UNINOVE

Introdução à engenharia de software

APRESENTAR OS ANTECEDENTES HISTÓRICOS DA ENGENHARIA DE SOFTWARE, COMENTAR SOBRE SEUS PRINCIPAIS CONCEITOS DE SOFTWARE, DESCREVER O QUE CARACTERIZOU A CRISE DO SOFTWARE E EXEMPLIFICAR ALGUMAS DAS SUAS PRINCIPAIS ÁREAS DE APLICAÇÃO.

Antecedentes históricos da engenharia de software

Vamos, inicialmente, fazer um breve histórico dos principais fatos que ocorreram no mundo da tecnologia que justificaram a criação deste ramo da computação. É interessante você notar que algumas décadas atrás, não se tinha ideia da importância que os computadores e, em especial, o software (programas de computadores), iriam ter e como iriam afetar profundamente a vida de todos nós.

A denominação "engenharia de software" surgiu em 1968, em uma conferência organizada para discutir a chamada "crise do software". Para sermos mais precisos, o termo foi criado no início da década de 1960, tendo sido utilizado oficialmente na *NATO Conference on Software Engineering* (Conferência da OTAN sobre Engenharia de Software), que aconteceu na cidade de Garmisch, Alemanha.

Nesta ocasião, todos estavam preocupados em contornar os efeitos da crise do software e buscar maneiras de dar um tratamento de engenharia mais sistemático e controlado, ao desenvolvimento de sistemas de software complexos.

Desta forma, como o principal objetivo dessa conferência foi estabelecer práticas mais maduras para o processo de desenvolvimento de software, ela é considerada como o evento que deu origem à disciplina de Engenharia de Software.

A crise do software

A "crise do software" foi um termo criado para descrever as dificuldades enfrentadas no desenvolvimento de software no final da década de 1960. A complexidade dos problemas, a ausência de técnicas bem estabelecidas e a crescente demanda por novas aplicações começavam a se tornar um problema sério.

Um dos fatores geradores dessa crise foi a introdução de computadores com maior poder computacional. Esses novos computadores tornavam viáveis softwares bem maiores e mais complexos que os sistemas existentes.

A experiência inicial de construção desses sistemas mostrou que uma abordagem informal de desenvolvimento de software não era suficiente, pois

- Projetos importantes sofriam atrasos (às vezes, de alguns anos).
- Os custos eram muito maiores do que os inicialmente estimados.
- Os softwares desenvolvidos não eram confiáveis e eram de difícil manutenção, ou seja, a qualidade deixava a desejar.

Novas técnicas e novos métodos eram necessários para controlar a complexidade inerente aos grandes sistemas de software. A criação e adoção da engenharia de software por parte da comunidade de desenvolvedores foi a melhor maneira encontrada para solucionar esses problemas.

Características do software

É importante você saber que o software é um produto que exige uma visão ampla, portanto, o controle da qualidade não pode ser uma atividade secundária, devendo estar presente desde o início de seu desenvolvimento até a análise final de entrega.

É interessante a gente comparar o software com produtos de hardware porque isso auxilia na compreensão das diferenças existentes entre eles e enfatiza as características inerentes a um software.

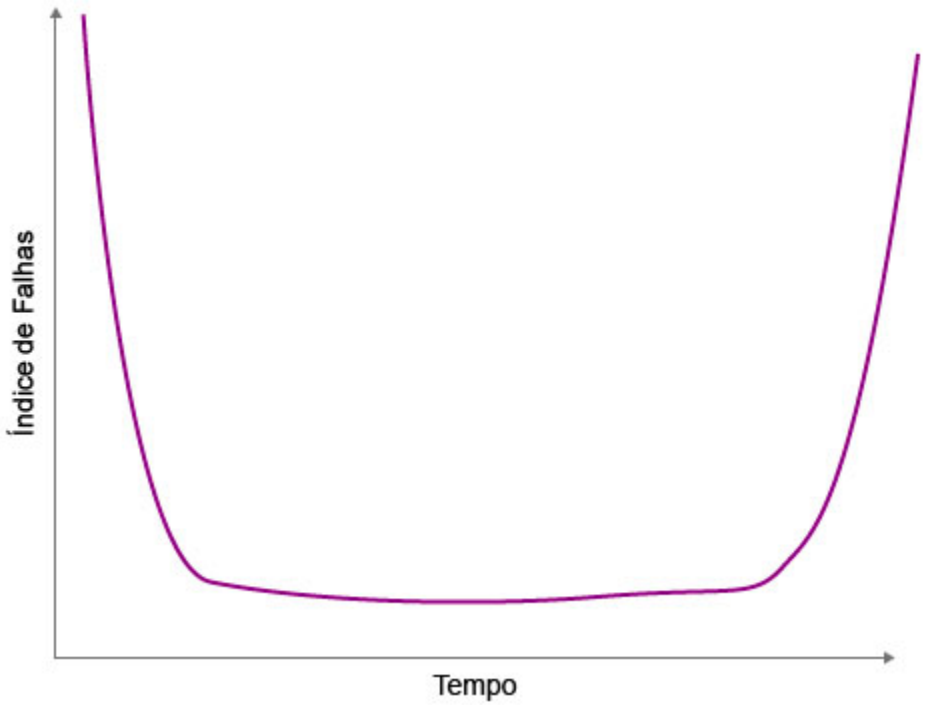
Bom, vamos relacionar algumas diferenças básicas entre o processo de desenvolvimento de software em relação ao hardware (PRESSMAN, 2010):

- O processo criativo do hardware gera algo físico (por exemplo, placas de circuitos). O desenvolvimento de software resulta em um elemento pertencente a um sistema lógico, intangível.
- O software geralmente é desenvolvido sob medida, ao contrário do hardware, no qual o projetista tem acesso a componentes existentes que executam tarefas definidas. O projetista do software nem sempre terá acesso a módulos prontos para utilização e quando o faz, pode elevar o risco do produto devido a questões de segurança.
- Os custos do software estão concentrados no desenvolvimento e não no processo de manufatura. Isso significa que não pode ser gerido como projeto de manufatura.
- Ao longo do tempo, o produto de software não se desgasta, mas se deteriora em função da introdução de erros oriundos de atividades de manutenção ou evoluções implícitas no processo que devem ser reconsideradas no modelo original.

Desta forma, o software sofre deterioração ocasionada por diversos fatores, sendo uma característica peculiar do produto.

Segundo Pressman (2010), no caso do hardware, temos um alto índice de falhas no início do seu ciclo de vida ocasionadas por defeitos de fabricação e projeto. Posteriormente os defeitos são corrigidos dando estabilidade nas falhas ou mantendo-a em um nível muito baixo e suportável para a estrutura.

Já no final do ciclo de vida do produto podem surgir problemas relacionados ao envelhecimento, acúmulo de poeira, vibração, abuso, temperaturas extremas, entre outros. Esse processo pode ser visto no gráfico apresentado na figura 1.



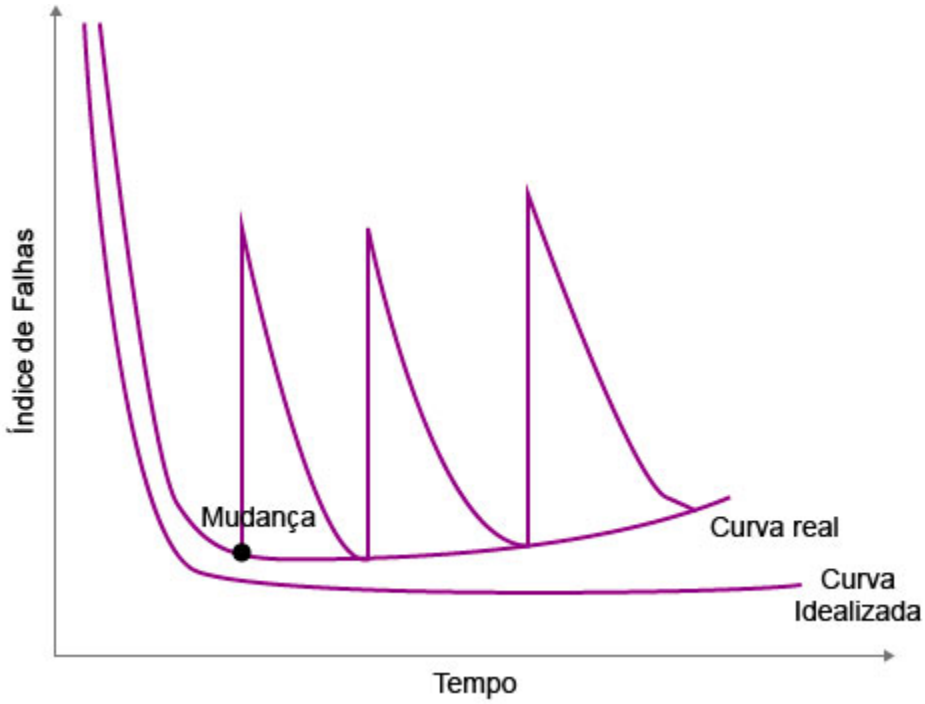
Mas você deve estar se perguntando: e com o software, como é que é?

Diferentemente da curva teórica de falhas do hardware, a do software leva em conta que ele não sofre processos de envelhecimento como o hardware, pois ele não é algo físico.

No início do ciclo de vida do software, teremos problemas (bugs) que serão ajustados no decorrer do seu desenvolvimento e se estabilizarão gerando uma tendência de achatamento da curva. Note que esta é apenas uma teoria, já que a curva real do índice de falhas de um software considera o processo de manutenção e mudanças.

Durante o processo de refinamento do produto ou mudanças, aumenta-se consideravelmente a probabilidade de inserção de novos erros, gerando picos na curva de falhas.

As sucessivas alterações do software tendem a introduzir mais erros antes da estabilização dos erros de alterações anteriores, ocasionando a tendência crescente do índice de falhas, conforme pode ser visto na figura 2.



Se você for projetar e construir hardware, você poderá utilizar catálogos de componentes digitais. Cada circuito integrado tem um código de componente, uma função definida e validada, uma interface bem delineada e um conjunto padrão de integração.

Depois que você selecionar cada componente, ele poderá ser requisitado do estoque e aí você poderá utilizá-lo em diferentes projetos de hardware com alta confiabilidade. No mundo do software, isso é algo que está apenas começando a ser utilizado numa escala mais ampla, apesar de existirem alguns casos antigos de "reuso", como as bibliotecas de subrotinas científicas.

Atualmente, a visão de reuso foi ampliada para abranger não apenas algoritmos consagrados, mas também estruturas de dados, interfaces gráficas e diferentes classes e componentes orientados a objetos.

Aplicações do software

O software pode ser aplicado em qualquer situação para que um conjunto previamente especializado de procedimentos (um algoritmo) tenha sido definido.

Vamos ver algumas áreas de aplicações de software. Com certeza você já deve ter tido contato com pelo menos uma delas.

- Software de sistema: coleção de programas para servir outros programas (compiladores, editores, utilitários para gestão de arquivos, componentes de sistemas operacionais etc). Esses tipos de programas se caracterizam por manterem uma interação intensa com o hardware.
- Software de tempo real: monitora, analisa e controla eventos do mundo real à medida em que eles ocorrem. Exemplos: sistema de radar aeroespacial, teclado que gera inputs de teclas pressionadas para

um sistema microprocessado, sensor de temperatura que gera um input para um microcontrolador, sistema de controle de voos, dentre outros.

- Software comercial: maior área de aplicação de software. Estão na categoria de software de aplicação que resolvem necessidades específicas do negócio, como controle de reservas de voos, sistemas de pagamento, de controle de estoque etc.
- Software científico e de engenharia: caracterizado por algoritmos que "processam números" e realizam operações matemáticas e cálculos mais complexos (astronomia, análise automotiva de tensões, manufatura automatizada, previsão de tempo, simuladores, etc).
- Software embutido: produtos "inteligentes". São programas armazenados em ROMs – *Read Only Memories*, como controle de teclado em um forno de microondas, funções digitais em um automóvel – controle de combustível, mostradores do painel, sistemas de frenagem, etc.
- Software para web: as páginas da web recuperadas por um browser constituem softwares que incorporam instruções executáveis na forma de scripts, permitindo a inclusão de elementos dinâmicos, animações, acesso a banco de dados e diversas outras características.

- Software para computadores pessoais: engloba uma enorme lista de aplicativos para desktops, notebooks, etc, como processadores de texto, planilhas, aplicações gráficas, aplicações multimídia, etc.
- Software para inteligência artificial: faz uso de algoritmos não numéricos para resolver problemas complexos, como sistemas de reconhecimento de padrões (de imagem e de voz), redes neurais artificiais, sistemas de controle de robôs, etc.

Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.
EXERCÍCIO (https://ead.uninove.br/ead/disciplinas/web/_g/pdsoft80_100/a01ex01_pdsoft80_100.htm)
Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.
EXERCÍCIO (https://ead.uninove.br/ead/disciplinas/impressos/_g/pdsoft80_100/a01ex02_pdsoft80_100.pdf)

REFERÊNCIA

PRESSMAN, R. S. *Engenharia de software*. 7. ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, Ian. *Engenharia de software*. São Paulo: Addison-Wesley, 2007.