Manutenção e Evolução de Software

Marco Túlio Valente

mtov@dcc.ufmg.br

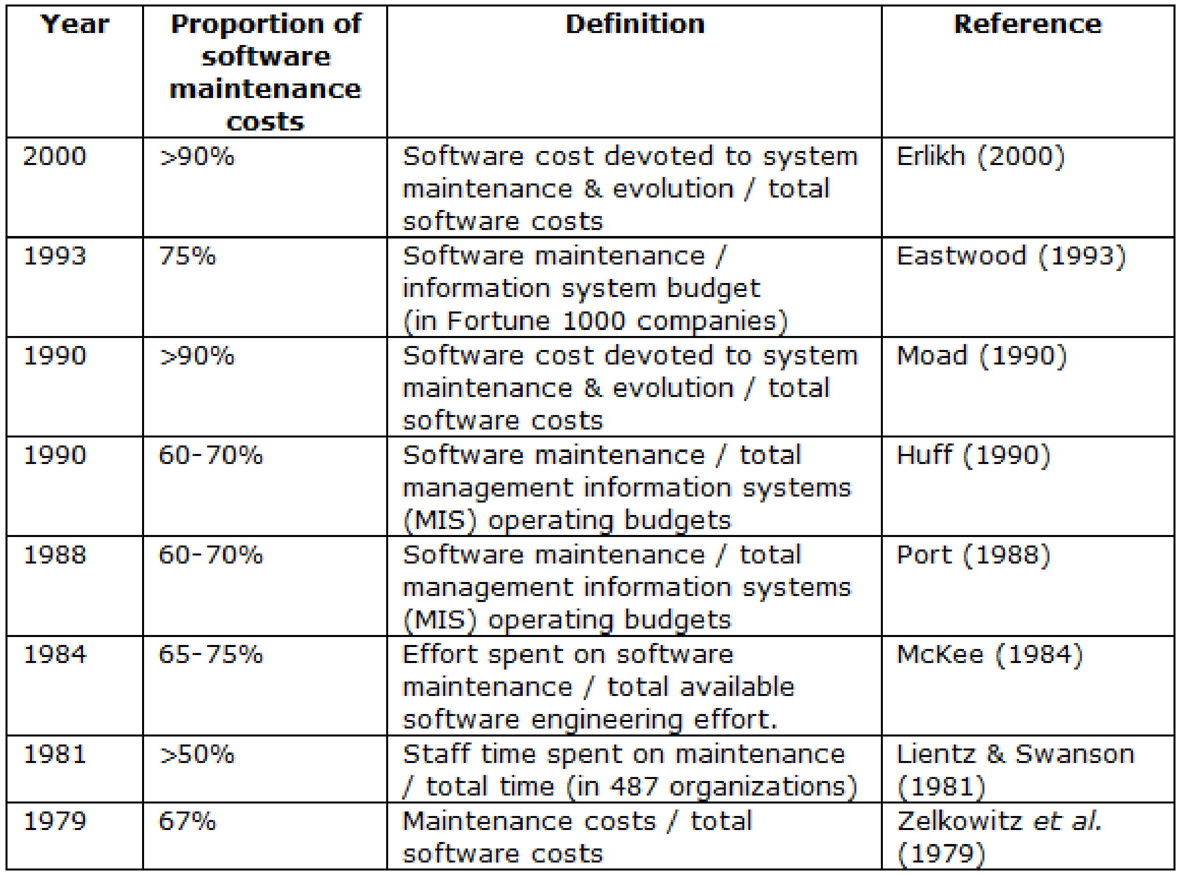
DCC - UFMG

Motivação

- “Our civilization runs on software” – Bjarne Stroustrup
- Construir software é complexo e desafiador
  - Um dos artefatos mais complexos construídos por humanos
- Manter e evoluir software é tão complexo quanto construir!

Motivação

- Fonte: http://users.jyu.fi/~koskinen/smcosts.htm



Year	Proportion of software maintenance costs	Definition	Reference
2000	>90%	Software cost devoted to system maintenance & evolution / total software costs	Erlikh (2000)
1993	75%	Software maintenance / information system budget (in Fortune 1000 companies)	Eastwood (1993)
1990	>90%	Software cost devoted to system maintenance & evolution / total software costs	Moad (1990)
1990	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Huff (1990)
1988	60-70%	Software maintenance / total management information systems (MIS) operating budgets	Port (1988)
1984	65-75%	Effort spent on software maintenance / total available software engineering effort.	McKee (1984)
1981	>50%	Staff time spent on maintenance / total time (in 487 organizations)	Lientz & Swanson (1981)
1979	67%	Maintenance costs / total software costs	Zelkowitz *et al.* (1979)

Table 1. Proportional software maintenance costs for its supplier.

Motivação

- **Absolute software maintenance costs:**
  - Annual software maintenance cost in USA has been estimated to be more than **$70 billion**
  - The federal government alone spent about **$8.38 billion** during a 5-year period to the Y2K-bug corrections.
  - At company-level, e.g. Nokia Inc. used about **$90 million** for preventive Y2K-bug corrections.

- **Maintenance task types:**
  - **75%** of maintenance costs are *enhancements* (adaptive and perfective maintenance)
  - Studies of software maintainers have shown that **50%** of their time is spent *understanding the code* they are to maintain

Motivação

- **Legacy code amount**
  - 1990: **120 billion lines** of source code being maintained
  - 2000: **250 billion lines** of source code being maintained
  - An average Fortune 100 company maintains **35 million lines** of code (1994).
    - These companies add in average **10%** each year only in enhancements
    - The amount of code maintained doubles in size every **7 years**
  - Older languages are not dead.
    - **70%** of the active business applications are written in COBOL
    - **200 billion lines** of COBOL-code still existing in mainframe computers alone

Programa do Curso

- Introdução
  - Conceitos Básicos
  - Relevância das Fases de Manutenção e Evolução
  - Leis da Evolução de Software
  - Processos de Manutenção
  - Modelos de Custo

- Reengenharia de Software
  - Catálogos de Refactorings
  - Linguagens e Ferramentas para Refactoring
  - Refatoração para Novos Paradigmas de Modularização (Aspectos, Features, Linhas de Produtos de Software).

Programa do Curso

- Compreensão de Programas
  - Program Slicing
  - Localização de Features
  - Sistemas de recomendação
  - Visualização de Software
- Métricas e Qualidade de Software
  - Métricas de Qualidade Interna
  - Detecção e Gerenciamento de Clones
  - Modelos de Predição de Defeitos
- Estudos Empíricos e Mineração de Repositórios de Software

Objetivos do Curso

- Dois objetivos:
  - Conhecer, analisar, discutir e criticar a literatura sobre Manutenção e Evolução de Software
  - Definir, formatar e concluir um projeto na área de Manutenção e Evolução de Software

- Participação, empenho e dedicação dos alunos é fundamental
  - Seminários
  - Projeto

- Bibliografia
  - Não existe um livro texto
  - Maioria das aulas serão sobre artigos recentes da área

Distribuição de Pontos

- Seminário: 15 pontos (individual)
  - Artigos indicados pelo professor
  - Duração: 30 minutos; discussão: 10-15 minutos

- Projeto: 35 pontos (individual ou grupo de dois alunos)
  - Definição do tema/escopo pelo aluno
  - Uma entrega intermediária (15 pontos)
  - Apresentação final + artigo (10 páginas)

- Provas: 35 pontos (individual; sem consulta)
  - Duas provas (15 e 20 pontos)

- Trabalhos práticos (15 pontos)

Manutenção e Evolução de Software

Marco Túlio Valente

mtov@dcc.ufmg.br

DCC - UFMG

Basic Definitions

Definitions

- (from IEEE Standard Glossary of Software Engineering Terminology)

- Maintenance:
  - “The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment”

- Maintainability:
  - “The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”

Types of Maintenance

- (from Lientz and Swanson, 1980)

1. **Corrective maintenance** deals with the repair of faults found

2. **Adaptive maintenance** deals with adapting software to changes in the environment such as new hardware or OS
   - It does not lead to changes in the system's functionality

3. **Perfective maintenance** mainly deals with accomodating new or changed user requirements
   - It concerns functional enhancements to the system
   - Perfective maintenance also includes activities to increase the system's performance or to enhance its user interface

Types of Maintenance

4. **Preventive maintenance** concerns activities aimed at <u>increasing the system's maintainability</u>
   - Such as updating documentation, adding comments, and improving the modular structure of the system.

ISO 12207

ISO 12207 - Lifecycle Process

- ISO standard for the software lifecycle processes
- It aims to be the standard that <u>defines all the tasks required for developing and maintaining software</u>

- The primary lifecycle processes contain the <u>core processes involved in creating a software</u> product.

- These processes are divided into five different main processes:
  1. Acquisition (activities involved in initiating a project)
  2. Supply (when a project management plan is developed)
  3. Development
  4. Operation
  5. Maintenance
- The operation and maintenance phases occur simultaneously

ISO/IEC 14764

ISO 14764 – Maintenance Process

- Purpose:
  - to provides guidance on the management of (or how to perform) the maintenance process

- Limitations:
  - This standard describes the framework of the Software Maintenance Process
  - but does not specify the details of how to implement or perform the activities and tasks included in the process.

Terms and Definitions

- **Corrective maintenance** refers to <u>changes necessitated by actual errors</u> in a software product
  - If the software product does not meet its requirements, corrective maintenance is performed

- **Preventive Maintenance** refers to the changes necessitated by <u>detecting potential errors</u> in a software product.
  - It is commonly performed on software products which have safety or prevention of loss of life as a concern.

Terms and Definitions

- **Adaptive and Perfective** changes are enhancements to a software product. These changes are those that were not in the design specifications of the released software.

- **Adaptive** changes are those changes necessary to accommodate a changing environment
  - The modification of a software product, performed after delivery, to keep a software product usable in a changed or changing environment.

- **Perfective** changes improve the software product's performance or maintainability. A perfective change might entail providing new functionality improvements for users or reverse engineering to create maintenance documentation that did not exist previously or to change existing documentation.

Modification Request (MR)

- MR: a generic term used to identify proposed changes to a software product that is being maintained

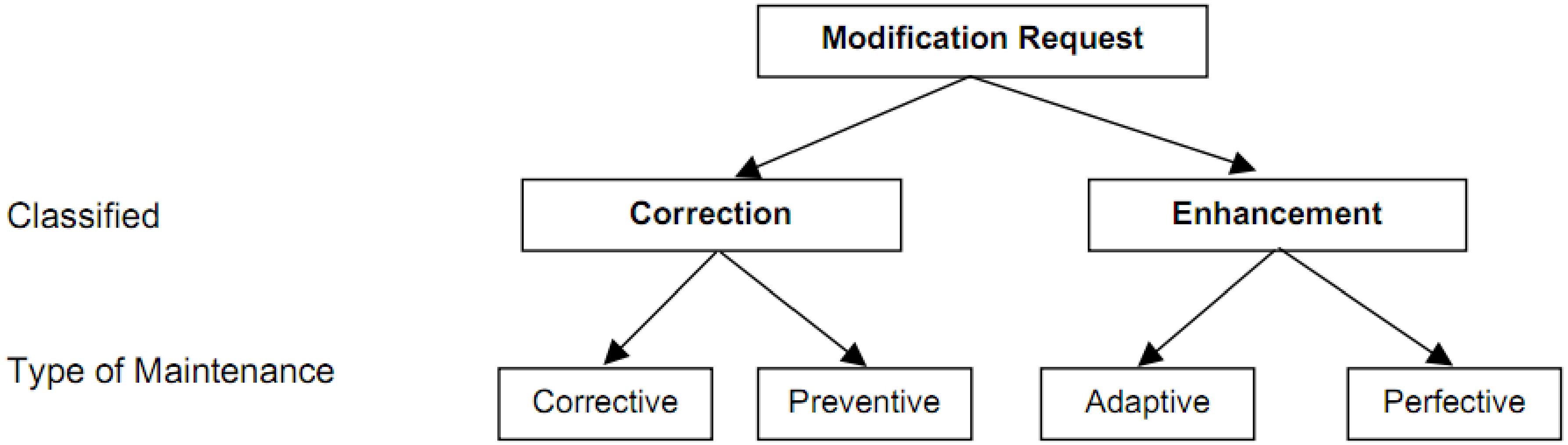


Figure 1 — Modification Request

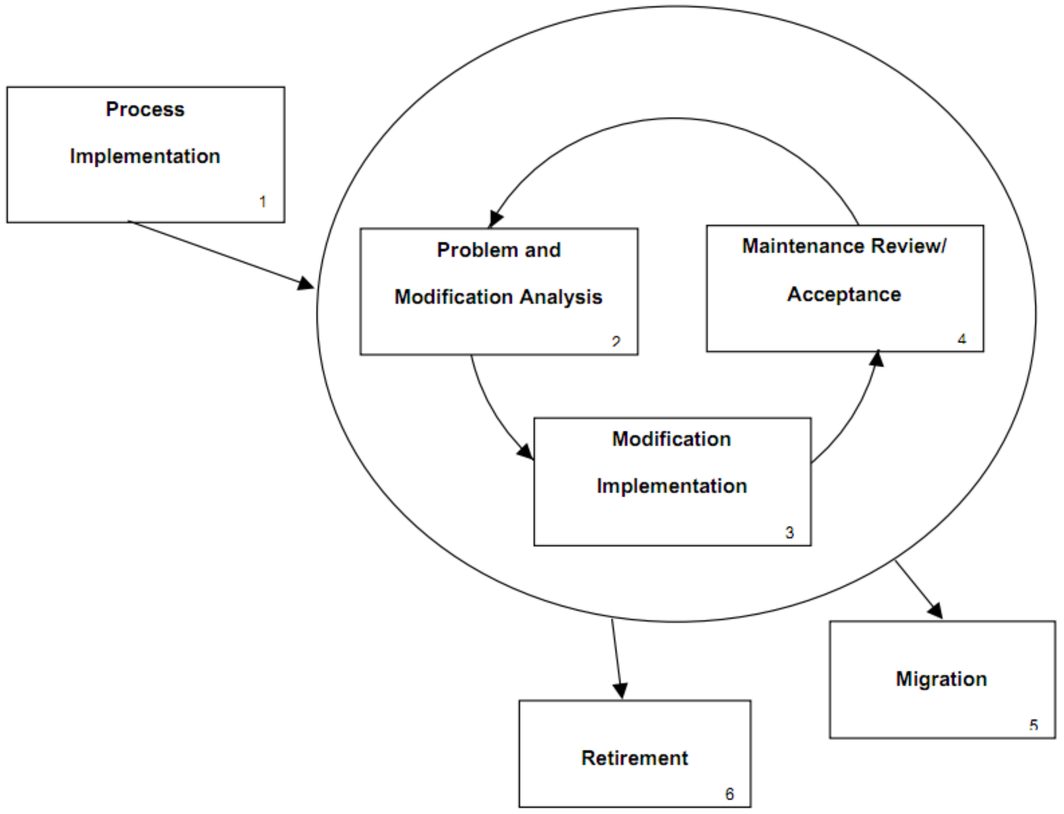
Maintenance Process

- The Maintenance Process contains the activities and tasks necessary to modify an existing software product while preserving its integrity

- The Maintenance Process should be activated when a requirement exists to maintain a software product

Activities

1. Process Implementation
2. Problem and Modification Analysis
3. Modification Implementation
4. Maintenance Review/Acceptance
5. Migration
6. Retirement


Process Implementation 1
Problem and Modification Analysis 2
Maintenance Review/ Acceptance 4
Modification Implementation 3
Migration 5
Retirement 6

Process Implementation

- During Process Implementation, the maintainer establishes the plans and procedures which are to be executed during the Maintenance Process

- Tasks
  - Develop Maintenance Plans and Procedures;
  - Establish MR/PR Procedures;
  - Implement Configuration Management

Maintenance Plans and Procedures

- In order to develop effective Maintenance Plans and Procedures, the maintainer should perform the following task-steps:

1. Assist the acquirer in developing the maintenance concept
2. Assist the acquirer in determining the scope of maintenance
3. Assist the acquirer in analyzing maintenance organization alternatives
4. Ensure written designation as the maintainer for the sw product
5. Conduct resource analyses and estimate maintenance costs
6. Perform a maintainability assessment of the system
7. Determine transition requirements and milestones
8. Identify the maintenance process which will be used
9. Document the maintenance process in the form of operating procedures.

MR/PR Procedures

1. Develop an identification numbering scheme for MRs/PRs
2. Develop a scheme for categorizing and prioritizing MRs/PRs
3. Develop procedures for determining trend analysis
4. Determine the procedures for an operator to submit a MR/PR
5. Determine how initial feedback will be provided to the users
6. Determine how temporary work-arounds will be provided to the users
7. Determine how data is entered into the status accounting database
8. Determine what follow-up feedback will be provided to the users

Problem and modification analysis

- During the Problem and Modification Analysis Activity, the maintainer:
  1. Analyzes MRs/PRs;
  2. Replicates or verifies the problem;
  3. Develops options for implementing the modification;
  4. Documents the MR/PR, the results, and implementation options;
  5. Obtains approval for the selected modification option

Maintenance review/acceptance

- This activity ensures that the modifications to the system are correct and that they were accomplished in accordance with the approved standards using the correct methodology

- Tasks:
  - Reviews are conducted to ensure that modifications are correct and approval is obtained for satisfactory completion of the modifications.

Migration

- During a system's life, it may have to be modified to run in different environments.
- In order to migrate a system to a new environment, the maintainer needs to:
  - Determine the actions needed to accomplish the migration
  - And then develop and document the steps required to effect the migration.

Software retirement

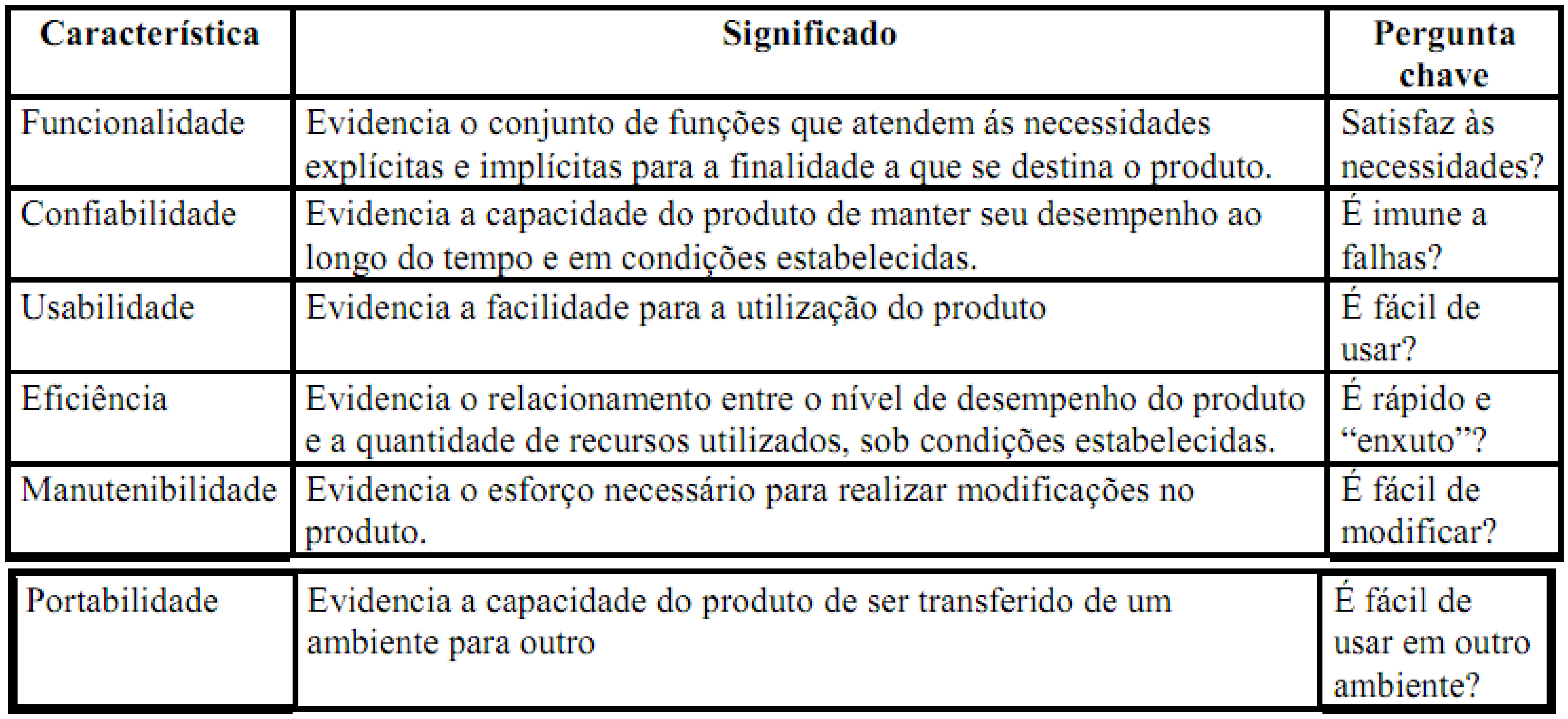
- Once a software product has reached the end of its useful life, it must be retired.
- An analysis should be performed to assist in making the decision to retire a software product. The analysis is often economic-based and may be included in the Retirement Plan.
- Analysis should determine if it is cost effective to:
  1. retain outdated technology;
  2. shift to new technology by developing a new product;
  3. develop a new software product to achieve modularity;
  4. develop a new software product to facilitate maintenance;
  5. develop a new software product to achieve standardization;
  6. develop a new software product to facilitate vendor independence.

ISO/IEC 9126

ISO/IEC 9126

- Evaluation of software quality

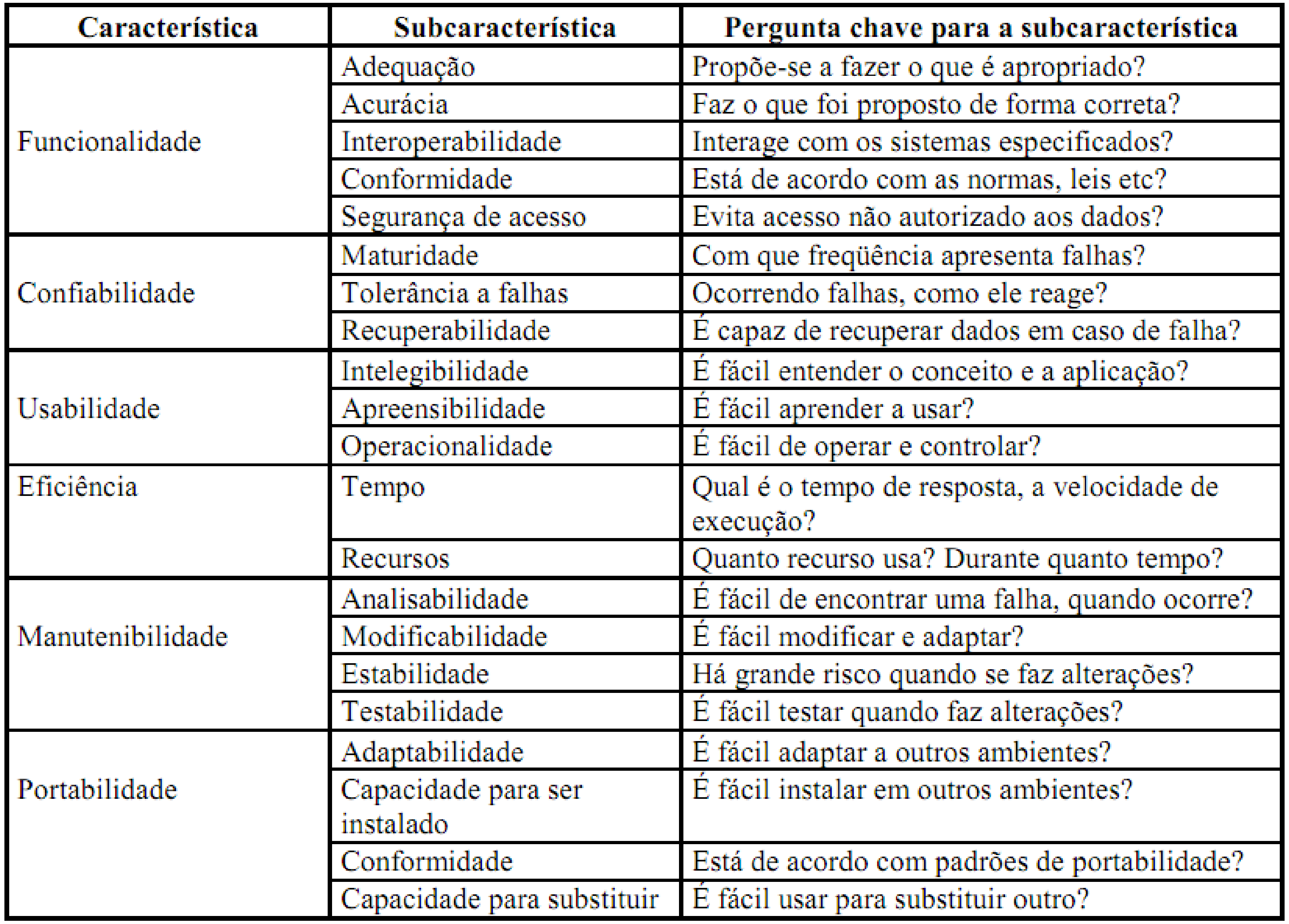
Tabela 1 – Características da Qualidade de Software segundo a ISO/IEC 9126-1



Característica	Significado	Pergunta chave
Funcionalidade	Evidencia o conjunto de funções que atendem ás necessidades explícitas e implícitas para a finalidade a que se destina o produto.	Satisfaz às necessidades?
Confiabilidade	Evidencia a capacidade do produto de manter seu desempenho ao longo do tempo e em condições estabelecidas.	É imune a falhas?
Usabilidade	Evidencia a facilidade para a utilização do produto	É fácil de usar?
Eficiência	Evidencia o relacionamento entre o nível de desempenho do produto e a quantidade de recursos utilizados, sob condições estabelecidas.	É rápido e "enxuto"?
Manutenibilidade	Evidencia o esforço necessário para realizar modificações no produto.	É fácil de modificar?
Portabilidade	Evidencia a capacidade do produto de ser transferido de um ambiente para outro	É fácil de usar em outro ambiente?

ISO/IEC 9126

Tabela 2 – Subcaracterísticas da Qualidade de Software segundo a ISO/IEC 9126-1



Característica	Subcaracterística	Pergunta chave para a subcaracterística
Funcionalidade	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Faz o que foi proposto de forma correta?
	Interoperabilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis etc?
	Segurança de acesso	Evita acesso não autorizado aos dados?
Confiabilidade	Maturidade	Com que freqüência apresenta falhas?
	Tolerância a falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falha?
Usabilidade	Intelegibilidade	É fácil entender o conceito e a aplicação?
	Apreensibilidade	É fácil aprender a usar?
	Operacionalidade	É fácil de operar e controlar?
Eficiência	Tempo	Qual é o tempo de resposta, a velocidade de execução?
	Recursos	Quanto recurso usa? Durante quanto tempo?
Manutenibilidade	Analisabilidade	É fácil de encontrar uma falha, quando ocorre?
	Modificabilidade	É fácil modificar e adaptar?
	Estabilidade	Há grande risco quando se faz alterações?
	Testabilidade	É fácil testar quando faz alterações?
Portabilidade	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capacidade para ser instalado	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capacidade para substituir	É fácil usar para substituir outro?

Lehman's Laws of Software Evolution

Lehman's laws of software evolution

I - Continuing change (1974)

A program that is used in a real-world environment necessarily must change or become progressively less useful

II - Increasing complexity (1974)

As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.

III - Continuing growth (1980)

The functionality offered by systems has to continually increase to maintain user satisfaction

Maintenance Costs

Maintenance Costs

- Y. Tan, V. Mookerjee: Comparing Uniform and Flexible Policies for Software Maintenance and Replacement. IEEE TSE, 2005.

- A period is the time between one activity (maintenance or replacement) and the next activity (maintenance or replacement).
- N_i requests are collected in the ith period.
- Requests are analyzed; new features are added to the system
- M_i function points are needed to satisfy the N_i requests
- This quadratic estimates the costs for a maintenance activity:

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2}\gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j. \qquad (6)$$

Fixed Costs

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2}\gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j. \qquad (6)$$

- The first term is the fixed cost of a maintenance activity.
- This cost is associated with the effort needed to plan and organize the task of implementing a new set of features.

Variable Costs

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2}\gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j. \qquad (6)$$

- The remaining three terms represent the (variable) cost of developing and integrating the new features.

Development Costs

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2}\gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j. \qquad (6)$$

- The second term is the cost associated with analyzing requests and coding the new features.
- This effort can be reasonably assumed to be proportional to the number of function points associated with the new features.
- The linear form used for this term is reasonable because the second term is analogous to the cost of developing a new module; but does not include the cost of integrating the module with the rest of the system.

Integration Costs

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2}\gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j. \tag{6}$$

- The third term models the cost of integrating and testing the new features among themselves.
- The last term describes the cost of integrating the new features into the existing system.
- The third and fourth terms represent integration effort that can be expected to be of a multiplicative nature
  - Integrating *n* features with *m* features can be expected to be of the order of *n x m*, rather than simply *n + m*.
  - The third term has a quadratic form because it represents the internal interaction between a set of n features, i.e., *n x n*.

Degradation Effect

- The complexity of the system increases with the size and with the number of times that the software has been maintained.
- This (degrading) effect depends on the number of times the system has been maintained since it was last replaced
- The degradation effect is explicitly captured through the coefficient c_3 in (6):

$$c_3^i = \gamma_3^i \exp(\kappa n_M), \qquad (7)$$

Degradation Effect

$$c_3^i = \gamma_3^i \exp(\kappa n_M), \qquad (7)$$

- n_M is number of times the system was maintained since last replacement
- *k* is the rate at which the system degrades from maintenance. The parameter can be related to the entropy of the system -- a measure of degree of disorder.
- γ_3^i is the value of the cost coefficient for the first maintenance activity after a replacement activity.
- Each maintenance after the first one will be more expensive because the multiplier (the exponential term) that accounts for degradation increases with each maintenance activity

Simulação

- Content Management System (CMS), com 4.217 PFs
- Suponha um projeto de manutenção com as melhorias:
  - Inclusão dos campos tipo de documento e data de criação no filtro das pesquisas de documentos existentes – 32 PF
  - Inclusão de uma funcionalidade para alertar os usuários sobre a ocorrência de mudanças nos documentos – 68 PF.
  - Inclusão de um mecanismo para controle do vocabulário utilizado durante o cadastro dos documentos – 76 PF.
  - Inclusão de um mecanismo para acesso ao sistema por Secure Socket Layer (SSL) – 13 PF.
  - Total: 189 PF
- Uma análise de impacto identificou a necessidade de se integrar as novas funcionalidades a outras quatro funcionalidades existentes. Essas funcionalidades têm o tamanho de 226 PF.

Simulação: Custo Fixo

- Quantidade de horas gastas com as reuniões de planejamento, priorização e gerenciamento das solicitações é de 32 horas
- Custo por hora dos profissionais envolvidos: R$ 65,00

- A equipe de garantia da qualidade custa R$ 41,00 por hora
- Ela é responsável pelos procedimentos de preparação dos ambientes de desenvolvimento e homologação
- Essas atividades demandam seis horas e são realizadas por quatro profissionais

Simulação: Custo Fixo

- A liberação de modificações semelhantes às solicitadas demandam em média quatro horas
- Valor hora da equipe de suporte: R$ 25,00.
- Essas atividades são geralmente realizadas por um único membro da equipe de suporte

- Custo de aquisição de um certificado digital: R$ 3.500,00

- Custo fixo= (32 * 65,00) + (4 * 25,00) + (6 * 4 * 41,00) + 3500,00
- Custo fixo= **R$ 6.664,00**

Simulação: Custo Variável e Total

- Custo / ponto de função: R$ 100,00
- Custo implementação= R$ 100,00 * 189 = **R$ 18.900,00**

- Custo de integração (1): 0,5 * R$ 1,00 * 189 * 189= **R$ 17.860,17**

- Efeito de degradação: 0,07 * exp (0,02 * 5) = 0,07 * 1,10 = 0,077
  - K= 0,02 (entropia)
  - n_M = 5
  - γ_3^i = 0,07

- Custo de integração (2): 0,077 * 189 * 226 = **R$ 3.288,98**

- Custo total: 6.664,00 + 18.900,00 + 17.860,17 + 3.288,98
- Custo total: **R$ 46.713,15** (ou **R$ 247,15 / PF**)