Engenharia de Software

Prof. Dr. Erik Aceiro Antonio

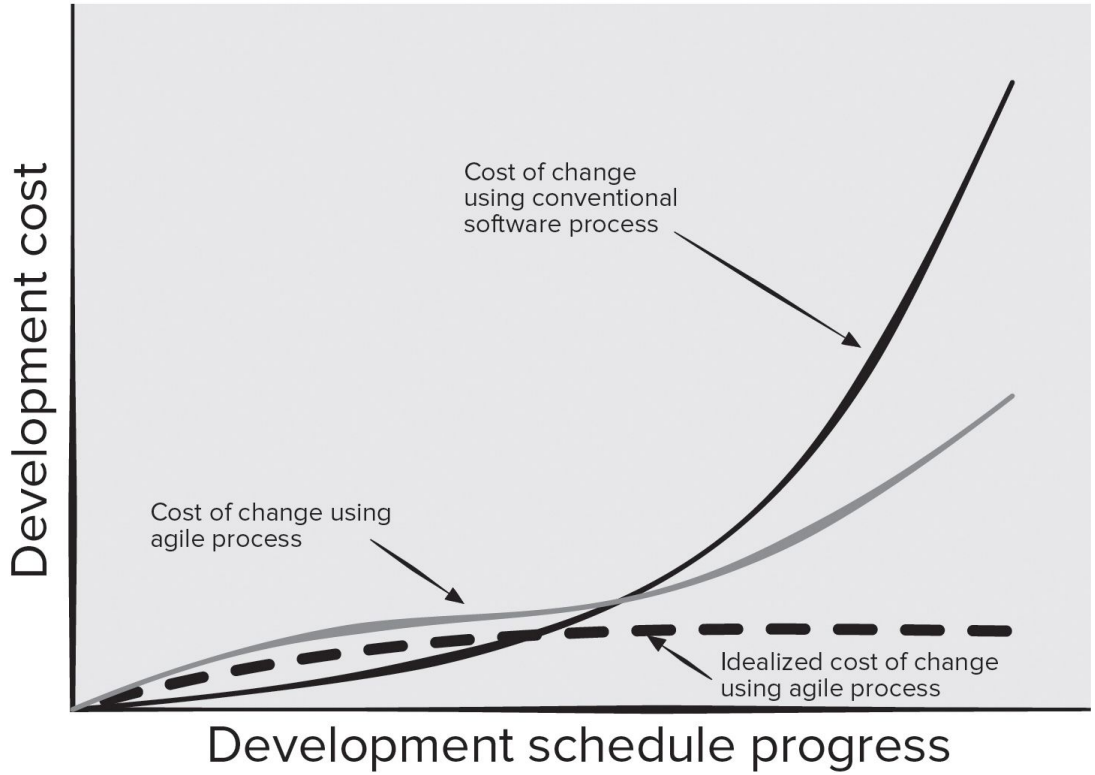
Objetivos

- Agilidade **Pressman (2010, c.3)**
  - O que é agilidade
  - O que é um processo Ágil
  - Princípios Agilidade
  - Scrum
  - XP
  - Kanban Framework
  - DevOps
- Modelo de Processo Recomendado **Pressman (2010, c.4)**
  - Adaptando o processo
  - Características de um Processo Ágil

O que é Agilidade ?

- Efetividade (rápido e adaptativo) resposta a mudança
- Efetividade comunicação com todos os stakeholders
- Aproximar o consumidor no time
- Organizar um time para que o controle do trabalho seja realizado
- Rapido, incremental e *delivery of software*

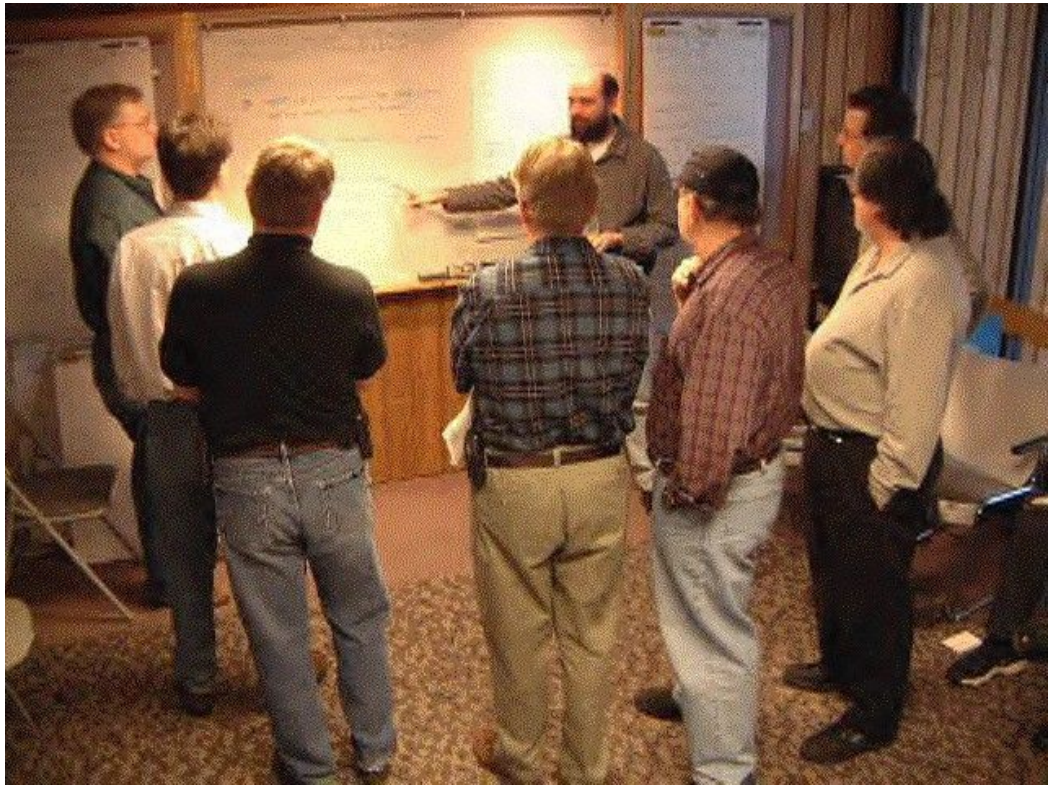
Agilidade & Custo de Alteração



O que é um Processo Ágil

- Driven by customer
  - O que é necessário (cenários)
  - Geralmente utilizando uma notação semi-formal como o BDD (*Behaviour Driven-Design*)
- Reconhece que planos são de curto-prazo
- Desenvolvimento de software guiado iterativamente
- Forte ênfase em atividades de construção
- *Delivers multiple software increments*
- Adaptativo a mudanças de requisitos, projetos e aspectos técnicos

Engenharia de Software clássica | Manifesto Ágil | Engenharia de Software Ágil



Engenharia de Software clássica | Manifesto Ágil | Engenharia de Software Ágil

Princípios de Agilidade

Princípios de Agilidade

Manifesto para Desenvolvimento Ágil de Software

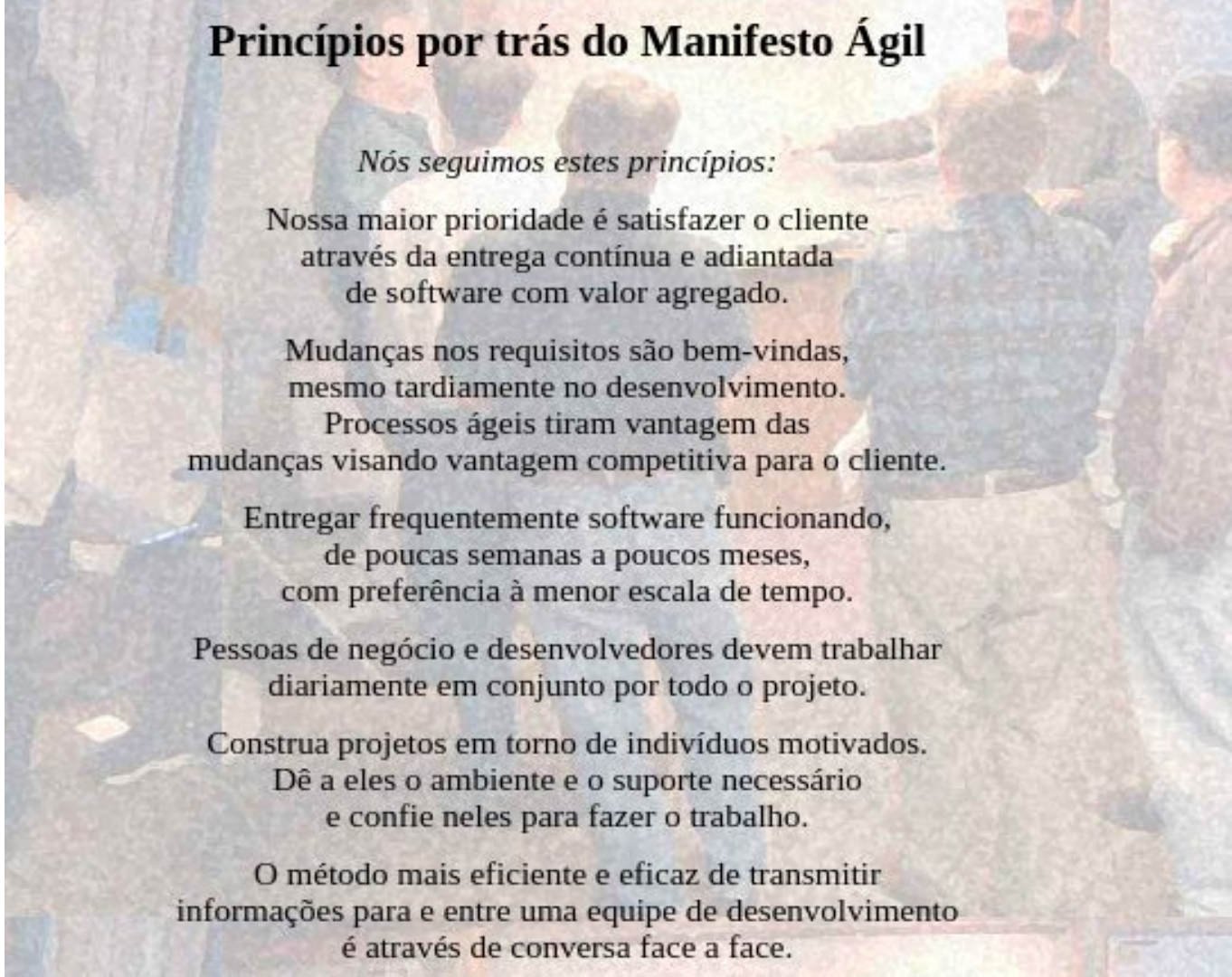
Estamos descobrindo maneiras melhores de desenvolver
software, fazendo-o nós mesmos e ajudando outros a
fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita,
valorizamos mais os itens à esquerda.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

12 Princípios de Agilidade



Princípios por trás do Manifesto Ágil

Nós seguimos estes princípios:

Nossa maior prioridade é satisfazer o cliente
através da entrega contínua e adiantada
de software com valor agregado.

Mudanças nos requisitos são bem-vindas,
mesmo tardiamente no desenvolvimento.
Processos ágeis tiram vantagem das
mudanças visando vantagem competitiva para o cliente.

Entregar frequentemente software funcionando,
de poucas semanas a poucos meses,
com preferência à menor escala de tempo.

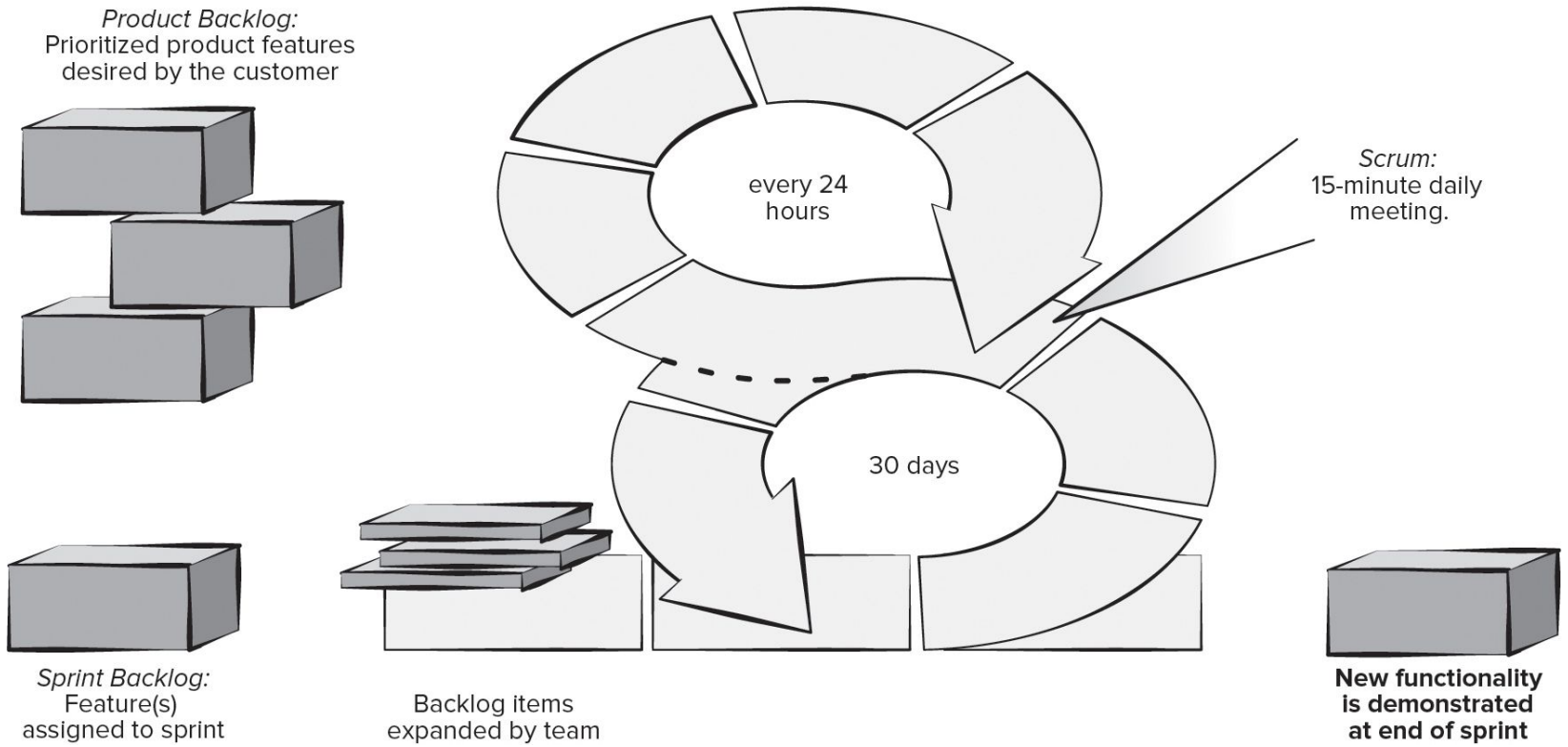
Pessoas de negócio e desenvolvedores devem trabalhar
diariamente em conjunto por todo o projeto.

Construa projetos em torno de indivíduos motivados.
Dê a eles o ambiente e o suporte necessário
e confie neles para fazer o trabalho.

O método mais eficiente e eficaz de transmitir
informações para e entre uma equipe de desenvolvimento
é através de conversa face a face.

12 Princípios de Agilidade

Scrum Framework



Detalhes do Scrum

- Backlog Refinement Meeting
  - Desenvolvedor trabalha com stakeholders para criar Backlog
- Sprint Planning Meeting
  - Backlog particionado em “sprints” derivados do Backlog & próximas sprints
- Daily Scrum Meeting
  - Membros do time sync (geralmente de pé e em 15 min)
- Sprint Review
  - Protótipos como “demos” são apresentados para aprovação ou rejeição
- Sprint Retrospective
  - Após a sprint estar completa, o time avaliam o que precisam ser melhorado

Detalhes do Scrum

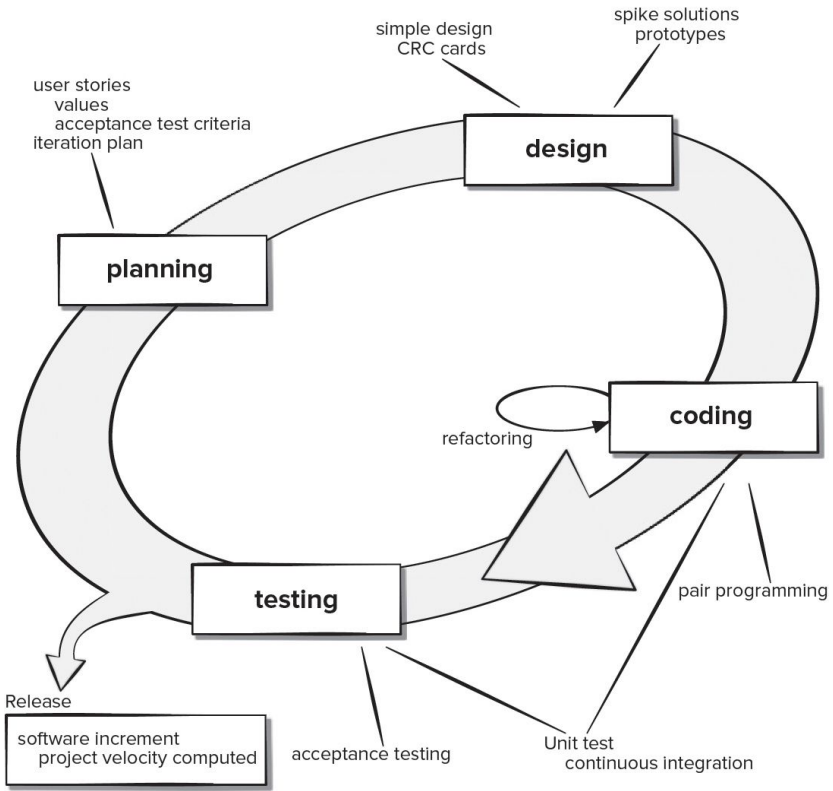
Prós

- Product Owner configura a prioridade
- Times donos sobre as decisões e alterações
- Documentação é *lightweight* (exemplo ADRs)
- Suporta atualizações frequentes

Contras

- Dificuldade para controlar o custo de alterações
- Pode não ser apropriado para times grandes
- Necessita de times especializados

Extreme Programming (XP) Framework



Detalhes do XP

- **XP Planning**
  - Começa com histórias de usuários, estimativas de custo da equipe, histórias agrupadas em incrementos, compromisso feito na data de entrega, velocidade do projeto de computador
- **XP Design**
  - Segue o princípio KIS, incentiva o uso de cartões CRC, protótipos de design e refatoração.
- **XP Coding**
  - Constrói testes de unidade antes de codificar, usa *pair-programming*.
- **XP Testing**
  - Testes unitários executados diariamente, testes de aceitação definidos pelo cliente.

Detalhes do XP

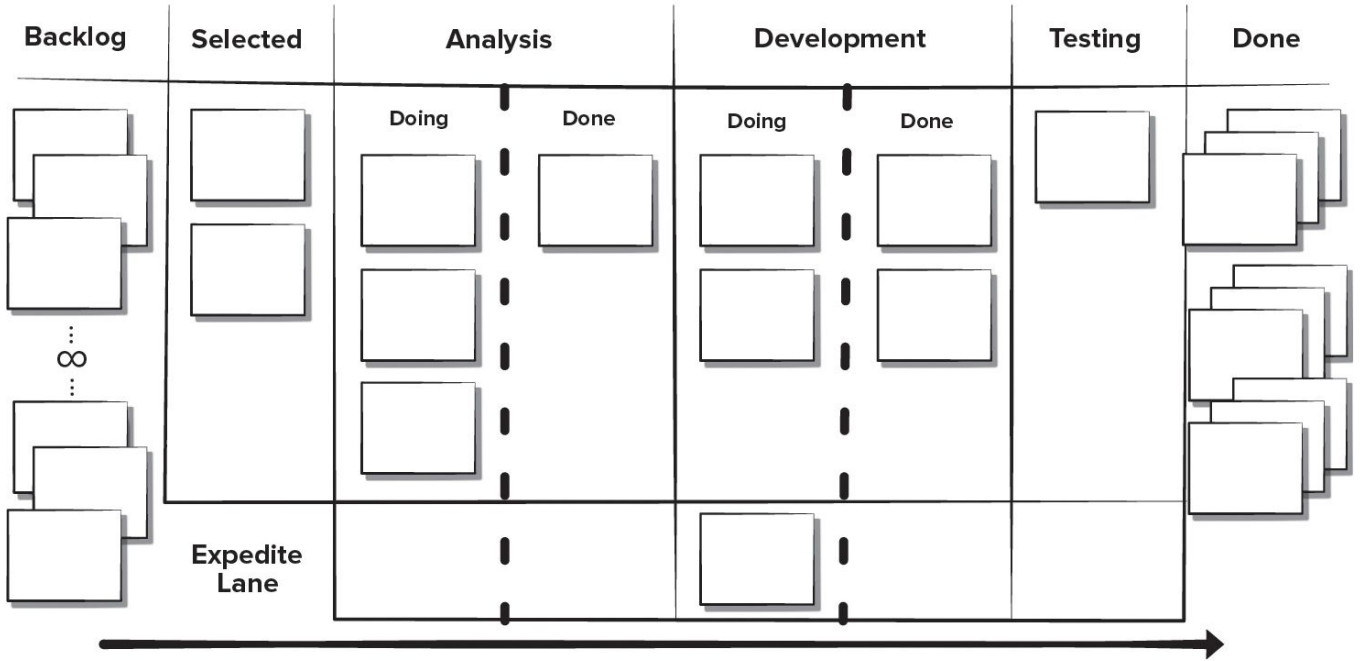
Prós

- Enfatiza o envolvimento do cliente.
- Estabelece planos e horários racionais.
- Alto comprometimento do desenvolvedor com o projeto.
- Redução da probabilidade de rejeição do produto.

Contras

- Tentação de “enviar” um protótipo.
- Requer reuniões frequentes sobre o aumento de custos.
- Permite mudanças excessivas.
- Depende de membros da equipe altamente qualificados.

Kanban Framework



Detalhes do Kanban

- Visualizando o fluxo de trabalho usando um quadro Kanban.
- Limitar a quantidade de trabalho em andamento a qualquer momento.
- Gerenciando o fluxo de trabalho para reduzir o desperdício, entendendo o fluxo de valor atual.
- Tornar explícitas as políticas de processo e os critérios usados para definir “feito”.
- Foco na melhoria contínua, criando ciclos de feedback onde as mudanças são introduzidas.
- Faça mudanças no processo de forma colaborativa e envolva todas as partes interessadas conforme necessário.

Detalhes do Kanban

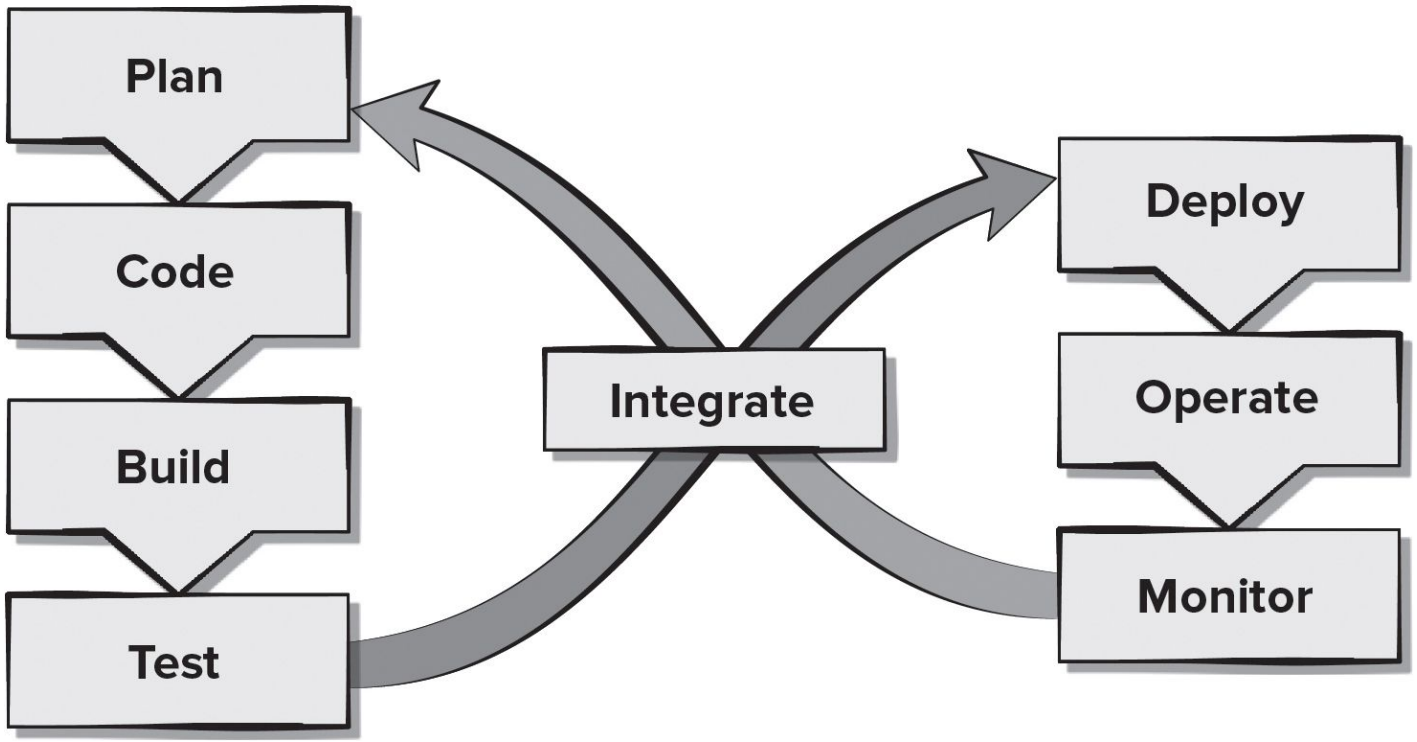
Prós

- Menores requisitos de orçamento e tempo.
- Permite a entrega antecipada do produto.
- Políticas de processo escritas.
- Melhoria contínua de processos.

Contras

- As habilidades de colaboração em equipe determinam o sucesso.
- Uma má análise de negócios pode condenar o projeto.
- A flexibilidade pode fazer com que os desenvolvedores percam o foco.
- Relutância do desenvolvedor em usar a medição.

DevOps



Detalhes do DevOps

- **Continuous development**
  - Software *delivered* (entregue) em múltiplas sprints
- **Continuous testing**.
  - Teste automatizado usado para integração inicial (apriori)
- **Continuous integration.**
  - Pedaços de código com novas funcionalidades adicionadas para o existente código entregue em produção
- **Continuous deployment.**
  - Código integrado com o ambiente de Desenvolvimento & Produção
- **Continuous monitoring.**
  - Times de operações faz o monitoramento proativamente da performance do software em ambiente produtivo

Detalhes do DevOps

Prós

- Tempo reduzido para implantação de código.
- A equipe tem desenvolvedores e equipe de operações.
- A equipe tem a propriedade do projeto de ponta a ponta.
- Monitoramento proativo do produto implantado.

Contras

- Pressão para trabalhar em código antigo e novo.
- Grande dependência de ferramentas automatizadas para ser eficaz.
- A implantação pode afetar o ambiente de produção.
- Requer uma equipe de desenvolvimento especializada.

bitrise Why Bitrise? Learn Community Integrations Pricing Request Demo NEW Log in

Build bett
applicatio

Automate and accelerate wi
Platform, featuring industry

Get Started For Free

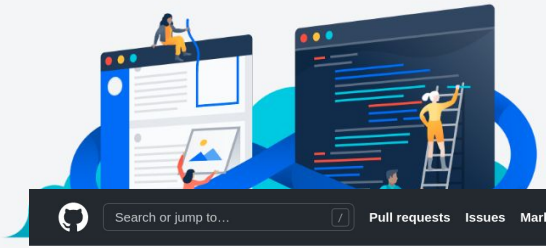
ATLASSIAN Bitbucket

Obtenha grátis

Crie fluxos de trabalho potentes e automatizados

Automatize seu código de teste para produção com o Bitbucket Pipelines, nossa ferramenta de IC/Implementação contínua integrada ao Bitbucket Cloud.

Experimente grátis



Search or jump to… / Pull requests Issues Marketplace Explore

Features Actions Packages Codespaces Copilot Code review Issues Discussions

GitHub Actions

Automate your workflow from idea to production

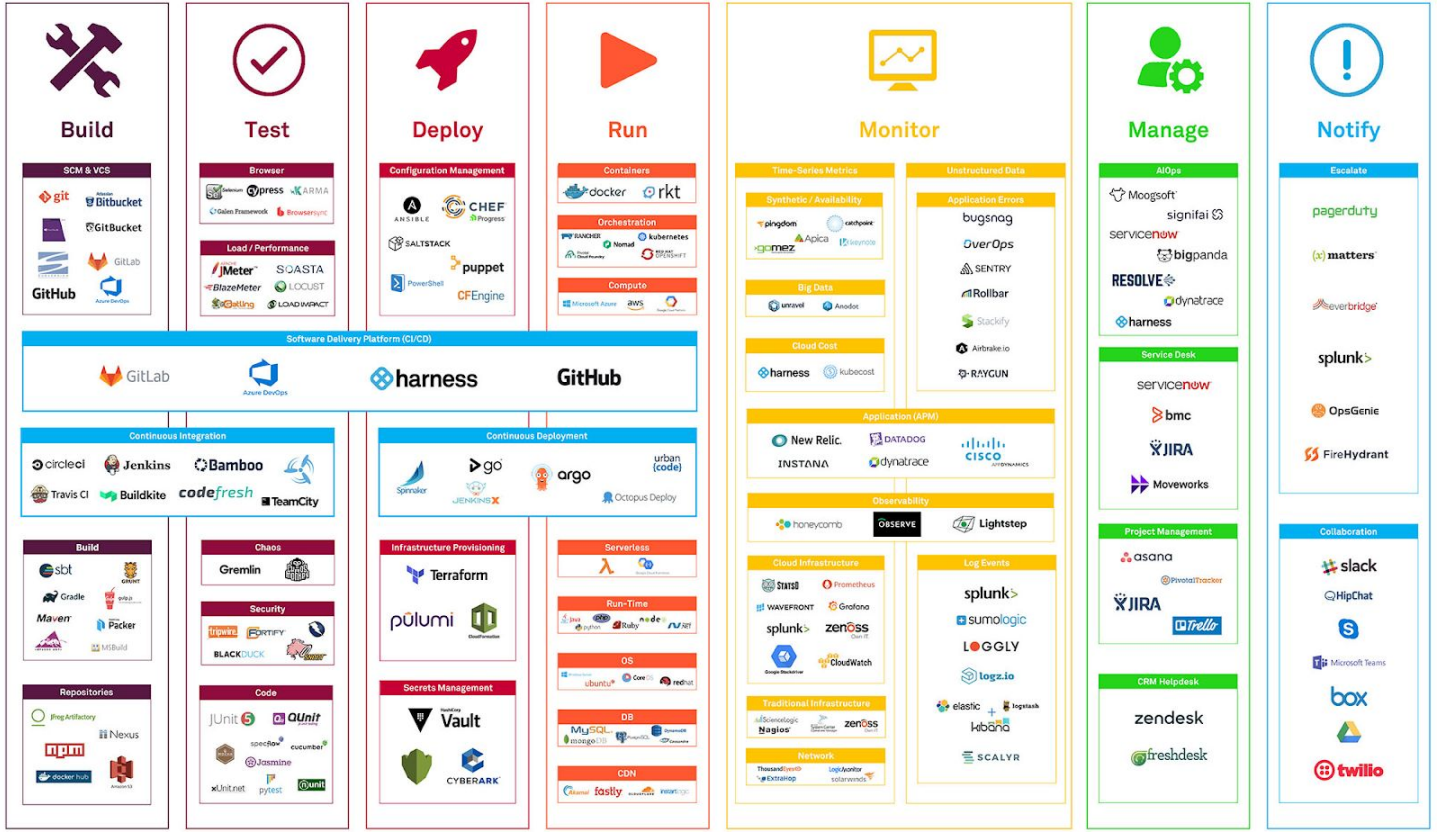
GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

Get started with Actions >

CI/CD integrada ao Bitbucket

Fácil instala

DevOps Tools Ecosystem 2021



harness

The Platform for Software Delivery

www.harness.io

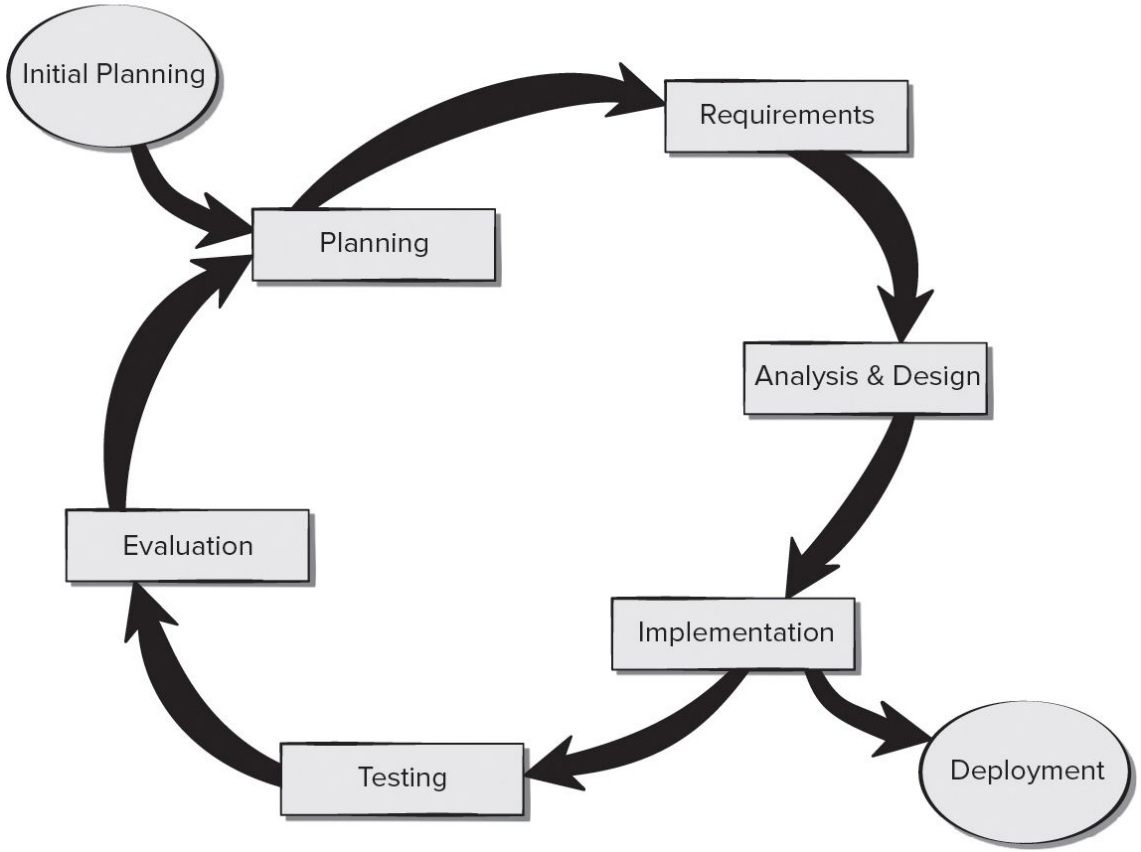
Adaptando o Modelo do Processo

- Todo projeto de software precisa de algum tipo de “road map” ou “processo de software genérico”.
- Cada projeto é diferente, e cada equipe é diferente.
- Nenhuma estrutura de engenharia de software é apropriada para cada produto de software.
- Qualquer road map ou processo genérico deve ser baseado nas melhores práticas do setor.
- Desenvolvedores e partes interessadas adaptam modelos de processos genéricos e os adaptam para se adequarem ao projeto atual, às habilidades dos membros da equipe e às necessidades do usuário.

Princípios para Organizar Projetos de Software

- É um risco usar processos lineares sem um amplo feedback
- Não é possível prever um grande plano de investimentos
- O gerenciamento do projeto deve ser parte do projeto
- Documentos (diagramas, requisitos e UML) devem evoluir com o software
  - Documentação NÃO DEVE ATRASAR A CONSTRUÇÃO DO SOFTWARE
- Envolver os interessados (stakeholders) no início do projeto
- Testadores devem estar envolvidos nas primeiras fases do projeto

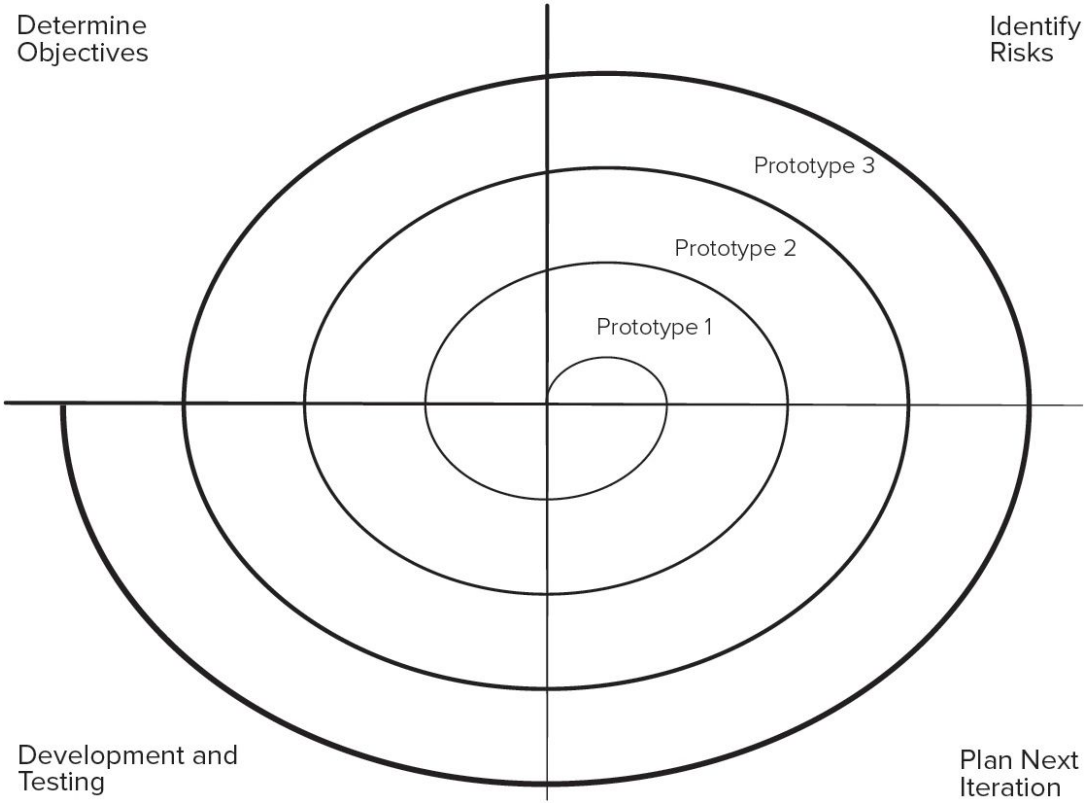
Modelo Incremental Com Prototipação



Características de Modelos de Processos Ágeis

- Não apropriado para projetos de alto-risco ou de missão crítica
- Documentação mínima & regras mínimas
- Envolvimento contínuo de testadores
- Fácil de acomodar mudanças de negócios
- Dependente da interação dos stakeholders
- Fácil de gerenciar
- Early delivery (entregáveis parciais logo no início)
- Gerenciamento informal de riscos
    - não requer normas de certificação ISO, UL ou DO-158 por exemplo
- Construído com processo contínuo

Modelo de Projeto Espiral



Características de Modelos de Processos em Espiral

- Não é apropriado para projetos pequenos e com baixos riscos
- Muitas etapas junto com muitas documentações
- Envolvimento logo cedo de testadores
  - Devem participar como ITT - Independent Test-Team
- Difícil de acomodar novas alterações
- Envolvimento de stakeholders
- Requer um gerenciamento de riscos e coordenação formal
- Bom gerenciamento de riscos

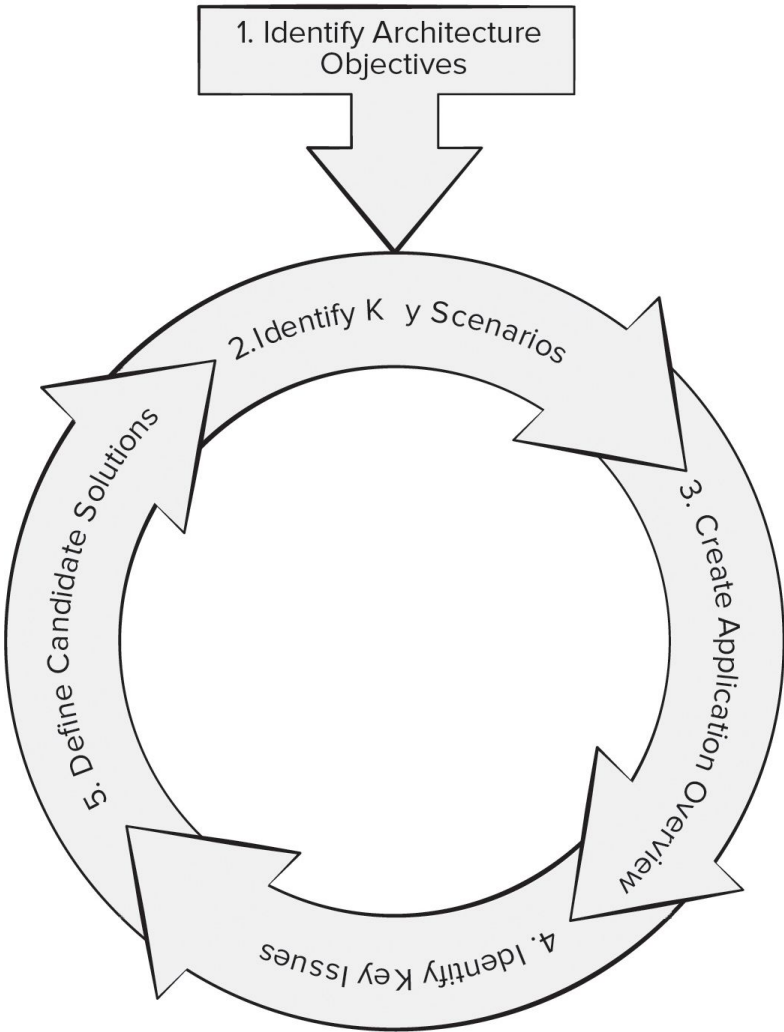
Requisitos Ágeis

- Incentive a participação ativa das partes interessadas, combinando sua disponibilidade e valorizando suas contribuições.
- Use modelos simples (por exemplo, post-its, esboços rápidos, histórias de usuários) para reduzir as barreiras à participação.
- Reserve um tempo para explicar suas técnicas de representação de requisitos antes de usá-las.
- Adote a terminologia das partes interessadas e evite jargões técnicos sempre que possível.
- Use uma abordagem abrangente para obter a visão geral do projeto antes de se prender aos detalhes.

Requisitos Ágeis

- O desenvolvedor e as partes interessadas refinam os requisitos “just in time” à medida que as histórias do usuário estão prontas para serem implementadas.
- Trate a lista de recursos como uma lista priorizada e implemente as histórias de usuários mais importantes primeiro.
- Colabore estreitamente com as partes interessadas e documente os requisitos para que sejam úteis a todos ao criar o próximo protótipo.
- Questionar a necessidade de manter modelos e documentos não mencionados no futuro.
- Garanta o suporte de gerenciamento para as partes interessadas e a disponibilidade de recursos durante a definição dos requisitos.

Protótipo de Design Arquitetural



Elementos de Design Arquitetural Ágil

- Concentre-se nos principais atributos de qualidade e incorpore-os em protótipos à medida que são construídos.
- Lembre-se de que produtos de software bem-sucedidos combinam recursos visíveis para o cliente e a infraestrutura necessária para habilitá-los.
- As arquiteturas ágeis permitem a manutenção e a evolução do código se for dada atenção às decisões de arquitetura e aos problemas de qualidade.
- Gerenciar e sincronizar dependências entre requisitos funcionais e arquiteturais é necessário para garantir que a arquitetura em evolução esteja pronta para incrementos futuros.

Diretrizes para Prototipação

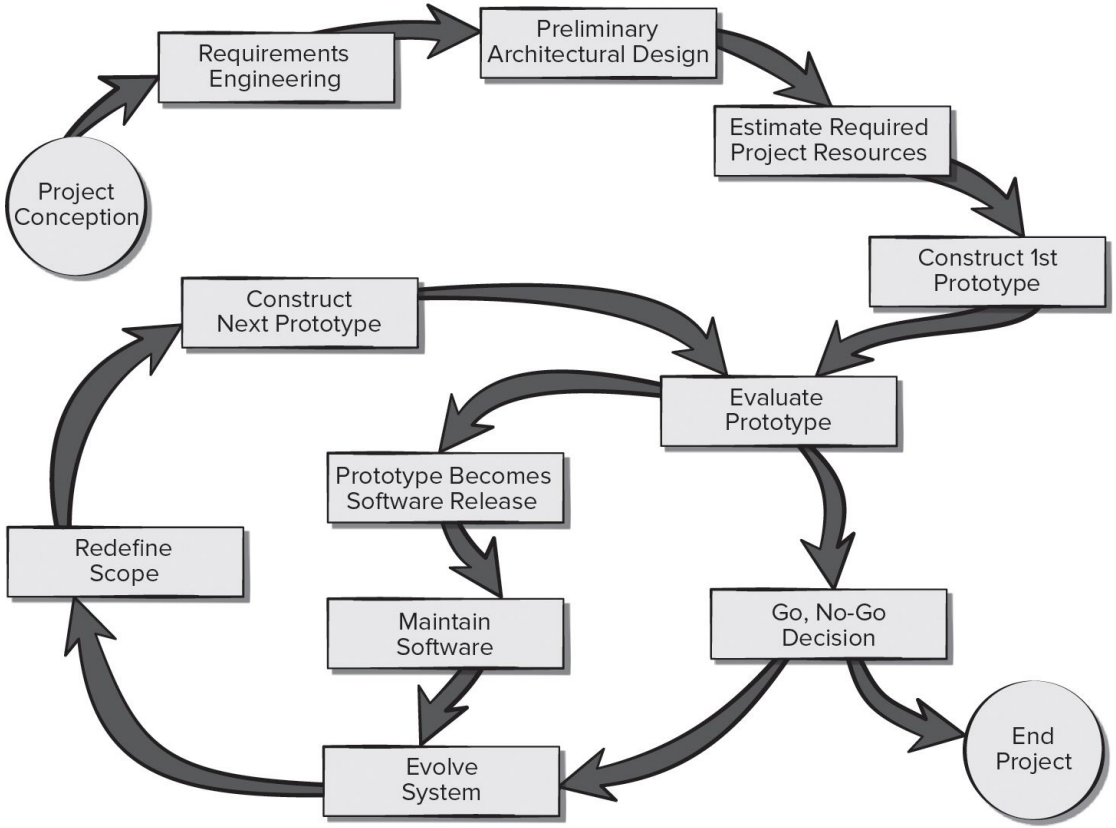
- Transição do protótipo de papel para o design de software.
- Protótipo de uma interface de usuário.
- Crie um protótipo virtual.
- Adicione entrada e saída ao seu protótipo.
- Projete seus algoritmos.
- Teste seu protótipo.
- Protótipo com implantação em mente.

Go/No Go Decision

Avaliação necessária para garantir

1) **Satisfação do usuário**
2) **Riscos envolvidos**
3) **Custo**
4) **Objetivos são revistos**

Processo Evolucionário com Prototipação



Esforço de Manutenção e Distribuição