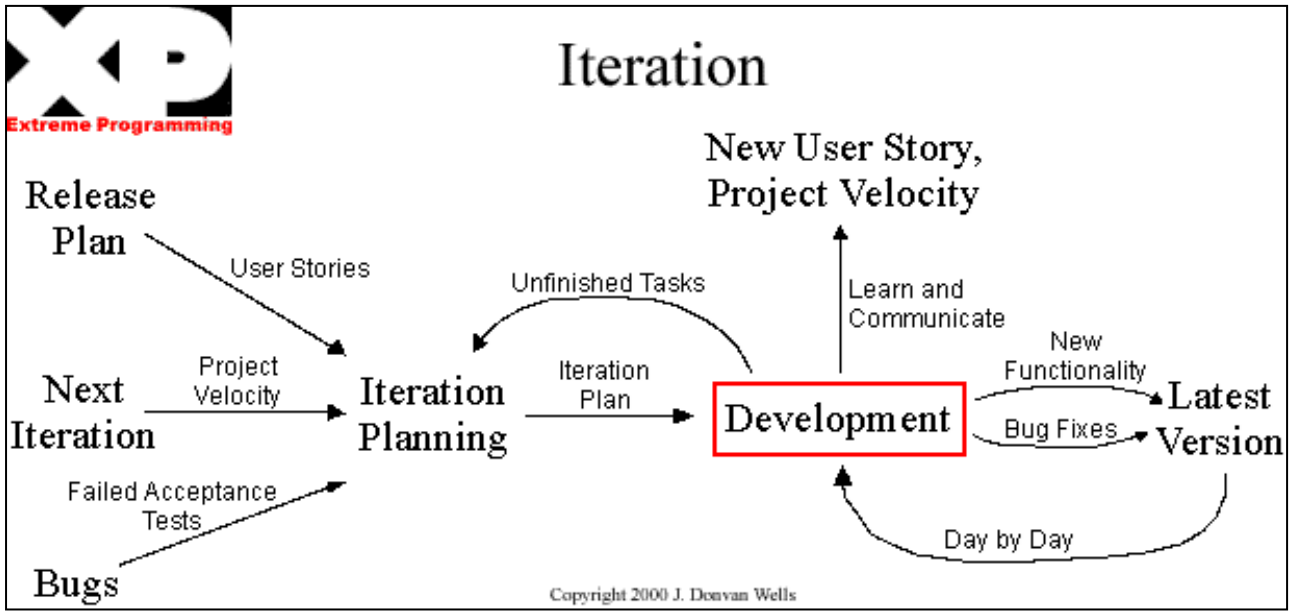
[www.extremeprogramming.org]

Processo *Extreme Programming* (XP)

No processo *Extreme Programming* (XP) uma iteração inicia-se com o planeamento da iteração, tendo por base o plano de entregas (*release plan*) definido, nomeadamente, em termos das histórias de utilização a realizar na iteração, dos erros pendentes e da velocidade a que o projecto está a decorrer

No dia-a-dia da iteração são desenvolvidas novas funcionalidades que concretizam os requisitos e corrigidos erros pendentes

É dado ênfase à comunicação e ao conhecimento partilhado, incluindo informação referente à velocidade a que o projecto está a ocorrer, relativa, entre outros aspectos, ao número de requisitos concretizados por unidade de tempo



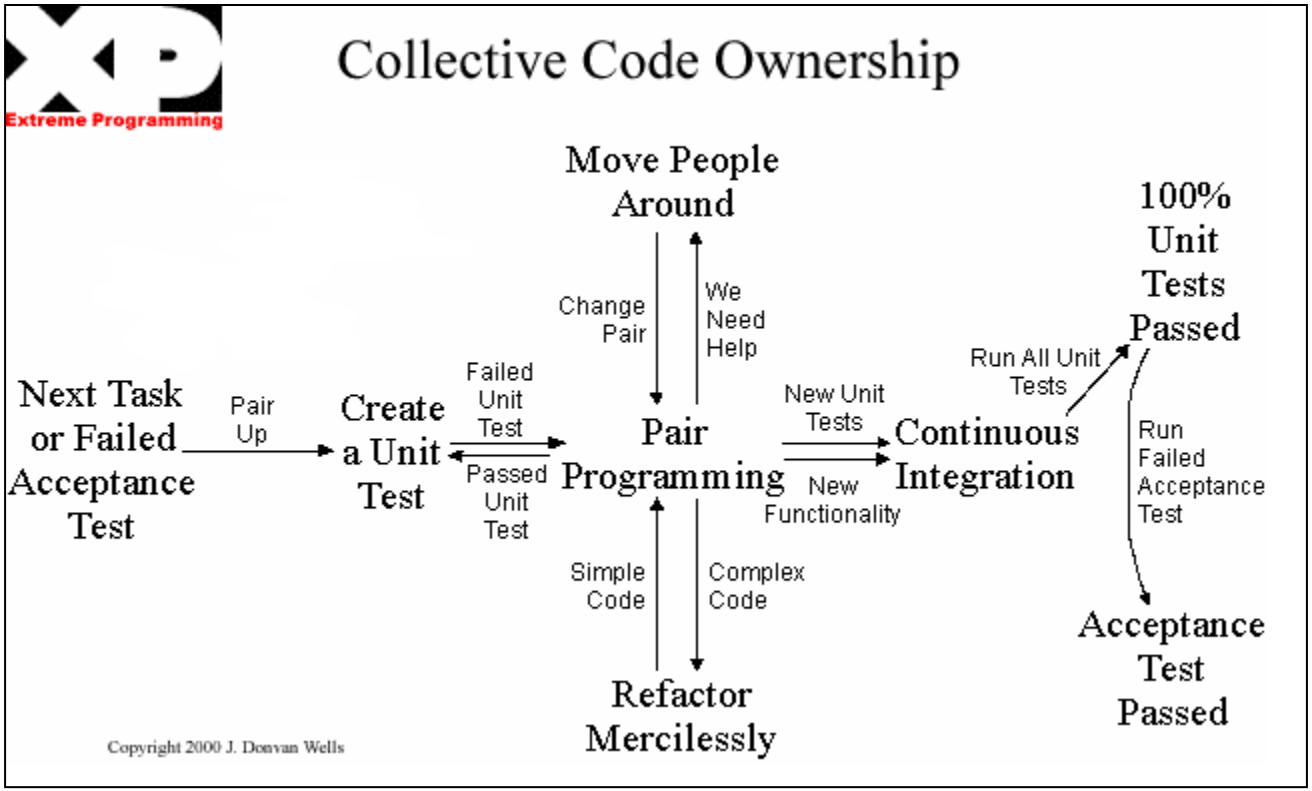
[www.extremeprogramming.org]

Processo *Extreme Programming* (XP)

No processo *Extreme Programming* (XP) é enfatizado **o conhecimento partilhado**, nomeadamente, no que é designado como ***propriedade colectiva do código*** (*collective code ownership*)

A propriedade colectiva do código é um método de trabalho na qual todos os membros da equipa têm a capacidade de aceder e alterar qualquer parte do código do projecto de acordo com o necessário, seja para completar uma tarefa de desenvolvimento, para corrigir erros, ou para melhorar a estrutura do código

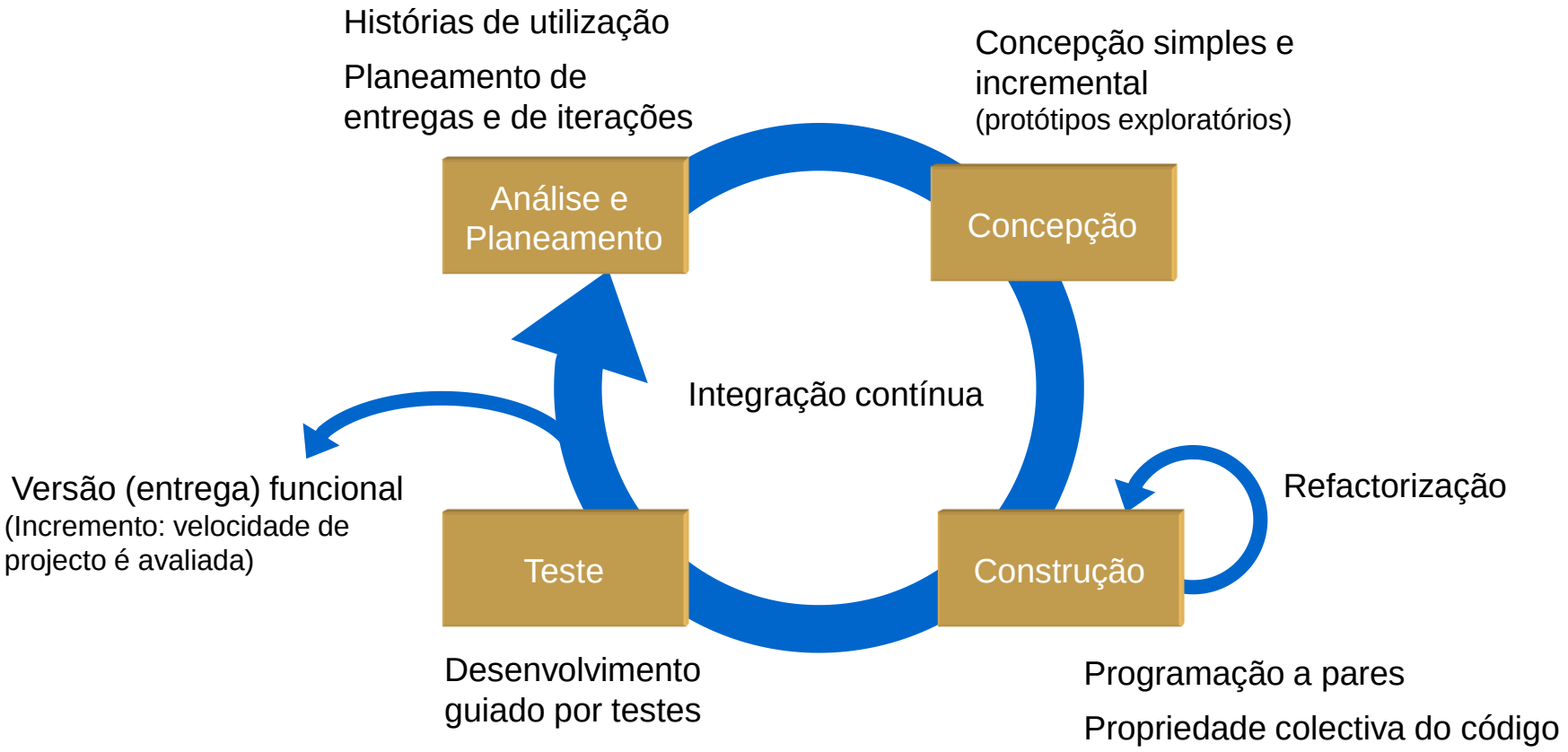
A propriedade colectiva do código também é dinamizada pela prática da *programação a pares* (*pair programming*) e pela *refactorização*



[www.extremeprogramming.org]

Processo *Extreme Programming* (XP)

Organização geral do processo *Extreme Programming* (XP) em termos das principais actividades de um processo de desenvolvimento de software



Processo *Extreme Programming* (XP)

O processo *Extreme Programming* (XP) é uma metodologia ágil de desenvolvimento de software que enfatiza a entrega contínua de software funcional e de alta qualidade, com foco particular na obtenção de conhecimento de forma rápida e permanente, e na colaboração e no trabalho em equipa, apresentando vantagens e desvantagens em termos da sua utilização concreta, nomeadamente:

- **Vantagens**
  - Adaptação dinâmica e flexível aos requisitos do projecto
  - Ênfase na colaboração e no trabalho em equipa
  - Entrega frequente de software funcional
  - Foco na obtenção de conhecimento (*feedback*) de forma rápida e permanente
  - Foco na qualidade do código
  - Processo flexível com baixo nível de formalidade
- **Desvantagens**
  - Requer uma equipa de desenvolvimento altamente qualificada e motivada
  - Pode ser difícil de implementar em projectos de grande dimensão
  - Pode ser difícil de implementar com clientes com uma cultura mais formal
  - Requer alguma mudança de mentalidade em relação às abordagens clássicas de desenvolvimento de software
  - Pode ser difícil de gerir para equipas com pouca experiência

Desenvolvimento Ágil: Evolução

O ***desenvolvimento de ágil*** define um conjunto de práticas de desenvolvimento de software que enfatizam aspectos como a **simplicidade, a comunicação, a obtenção e partilha rápida de conhecimento, ou a entrega contínua de valor**

No entanto, processos de desenvolvimento ágil como o *Extreme Programming* (XP) têm uma **elevada dependência de equipas de desenvolvimento altamente qualificadas, motivadas e auto-organizadas**, o que as torna capazes de realizar trabalho com elevada eficiência e de lidar de forma flexível com mudanças rápidas ou não antecipadas nos requisitos

No entanto, esse tipo de equipas tem por base relações de grande proximidade e dependência entre os elementos das equipas, constituindo um **factor limitativo da dimensão dos projetos** que podem ser desenvolvidos com processos ágeis, nomeadamente no que se refere aos seguintes aspectos:

- **Comunicação**: Em equipas maiores a comunicação pode ser menos eficiente, o que pode levar, por exemplo, a problemas de coordenação
- **Complexidade**: Projetos maiores, mais complexos e difíceis de gerir, dificultam as entregas em ciclos curtos e a adaptação a mudanças nos requisitos
- **Eficiência**: Equipas maiores requerem maior formalidade na sua organização, o que as pode tornar menos eficientes

Desenvolvimento Ágil: Evolução

- Pelas suas características, as metodologias de desenvolvimento ágil **tornaram-se progressivamente dominantes** em projectos de desenvolvimento de software
- No entanto, para além do problema da escalabilidade e dificuldade de aplicação, o entusiasmo e a falta de conhecimento efectivo em relação às metodologias ágeis, levou a abusos e má utilização, servindo para justificar a **pouca ou nenhuma disciplina no desenvolvimento de software**
- O resultado foi a **anarquia e o caos**, levando a falhas no desenvolvimento de muitos projectos
- Para ultrapassar esses problemas e as limitações de escala do desenvolvimento ágil, têm sido propostas metodologias no sentido de, **mantendo os princípios de desenvolvimento ágil, incorporar a disciplina necessária ao desenvolvimento de software de forma organizada e sistemática** de modo a garantir a eficiência e a qualidade do trabalho realizado

Desenvolvimento Ágil Disciplinado

O *desenvolvimento ágil disciplinado* mantém os princípios e práticas de desenvolvimento ágil, mas organiza o ciclo de vida de um projecto de forma sistemática, definindo:

- **O que fazer**
  - O que produzir em cada etapa de um projecto
- **Quem faz**
  - Quem participa no desenvolvimento do que é produzido
- **Quando fazer**
  - Como se organizam as actividades de desenvolvimento ao longo do *ciclo de vida de um projecto*

Desenvolvimento Ágil Disciplinado

- **O que fazer:** organização do que vai ser produzido
  - **Itens de trabalho do produto**
    - O que é necessário realizar para concretizar a solução
    - Âmbito de projecto
  - **Itens de trabalho da iteração**
    - O que foi planeado concretizar na iteração
    - Âmbito de iteração
  - **Tarefas a realizar na iteração**
    - Para concretizar os itens de trabalho da iteração
- **Listas de itens de trabalho (*backlog*)**
  - Sequência ordenada de itens de trabalho
    - Prioridade
    - Esforço
  - De produto
  - De iteração

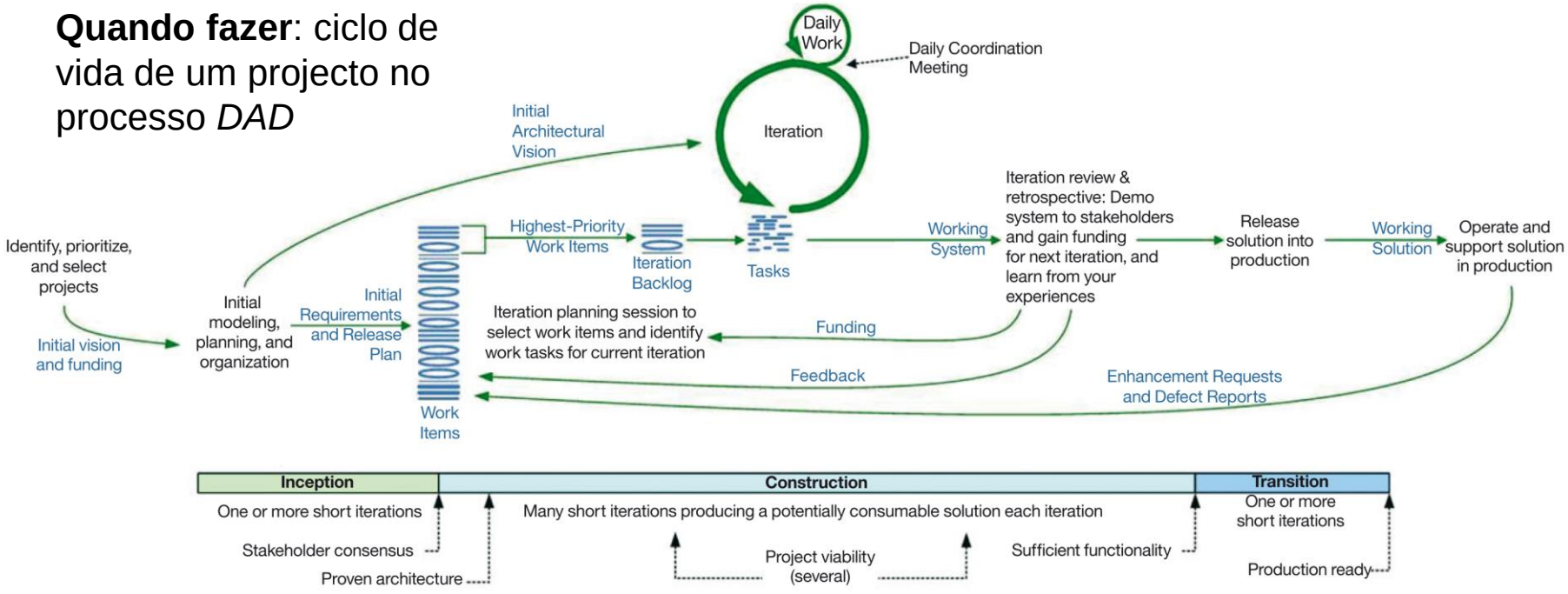
Desenvolvimento Ágil Disciplinado

- **Quem faz:** 3 papéis principais
  - **Coordenador da equipa de desenvolvimento**
    - Responsável pelo sucesso do projeto e pela correcta aplicação do processo de desenvolvimento
  - **Equipa de desenvolvimento**
    - Responsável pelo desenvolvimento do produto com base no processo de desenvolvimento definido
  - **Dono do produto (*Product Owner*)**
    - Define e promove a visão, os objectivos e as características do produto, para que a equipa de desenvolvimento possa tomar as decisões adequadas de arquitectura e implementação
    - É responsável pela *lista de itens de trabalho* e define os critérios de aceitação para os itens de trabalho
    - Determina o âmbito/conteúdo de cada entrega (planeamento da entrega)
    - Define os critérios de aceitação de cada entrega e define quando o produto está pronto para ser disponibilizado

Processo *Disciplined Agile Delivery* (DAD)

Um exemplo de metodologia ágil disciplinada é o processo *Disciplined Agile Delivery* (DAD)

Quando fazer: ciclo de vida de um projecto no processo *DAD*



[Ambler, 2011]

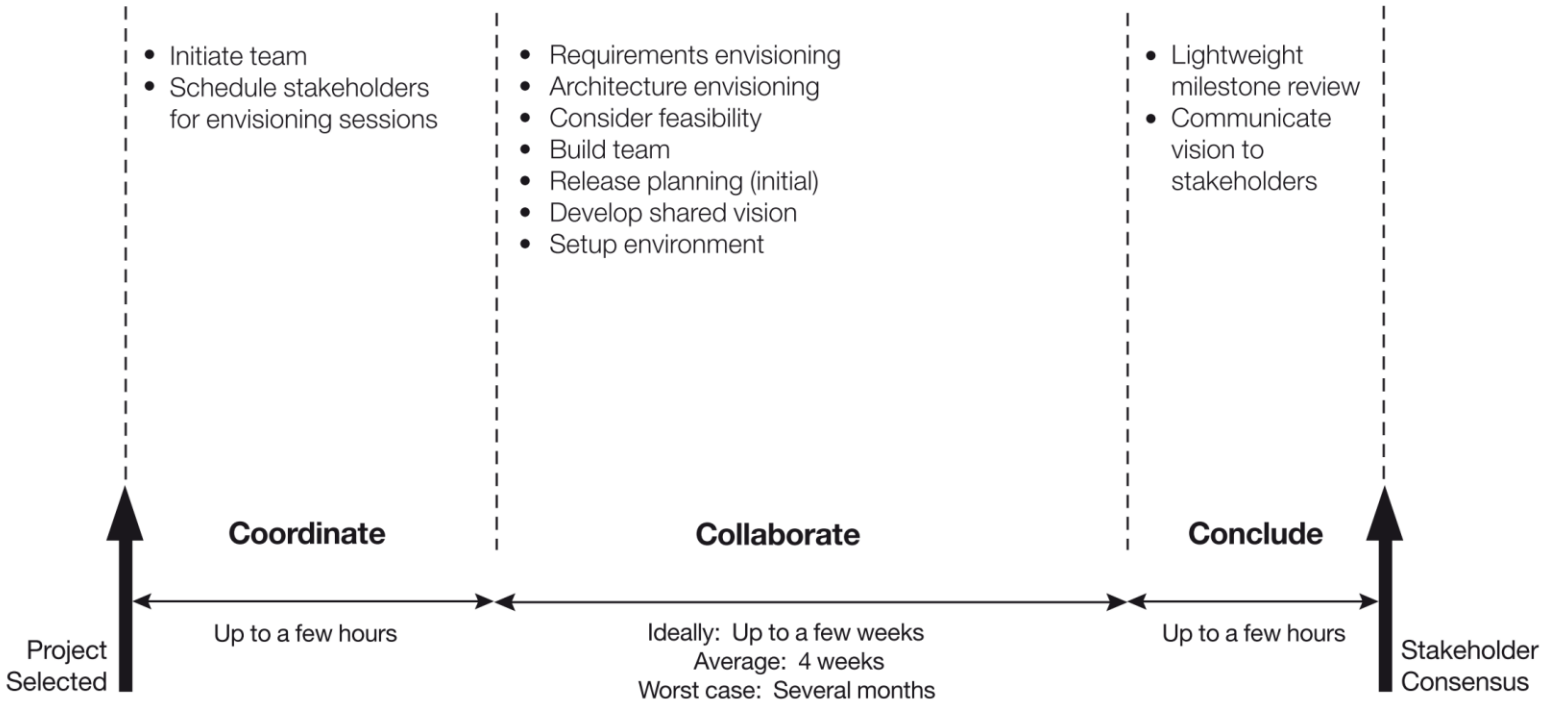
Ciclo de vida de um projecto organizado em 3 fases:

- **Fase inicial (*Inception*)**
  - É iniciado o projecto em termos de requisitos e recursos envolvidos
- **Fase de construção**
  - São realizadas as iterações de desenvolvimento do produto
- **Fase de transição**
  - É preparada e implantada a solução em produção

Fase Inicial (*Inception*)

Objectivos para a fase inicial (*Inception*)

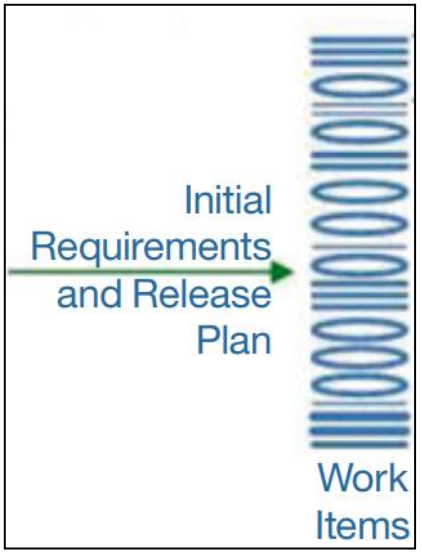
- Definir a visão do projeto
- Chegar a um acordo entre as partes interessadas em torno da visão
- Definir a estratégia técnica inicial, requisitos iniciais e plano do projeto
- Formar a equipa inicial
- Garantir o financiamento
- Identificar os riscos



[Ambler, 2011]

Desenvolvimento Ágil Disciplinado

- **Listas de itens de produto (*backlog* de produto)**
  - Lista do trabalho a realizar para desenvolver um produto
  - Organizada por prioridade
  - Deve incluir características visíveis ao cliente e requisitos técnicos necessários ao desenvolvimento do produto
  - Características a implementar mais tarde podem ser menos detalhadas
  - Nível de detalhe
    - Global
    - 10 dias/pessoa de trabalho
  - Produzida como resultado da fase incial, considerando os requisitos definidos e o plano de entregas
  - Serve de base ao planeamento das iterações

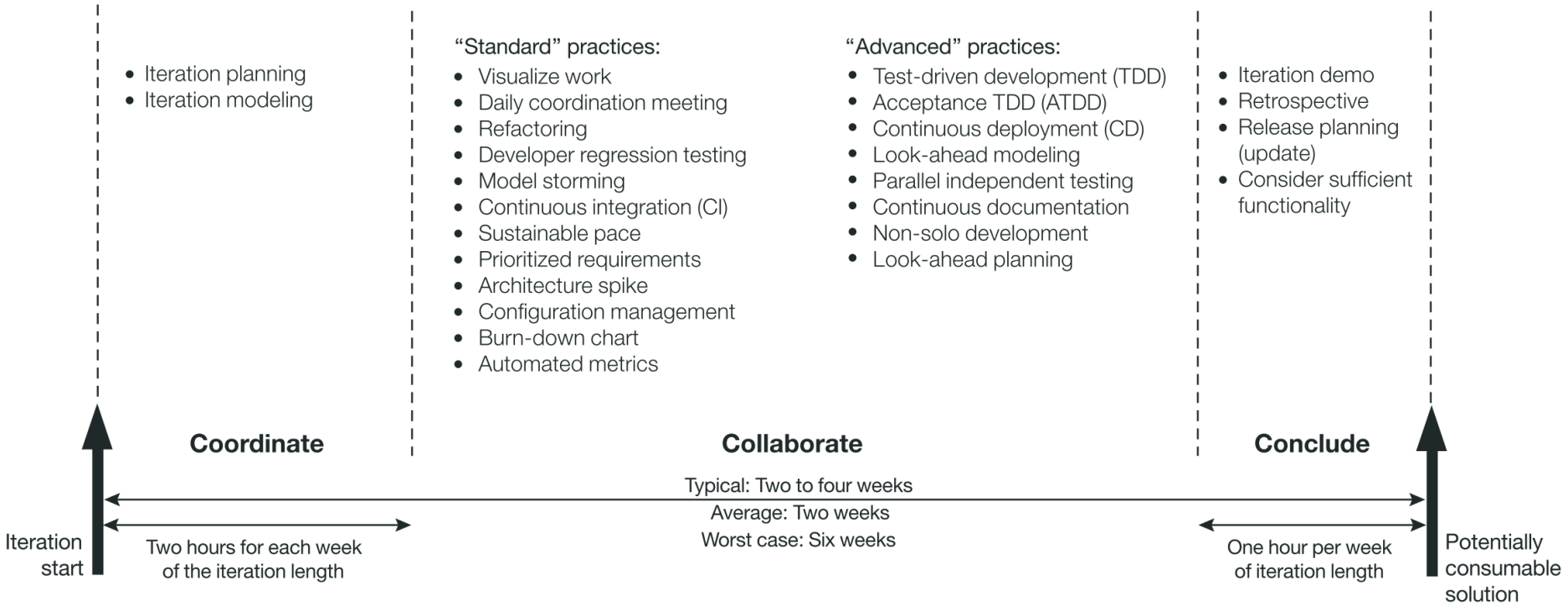


[Ambler, 2011]

Fase de Construção

Objectivos para as Iterações da fase de construção

- Produzir uma solução potencialmente utilizável
- Atender às necessidades das partes interessadas que podem variar
- Aproximar-se de uma versão implantável para entrada em produção
- Manter ou melhorar os níveis de qualidade existentes
- Abordar o(s) risco(s) mais elevado(s)



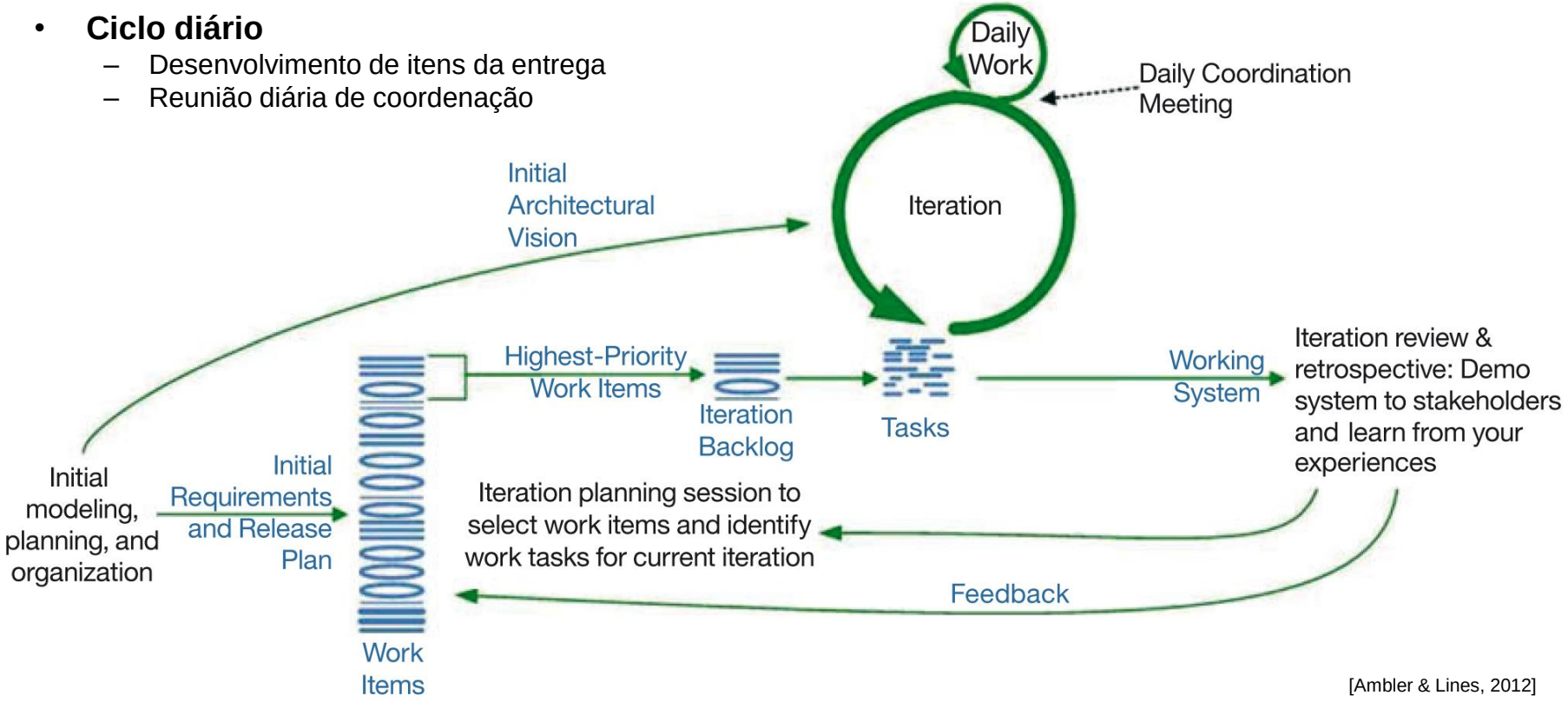
[Ambler, 2011]

Processo *Disciplined Agile Delivery* (DAD)

Ciclo de vida de um projecto no processo *DAD*: **Fase de construção**

3 ciclos principais de desenvolvimento

- **Ciclo de produto**
  - Desenvolvimento de uma versão do produto
- **Ciclo de iteração**
  - Desenvolvimento de uma entrega
- **Ciclo diário**
  - Desenvolvimento de itens da entrega
  - Reunião diária de coordenação



[Ambler & Lines, 2012]

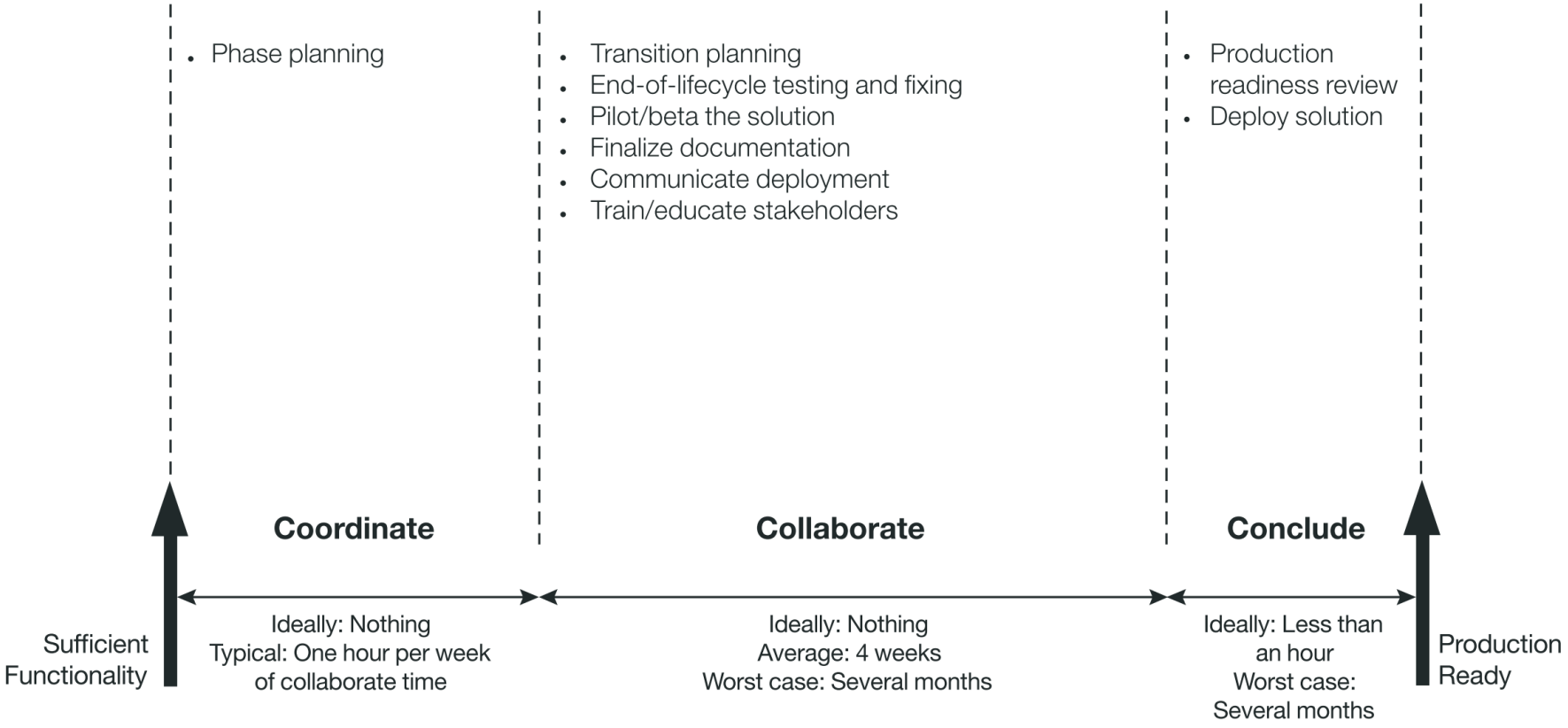
Desenvolvimento Ágil Disciplinado

- **Lista de itens de iteração (*backlog* de iteração)**
  - Lista de itens de trabalho a concretizar na iteração
  - Organizado por prioridade
  - Incluí os requisitos de maior prioridade no *backlog* de produto
  - Nível de detalhe
    - Específico
      - Itens detalhados (e.g. cenários de casos de utilização)
    - 2 dias/pessoa de trabalho

Fase de Transição

Objectivos para a fase de transição

- Garantir que a solução esteja pronta para produção
- Assegurar que as partes interessadas estão preparadas para receber a solução
- Implantar a solução em produção



[Ambler, 2011]

Processo *Disciplined Agile Delivery* (DAD)

O processo *Disciplined Agile Delivery* (DAD) é baseado nos princípios de desenvolvimento ágil, mas com uma maior ênfase na disciplina e no rigor na implementação do processo, facilitando a adopção para o desenvolvimento de projectos de maior dimensão e mais complexos, no entanto, pode ser menos flexível devido a um maior nível de formalidade

Vantagens do processo DAD em relação ao processo XP:

- **Maior disciplina**: O DAD é mais disciplinado em relação ao XP, pois estabelece regras e processos mais rígidos para garantir a qualidade e a entrega do produto, isso pode ser benéfico para equipas que precisam de uma estrutura mais rigorosa para trabalhar de forma eficiente
- **Menos improviso**: O DAD é menos baseado no improviso que o XP, pois estabelece uma estrutura mais precisa para a realização das atividades de desenvolvimento, o que contribui para evitar que o processo de desenvolvimento se torne desorganizado
- **Melhor suporte para projetos de maior dimensão**: O DAD é mais adequado para projectos maiores e mais complexos do que o XP, pois estabelece uma estrutura com maior formalidade e rigor para a gestão de um projecto

Processo *Disciplined Agile Delivery* (DAD)

Desvantagens do processo DAD em relação ao processo XP:

- **Menos flexibilidade**: O DAD é menos flexível do que o XP, pois estabelece regras e processos mais rígidos que podem dificultar a adaptação a mudanças no projecto ou na equipa
- **Maior formalidade**: O DAD é mais formal do que o XP, pois estabelece uma estrutura melhor definida e mais rígida para a gestão de projectos, o que pode aumentar a quantidade de documentação a produzir e introduzir algum nível de burocracia
- **Menos envolvimento da equipa**: O DAD pode ser menos participativo do que o XP, pois estabelece uma estrutura mais rígida para a gestão de projectos, o que pode limitar a participação da equipa de desenvolvimento na tomada de decisões
- **Maiores custos**: O DAD pode ter custos mais elevados do que o XP, pois estabelece uma estrutura mais formal e rígida para a gestão de projetos, o que pode implicar maiores custos de formação e de implementação

Bibliografia

[Watson, 2008]
Andrew Watson, *Visual Modeling: past, present and future*, OMG, 2008.

[Meyer, 1997]
B. Meyer, *UML: The Positive Spin*, American Programmer - Special UML issue, 1997.

[Ambler & Lines, 2012]
S. Ambler, M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, IBM Press, 2012.

[Selic, 2003]
B. Selic, *Brass bubbles: An overview of UML 2.0*, Object Technology Slovakia, 2003.

[Graessle, 2005]
P. Graessle, H. Baumann, P. Baumann, *UML 2.0 in Action*, Packt Publishing, 2005.

[Eriksson et al., 2004]
H. Eriksson, M. Penker, B. Lyons, D. Fado, *UML 2 Toolkit*, Wiley, 2004.

[USDT, 2005]
U.S. Department of Transportation, *Clarus: Concept of Operations*, Publication No. FHWA-JPO-05-072, 2005.

[Douglass, 2006]
B. Douglass, *Real-Time UML*, Telelogic, 2006.

[OMG, 2020]
Unified Modeling Language (Specification), OMG, 2020.

[OpenUP, 2020]
Introduction to OpenUP (Open Unified Process), Eclipse Process Framework, 2020

[Ambler, 2011]
S. Ambler, *Disciplined Agile Delivery: An introduction*, IBM Corporation, 2011