L19 - Introduction to Software Quality

CS3028 - Principles of Software Engineering

Ernesto Compatangelo

Department of Computing Science

UNIVERSITY OF ABERDEEN

19.1 Reminding past issues and mapping them to current topics

Where are we now?

⇒ · · · · · ·

⇒ Elaboration

⇒ Construction (third UP phase)

⇒ From Elaboration (design) to construction (implementation)

⇒ SW engineering: a total SW quality approach

19.2 From UP elaboration artifacts to UP construction

Life after design: coding in a programming language

- Detailed design defines the software structure completely
- The only thing left out is the choice of a specific programming language to express a detailed design (*e.g.*, UML class and sequence diagrams) AND to implement the body of each method (**SW coding**)
- Note that a 'perfect design' that complies with all criteria for good design (low coupling, high cohesion, small modules, compliance with all applicable patterns) can turn into an awful codebase if methods are not programmed following *criteria for algorithmic quality* (as taught in UG CS core programming and algorithmic problem solving courses)

19.2.1 Artifacts resulting from the UP elaboration phase

1. **Use case model** *(Req.)*: captures the functional requirements for the system, identifying all use cases and describing most of them. *Supplementary specifications* are added to capture non-functional requirements.
2. **Architecture baseline** *(Des.)*: provides an initial implementation of the system architecture — a lean working version used as a foundation for continued development *(the ALPHA release)*
3. **System acceptance test specification** *(Test)*: describes the measurable standards of system performance based on performance requirements
4. **Project plan** *(Proj. mgmt)*: encompasses the whole project, with milestones, iterations, and evaluation criteria for each iteration. It also includes a detailed plan for the construction phase.

19.3 Mapping a detailed design to object-oriented code

Summarising software engineering

- Quality in requirements specification (comprehensive information well articulated) leads to a better design
- Quality in architectural and detailed design (adherence to criteria of good design and compliance with all applicable patterns patterns) leads to a better coding
- Quality in coding (adherence to criteria of good programming) leads to a better SW system (less bugs, better features)

19.4 Implementation concern - focus on Software Quality Control

Talking about quality . . .

- Implementation (construction) is generally the phase where SW quality issues are first raised.
- Too little too late, as SW quality begins with proper requirement specification and proper (pattern-based) design.
- Whatever the waterfall or Agile approach chosen, SW quality must be **controlled** from requirements to implementation and beyond.
- **Software Quality Control** thus encompasses
  1. **Software Quality Assurance** (SQA)
  2. **Software Verification & Validation** (V&v)

19.5 Software Quality Assurance (SQA)

What is Software Quality Assurance (SQA)?

- An approach used to monitor the software engineering **process** used to ensure **software quality**
- Whereas software quality control is a *control of products*, software quality assurance is a *control of processes*
- SQA encompasses the entire software development process, from requirements elicitation to deployment
- SQA ensures process quality by means of audits of the **quality management system** under which the software system is created
- SQA audits are backed by one or more **standards**, usually the *ISO 9000* family of standards (ISO 9001, ISO 9000-3)

Why SQA?

- Necessary in the modern market
  - to get edge over competitors
  - to cut costs
  - to comply with contractual obligations
- Derived from the industrial philosophy that *good process leads to good products*
- Derived from the industrial analysis that quality control on the product is not good enough, but that an assurance of the process quality itself is also needed

19.6 The ISO 9000 family and the ISO 9001 SQA standard

The ISO 9001 requirements

- **ISO 9001:2000** (*quality management systems - requirements*) is the relevant standard for the SW sector (aka *tertiary industrial sector*)
- It defines **requirements which an organisation needs to fulfill** if it is to achieve customer satisfaction through consistent products and services which meet customer expectations
- Requirements include management responsibility, quality system, contract review, design control, document control
- Specifies the organisation, infrastructure and requirements of a **Quality Management System** (QMS) in terms of *what is to be done for QA, not how it is to be done*
- Organisations are free to choose their own QM methods and tools

19.6.1 The ISO 9001 certification

- **ISO 9001:2000** certification is given **for the QMS, not for the products**
- Serves as an agreed basis for customer and developer about developer's QMS
- Organisations need external certification of their QMS
  - to comply with *mandatory* customer requirements (as specified, for instance, by governmental agencies)
  - to increase customer confidence
  - to gain international status and acceptance
  - to find weaknesses in their QMS and to receive remedial advice

The problems with quality management systems

- Overemphasis on QMSs **undermines domain expertise**
- Systematic and elaborate QMS activities:
  - **consume** time and resources
  - **increase** software development costs considerably
- QMSs are acceptable (and accepted) for large organisations carrying out **critical projects**
- QMSs are **out of budget** for smaller organisations

19.7 Software quality control and the role of Verification & Validation

Software Verification & Validation

- Also known as software **testing**
- Consists of a means of controlling the **quality** of (engineered) software products (not processes!) through
  1. Formal technical reviews
  2. Performance audits
  3. **Tests** of the software system

Differences between SW verification and SW validation

- **Software verification, aka 'building the software right'** is the process of checking whether the software product meets the *specified requirements* (as understood by the analyst/developer).

  It ensures that the specified functions are correctly implemented – often referred to as 'internal testing'

- **Software validation, aka 'building the right software'** is the process of checking whether the software system meets the *actual* user requirements (which may differ from the specified ones).

  It traces implemented functions back to user requirements – often referred to as 'acceptance testing'; may include *field trialling*

19.8 Software testing: an introduction

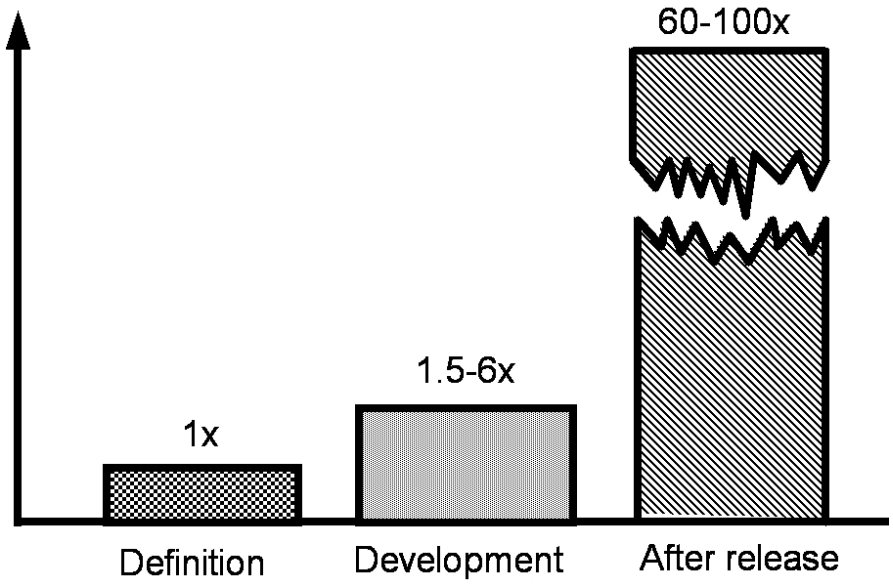
What is 'internal testing' testing in a nutshell?

- The process of **operating a system or component under specified conditions**, observing or recording the results and making an evaluation of some aspect of the system or component (IEEE, 1998)
- The ultimate **review of constructive tasks** such as specification, design and coding
- A **psychologically destructive** approach, because the main objective is to break the software
- Something **very unpopular with developers**, because of conflict of interests
- A process that **cannot be escaped** because if we don't find errors, customer will...
- A process that **cannot ultimately show absence of any defects**

Some figures about software error (bugs) and failures

- Software industry data show averages of **around 200 errors per 1,000 lines of code** (scaffolding or algorithmic) before testing
- The *Pareto principle* applied to software errors: **80% of errors traceable to 20% of 'units'**
- Cost to the global economy: **USD 1.1 trillion** in 2016 only; **4.4 billion users** affected, **315 years of effort** lost.
  In comparison, Eurozone bailout (2008-2013) costed USD 591 billion
- **Real** bug-fixing cost distribution bill: over half passed onto users, the rest footed by developers/vendors
- Bug-fixing cost: increases when find later in the development process

The cost of bug fixing



Next lecture. . .

Testing

More specifically, we will focus on:

- Introduction to testing
- Black-box testing