|  |  |  |
| --- | --- | --- |
| Atributos de Qualidade de Software | | |
| Segurança | *Compreensibilidade (ME)* | Portabilidade |
| Proteção | Testabilidade | *Usabilidade* |
| Confiabilidade (MD) | Adaptabilidade | Reusabilidade |
| Resiliência | Modularidade (AS) | Eficiência (MD) |
| Robustez (AS) | Complexidade (ME) | *Capacidade de Aprendizado* |
| Manutenibilidade (ME)(AS) |  | *Capacidade de Distribuição (AS)* |
| MD = Métricas dinâmicas → fácil de medir: tempo de execução, número e tipo de falhas de sistema ME = Métricas estáticas → experimentos não conclusivos; tamanho do programa e complexidade de controle são confiáveis *Qualidades subjetivas; AS = Arquitetura de Software* | | |

|  |  |
| --- | --- |
| Métrica de Software Estática | Descrição |
| Fan-in/Fan-out | Fan-in: num. de métodos que chamam X. Grande significa X fortemente acoplado, alterações causam grandes repercussões Fan-out: num. de funções que são chamadas por X. Sugere complexidade alta por causa da lógica de controle dos componentes chamados |
| Comprimento de código | Tamanho do programa. Quanto maior, mais complexo e sujeito a erros. |
| Complexidade ciclomática | Medida de complexidade de controle, relacionada à compreensibilidade |
| Comprimento de identificadores | Comprimento médio de nomes de variáveis, métodos e classes; mais longos mais significativos |
| Profundidade de aninhamento condicional | Profundidade de IF, potencialmente sujeito a erros |
| Índice Fog | Comprimento de palavras e sentenças em documentos. Maior valor do índice mais difícil compreensão |

Gerenciamento de Configurações – Atividades:

* Gerenciar mudanças – acompanhar solicitações, definir custos e impacto, decidir se faz ou não
* Gerenciar versões – acompanhar e evitar interferência
* Construir sistema – criação do sistema executável (montagem, compilação, etc.)
* Gerenciar releases – preparar SW para lançamento externo, saber o que está com o cliente

Visões de Arquitetura de Software – como documentar

* Lógica – abstrações fundamentais, como classes e objetos – relaciona requisitos de sistema e entidades
* De Processo – composição em processos interativos em execução, RNF (desempenho, disponibilidade)
* De Desenvolvimento – distribuição de componentes implementados por desenvolvedores, útil para gerentes e programadores
* Física – hardware e como componentes são distribuídos, engenheiros planejando implantação

Padrões de Arquitetura – afetam desempenho, proteção e disponibilidade

* MVC – separa Modelo (sistemas de dados e operações sobre eles), Visão (como apresentar para usuário) e Controle (gerencia interação e passa para os outros dois), permite alterações independentes, overhead if M ou DATA são simples
* Camadas – organização com funcionalidade associada a cada camada, camada abaixo fornece serviços para as superiores (através das interfaces), permite substituição de camadas inteiras, aumenta confiança, mas difícil separar claramente as camadas na prática, desempenho pode cair
* Repositório – dados gerenciados em repositório central, componentes interagem apenas por meio do repositório, podem ser independentes, repositório é ponto único de falha
* Cliente-servidor – funcionalidade organizada em serviços prestados por servidores distribuídos pela rede, cada serviço é um ponto único de falha e desempenho é imprevisível
* Duto e Filtro (pipeline) – processamento discreto em cada componente (filtro), dados fluem pelo duto, pode ser sequencial ou concorrente, fácil compreensão, mas overhead a cada entrada e saída com transformações