Desenvolvimento de Sistemas

Processos de *software*: importância e aplicabilidade, metodologias ágeis, rotina de desenvolvimento com metodologia ágil

Continuando os estudos sobre a utilização das metodologias ágeis, você pode imaginar agora uma produção de *notebooks*. Muitos processos estão envolvidos. Há toda a parte plástica que tem que ser injetada, bem como a estrutura metálica que suporta todos os componentes.

Inclusive, sobre os componentes, existem vários que são fabricados por outros fabricantes que não são necessariamente os fabricantes do próprio *notebook*. A placa-mãe tem muitos componentes de vários fabricantes diferentes.

Finalmente, toda a integração é feita, o sistema operacional é instalado e o produto final deve funcionar perfeitamente, sem nenhum tipo de falha. Porém, nessa integração há milhares de componentes sujeitos a diversas falhas.

Figura 1 – Linha de produção de *notebooks*
Fonte: QNC (2019)

Comparando com *notebooks*, o desenvolvimento de *software* também tem vários processos, incluindo a integração final, ou seja, a aplicação pronta para o usuário final. Obviamente, tais processos são muito mais intelectuais do que físicos quando se trata de *notebooks*. Mesmo assim, todo o processo envolve várias tarefas que vão sendo realizadas por várias pessoas ao mesmo tempo, dependendo da complexidade do projeto.

Hoje ainda se ouve falar sobre grandes programadores que desenvolveram grandes projetos sozinhos ou pelo menos começaram projetos sozinhos. Um grande exemplo é o finlandês Linus Torvalds, que iniciou, sozinho, como um projeto para uso pessoal, o desenvolvimento do famoso Linux. Após divulgar suas intenções em 1991, hoje existe um projeto em pleno funcionamento com milhares de desenvolvedores em todo o mundo trabalhando em conjunto.

A maioria das empresas, nos dias de hoje, trabalha com equipes, o que demonstra a importância dos processos.

Como já visto anteriormente, as metodologias tradicionais como a em cascata (*waterfall*) eram baseadas em processos de engenharia que derivaram de áreas de projeto mecânica, aeronáutica, civil, enfim, áreas nas quais os

projetos eram sempre planejados inicialmente em detalhes, eram sequenciais e muito bem documentados.

Logo os coordenadores começaram a perceber que o desenvolvimento de *software* era diferente e não poderia seguir os mesmos métodos. As aplicações sofrem várias mudanças ao longo do desenvolvimento e, muitas vezes, o dono do projeto não tem uma noção exata do que precisa, o que é bem diferente de um projeto de ponte ou avião, por exemplo.

Então, em 2001, um grupo de profissionais criou um novo método de controle de processos de *software* chamado de “manifesto ágil”. O principal ponto, para este conteúdo, é o uso de **ciclos curtos e iterativos de desenvolvimento**. Dessa maneira, o sistema vai sendo desenvolvido gradativamente. O que é mais urgente tem mais prioridade, o que o cliente mais precisa é o que será trabalhado primeiro hoje.

Assim que o módulo de *software* for validado e aprovado pelo cliente, inicia-se um novo ciclo, ou uma nova **iteração**, de desenvolvimento com a próxima prioridade. Cada iteração costuma durar de duas a quatro semanas em média. Dessa forma, o sistema completo vai sendo criado de modo incremental e termina quando o cliente estiver satisfeito com todos os requisitos, que devem estar em pleno funcionamento.

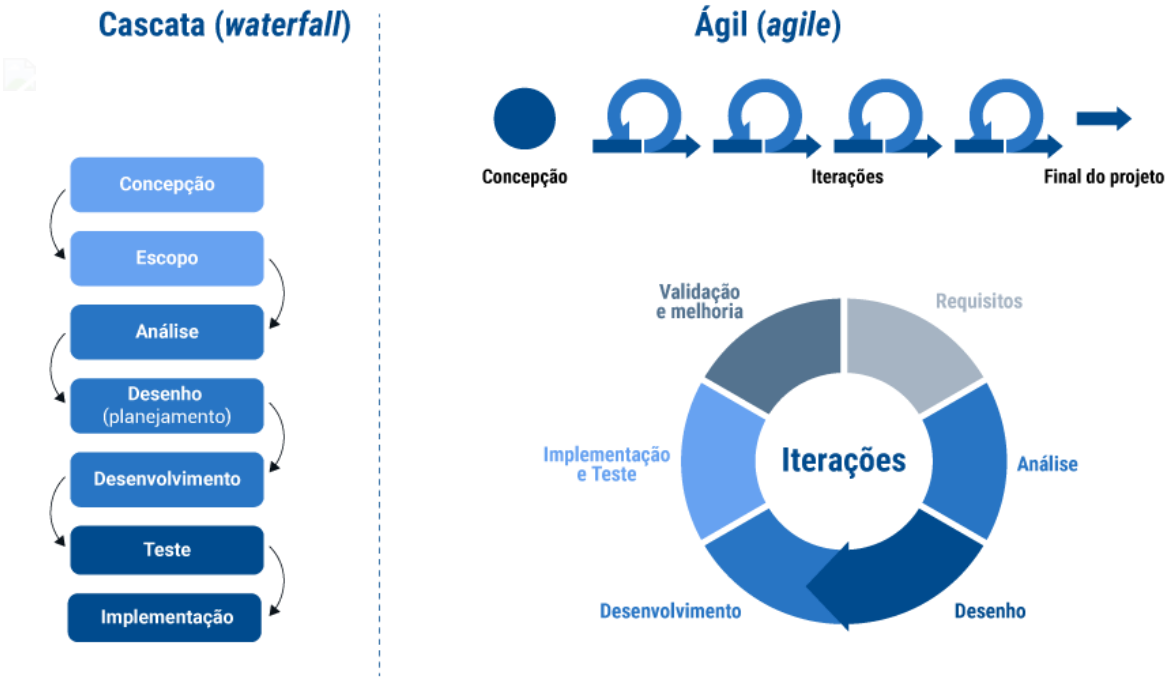

Figura 2 – Comparação de metodologias: cascata x ágil
Fonte: Adaptado de InovaLab (2018)

As metodologias ágeis usam o modelo de iterações que preza a aprendizagem e a melhoria contínua. Tanto o cliente quanto o time de projeto não sabem exatamente o que querem até que tudo esteja concluído e aprovado. O projeto como um todo é dividido em pequenas partes que já podem ser utilizadas pelo cliente logo após serem concluídas. Essas partes costumam ser chamadas de “mínimo produto viável” (MVP).

Com essa técnica, vários problemas são minimizados, pois as iterações curtas, também chamadas de “*sprints*” na metodologia Scrum, promovem um ambiente de produtividade constante. No método Scrum, a equipe planeja apenas as tarefas necessárias para aquela entrega específica (MVP). Esses objetivos de curto prazo têm, assim, menos chances de atrasos e erros. Reuniões diárias e curtas tornam o processo mais transparente para todos, pois as conversas tratam do que foi feito ontem e do que será feito hoje.

Outro destaque é a documentação do projeto. A máxima é a de que **o tempo que a documentação levou para ser gerada não pode custar mais que o seu próprio valor para o projeto**.

Todas essas novas técnicas diminuíram consideravelmente os problemas dos métodos tradicionais. Você conhece o projeto do balanço na árvore que foi desenvolvido usando o método em cascata? Veja:

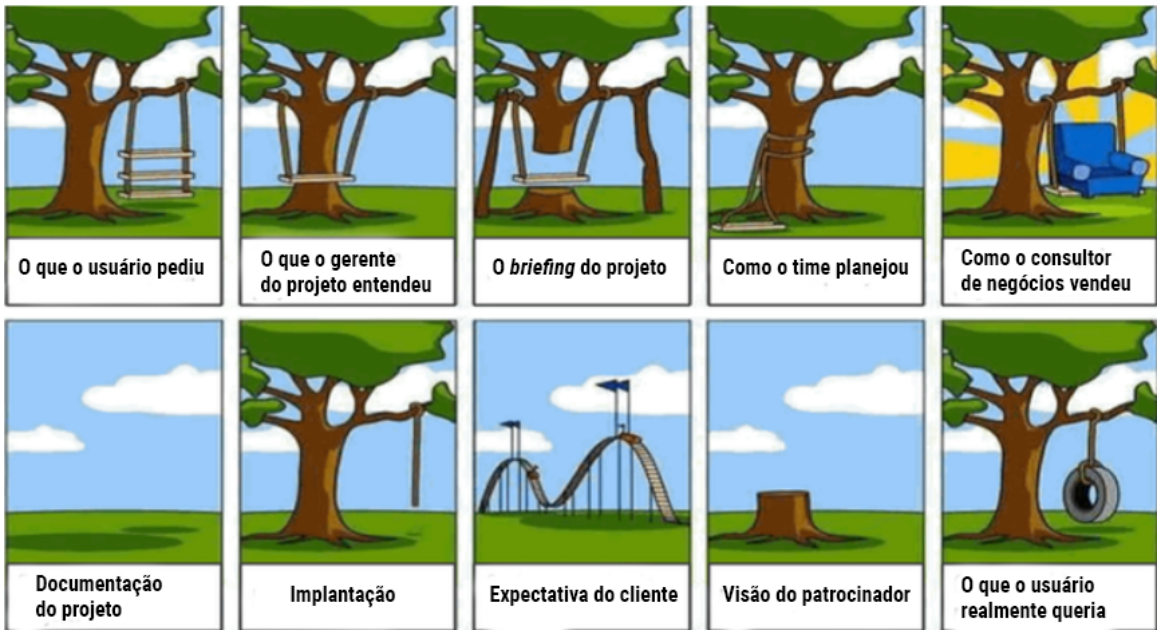

Figura 3 – Projeto do balanço na árvore
Fonte: Adaptado de ClipaTec Informática (2011)

É possível listar alguns problemas tradicionais nos métodos antigos:

- O próprio cliente pode não saber exatamente o que quer.
- As primeiras reuniões com o resumo do produto são vagas ou incompletas.
- O planejamento não corresponde à realidade.
- A implantação não tem consistência.
- Não existe alinhamento com as expectativas do cliente.
- Não existe uma visão do produto final.

Todos os problemas citados podem ser consideravelmente minimizados com a aplicação das iterações rápidas proporcionadas pelos métodos ágeis. Pequenos erros não impedem grandes acertos. Projetos falham, e, nas metodologias tradicionais, tais falhas só serão descobertas depois de muito tempo e a um custo mais elevado. Com o uso de *sprints*, desde o início já é possível avaliar se o projeto é viável para o time e o cliente.

Metodologias ágeis

As metodologias ágeis mais presentes no mercado hoje são estas: XP (*extreme programming*), Scrum e *kanban*. Veja a seguir mais detalhes sobre a rotina de cada uma delas.

Complemente seus estudos sobre o tema relendo o conhecimento **Desenvolvimento ágil: cultura ágil, metodologias, rotinas, *frameworks***, desta unidade curricular.

Rotina de desenvolvimento com metodologia ágil

Afinal, como funciona o método Scrum na prática? Como seria a rotina diária de uma fábrica de *software*, por exemplo? Claro que as várias empresas de desenvolvimento de *software* têm seus próprios métodos, os quais são adaptados às suas necessidades.

Confira a seguir um exemplo ilustrativo de um sistema de biblioteca:

Projeto de *software*

O projeto de *software* compreende as definições do projeto, ou seja, os requisitos do sistema:

- Telas da aplicação (*front-end*)
- Operações e banco de dados (*back-end*)
- Requisitos funcionais e não funcionais

Papéis

É necessário definir os papéis de todos os envolvidos no projeto:

- *Product owner* (dono do produto): representante da biblioteca.
- *Scrum master*: especialista e facilitador.
- Time de desenvolvimento: dois desenvolvedores *front-end* e dois desenvolvedores *back-end*.

Primeira reunião de planejamento (*sprint planning*)

Todos se reúnem pela primeira vez a fim de planejar o desenvolvimento do sistema, ou seja, os artefatos do Scrum. Após a definição do planejamento geral, os eventos (do Scrum) são determinados.

- O *scrum master* guia a reunião, que deve durar no máximo oito horas.
- O *product owner* apresenta o *product backlog* ou a lista das histórias com as prioridades.
- A equipe define o *sprint backlog* com a lista das histórias para a primeira *sprint*:
  ◦ Tela de cadastro de livros e usuários
  ◦ Banco de dados com as tabelas de livros e usuários
- As histórias então são divididas em tarefas:
  ◦ Tela inicial da aplicação (nome, logo, informações etc.) com os *links* (botões) para as outras telas (duas por enquanto)
  ◦ Tela de cadastro de livros com todos os campos necessários
  ◦ Tela de cadastro de usuários com todos os campos necessários
  ◦ Banco de dados com as tabelas completas de livros e usuários
  ◦ Verificação das inserções corretas no banco (se as informações digitadas nas duas telas alimentarão o banco corretamente)
  ◦ Verificação de erros e testes finais
- A equipe vota (*planning poker*) no tempo necessário para a primeira *sprint* e preenche o quadro do Scrum com as tarefas prioritárias ou com o *backlog* da *sprint*. Para facilitar, confira um exemplo de quadro inspirado no *kanban*. Esse quadro normalmente fica fixado na parede e deve estar visível para todo o time:

Sprint backlog	A fazer	Fazendo	Teste		Pronto
			Fazendo	Feito	
Tarefa 1					
Tarefa 2					
Tarefa 3					

Quadro 1 – Modelo de quadro inspirado no *kanban*

Fonte: Senac EAD (2022)

Reuniões diárias (*daily scrum*) durante a *sprint*

As reuniões diárias servem para que todos os membros do time se comuniquem e se atualizem de suas situações nas tarefas.

- Todos devem participar.
- A reunião deve ser breve (15 minutos em média).
- Cada membro do time responderá a estas três perguntas:
  - O que eu fiz ontem?
  - O que eu farei hoje?
  - Algum problema ou algum impedimento para realizar a tarefa?

Figura 5 – Reunião diária (*daily scrum*)
Fonte: Método Ágil (s.d.)

Revisão da *sprint* (*sprint review*)

As tarefas estão prontas, e o time de desenvolvedores fará a demonstração para o *product owner* e para os demais convidados. Se alguma tarefa precisar de ajustes, ela volta a ser comentada na próxima *sprint*. O *product owner* deve aprovar as tarefas para que o processo continue com novas tarefas.

- Todos devem participar.
- O time de desenvolvimento demonstra as tarefas feitas.
- O *product owner* e algum convidado utilizador devem validar as tarefas.
- A apresentação deve durar no máximo quatro horas e ocorre normalmente na sexta-feira.

Retrospectiva da *sprint* (*sprint retrospective*)

A retrospectiva é o último evento de uma *sprint*. Depois do encerramento, inicia-se uma nova *sprint*, com novas tarefas ou com tarefas anteriores que necessitem de ajustes.

- Somente o time *scrum* participa.
- Identificam-se melhorias, dificuldades, sugestões e reclamações.
- O evento dura no máximo três horas e ocorre normalmente na sexta-feira.

Após a retrospectiva, uma nova iteração ou uma nova *sprint* se repete com a reunião de planejamento a fim de definir as próximas tarefas, sempre de acordo com as prioridades apontadas pelo *product owner*. Normalmente, essas reuniões ocorrem nas segundas-feiras para o reinício das *sprints*. No novo ciclo, novas tarefas serão executadas. Se houver algum problema ou alguma parte que o time não concluiu, a(s) tarefa(s) em questão retorna(m) para um retrabalho.

Encerramento

Você viu neste conhecimento a importância da aplicação de metodologias ágeis nos processos de desenvolvimento de *software* e um exemplo prático de como elas funcionariam na rotina de uma empresa.

Vive-se em um mundo em constante evolução e transformação. Para sobreviver e prosperar, é preciso então estar aberto à adaptação e ao entendimento das novas tecnologias. Os modelos de negócios também estão cada vez mais rompendo todas as barreiras dos métodos tradicionais de gestão mais estáticos e não adaptativos. Assim, a adoção de métodos ágeis está mais alinhada à nova realidade atual e mais adequada aos desafios do futuro.