Development Methodologies

Prof. Dr. Josef M. Joller
jjoller@hsr.ch

THE SOFTWARE PROCESS

Overview

Client, Developer, and User

Requirements Phase

Specification Phase

Design Phase

Implementation Phase

Integration Phase

Maintenance Phase

Retirement

Problems with Software Production: Essence and Accidents

Improving the Software Process

The Software Process

The life-cycle model - Technique / Method

CASE tools - Tool

The individuals - Teams

Three fundaments on which software engineering is built!

All three have to be in balance

- Tools for untrained teams is a vast of money
- Methods without training and coaching support will never work

Testing Phase?

There is NO testing phase

Testing is an activity performed throughout software production

Verification

- Performed at the end of each phase

Validation

- Performed before delivering the product to the client

Documentation Phase?

There is NO documentation phase

Every phase must be fully documented before starting the next phase

- Postponed documentation may never be completed
- The responsible individual may leave
- The product is constantly changing—we need the documentation to do this
- The design (for example) will be modified during development, but the original designers may not be available to document it

Requirements Phase

The development process usually begins, when the client approaches a development organization with regard to a piece of software that, in the opinion of the client is either essential to the profitability of his or her enterprise or can be economically justified.

Assumption

- The software being considered is economically justifiable

Concept exploration

- Determine what the client **needs**, *not* what the client wants

Moving target problem

- Unforeseeable change in circumstances

Requirements Phase Testing

Rapid prototyping

- Carefully check the prototype!
- Ensure that the delivered product is what the client ordered
- Ensure that the product is built correctly in every way

Documentation / Result

- Prototype
  maybe
- Requirement document (partly based on results from prototype)

Specification Phase

Specifications document ("specifications")

- The Specification Document is drawn up by the specification team.
- Legal document
- Must not have phrases like "optimal," or "98% complete"
- WHAT has to be delivered
- Must be signed off
- Leads to a software product management plan (SPMP)

Must NOT be

- Ambiguous
- Incomplete (?)
- Contradictory (?)

Specification Phase Testing

Traceability

- Trace statements in the specification back to a statement made by the client team during requirement phase

Review

- Representatives of the specification team and the client are present
- The aim of the review is to determine whether the specifications are correct

Check the SPMP

- Maybe: obtain two independent estimates

Specification Phase Documentation

Specification document (specifications)

- Explicitly describes the functionality of the product, precisely what the product is supposed to do
- Lists any constraints that the product must satisfy
- Includes inputs to the product(s) and the required outputs

SPMP

- Standard based document
  - MIL Standard
  - IEEE Standard
  - Corporate Standard

Design Phase

Specification—*what*

Design—*how*

Retain design decisions

- When a dead-end is reached
- For the maintenance team
- Ideally, the design should be open-ended

Architectural design

- Decompose the product into modules

Detailed design

- Design each module: data structures, algorithms

Design Phase Testing

Traceability

- Again!

Review

- Again!
- However
  - In view of the technical nature of most design documents, the client is not usually present
  - Members of the design team and the SQA (software quality assurance) group work through the design as a whole as well as through each separate module, ensuring that the design is correct
- Possible faults
  - Logic fault, interface fault, lack of exception handling, nonconformance to the specification

Implementation Phase

Implement the detailed design in code

- Code the component modules of the design

Testing

- Review
  - Again!
- Test cases
  - Informal testing (desk checking)
  - Formal testing (SQA)

Documentation

- Source code
  - Test cases (with expected output)

Integration Phase

Combine the modules and check the product as a whole

- Integration sequence may be important
  - Top down
    - Design errors show up soon
  - Bottom up
    - Design errors become hard to correct

Tests

- Product testing
  - Integration testing
  - Check that the modules combine together correctly to achieve a product that satisfies its specification
  - Module interfaces
- Acceptance testing
  - The software is delivered to the client, who tests the software on the actual hardware, using actual data

Documentation

- Commented source code
- Test cases for the product as a whole

Maintenance Phase

Maintenance

- Any change once the client has accepted the software
- The most money is devoted to this phase
- The problem of lack of documentation

Testing

- Regression testing
  - Tests are repeatedly run, whenever a new update becomes available
  - “new” parts must be tested to conform to the “old” tests

Documentation

- Record of all changes made, with reasons
- Regression test cases

Retirement

Good software is maintained

Sometimes software is rewritten from scratch

- Software is now unmaintainable because
  - A drastic change in design has occurred
  - The product must be implemented on a totally new hardware/operating system
  - Documentation is missing or inaccurate
  - Hardware is to be changed—it may be cheaper to rewrite the software from scratch than to modify it

True retirement is a rare event

Process-Specific Difficulties

Does the product meets the user's real needs?

Is the specification document free of ambiguities, contradictions, and omissions?

Inherent Problems of Software Production

Hardware has inherent limits

So does software

No Silver Bullet

- Complexity
- Conformity
- Changeability
- Invisibility

Aristotelian categories

- Essence : things that are inherent / cannot be changed
- Accidents : maybe changed using new techniques

Complexity

Software is far more complex than hardware

- Traditional abstraction will not work
- We cannot understand the whole, so we cannot understand any part
- Management is difficult
- Maintenance is a nightmare (documentation, too)

Conformity

Type 1: Existing gold refinery

- Automation of an existing system / process
  - Build software according tol the current practice
  - Software must conform to the existing processes

Type 2: New gold refinery

- Come up with new innovative ideas about process control
  - Build software as innovative as you like / fitting the new designed business processes
  - Software must conform to the new process concept

Changeability

***Soft*ware is easier to change than *hard*ware**

Pressure to change

- Reality
- Useful software
- Easier to change
- Software has a long lifetime (~15 yrs) compared to hardware (~4 yrs)

Invisibility

Software is invisible and unvisualizable

Complete views are incomprehensible

Partial views are misleading

However, all views *can* be helpful

Is There a Silver Bullet?

What about

- High-level languages
- Time sharing
- CASE tools

These did not solve the intrinsic problems

We have experienced

- 6% annual productivity increase

But, **no "silver bullet" (order-of-magnitude increase) is possible**

Improving the Software Process

U.S. Department of Defense initiative

Software Engineering Institute (SEI) at Carnegie Mellon

The fundamental problem with software

- The software process is badly managed

Software process improvement initiatives

- Capability maturity model (CMM)
  SEI (Software Engineering Institute @ CMU)
- ISO 9000-series
- ISO/IEC 15504

Capability Maturity Model

Not a life-cycle model

Set of strategies for improving the software process

- SW–CMM for software
- P–CMM for human resources (“people”)
- SE–CMM for systems engineering
- IPD–CMM for integrated product development
- SA–for software acquisition

These strategies are being unified into CMMI (capability maturity model integration)

http://www.sei.cmu.edu/cmm/

SW–CMM

A strategy for improving the software process

- Put forward in 1986 by the SEI
- Fundamental idea:
- Improving the software process leads to
  - Improved software quality
  - Delivery on time, within budget
- Improved management leads to
  - Improved techniques

Five levels of “maturity” are defined

- Organization advances stepwise from level to level

Levels

Level 1 = Ad hoc approach

- Entire process is unpredictable
- Management consists of responses to crises

Most organizations world-wide are at level 1

Examples (http://www.sei.cmu.edu/sema/pub_ml.html)

- Level 2
  - Bosch Japan
  - Hewlett Packard
  - Oerlikon Aerospace
- Level 3
  - Boeing
  - General Dynamics
- Level 4
  - Lockheed Martin Air Traffic Management
- Level 5
  - Boeing Defense & Space Group

Level 2. Repeatable Level

Basic software management

- Management decisions should be made on the basis of previous experience with similar products
- Measurements (“metrics”) are made
- These can be used for making cost and duration predictions in the next project
- Problems are identified, immediate corrective action is taken

Level 3. Defined Level

The software process is fully documented

- Managerial and technical aspects are clearly defined
- Continual efforts are made to improve quality, productivity
- Reviews are performed to improve software quality
- CASE tools are applicable *now* (and not at levels 1 or 2)

Level 4. Managed Level / Level 5. Optimizing Level

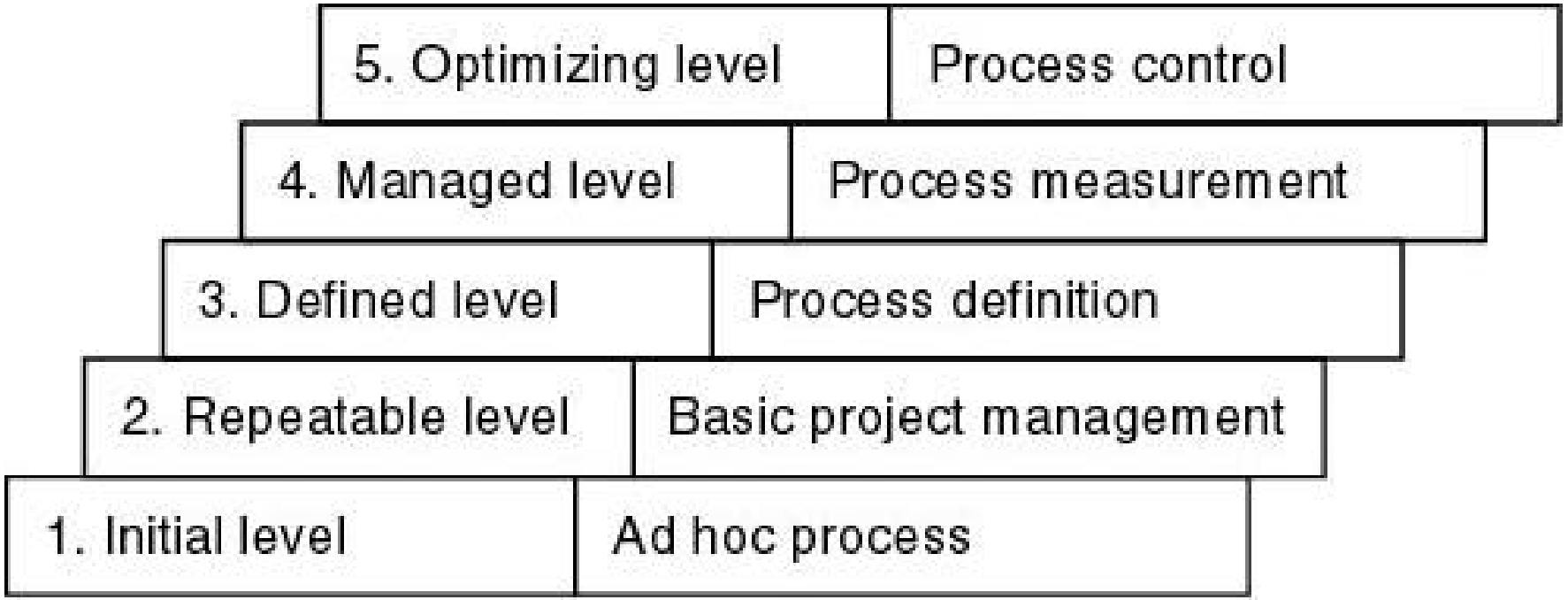
Level 4

- Quality and productivity goals are set for each project
- Quality, productivity are continually monitored
- Statistical quality controls are in place

Level 5

- Continuous process improvement
- Statistical quality and process controls
- Feedback of knowledge from each project to the next

Summary


5. Optimizing level
Process control
4. Managed level
Process measurement
3. Defined level
Process definition
2. Repeatable level
Basic project management
1. Initial level
Ad hoc process

Key Process Areas

There are key process areas (KPAs) for each level

Level 2 KPAs include:

- Requirements management
- Project planning
- Project tracking
- Configuration management
- Quality assurance

Compare

- Level 2: Detection and correction of faults
- Level 5: Prevention of faults

Experience

It takes:

- 3 to 5 years to get from level 1 to level 2
- 1.5 to 3 years from level 2 to level 3
- SEI questionnaires highlight shortcomings, suggest ways to improve the process

Original idea: Defense contracts would be awarded only to capable firms

Profitability

- Hughes Aircraft (Fullerton, CA) spent $500K (1987–90)
  - Savings: $2M per year, moving from level 2 to level 3
- Raytheon moved from level 1 in 1988 to level 3 in 1993
  - Productivity doubled
  - Return of $7.70 per dollar invested in process improvement

ISO 9000

Set of five standards for industrial activities

- ISO 9001 for quality systems
- ISO 9000-3, guidelines to apply ISO 9001 to software
- There is an overlap with CMM, but they are not identical
- *Not* process improvement
- Stress on documenting the process
- Emphasis on measurement and metrics
- ISO 9000 is required to do business with the E.U.
- Also by many U.S. businesses, for example, GE
- More and more U.S. businesses are ISO 9000 certified

ISO/IEC 15504

Original name: Software Process Improvement Capability dEtermination (SPICE)

- International process improvement initiative
- Started by British Ministry of Defence (MOD)
- Includes process improvement, software procurement
- Extends and improves CMM, ISO 9000
- Framework, not a method
  - CMM, ISO 9000 conform to this framework
- Now referred to as ISO/IEC 15504
- Or just 15504 for short

Process Improvement Data

Category	Range	Median	Number of Data Points
Years engaged in software process improvement (SPI)	1–9	3.5	24
Yearly cost of SPI per software engineer	$490–$2004	$1375	5
Productivity gain per year	9%–67%	35%	4
Early defect detection gain per year	6%–25%	22%	3
Yearly reduction in time to market	15%–23%	19%	2
Yearly reduction in post-release defect reports	10%–94%	39%	5
Business value (saving/cost of SPI)	4.0–8.8:1	5.0:1	5

SEI report on 13 organizations in the original study

They used a variety of process improvement techniques, not just SW–CMM

Process Improvement Data (contd)

CMM Level	Number of Projects	Relative Decrease in Duration	Faults per MEASL Detected during Development	Relative Productivity
Level 1	3	1.0	—	—
Level 2	9	3.2	890	1.0
Level 3	5	2.7	411	0.8
Level 4	8	5.0	205	2.3
Level 5	9	7.8	126	2.8

Results of 34 Motorola projects