DEVELOPMENT ENVIRONMENT

MAS COMO UM **SOFTWARE É CRIADO?**

2

LISTA DE FIGURAS

SUMÁRIO

1 MAS COMO UM SOFTWARE É CRIADO?

1.1 Antes de começar...

Tenho certeza de que está ansioso(a) para começar a desenvolver o Fintech. No entanto, você pode ser novo no “ramo” e talvez nem sequer saiba o que é um *software*. Na verdade, muitos sabem o que é, mas não sabem conceituá-lo: o que, de fato, é um *software*? Tanta coisa mudou nas últimas décadas! Veremos tudo isso a seguir.

1.2 Introdução

Diante da crescente demanda por desenvolvimento de *software* e, por conseguinte, do surgimento de novos *softwares* e *hardwares*, as indústrias de *softwares* precisam engajar-se nessa onda de competitividade, melhorando de maneira eficaz sua produtividade a fim de enfrentar adequadamente essa realidade em constante evolução.

Uma particularidade inerente aos sistemas de *software* é a dificuldade de seu desenvolvimento, que evolui à medida que surgem novas tecnologias e cresce o tamanho do sistema.

Durante as fases de desenvolvimento do *software*, ao combinar os métodos, as melhores ferramentas para automatizar esses métodos, as técnicas para a garantia da qualidade do *software* e os procedimentos de controle e gestão, é possível aplicar as boas práticas sugeridas pela engenharia de *software*.

1.2.1 Conceitos

1.2.1.1 Definição de *software*

Rezende (2005, p. 2) define: “*Software* é um subsistema de um sistema computacional. São os programas de computadores”.

Muitos entendem *software* como um programa de computador escrito numa linguagem específica a fim de produzir a função e o desempenho esperados.

De acordo com Pressman (2009), "*software* é um conjunto composto de instruções de computador, estruturas de dados e documentos".

1.2.1.2 Definição de sistema

Sistema é um conjunto de informações e procedimentos que interagem entre si para que os objetivos sejam alcançados.

1.2.1.3 Definição de engenharia

Rezende (2005, p. 1) conceitua: "Engenharia é a arte das construções, com base no conhecimento científico e empírico. Arte adequada ao atendimento das necessidades humanas".

A engenharia representa uma metodologia unida ao esforço para empreender resultados. Esses resultados são provenientes de trabalhos focados em diversas áreas, nas quais se possui um amplo conhecimento a fim de propor soluções às necessidades.

1.2.1.4 Definição de engenharia de software

A engenharia de *software* envolve um conjunto de tarefas que se iniciam no momento que são registradas as necessidades das partes interessadas e terminam no momento que o *software* deixa de ser utilizado por seus usuários.

De acordo com Pressman (2009, p. 31), é necessário o "estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um *software* que seja confiável e que funcione eficientemente em máquinas reais".

O crescimento da competição no mercado e o desenvolvimento da inteligência computacional das máquinas ocasionaram demanda por *softwares* cada vez mais complexos. O surgimento de tais sistemas resultou na necessidade de reavaliação da forma de desenvolver *softwares*.

Conforme Sommerville (1998, p. 6), “Engenharia de *software* é uma engenharia que se ocupa de todos os aspectos da produção de *software*”.

A engenharia de *software* consiste num conjunto de métodos, ferramentas e procedimentos que têm por objetivo o desenvolvimento do *software* com qualidade, agregando valor ao seu empreendimento e atendendo às necessidades ou expectativas dos *stakeholders*.

A engenharia de *software* aplica práticas existentes em algumas áreas da engenharia, tais como:

- Desenvolvimento de *softwares* com qualidade.
- Trabalho em equipe.
- Gerenciamento do processo de desenvolvimento.
- Custos e prazos admissíveis.

1.2.2 Evolução do *hardware* e do *software*

Ao iniciar os estudos sobre a engenharia de *software*, um dos primeiros questionamentos que surgem diz respeito à sua origem e evolução até os dias atuais; o que nos remete ao século XVII, em busca de suas raízes.

O contexto em que o *software* foi desenvolvido está ligado a várias décadas de evolução dos *hardwares*. O melhor desempenho do *hardware* e o custo mais baixo aceleraram o aparecimento de sistemas de *software*.

Os autores Velloso (2014) e Pati (2016) comentam sobre a evolução dos *hardwares* e dos *softwares*. Ela está relacionada ao aprendizado contínuo de novas competências e habilidades ao longo do tempo.

Até o início do século XVIII, não se diferenciava com clareza o que era *hardware* ou *software*, pois os equipamentos eram basicamente eletromecânicos.

As figuras a seguir detalham a evolução do *hardware* e do *software*, a sua importância e como se relaciona com o desenvolvimento do *software*.

- Século XVII

O francês Blaise Pascal inventa uma calculadora que soma e subtrai, e o alemão Gottfried Wilhelm von Leibniz adiciona as operações de multiplicar e dividir à máquina.

Figura 1 – Calculadora
Fonte: Adaptado por FIAP (2016)

- Século XVIII

O francês Joseph-Marie Jacquard desenvolve um tear automatizado, no qual os cartões perfurados controlam o movimento da máquina.

Figura 2 – Tear automatizado
Fonte: Adaptado por FIAP (2016)

- 1834

O inglês Charles Babbage prepara a Máquina Analítica capaz de armazenar informações.

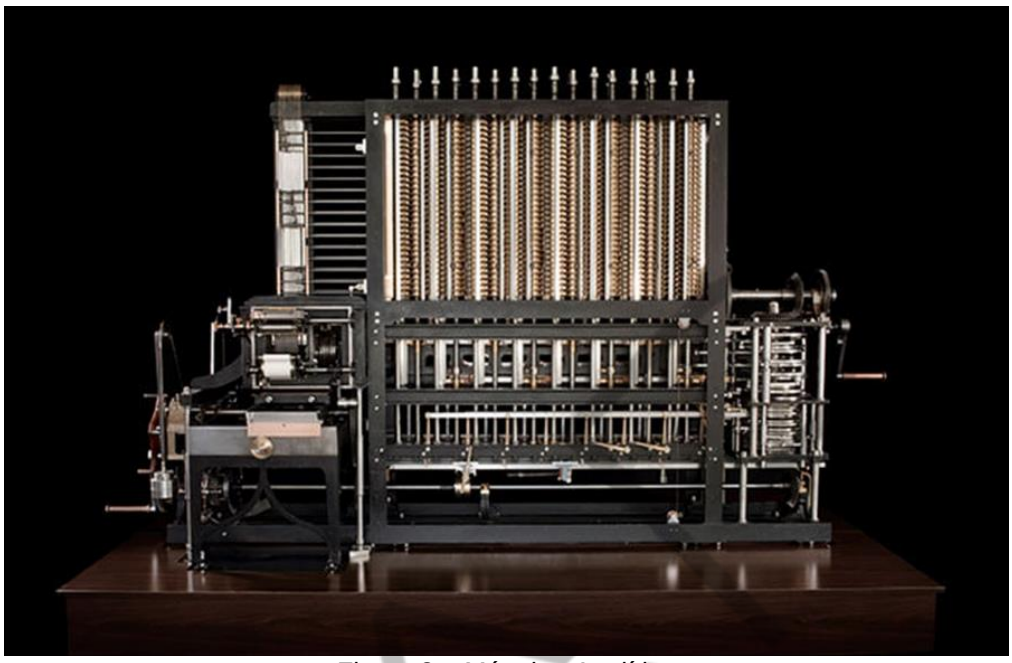


Figura 3 – Máquina Analítica
Fonte: Adaptado por FIAP (2016)

- 1842

Ada Lovelace escreve instruções para a Máquina Analítica de Babbage, concebida para executar um conjunto de tarefas de cálculo.

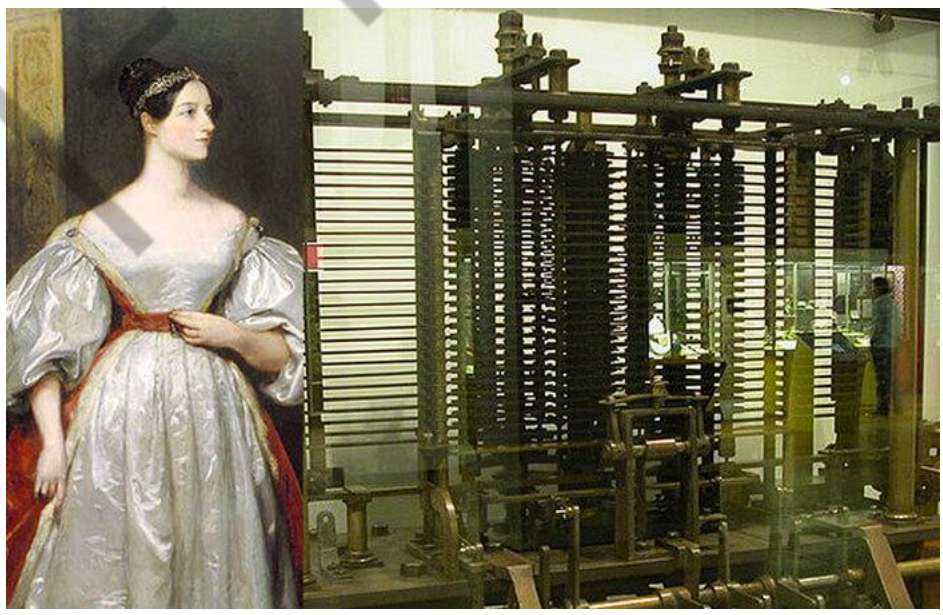


Figura 4 – Máquina Analítica de Babbage
Fonte: Adaptado por FIAP (2016)

- 1847

O inglês George Boole sistematiza a lógica binária para armazenar informações.

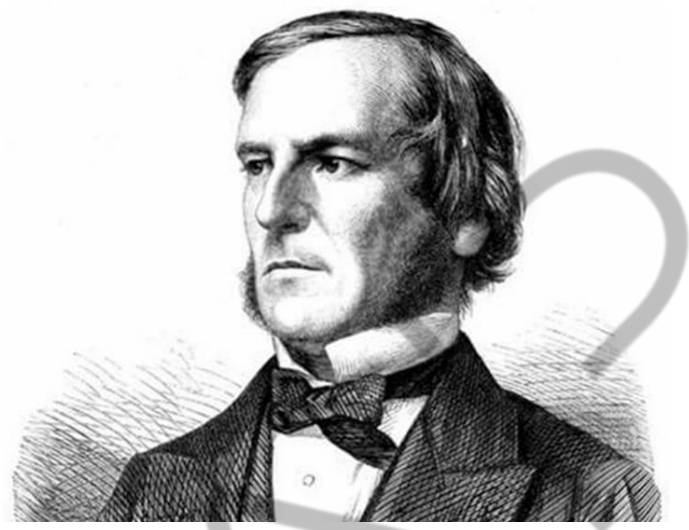


Figura 5 – George Boole
Fonte: Adaptado por FIAP (2016)

- 1849

Surge o balão austríaco, o mais antigo veículo aéreo não tripulado, conhecido como o antepassado dos drones.

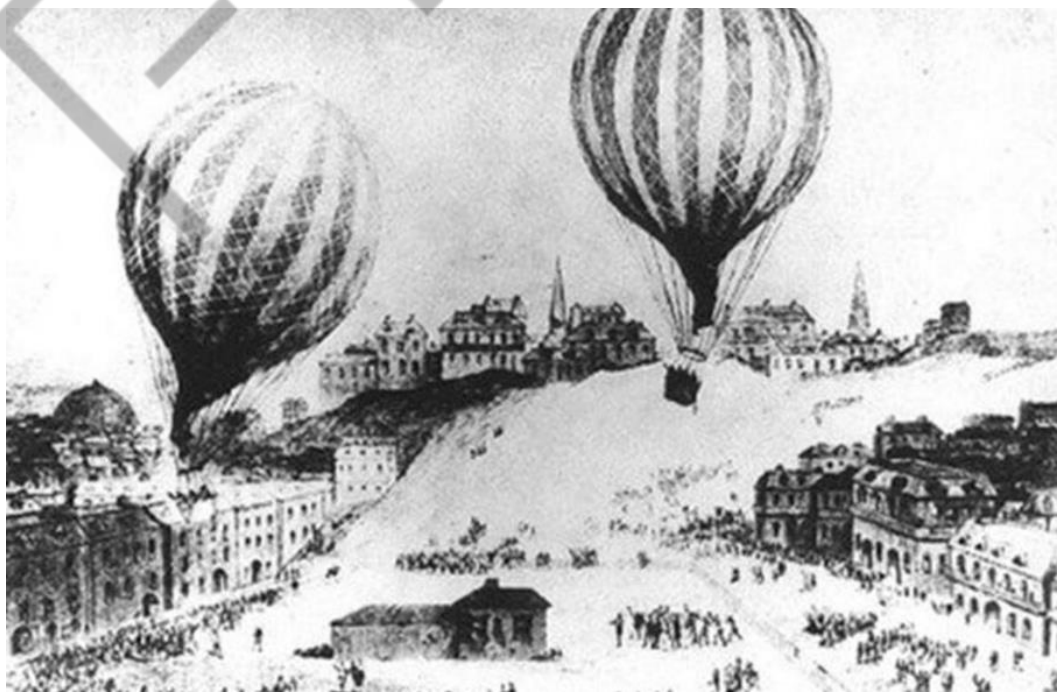


Figura 6 – Balões austríacos
Fonte: Adaptado por FIAP (2016)

- 1890

O norte-americano Herman Hollerith constrói o primeiro computador mecânico.

Figura 7 – Computador mecânico de Herman Hollerith
Fonte: Adaptado por FIAP (2016)

- 1924

Nasce a International Business Machines Corporation (IBM), nos Estados Unidos.

Figura 8 – International Business Machines Corporation (IBM)
Fonte: Adaptado por FIAP (2016)

- 1938

O alemão Konrad Zuse cria o primeiro computador elétrico usando a teoria binária.

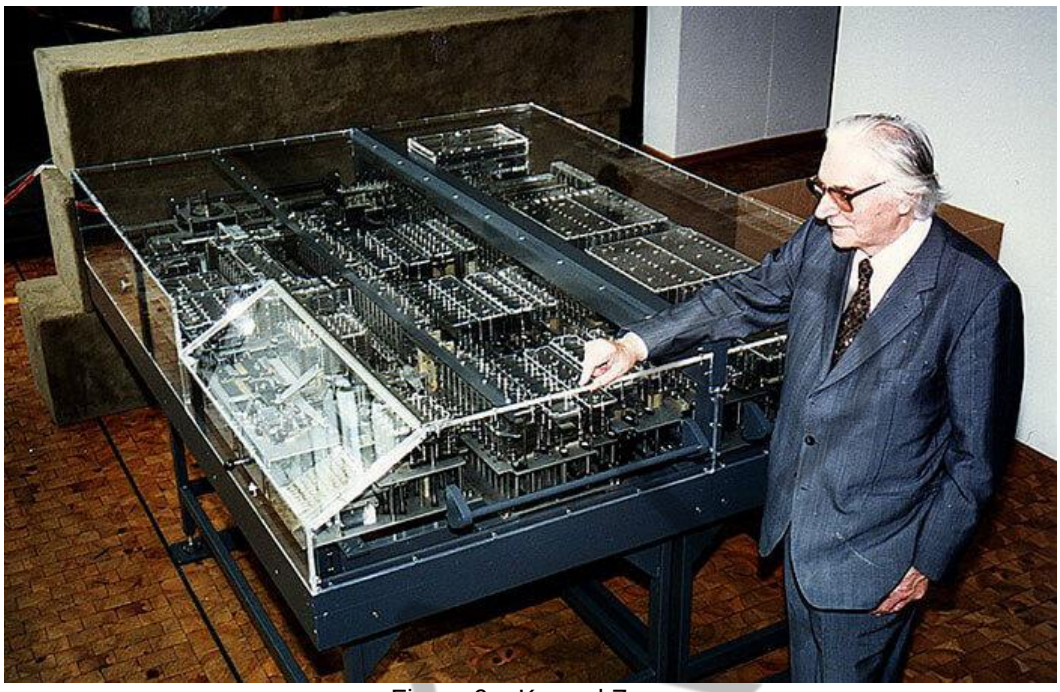


Figura 9 – Konrad Zuse
Fonte: Adaptado por FIAP (2016)

- 1943

O inglês Alan Turing constrói a primeira geração de computadores modernos, que utilizavam válvulas.

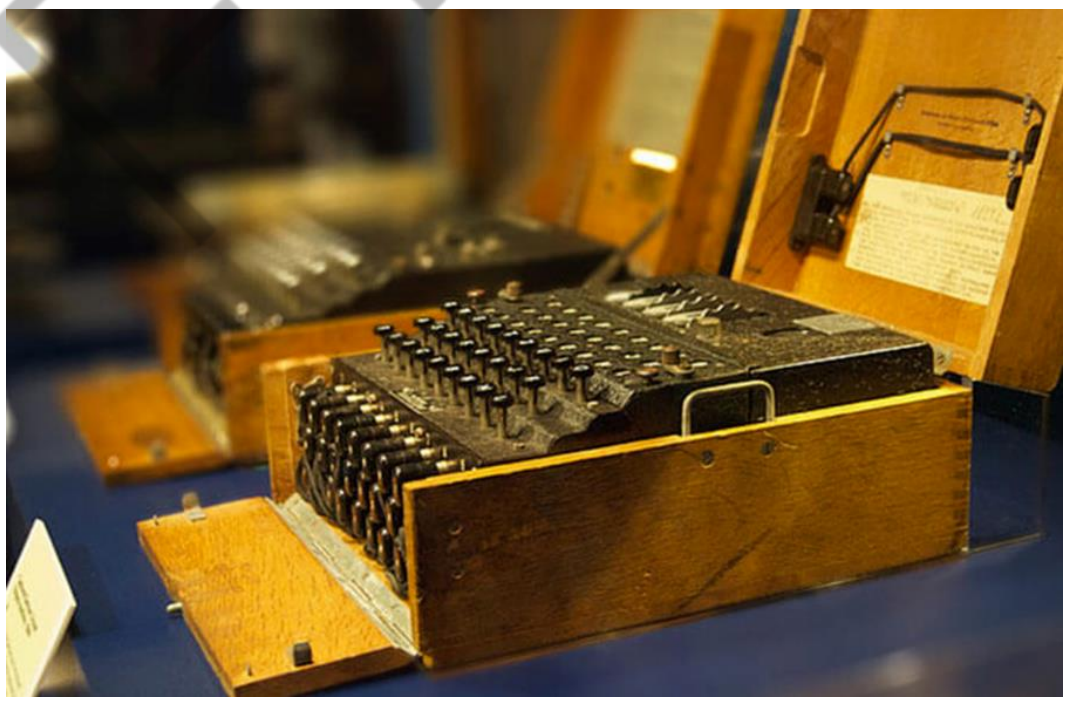


Figura 10 – Computador criado por Alan Turing
Fonte: Adaptado por FIAP (2016)

- 1944

O norte-americano Howard Aiken termina o Mark I, primeiro computador eletromecânico.

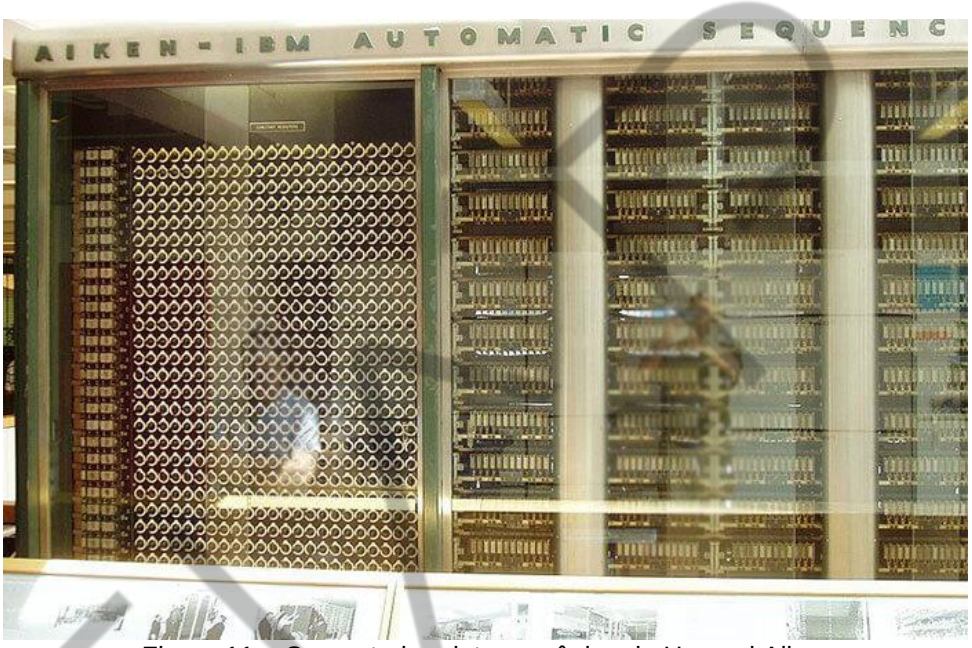


Figura 11 – Computador eletromecânico de Howard Aiken
Fonte: Adaptado por FIAP (2016)

- 1946

É criado nos EUA o Electronic Numerical Integrator and Computer (Eniac), primeiro computador eletrônico.

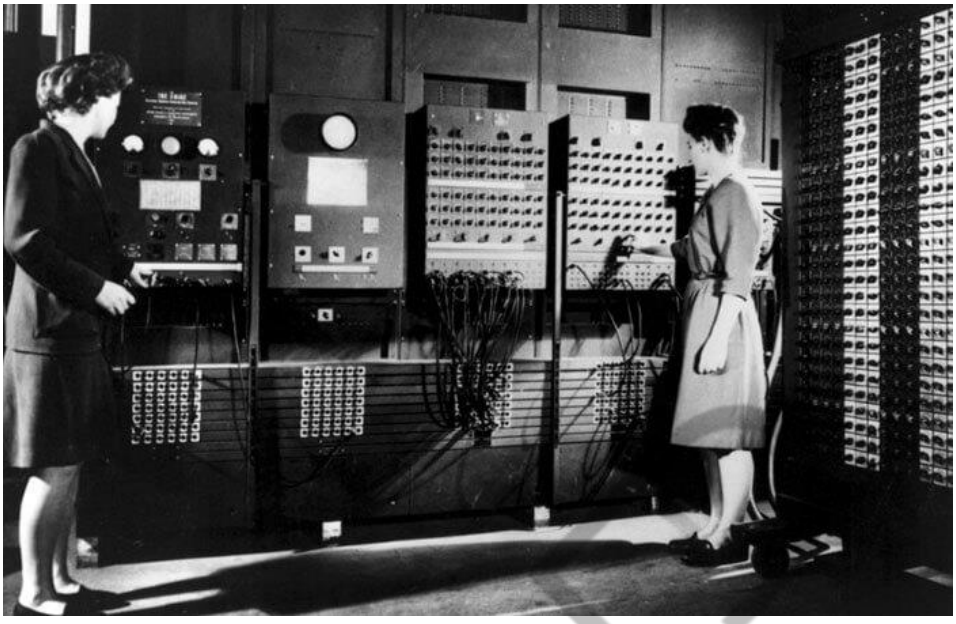


Figura 12 – Eniac
Fonte: Adaptado por FIAP (2016)

- 1947

É criado o transistor, substituto da válvula, que possibilita a criação de máquinas mais rápidas.

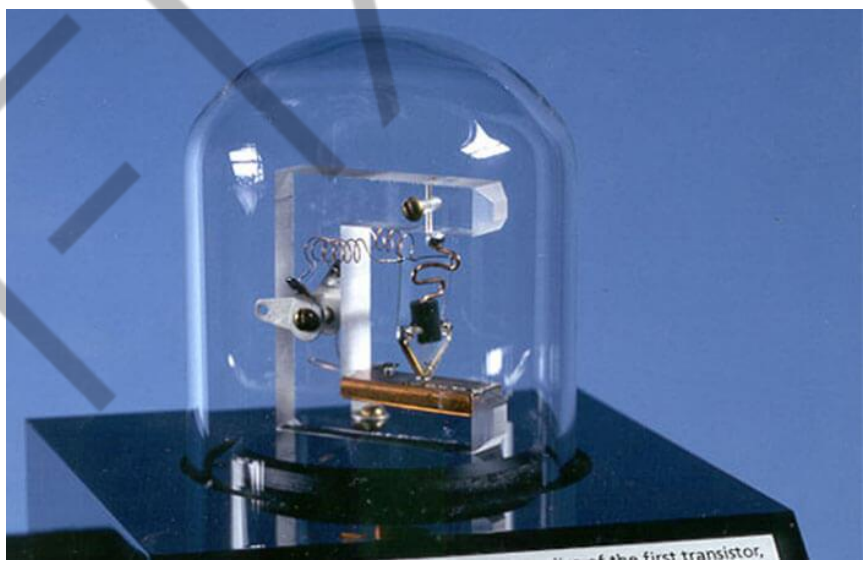


Figura 13 – Transistor
Fonte: Adaptado por FIAP (2016)

- 1957

Os primeiros modelos de computadores com transistor aparecem no mercado e oferecem mais confiabilidade e velocidade que as válvulas.

Figura 14 – Computador com transistor
Fonte: Adaptado por FIAP (2016)

- 1958

É criado o chip, circuito integrado que permite a miniaturização dos equipamentos eletrônicos.

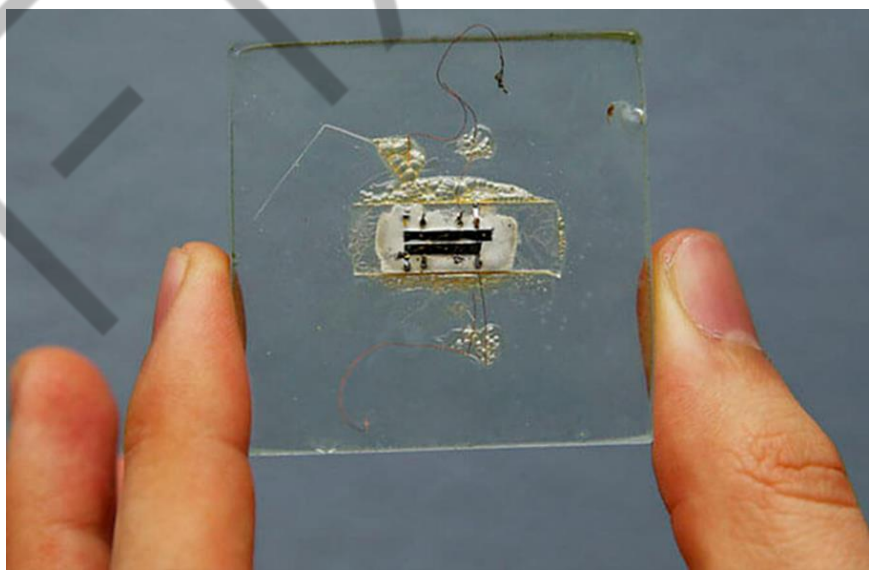


Figura 15 – Chip
Fonte: Adaptado por FIAP (2016)

- 1969

É constituída a Arpanet, rede de informação do Departamento de Defesa norte-americano que interliga universidades e empresas. Mais tarde, a Arpanet dará origem à internet.

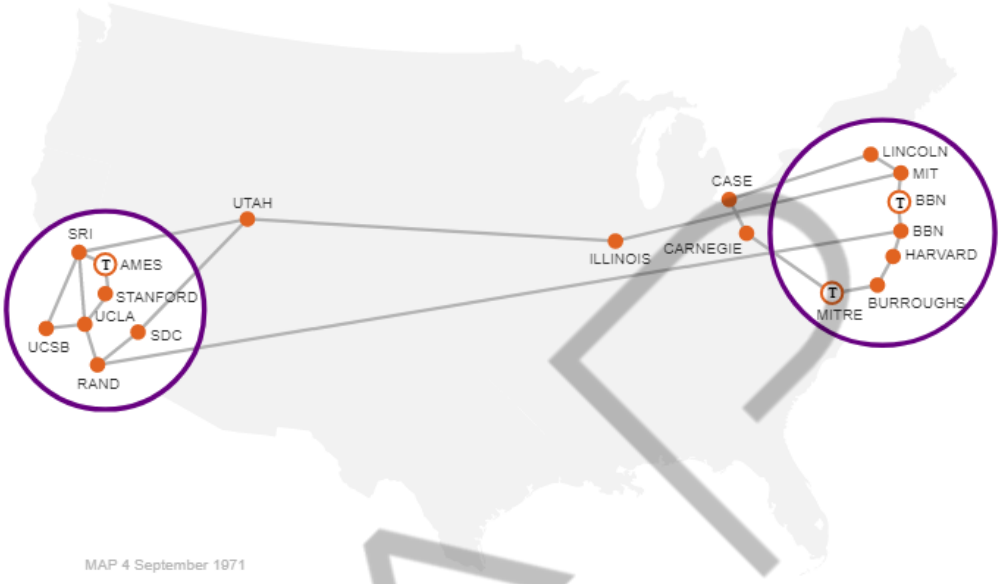


Figura 16 – Arpanet
Fonte: Adaptado por FIAP (2016)

- 1974

A Intel projeta o microprocessador 8080, que origina os microcomputadores. Nessa época, os *softwares* e sistemas se tornaram tão importantes quanto o *hardware*.

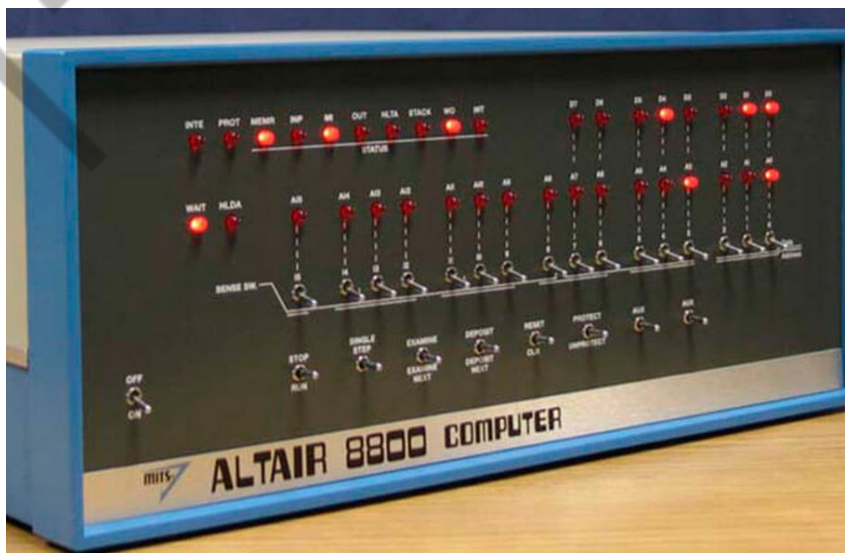


Figura 17 – Microprocessador 8080
Fonte: Adaptado por FIAP (2016)

- 1975

É fundada a Microsoft pelos norte-americanos Bill Gates e Paul Allen.

Figura 18 – Paul Allen e Bill Gates
Fonte: Adaptado por FIAP (2016)

- 1976

O Apple I, primeiro microcomputador comercial, é lançado por Steve Jobs e Steve Wozniak.

Figura 19 – Apple I
Fonte: Adaptado por FIAP (2016)

- 1981

A IBM lança seu microcomputador, o PC, com o sistema operacional MS-DOS elaborado pela Microsoft. O exército da África do Sul utiliza os drones israelenses em combate contra a Angola.

Figura 20 – Drone do exército
Fonte: Adaptado por FIAP (2016)

- 1983

A IBM lança o PC-XT, com disco rígido.

Figura 21 – PC-XT
Fonte: Adaptado por FIAP (2016)

- 1984

A National Science Foundation, nos Estados Unidos, financia o programa da Internet, que conecta governos, universidades e companhias.

A Apple lança o Macintosh, o primeiro computador a utilizar ícones e mouse.

Figura 22 – Macintosh
Fonte: Adaptado por FIAP (2016)

- 1985

A Microsoft lança o Windows para o PC, que só alcança êxito com a versão 3.0, em 1990.

Figura 23 – Windows para PC
Fonte: Adaptado por FIAP (2016)

- 1990

Os computadores portáteis (*laptops* e *palmtops*) se popularizam.

Figura 24 – Computador portátil
Fonte: Adaptado por FIAP (2016)

- 1992

O *smartphone* torna-se um aparelho prioritário para muita gente.

A IBM lança um dos primeiros aparelhos, o Simon.

Figura 25 – Simon
Fonte: Adaptado por FIAP (2016)

- 1996

A inteligência artificial ganha destaque ao estudar métodos de simular o pensamento humano nos computadores com o objetivo de substituir o homem pela máquina em atividades mecanizadas.

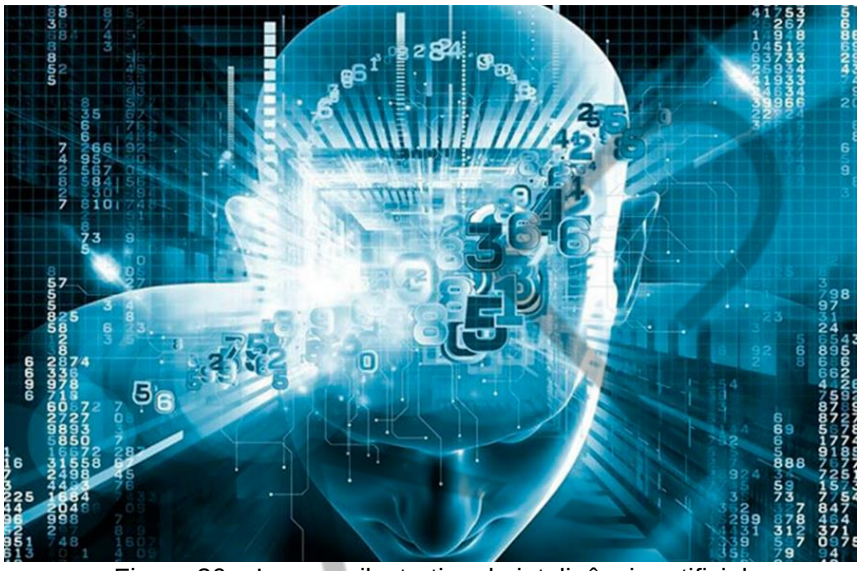


Figura 26 – Imagem ilustrativa de inteligência artificial
Fonte: Adaptado por FIAP (2016)

O Deep Blue, computador da IBM capaz de analisar 200 milhões de lances por segundo em um jogo de xadrez, vence uma disputa com o russo Garry Kasparov, campeão mundial de xadrez.

Figura 27 – Deep Blue vence disputa contra o campeão mundial de xadrez
Fonte: Adaptado por FIAP (2016)

- Fim do século 20

O Instituto de Tecnologia de Massachusetts (MIT) produz o COG, protótipo de robô cujo sistema nervoso artificial é 64 vezes mais potente do que um Macintosh e simula as fases de crescimento do homem, seus pensamentos e sentimentos.

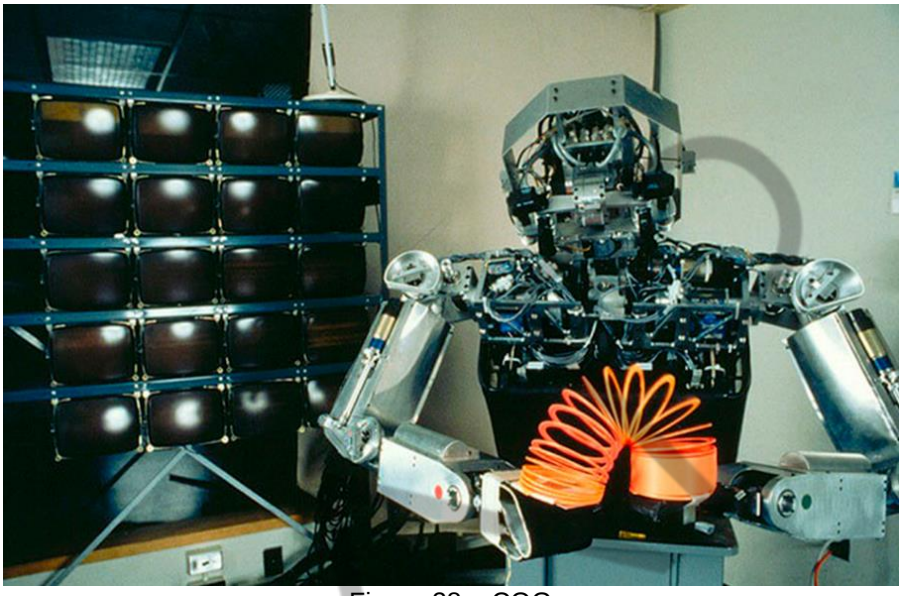


Figura 28 – COG
Fonte: Adaptado por FIAP (2016)

- 2000

Outros recursos são incluídos nos navegadores graças à evolução da internet. Isso possibilita o desenvolvimento de vários sistemas web que oferecem serviços inovadores e são acessados por meio da internet.

Figura 29 – Imagem ilustrativa
Fonte: Adaptado por FIAP (2016)

- 2007

É lançado o primeiro iPhone, que conquista o mercado.

Figura 30 – Primeiro iPhone
Fonte: Adaptado por FIAP (2016)

- 2008

A HTC lança o G1 com o sistema Android, porém, este não é vendido no Brasil. A Samsung foi a primeira a trazer para o Brasil um celular com Android, o Galaxy.

Figura 31 – G1
Fonte: Adaptado por FIAP (2016)

- 2012

Os estudantes da Singularity University, do Vale do Silício, criam o protótipo do robô Matternet, capaz de transportar medicamentos e alimentos para regiões de difícil acesso.

Figura 32 – Robô Matternet
Fonte: Adaptado por FIAP (2016)

- 2013

A Amazon anuncia o seu plano de utilizar drones para enviar mercadorias aos compradores com o objetivo de fazer as entregas apenas meia hora após a ordem de compra.

Figura 33 – Drone da Amazon para a entrega de produtos
Fonte: Adaptado por FIAP (2016)

- 2016

É realizado o Fórum Econômico Mundial, em Davos, cujo tema é a 4ª Revolução Industrial. Discute-se a forma de produzir e/ou consumir a informação digital e como se preparar para a era digital.

Figura 34 – 4ª Revolução Industrial
Fonte: Adaptado por FIAP (2016)

- 2019

O Google chega à supremacia quântica, lançando um computador quântico capaz de realizar em apenas alguns minutos cálculos que antes levariam anos.

Figura 35 – Computador quântico
Fonte: Adaptado por FIAP (2019)

2 PRINCIPAIS TIPOS DE SOFTWARE

A partir do contexto histórico da evolução do *hardware* e do *software*, foram construídos os alicerces do funcionamento do *software*.

2.1 Tipos de software

- **Software de Sistema:** é o conjunto de instruções que transformam o *hardware* num sistema com o qual o usuário pode interagir e fazer funcionar os seus programas. Exemplos: MAC, Linux e Windows.
- **Software de Aplicativo:** são programas que permitem aos usuários executar determinadas tarefas. Exemplos: Word, Paint, Excel e calculadora.
- **Software como Serviço**: é um modelo de compartilhamento de *software*, no qual é liberado apenas o acesso ao serviço oferecido, licenciado para uso por meio da internet; não é vendido nem instalado localmente. Exemplos: Google Docs e PDF Converter.
- **Software de Gestão:** auxilia na gestão empresarial; tem como objetivo otimizar a rotina da empresa, fornecendo uma informatização inteligente dos processos, como controle financeiro, compras e estoque. Exemplo: ERP.
- **Software Livre:** está disponível e é distribuído livremente; pode ser executado, adaptado, modificado e redistribuído de acordo com a necessidade de cada usuário. Ao redistribuir a nova versão, pode ser gratuito ou não.
- **Software Aberto:** é um *software* cujo código é acessível a qualquer pessoa; pode ser executado, modificado, estudado e distribuído gratuitamente. Sua filosofia é voltada para a colaboração entre usuários.
- **Software Proprietário:** pertence a um fabricante, que detém seus direitos de uso, edição e redistribuição. O usuário deve pagar por uma licença e só pode utilizar o *software* em um contexto restrito. Exemplo: Office, Photoshop e Winzip.

2.2 Crise do *software*

A fim de se obter uma clara percepção da engenharia de *software,* é relevante explorar os problemas que afetam seu desenvolvimento.

A Figura “Crise do *software* e seu processo de desenvolvimento” destaca o questionamento: a crise é pertinente ao *software* ou ao processo de desenvolvimento de *software*?

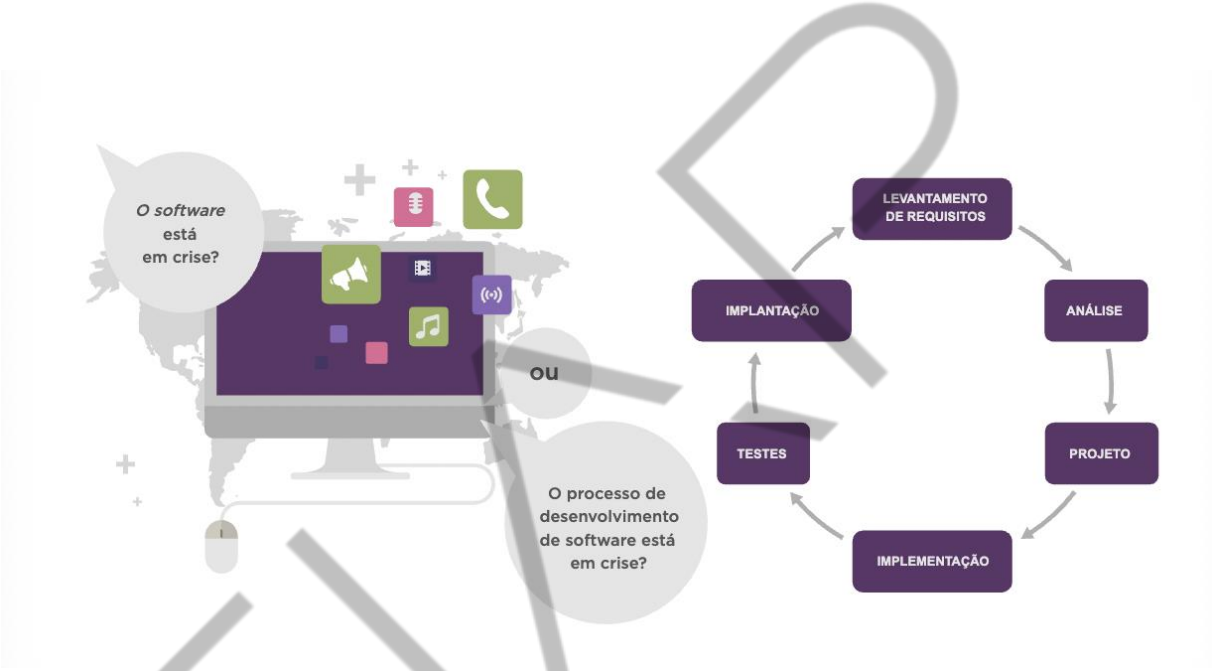


Figura 36 – Crise do *software* e seu processo de desenvolvimento
Fonte: Elaborado pelas autoras (2016)

O estudo mostra que não há crise no desenvolvimento de *software*, pois o crescimento da competição e da inteligência computacional dos *hardwares* resultou na demanda por sistemas de *software* cada vez mais complexos, que tirem proveito de tal capacidade. A chegada desses sistemas mais complexos ocasionou a necessidade de reavaliação da forma de desenvolver sistemas. Desde o aparecimento do primeiro computador até os dias de hoje, as técnicas para o desenvolvimento de *software* têm evoluído a fim de suprir as necessidades das partes interessadas.

Falhas são encontradas com frequência, há uma grande probabilidade de elas terem sido introduzidas durante a especificação de requisitos e passarem despercebidas durante o desenvolvimento e os testes do *software*.

Um componente fundamental do desenvolvimento de *software* é a qualidade da comunicação entre os clientes e a equipe de desenvolvimento. Caso ela falhe, o sistema também falhará. É essencial compreender o que o cliente quer e quais são suas necessidades, antes de começar a desenvolver um *software* que o ajude a resolver problemas.

Para entender e avaliar a origem das falhas em projetos de desenvolvimento de *software,* foram efetuados estudos e pesquisas, entre eles o do Departamento de Defesa dos Estados Unidos (DOD) e os do Standish Group.

O estudo efetuado pelo DOD na década de 1990 indicou que 75% de todos os grandes sistemas de *softwares* desenvolvidos falham, e a principal causa disso é a falta de gerenciamento adequado por parte do desenvolvedor e do cliente, e não em razão do desempenho técnico (SOTILLE, 2014).

Buscando os fatores que contribuem para o fracasso no desenvolvimento de *software*, a Figura “Fatores do fracasso no desenvolvimento de *softwares*” destaca como principal fonte de erros no ciclo de vida de um *software* a fase de identificação dos problemas, conhecida como análise de requisitos (CORDEIRO, 2002).

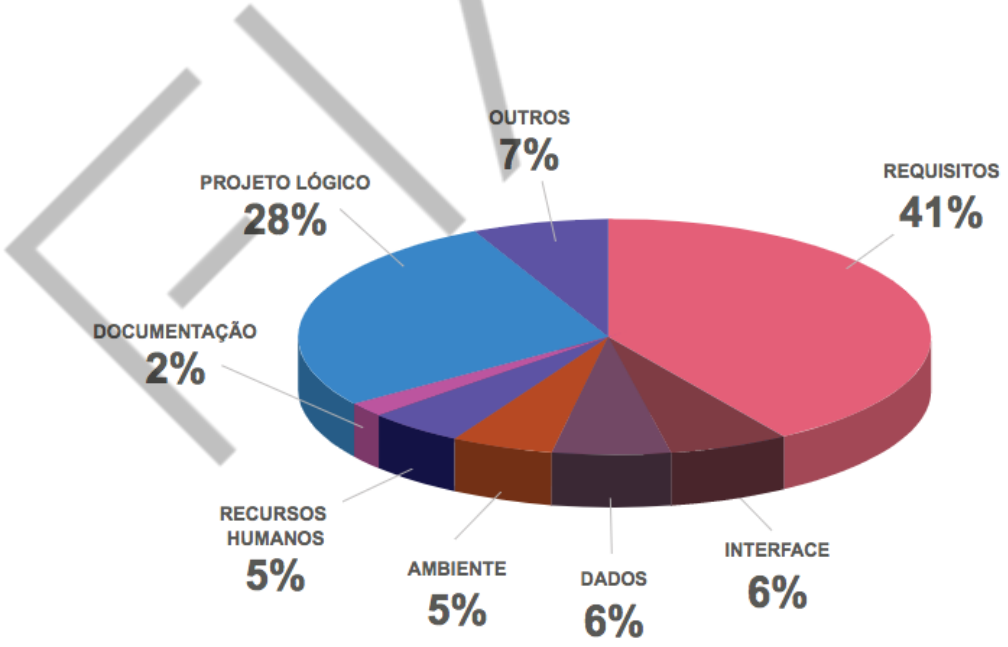


Figura 37 – Fatores do fracasso no desenvolvimento de *softwares*
Fonte: Cordeiro (2002)

O período de 1994 a 2015, apresentado na Figura “Resultado dos projetos de sistemas de *software*”, mostra a evolução do cenário anteriormente apresentado.

Destaca-se que em 2002 houve uma melhoria significativa no percentual de projetos entregues dentro do tempo, do custo e das especificações previstos, o que subiu para 34%; o percentual de projetos cancelados ou fracassados antes de serem concluídos caiu para 15%; porém os atrasados representaram 51%.

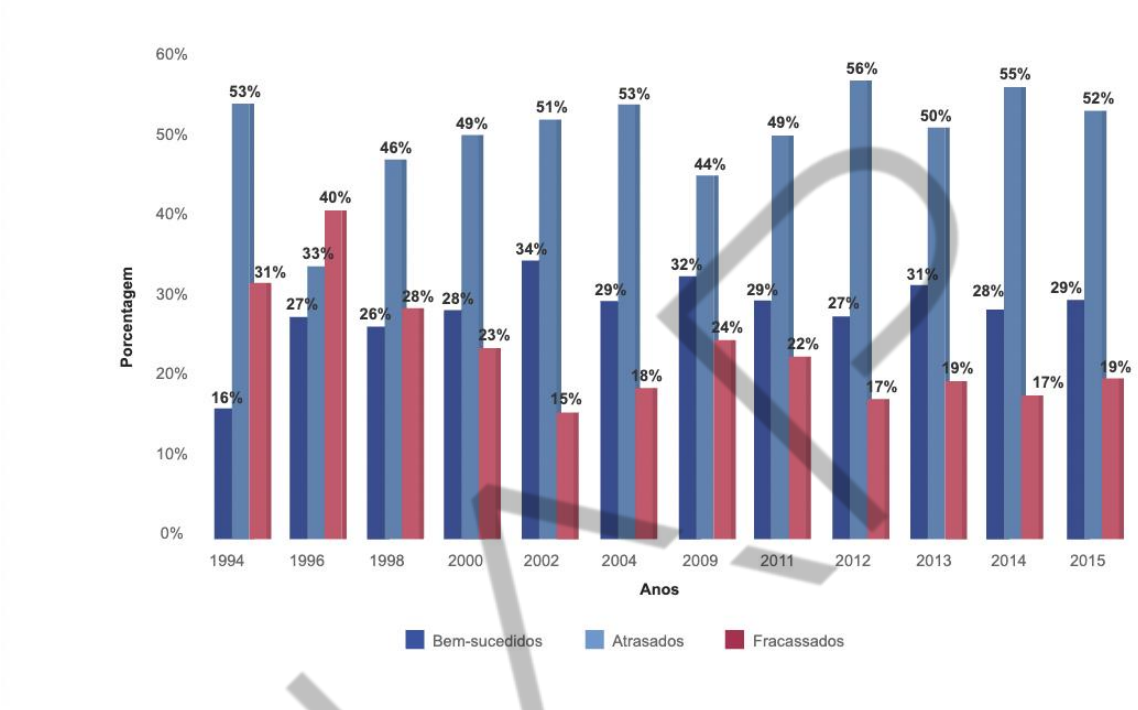


Figura 38 – Resultado dos projetos de sistemas de *software*
Fonte: Standish Group (2009)

O Standish Group foi um marco na história dos estudos de falhas de projetos de sistemas de *software*, responsável pela publicação dos relatórios Chaos, desde 1994.

De acordo com o Standish Group (2009), as principais causas de falhas nos projetos estão associadas a dificuldades com os seguintes temas: ausência de apoio da alta gerência, comportamento ao trabalhar em grupo, envolvimento superficial do usuário, inexperiência do gerente de projeto, falta de equipe qualificada e dúvidas relacionadas às regras de negócio e ao escopo do projeto.

Podemos citar também como possíveis causas da crise do *software*: alteração nos requisitos; mudanças nas legislações, normas ou processos aos quais a empresa esteja subordinada; e metodologia de desenvolvimento. É possível que essas alterações impostas gerem a necessidade de se executarem mudanças no cronograma e no custo previsto para o desenvolvimento do *software*. Muitos projetos

de *software* nascem com o cronograma estabelecido antes mesmo de os principais requisitos serem definidos.

Um requisito representa uma característica do sistema ou a descrição de algo que o sistema deve executar para satisfazer as necessidades das partes interessadas.

2.3 Origem da engenharia de *software*

De acordo com Swebok (2004), o termo “engenharia de *software*” foi criado na década de 1960 e utilizado oficialmente em 1968 na Conferência sobre Engenharia de *Software* da Otan. Sua origem está relacionada a uma tentativa de contornar a crise do *software*, com a finalidade de dar um tratamento sistemático e controlado ao desenvolvimento de sistemas de *software* complexos.

A seguir, os principais marcos pertinentes à origem da engenharia de *software.*

Origem da engenharia de *software*:

- **1960**

Surgiu o termo engenharia de *software*.

- **1968**

Oficializou-se o conceito de engenharia de *software.*

- **1972**

Conferência da Otan apresentou um relatório sobre engenharia de *software.*

- **1976**

Foi criada uma comissão para dar origem aos padrões de engenharia de *software.*

- **1979**

Inicia-se o padrão IEEE 730 para uniformizar a preparação e o conteúdo da engenharia de *software.*

- **1990**

O IEEE 1.074, padrão internacional sobre os assuntos relacionados ao processo de desenvolvimento de software, é instaurado.

- **1995**

Cria-se a SO/IEC 12.207 com o título de padrão para os processos e ciclo de vida do *software*.

A IEEE Computer Society aprova a elaboração do Swebok, guia para ser a principal referência da engenharia de *software*.

Em 1972, a IEEE Computer Society publica, pela primeira vez, seu relatório sobre engenharia de *software*.

Em 1976, é fundada uma comissão dentro da IEEE Computer Society com a responsabilidade de desenvolver padrões de engenharia de *software*. A primeira norma foi o padrão IEEE 730, cujo objetivo era fornecer requisitos mínimos de uniformidade para a preparação e o conteúdo do planejamento de *software*, que foi concluído em 1979.

Da norma IEEE 730, resultou o desenvolvimento de outros temas pertinentes à engenharia de *software*: gerenciamento de configuração, testes, requisitos, *design*, verificação e validação de *software*.

No período de 1981 a 1985, surgiu a IEEE 1.002, padrão de taxonomia que descreve a forma e o conteúdo dos princípios de engenharia de *software*.

Em 1990, foi iniciado o planejamento para um padrão internacional focado em conciliar os pontos de vista do processo de *software*, por meio da IEEE 1.074. Esse padrão foi finalizado em 1995, com a designação da ISO/IEC 12.207, cujo título é: padrão para os processos do ciclo de vida do *software*.

Por meio da ISO/IEC 12.207, o conselho de tutores da IEEE Computer Society aprovou a proposta apresentada em 1993 para a elaboração do Swebok, guia criado pelo Institute of Electrical and Electronics Engineers (IEEE), conhecido como entidade de especificação de padrões, para ser a principal referência à engenharia de *software*.

2001

Um manifesto foi publicado em 2001, quando Kent Beck e dezesseis outros notáveis desenvolvedores, produtores e consumidores de *software*, conhecidos como

Aliança Ágil, assinaram o Manifesto para o Desenvolvimento Ágil de *Software*, cujo objetivo era apontar novas direções na execução de projetos.

Os *frameworks* de gerenciamento de projetos ágeis surgiram em função da criação de processos de desenvolvimento ágeis, como o Extreme Programming (XP). O objetivo é adequar a gestão aos princípios ágeis de construção, tais como: cliente participativo e colaborativo, mudanças frequentes e entregas constantes.

A base para a elaboração desse manifesto considera que os requisitos não são estáveis durante todo o projeto, e as mudanças ocorrerão em qualquer cenário, por isso as execuções sequenciais das fases de um projeto não são tão previsíveis como gostaríamos que fossem. Isso posto, ficam definidos quatro fundamentos básicos que guiam o manifesto:

- Indivíduos e interações, em vez de processos e ferramentas.
- *Software* funcionando, em vez de documentação abrangente.
- Colaboração do cliente, em vez de negociação de contratos.
- Resposta às modificações, em vez de seguir um plano.

Esses fundamentos são a base de todos os *frameworks* ágeis, estabelecendo o alicerce sobre o qual são desenvolvidos os padrões de agilidade: foco no time, cliente fazendo parte do projeto, entregas durante o ciclo do projeto e mudanças são bem-vindas em qualquer momento do projeto.

3 ÁREAS DE CONHECIMENTO DA ENGENHARIA DE *SOFTWARE*

O Swebok (2004) organizou dez áreas de conhecimento com o objetivo de estabelecer um limite para a engenharia de *software*.

Independentemente da área de conhecimento, a engenharia de *software* tem elementos fundamentais, tais como: métodos, ferramentas e processos, conforme destaca a Figura “Elementos da engenharia de software”.

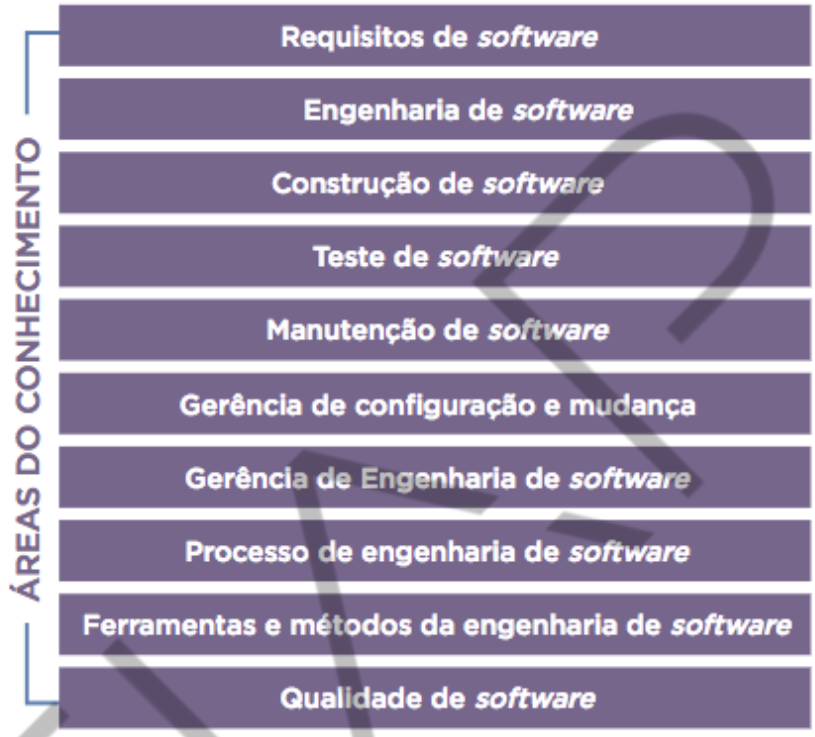


Figura 39 – Áreas de conhecimento da engenharia de *software*
Fonte: Swebok (2004), adaptado pelas autoras (2016)

Figura 40 – Elementos da engenharia de *software*
Fonte: Elaborado pelas autoras (2016)

Os processos representam as ações realizadas, ou seja, a aplicação de métodos e ferramentas para se obter um resultado específico, e definem a sequência de métodos que serão executados bem como quais ferramentas serão disponibilizadas.

Os métodos podem ser entendidos como maneiras de fazer, quais roteiros serão seguidos para a realização de determinadas tarefas. Exemplos: atividades de planejamento, levantamento de requisitos, *design*, definição da arquitetura de *software*, execução dos testes, entre outros.

As ferramentas representam o meio automatizado para a realização das tarefas e permitem que as atividades sejam executadas de forma mais eficiente e eficaz. Exemplo: *software* de modelagem de dados ou *softwares* para a implementação do *software*.

A equipe envolvida no desenvolvimento do sistema de *software* utiliza ferramentas, métodos e processos com o objetivo de melhorar a qualidade dos produtos de *software*. O foco é fazer uso de abordagens eficientes e eficazes para gerar soluções efetivas às necessidades das partes interessadas.

3.1 Visão geral sobre a engenharia de *software*

Conforme Pfleeger (2004), construir um sistema é como construir uma casa.

As figuras abaixo exemplificam o processo de desenvolvimento de um sistema de *software* de maneira semelhante à construção de uma casa.

Figura 41 – Levantamento das necessidades do cliente
Fonte: Elaborado pelas autoras (2016)

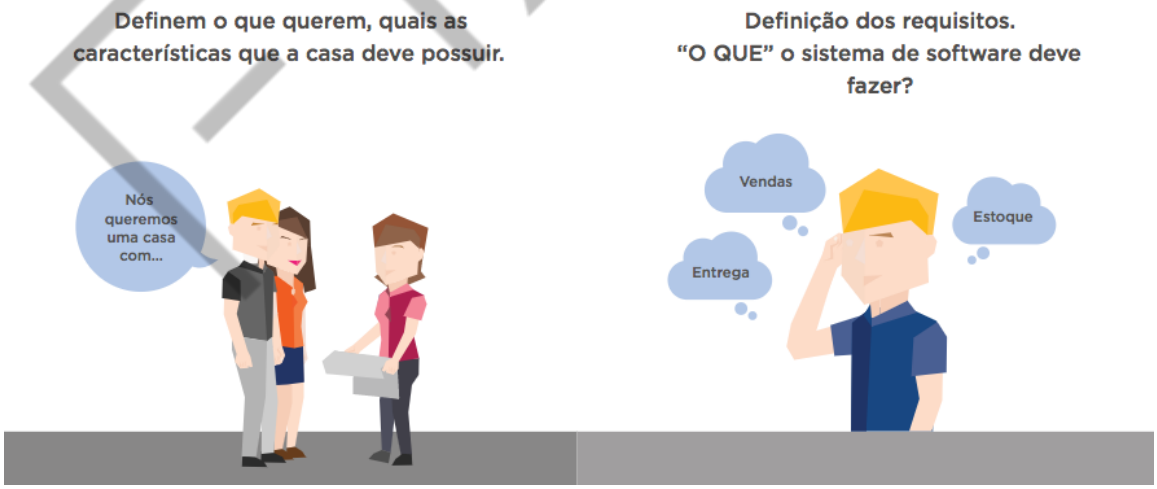


Figura 42 – Análise das necessidades do cliente
Fonte: Elaborado pelas autoras (2016)

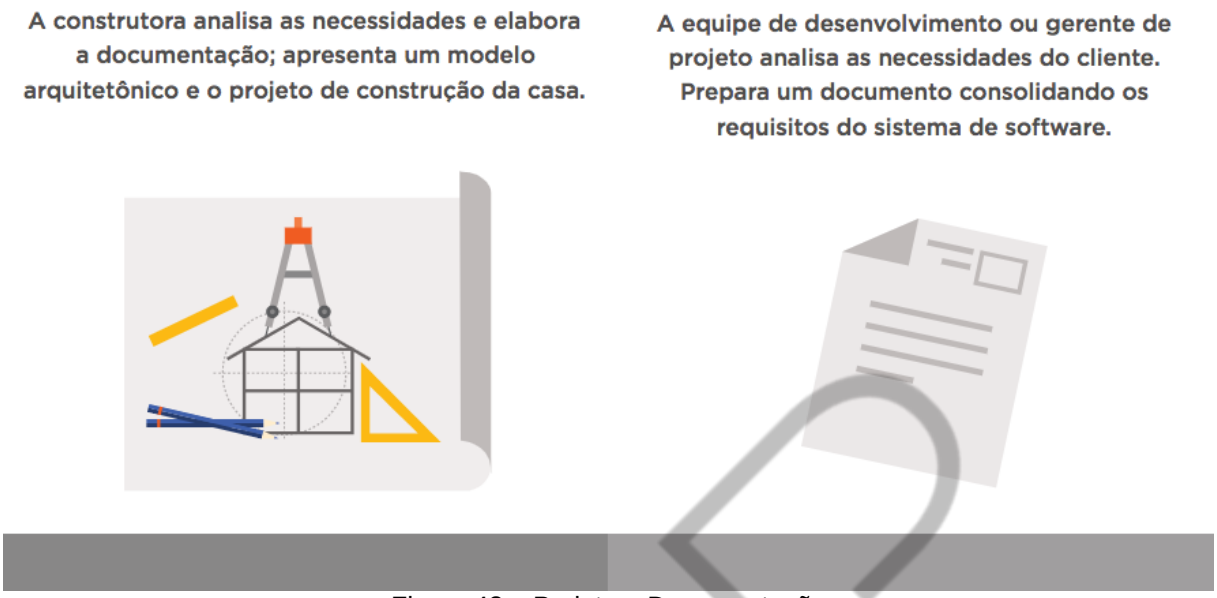


Figura 43 – Projeto – Documentação
Fonte: Elaborado pelas autoras (2016)

Figura 44 – Ateste, construção e implementação
Fonte: Elaborado pelas autoras (2016)

Figura 45 – Mudanças das necessidades
Fonte: Elaborado pelas autoras (2016)

Figura 46 – Testes
Fonte: Elaborado pelas autoras (2016)

Figura 47 – Entrega
Fonte: Elaborado pelas autoras (2016)

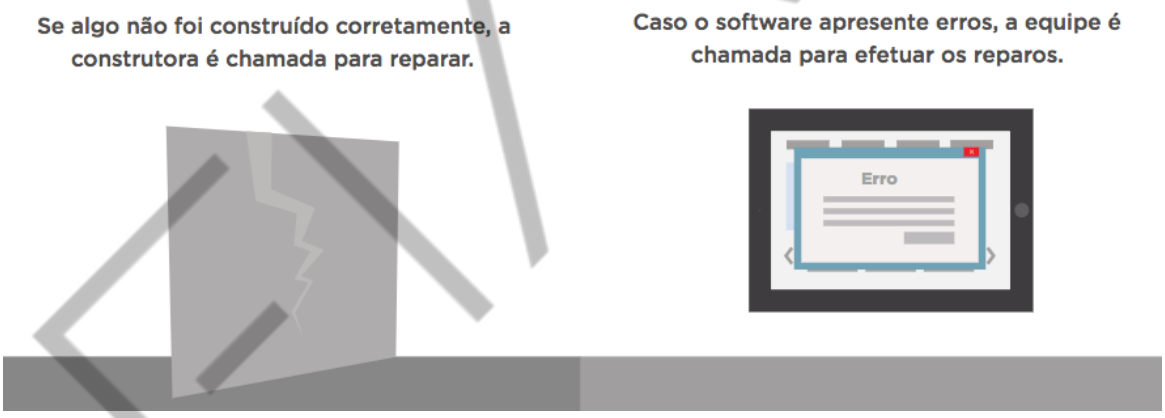


Figura 48 – Manutenção
Fonte: Elaborado pelas autoras (2016)

3.2 Desafios encontrados pela engenharia de *software*

No desenvolvimento de *software*, em geral, encontram-se problemas do tipo:

- Recursos destinados aos projetos tornam-se insuficientes.
- Custos dos produtos são altos.
- Soluções propostas não agradam às partes interessadas.
- Custos dos *softwares* são maiores que o custo do *hardware*.

- Manter um *software* custa mais que desenvolvê-lo.

Consequentemente, a engenharia de *software* foca na missão de superar os desafios:

- Reduzir custo.
- Seguir o cronograma de acordo com as expectativas.
- Melhorar a qualidade do *software*.
- Documentar; todos os detalhes devem ser escritos de modo que qualquer parte interessada possa entender.
- Adaptar as alterações sugeridas e/ou necessárias no tempo e no custo adequados.
- Atender às necessidades do cliente.
- Nortear o desenvolvimento do sistema de *software* de acordo com as mudanças tecnológicas.

A criação de um *software* é uma atividade de alta complexidade. É fundamental que se conheçam as etapas que comporão tal processo. A adoção de processos bem definidos melhora a visibilidade da construção do *software*.

No desenvolvimento de um *software,* é desejável que se siga um processo que reduza, dentro do possível, o desperdício de tempo e de custo e maximize com qualidade os resultados obtidos.

Conforme Pressman (2009), todo *software* é construído com base em um modelo de processo, também conhecido como ciclo de vida. O ciclo de vida de um *software* compreende um conjunto de etapas que envolvem métodos, ferramentas e procedimentos para a obtenção do produto ou serviço de *software* desejado.

Quando falamos em desenvolvimento de software em ambiente de desenvolvimento ágil, produtos e serviços em geral passam por um processo de mudanças. Os projetos tornam-se cada vez mais complexos, sujeitos a constantes alterações e com prazos cada vez mais restritos para atender à demanda do mercado.

O gerenciamento de projetos acompanha essa transformação do ambiente. Se de um lado, o método tradicional é centrado em um processo sequencial e dirigido à

qualidade dos artefatos, do outro, o método ágil tem foco na rapidez de adequação às mudanças e conta com elementos como: cliente integrado à equipe e prazos curtos de desenvolvimento (BOEHM; TURNER, 2002). Esses e outros aspectos trazem muitas discussões e controvérsias entre os especialistas em gestão.

O método de gerenciamento de projetos tradicional é criticado por sua falta de flexibilidade em se adaptar a essa nova realidade, enquanto o gerenciamento ágil vem ganhando popularidade justamente por sua capacidade de adequação às situações voláteis do ambiente. Porém também recebe críticas pelo grau de informalidade implícita em seu processo.

Ambas correntes acreditam estar corretas em seus conceitos e na aplicação de seus métodos, apresentando vantagens e desvantagens para justificarem sua utilização.

No manifesto ágil, considera-se que os requisitos não são estáveis durante todo o projeto, que as mudanças vão ocorrer em qualquer cenário e que as execuções sequenciais das fases de um projeto não são tão previsíveis como gostaríamos que fossem.

A base de todos os *frameworks* ágeis, estabelecendo o alicerce sobre o qual são desenvolvidos os padrões de agilidade, é composta de: foco no time, cliente fazendo parte do projeto, entregas durante o ciclo do projeto e mudanças bem-vindas em qualquer momento do projeto.

As etapas ou o ciclo de vida do processo de desenvolvimento de software são destacados na Figura “Ciclo de vida do processo de desenvolvimento de software”.

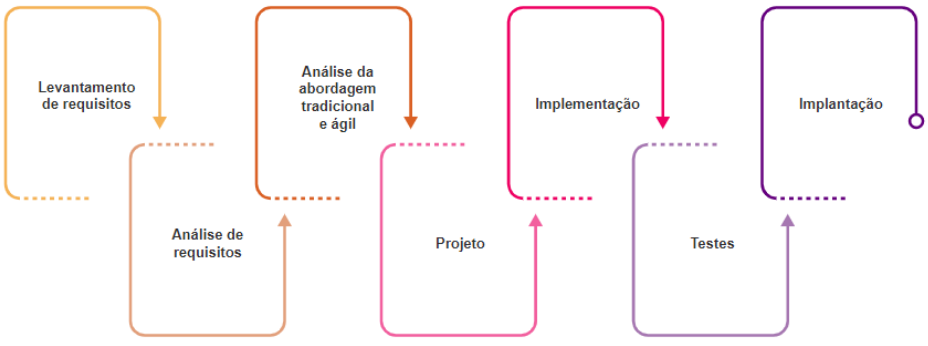


Figura 49 – Ciclo de vida do processo de desenvolvimento de *software*
Fonte: Sommerville (2011)

EXERCÍCIO – PESQUISA DA CONCORRÊNCIA

Vamos ao trabalho? A primeira coisa que lhe é solicitada é uma análise da concorrência: pesquise por dois aplicativos móveis para a área de finanças. Relacione pelo menos três vantagens e três desvantagens de cada um ao utilizá-los.

Insira sua resposta no canal da atividade no Microsoft Teams com as informações que você analisou dos aplicativos.

Insira os dados da sua pesquisa na plataforma **Microsoft Teams** no canal relacionado ao exercício.

Você encontrará o canal **"Fase 1 – Cap 2 – Pesquisa da concorrência"** no grupo da sua turma, no qual deverá inserir as informações que pesquisou.

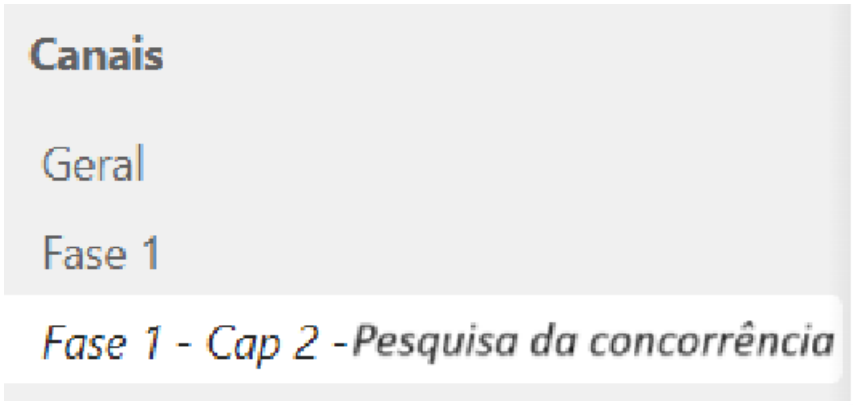
Tutorial de acesso ao canal:

Acesse “Equipes” no Teams:

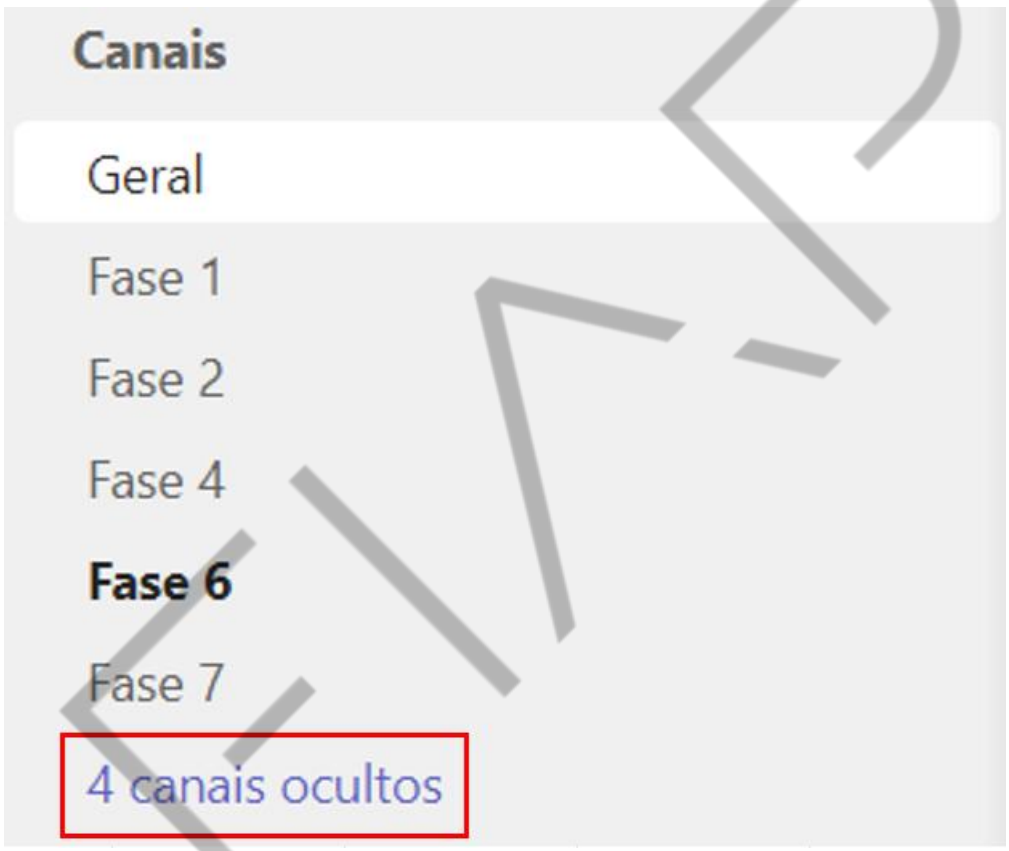
Acesse a equipe de sua turma:

1TDSO

Acesse o canal da atividade **Fase 1 – Cap 2 – Pesquisa da Concorrência:**



Atenção: O canal pode estar em “Canais ocultos”, Exemplo:



EXEMPLO FORMATO DE RESPOSTA:

APLICATIVO X

Pontos Positivos	Pontos Negativos
1 -	
2 -	
3 -	

APLICATIVO Y

Pontos Positivos	Pontos Negativos
1 -	
2 -	
3 -	

Qualquer dúvida, escreva para o tutor da sua turma.

REFERÊNCIAS

ABRAHAMSSON, P. et al. New Directions on Agile Methods: A Comparative Analysis. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 25, 2003, Portland. **Proceedings of…** Portland: IEEE, 2003.

ANTONIONI, J. A. **Qualidade em "Software"**: Manual de Aplicação da ISO-9000. 2. ed. São Paulo: Makron Books, 1995.

BECK, K. **Extreme Programming Explained**: Embrace Change. Boston: Addison-Wesley, 1999.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 2. ed. São Paulo: Campus, 2003.

BOEHM, B. W.; TURNER, R. **Balancing agility and discipline**: a guide for the perplexed. Boston: Addison-Wesley, 2002.

BOOCH, G.; RUMBAUGH, J.; JACBSON, I. **UML – Guia do Usuário**. 2. ed. São Paulo: Campus, 2006.

CESNE, E.; JUMBERT, M. G.; SANDIVK, K. B. Drones como Veículos para a Ação Humanitária: Perspectivas, Oportunidades e Desafios. **Revista Conjuntura Austral.** Porto Alegre, v. 7, pp. 45-60. Disponível em: <http://seer.ufrgs.br/index.php/ConjunturaAustral/article/view/60267/36712>. Acesso em: 14 dez. 2023.

CORDEIRO, M. A. **Uma Ferramenta Automatizada de Suporte ao Processo de Gerenciamento de Requisitos.** Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2002/2002_marco_aurelio.pdf>. Acesso em: 14 dez. 2023.

GADELHA, J. **A Evolução dos Computadores**. Disponível em: <http://www2.ic.uff.br/~aconci/evolucao.html>. Acesso em: 14 dez. 2023.

GUEDES, G. T. A. **UML 2 – Uma Abordagem Prática**. 2. ed. São Paulo: Novatec, 2011.

IEEE 610.12-1990. **IEEE "Standard Glossary of Software" Engineering Terminology**. IEEE, 1990.

IIBA. **Um Guia para o Corpo de Conhecimento de Análise de Negócios**, Guia BABOK, versão 2.0. Traduzido pelo IIBA Capítulo São Paulo, 2011.

ISO, IEC 42010:2007. **Systems and Software Engineering Recommended Practice for Architectural Description of Software – Intensive Systems**. ISO & IEC, 2007.

KERZNER, H. **Gerenciamento de projetos**: uma abordagem sistêmica para planejamento, programação e controle. São Paulo: Edgard Blucher, 2011.

KOSCIANSKI, A.; SOARES, M. S. **Qualidade de "Software"**. São Paulo: Novatec, 2006.

KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering:** Processes and Techniques. 1. ed. Michigan: John Wiley, 1998.

LAIA, W. **A Evolução do Software**. 2013. Disponível em: <http://www.tiespecialistas.com.br/2013/03/a-evolucao-do-software>. Acesso em: 14 dez. 2023.

LARMAN, C. **Utilizando UML e Padrões**. 3. ed. Porto Alegre: Bookman, 2007.

PATI, C. Como os jovens vão enfrentar a quarta revolução industrial. 2016. **Exame**. Disponível em: <https://exame.com/carreira/como-os-jovens-vao-enfrentar-a-quarta-revolucao-industrial/>. Acesso em: 14 dez. 2023.

PFLEEGER, S. L. **Engenharia de "Software" – Teoria e Prática**. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de "Software"**. 7. ed. São Paulo: Makron Books, 2011.

O QUE é a Quarta Revolução Industrial. **Forbes Brasil,** 2016. Disponível em: <http://www.forbes.com.br/fotos/2016/02/o-que-e-a-quarta-revolucao-industrial>. Acesso em: 14 dez. 2023.

REZENDE, D. **Engenharia de Software e Sistemas de Informação**. 2. ed. Rio de Janeiro: Brasport, 2005.

SOFTWARE Engineering Body of Knowledge (Swebok). 3. ed. **IEEE**, 2004. Disponível em: <https://www.computer.org/web/swebok/v3>. Acesso em: 14 dez. 2023.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. Trad. Maurício de Andrade. São Paulo: Pearson, 2011.

SOTILLE, M. **Gerenciamento de Projetos na Engenharia de Software**. 2014. Disponível em: <http://www.pmtech.com.br/artigos/Gerenciamento_Projetos_Software.pdf>. Acesso em: 14 dez. 2023.

STANDISH GROUP. **The Standish Group Report Chaos**. 2009. Disponível em: <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Acesso em: 14 dez. 2023.

VELLOSO, F. C. **Informática:** conceitos básicos. 9. ed. Rio de Janeiro: Elsevier, 2014.