Evolução de Software

Marco Antônio Pereira Araújo

Introdução

- "Programas, assim como as pessoas, envelhecem" (Parnas, 1994)
- Manutenção: a correção de erros, e a implementação de modificações necessárias para permitir a um sistema existente a executar novas tarefas, e para executar antigas sob novas condições
- Evolução: o comportamento dinâmico dos sistemas como eles são mantidos e expandidos ao longo de seu ciclo de vida
- Termos comumente utilizados: evolução, deterioração, decaimento, envelhecimento, rejuvenescimento

Introdução

- Sistemas de software estão crescendo rapidamente em termos de funcionalidade, complexidade, tamanho e estrutura (Lehman, 1998)
- Apenas 2% dos estudos experimentais focam em manutenção, a despeito de que publicações reportam que ao menos 50% do esforço de software é dedicado a esta atividade (Kemerer, 1999)
- É necessário antecipar os caminhos em que o software sofre mudanças, assim pode-se modificá-lo mais facilmente para acomodar estas necessidades. Entretanto, antecipar-se às mudanças não é uma tarefa fácil, uma vez que existem muitas razões pelas quais sistemas mudam (Pfleeger, 1998)

Introdução

- Evolução de Software
  - O termo evolução reflete um processo de mudança progressiva em atributos ou elementos que constituem a entidade em evolução (LEHMAN e RAMIL 2002).
  - Na Engenharia de Software, algumas vezes o termo evolução pode ser confundido com manutenção (LEHMAN 2000).
  - De acordo com (IEEE/EIA 12207), manutenção é um dos principais processos do ciclo de vida de desenvolvimento onde o produto passa por modificações, no código e na documentação associada, devido a problemas ou necessidade de melhorias. O objetivo é modificar o produto de software já existente preservando sua integridade.

O que é decaimento?

- Código decai se é mais difícil modificá-lo do que deveria ser (Eick et al, 1999)
- Questões chave:
  - Custo da mudança, que é efetivamente apenas o custo dos desenvolvedores
  - Intervalo para completar a mudança, o tempo requerido
  - Qualidade do software modificado

Causas para o Decaimento de Software

1. Arquitetura inapropriada que não suporta as mudanças ou abstrações requeridas para o sistema
2. Violações dos princípios originais do projeto, que pode forçar mudanças não previstas e violar as suposições originais
3. Requisitos imprecisos, que podem impedir programadores de desenvolver códigos corretos, causando excessivo número de mudanças
4. Pressões de tempo, que podem levar desenvolvedores a produzir código de baixa qualidade ou fazer mudanças sem o entendimento do impacto no sistema

Causas para o Decaimento de Software

5. Ferramentas inadequadas de programação, isto é, indisponibilidade de ferramentas CASE
6. Ambiente organizacional, manisfestado, por exemplo, em baixa estima, rotatividade excessiva, inadequada comunicação entre os desenvolvedores, tudo que possa produzir frustração e prejudicar o trabalho
7. Variabilidade do programador, ou seja, programadores que não entendem ou fazem delicadas mudanças em código complexo escrito pelos colegas mais experientes
8. Processo de mudança inadequado, como a falta de controle de versões ou inabilidade para conduzir mudanças em paralelo

Sintomas de Decaimento de Software

1. Complexidade excessiva
2. Histórico de mudanças frequentes
3. Histórico de falhas
4. Mudanças amplamente dispersas
5. Impropriedades no código quando desenvolvedores propositadamente fazem mudanças que deveriam ter sido feitas de forma mais elegante ou eficiente
6. Interfaces numerosas

Fatores de Risco para Decaimento de Software

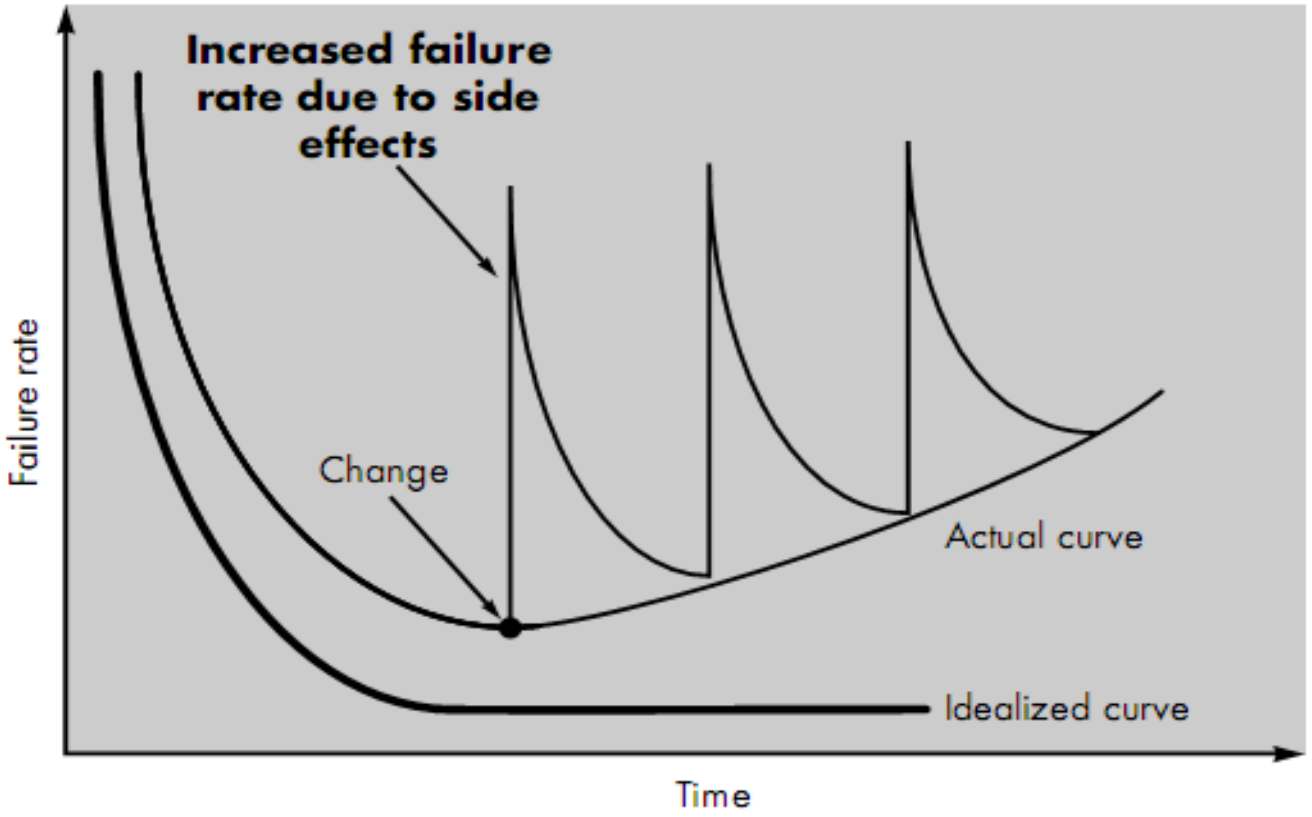
1. O tamanho de um módulo, em NCSL (número de linhas de código fonte não comentadas)
2. A idade de um código (intuitivo)
3. Complexidade inerente
4. Rotatividade ou reorganização (degradação da base de conhecimento), podendo aumentar a probabilidade de desenvolvedores inexperientes fazendo mudanças em código
5. Código portado ou reutilizado, originalmente desenvolvido em uma outra linguagem, para um sistema diferente ou para outra plataforma de hardware

Fatores de Risco para Decaimento de Software

6. Carga de requisitos, significando que o código tem funcionalidade extensiva e está sujeito a muitas restrições (dificuldades de entendimento e de implementação)
7. Desenvolvedores inexperientes, pela falta de conhecimento em desenvolvimento, falta de entendimento da arquitetura do sistema, e potencial para baixa habilidade em desenvolvimento

Introdução

- Decaimento de Software (PRESSMAN 2005)

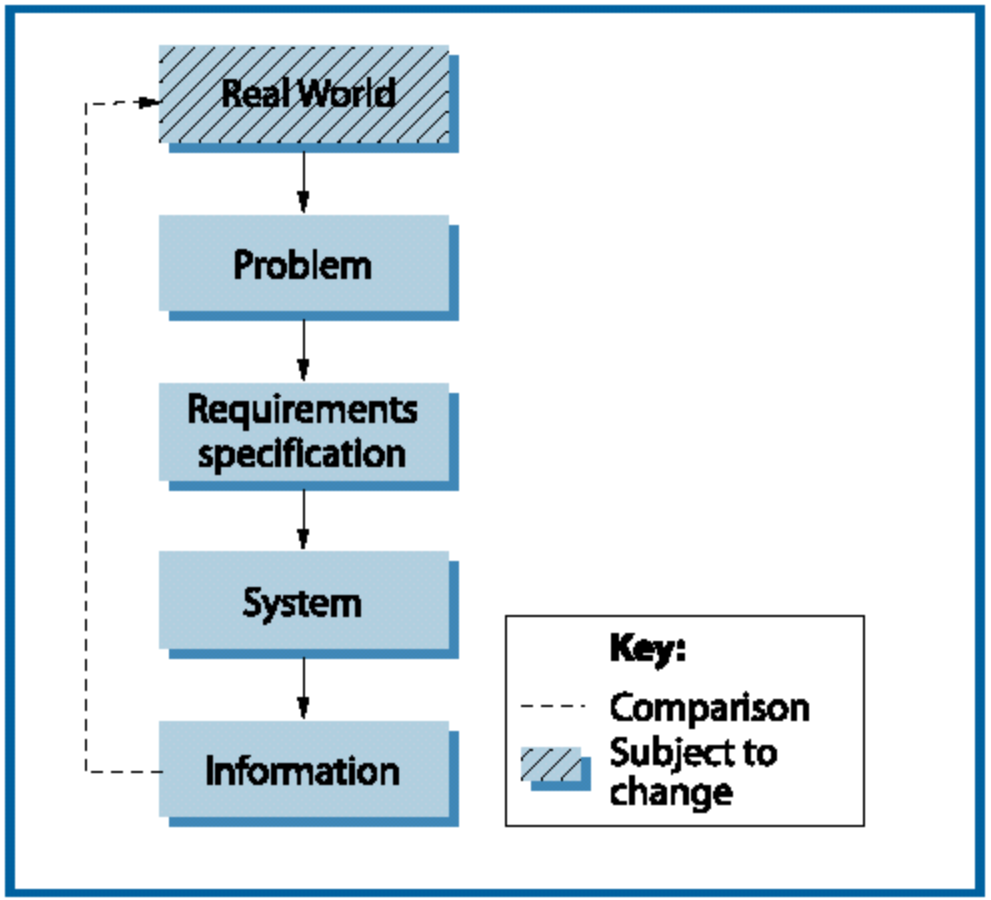


Leis de Evolução de Software

- O trabalho de Lehman ("Programs, Life Cycle and the Laws of Software Evolution", Proc. IEEE, IEEE,1980), é um dos precursores da área e descreve um sistema em termos dos caminhos em que se relaciona com o ambiente em que opera
- Vários trabalhos relatam a evolução de sistemas iniciando no início da década de 70 estudando a evolução do IBM OS/360, seguido de outros estudos inclusive com avaliação na indústria
- "As novas análises suportam, ou melhor, não contradizem, as leis de evolução de software, sugerindo que a abordagem da década de 70 para analisar as medidas de evolução de software é ainda relevante atualmente" (Lehman, 1998).

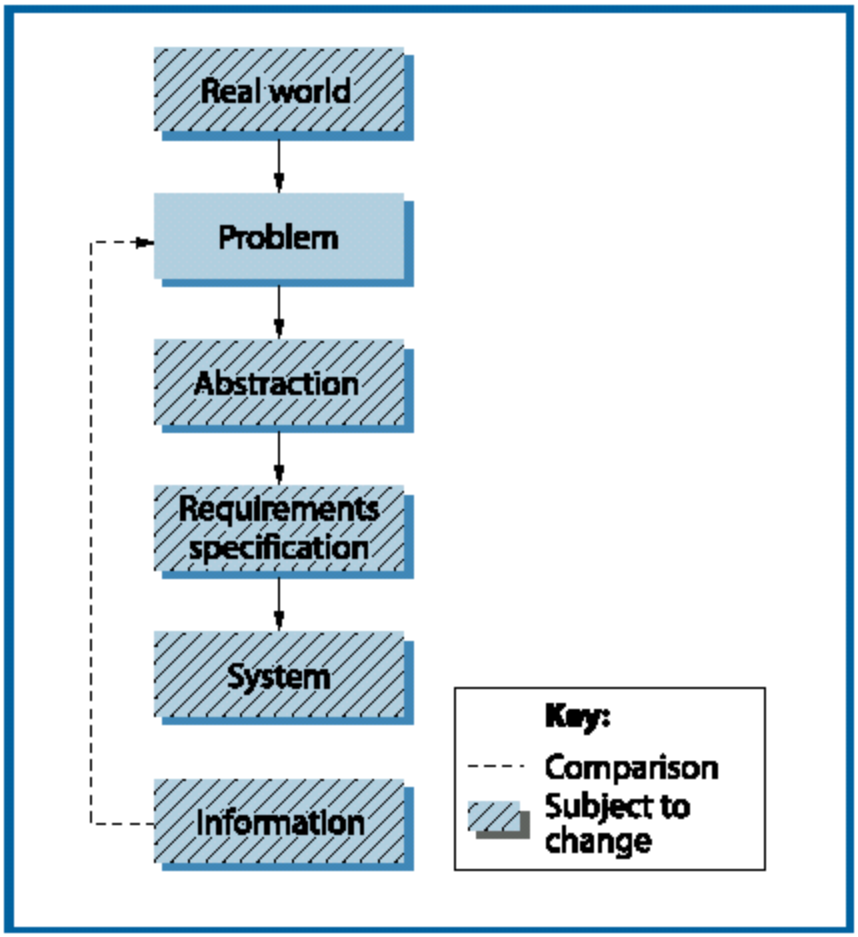
Classificação de Lehman

- S-Systems *(specification)*
  - O problema é bem definido
  - A solução é bem conhecida
  - Relacionado com o mundo real
  - Se o mundo real muda, o resultado é um problema completamente novo que deve ser especificado
  - São improváveis de mudar
  - Ex. Sistemas para operações em matrizes



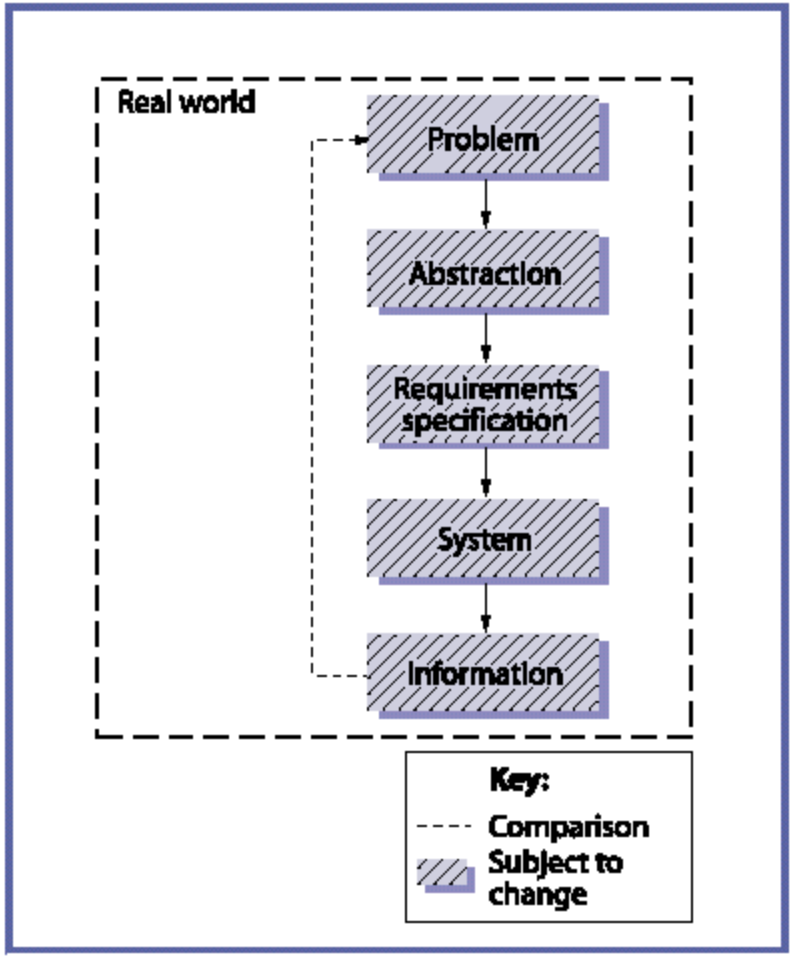
Classificação de Lehman

- P-Systems *(problem)*
  - Baseado em uma abstração prática do problema, ao invés de uma especificação completamente definida
  - É mais dinâmico que um S-System
  - A solução produz informação que é comparada com o problema
  - A solução depende em parte do analista que gerou os requisitos
  - São sujeitos à mudanças incrementais
  - Ex. Sistemas para jogo de xadrez



Classificação de Lehman

- E-Systems *(embedded)*
  - Incorpora mudanças naturais do mundo real
  - Embutido no mundo real e muda com ele
  - A solução é baseada no modelo do processo abstrato envolvido
  - O sistema é uma parte do mundo modelado
  - São provavelmente submetidos à mudanças quase constantes
  - Ex. Sistemas para a área financeira



Leis de Evolução de Software

- As Leis de Evolução de Software e seu desenvolvimento como a base para uma teoria de evolução de software, representa a maior contribuição intelectual e desafios para a comunidade de pesquisa em engenharia de software (Scacchi, 2003)
- Focam, primeiramente, no entendimento de como sistemas de software mudam ao longo do tempo
- Os conceitos de classificação de sistemas de Lehman, acompanhado de suas leis, provêem um vocabulário amplamente aceito para a discussão da natureza da mudança de software
- Através destas idéias, pode-se projetar sistemas para serem flexíveis, planejar manutenções, além de melhor entender e controlar o desenvolvimento de software, mais do que simplesmente reagir aos problemas que acontecem (Pfleeger, 1998)

Leis de Evolução de Software

I - Mudança Contínua

- Um produto em uso ou está em mudança constante ou se torna progressivamente menos útil
- O processo de decaimento continua até que seja mais barato substituir o sistema com uma versão recriada

II - Incremento da Complexidade

- A mudança contínua introduz continuamente complexidade no produto deteriorando a estrutura do mesmo
- Se não for desenvolvida nenhuma atividade explícita do controle da complexidade, a manutenção do produto deixa de ser possível e este torna-se inútil

Leis de Evolução de Software

III - Auto-Regulação

- Lei Fundamental da Evolução do Produto
- O processo de evolução do produto é uma dinâmica auto regulada com tendências estatísticas determináveis e invariâncias
- Sistemas de software exibem comportamentos regulares e tendências que podem ser medidas e previstas

IV - Conservação da Estabilidade Organizacional

- A taxa de atividade global em um produto em evolução é estatisticamente invariante ao longo de seu ciclo de vida
- Atributos organizacionais, como produtividade, não exibem grandes flutuações
- Recursos e resultados alcançam um nível ótimo, e adicionar mais recursos não muda significativamente os resultados

Leis de Evolução de Software

V - Conservação da Familiaridade

- Durante o ciclo de vida de um produto o conteúdo de cada versão é estatisticamente invariante
- O conteúdo de sucessivas versões de programas em evolução (mudanças, adições, exclusões) é estatisticamente invariante
- Após um tempo, o efeito de versões de manutenções sucessivas faz pouca diferença na funcionalidade geral

VI - Crescimento Contínuo

- O conteúdo funcional de um sistema deve ser continuamente incrementado para manter a satisfação do usuário ao longo do ciclo de vida

Leis de Evolução de Software

VII - Declínio da Qualidade

- A qualidade de um sistema entrará em declínio a menos que seja rigorosamente mantida e adaptada às mudanças do ambiente operacional

VIII - Feedback do Sistema

- O processo de evolução de um sistema constitui em feedback de multi-nível, multi-interação e multi-agente do sistema e deve ser tratado de forma a alcançar significativas melhorias

Métricas de Evolução em Manutenção de Software

- Sequência de número de versões
- Tamanho do sistema (subsistemas, módulos, arquivos, etc.)
- Elementos tratados (um módulo com n mudanças independentes é contado n vezes)
- Elementos adicionados
- Elementos modificados
- Elementos apagados
- Elementos em tratamento
- Intervalo entre versões ou disponibilidade geral
- Esforço aplicado (em unidades apropriadas)
- Erros detectados (por versão)
- Erros corrigidos (por versão)

Métricas associadas por Característica em cada etapa do Processo

	Tamanho	Periodicidade	Complexidade	Modularidade	Confiabilidade	Eficiência	Manutenibilidade
Especificação de Requisitos	• Qtde Requisitos • Qtde Casos de Uso • Qtde Requisitos Tratados • Qtde Casos de Uso Tratados	• Intervalo entre Versões	• Qtde Pontos de Função • Qtde Pontos de Casos de Uso	• Acoplamento entre Casos de Uso (Qtde Extensões e Usos)	• Qtde Defeitos	• Qtde Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Eficiência no Diagnóstico de Defeitos • Eficiência na Remoção de Defeitos
Projeto de Alto Nível	• Qtde Diagramas Classes • Qtde Diagramas Sequência • Qtde Diagramas Estado • Qtde Diagramas Empacotamento • Qtde Diagramas Atividades • Qtde Diagramas Classes Tratados • Qtde Diagramas Sequência Tratados • Qtde Diagramas Estado Tratados • Qtde Diagramas Empacotamento Tratados • Qtde Diagramas Atividades Tratados	• Intervalo entre Versões	• Qtde Classes • Qtde Métodos por Classe • Profundidade de Herança por Classe • Qtde Filhos por Classe	• Acoplamento entre Classes	• Qtde Defeitos	• Qtde Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Eficiência no Diagnóstico de Defeitos • Eficiência na Remoção de Defeitos
Projeto de Baixo Nível	• Qtde Diagramas Classe • Qtde Diagramas Sequência • Qtde Diagramas Classe Tratados • Qtde Diagramas Sequência Tratados	• Intervalo entre Versões	• Qtde Classes de Domínio • Qtde Classes de Suporte • Qtde Métodos por Classe • Profundidade de Herança por Classe • Qtde Filhos por Classe • Acoplamento entre Objetos • Resposta de uma Classe • Perda de Coesão em Métodos • Qtde Subsistemas	• Coesão em Métodos • Acoplamento entre Classes	• Qtde Defeitos	• Qtde Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Eficiência no Diagnóstico de Defeitos • Eficiência na Remoção de Defeitos
Codificação	• Qtde Linhas de Código Fonte • Qtde Linhas de Código Fonte Tratadas	• Intervalo entre Versões	• Qtde Métodos por Classe • Profundidade de Herança por Classe • Qtde Filhos por Classe • Acoplamento entre Objetos • Resposta de uma Classe • Perda de Coesão em Métodos • Qtde Subsistemas • Complexidade Ciclomática	• Coesão em Métodos • Acoplamento entre Classes	• Qtde Defeitos • Disponibilidade do Sistema	• Qtde Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Eficiência no Diagnóstico de Defeitos • Eficiência na Remoção de Defeitos

Classificação de Sistemas e as Leis de Evolução

- Os conceitos de classificação de sistemas de Lehman, acompanhado de suas leis, provêm um vocabulário amplamente aceito para a discussão da natureza da mudança de software
- Através destas ideias, pode-se projetar sistemas para serem flexíveis, planejar manutenções, além de melhor entender e controlar o desenvolvimento de software, mais do que simplesmente reagir aos problemas que acontecem (Pfleeger, 1998)

Ferramentas para Controle Estatístico de Processos de Software

- Algumas ferramentas
  - Gráfico de Barras
  - Diagramas de Tendência
  - Histogramas
  - Gráficos de controle
  - Entre outros

Gráfico de Controle

- Instrumento que permite identificar as causas de variação **não natural** do processo
- Utiliza limites de controle, superior, inferior e, por vezes, auxiliares.

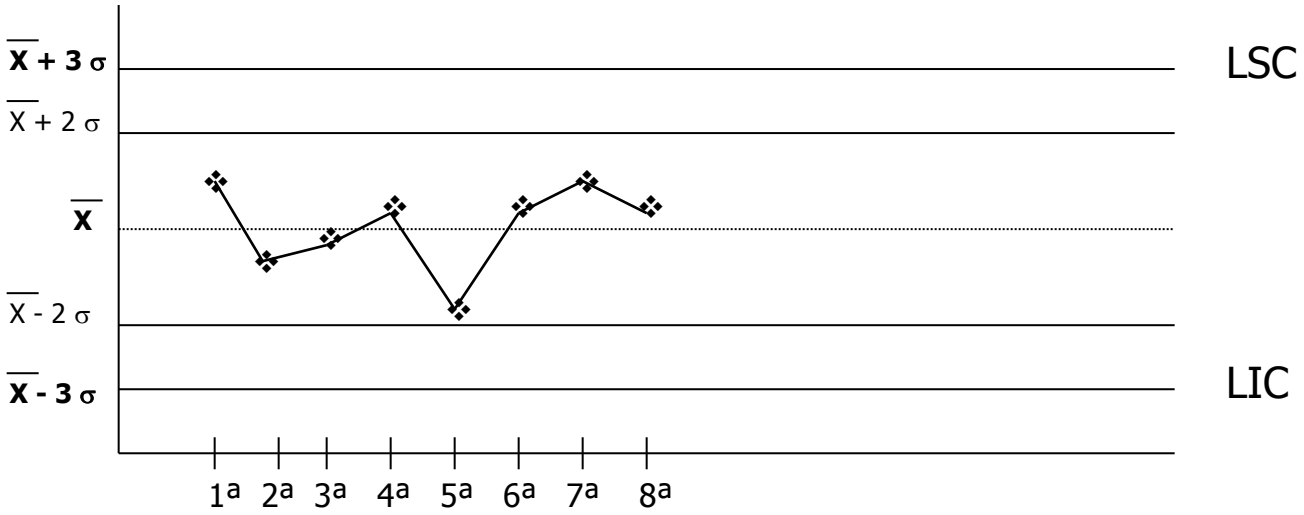


Gráfico de Controle

- Processo estável

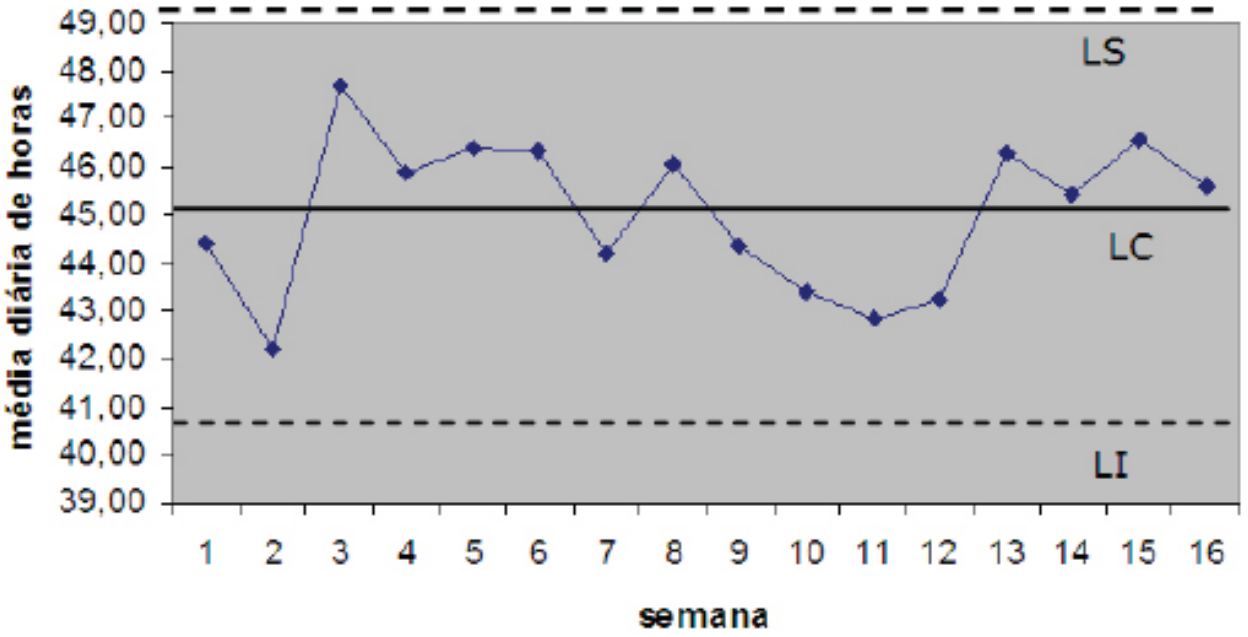


Gráfico de Controle

- Processo extrapolando níveis de controle aceitáveis

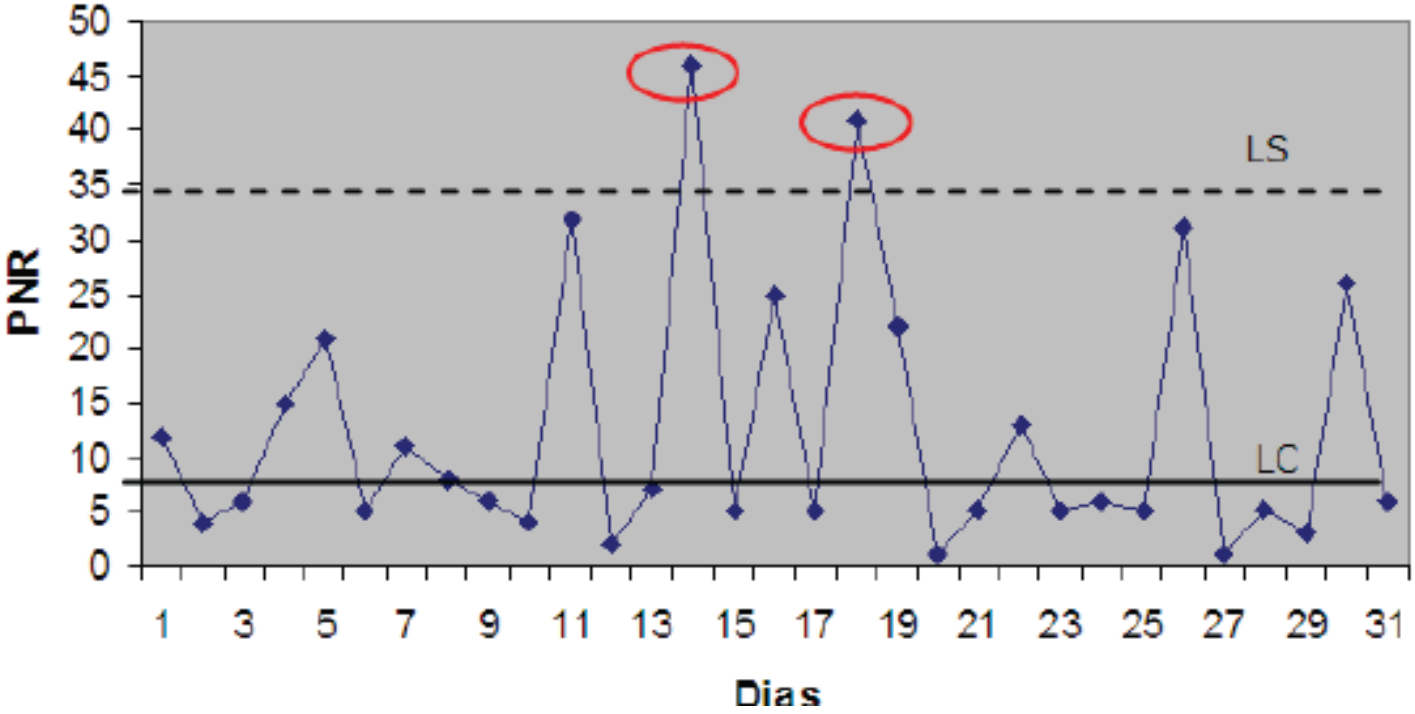


Gráfico de Controle

- Testes para verificação de estabilidade de processos
  1. Presença de algum ponto fora dos limites de controle superior ou inferior
  2. Presença de dois ou três valores sucessivos do mesmo lado a mais de 2 desvios padrão da linha central (zona C)
  3. Presença de quatro ou cinco valores sucessivos do mesmo lado a mais de um desvio padrão da linha central (zona B)
  4. Presença de oito pontos sucessivos do mesmo lado da linha central

Gráfico de Controle

- Testes para verificação de estabilidade de processos