UNICARIOCA
Centro Universitário

Tema 1 - História, desafios atuais, áreas de atuação, evolução dos processos de desenvolvimento de software.

UNICARIOCA
Centro Universitário

CONTEXTO HISTÓRICO

Por muitos anos a Engenharia de Software foi utilizada com o objetivo de criar software de qualidade dentro dos custos e dos prazos estimados pelo cliente, evitando desperdícios de tempo, esforços, direções erradas e atrasos. A criação de software foi subestimada e realizada sem nenhuma metodologia, gerando erros em sistemas, como problemas de cálculos e perdas financeiras e de tempo. Nesse período, podemos dizer que houve a crise do software. Com isso, em 1967, a Organização do Tratado do Atlântico Norte (OTAN) designou o termo Engenharia de Software para adequar o processo de desenvolvimento de software com metodologias já utilizadas em outras Engenharias. Uma série de metodologias e técnicas passaram a ser utilizadas antes, durante e depois da criação de software. Dados históricos apontam que houve uma diminuição brutal nos problemas em software após a adoção dessas metodologias, fazendo com que a indústria de software pudesse entregar sistemas com maior qualidade, em menos tempo e com custos reduzidos de manutenção.

Histórico e conceitos fundamentais da Engenharia de Software

A Engenharia de Software é uma disciplina da Engenharia, mais especificamente da Ciência da Computação, que estuda todos os processos envolvidos no desenvolvimento de software, uma atividade complexa que envolve a realização conjunta de diversas atividades distintas, as quais exigem habilidades multidisciplinares e, por consequência, trabalho colaborativo de um grande grupo de profissionais.

A Engenharia de Software é uma área de grande importância, uma vez que as pessoas e a sociedade como um todo estão a cada dia mais dependentes de software. Por isso, faz-se necessário que seja produzido software mais confiável e de forma mais rápida a cada dia. Além disso, para as empresas desenvolvedoras de sistemas, geralmente é mais barato, a longo prazo, usar métodos e técnicas da Engenharia de Software para o desenvolvimento de sistemas do que desenvolver os sistemas sem documentação e estruturação, uma vez que, desta forma, é desestruturada e dificultada a manutenção do software.

Contudo, esta disciplina nem sempre foi alvo de atenção dos profissionais de Tecnologia da Informação. Por muito tempo, o desenvolvimento de sistemas foi realizado sem atenção a processos, metodologias e estruturas organizacionais no que diz respeito a tarefas, atividades e responsabilidades. Essa falta de controle sobre os processos fez com que, muitas vezes, o software fosse entregue aos clientes sem a devida qualidade e com grande número de erros, como problemas de cálculos e perdas financeiras e de tempo. Nesse período, podemos dizer que houve a crise do software. Com isso, em 1967, a OTAN designou o termo Engenharia de Software para adequar o processo de desenvolvimento de software com metodologias já utilizadas em outras engenharias.

A partir desse momento, os profissionais e as empresas de Tecnologia da Informação passaram a preocupar-se mais com os diversos setores que envolvem o desenvolvimento de sistemas, como análise de requisitos, análise de sistemas, desenvolvimento, testes e implantação. Neste contexto, derivaram diversas metodologias, métodos e processos para auxiliar e guiar o trabalho de cada um desses segmentos.

Os principais fundamentos científicos para a ES envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliar e garantir sua qualidade.

A Importância da Engenharia de Software

A Engenharia de Software é uma disciplina de Engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até a sua manutenção, quando o sistema já está sendo usado. Neste contexto, é clara a importância fundamental da Engenharia de Software, uma vez que, se o processo de desenvolvimento de sistemas envolve diversas atividades distintas e a Engenharia de Software é a disciplina que preocupa-se em estudar e monitorar o bom andamento de todas essas atividades e a integração entre elas, é trivial notar que é baseado na Engenharia de Software o sucesso de um projeto no que tange a sua organização.

Engenharia de Software envolve planejamento. **Planejamento** diz respeito também a pessoas e cronograma de trabalho. **Pessoas** porque divide as responsabilidades, de forma individual ou coletiva. **Cronograma** porque conforme o planejamento é que os gestores têm a possibilidade de mensurar o tempo necessário para o desenvolvimento de cada projeto. *Engenharia de Software envolve também a preocupação com a qualidade do produto*. Qualidade, neste contexto, não se refere apenas à entrega de produtos em funcionamento, mas também ao atendimento das necessidades do cliente, e, por isso, a área da Engenharia de Software, que trata de engenharia de requisitos ou análise de requisitos, tem importância fundamental, na medida em que é por meio dela que as equipes de desenvolvimento recebem a expectativa do cliente sobre o produto que está sendo desenvolvido para buscar atendê-la.

Além disso, na fase de desenvolvimento da programação em si, a Engenharia de Software se faz presente, uma vez que a escolha do processo ideal irá influenciar diretamente no trabalho cotidiano de todos os envolvidos, incluindo a programação. Esse fator ganha relevância ainda maior em times que trabalham em horários distintos ou locais geograficamente distribuídos. Uma vez entregue o software para o cliente, a Engenharia de Software tem sua importância revelada no momento de realizar a manutenção nesse software. Isto porque, se o sistema tiver sido corretamente planejado, o código do sistema tende a estar mais limpo e com menos defeitos. Isso irá causar menos manutenção e facilitar as manutenções que precisam ser realizadas.

OS DESAFIOS ATUAIS

A Associação Brasileira das Empresas de Software (ABES), em parceria com o International Data Corporation (IDC), divulgou o "**Mercado Brasileiro de Software: Panorama e Tendências 2024**".

O Instituto de engenheiros eletricistas e eletrônicos (IEEE) define engenharia de software como "*a utilização de uma metodologia sistemática, disciplinada e mensurável no desenvolvimento, operação e manutenção de software*".

Esses procedimentos visam **assegurar a qualidade de um produto de software através da estruturação do projeto**, promovendo maior eficiência e oferecendo os recursos necessários para a produção de software de alta qualidade e desempenho.

A engenharia de software possui um modelo em camadas que define uma estrutura com foco na qualidade.

Essas camadas são **qualidade**, **processos**, **métodos** e **ferramentas**, como apresentado na figura abaixo.

UNICARIOCA
Centro Universitário

Camadas da engenharia de software

As **ferramentas** dão suporte às metodologias e processos adotados para a engenharia de software.

Os **métodos** definem como um software deverá ser produzido.

Os **processos** correspondem a um conjunto de atividades para a produção de um software.

O foco na **qualidade** deve ser fundamento para todas as camadas.

UNICARIOCA
Centro Universitário

As camadas da engenharia de software são fundamentadas sobre a base do foco na qualidade. Cada camada define, de forma geral, um conjunto de processos, métodos e ferramentas que podem ser adaptados para diferentes contextos de projetos de software.

As principais camadas da Engenharia de Software:

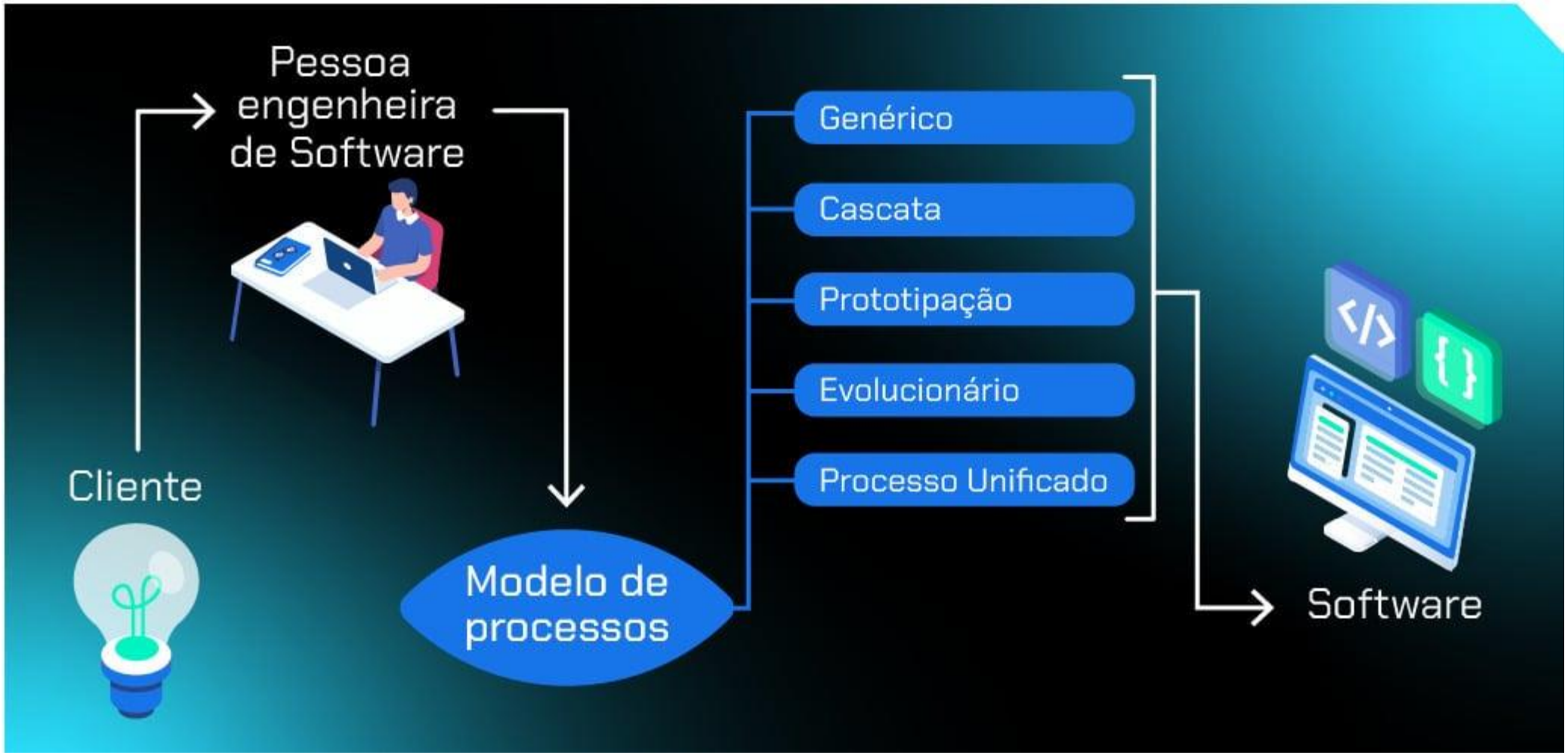
1) Qualidade

Para garantir que um software tenha a utilidade e qualidade desejada, é preciso definir métricas de qualidade a serem observadas durante todo o processo de desenvolvimento, como, por exemplo, desempenho do software, boa usabilidade, tolerância ao erro, acessibilidade, entre outros.

2) Processos

A engenharia de software segue uma metodologia com cinco etapas principais: comunicação, planejamento, modelagem, construção e entrega.
Essas etapas são flexíveis e podem ser adaptadas para criar modelos de desenvolvimento de software, como os paradigmas em cascata, prototipação, evolucionário, unificado, entre outros, como mostra a imagem:

UNICARIOCA
Centro Universitário



Vale ressaltar que esses modelos podem ser ajustados ao longo do tempo para atender às necessidades específicas de cada projeto.

Como apresentado na figura, o cliente expressa para a pessoa engenheira de software o que deseja e esse profissional define qual processo é mais adequado para construção do software desejado pelo cliente.

Vejamos um pouco sobre cada processo citado:

- **Genérico**: estabelece uma metodologia de processos genérica, ou seja, definindo a **comunicação, planejamento, modelagem, construção e entrega** como as cinco atividades metodológicas que podem ser aplicadas a todos os projetos de software, independente da complexidade ou do tamanho do software que desejamos desenvolver.

- **Cascata**: é um método linear e sequencial na engenharia de software, com atividades como **especificação de requisitos, planejamento, modelagem, construção, entrega e suporte**. Porém, sua rigidez sequencial não se adapta bem a projetos reais, já que é difícil para o cliente antecipar todas as suas necessidades no início.

- **Prototipação**: um protótipo é uma simulação ou parte de um sistema usado para identificar e definir requisitos de software com os *stakeholders* (partes interessadas). Começa-se com comunicação para definir objetivos gerais e requisitos. Depois, um protótipo é rapidamente projetado, construído e entregue para validação dos stakeholders. O feedback deles é então usado para refinar os requisitos, melhorando a entrega final. Este modelo pode ser combinado com outros para alinhar expectativas com o cliente e definir requisitos de forma eficaz.

• **Evolucionário**: propõe criar uma versão inicial do software que evolui à medida que os requisitos são refinados. Isso permite criar um produto mínimo viável, que cresce em recursos conforme as demandas do projeto.

• **Unificado**: o Processo Unificado (UP), também conhecido como RUP (Rational Unified Process), combina elementos dos modelos tradicionais em um fluxo iterativo e incremental. Dividido em cinco fases - concepção, elaboração, construção, transição e produção - é adequado para projetos complexos. A abordagem iterativa e incremental do RUP permite entregas contínuas, agilizando o processo de desenvolvimento e entrega de software.

3) Métodos

Os métodos definem como os processos serão aplicados, visando trazer mais agilidade e qualidade para a execução dos processos e consequentemente para o produto final.

Um grupo de desenvolvedores percebeu a necessidade de métodos mais ágeis para a produção de software.

Para promover essa cultura elaboraram o "**Manifesto para o Desenvolvimento Ágil de Software**", definindo valores fundamentais:

1.Priorizar indivíduos e interações sobre processos e ferramentas;
2.Funcionalidade do software sobre documentação extensiva;
3.Colaboração com clientes em vez de negociações contratuais;
4.Capacidade de resposta a mudanças em vez de aderência a um plano.

O manifesto ágil também estabelece 12 princípios:

- 1 - **Geração de valor**: satisfação do cliente com entregas rápidas e contínuas;
- 2 - **Flexibilidade**: aceitação de mudanças tardias no escopo do projeto;
- 3 - **Entregas contínuas**: fornecimento regular de incrementos funcionais em prazos curtos;
- 4 - **União e cooperação**: colaboração constante entre equipe e stakeholders;
- 5 - **Motivação**: criação de ambiente favorável para construir confiança;
- 6 - **Comunicação**: diálogo direto entre membros da equipe;
- 7 - **Funcionalidade**: progresso medido pelo funcionamento do software;
- 8 - **Sustentabilidade e evolução**: desenvolvimento constante e sustentável até a conclusão;
- 9 - **Qualidade**: foco na excelência técnica no design do software;
- 10 - **Simplicidade**: minimização do trabalho desnecessário;
- 11 - **Organização**: melhores resultados com equipes auto-organizadas;
- 12 - **Autoavaliação**: reflexão regular para aprimoramento do desempenho.

Antes mesmo que o manifesto ágil ficasse conhecido, algumas **metodologias ágeis como XP e SCRUM** já vinham sendo adotadas por algumas empresas e instituições.

Essas metodologias, permitem o alinhamento entre processos tradicionais de desenvolvimento e o desenvolvimento ágil.

4) Ferramentas

As ferramentas **são softwares utilizados para dar suporte a cada etapa do desenvolvimento de um projeto de software**.
Essas ferramentas também podem servir para automatizar processos, tornando o desenvolvimento mais rápido e eficiente.
Vejamos algumas ferramentas que podem ser utilizadas em diferentes fases do desenvolvimento, considerando o modelo genérico da engenharia de software (comunicação, planejamento, modelagem, construção e entrega).

4.1 - Comunicação e Planejamento

A fase de comunicação e planejamento corresponde ao levantamento de requisitos junto às partes interessadas e a criação de um backlog para desenvolvimento. Para isso, podem ser utilizadas as seguintes ferramentas:

- Slack para a comunicação e a colaboração entre as equipe;
- Jira ou Azure para criação do backlog do produto e das entregas contínuas.

4.2 - Modelagem

A modelagem é essencial para criar representações visuais e abstratas do sistema que está sendo desenvolvido. Ferramentas utilizadas nesta fase incluem:

- Figma para design e prototipação;
- Astah para criação de diagramas UML (Unified Modeling Language).

4.3 – Construção

A fase de construção corresponde ao processo de implementação ou codificação e verificação com a realização de testes. Para isso, podem ser utilizadas as seguintes ferramentas:

- Eclipse, Visual Studio Code, entre outros como ambiente de desenvolvimento integrado;
- Git para manter um controle e versionamento do projeto. Além disso, o uso dessa ferramenta também auxilia no merge (integração) com códigos anteriores caso existam;
- Jira para gerenciamento das tarefas a serem desenvolvidas.

Os testes devem ser realizados de forma contínua. Para isso, podem ser adotadas ferramentas e metodologias como Behavior-Driven Development (BDD).

Nessa fase, poderão ser utilizadas as seguintes ferramentas:

- jUnit para criação de testes de unidade;
- Vagrant para criar e manter ambientes de desenvolvimento virtuais portáteis, utilizando VirtualBox, KVM, Hyper-V, Docker containers, VMware, e AWS;
- Selenium para criação de testes funcionais automatizados;
- jMeter para realização de testes de carga e de estresse;
- Mantis para o gerenciamento de bugs.

4.4 - Entrega

A fase de entrega corresponde ao processo de integração, em que a parte do software desenvolvida é integrada ao software em desenvolvimento. Essa integração deve ser feita de forma contínua com uma boa comunicação entre a equipe de desenvolvimento e a equipe de operações. Nessa fase, poderão ser utilizadas as seguintes ferramentas:

- Git para controle de versão e gerenciamento dos merges de integração;
- Docker para virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres;
- Jenkins para automatização do processo de integração contínua;
- Ansible para o gerenciamento de implantação da aplicação.

Essas camadas interagem e se sobrepõem na prática diária de pessoas engenheiras de software, fornecendo uma estrutura sistemática para desenvolver software de alta qualidade, seguro e eficiente.

Por isso é importante ser versátil, ter habilidades em análise, design, codificação, teste, manutenção e gestão, além de se manter atualizado com as melhores práticas e ferramentas do mercado.

As áreas de atuação da Engenharia de Software

Por ser uma profissão com amplas possibilidades de atuação, as pessoas engenheiras de software pode atuar em:

- **Análise de requisitos:** compreender os requisitos do usuário e transformá-los em especificações técnicas para o desenvolvimento do software;
- **Design de software:** criar arquiteturas e designs de software que atendam aos requisitos e sejam escaláveis, eficientes e seguros;
- **Programação:** escrever código limpo e eficiente em várias linguagens de programação para implementar funcionalidades e resolver problemas;
- **Teste de software:** desenvolver e executar testes para garantir a qualidade do software, identificando e corrigindo bugs e falhas;
- **Manutenção e atualização:** realizar atualizações, correções de bugs e melhorias no software existente para garantir que ele funcione de forma confiável e eficaz ao longo do tempo.
- **Qualidade de software:** conhecer o máximo que puder sobre as tecnologias envolvidas no produto, a metodologia de trabalho, a arquitetura do software e as técnicas e estratégias de teste.

Essas são apenas algumas das principais possibilidades de atuação e habilidades técnicas (**hard skills**) necessárias para engenharia de software.

UNICARIOCA
Centro Universitário

Pensado de maneira mais prática, uma pessoa engenheira de software pode atuar em projetos que envolve:

- **Desenvolvimento de aplicações**: web, mobile, desktop, games etc.
- **Engenharia de sistemas**: operacionais e sistemas embarcados para dispositivos específicos, como eletrodomésticos, automóveis, e equipamentos médicos.
- **Desenvolvimento de infraestrutura e ferramentas**: nas áreas de DevOps e computação em nuvem.
- **Ferramentas de desenvolvimento**: como editores de código, IDEs e frameworks.
- **Engenharia de dados**: big data e com design, implementação e manutenção de bancos de dados.
- **Inteligência artificial e machine learning**: no desenvolvimento de algoritmos e sistemas de aprendizado de máquina.
- **Segurança da informação**: atuando em áreas como cibersegurança e criptografia.
- **Teste e Qualidade de Software**: atuando como QA (Quality Assurance) e realizando automação de testes.
- **Gestão de projetos e produtos de software**: como gerente de produto (product management) para gerir o ciclo de vida de produtos de software.
- **Consultoria e treinamento**: realizando aconselhamento técnico para empresas e projetos específicos, além da possibilidade de ministrar cursos e treinamentos para novas pessoas desenvolvedoras e equipes.

UNICARIOCA
Centro Universitário

Habilidades necessárias

Confira algumas habilidades importantes (soft skills) para uma pessoa engenheira de software:

- **Colaboração em equipe**: trabalhar em equipe com outras pessoas engenheiras de software, designers, gerentes de projeto e outras partes interessadas para desenvolver e entregar produtos de software de alta qualidade;
- **Aprendizado contínuo**: manter-se atualizado com as últimas tecnologias, ferramentas e práticas de desenvolvimento de software para melhorar constantemente suas habilidades e conhecimentos;
- **Comunicação clara**: comunicar de forma clara e objetiva é essencial na engenharia de software. Uma comunicação eficaz pode melhorar a colaboração, reduzir conflitos e aumentar a eficiência da equipe;
- **Resolução de problemas**: como pessoa engenheira de software, enfrentará diversos desafios, não apenas técnicos. A habilidade de pensar criticamente e encontrar soluções eficazes para diferentes situações é crucial e valorizada nesse contexto;
- **Adaptabilidade**: os profissionais de engenharia de software precisam se adaptar constantemente às mudanças tecnológicas, como o surgimento de novas linguagens de programação, frameworks e tecnologias. Estar aberto a aprender novos conhecimentos é essencial para o sucesso nesse campo em constante evolução.

Evolução do desenvolvimento de software

O desenvolvimento de software, bem como outras Ciências, empregou diversas mudanças e adaptações para melhorar, facilitar e adaptar-se ao cotidiano dos profissionais que realizam esse trabalho. As principais evoluções no desenvolvimento de software podem ser classificadas em dois grandes grupos: mudanças processuais e mudanças tecnológicas.

Mudanças tecnológicas

Embora atualmente, quando se fala em software, sejamos remetidos a lembrar de computadores modernos, smartphones, tablets, etc., o desenvolvimento de software começou muito antes desses dispositivos serem criados, sendo programado por volta de 1725, em cartões perfurados. Posteriormente, surgiram as primeiras linguagens de programação, tais quais as que existem hoje, sendo elas FORTRAN (1955), List Processor (LISP) e Common Business Oriented Language (COBOL). Posteriormente, surgiram linguagens de programação de alto nível, isto é, que se aproximam mais da linguagem humana, são exemplos: Java, JavaScript, Visual Basic, Object Pascal e PHP.

Junto com as linguagens de programação, foram sendo criados paradigmas para o desenvolvimento de sistemas. Um paradigma nada mais é do que a forma como um sistema é construído e seu desenvolvimento é organizado.

Os paradigmas mais conhecidos são o paradigma estruturado e o paradigma orientado a objetos, sendo o paradigma orientado a objetos o mais utilizado atualmente. A programação orientada a objetos é um paradigma em que o software é construído considerando que tudo o que é inserido no programa é um objeto e que esse objeto pertence a uma classe e tem características (atributos) específicas sobre as quais podem ser feitas ações (métodos).

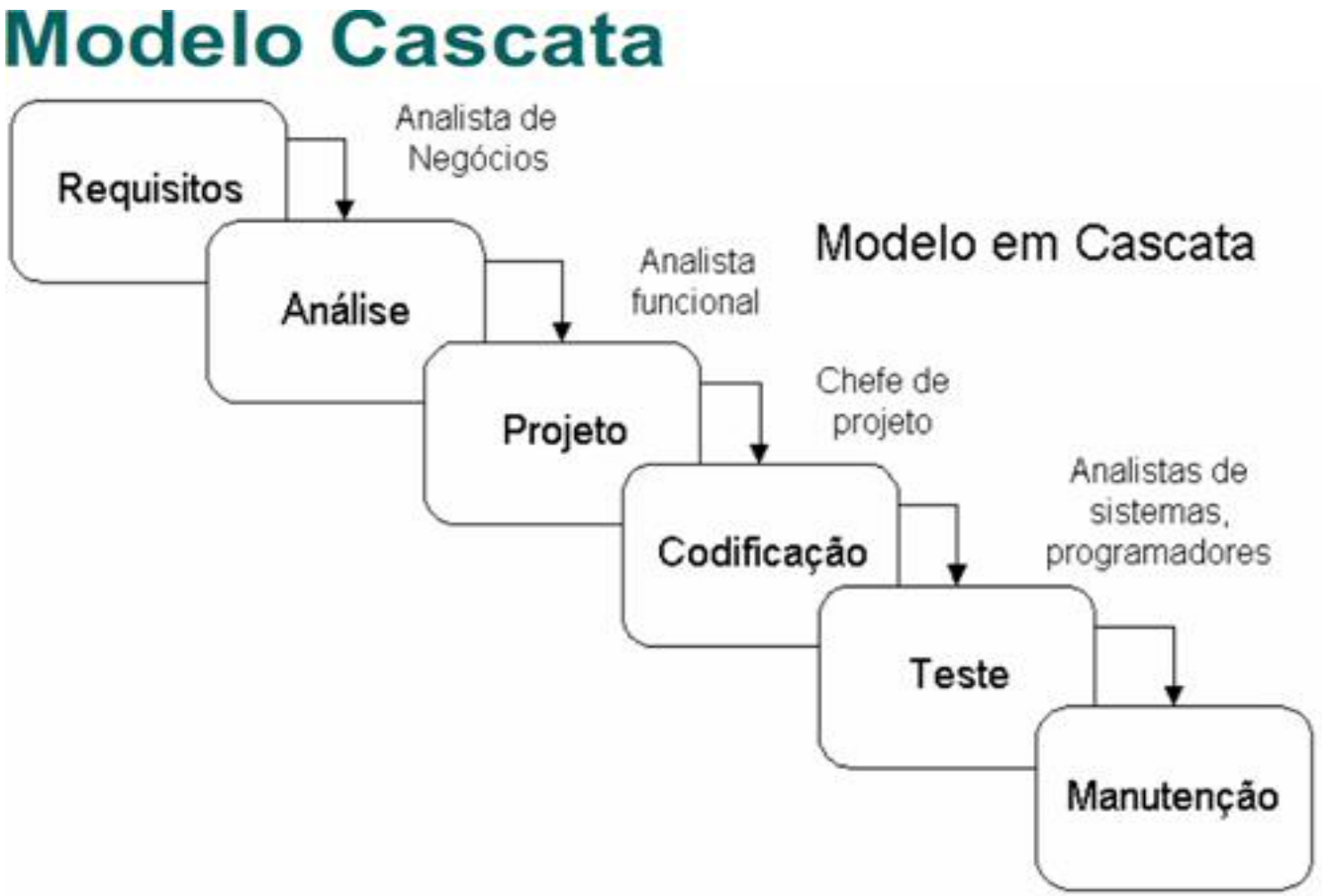
Por outro lado, um princípio básico da programação estruturada é que um programa pode ser dividido em três partes que se interligam, sendo elas sequência, seleção e iteração. Na sequência, o código do programa é criado para ser executado de forma sequencial, seguindo estritamente a ordem na qual foi programado.

Na seleção, o programa encontra locais onde pode seguir um ou mais caminhos distintos. Na interação, é permitido ao programa executar diversas vezes o mesmo trecho de código.

Mudanças de processo

No desenvolvimento de sistemas, além das mudanças tecnológicas, foram ocorrendo mudanças na forma como as empresas se organizam e estruturam o trabalho. A forma tradicional de desenvolvimento de sistemas foi a primeira a ser criada, empregando o ciclo de vida em estrutura de cascata (1970), na qual as etapas são executadas de forma sequencial, sem que seja possível retornar de uma etapa posterior para uma etapa anterior.

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.



O modelo em espiral se assemelha muito ao modelo iterativo e incremental, uma vez que ele também considera pequenas entregas de software e a execução de todas as etapas espiralmente (várias vezes). Contudo, o ciclo de vida espiral considera a presença explícita da análise de riscos como uma das etapas de cada iteração. Tanto que a fase de desenvolvimento é realizada após a fase de análise de riscos.

Nesse processo em espiral, o ciclo de vida do software é representado como uma espiral em que a volta na espiral representa uma fase do processo de software, sendo que a volta mais interna pode preocupar-se com a viabilidade do sistema, o ciclo seguinte, com definição de requisitos, o seguinte, com o projeto do sistema, e assim por diante.

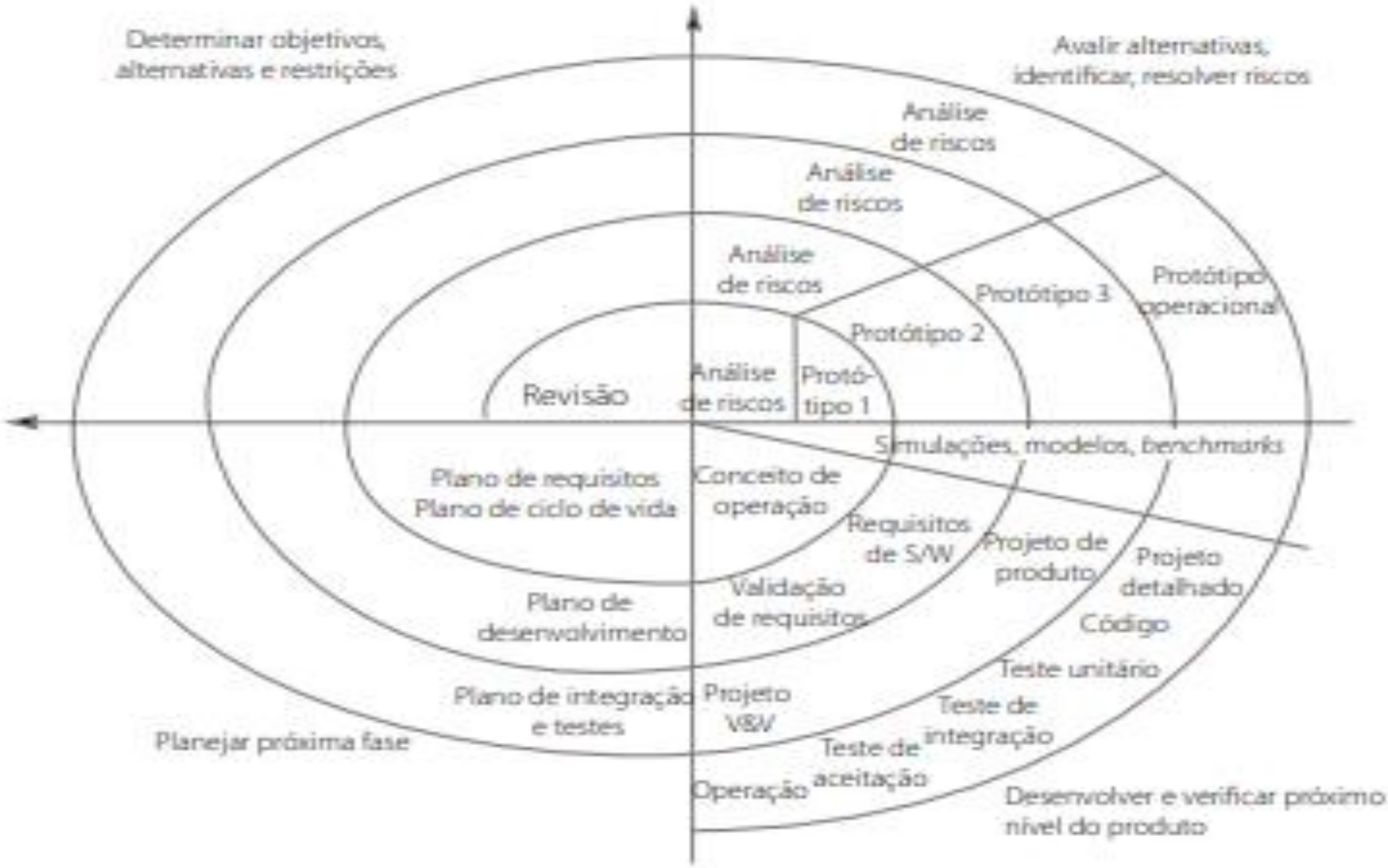


Figura 3. Modelo Espiral.

Fonte: Sommerville (2011, p. 33).

No ano de 2001, um grupo de profissionais de tecnologia da informação lançou um documento chamado Manifesto Ágil para o Desenvolvimento de Sistemas. Deste então popularizaram-se os métodos ágeis de desenvolvimento de sistemas, sendo os mais conhecidos o método Scrum e o método XP. Todos eles têm em comum a aplicação dos valores propostos pelo Manifesto Ágil para o Desenvolvimento de Sistemas, sendo eles: indivíduos e interações são mais que processos e ferramentas, software em funcionamento é mais que documentação abrangente, colaboração com o cliente é mais que negociação de contratos e respostas a mudanças são mais que seguir um plano.

Todos esses ciclos de vida, somados aos Métodos Ágeis de Desenvolvimento de Sistemas, apresentaram estrutura e organização maiores para o processo de desenvolvimento de sistemas, propiciando melhoria da comunicação entre os envolvidos no processo, seja entre os próprios profissionais de Tecnologia da Informação ou destes com o cliente. Com a adoção de processos e a atenção às evoluções tecnológicas, buscando sempre acompanhar aquilo que o mercado tem de melhor para oferecer, pode-se atingir maior excelência nos produtos entregues e atender melhor às necessidades do cliente.

Obrigado!

UNICARIOCA
Centro Universitário

1º LUGAR
Melhor Centro Universitário privado do Rio de Janeiro

Melhor EAD do Brasil
Conceito **5** do MEC