PV260 - SOFTWARE QUALITY

LECT 12. Software Quality Management Process

Bruno Rossi
brossi@mail.muni.cz

LAB OF SOFTWARE ARCHITECTURES
AND INFORMATION SYSTEMS

FACULTY OF INFORMATICS
MASARYK UNIVERSITY, BRNO

lasaris

Outline

- Software Quality Management (SQM) introduction
- The impact of Quality on Users' choices
- SQM categories
  - Software Quality Planning (SQP)
  - Software Quality Assurance (SQA)
  - Software Quality Control (SQC)
  - Software Process Improvement (SPI)
- Software Process Maturity Levels
  - Process Levels and attributes
  - ISO/IEC 15504 (SPICE)
  - Mapping CMMI to ISO/IEC 15504
- Models for Software Quality
  - Personal Software Process (PSP)
  - Agile Process Maturity
  - Open Source Maturity Model (OMM)
  - Six Sigma
- Do better processes lead to a better software product?

"Essentially, all models are wrong, but some are useful"

George Box

Introduction

- Kitchenham gave five perspectives of software quality:

  1. Transcendental view
  → can be recognized but difficult to define exactly
  2. User view
  → fitness for purpose
  3. Manufacturing view
  → conformance to specification
  4. Product view
  → from inherent product characteristics
  5. Value-based view
  → depends on customer's willingness to pay

B. Kitchenham and S. L. Pfleeger, "Software quality: the elusive target [special issues section]," IEEE Software, vol. 13, no. 1, pp. 12–21, Jan. 1996.

Introduction

4 Views of Quality

Quality in Use
What' s the end-user' s experience?)

External Quality Attributes
(Does it pass all the tests?)

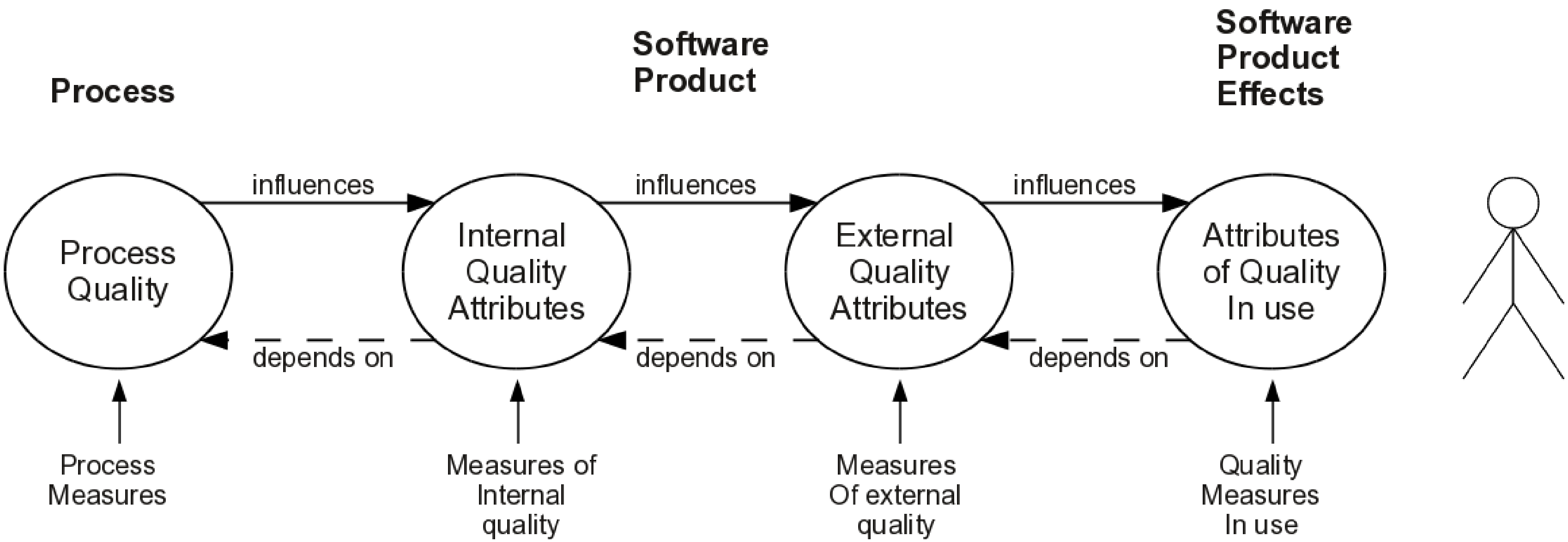
Internal Quality Attributes
(Is it well-designed?)

Process Quality
(Is it assembled correctly?)

We deal with process quality in this lecture

Introduction

- Quality is a concept that starts with the **development process**, goes on with the **software product** and finally to the **user** with the effects of **software usage**



What is Software Quality Management (SQM)

- *“**Software quality management (SQM)** is the collection of all **processes** that ensure that software products, services, and life cycle process implementations meet organizational software quality objectives and achieve stakeholder satisfaction”* (SWEBOK 3.0)
- SQM defines **processes**, **process owners**, **requirements for the processes**, **measurements of the processes and their outputs**, and **feedback** channels throughout the whole software life cycle.

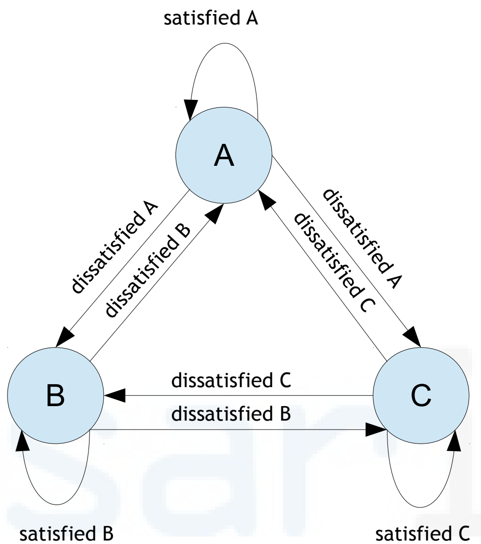
The Impact of Quality

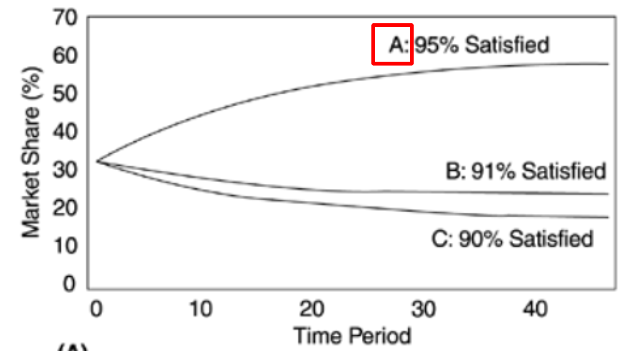
The impact of Quality

How much Quality is needed? Should we have 100% quality in our software products?

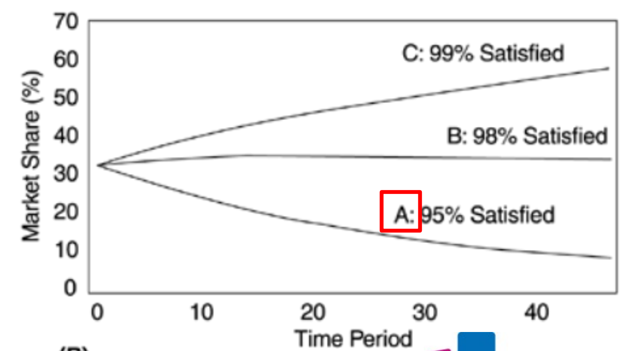
Think about some of the systems that are/have been market leaders, how was the quality of the proposed solution?

Babich has shown that **even by starting with the same level of customer satisfaction** what really matters is the **competition with the other companies**
This implies that you need to do better than your competitors in terms of quality





(A)



(B)

Babich, P. (1992) “Customer Satisfaction: How Good is Good Enough?” Quality Progress

The impact of Quality

- What makes software different
  - Really big economies of scale

    → additional physical inputs for production result in a non-proportional increase in output with a decrease in average costs
  - Increasing returns

    → *"tendency for that which is ahead to get further ahead, for that which loses advantage to lose further advantage"* (W.B. Arthur)
  - Network Externalities

    → the value of goods to users increases as more people adopt them
  - High initial costs

    → software is complex to design and to deliver to the market
  - Switching costs

    → switching to other software might be costly (e.g. training to re-do, change of infrastructure)

How users make their choice

- If we take a model like (Windrum, 1998), modelling the convergence of users to a single market, quality is one direct/indirect characteristic of goods:

$$u_i(x, p) = v_i(m_i - p) + w_i(x)$$

vector of characteristics of good

money available

price of good

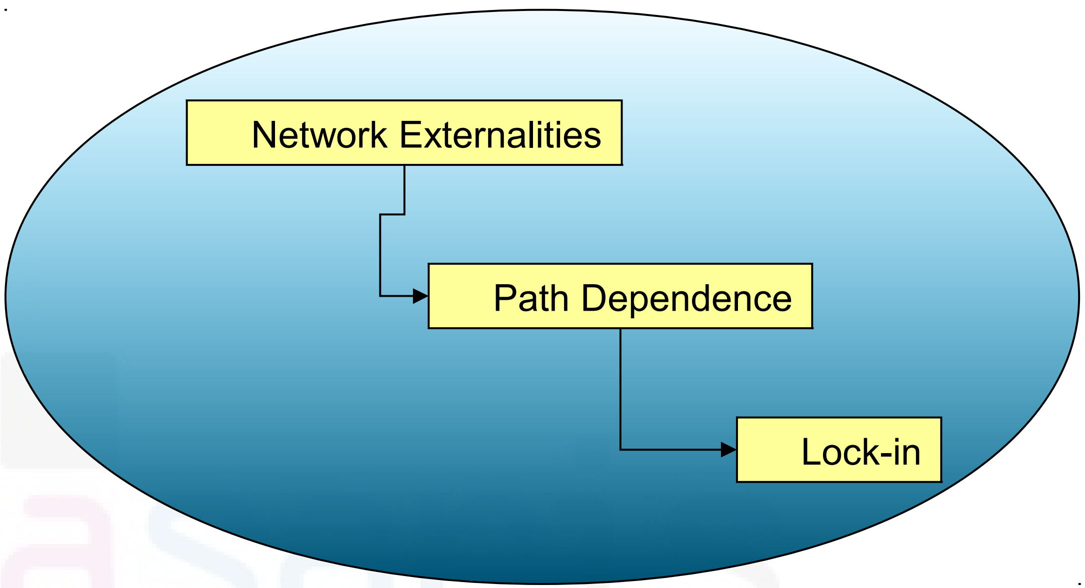
direct utility function

indirect utility of money that can be obtained in other markets

Network Externalities

Network Externalities: for some types of goods, the value of a good increases with the number other people adopting the same good

- *"These effects arise both because the ability to communicate and share data with others will be greater, and because it is more likely that complementary hardware, software, and wetware (i.e., brain cells) will be available, when there is a large base of users of the software"*, Katz & Shapiro



lasaris

Path Dependence

- **Path Dependence:** in common knowledge means 'history matters'
- But there is more.. If we consider **historical paths, not only the path depends on previous events** but also **produces a self-reinforcing mechanism that leads to the reinforcement of the path selected**. In this way, switching to another path will become at every step more costly
- In general, due to **increasing returns**, a phenomenon can assume contagious effects

A Formal definition of Path Dependence

- Given a discrete set of values of time T={t_1,t_2,..,t_n}, X_t represents the outcome at time t
- h_t, the history at time t, that is the combination of all the outcomes up to time t-1
- a dynamic process is a stochastic process equipped with a outcome function G that maps current history to next outcome
- A process is path-dependent if the outcome in any period depends on history

$$x_{t+1} = G(h_t)$$

- **Weak Path Dependent:** if the outcome probabilities depend upon the set of outcomes; the sequence of events {A,B,B,A} is considered the same as {A,B,A,B} for the evolution of the process at later stages;
- **Strong Path Dependent:** if the outcome probabilities depend upon the sequence of outcomes; events {A,B,B} has a set of outcomes completely different from {B,B,A}

This **essentially means that nr. of users of a sw will depend on the past history** – for some platforms **quality will be less important**

This is specific to software → if I have a car I do not really care about the types of cars that other users will have

Sustaining and Disruptive Innovations (1/3)

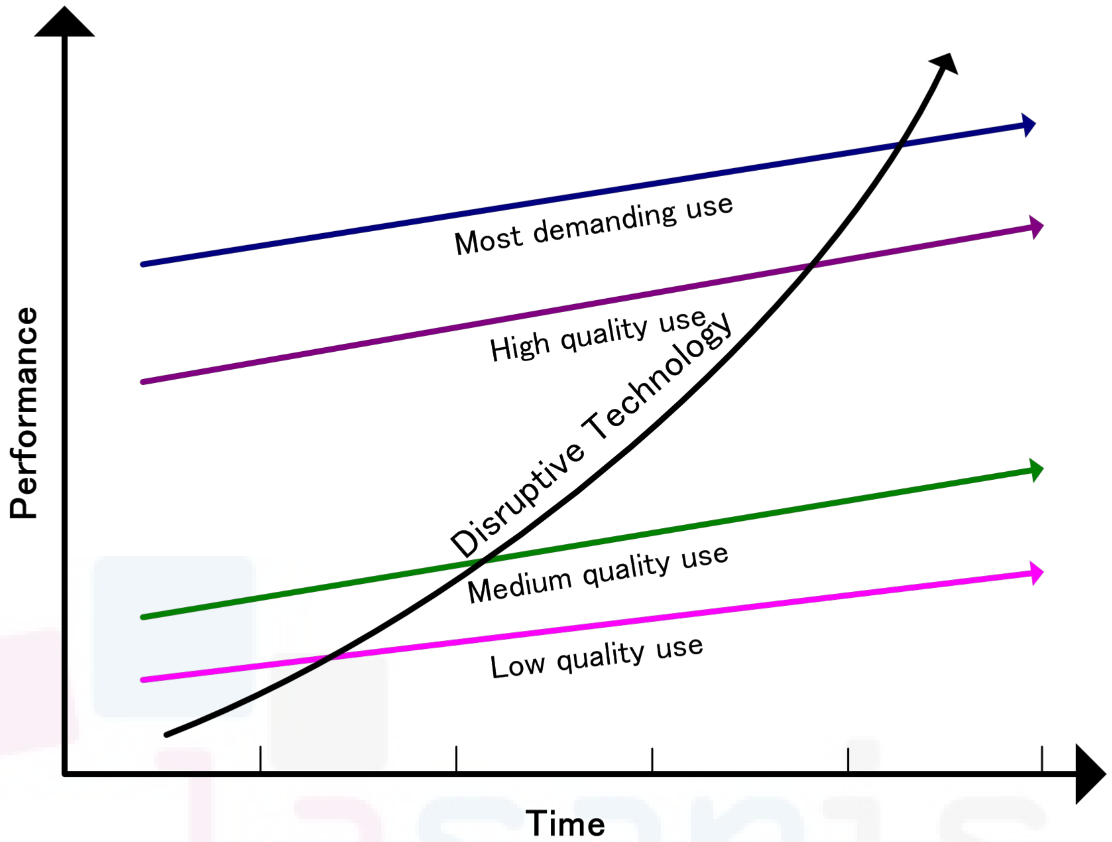
- So are we destined to keep the current status in the future (e.g. using Facebook, Google, etc..)?
- One nice answer comes from Christensen (2013) that identified two types of technologies:
  - **Sustaining** – an innovation that improves a product in either expected or unexpected way, **but does not lead to a paradigm shift**
  - **Disruptive** – an innovation that can potentially create a new market (e.g. *facebook* creating the whole social networking idea, or touch screens for the whole mobile phones industry)
- Disruptive innovations are a way to change from the *status quo*

→ can you name some companies that were market leaders before a disruptive technology appeared?

Christensen, C. (2013). *The innovator's dilemma: when new technologies cause great firms to fail.* Harvard Business Review Press.

Sustaining and Disruptive Innovations (2/3)

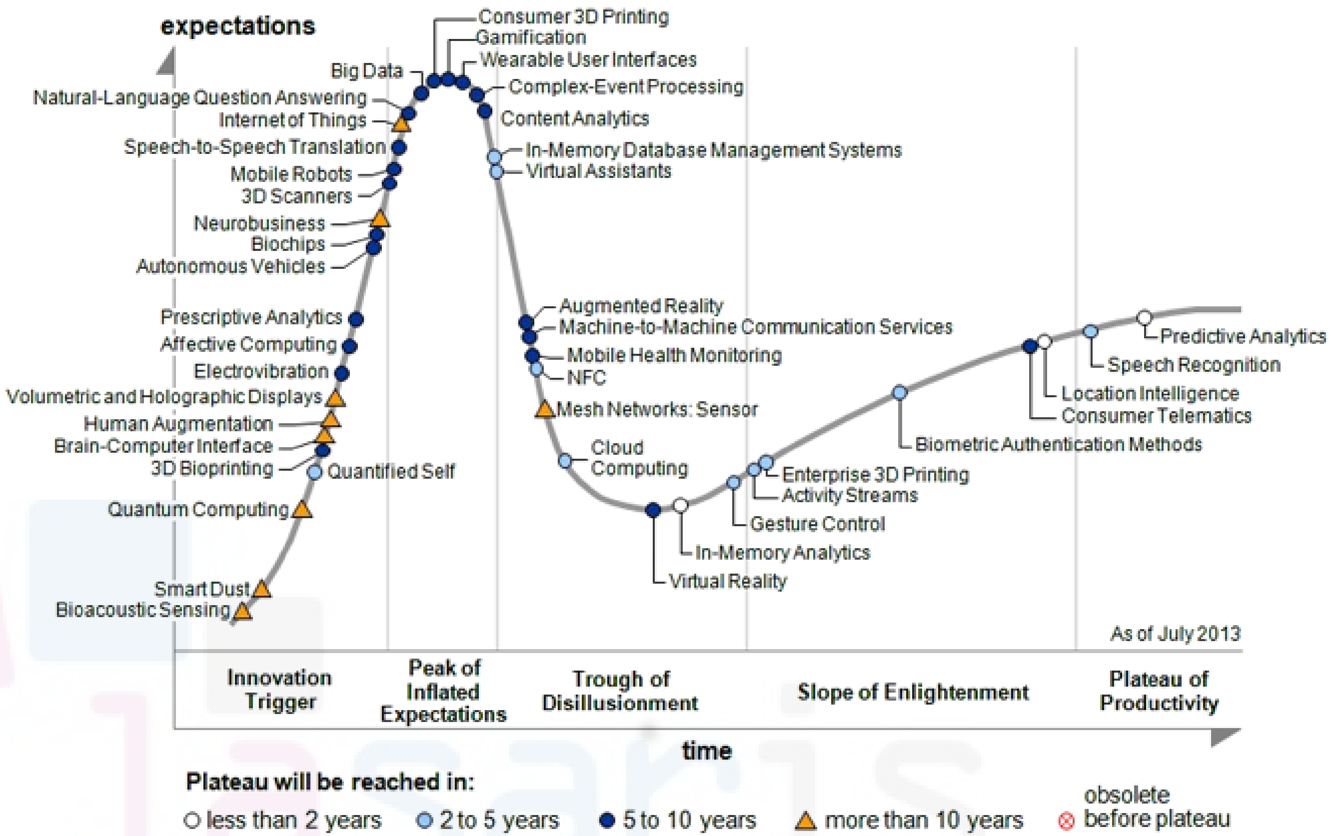
- **Disruptive technologies:** initially the **quality might be lower** than current technologies, but will catch-up quickly



Christensen, C. (2013). *The innovator's dilemma: when new technologies cause great firms to fail.* Harvard Business Review Press.

Sustaining and Disruptive Innovations (3/3)

- Gartner's Hype Cycle (GHC) for Emerging Technologies → maturity of technologies in a domain



← 2013 GHC

Q&A

General Questions

a) Can you **name one innovation** either in Software or in the IT world that impressed you particularly in the last few years (that is a **disruptive innovation**)?

b) Thinking **in terms of 5-10 years from now**, what do you think will **be a disruptive technology in the software/IT world**?

SQM Categories

SQM comprises four subcategories

A. Software quality planning (SQP)

which quality standards are to be used, defining specific quality goals, and estimating the effort and schedule of software quality activities

B. Software quality assurance(SQA)

define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes

C. Software quality control (SQC)

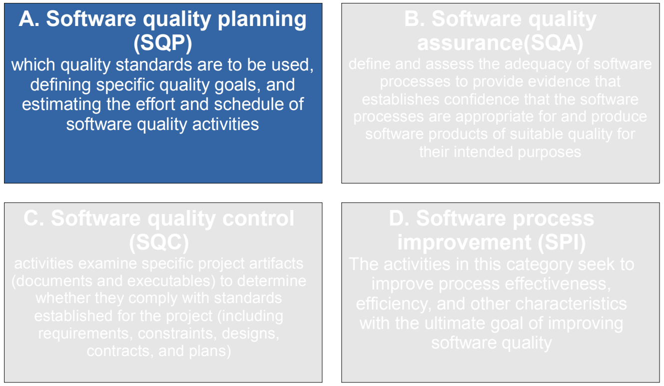
activities examine specific project artifacts (documents and executables) to determine whether they comply with standards established for the project (including requirements, constraints, designs, contracts, and plans)

D. Software process improvement (SPI)

The activities in this category seek to improve process effectiveness, efficiency, and other characteristics with the ultimate goal of improving software quality

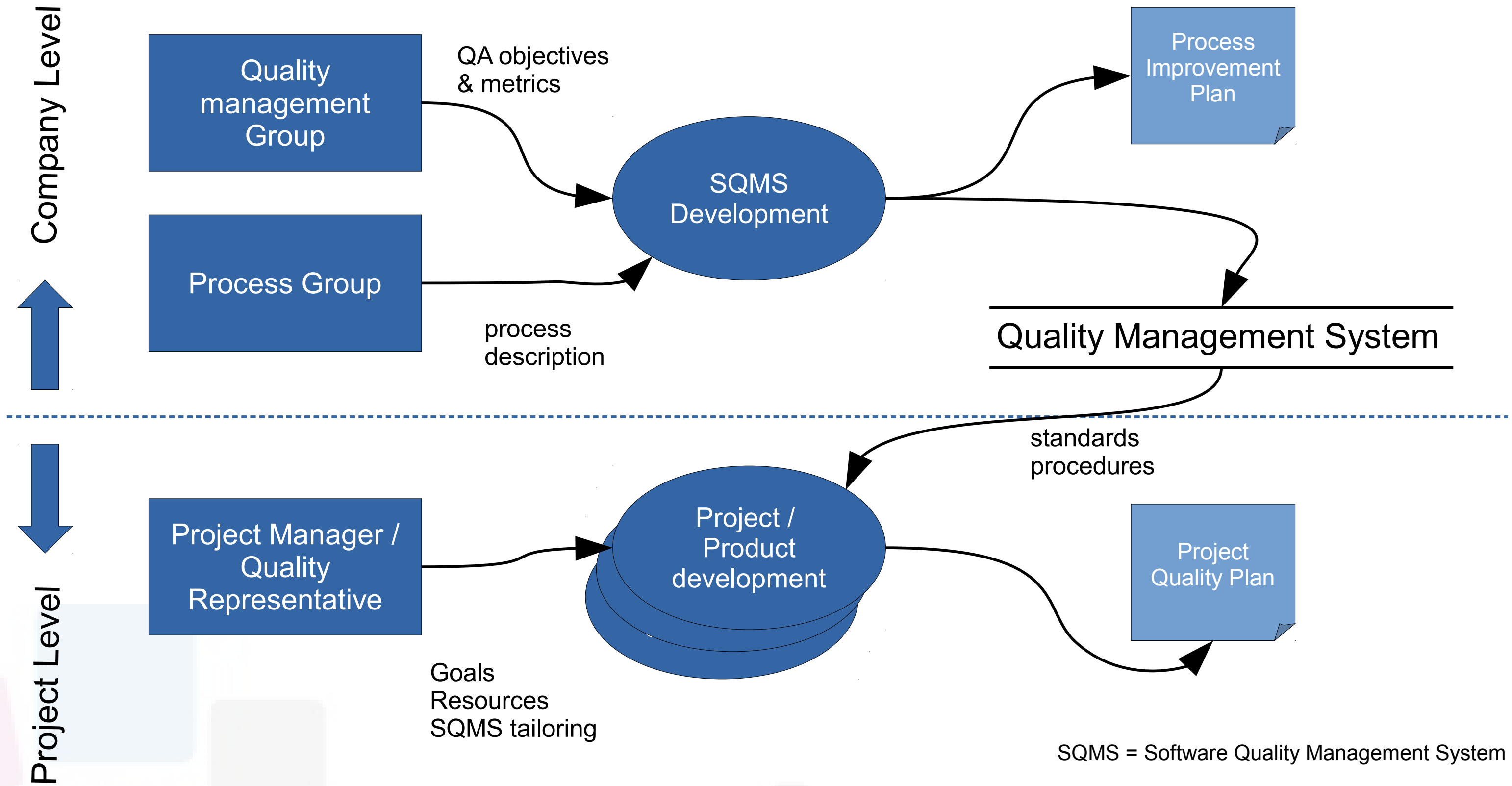
A. Software Quality Planning

- A quality plan defines how an **organization will reach the quality objectives**
- Usually covers
  - **Quality objectives** and **goals**
  - **Quality management scope**
  - **Organisation & responsibilities**
  - **Resource requirements**
  - **Cost benefit analysis**
  - **Activities and deliverables**
  - **Schedule**
  - **Risk analysis**



A. Software Quality Planning

- Quality at the organization and project Levels



Source: http://www.chambers.com.au/glossary/quality_planning.php

A. Software Quality Planning

- IEEE Std 730-2014 – Outline for software quality planning:

1. Purpose & Scope

2. Definitions & acronyms

3. Reference documents

4. SQA Plan Overview:

4.1 Organization & independence
4.2 Software Product Risk
4.3 Tools
4.4 Standards, practices and conventions
4.5 effort, resource, schedules

5. Activities, Outcomes and tasks:

5.1 Product Assurance:
5.1.1 Evaluate plans for conformance
5.1.2 Evaluate product for conformance
5.1.3 Evaluate product for acceptability
5.1.4 Evaluate product life cycle support for conformance
5.1.5 Measure products
5.2 Process assurance:
5.2.1 Evaluate life cycle support for conformance
5.2.2 Evaluate environments for conformance
5.2.3 Evaluate subcontractor processes for conformance
5.2.4 Measure processes
5.2.5 Assess staff skills & knowledge

6. Additional Considerations

6.1 Contract review
6.2 Quality Measurement
6.3 Waiver and deviations
6.4 Task repetition
6.5 Risks in performing SQA
6.6 Communication strategy
6.7 Conformance process

4. SQA Records:

7.1 Analyze, identify, collect, file, maintain, dispose
7.2 Availability of records

Heimann, D. I. (2014). *An Introduction to the New IEEE 730 Standard on Software Quality Assurance*, SQP Vol 16, N.3

A. Software Quality Planning

IEEE Std 730-2014 – how is Agile considered?

- **Agile → the product backlog is the contract**, the standards help in defining **the role of the backlog** as contract
- The SQA product part in IEEE730 can be used to defined **the "done" criteria**
- **Non-conformances** to standards are **inserted in the backlog** and **addressed in sprints in which are scheduled**
- Acceptance is a **continuous process in Agile**
- IEEE730 contains an appendix with details about Agile adoption of the standard

Heimann, D. I. (2014). *An Introduction to the New IEEE 730 Standard on Software Quality Assurance*, SQP Vol 16, N.3

B. Software Quality Assurance

- SQA means **monitoring constantly the software engineering process** to ensure that the approaches/methods/processes applied lead to quality within the project

A. Software quality planning (SQP)
which quality standards are to be used, defining specific quality goals, and estimating the effort and schedule of software quality activities

B. Software quality assurance(SQA)
define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes

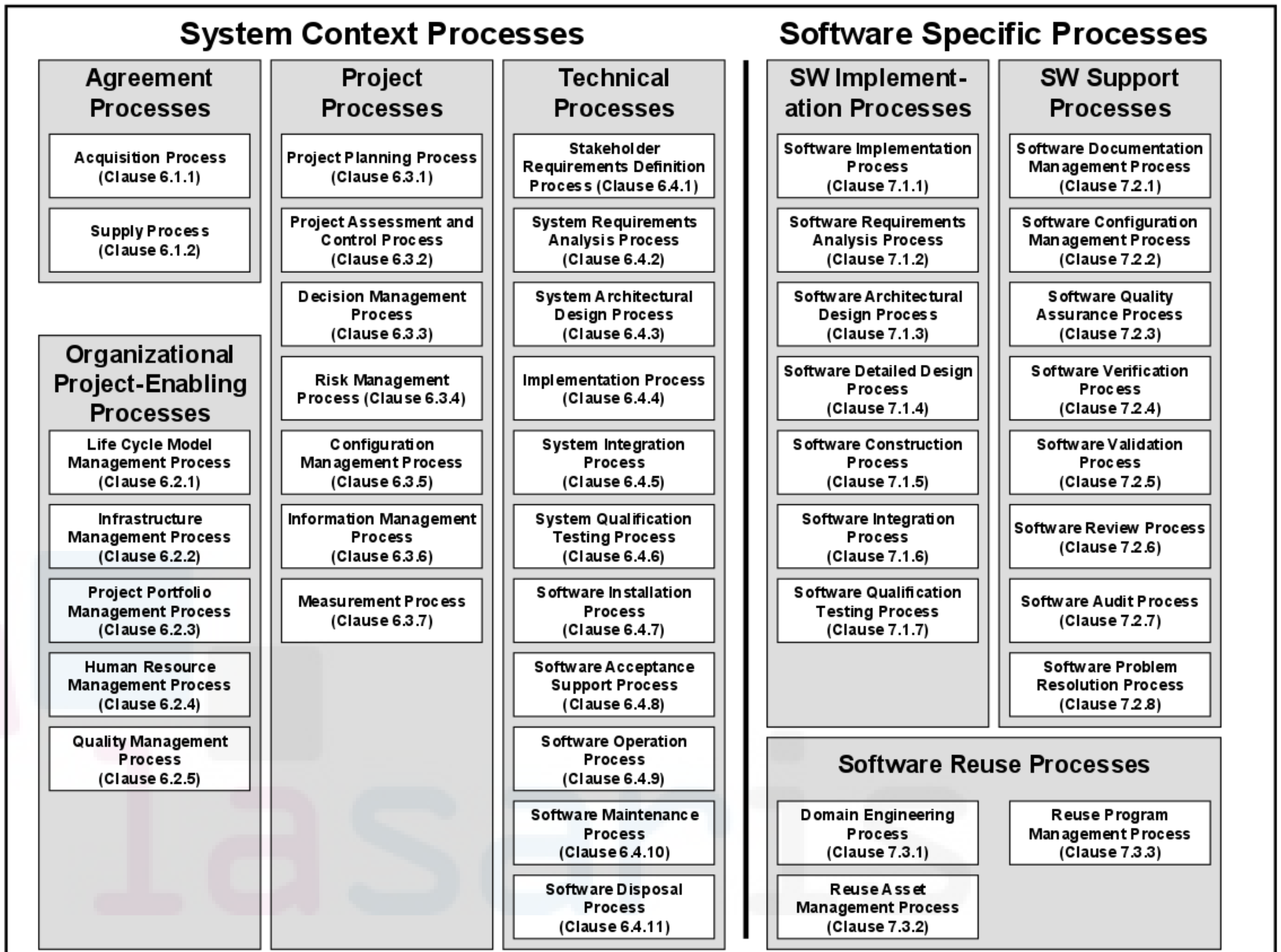
C. Software quality control (SQC)
activities examine specific project artifacts (documents and executables) to determine whether they comply with standards established for the project (including requirements, constraints, designs, contracts, and plans)

D. Software process improvement (SPI)
The activities in this category seek to improve process effectiveness, efficiency, and other characteristics with the ultimate goal of improving software quality

B. Software Quality Assurance

- Relevant Standard: ISO/IEC/IEEE 12207:2008: Systems and software engineering - Software life cycle processes

  → can you spot SQA in this schema?



lasaris

B. Summary of IEEE Stds related to SWQA

- **IEEE Std 730** → format and content of a **software quality assurance plan**
- **IEEE Std 1061** → describes a methodology—spanning the life cycle—for **establishing quality requirements** and for identifying, implementing, and validating the **corresponding measures**.
- **IEEE Std 1465** *(withdrawn standard)* → describes quality requirements specifically suitable for software "packages". It is expected to be replaced by an IEEE adoption of ISO/IEC 25051

C. Software Quality Control

- SQC **means to constantly monitor** the software engineering process to **check for conformance to applied standards** (e.g. CMMI) or produced artefacts
- Some examples of methods
  - → The Goal Question Metrics Approach (seen on lecture 2)
  - → The Plan-Do-Check-Act method
  - → Total Software Quality Control

D. Software Process Improvement

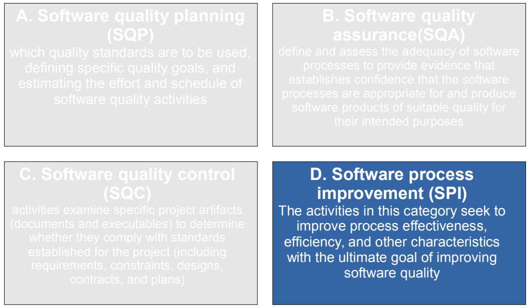
- Improve process effectiveness, efficiency and other characteristics **with the aim to improve software quality**
- Very often **software process improvement practices are embedded within the process** (e.g. capability models)
- Some methods:

→ Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI)

→ ISO/IEC 15504 (SPICE)

→ ISO 9001 Specification (seen during PA017 SEII)

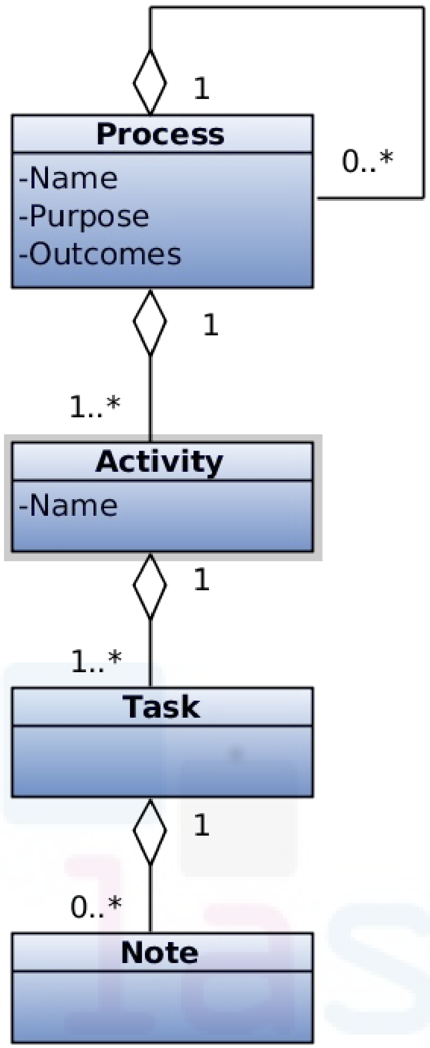
→ PSP and TSP



Software Process Maturity Levels

What is a process?

- According to ISO/IEC 12207/15288:2007



Processes require a purpose and outcome. All processes have at least one activity

Activities are constructs for grouping together related tasks

A task is a detailed arrangement for the implementation of a process. It can be a requirement ("shall"), a recommendation ("should") or a permission ("may")

Notes are used to explain better the intent or mechanism of a process

Process Maturity Levels

- As defined in ISO/IEC 15504-2 (SPICE)

Capability Level	Process Capability	
0	Incomplete Process	→ The process is not implemented or fails to achieve the purpose
1	Performed Process	→ The process is achieving the purpose
2	Managed Process	→ The process is now running in a managed way (planned, monitored, adjusted) – work products are established, controlled and maintained
3	Established Process	→ The managed process is now implemented using a defined process capable of achieving process outcomes
4	Predictable Process	→ The established process now operates within defined limits to achieve its process outcomes
5	Optimizing Process	→ The predictable process is continuously improved to meet relevant current and projected business goals

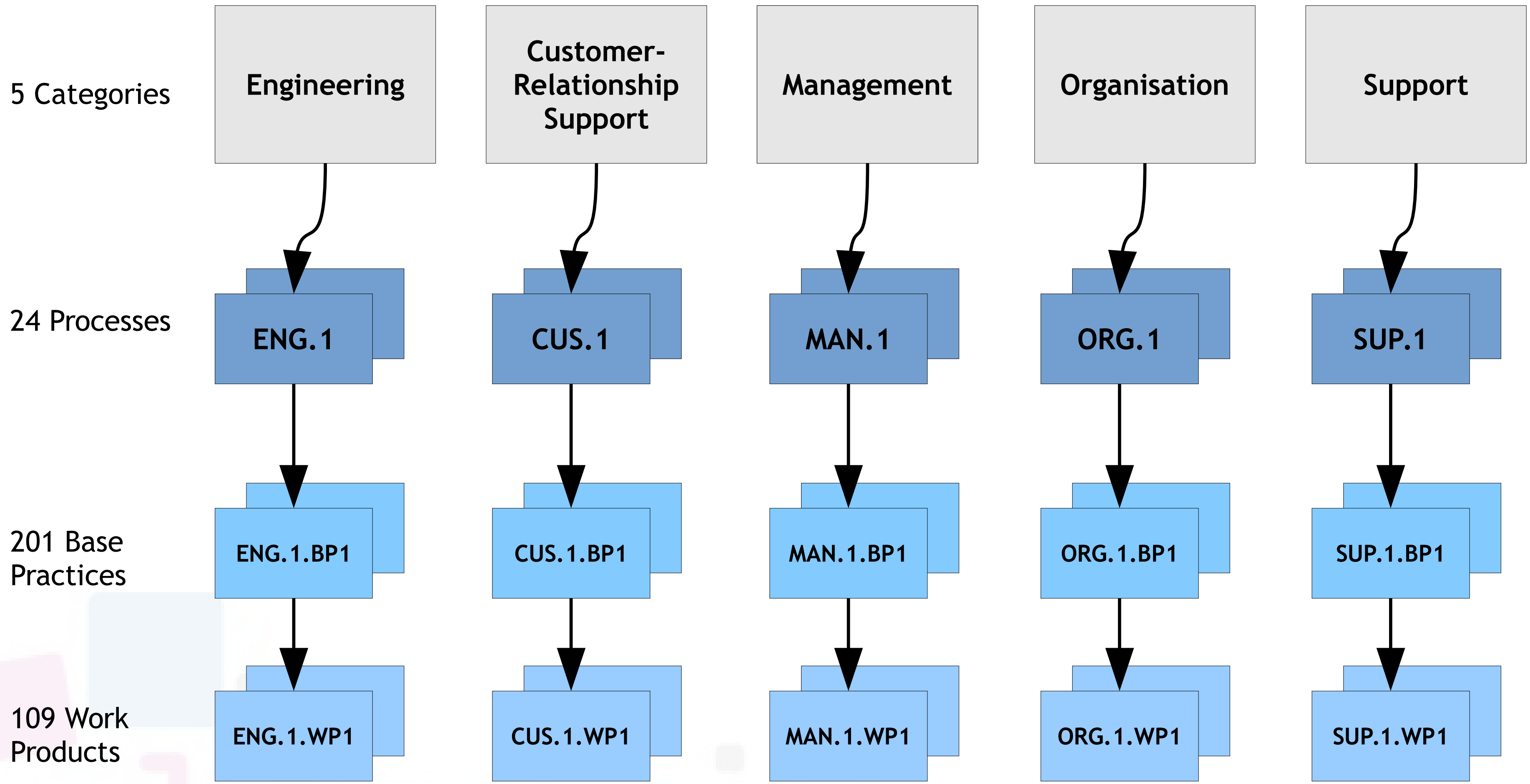
Process Maturity Levels & Attributes

- As defined in ISO/IEC 15504-2 (SPICE)

Process Attribute ID	Capability Levels and Process Attributes
	Level 0: Incomplete process
	Level 1: Performed process
PA 1.1	Process performance
	Level 2: Managed process
PA 2.1	Performance management
PA 2.2	Work product management
	Level 3: Established process
PA 3.1	Process definition
PA 3.2	Process deployment
	Level 4: Predictable process
Process Attribute ID	**Capability Levels and Process Attributes**
PA 4.1	Process measurement
PA 4.2	Process control
	Level 5: Optimizing process
PA 5.1	Process innovation
PA 5.2	Continuous optimization

lasaris

SPICE - Overall



Example Process Definition (1/2)

Process ID	SUP.1
Process Name	Quality assurance
Process Purpose	The purpose of the Quality assurance process is to provide independent assurance that work products and processes comply with predefined provisions and plans.
Process Outcomes	As a result of successful implementation of this process: 1) a strategy for conducting quality assurance is developed, implemented and maintained; 2) quality assurance is performed independent of the activity or project being performed; 3) evidence of quality assurance is produced and maintained; 4) adherence of products, processes and activities to agreed requirements are verified, documented, and communicated to the relevant parties; 5) problems and/or non-conformance with agreement requirements are identified, recorded, communicated to the relevant parties, tracked and resolved; and 6) quality assurance has the independence and authority to escalate problems to appropriate levels of management. NOTE 1: Quality assurance should be coordinated with, and make use of, the results of other supporting processes such as verification, validation, joint review, audit and problem management. NOTE 2: Verification and validation may be subject to quality assurance. NOTE 3: Independent quality assurance should be established as a separate functional role within an organization.
Base Practices	**SUP.1.BP1 : Develop project quality assurance strategy.** A project level strategy for conducting quality assurance is developed. This strategy is consistent with the organisational quality management strategy. [Outcome 1] NOTE 1: The quality assurance process may be co-ordinated with the related SUP.2 Verification, SUP.4 Joint Review, Validation and Audit processes.

5 Process Categories
(SUP=Support)

Process ID	SUP.1
	SUP.1.BP2: Develop and maintain an organisation structure which ensures that quality assurance is carried out and report independently. Quality assurance team members are not directly responsible to the project organisation – they work independently from it. [Outcome 2] **SUP.1.BP3: Develop and implement a plan for project quality assurance based on a quality assurance strategy.** [Outcome 3] NOTE 2: Quality assurance plan may contain quality assurance activities, a schedule of activities, assigned responsibilities, resources required, guidelines and quality standards for requirement, design, coding and testing work products. **SUP.1.BP4: Maintain evidence of quality assurance.** Define and maintain the records that demonstrate that planned quality assurance activities have been implemented. [Outcome 3] **SUP.1.BP5: Assure quality of work products.** Carry out the activities according to the quality assurance plan to ensure that the work products meet the quality requirements. [Outcome 4] NOTE 3: Product quality assurance activities may include reviews, audits, problem analysis, reports and lessons learned that improve the work products for further use. NOTE 4: Non conformances detected in work products may be entered into the problem resolution management process (SUP.9) to document, analyze, resolve, track to closure and prevent the problems. **SUP.1.BP6: Assure quality of process activities.** Carry out the activities according to the quality assurance plan to ensure assurance that the processes meet the defined requirements of the project [Outcome 4] NOTE 5: Problems detected in the process definition or implementation should be entered into the process improvement process (PIM.3) to describe, record, analyze, resolve, track to closure and prevent the problems. NOTE 6: Process quality assurance activities may include process assessments and audits, problem analysis, regular check of methods, tools, documents and the adherence to defined processes, reports and lessons learned that improve processes for future projects.

Example Process Definition (2/2)

Process ID	SUP.1
	NOTE 7: In case of supplier involvement, the quality assurance of the supplier should cooperate with the quality assurance of the customer and all other involved parties. **SUP.1.BP7: Track and record quality assurance activities.** Records of quality assurance activities are produced and retained. [Outcome 3, 4, 5] **SUP.1.BP8: Report quality assurance activities and results.** Regularly report performances, deviations, and trends of quality assurance activities to relevant parties for information and action. [Outcome 5] NOTE 8: The quality assurance may use an independent path to report regularly the results to the management and other relevant stakeholders. **SUP.1.BP9: Ensure resolution on non-conformances**. Deviations or non-conformance found in process and product quality assurance actitvities should be analyzed, corrected and further prevented. [Outcome 5] **SUP.1.BP10: Implement an escalation mechanism.** Develop and maintain the escalation mechanism that ensures that quality assurance may escalate problems to appropriate levels of management to resolve them. [Outcome 6]

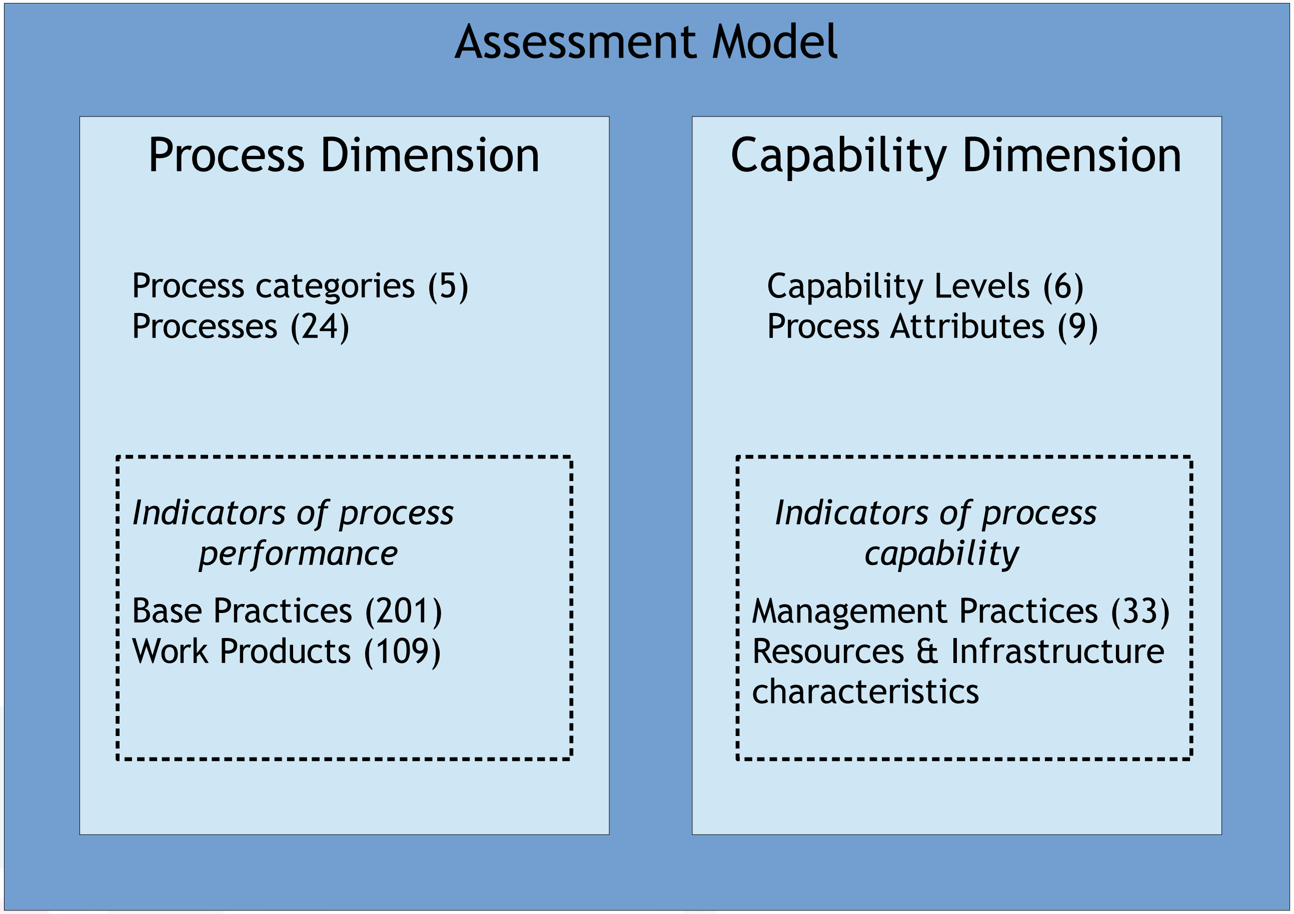
Overall **24 processes** are specified

Output Work Products	
08-13	Quality plan [Outcome 3, 5, 6]
13-04	Communication Record [Outcome 5]
13-07	Problem record [Outcome 3, 4]
13-18	Quality Record [Outcome 2, 3, 4]
13-19	Review Record [Outcome 2, 3, 4]
14-02	Corrective action plan [Outcome 3, 5]
18-07	Quality criteria [Outcome 4]

Process Maturity Levels & Attributes

- SPICE is a **two-dimensional level model**
  - Processes and categories on one side **(Process Dimension)** → includes Base practices, work products, characteristics → Does the process reach its goals?
  - Capability of processes on the other side **(Capability Dimension)** → includes levels, process attributes, process attributes, management practices → How well is a specific goal met?

Process Maturity Levels & Attributes



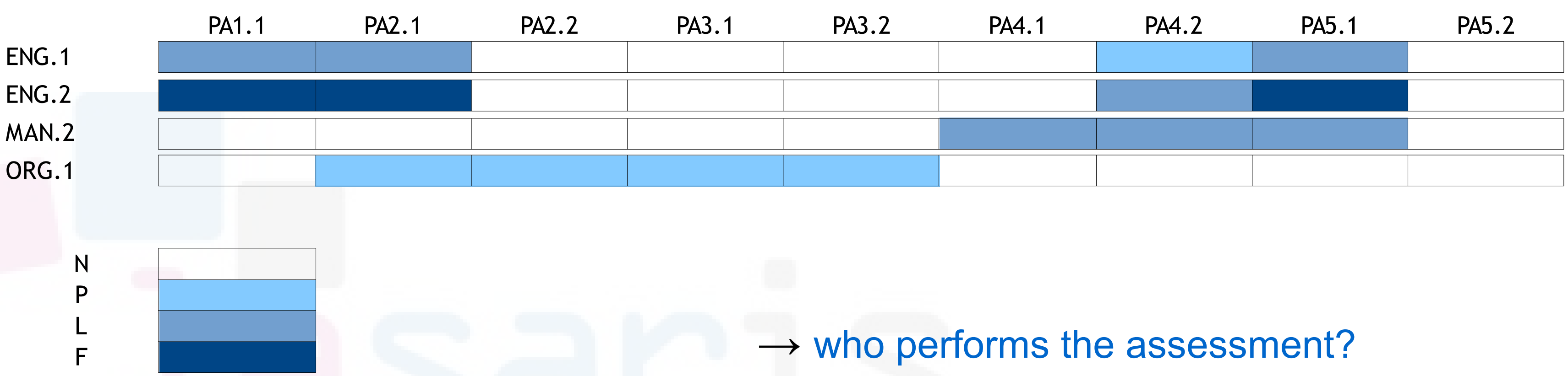
Process Maturity Levels & Attributes

Assessment Model

Process Dimension	Capability Dimension
Process categories (5) Processes (24)	Capability Levels (6) Process Attributes (9)
Indicators of process performance Base Practices (201) Work Products (109)	*Indicators of process capability* Management Practices (33) Resources & Infrastructure characteristics

Assessment Scale

- up to 15% – N (Not performed/achieved)
- > 15% to 50% – P (Partial)
- > 50 to 85% - L (Large)
- > 85% - F (Full performance / achievement)



→ who performs the assessment?

CMMI

Capability Maturity Model Integrated

Upgrade from CMM appearing around year 2000

5 — Continuous organizational **focus** on **innovation** and **improvement**

Optimized
Organizational strategy

4 — Process and projects managed **based on measures**

Quantitatively Managed
Processes and activities are managed based on measures

3 — **Processes defined**, with performance managed

Defined
Work done according processes

2 — **Projects managed**, based on defined plans

Managed
Activities are managed

1 — Process **imprédictible**, **reactively** managed

Initial
Project success depends on individual performance

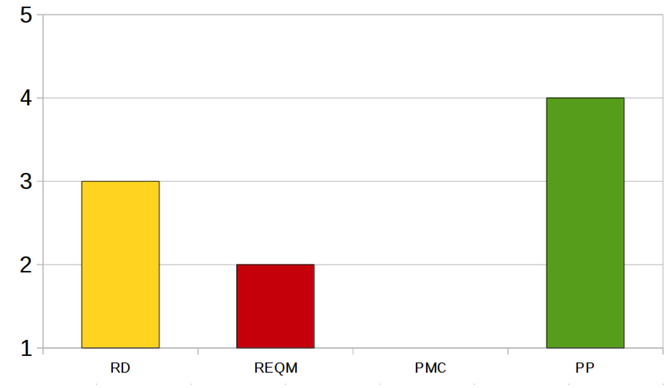
CMMI

Staged → assessment of the maturity of the **entire software process**

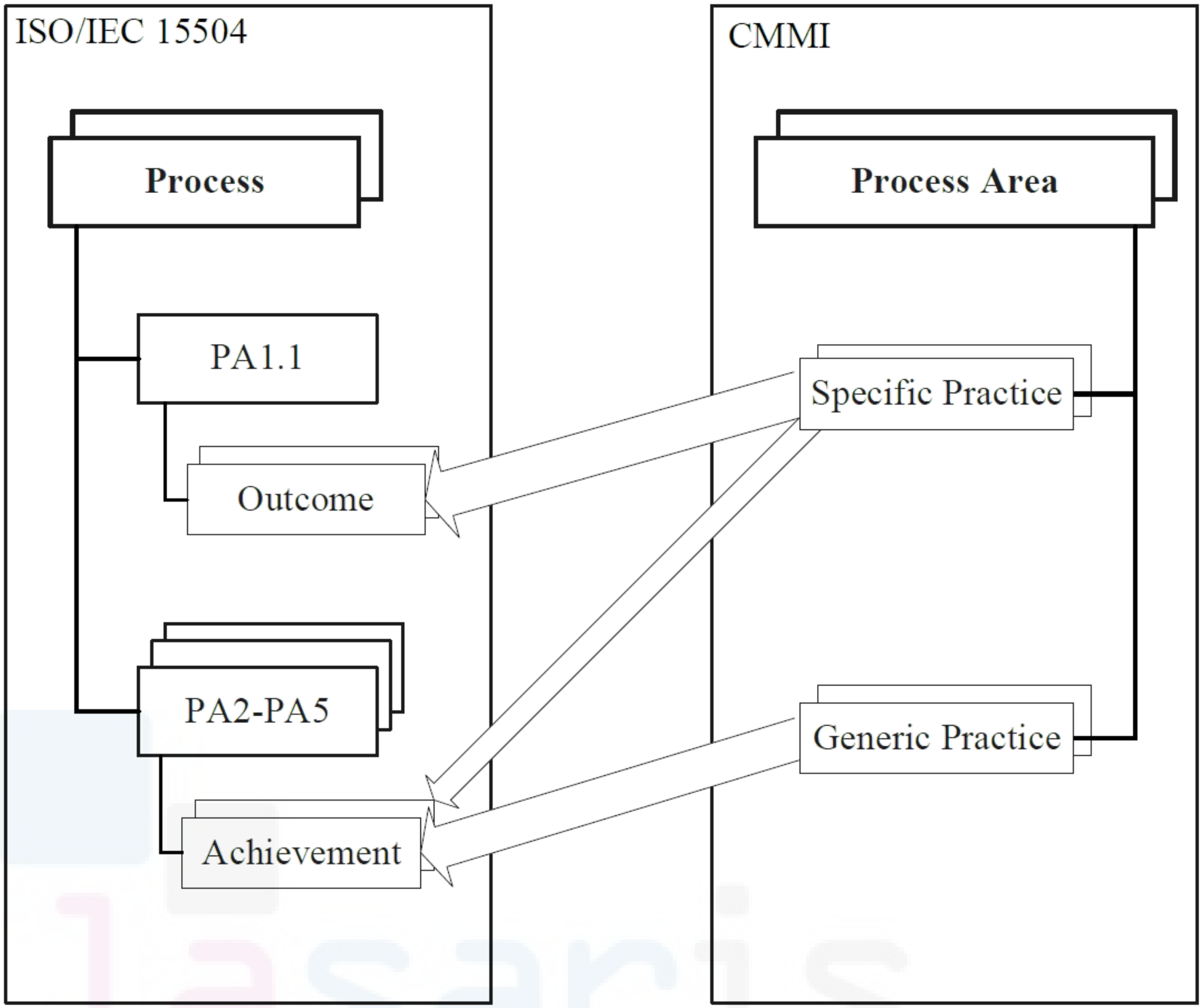
Continuous → assessment of the **capabilities of different Process Areas (PA)**

Sample Pas:

- Requirements Development (RD)
- Requirements Management (REQM)
- Project Monitoring and Control (PMC)
- Project Planning (PP)
- Process and Product Quality Assurance (PPQA)
- Quantitative Project Management (QPM)
- Risk Management (RSKM)
- Supplier Agreement Management (SAM)
- Technical Solution (TS)
- Validation (VAL)
- ...

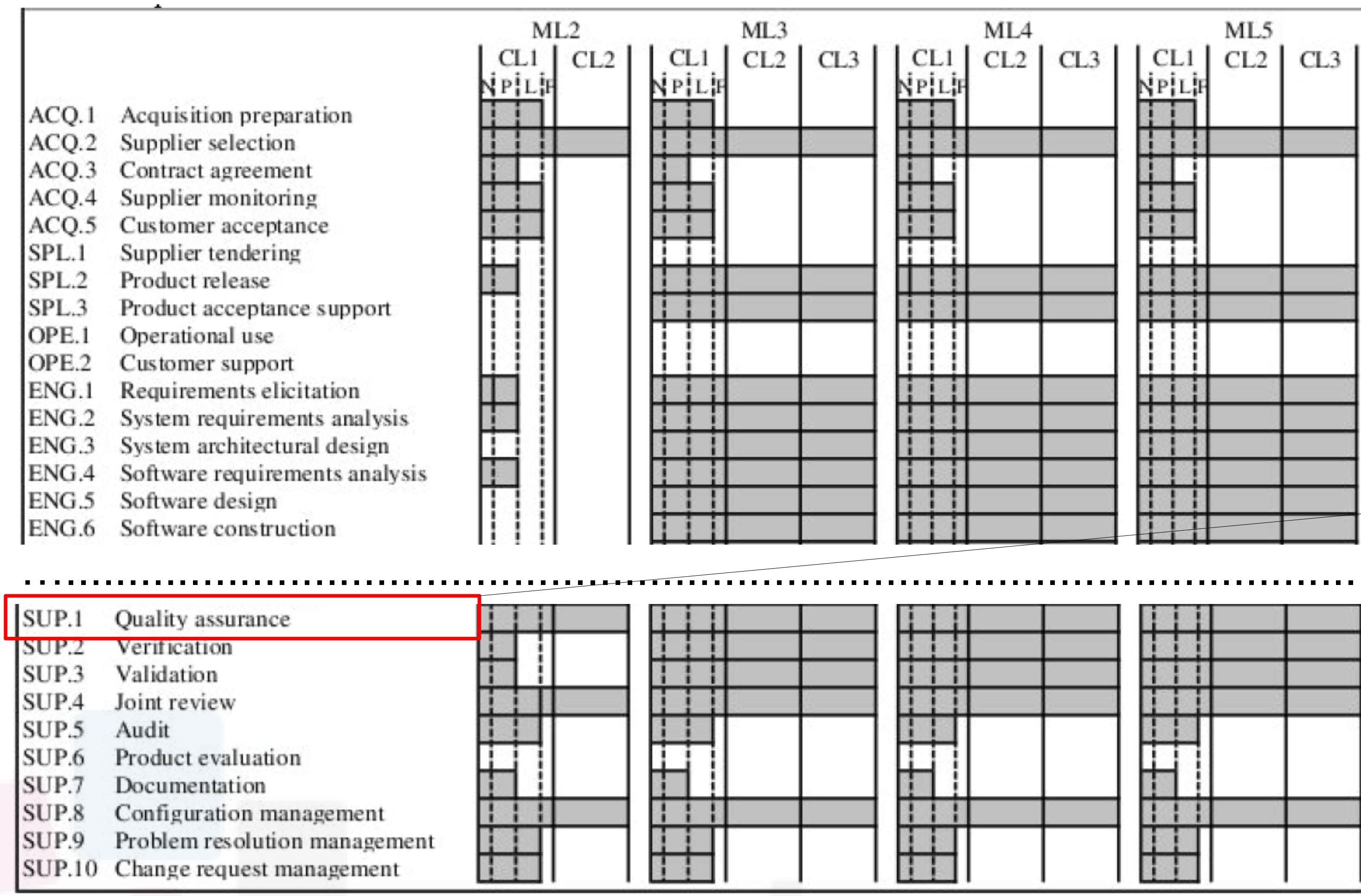


Mapping CMMI to ISO/IEC 15504



Ragaisis, S., Peldzius, S., & Simenas, J. (2010). Mapping CMMI-DEV Maturity Levels toISO/IEC 15504 Capability Profiles. Assessment, 23, 24.

Mapping CMMI to ISO/IEC 15504



This was the sample process we saw some slides ago

Legenda: up to 15 % – N (Not performed/achieved), > 15 % to 50 % – P (Partial), > 50 to 85 % – L (Large), and F (Full performance / achievement) > 85 % - “ML2”- “ML5” maturity levels in CMM-i - “CL1”-“CL5” are capability levels in ISO/IEC 15504

Ragaisis, S., Peldzius, S., & Simenas, J. (2010). Mapping CMMI-DEV Maturity Levels toISO/IEC 15504 Capability Profiles. Assessment, 23, 24.

Models for Software Quality

Simplest Quality Management Form

- What is the name of the **simplest quality process management** practice in your opinion?

  → *Actually, it involves no process*

Cowboy Coding

- Cowboy Coders write code according to their rules
- Some sentences you might have heard:

“If possible, the **customer should only see the final versions of the product.** It is important to **minimize the contact with the customer so time is not wasted”**

“The **code is mine** and **none is allowed to touch it!”**

“I **do not need** any **analysis, design** nor **documentation”**

“Even if it is **broken, do not touch it! Try to hide it!”**

“People who **need comments** in order to **understand my code** are **too dumb to be working with me “**

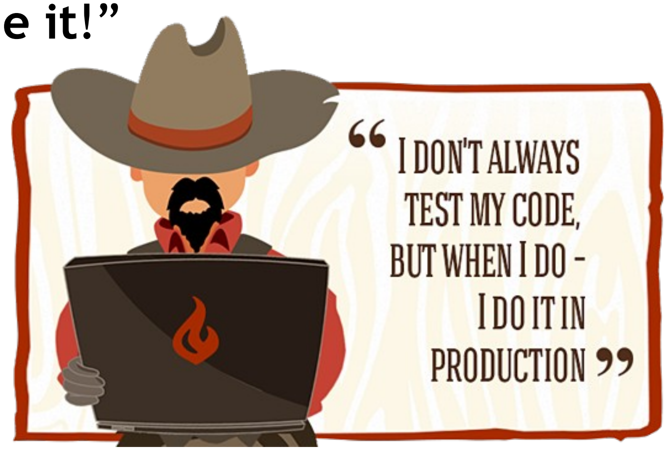
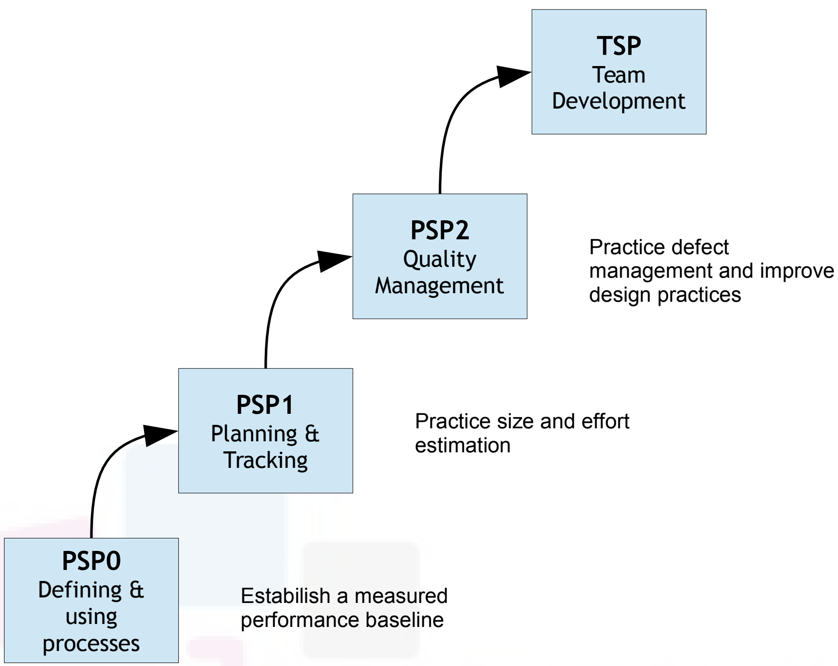


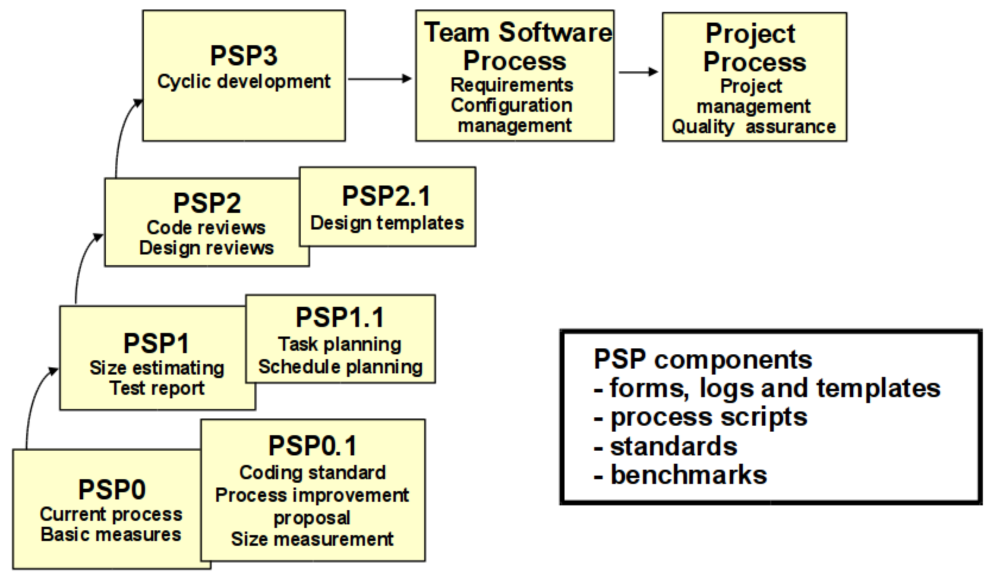
Image: https://www.cs.utexas.edu/blog/cowboy-rides-away-now

→ See http://c2.com/cgi/wiki?CowboyCoder

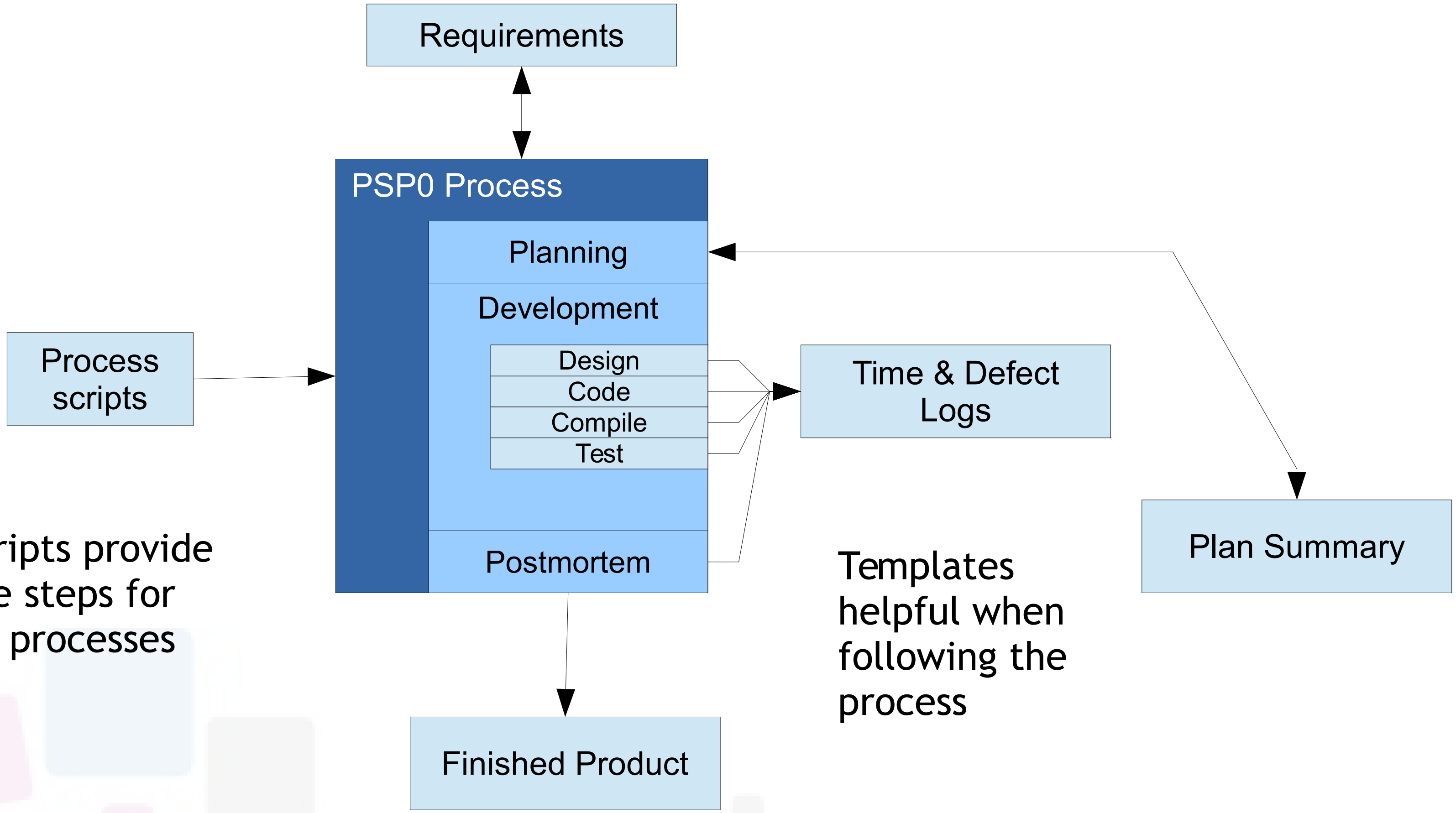
Personal Software Process (PSP)

- The Personal Software Process (PSP) is a **disciplined software development process** that works at the **individual level**





PSP0 – First Level



Scripts provide the steps for all processes

Templates helpful when following the process

PSP0.1 – Improvement

- Emphasizes making accurate and precise size measurements
- Incorporates measuring the size of the programs produced
- Accounts for various types of LOC in the programs produced
- Begins to look at process improvement
- The following elements are added
  - **Estimating** and **reporting software size**
  - Use of a **coding standard**
  - Recording **process problems** and **improvement ideas**

PSP0.1 – Improvement

- All is based on forms, scripts and logs

Table C3 PSP0.1 Project Plan Summary Example

Student	Student 11	Date	2/1/94
Program	Object LOC Counter	Program #	3A
Instructor	Humphrey	Language	C

Program Size (LOC)	*Plan*	*Actual*	*To Date*
Base(B)		*87* (Measured)	
Deleted (D)		*0* (Counted)	
Modified (M)		*6* (Counted)	
Added (A)		*113* (T-B+D-R)	
Reused (R)		*0* (Counted)	*0*
Total New & Changed (N)	*90*	*119* (A+M)	*315*
Total LOC (T)		*200* (Measured)	*396*
Total New Reuse		*0*	*0*

Time in Phase (min.)	*Plan*	**Actual**	**To Date**	**To Date %**
Planning	10	11	36	6.4
Design	25	21	63	11.2
Code	75	97	249	44.2
Compile	20	4	35	6.2
Test	45	39	105	18.7
Postmortem	20	33	75	13.3
Total	195	205	563	100.0

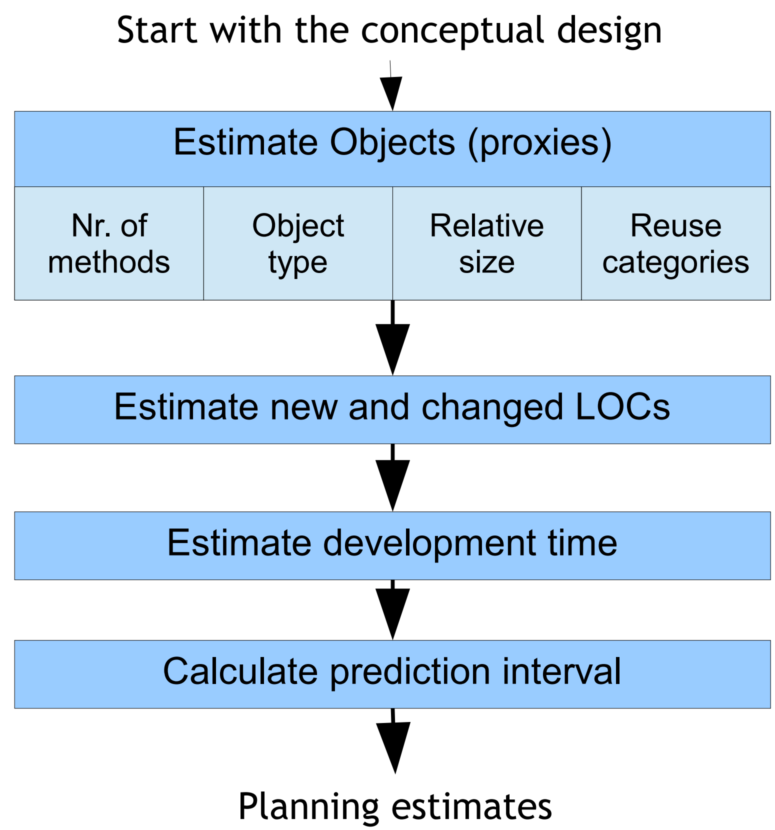
Defects Injected	**Actual**	**To Date**	**To Date %**
Planning	0	0	0
Design	1	3	11.5
Code	8	23	88.5
Compile	0	0	0
Test	0	0	0
Total Development	9	26	100.0

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning	0	0	0
Design	0	0	0
Code	0	0	0
Compile	2	13	50.0
Test	7	13	50.0
Total Development	9	26	100.0
After Development	0	0	

W. S. Humphrey, "A Discipline for Software Engineering", 1995

PSP1 – Second Level

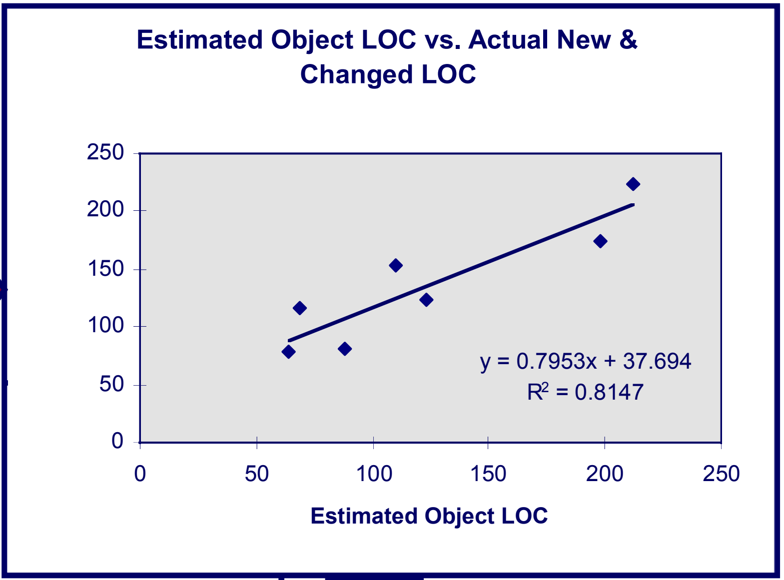
- PSP1 introduces the concept of software effort estimation and the usage of historical data
- Using the PROBE (PROxy-Based Estimating) size estimating method
- Using linear regression, PROBE size estimation, regression analysis is based on historical estimated object LOC (the x data) and actual new and changed LOC (the y data)



W. S. Humphrey, "A Discipline for Software Engineering", 1995

PSP1 - Second Level

Object/Method	Type	Obj LOC
Input_Data	I/O	22
List	Data	27
Calc_Mean	Calc	11
Calc_SD	Calc	11
Print_Result	I/O	22
		93



Note: The *est obj LOC* would typically include estimated modifications (M) and additions (BA) to the base code. For this example, there is no base program.

Regression Parameters

β_0 = 38 β_1 = 0.8 r^2 = 0 .8

Est N&C LOC = $\beta_0 + \beta_1$ * Est obj LOC

Est N&C LOC = 38 + 0.8 * 93

Est N&C LOC = 112 LOC

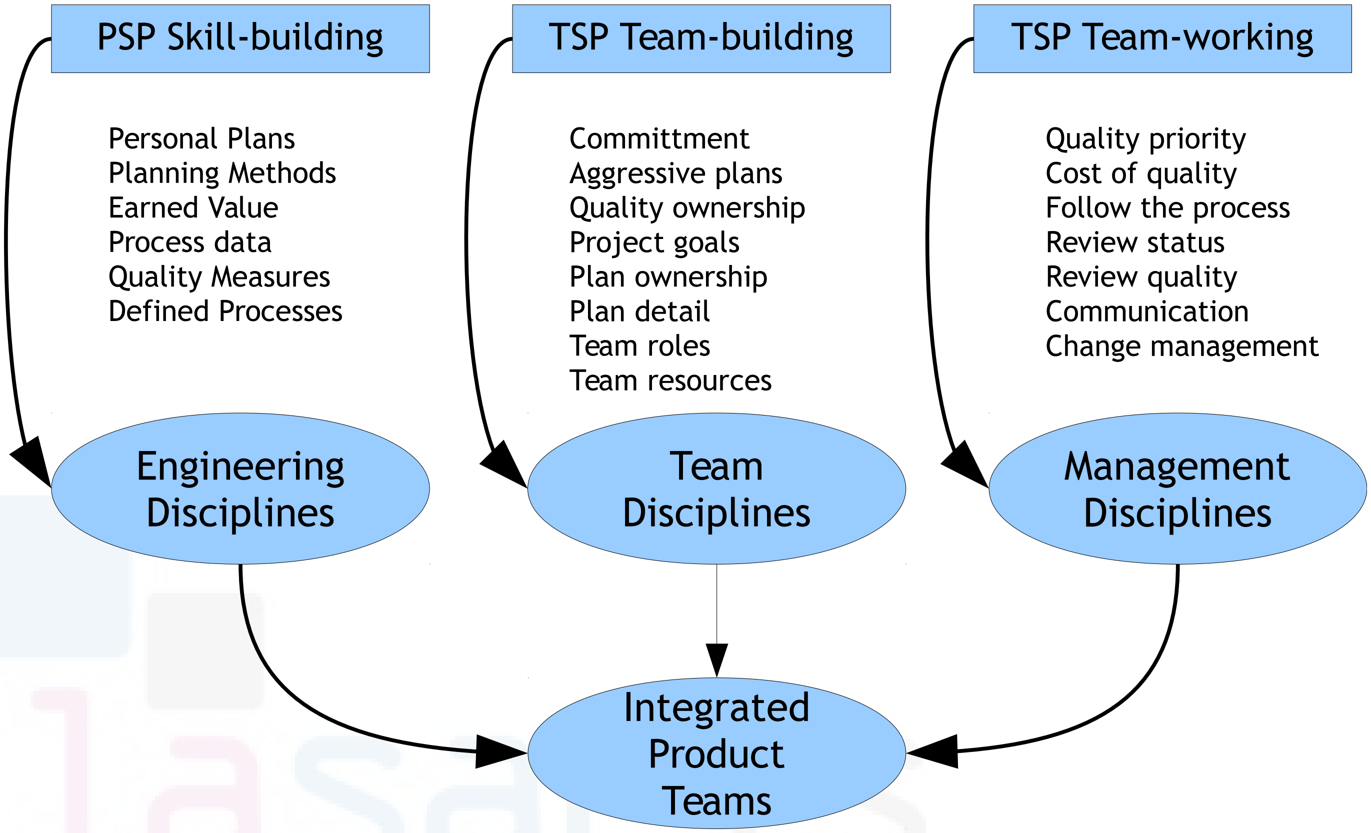
W. S. Humphrey, "A Discipline for Software Engineering", 1995

PSP2 – Third Level

- PSP2 introduces **design and code reviews methods** for evaluating and improving the quality of your reviews
- There are three new process elements
  - PSP2 **project plan summary**
  - PSP2 **design review checklist**
  - PSP2 **code review checklist**

Personal Software Process (PSP)

- The idea behind PSP is that it should lead to more team-aware processes once developers have tried a self-disciplined approach



→ see hackystat

Agile Process Maturity

There are around 40 Agile Maturity models that sometimes adapt SPICE/CMMI levels/processes – each one uses different naming for levels:

Level1	Level2	Level3	Level4	Level5
• Rhetorical stage • Team level maturity • Neutral or Chaotic • Emergent • Engineering Best Practices • Introductory • Collaborative • Dormant • No Agile • Waterfall • Non-Agile • Core Agile Development • Adherence to Agile Principles • Getting Started • Improvising	• Certified stage • Department Level Maturity • Collaborative • Continuous Practices at Component Level • Learn • Novice • Evolutionary • Speed • Early Adoption • Forming • Minimum • Discipline Agile Delivery • Repeteable Process across the organization • Scrum at project level • Practicing	• Plausible stage • Business Level Maturity • Operating (consistent exhibition of competence) • Cross component continuous integration • Leverage • Intermediate • Effective • Reactive • Self Service • Agile • Consolidated • Agility at scale • Scalability – SCRUM of SCRUMS	• Respectable stage • Project Management Level Maturity • Adaptive • Cross Journey Continuous integration • Advanced • Adaptive • Responsive • The Lake effect • Performing • Items on the right • SCRUM at Enterprise Level • Governed	• Measured stage • Management Level Maturity • Innovating • On Demand Just in Time Release • Optimise • Insane • Ambient • Scaling • Coexistence with non-agile • Enterprise transformation • Matured

T. Schweigert, D. Vohwinkel, M. Korsaa, R. Nevalainen, and M. Biro, “Agile Maturity Model: A Synopsis as a First Step to Synthesis,” in Systems, Software and Services Process Improvement, F. McCaffery, R. V. O’Connor, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2013, pp. 214–227.

Agile Process Maturity

Some approaches suggest even to reuse SPICE Process definitions, example:

Process ID	AMP.1	
Process name	Backlog management	
Process purpose	The purpose of the Backlog Management process is to make sure that relevant items like requirements are collected, described, properly stored, prioritized estimated and solutions are delivered.	
Process outcomes	Outcome 1	The backlog is visible at a defined storage place
	Outcome 2	The items in the backlog are agreed between the relevant stakeholders
	Outcome 3	The backlog contains the complete set of items Note: no shadow backlogs are used
	Outcome 4	The items in the backlog are prioritized Note: the classification of prioritizing shall be clear. It shall be defined in which manner the backlogs are to be prioritized (e. g. regarding business value, complexity, etc.)
	Outcome 5	The items in the backlog are estimated
	Outcome 6	The assignment of items to an iteration (or increment) is visible
	Outcome 7	Delivery of items are visible

T. Schweigert, M. Ekssir-Monfared, and M. Ofner, “An Agile Management Process Group for TestSPICE®,” in Systems, Software and Services Process Improvement, F. McCaffery, R. V. O’Connor, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2013, pp. 228–236.

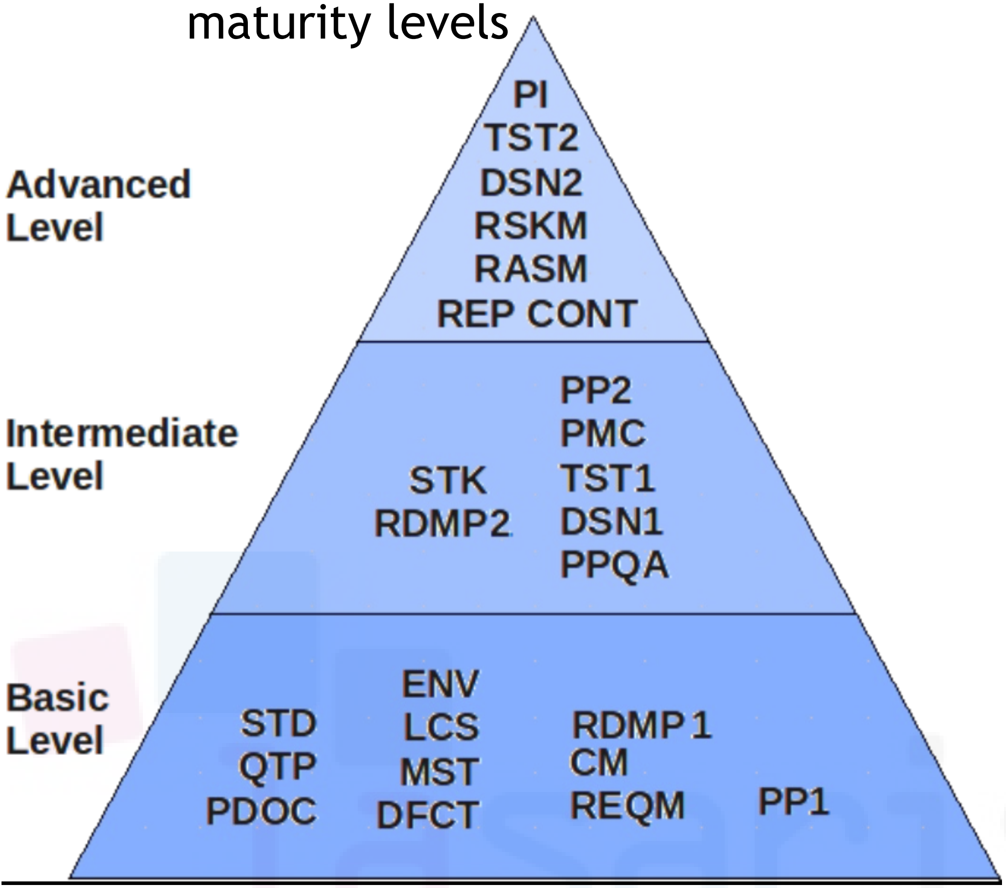
Agile Maturity Matrix

Level1 – ad hoc Agile	Level2 – doing agile	Level3 – being agile	Level4 – thinking agile	Level5 – culturally agile
Agile is not yet used or agile practices are used sporadically	Teams start to exhibit some agile habits	Lean portfolio management	Communities of practice support agile habits	Lean and agile are part of organizational culture
Variable quality	Consistency across teams is still variable	Mature embodiment of essential characteristics and behaviour of agile	Successful use of agile at scale	Perfecting waste reduction, smooth flow of delivery
Predominantly manual testing	Some knowledge sharing activities under way	Disciplined Agile delivery processes and practices with continual improvement and repeatable results	Success even with teams in multiple geographies	Sustainable pace of innovation
Very little cross-project knowledge and collaboration	Use of agile tools and practices become commonplace	Respect for people and continuous improvement	Measurement systems in place keep track of business value delivered	Continuous organizational learning and optimization of the work process and the work products
Success achieved primarily through heroic individual efforts	Solution quality improves	Appropriate agile governance	Autonomation: automation with a human touch	
	Standard work is defined			

Source: http://www.agigante.it/different-levels-of-agile-in-a-company/

Open Source Maturity Model (OMM)

- One of the existing models is a simplified version of CMMI taking into account the distributed nature of Open Source development
- Trustworthy elements (TWE) are micro-characteristics that allow to define maturity levels



Source: http://qualipso.icmc.usp.br/OMM/

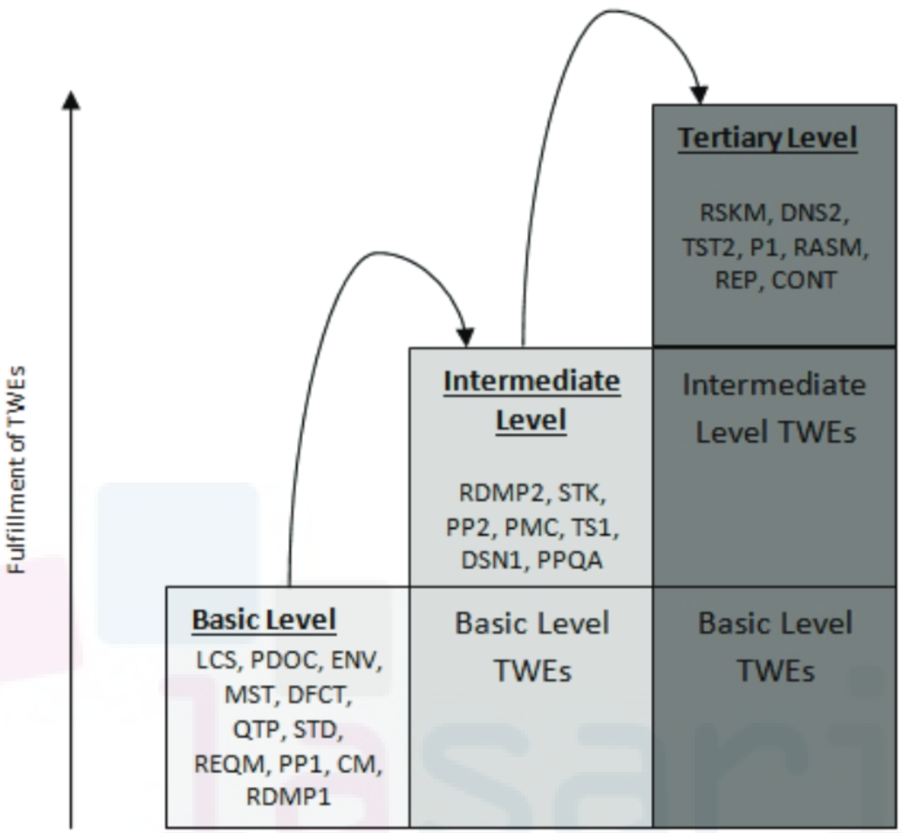
PDOC - Product Documentation
STD - Use of Established and Widespread Standards
QTP - Quality of Test Plan
LCS - Licenses
ENV - Technical Environment
DFCT - Number of Commits and Bug Reports
MST - Maintainability and Stability
CM - Configuration Management
PP1 - Project Planning Part 1
REQM - Requirements Management
RDMP1 - Availability and Use of a (product) roadmap

RDMP2 - Availability and Use of a (product) roadmap
STK - Relationship between Stakeholders
PP2 - Project Planning Part 2
PMC - Project Monitoring and Control
TST1 - Test Part 1
DSN1 - Design Part 1
PPQA - Process and Product Quality Assurance

PI - Product Integration
RSKM - Risk Management
TST2 - Test Part 2
DSN2 - Design 2
RASM - Results of third party assessment
REP - Reputation
CONT - Contribution to FLOSS Product from SW Companies

Open Source Maturity Model (OMM)

- The Goal Question Metric approach is used to define the elements of the current OMM based on the TWEs
- All levels of the GQM are aggregated according to following rating calculation:



$$R(P_i)=\frac{\sum M_i}{count(M)}$$

$$R(G_i)=\frac{\sum P_i}{count(P)}$$

$$R(TWE_i)=\frac{\sum G_i}{count(G)}$$

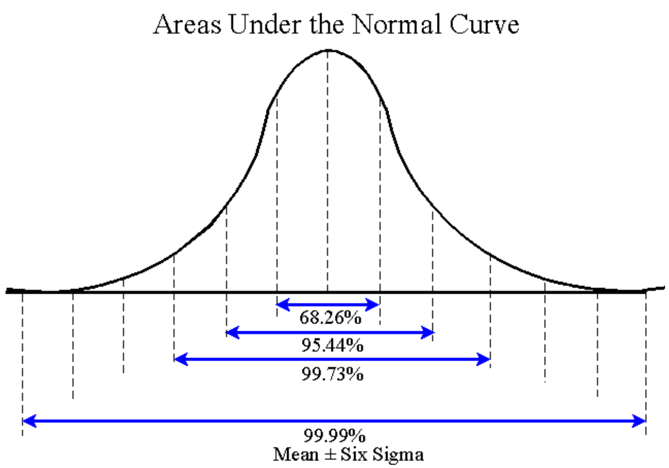
No weighting of metrics, Maturity Level calculated by using practices (not TWEs)

$$R(ML)=\frac{\sum P_i}{max\sum P_i}$$

Six Sigma

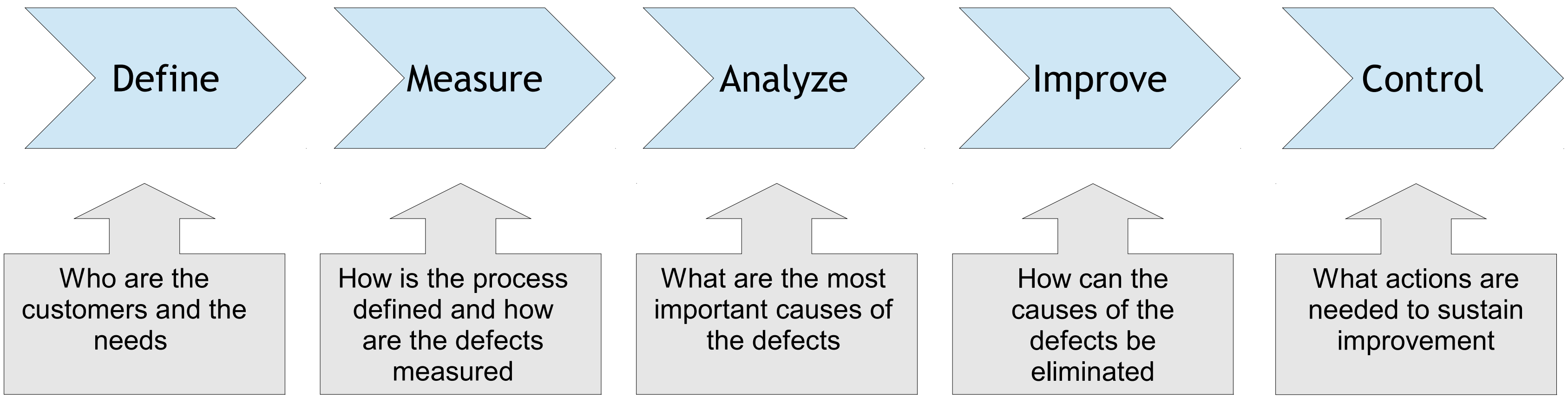
- A measure of Software Quality developed at Motorola
- The focus of Six Sigma is on **eliminating defects**, that is everything that is **outside from the customers specifications**
- **No more than +/- six times the standard deviation from the process mean** → 3,4 Defects Per Million datapoints
- Data is a key to understand the underlying processes and take decisions

Sigma Level	Defects x 1M
2	308.537
3	66.807
4	6.210
5	233
6	3,4



6σ

Six Sigma - Process



In Six Sigma, a defect is defined as *"Any product, service, or process variation which prevents meeting the needs of the customer and/or which adds cost, whether or not it is detected"*

Six Sigma - Process

Define
- Define Project scope
- Estailish formal project

Measure
- Identify needed data
- Obtain data set
- Evaluate data quality
- Summarize & baseline data

Analyze
- Explore data
- Characterize process & problem
- Update improvement project scope & scale

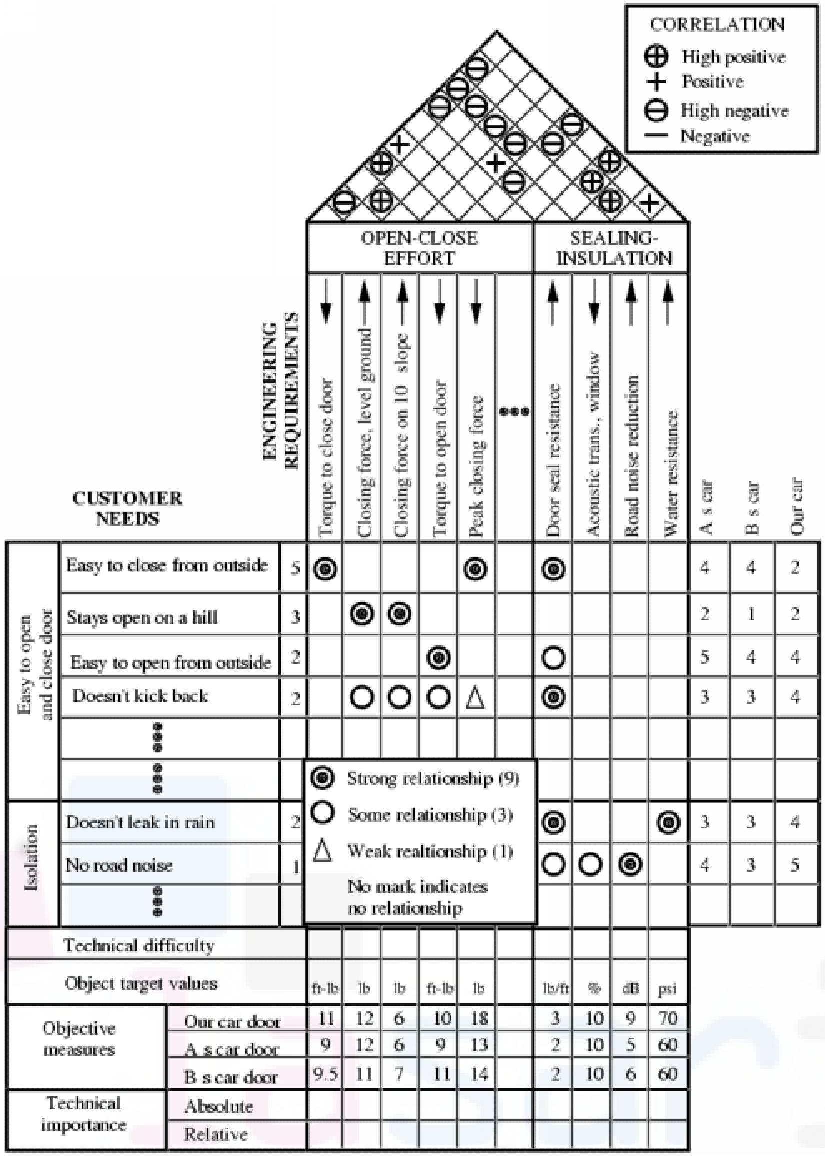
Improve
- Identify possible solutions
- Select solution
- Implement (pilot as needed)
- Evaluate

Control
- Define control method
- Implement
- Document

Six Sigma - Tools

Define	Measure	Analyze	Improve	Control
• Benchmark • Contract/charter • Kano Model • Voice of the customer • Voice of the Business • Quality Function Deployment (QFD)	• GQ(I)M and indicator templates • Data collection methods • Measurement System Evaluation	• Cause & Effect Diagram/Matrix • Failure models & effects analysis • Statistical inference • Reliability Analysis • Root Cause Analysis • Hypothesis Test	• Design of experiments • Modeling • ANOVA • Tolerancing • Robust Design • System Thinking • Decision & Risk Analysis • Performance Analysis Model	• **Statistical Controls:** • Control charts • Time series methods • **Non-Statistical controls:** • Procedural adherence • Performance Management • Preventive measures
Basic Tools (Histogram, scatter plots, run charts, pareto charts, cause & effect diagram, Control chart, descriptive statistics), baseline process flow map, project management, management by fact, sampling techniques, survey methods, defect metrics				

Six Sigma – SQFD

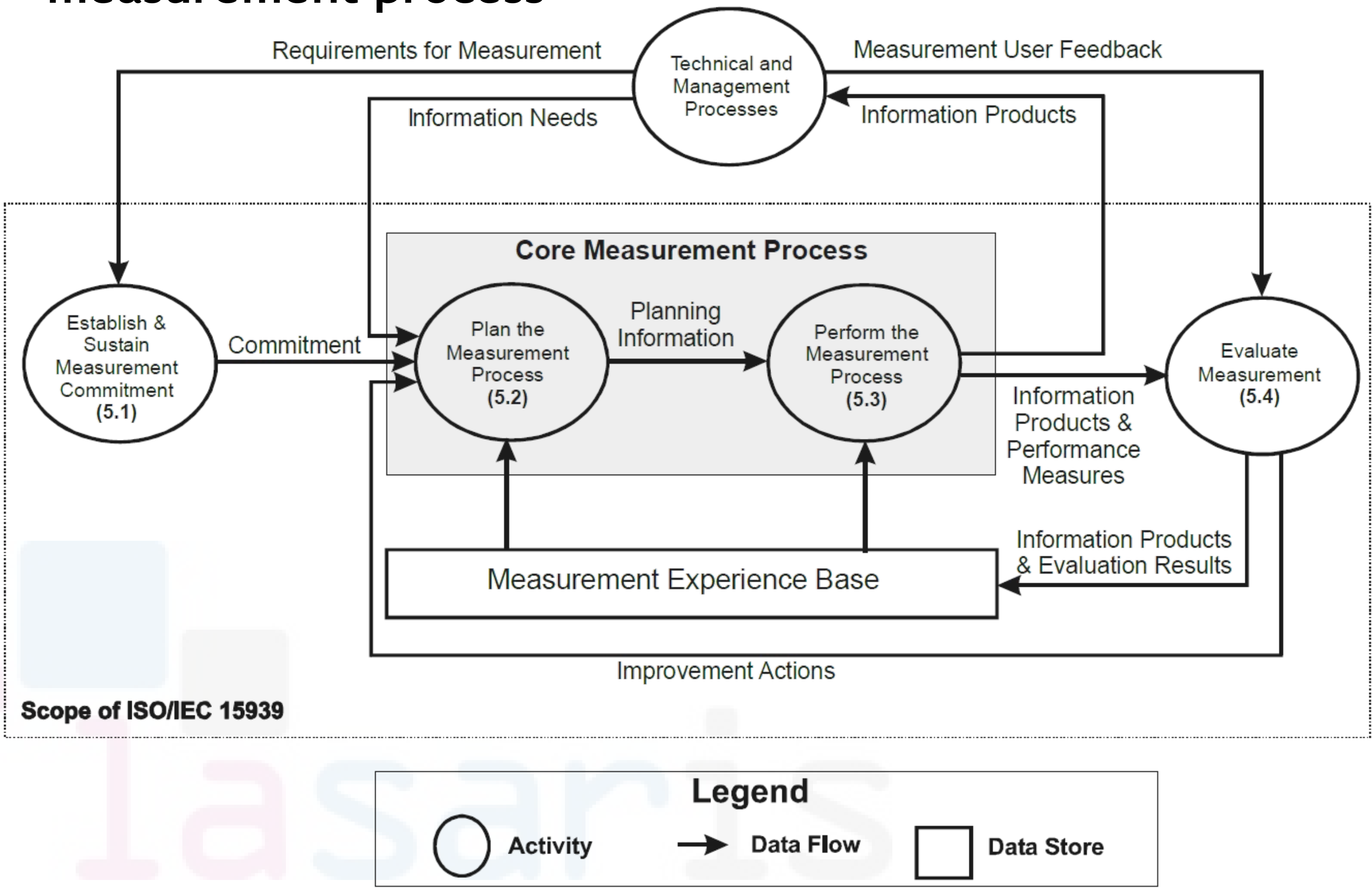


SQFD = Software Quality Function Deployment

Diagram Source: Becker and Associates, 2000

ISO/IEC 15939

- In lecture 2, we have seen the Measurement Information Model of ISO/IEC 15939:2002. To complete the view, now we discuss the **measurement process**



ISO/IEC 15939 - Example

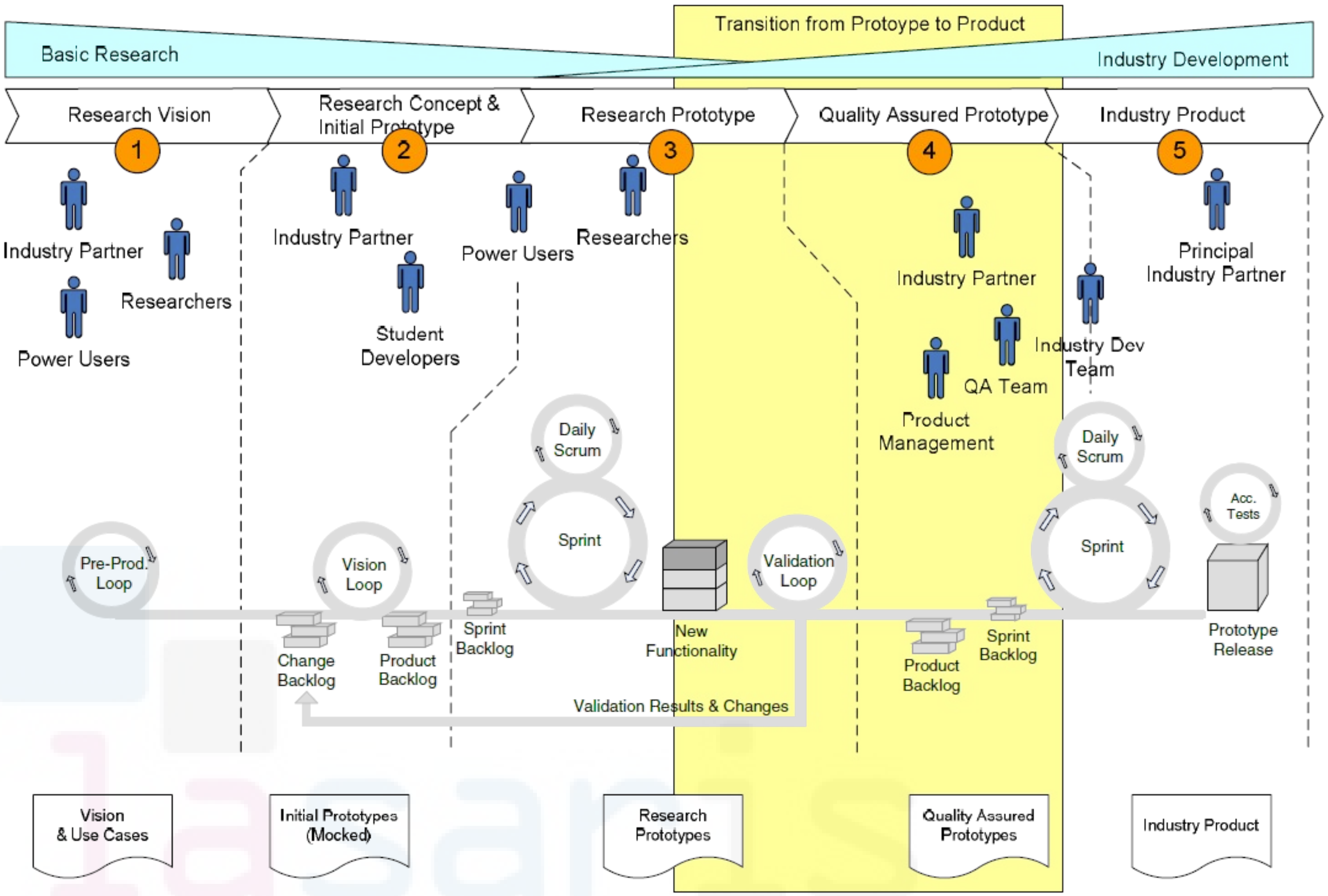
- Defining the measurement construct

Information Need	Estimate productivity of future project
Measurable Concept	Project productivity
Relevant Entities	1. Code produced by past projects 2. Effort expended by past projects
Attributes	1. C++ language statements (in code) 2. Timecard entries (recording effort)
Base Measures	1. Project X Lines of code 2. Project X Hours of effort
Measurement Method	1. Count semicolons in Project X code 2. Add timecard entries together for Project X
Type of Measurement Method	1. Objective 2. Objective
Scale	1. Integers from zero to infinity 2. Real numbers from zero to infinity
Type of Scale	1. Ratio 2. Ratio
Unit of Measurement	1. Line 2. Hour
Derived Measure	Project X Productivity
Measurement Function	Divide Project X Lines of Code by Project X Hours of Effort
Indicator	Average productivity
Model	Compute mean and standard deviation of all project productivity values
Decision Criteria	Computed confidence limits based on the standard deviation indicate the likelihood that an actual result close to the average productivity will be achieved. Very wide confidence limits suggest a potentially large departure and the need for contingency planning to deal with this outcome.

Final Considerations (1/2)

Customization of processes is usually quite heavy in the development world, it is difficult to find 1-1 correspondence with the original model – all derives from the project's quality objectives

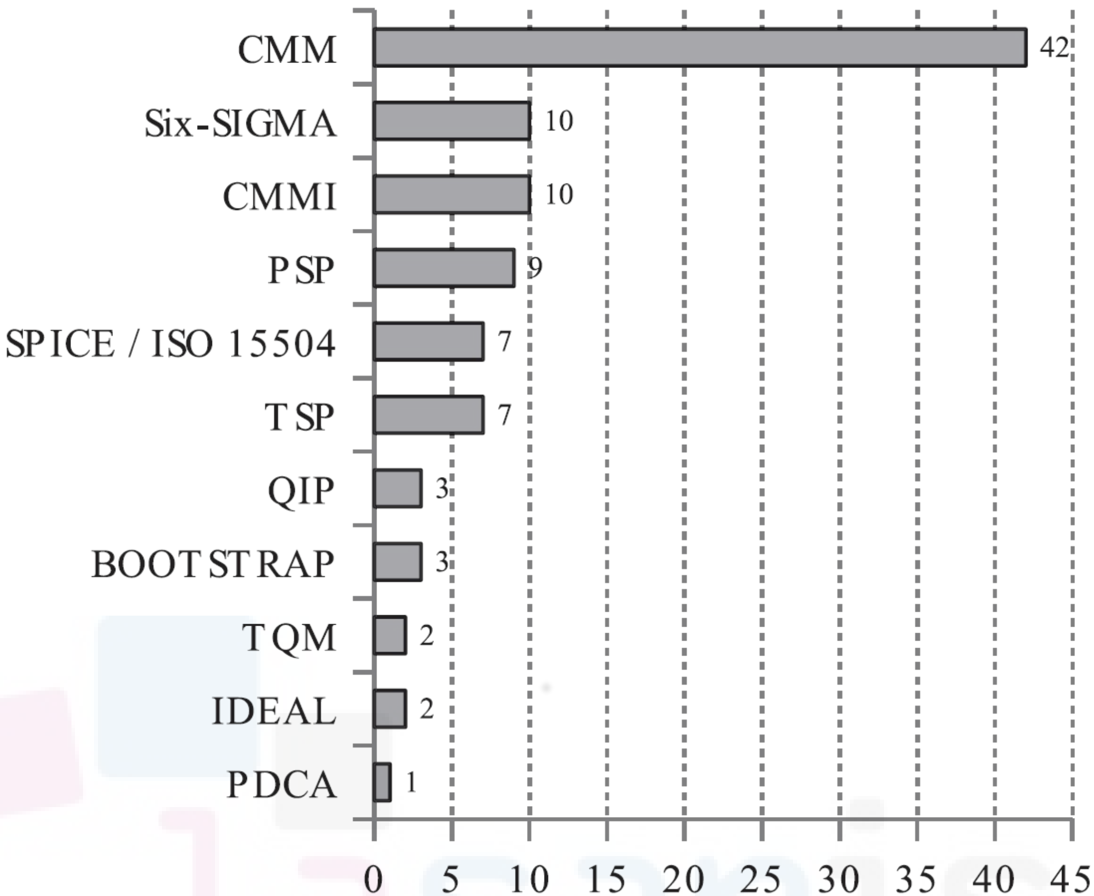
The following is an adaptation of SCRUM and **introduction of some maturity phases** to bridge the gap between research prototypes and industrial products releases



D. Winkler, R. Mordinyi, and S. Biffl, "Research Prototypes versus Products: Lessons Learned from Software Development Processes in Research Projects," in Systems, Software and Services Process Improvement, F. McCaffery, R. V. O'Connor, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2013, pp. 48–59.

Final Considerations (2/2)

The overall interest of researchers in Software Process Improvement area for the established techniques (that is not considering mixture and ad-hoc created methods)



M. Unterkalmsteiner, T. Gorschek, A. K. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt, “Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review,” IEEE Transactions on Software Engineering, vol. 38, no. 2, pp. 398–424, Mar. 2012.

Do better processes lead to a better software product?

Relation of Process Quality to Product Quality

- The problem was studied in * by means of a systematic literature review
- The starting point is that models such as CMMI were born under the assumption that *"the* ***quality of a system or product*** *is highly* ***influenced*** *by the* ***quality of the process*** *used to develop and maintain it"*

* Lavallée, M., & Robillard, P. N. (2011, September). Do software process improvements lead to ISO 9126 architectural quality factor improvement. In Proceedings of the 8th international workshop on Software quality (pp. 11-17). ACM.

Relation of Process Quality to Product Quality

- Projects that use some SPI initiatives show **higher levels of software maintainability**
- **It takes** significant time to see the benefits **of an SPI program**
- **Most of the SPI initiatives focus on defect reduction, not on other quality concerns**

- The major conclusion is that **Software Process Improvement** activities **always increase** process quality, but **not always product quality**

* Lavallée, M., & Robillard, P. N. (2011, September). Do software process improvements lead to ISO 9126 architectural quality factor improvement. In Proceedings of the 8th international workshop on Software quality (pp. 11-17). ACM.

References

For the part about increasing returns, path dependency, if you are interested :)

[1] Arthur, W. Brian (1989). *Competing Technologies, Increasing Returns, and Lock-In by Historical Events*, 97 Economic Journal 642-65.

[2] Farrell, Joseph and Garth Saloner (1985). *Standardization, Compatibility, And Innovation*, 16 Rand Journal 70-83.

[3] Katz, M. L., & Shapiro,C. (1985). *Network Externalities, Competition, and Compatibility*. The American Economic Review, 75(3), 424-440.

[4] Katz, M. L., & Shapiro, C. (1986). *Technology Adoption in the Presence of Network Externalities*. Journal of Political Economy 822-841.

[5] Liebowitz, S. J. and Stephen E. Margolis (1990). *The Fable of the Keys*, Journal of Law and Economics, 33:1, 1-26.

[6] Liebowitz, S. J. and Stephen E. Margolis (1994). *Path Dependency, Lock-In, and History*, working paper, 1994b.

[7] Liebowitz, S. J. and Stephen E. Margolis (1994). *Network Externality: An Uncommon Tragedy*, 8 Journal of Economic Perspectives 133-50.

[8] Economides, N. (1996). *The Economics of Networks*, International Journal of Industrial Organization, vol. 14, no. 6, pp. 673-699

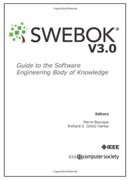
[9] Paul A. David (2000), *Path dependence, its critics and the quest for 'historical economics*, in P. Garrouste and S. Ioannides (eds), Evolution and Path Dependence in Economic Ideas: Past and Present, Edward Elgar Publishing, Cheltenham, England.

[10] Shapiro, C. e Varian, H.R. (1999). *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press.

[11] Windrum, P., (2003). *Unlocking a lock-in: towards a model of technological succession*, in Applied Evolutionary Economics: New Empirical Methods and Simulation Techniques, P.P. Saviotti (ed.), Cheltenham: Edward Elgar.

References

References mentioned in the slides, plus:



- Bourque, P., & Fairley, R. E. (2014). Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.