Basic Principles of Software Evolution: Summary

Anthony Rouneau

24 mai 2016

1 Aspects of Software Evolution

1.1 Technical aspects

Here are some technical aspects of the Software Evolution.

— Version management
— Software quality measurement and improvement
— Legacy systems and migration
— Reverse Engineering and program comprehension
— Model-driven evolution
— Change propagation and program comprehension
— Traceability
— Co-evolution
— Consistency maintenance
— Regression testing
— Design for change
— Visualisation and statistical analysis of evolution histories
— Software product lines and product families

Definition 1. ***Reverse engineering*** – *Process of analysing a project to fully understand it and to extract the key issues.*

Definition 2. ***Re-engineering*** – *Horseshoe process (cf. figure 1) that targets to rethink a project in order to address one or multiple problems.*

FIGURE 1 – Re-engineering

1.2 Managerial aspects

— Evolutionary process model
  — Staged life-cycle
  — Iterative and incremental process
  — Agile methods
— Software configuration management
  — Change management
  — Version management
— Estimation techniques
  — Change impact analysis
  — Effort estimation
  — Cost estimation
  — Change metrics

2 Software Maintenance

The software maintenance is used to avoid **large** software to become legacy systems. In fact, adding new functionalities without considering code quality gives rise to **technical debt**.

Definition 3. ***Technical debt*** *– Lack of quality in the code due to modifications done in a hurry. The project is unclean while the debt has not been paid back through re-engineering.*

Definition 4. ***Software maintenance (1)*** *– 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*

Definition 5. ***Software maintenance (2)*** *– The software product undergoes modification to code and associated documentation due to a problem or the need for improvement. The objective is to modify the existing software product while preserving its integrity*

The software change is :

- **Unpredictable** – All the changes and bugs can not be anticipated in the design.
- **Expensive** – You generally don't get paid to solve bugs but to develop as fast as possible.
- **Difficult**– The errors are hard to find and the source code can be messy.

2.1 Legacy system

Characteristics :

- Original developers no longer available
- Outdated development methods used
- Extensive patches and modifications have been made
- Missing or outdated documentation

2.2 Parnas' ageing software

Symptoms :

- Lack of knowledge
  - Insufficient, inconsistent or obsolete documentation
  - Departure of original developers
  - Disappearance of inside knowledge about the system
  - Missing tests
- Code symptoms
  - Duplicated code, code smells, lack of modularity
  - Lack of overall structure or architecture
- Process symptoms
  - Constant need for bug fixes Too long time to fix bugs or to add new functionality
  - Difficult to separate functionalities

2.3 Necessity of software change

Software change is inevitable because :

— New requirements emerge when the software is used or developed.
— The business environment changes.
  — New customers
  — New demands
  — Organisational changes
  — Competitors
— Errors must be repaired
  — Bug fix routine
  — Emergency fix
— Hardware changed
— Improvements needed in efficiency
— New technologies have to be used
— Changes in data formats
  — New standards
  — New currency, ...

2.4 Types of software maintenance

— **Adaptive** – Adapt a software after delivery to support new technologies
— **Corrective** – Repair errors discovered after delivery
— **Perfective** – Add new functionalities to the software or improve it after delivery
$\Longrightarrow$ *main reason*
— **Preventive** – Correct latent faults before they become effective ones.

When ? / Why ?	**Correction**	**Enhancement**
Proactive (before it happens)	Preventive	Perfective
Reactive (after it happened)	Corrective	Adaptive