Engenharia de Software

A Engenharia de Software é uma disciplina que estuda todo o processo envolvido no desenvolvimento de software, usando métodos sistemáticos, disciplina e rigor em todas as fases do ciclo de vida do software. É uma área importante, pois as pessoas e a sociedade estão cada vez mais dependentes de software. Por muito tempo, o desenvolvimento de sistemas foi realizado sem atenção a processos, metodologias e estruturas organizacionais, o que resultava em software com pouca qualidade e muitos erros.

Ao longo do tempo, o desenvolvimento de sistemas passou por mudanças significativas não apenas no aspecto tecnológico, mas também na forma como as empresas organizam e estruturam seu trabalho. A abordagem tradicional de desenvolvimento de sistemas, que empregava o modelo em cascata (criado em 1970), foi a primeira a ser criada. Nessa abordagem, as etapas de desenvolvimento eram executadas de forma sequencial, sem possibilidade de retorno de uma etapa posterior para uma etapa anterior.

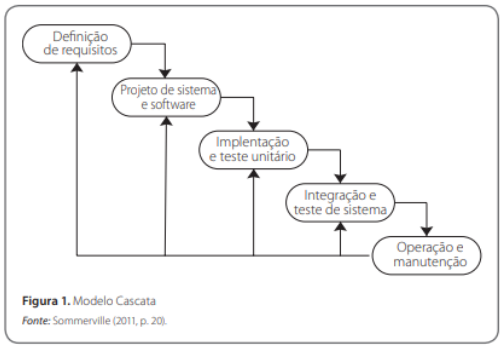

Figura 1. Modelo Cascata
Fonte: Sommerville (2011, p. 20).

Posteriormente, falou-se em desenvolvimento iterativo e incremental. Nesse modelo, implementa-se pequenas partes entregáveis do software para que o cliente tenha um feedback mais rápido sobre o produto que está sendo desenvolvimento.

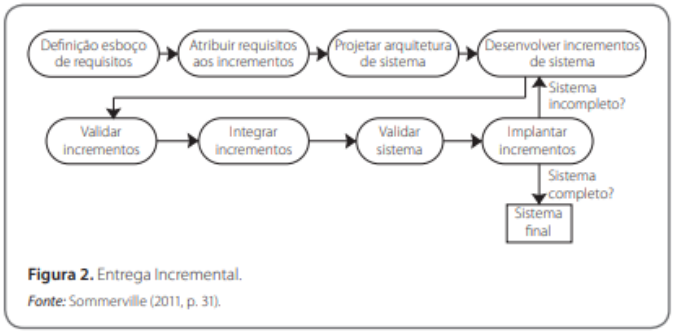

Figura 2. Entrega Incremental.
Fonte: Sommerville (2011, p. 31).

A abordagem em espiral implementa os sistemas baseado no conceito de maior necessidade. Ela entrega o sistema em versões. O fluxo que esse modelo propõe é em formato de espiral. A volta mais interna pode se preocupar com a viabilidade do sistema, o próximo ciclo com a definição de requisitos, o seguinte com o projeto do sistema, e assim por diante.

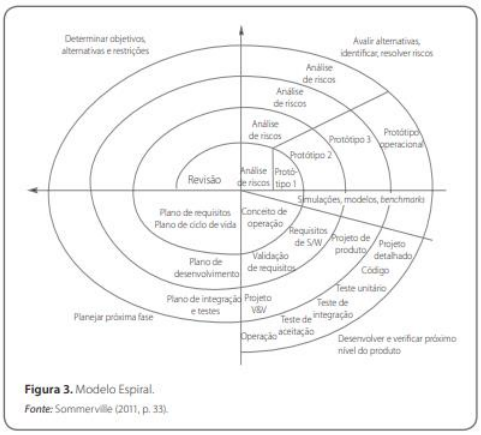

Figura 3. Modelo Espiral.
Fonte: Sommerville (2011, p. 33).

O **Modelo V** virou um padrão da indústria de software depois de 1980 e, após o surgimento da Engenharia de Sistemas, tornou-se um conceito padrão em todos os domínios da indústria. Foi baseado no modelo cascata, porém a principal diferença é a integração entre as etapas.

O modelo de **prototipagem** descreve uma abordagem que tenta satisfazer as necessidades do usuário focalizando a interface do usuário. É diferente do modelo descrito, onde só existe implementação e entrega ao final do projeto.

Fases do software

1. **Planejamento**: processo fundamental para o entendimento de por que um sistema de informações deve ser construído. Etapas:
   - Iniciação do projeto
     - Solicitação de Sistema
     - Análise de Viabilidade
       > Técnica (Podemos construí-lo?)
       > Econômica (Ele agregará valor?)
       > Organizacional (Se o construirmos, ele será usado?)
   - Gerenciamento do projeto
     - Plano de projeto
2. **Análise**:
   - Estratégia de análise
   - Levantamento de requisitos
     - Modelos de análise do negócio
   - Proposta de sistema (Conjunto resumido de todos os documentos gerados)
3. **Projeto**:
   - Estratégia de projeto
   - Projeto de arquitetura e interface
   - Especificações de bancos de dados e de arquivos
   - Projeto do programa

   Produto entregue: especificação do sistema (design de arquitetura, design de interface,

especificações de banco de dados e de arquivos e design do programa

4. **Implementação**: o sistema é realmente construído (internamente e externamente) ou comprado, no caso de um projeto de software pronto oferecido no comércio.
   - Construção do sistema
     - Testes
   - Instalação
     -Plano de Treinamento
   - Plano de suporte (incidentes e requisições)

Modelos Tradicionais X Métodos Ágeis

Manifesto ágil - Foi um conjunto de princípios e valores criados para ajudar a desenvolver softwares de maior qualidade. Nesse manifesto, foram unidas ideias para tornar o processo de desenvolvimento menos burocrático e mais colaborativo. Esse documento estabelece quatro valores fundamentais, sendo eles: indivíduos e interações mais que processos e ferramentas; software em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; e responder a mudanças mais que seguir um plano.

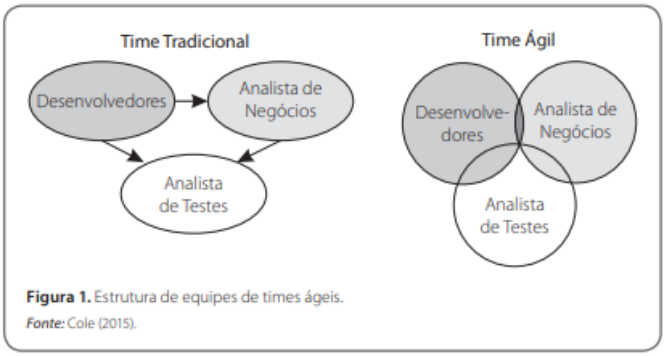

Figura 1. Estrutura de equipes de times ágeis.
Fonte: Cole (2015).

Métodos ágeis (cliente como centro do processo):

- Kanban: O Kanban lhe ajuda a assimilar e controlar o progresso de suas tarefas de forma visual. É, normalmente, utilizado um quadro branco com alguns pequenos papéis colados, que representam as suas tarefas.
- Scrum: O Scrum enfatiza o uso de um conjunto de padrões de software que se mostrou eficaz para projetos com cronogramas apertados, requisitos mutáveis e aspectos críticos de negócio. Cada padrão de processo define um conjunto de tarefas de desenvolvimento e permite à equipe Scrum construir um processo que se adapte às necessidades do projeto.

  Backlog: É uma lista com prioridades dos requisitos ou funcionalidades do projeto que fornecem valor comercial ao cliente.

  Sprints: Consiste em unidades de trabalho solicitadas para atingir um requisito estabelecido no registro de trabalho e que precisa ser ajustado dentro de um prazo já fechado.
- XP (Extreme Programming): é o processo ágil mais amplamente utilizado. Organizada em quatro atividades metodológicas – planejamento, projeto, codificação e testes – a XP sugere várias técnicas poderosas e inovadoras que possibilitam a uma equipe ágil criar versões de software com frequência, propiciando recursos e funcionalidades descritos previamente e priorizados pelos envolvidos.
- Processo Unificado Ágil (AUP): O Processo Unificado Ágil (AUP) adota a filosofia do "serial para o que é amplo" e "iterativa para o que é particular" para o desenvolvimento de software.

Engenharia de requisitos

Elicitação de requisitos -> É o momento em que a equipe técnica precisa compreender o que deve ser feito. O processo de elicitação exige uma intensa comunicação com os stakeholders do projeto. É no processo de comunicação que o analista de requisitos identifica o que realmente é imprescindível para o desenvolvimento do produto e não perde tempo detalhando o que não é importante ou que não precisa de detalhamento naquele momento. A comunicação ineficaz pode levar a requisitos incompletos e incorretos.

Análise de requisitos -> é a etapa na qual nos concentramos em aprofundar o entendimento acerca dos requisitos, buscando possíveis conflitos que podem ser advindos das diferentes visões que os stakeholders possam ter sobre o requisito e sua prioridade no projeto de desenvolvimento.

Especificação de requisitos -> é a etapa dedicada a representar os requisitos de uma forma que eles possam perdurar ao longo do tempo e possam ser verificados e validados posteriormente. Isso pode implicar em formatos diferentes de especificação que envolvem textos, diagramas e tabelas.

Validação de requisitos -> A especificação dos requisitos precisa ser validada entre os stakeholders e a equipe de desenvolvimento para garantir que existe uma compreensão correta e comum sobre os requisitos e que a equipe de desenvolvimento possui as condições de implementar um produto que irá satisfazer as necessidades do negócio. A validação pode ser realizada

por meio de uma revisão sobre as especificações, que é feita por revisores designados para tal, com o apoio de um checklist, por exemplo.

Gerenciamento de requisitos -> estabelecimento de formas de rastrear os requisitos e facilitar a análise do impacto de uma solicitação de mudança para a tomada de decisão.

Matriz de rastreabilidade: A matriz de rastreabilidade é um artefato vivo, que nasce logo que os primeiros requisitos surgem e que vai sendo atualizada ao longo de todo o ciclo do produto de software. Ela documenta o relacionamento entre os requisitos e todos os demais elementos do produto. Para que ela possa ser fácil de manter, é preciso usar ferramentas automatizadas. * Não é viável construir matrizes que precisem ser mantidas de forma manual, usando planilhas ou outro tipo de tabela

Um requisito sistêmico é uma declaração/descrição do que o sistema deve fazer ou de quais características ele precisa possuir.

- Várias técnicas e ferramentas;
- -Para projetos ágeis: User Stories é uma boa opção;
- Entregas da etapa de análise ou design inicial: Definição de requisitos, Casos de uso, Modelo de processos e Modelos de dados;
- Determinar os requisitos é o aspecto isolado mais crítico de todo o ciclo de vida;
- Mudanças nas etapas seguinte são bem mais trabalhosas;

Determinação dos requisitos

- Requisitos gerados ao longo do Ciclo de Vida;
  -Necessidades do negócio (requisitos do negócio);
  -O que os usuários precisam fazer (requisitos dos usuários);
  -O que o software deve fazer (requisitos funcionais);
  -Características que o sistema deve ter (requisitos não funcionais);
  -Como o sistema deve ser construído (requisitos do sistema);

Requisito Funcional

- Suportar as necessidades dos usuários;
- Processo que o sistema realiza como parte do suporte fornecido a uma tarefa do usuário e/ou à informação que o sistema precisa fornecer quando o usuário estiver realizando uma tarefa;
- Orientado por processos:
  -Um processo que o sistema deve realizar, fazer, permitir, verificar, etc.
- Orientado por informações:
  -Informações que o sistema deve conter, manter, conversar, incluir, etc.

Requisito Não Funcional

- Propriedades comportamentais;
- Desempenho, usabilidade, segurança, etc.;
- Atributos de qualidade, restrições de design e implementação e interfaces externas que um produto deve possuir;
- Descrevem características relativas ao sistema, porém não descrevem processos ou informações do negócio;
- Na fase de Design, dão embasamento a tomada de decisões sobre a interface com o usuário, o hardware, o software a arquitetura básica do sistema;
- Operacional:
  -O ambiente físico e técnico no qual o sistema vai trabalhar (Web, Mobile, etc.);
- Desempenho:
  -Velocidade, a capacidade e a confiabilidade do sistema;
- Segurança:
  - Quem tem acesso autorizado ao sistema e sob que circunstâncias;
  - Cultural e Político:
  >Fatores culturais e políticos e exigências legais que afetam o sistema;

Regras de Negócio

- Regras referentes ao comportamento dos negócios;
- Maneira padronizada de agir e reagir perante as situações cotidianas relacionas ao negócio em que atuam.
- Exemplos:
  -Não é permitido excluir lançamentos pagos;
  -Não é permitido cadastrar clientes com idade inferior a 18 anos;
  -Somente usuários com perfil de administrador podem excluir uma venda;

Ferramentas de coleta

- Entrevistas;
- Questionários;
- Observação;

- Desenvolvimento conjunto de aplicações (JAD);
- Análise de documentos;
- Complementares, ajudam a clarificar (do modo visual) as ideias:
  -Casos de uso;
  -Modelos de Processos;
  -Modelos de dados;