Outros Processos

Prof. Marco Tulio Valente

mtov@dcc.ufmg.br

Primeiro, quando **não** usar métodos ágeis?

Quando **não** usar Métodos Ágeis?

- Requisitos estáveis
- Design conhecido e simples
- Desenvolvedores dominam a área do projeto
- Baixo risco
- Custos de mudanças é alto

Veja que iterações em XP são um mecanismo para levantar e validar requisitos, adaptar o design, dar tempo aos desenvolvedores para entender o domínio do projeto e, logo, reduzir riscos. Se essas características não estão presentes, para que iterar? Em outras palavras, pode-se implementar tudo em um única iteração

Source: Code Complete (2nd Edition), Steve McConnell

Qual método é mais adequado por tipo de sistema?

- Sistemas comerciais tendem a se beneficiar de métodos iterativos e ágeis
- Sistemas críticos (em termos de vidas humanas ou custos) tendem a requer métodos sequenciais; pois estabilidade dos requisitos é essencial para garantir altos níveis de confiabilidade

Source: Code Complete (2nd Edition), Steve McConnell

Veja este comentário do Grady Booch

- Different projects call for different methods—and it's important to weigh factors like the project's risk and the culture of the team that's executing.
- If I'm building a nuclear-power plant, I don't want to use incremental and iterative methods because testing failure is never a good thing.
- On the other hand, if I'm building a throwaway app for some new clone of Tinder for goats, whatever it might be, then sure, I'm gonna put a few people in a room and go build this.

Source: https://www.theatlantic.com/technology/archive/2017/12/agile-manifesto-a-history/547715/

E este comentário do Bertrand Meyer

- No serious engineering process can skip an initial step of careful planning. It is good to put limits on it, but irresponsible to remove it.
- The project failures that I tend to see nowadays …. are often due to an application of this agile rule; we don’t need no stinkin’ requirements phase, we are agile, let’s just produce user stories and implement them as we go. A sure way to disaster.
- No magic process can, through refactoring, turn bad design into good. Refactoring junk yields junk.

Source: Making Sense of Agile Methods, B. Meyer, IEEE Software, 2018

Meio termo também é possível

- Alternativa 1:
  - Definir 80% dos requisitos inicialmente;
  - 20% restantes serão definidos ao longo do desenvolvimento (só aceitando novos requisitos que de fato tenham valor)
- Alternativa 2:
  - Definir 20% dos requisitos inicialmente (os mais importantes);
  - Implementar os 80% restantes por meio de iterações.

Source: Code Complete (2nd Edition), Steve McConnell

Outros Processos (não ágeis)

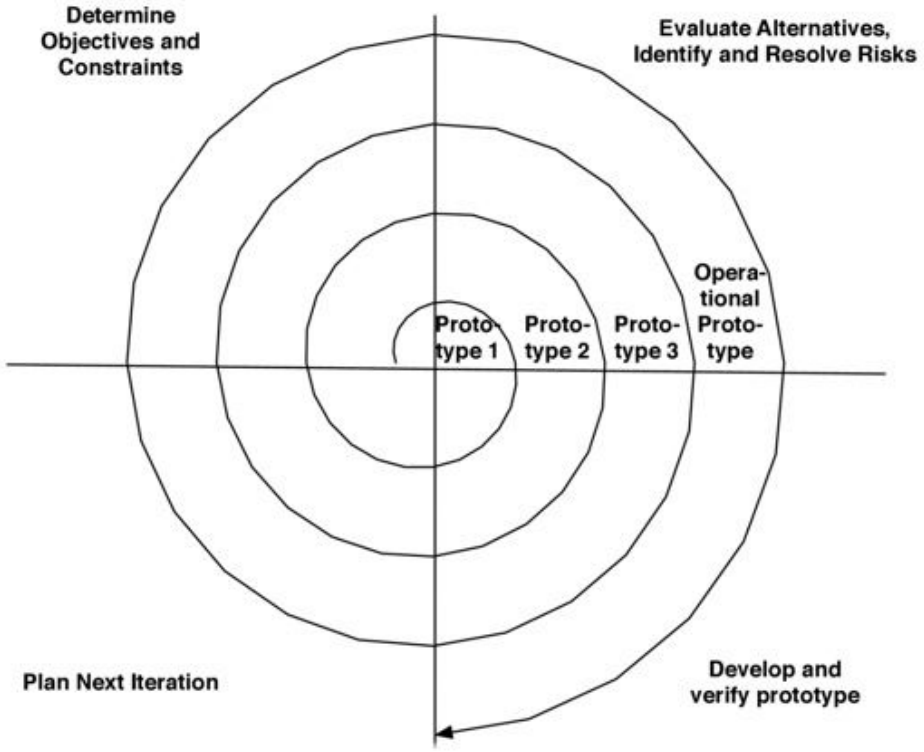
Transição de Waterfall para Ágil

- Antes da disseminação dos princípios ágeis de desenvolvimento, alguns métodos "iterativos" ou "evolucionários" foram propostos
- Ou seja, a transição entre Waterfall (~1970) e Ágil (~2000) foi gradativa
- Exemplos de métodos intermediários:
  - Espiral (1986)
  - Rational Unified Process (RUP) (2003)

Modelo em Espiral

- Proposto por Barry Boehm, 1986
- Ideia: sequência de iterações, com um protótipo sendo geral ao final de cada iteração; desenvolvimento termina quando um produto final, pronto para entrar em produção, é gerado
- Cada "iteração" possui 4 etapas (ver próximo slide):
    - Determinação de objetivos e constraints (custos, cronogramas etc)
    - Avaliação de alternativas e riscos (exemplo: framework X é robusto)
    - Desenvolvimento e testes (por exemplo, usando waterfall clássico)
    - Planejamento da próxima iteração (isto é, parar ou mais uma iteração)
- Cada iteração pode levar de 6 a 24 meses (logo, mais que em XP ou Scrum)

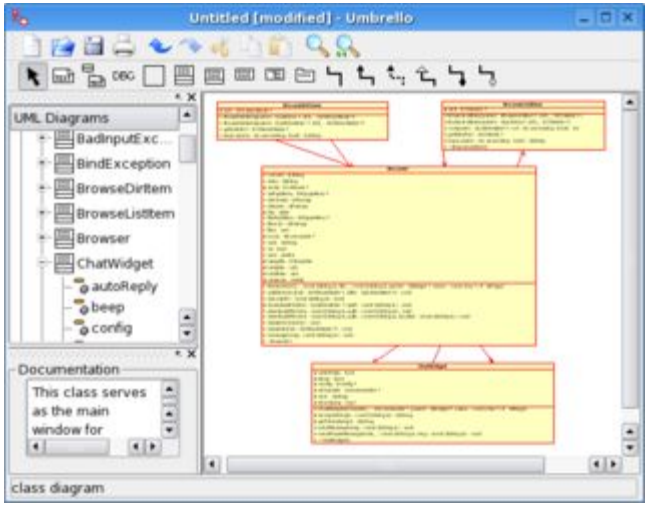
Modelo em Espiral



Source: Engineering Software As a Service: An Agile Approach Using Cloud Computing, Armando Fox & David Patterson

Rational Unified Process (RUP)

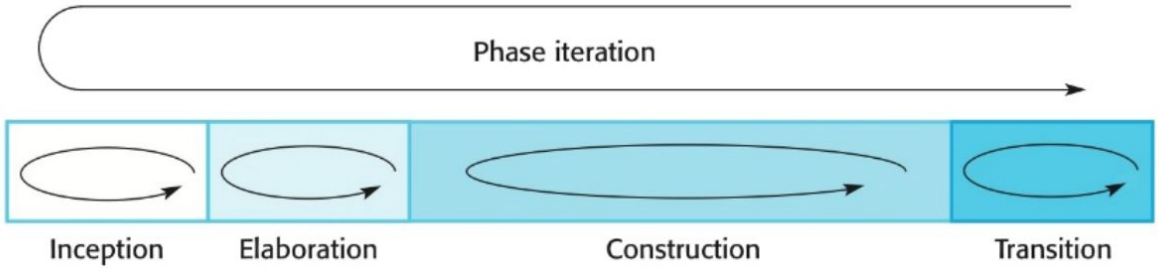
- Proposto pela Rational (uma empresa depois comprada pela IBM); centrado em diagramas UML e em ferramentas CASE, vendidas pela Rational/IBM
- CASE (Computer-Aided Software Engineering): ferramentas de modelagem e documentação de software; baseadas em diagramas UML, por exemplo



Fases do RUP

1. Inception: análise de viabilidade, orçamentos e definição de escopo
2. Elaboração: especificação de requisitos (via casos de uso) e da arquitetura
3. Construção: projeto de mais baixo nível, implementação e testes
4. Transição: disponibilização do sistema para produção

   Essas fases podem ser divididas em iterações (ex.: construção pode ter 4 iterações); pode-se também iterar sobre as 4 fases (como em Espiral)



Source:
Software Engineering,
Sommerville.

Modelos de Maturidade de Processos

Capability Maturity Model (CMMI)

- Proposto na década de 90 pelo Software Engineering Institute (SEI)
- Ideia: um processo maduro, leva a bons produtos de software
- Na prática, é uma certificação que verifica se uma organização desenvolve software usando bons processos e práticas
- Classifica organização desenvolvedora de software em cinco níveis:
  - Nível 1: inicial ou caótico (isto é, sem nenhum processo)
  - Nível 2: Repeatable (alguns processos já são repetíveis)
  - Nível 3: Defined (processos já estão devidamente documentados e definidos)
  - Nível 4: Managed (por exemplo, já monitora processos usando métricas)
  - Nível 5: Optimizing (já está no topo e agora procura basicamente otimizar partes do processo)

MPS.BR (Melhoria de Processos de Software)

- Tentativa de se criar um "CMMI Brasileiro"
- Composto por oito níveis:
  - Nível A - Em Otimização (maior maturidade)
  - Nível B - Gerenciado Quantitativamente
  - Nível C - Definido
  - Nível D - Largamente Definido
  - Nível E - Parcialmente Definido
  - Nível F - Gerenciado
  - Nível G - Parcialmente Gerenciado (menor maturidade)