



Modelos de Processo Ágeis

Engenharia de Software II

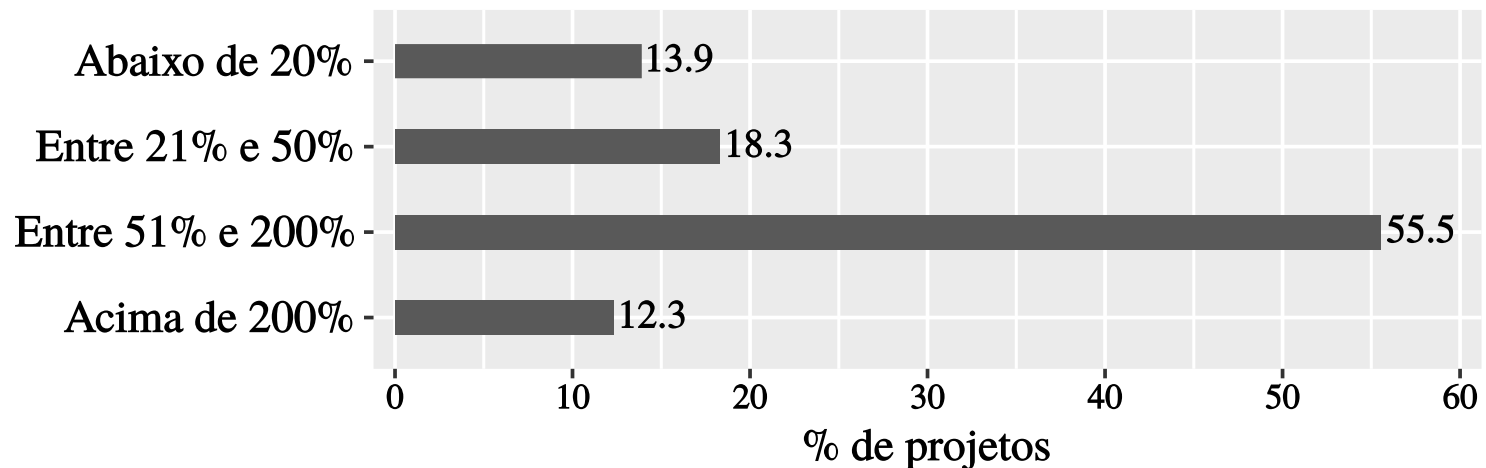
Profa. Andréa Sabedra Bordin

Perspectiva histórica

- Os primeiros processos de desenvolvimento de software — do tipo **Waterfall**, propostos ainda na década de 70 — eram estritamente sequenciais, começando com uma fase de especificação de requisitos até chegar às fases finais de implementação, testes e manutenção do sistema.
- Após cerca de uma década, começou-se a perceber que software é diferente de outros produtos de Engenharia.
 - Essa percepção foi ficando clara devido aos problemas frequentes enfrentados por projetos de software nas décadas de 70 a 90.
 - Por exemplo, os cronogramas e orçamentos desses projetos não eram obedecidos. Não raro, projetos inteiros eram cancelados, após anos de trabalho, sem entregar um sistema funcional para os clientes.

Perspectiva histórica

- Um relatório, conhecido pelo sugestivo nome de CHAOS Report ([link](#)), mostrou que mais de 55% dos projetos estourava os prazos planejados entre 51% e 200%; pelo menos 12% estouravam os prazos acima de 200%, conforme mostra o próximo gráfico.



CHAOS Report (1994): percentual de projetos que estourava seus **prazos** (para cada faixa de estouro).

Perspectiva histórica

- Em fevereiro de 2001, **um grupo de 17 profissionais da indústria** se reuniu na cidade de *Snowbird*, no estado norte-americano de *Utah*, para discutir e propor uma **alternativa** aos processos do tipo *Waterfall* que então predominavam.
- Essencialmente, eles passaram a defender que **software é diferente de produtos tradicionais de Engenharia**.
 - Por isso, software também **demandava um processo de desenvolvimento diferente**.

Grupo de profissionais da indústria



Quais os problemas que eles encontraram?

- Requisitos de um software mudam com frequência, mais do que os requisitos de um computador, de um avião ou de uma ponte.
- Clientes frequentemente não têm uma ideia precisa do que querem. Ou seja, corre-se o risco de projetar por anos um produto que depois de pronto não será mais necessário, ou porque o mundo mudou ou porque os planos e as necessidades dos clientes mudaram.
- A documentação prescrita por processos *Waterfall*, incluindo documentos de requisitos, diagramas, etc. era detalhada, pesada e extensa.
 - Assim, rapidamente se tornavam obsoletas, pois quando os requisitos mudavam os desenvolvedores não propagavam as alterações para a documentação, mas apenas para o código.

Manifesto Ágil

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

- Os profissionais propuseram a **criação de um documento** contendo os **valores, princípios e características** de um processo de desenvolvimento ágil.
- Esse documento foi chamado de **Manifesto Ágil.**

Manifesto ágil

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

*“O movimento ágil **não é anti-metodologia**; de fato, muitos de nós querem restaurar a credibilidade desta palavra. Também queremos restaurar um equilíbrio. Nós abraçamos a **modelagem e documentação**, mas não queremos meramente arquivar alguns diagramas num repositório corporativo empoeirado. Nós planejamos, mas reconhecemos os limites do planejamento em ambientes turbulentos.”*

Manifesto ágil - Valores

- **Indivíduos e interações** são MAIS IMPORTANTES do que processos e ferramentas
- **Software funcionando** é MAIS IMPORTANTE do que a documentação completa e detalhada
- **Colaboração com o cliente** é MAIS IMPORTANTE do que a negociação de contratos
- **Adaptação às mudanças** é MAIS IMPORTANTE do que seguir um plano inicial

12 princípios do desenvolvimento ágil

1. **Satisfação do cliente** → Entrega rápida e contínua de software funcional
2. **Mudanças de requisitos são bem vindas**, inclusive no final da etapa de desenvolvimento
3. Entrega de software funcionando **frequentemente** (semanas ao invés de meses)
4. **Cooperação próxima e diária** entre stakeholders e desenvolvedores
5. Os projetos são construídos em torno de **indivíduos motivados e confiáveis**
6. Uma **conversação presencial** (face-to-face) é a melhor forma de comunicação
7. **Software funcionando** é a melhor métrica de progresso
8. **Desenvolvimento sustentável**, capaz de manter um ritmo de evolução constante
9. Atenção contínua à **excelência técnica** e bom projeto
10. **Simplicidade é essencial** → Focar no importante (Menos às vezes é mais)
11. Melhores arquiteturas, requisitos, projetos emergem de **equipes auto-organizadas** (proativas, comprometidas)
12. Continuamente, a **equipe reflete** sobre como ser mais eficientes, e se auto-ajustam

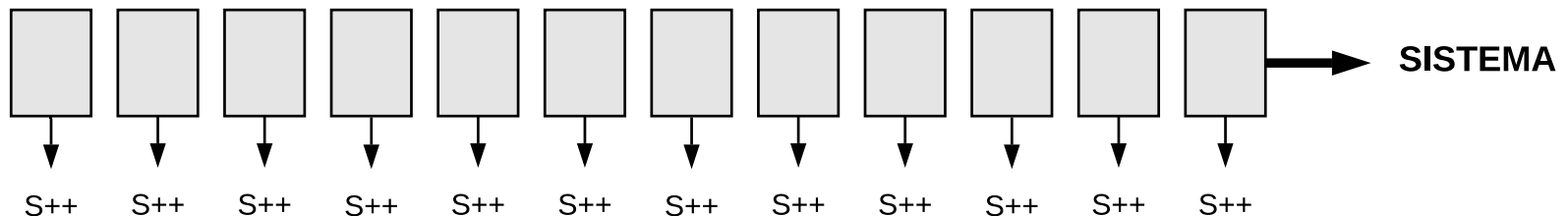


Características de processos ágeis

- A característica **principal** de processos ágeis é a adoção de **ciclos curtos e iterativos de desenvolvimento**, por meio dos quais um sistema é implementado de forma gradativa; começando por aquilo que é mais urgente para o cliente.
- De início, implementa-se uma primeira versão do sistema, com as funcionalidades que segundo o cliente são para **ontem**, isto é, possuem prioridade máxima.
- Em seguida, essa versão é validada pelo cliente. Se ela for aprovada, um novo ciclo — ou **iteração** — inicia-se, com mais algumas funcionalidades, também priorizadas pelos clientes.

Características de processos ágeis

- Normalmente, esses ciclos são **curtos**, com duração de um mês, talvez até um pouco menos.
- Assim, o sistema vai sendo construído de forma incremental, sendo cada incremento devidamente aprovado pelos clientes.
- O desenvolvimento termina quando o cliente decide que todos os requisitos estão implementados.



Outras características

- **Menor ênfase em documentação ou planos detalhados**, pois muitas vezes nem o cliente, nem os Engenheiros de Software têm, no início de um projeto, uma ideia clara dos requisitos que devem ser implementados.
 - Esse entendimento vai surgir ao longo do caminho, à medida que incrementos de produto sejam produzidos e validados. Em outras palavras, o importante em desenvolvimento ágil é conseguir avançar, mesmo em ambientes com informações imperfeitas, parciais e sujeitas a mudanças.



Outras características

- **Inexistência de uma fase dedicada a design** (*big design up front*). Em vez disso, o design também é incremental. Ele evolui à medida que o sistema vai nascendo, ao final de cada iteração.
- **Desenvolvimento em times pequenos**, com cerca de uma dezena de desenvolvedores.
 - Ou, em outras palavras, times que possam ser alimentados com duas pizzas, conforme popularizado pelo CEO da Amazon, Jeff Bezos.
- Ênfase em novas práticas de desenvolvimento (pelo menos, para o início dos anos 2000), como **programação em pares, testes automatizados e integração contínua**.

Outras características

- Os modelos ágeis de desenvolvimento de software seguem uma filosofia diferente da filosofia dos modelos prescritivos.
- Em vez de apresentar uma “receita de bolo”, com atividades e tarefas a serem executadas, **eles focam em valores humanos e sociais.**

Tipos de modelos ágeis de desenvolvimento de software

- FDD – Feature Driven Development
- DSDM – Dynamic Systems Development Method
- Scrum 
- XP – eXtreme Programming 
- Crystal Clear
- ASD – Adaptive Software Development

Mundo Real

- O sucesso e impacto de processos ágeis foi impressionante.
- Hoje, a grande maioria das empresas que desenvolvem software, independente de seu tamanho ou do foco de seu negócio, usam princípios ágeis, em maior ou menor escala.
- Para citar alguns dados, em 2018, o Stack Overflow survey incluiu uma pergunta sobre o método de desenvolvimento mais usado pelos respondentes ([link](#)).
 - Essa pergunta recebeu 57 mil respostas de desenvolvedores profissionais e a grande maioria mencionou métodos ou práticas ágeis como Scrum (63% das respostas), Kanban (36%) e Extreme Programming (16%).
 - Apenas 15% dos participantes marcaram Waterfall como resposta.

Quando **usar** os processos tradicionais?



Quando **não** usar os processos tradicionais?



Bibliografia

- Valente, Marco Tulio. Engenharia de Software Moderna: Princípios e práticas para desenvolvimento de software com produtividade. Belo Horizonte, 2020.
- Waslawick, R. Engenharia de Software. Conceitos e Práticas. Rio de Janeiro: Elsevier, 2013.
- Slides de aula da disciplina Engenharia de Software 2019.01 do ICMC-USP. Profa. Dra. Elisa Yumi Nakagawa e Profa. Dra. Lina Garcés.