ARQUITETURA DE SOFTWARE

Arquitetura de Software (Ponto de vista Ágil)

Geucimar Briatore
geucimar@up.edu.br

Atualizado em 10/2022

Manifesto Ágil (2001)

- **17 pessoas**: Alistair Cockburn, Andrew Hunt, Arie van Bennekum, Brian Marick, David Thomas, James Grenning, **Jeff Sutherland**, Jim Highsmith, Jon Kern, Ken Schwaber, **Kent Beck**, **Martin Fowler**, Mike Beedle, **Robert C. Martin**, Ron Jeffries, Steve Mellor e Ward Cunningham.
- **4 valores**
  1. Os **indivíduos** e **suas interações** acima de procedimentos e ferramentas;
  2. O **funcionamento** do software **acima de documentação** abrangente;
  3. A **colaboração** com o cliente **acima da negociação e contrato**;
  4. A capacidade de **resposta a mudanças acima de um plano pré-estabelecido**.

https://agilemanifesto.org/iso/ptbr/manifesto.html

Manifesto Ágil: 12 princípios

1. Nossa maior prioridade é satisfazer o cliente através da **entrega contínua** e adiantada de software com valor agregado.
2. **Mudanças nos requisitos são bem-vindas**, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. **Entregar frequentemente** software funcionando, de **poucas semanas a poucos meses**, com preferência à menor escala de tempo.
4. **Pessoas de negócio e desenvolvedores** devem trabalhar **diariamente em conjunto** por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de **conversa face a face**.

Manifesto Ágil: 12 princípios

7. **Software funcionando** é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua **atenção à excelência técnica e bom design** aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores **arquiteturas, requisitos e designs** emergem **de equipes auto-organizáveis**.
12. Em **intervalos regulares**, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Desvantagens da arquitetura clássica

- **Modelagens grandes e complexas**
  1. O código só é escrito depois da modelagem;
  2. Não captura as necessidades reais do usuário.

- **Falta de flexibilidade**
  1. Decisões técnicas centralizadas exclusivamente no arquiteto;
  2. Abordagem de desenvolvimento rígida, lenta, não propicia aprendizado, não permite tentativa-erro e não possibilita a evolução do código;
  3. Sistema burocrático de mudanças.

Casos de uso (Clássica) vs. histórias do usuário (Ágil)

- **Casos de uso** descrevem o que o sistema deve fazer. Ou seja, descreve as funcionalidades do ponto de vista do analista/desenvolvedor;
- **Histórias do usuário** descrevem o que o usuário precisa para realizar sua tarefa. Ou seja, descreve as funcionalidades do ponto de vista do usuário.

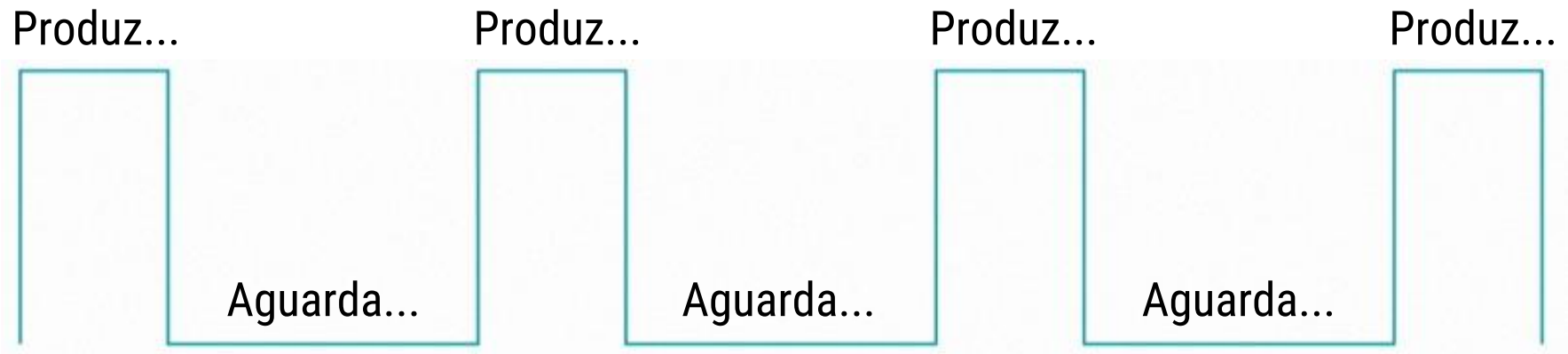
Separação de interesses em camadas (horizontal)

Camada de apresentação > Web Designers

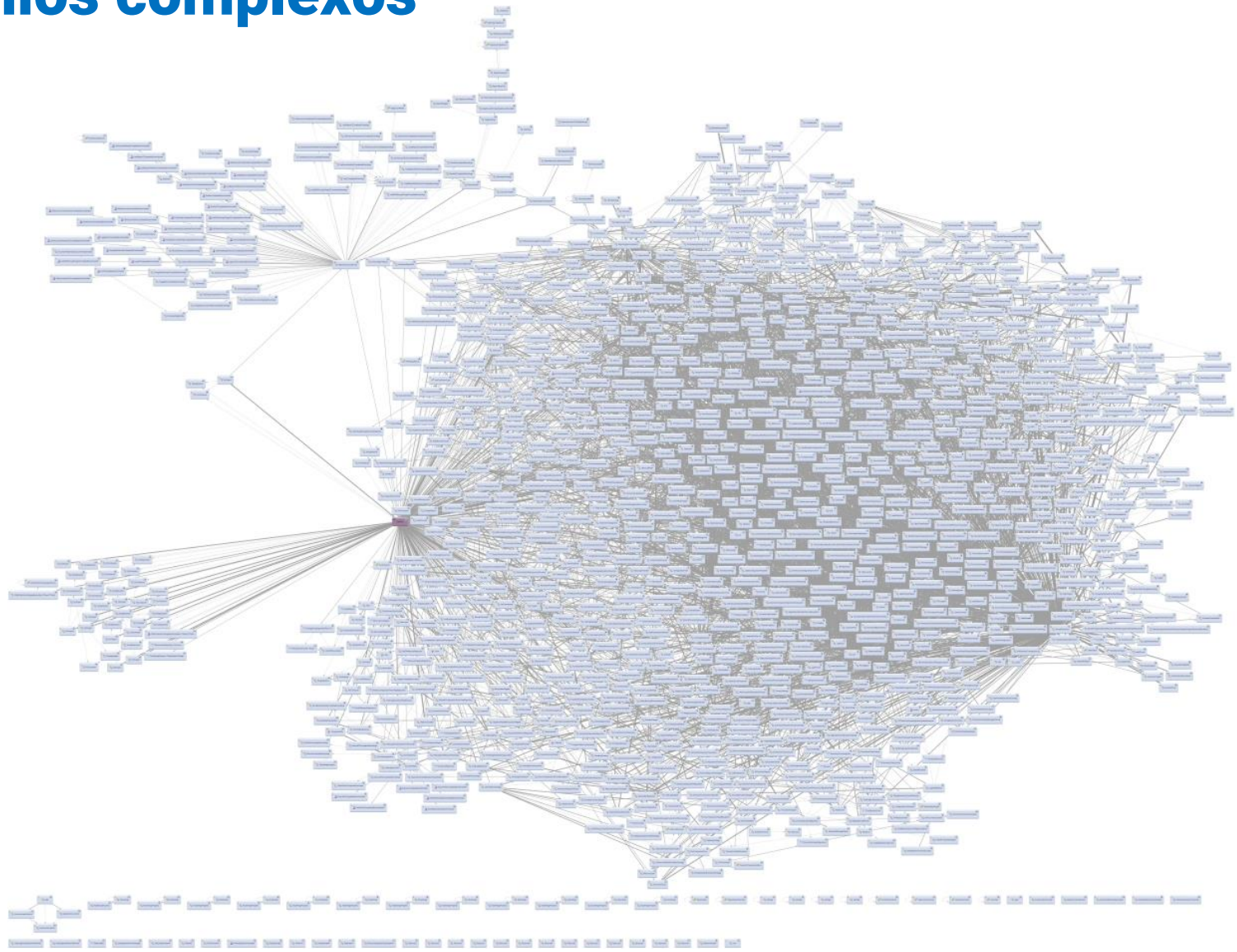
Camada de negócios > Analistas e Desenvolvedores

Camada de dados > DBAs

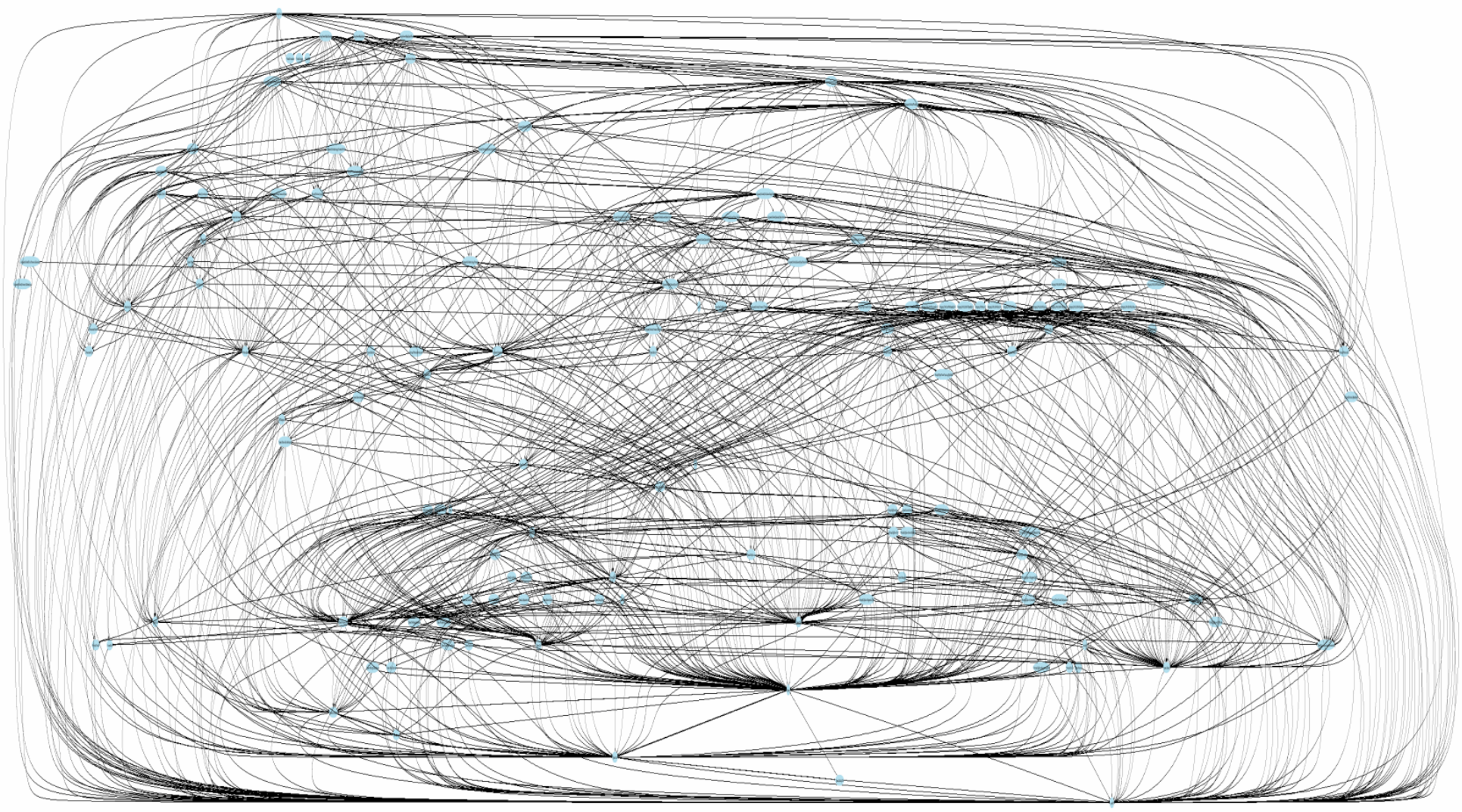
Equipes separadas por tecnologia mas dependentes...



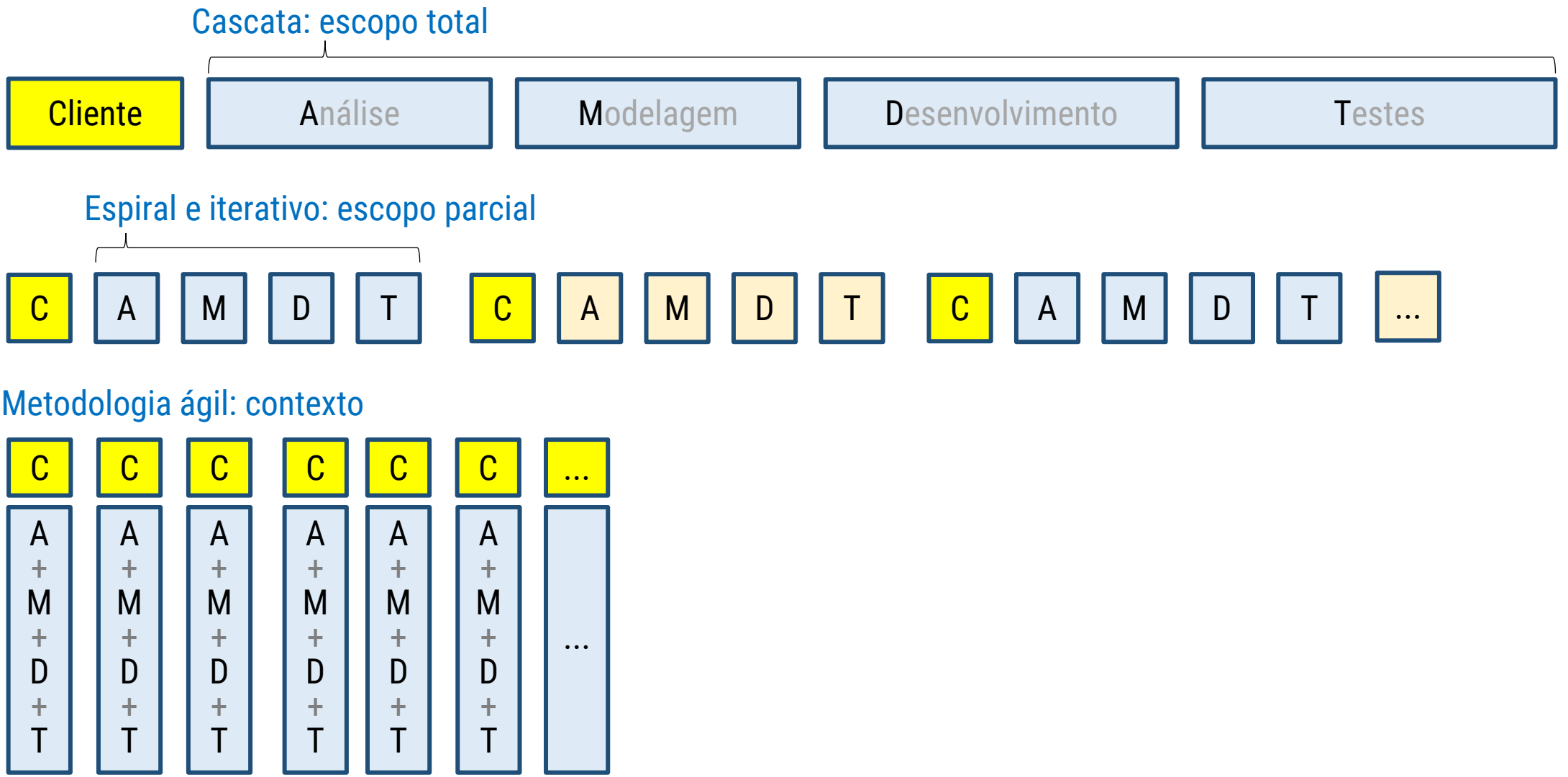
Domínios complexos



Funcionalidades interdependentes

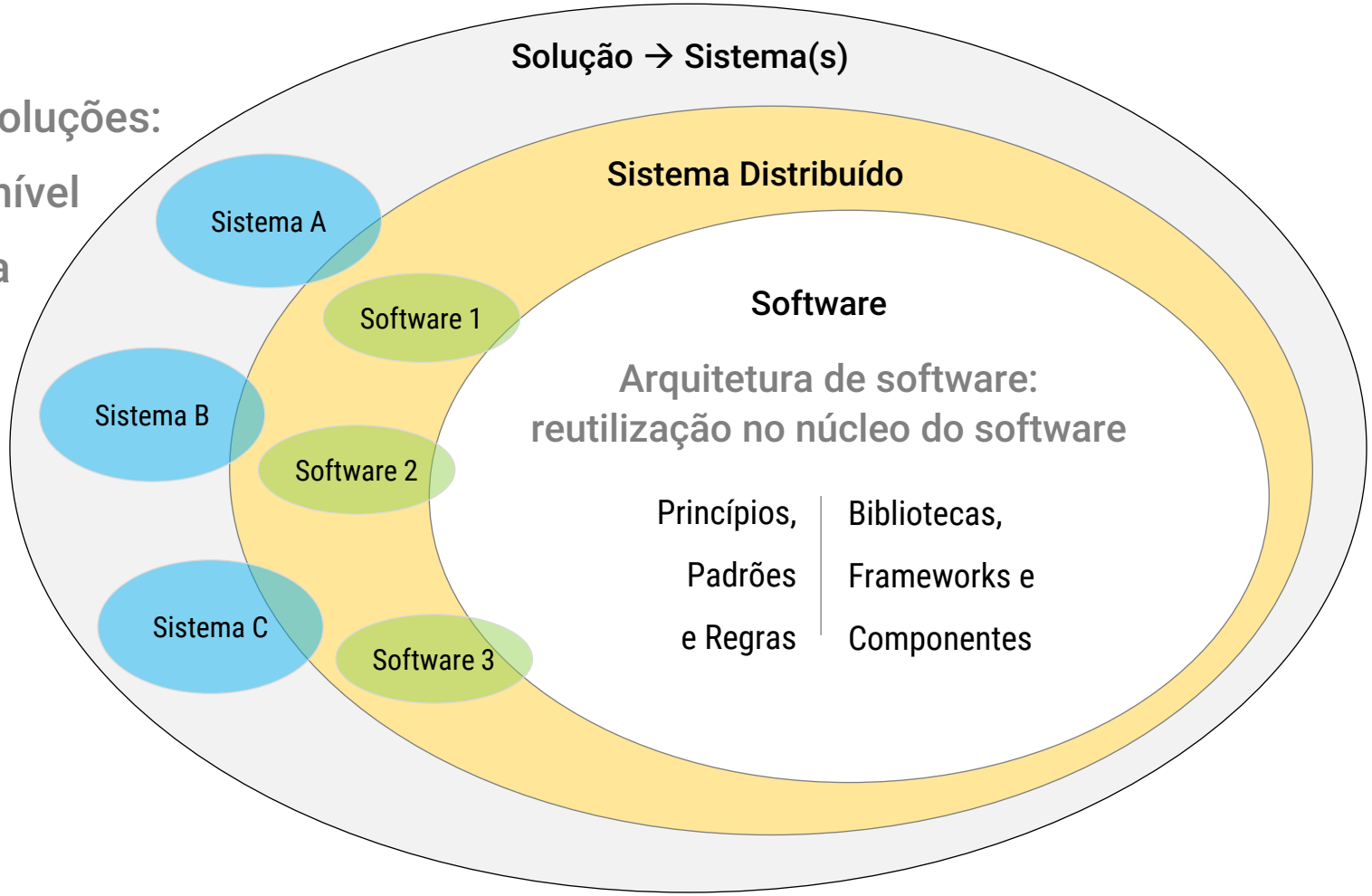


Do modelo cascata ao método ágil



Metodologia ágil: foco na automação e no núcleo do software

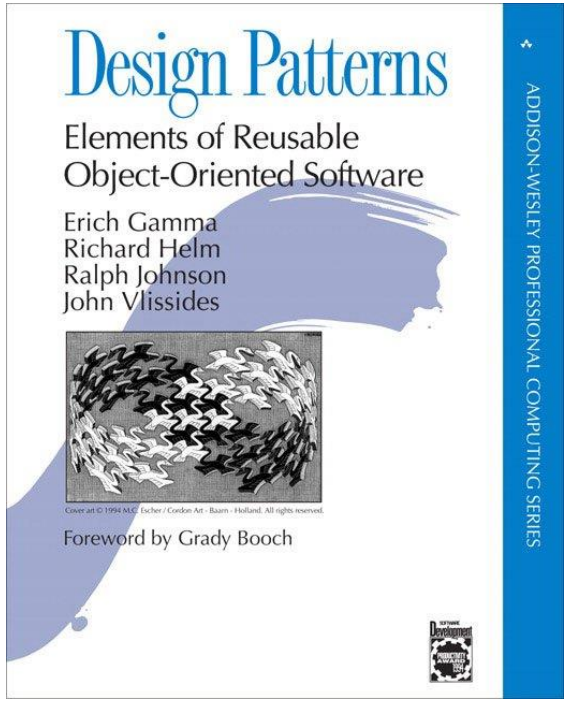
Arquitetura de soluções: reutilização em nível de infraestrutura



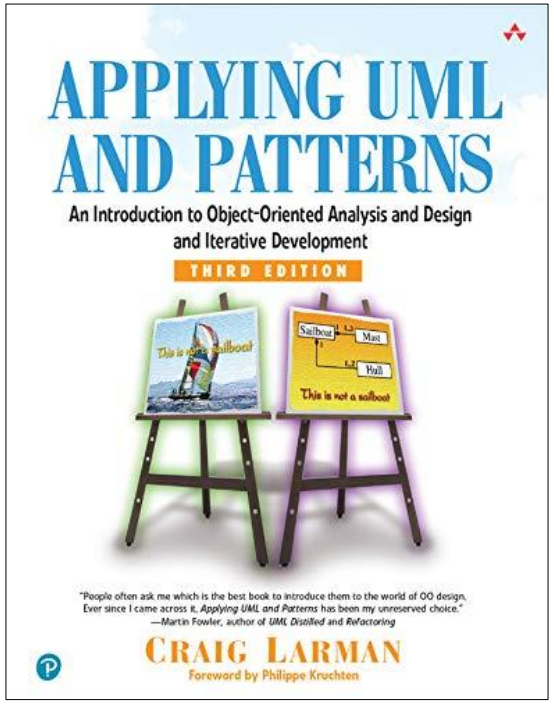
Arquitetura de software do ponto de vista ágil

- Na processo ágil **todos os desenvolvedores são ao mesmo tempo analistas, arquitetos, programadores e testers** em diferentes níveis técnicos;
- As equipes devem **focar e pensar nas funções comerciais** utilizando o conceito da modelagem baseada em domínio (DDD) ao invés da tecnologia;
- O objetivo da arquitetura de software moderna é minimizar a quantidade de recursos humanos necessários para **construir** e **manter** o sistema.

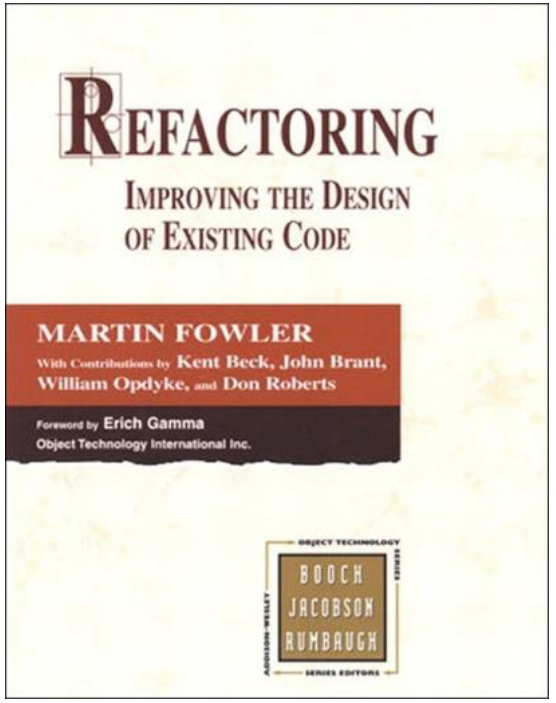
Literatura pré Manifesto Ágil



1994 (Gang of Four)

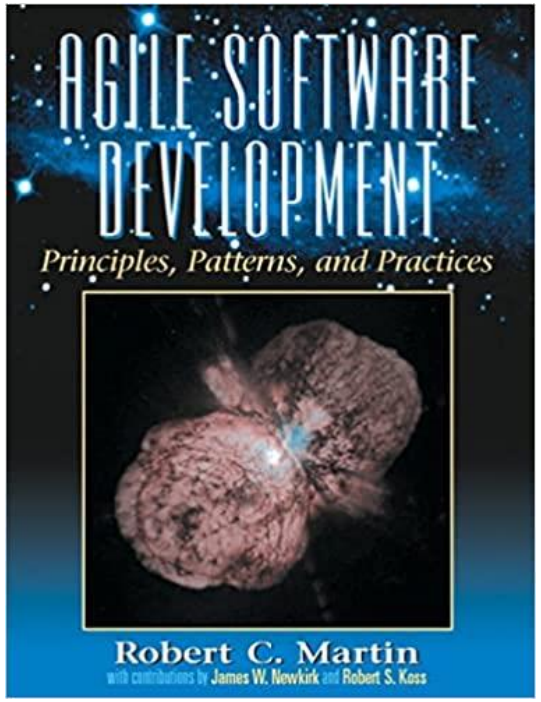


1997 (Craig Larman)

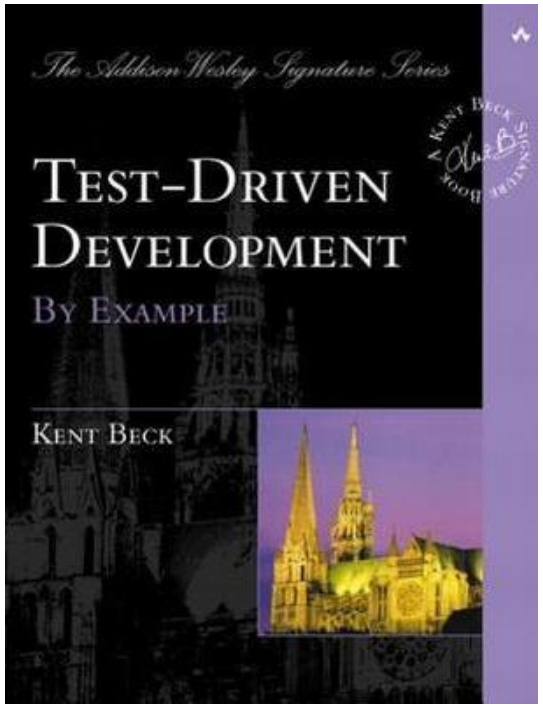


1999 (Martin Fowler)

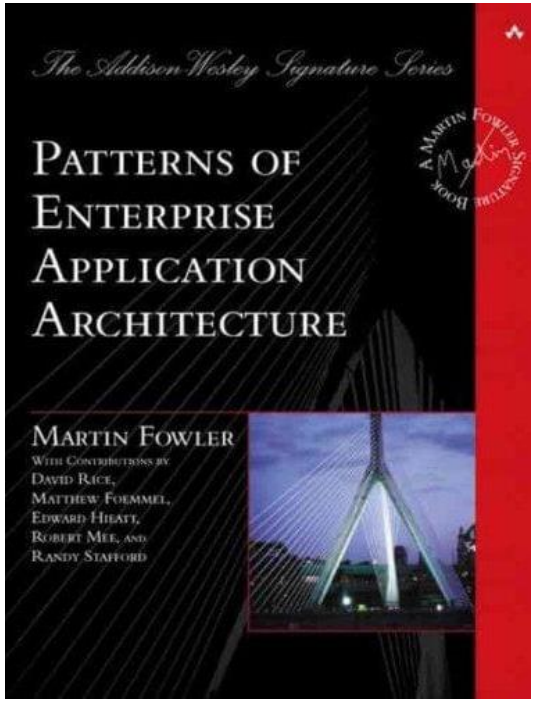
Literatura pós Manifesto Ágil



2002 (Robert C. Martin)

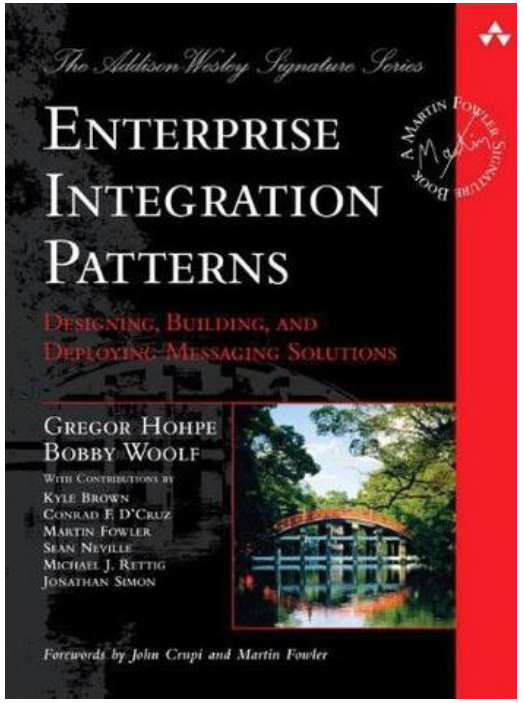


2002 (Kent Beck)

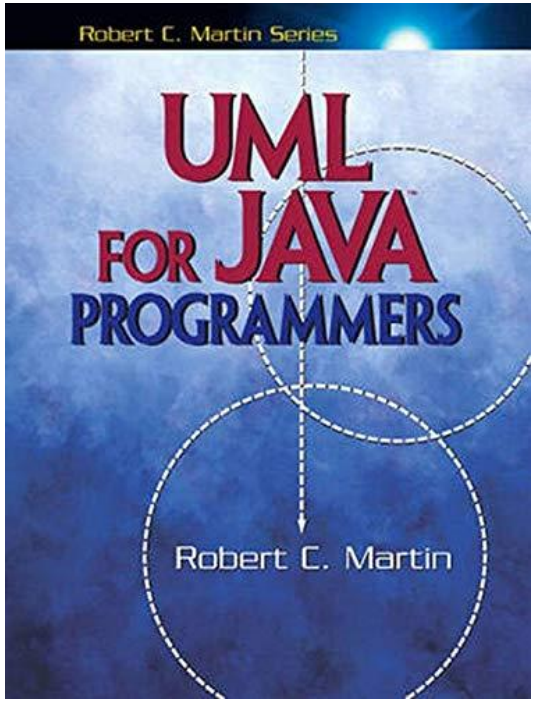


2002 (Martin Fowler)

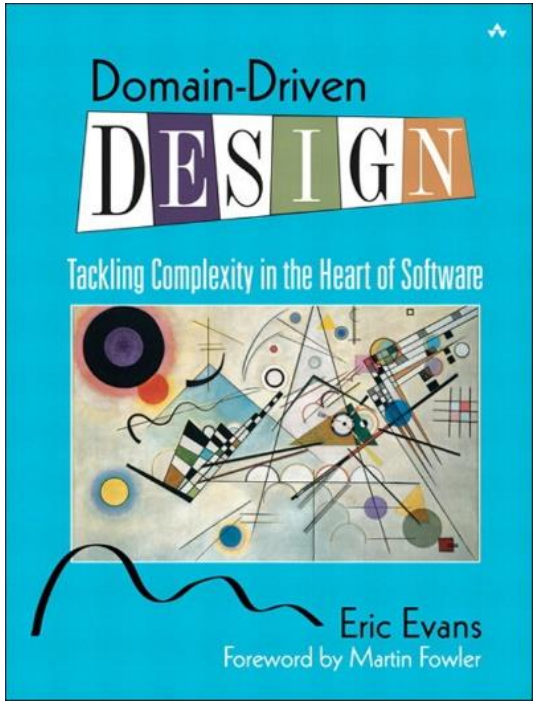
Literatura pós Manifesto Ágil



2003 (Gregor Hohpe)

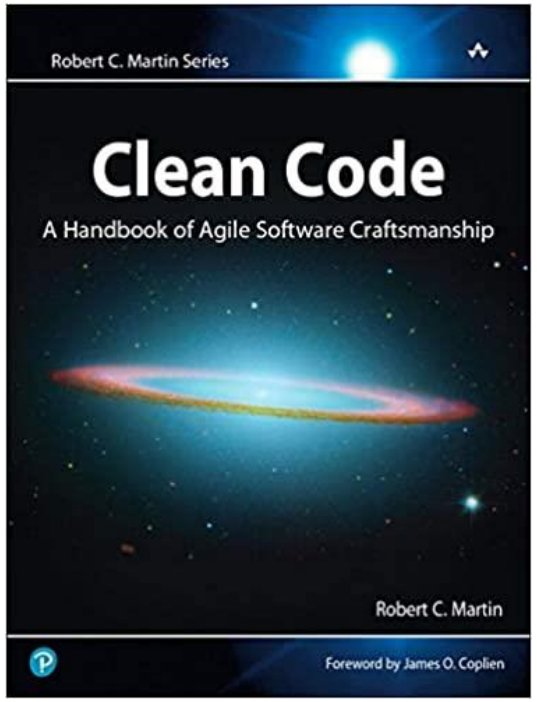


2003 (Robert C. Martin)

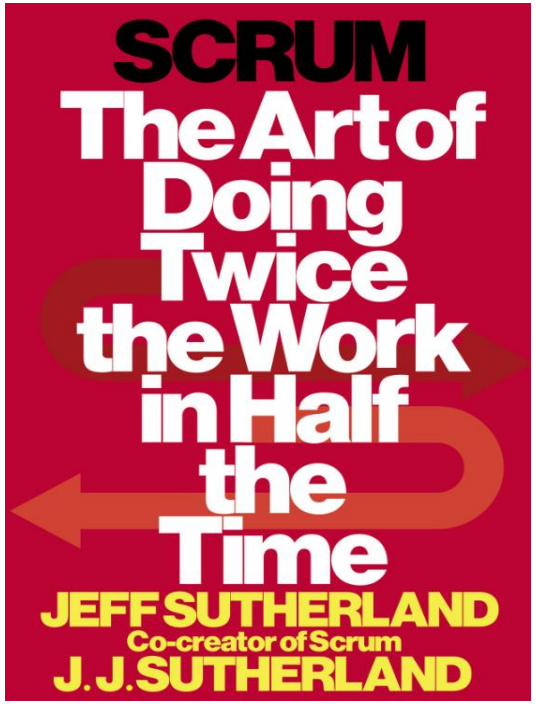


2003 (Eric Evans)

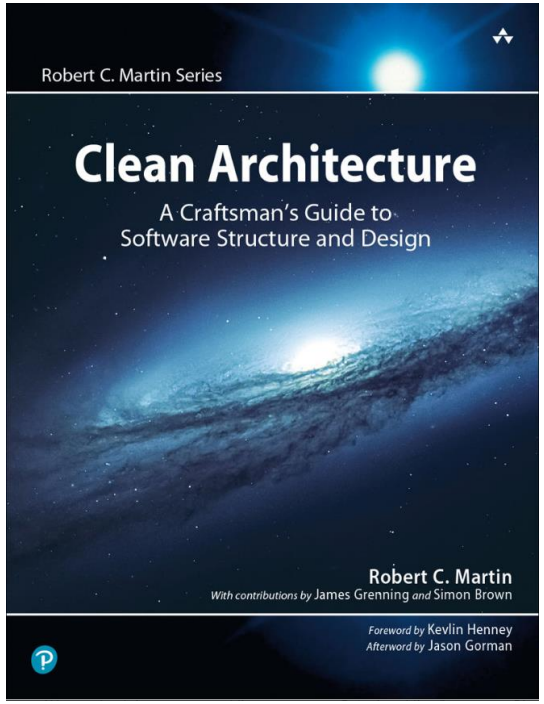
Literatura pós Manifesto Ágil



2008 (Robert C. Martin)



2014 (Jeff Sutherland)



2018 (Robert C. Martin)