

PV260 - SOFTWARE QUALITY

SOFTWARE QUALITY MANAGEMENT PROCESS

Bruno Rossi

brossi@mail.muni.cz

LAB OF SOFTWARE ARCHITECTURES
AND INFORMATION SYSTEMS

FACULTY OF INFORMATICS
MASARYK UNIVERSITY, BRNO



*"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

Introduction

4 Views of Quality



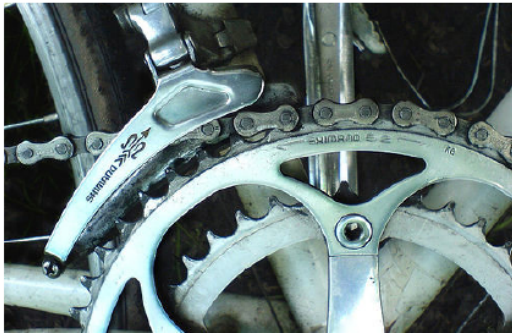
Quality in Use

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

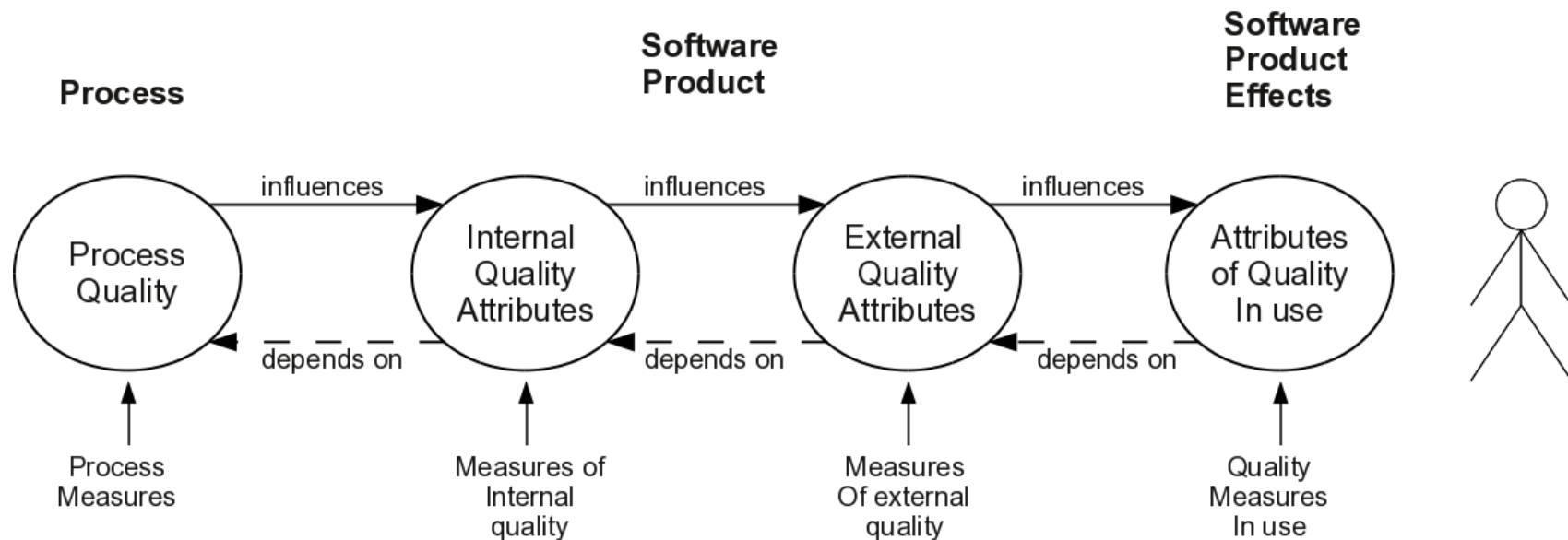
(Is it assembled correctly?)

We deal with process quality in this lecture

lasaris

Introduction

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



lasaris

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



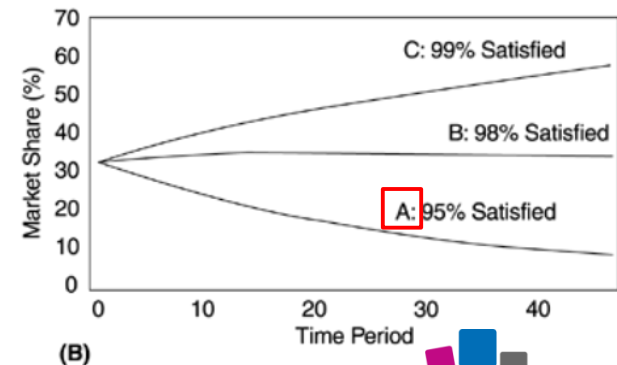
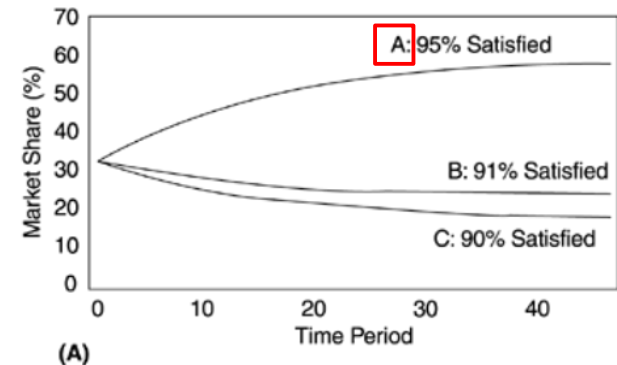
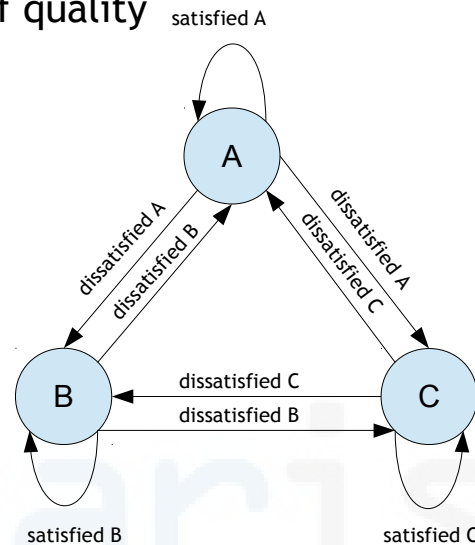
The impact of Quality

How much “Quality” is needed? If we could quantify it, should/could we have 100% quality in our software products?

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

It was 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



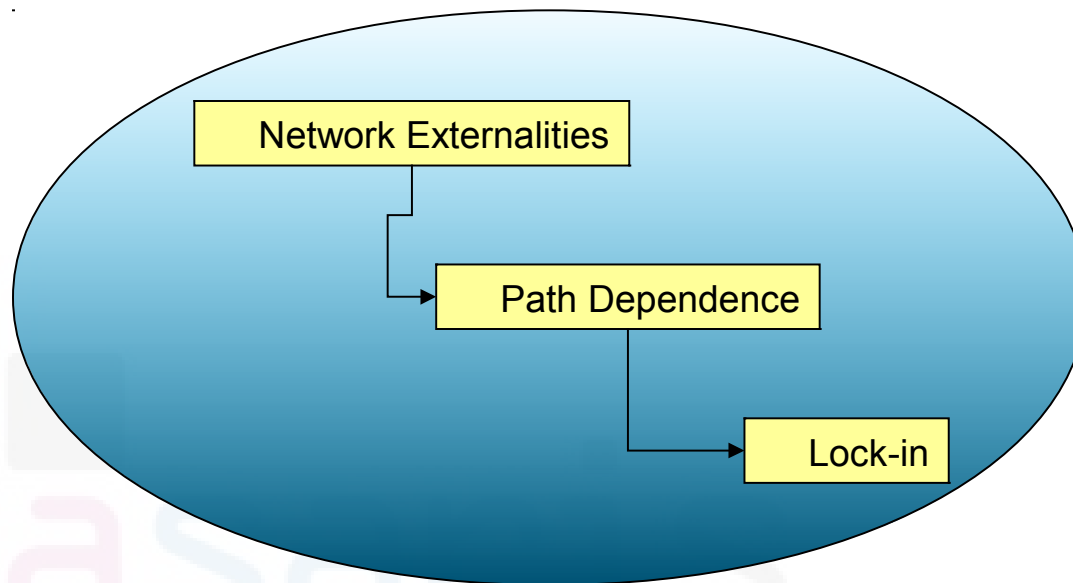
The impact of Quality

- 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)

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*



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

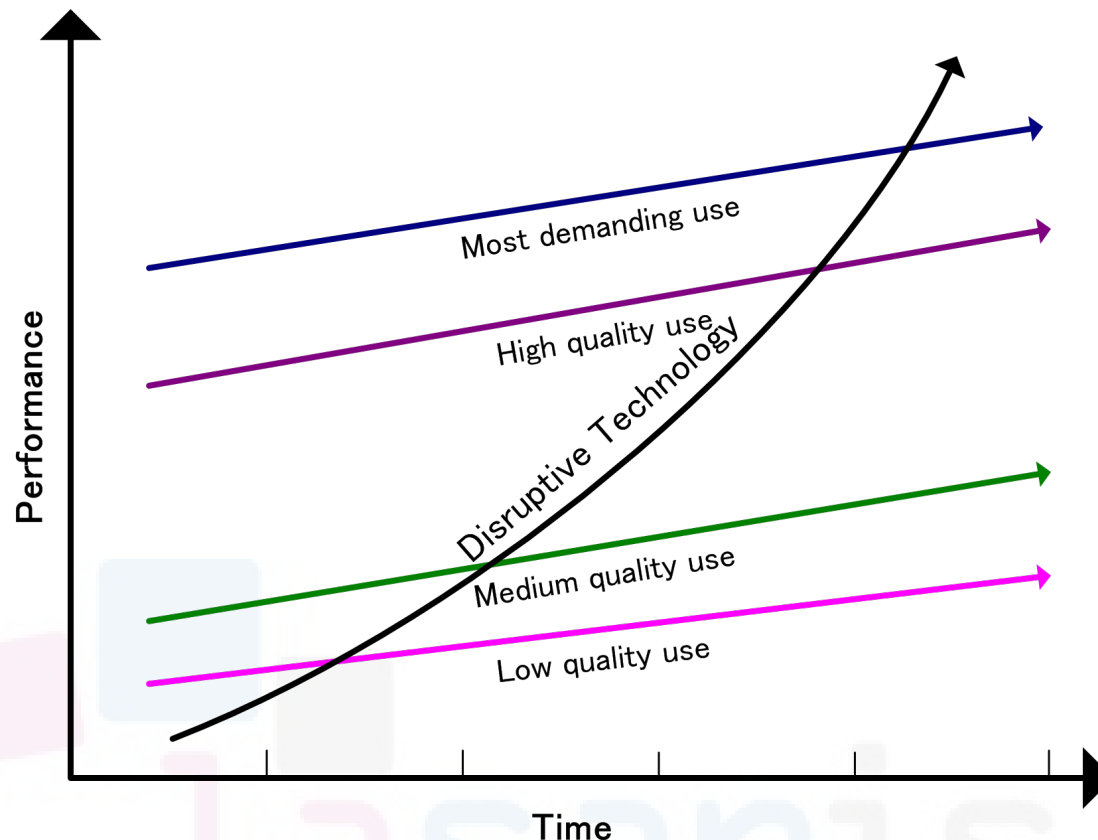


Sustaining and Disruptive Innovations (1/3)

- 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 and **constitutes a paradigm shift** (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?*

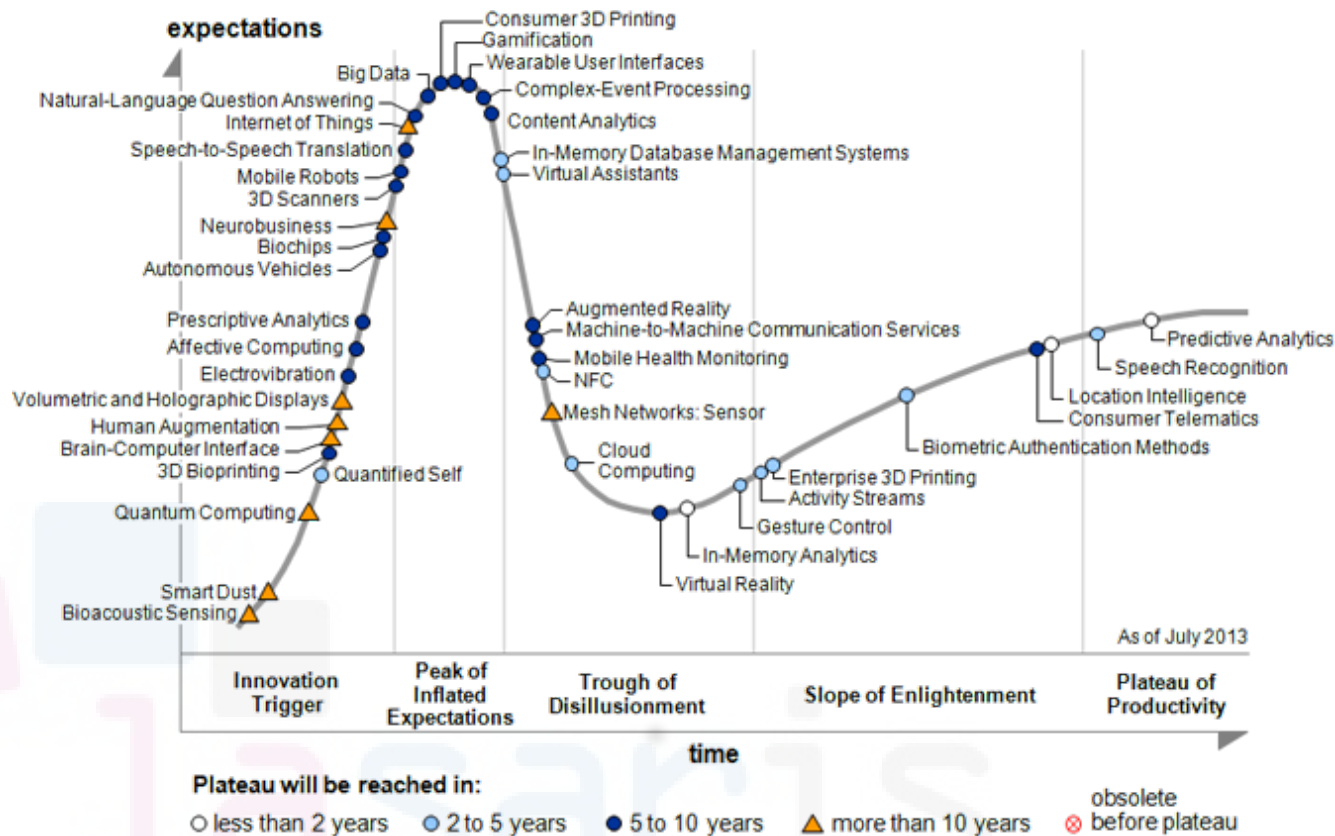
Sustaining and Disruptive Innovations (2/3)

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



Sustaining and Disruptive Innovations (3/3)

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



← 2013 GHC

Do you see any technology that maintained the “hype”?

Q&A



General Questions

- a) Can you name **one disruptive innovation** either in Software or in the IT world that impressed you particularly in the last few years?
- b) Thinking in terms of **5-10 years from now**, what do you think will be a **disruptive technology** in the software/IT world?

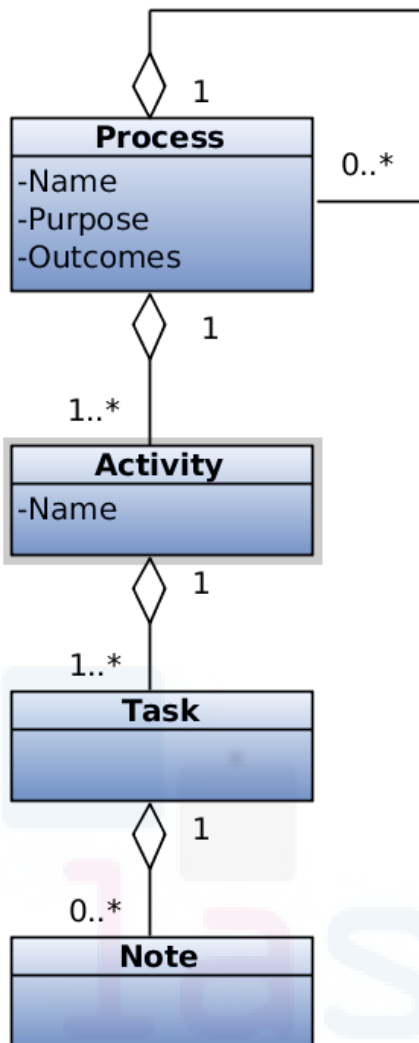


SQM Categories



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

SQM comprises four subcategories

A → All Planning for SWQ

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 → Assess the adequacy of software processes

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

C → Compliance to standards established for the project for products

D → Activities to improve software processes

SQA vs SQC - What's the difference?

Criteria	Software Quality Assurance (SQA)	Software Quality Control (SQC)
<i>Definition</i>	SQA is a set of activities for ensuring quality in software engineering processes (that ultimately result in quality in software products). The activities establish and evaluate the processes that produce products.	SQC is a set of activities for ensuring quality in software products. The activities focus on identifying defects in the actual products produced.
<i>Focus</i>	Process focused	Product focused
<i>Orientation</i>	Prevention oriented	Detection oriented
<i>Breadth</i>	Organization wide	Product/project specific
<i>Scope</i>	Relates to all products that will ever be created by a process	Relates to specific product
<i>Activities</i>	<ul style="list-style-type: none"> • Process Definition and Implementation • Audits • Training 	<ul style="list-style-type: none"> • Reviews • Testing

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 (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)

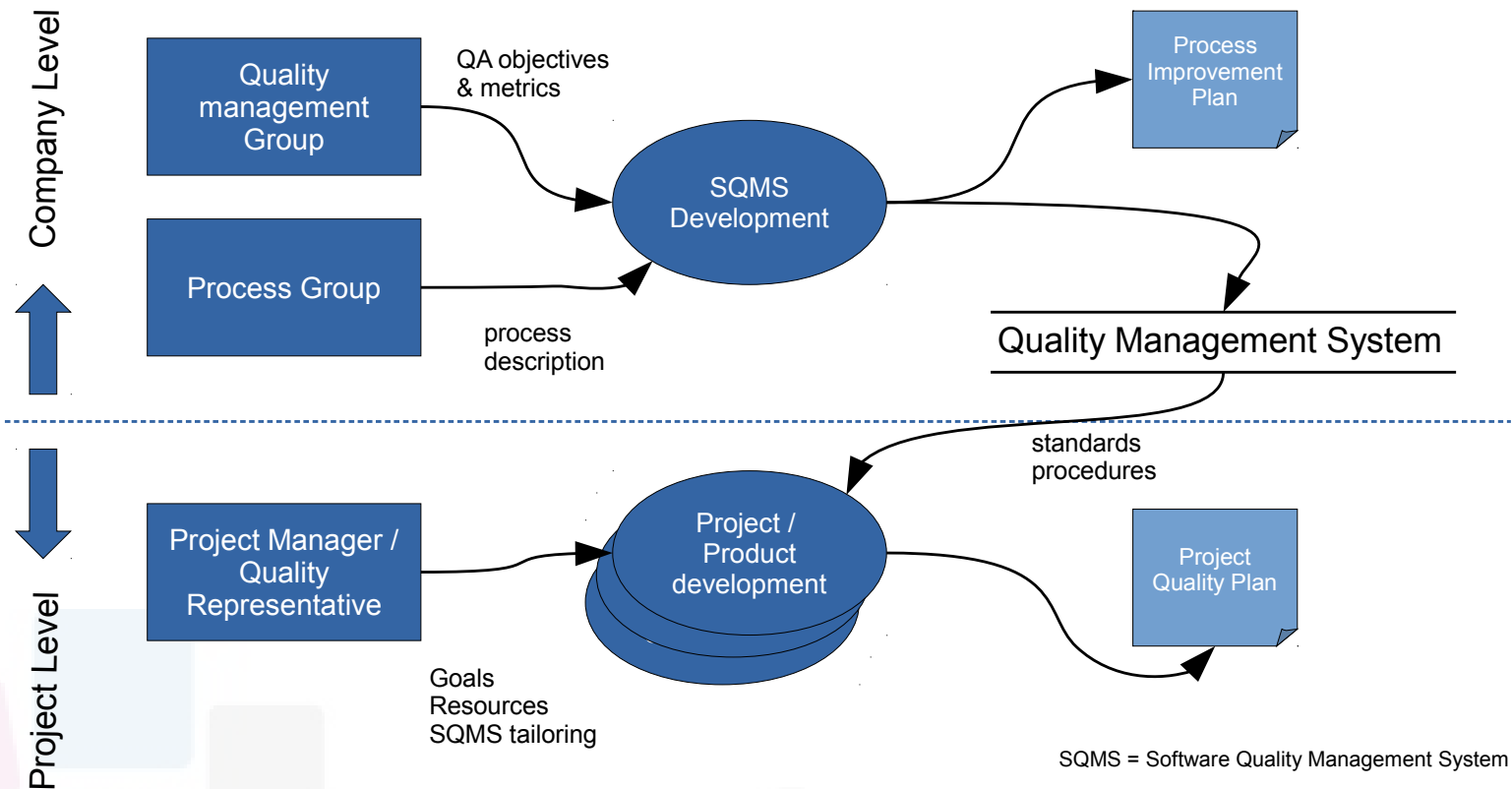
D. Software process improvement (SPI)

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

lasaris

A. Software Quality Planning

- Quality at the organization and project Levels



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

B. Software Quality Assurance 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

B. Summary of IEEE Std 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/product to **check for conformance to applied standards** (e.g. CMMI) or produced artifacts
- Some examples of methods
 - The Goal Question Metrics Approach ([seen on the measurement lecture](#))
 - The Plan-Do-Check-Act method

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

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)
 - Personal Software Process (PSP) and Team Software Process (TSP)

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

Some Historical Models for Software Quality Management



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 “



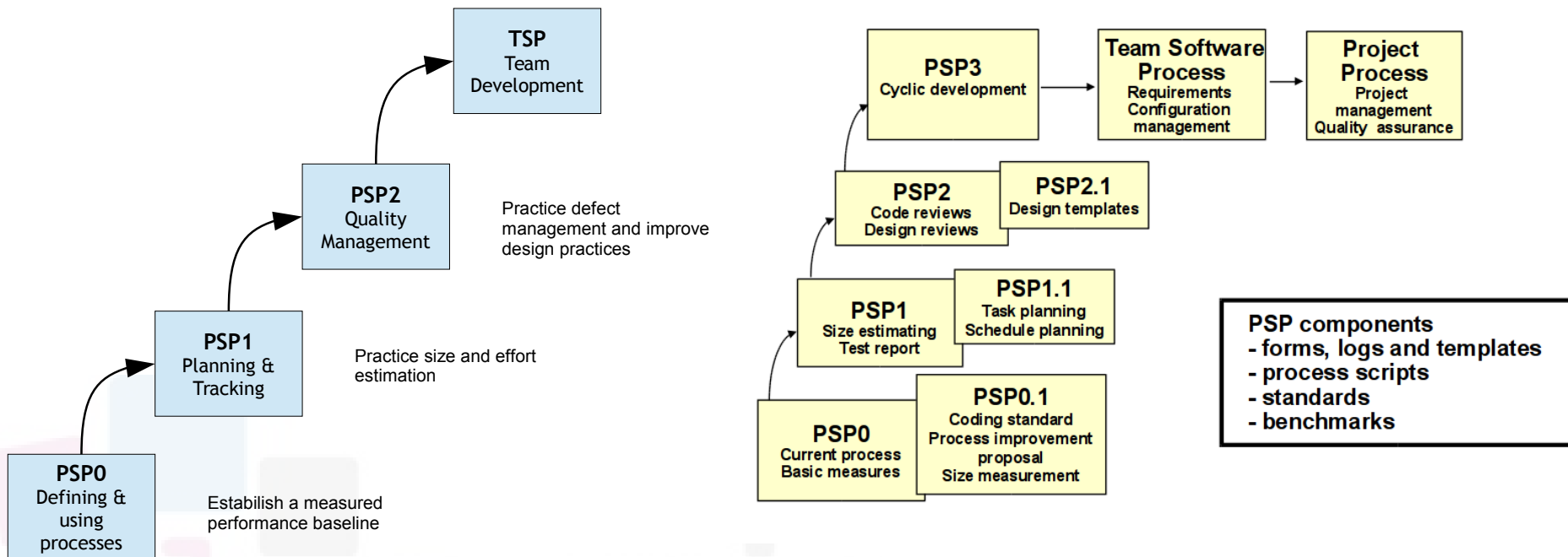
Image: <https://www.cs.utexas.edu/blog/cowboy-rides-away-now>

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

Personal Software Process (PSP)

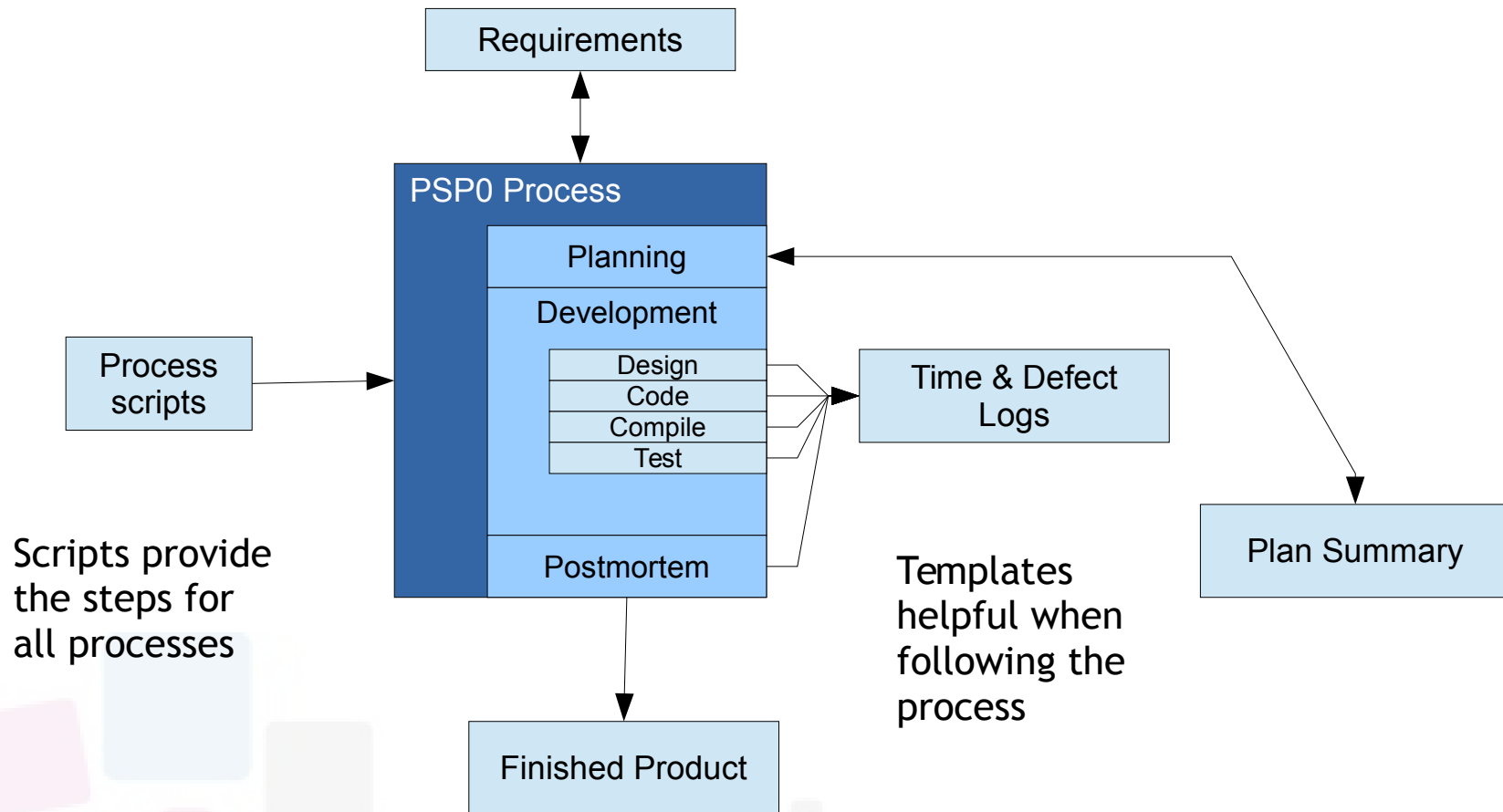
- The Personal Software Process (PSP) is a **disciplined software development process** that works at the **individual level**

Defined around 1995: Humphrey, Watts S. A discipline for software engineering. Addison-Wesley Longman Publishing Co., Inc., 1995.



lasaris

PSP0 - First Level



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**

lasaris

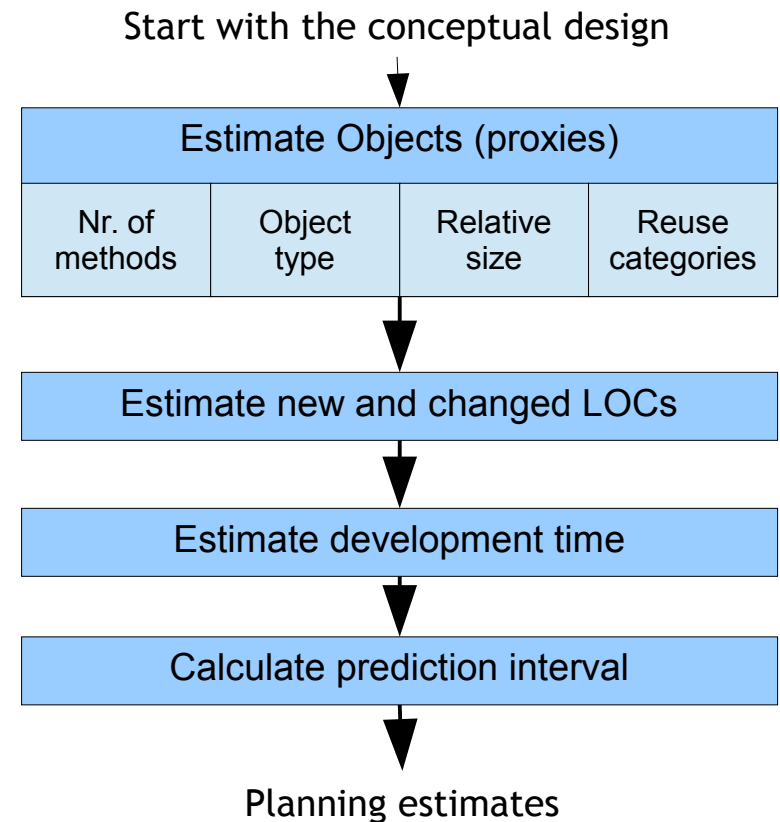
PSP0.1 - Improvement

- 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)		<i>(Measured)</i>	87	
Deleted (D)		<i>(Counted)</i>	0	
Modified (M)		<i>(Counted)</i>	6	
Added (A)		<i>(T-B+D-R)</i>	113	
Reused (R)		<i>(Counted)</i>	0	
Total New & Changed (N)	<u>90</u>	<i>(A+M)</i>	119	315
Total LOC (T)		<i>(Measured)</i>	200	396
Total New Reuse			0	
Time in Phase (min.)	Plan	Actual	To Date	To Date %
Planning	10	11	36	6.2
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	0.0
Design		1	3	11.5
Code		8	23	88.5
Compile		0	0	0.0
Test		0	0	0.0
Total Development		9	26	100.0
Defects Removed		Actual	To Date	To Date %
Planning		0	0	0.0
Design		0	0	0.0
Code		0	0	0.0
Compile		2	13	50.0
Test		7	13	50.0
Total Development		9	26	100.0
After Development		0	0	0.0

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)



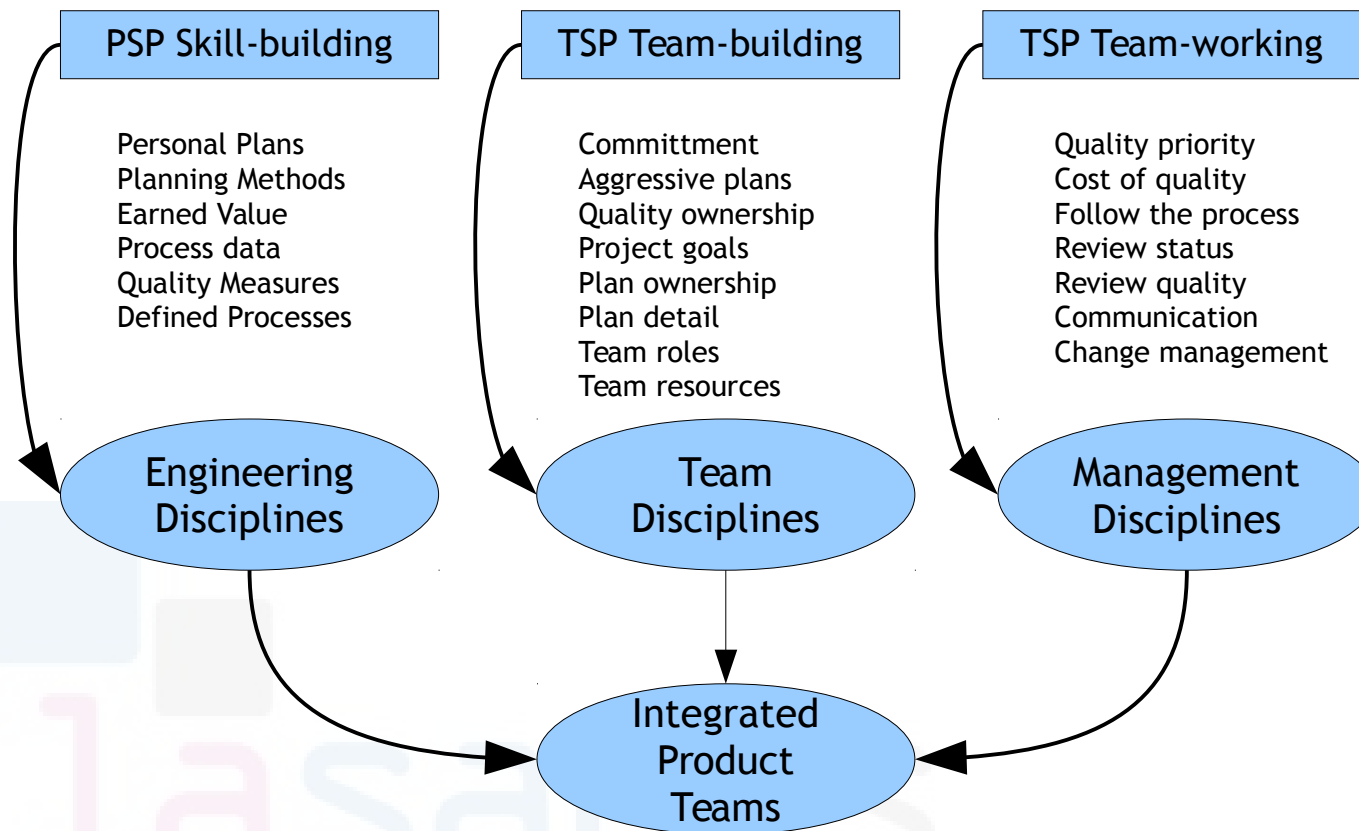
PSP2 - Third Level

- 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



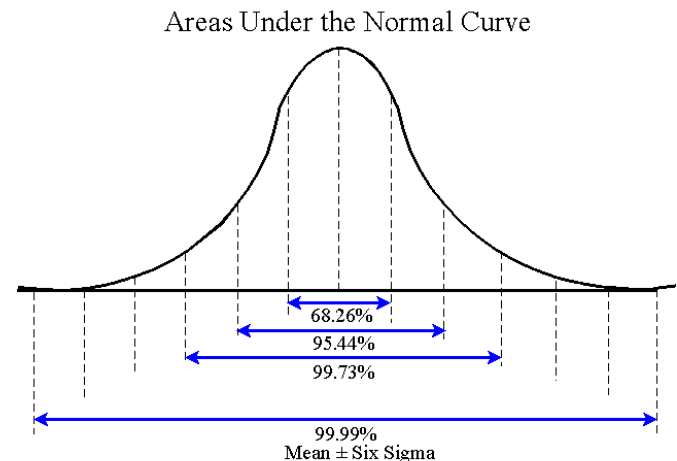
Six Sigma

Defined in the '80s in industry but only in 90's adopted widely, later on adopted by software engineering community.

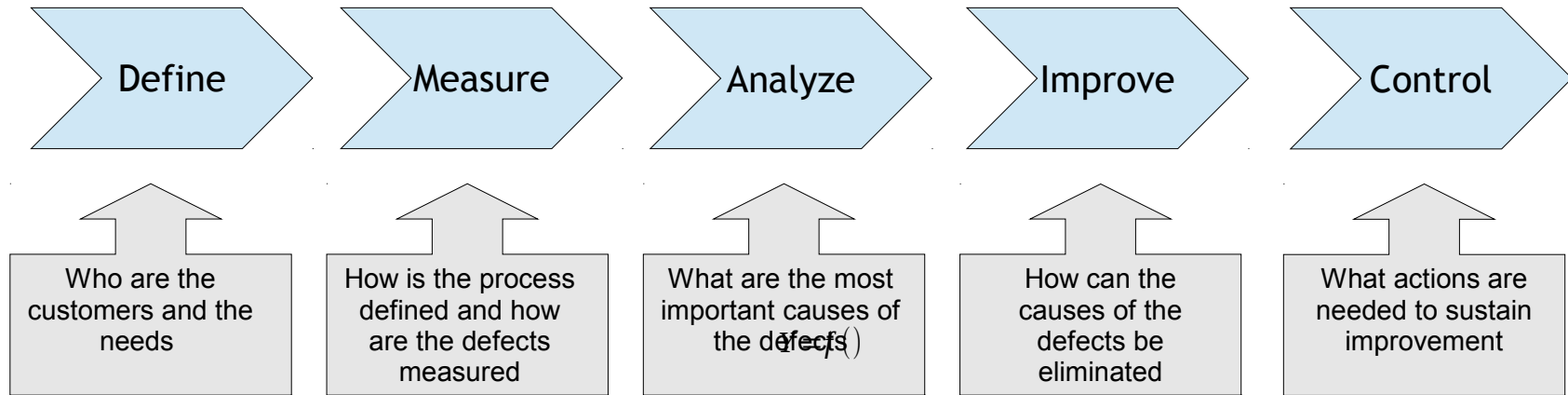
- A measure of Software Quality developed at Motorola
- The focus of Six Sigma is on **eliminating defects**, that is everything that is **outside from the customers specifications**
- **No more than +/- six times the standard deviation from the process mean** → 3,4 Defects Per Million datapoints
- Data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

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})}{n-1}}$$



Six Sigma - Process

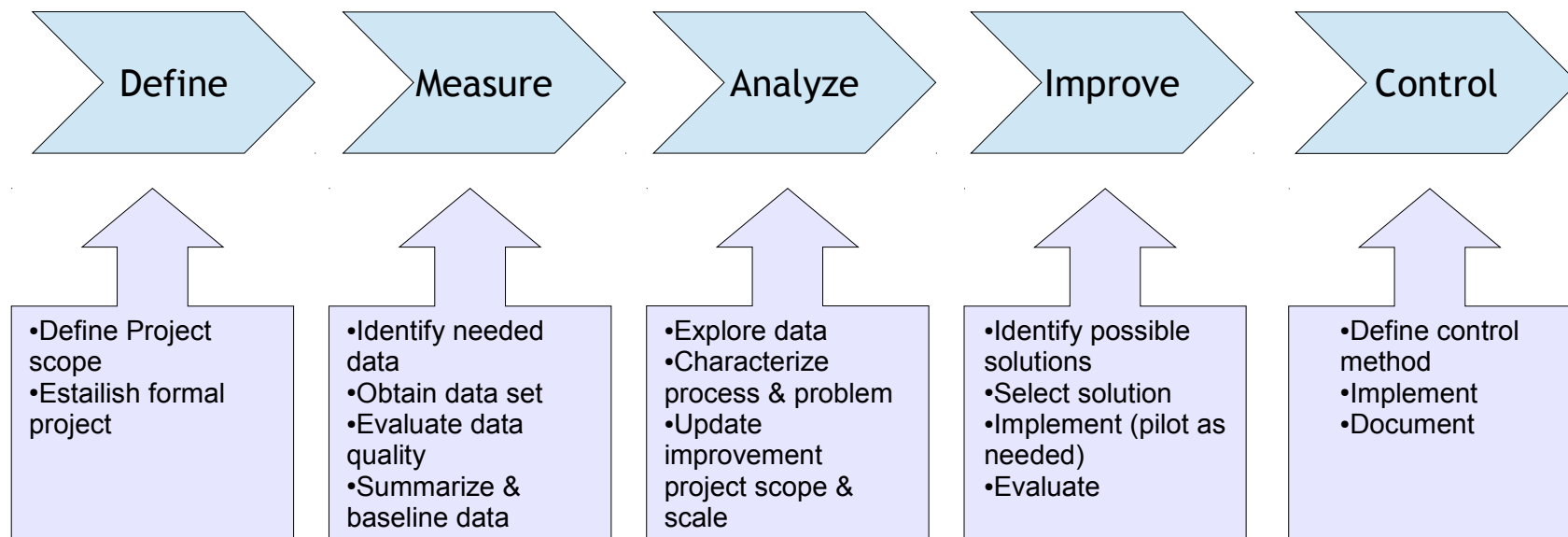


In Six Sigma, a defect is defined as *“Any product, service, or process variation which prevents meeting the needs of the customer and/or which adds cost, whether or not it is detected”*

Key aspect: understand the relation between depend variables (Y, defects) and independent variables (X, causes)

$$Y = f(x_1, x_2, x_3, \dots, x_n)$$

Six Sigma - Process



lasaris

Six Sigma - Tools

Define	Measure	Analyze	Improve	Control
<ul style="list-style-type: none"> Benchmark Contract/charter Kano Model Voice of the customer Voice of the Business Quality Function Deployment (QFD) 	<ul style="list-style-type: none"> GQ(I)M and indicator templates Data collection methods Measurement System Evaluation 	<ul style="list-style-type: none"> Cause & Effect Diagram/Matrix Failure models & effects analysis Statistical inference Reliability Analysis Root Cause Analysis Hypothesis Test 	<ul style="list-style-type: none"> Design of experiments Modeling ANOVA Tolerancing Robust Design System Thinking Decision & Risk Analysis Performance Analysis Model 	<ul style="list-style-type: none"> 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

lasaris

Software Process Maturity Models



Process Maturity Levels

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

Capability Level	Process Capability
0	Incomplete Process
1	Performed Process
2	Managed Process
3	Established Process
4	Predictable Process
5	Optimizing Process

→ The process is not implemented or fails to achieve the purpose

→ The process is achieving the purpose

→ The process is now running in a managed way (planned, monitored, adjusted) – work products are established, controlled and maintained

→ The managed process is now implemented using a defined process capable of achieving process outcomes

→ The established process now operates within defined limits to achieve its process outcomes

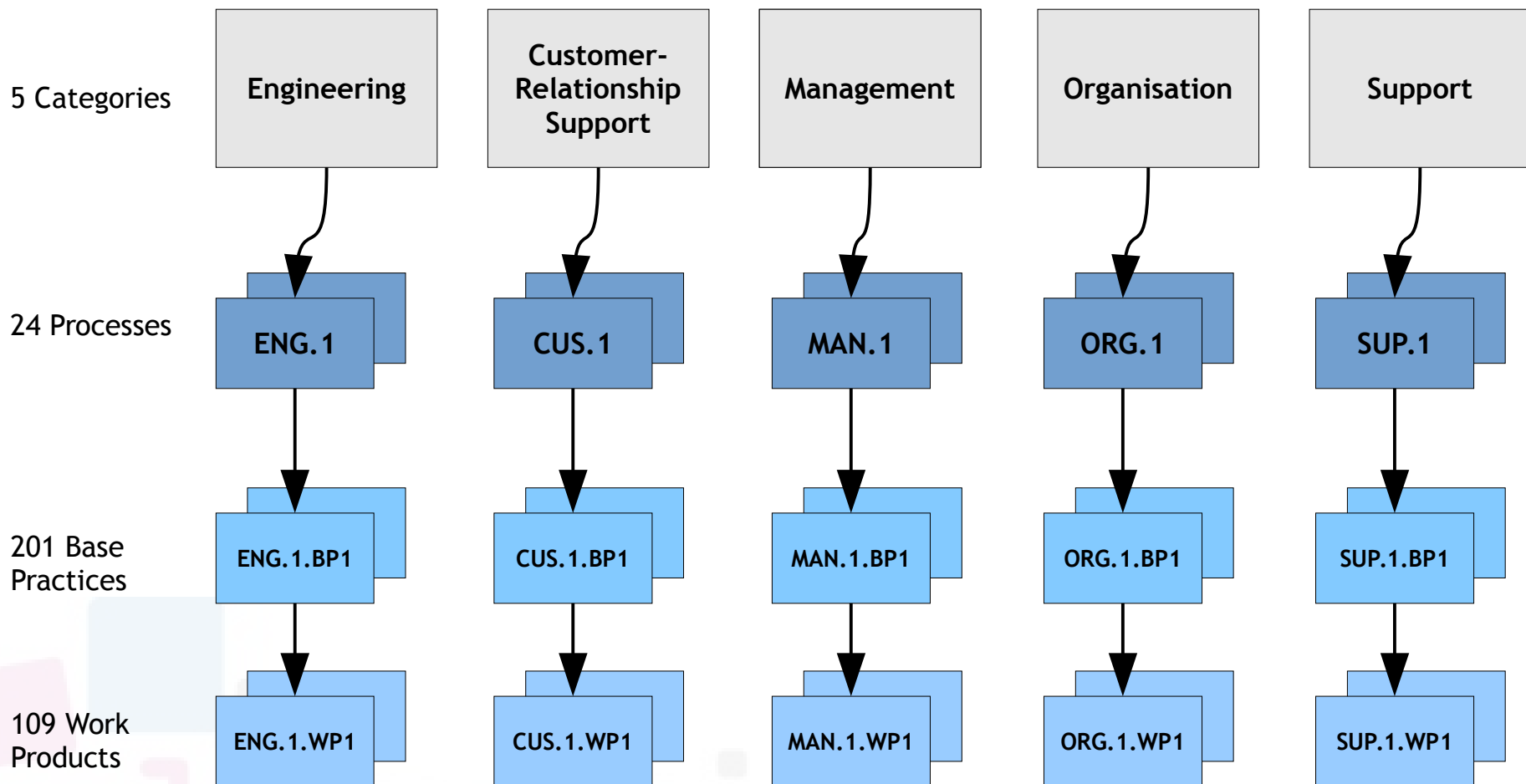
→ The predictable process is continuously improved to meet relevant current and projected business goals

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	<u>Process performance</u>
	Level 2: Managed process
PA 2.1	<u>Performance management</u>
PA 2.2	<u>Work product management</u>
	Level 3: Established process
PA 3.1	<u>Process definition</u>
PA 3.2	<u>Process deployment</u>
	Level 4: Predictable process
Process Attribute ID	Capability Levels and Process Attributes
PA 4.1	<u>Process measurement</u>
PA 4.2	<u>Process control</u>
	Level 5: Optimizing process
PA 5.1	<u>Process innovation</u>
PA 5.2	<u>Continuous optimization</u>

SPICE - Overall



lasaris

Example Process Definition (1/2)

Process ID	SUP.1	Process ID	SUP.1
<u>Process Name</u>	Quality assurance		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]
<u>Process Purpose</u>	The purpose of the Quality assurance process is to provide independent assurance that work products and processes comply with predefined provisions and plans.		
<u>Process Outcomes</u>	<p>As a result of successful implementation of this process:</p> <ol style="list-style-type: none"> 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. <p>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.</p> <p>NOTE 2: Verification and validation may be subject to quality assurance.</p> <p>NOTE 3: Independent quality assurance should be established as a separate functional role within an organization.</p>	5 Process Categories (SUP=Support)	<p>SUP.1.BP3: Develop and implement a plan for project quality assurance based on a quality assurance strategy. [Outcome 3]</p> <p>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.</p> <p>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]</p> <p>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]</p> <p>NOTE 3: Product quality assurance activities may include reviews, audits, problem analysis, reports and lessons learned that improve the work products for further use.</p> <p>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.</p> <p>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]</p> <p>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.</p> <p>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.</p>
<u>Base Practices</u>	<p>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]</p> <p>NOTE 1: The quality assurance process may be co-ordinated with the related SUP.2 Verification, SUP.4 Joint Review, Validation and Audit processes.</p>		

Example Process Definition (2/2)

Process ID	SUP.1
	<p>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.</p> <p>SUP.1.BP7: Track and record quality assurance activities. Records of quality assurance activities are produced and retained. [Outcome 3, 4, 5]</p> <p>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]</p> <p>NOTE 8: The quality assurance may use an independent path to report regularly the results to the management and other relevant stakeholders.</p> <p>SUP.1.BP9: Ensure resolution on non-conformances. Deviations or non-conformance found in process and product quality assurance activities should be analyzed, corrected and further prevented. [Outcome 5]</p> <p>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]</p>

Overall 24 processes
are specified

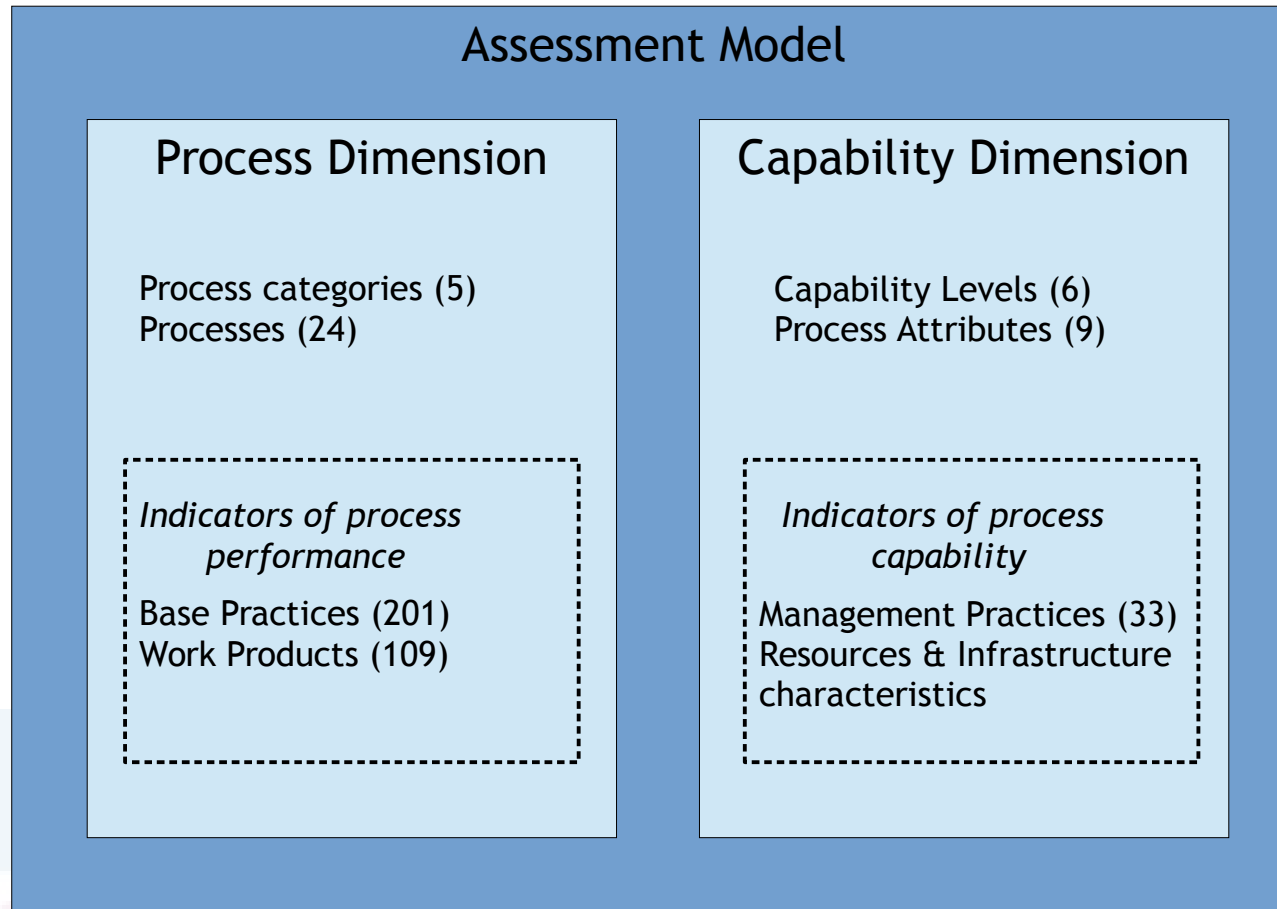
<u>Output Work Products</u>	
08-13	Quality plan [Outcome 3, 5, 6]
13-04	Communication Record [Outcome 5]
13-07	Problem record [Outcome 3, 4]
13-18	Quality Record [Outcome 2, 3, 4]
13-19	Review Record [Outcome 2, 3, 4]
14-02	Corrective action plan [Outcome 3, 5]
18-07	Quality criteria [Outcome 4]

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, management practices → How well is a specific goal met?

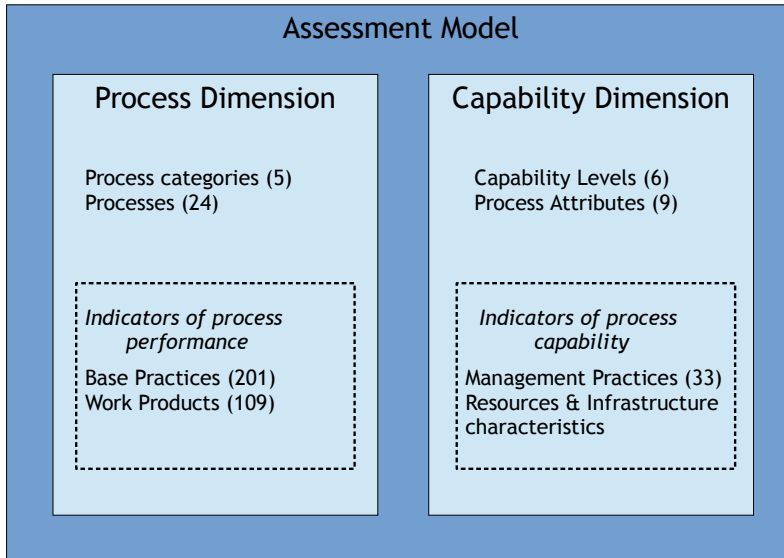


Process Maturity Levels & Attributes



lasaris

Process Maturity Levels & Attributes



Assessment Scale

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

Process Attributes

- 1.1 Process performance
- 2.1 Performance management
- 2.2 Work product management
- 3.1 Process definition
- 3.2 Process deployment
- 4.1 Process measurement
- 4.2 Process control
- 5.1 Process innovation
- 5.2 Process optimization

	PA1.1	PA2.1	PA2.2	PA3.1	PA3.2	PA4.1	PA4.2	PA5.1	PA5.2
ENG. 1	█	█					█	█	
ENG. 2	█	█					█	█	
MAN. 2						█	█	█	
ORG. 1		█	█	█	█				

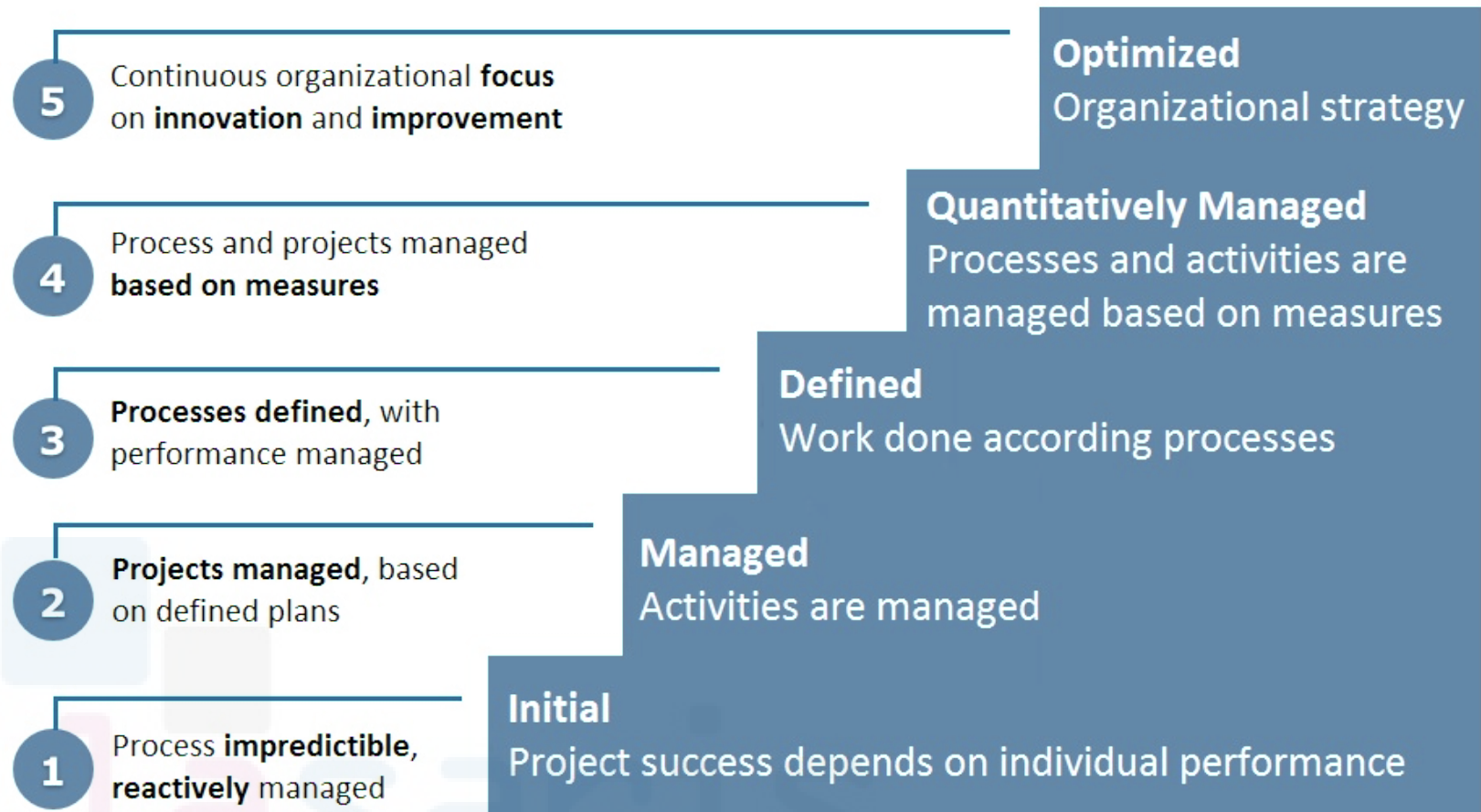
N	□
P	█
L	█
F	█

→ who performs the assessment?

CMMI

Capability Maturity Model Integrated

Upgrade from CMM appearing around year 2000



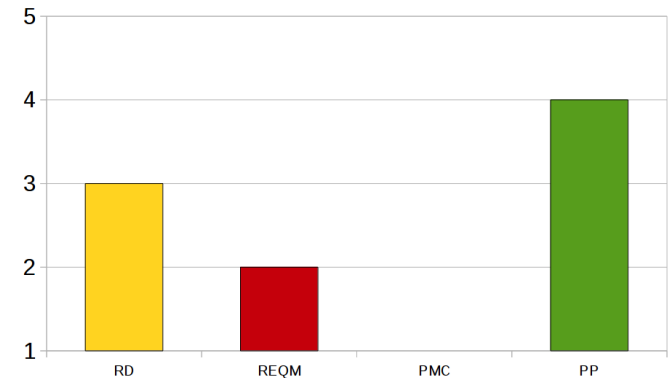
CMMI

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

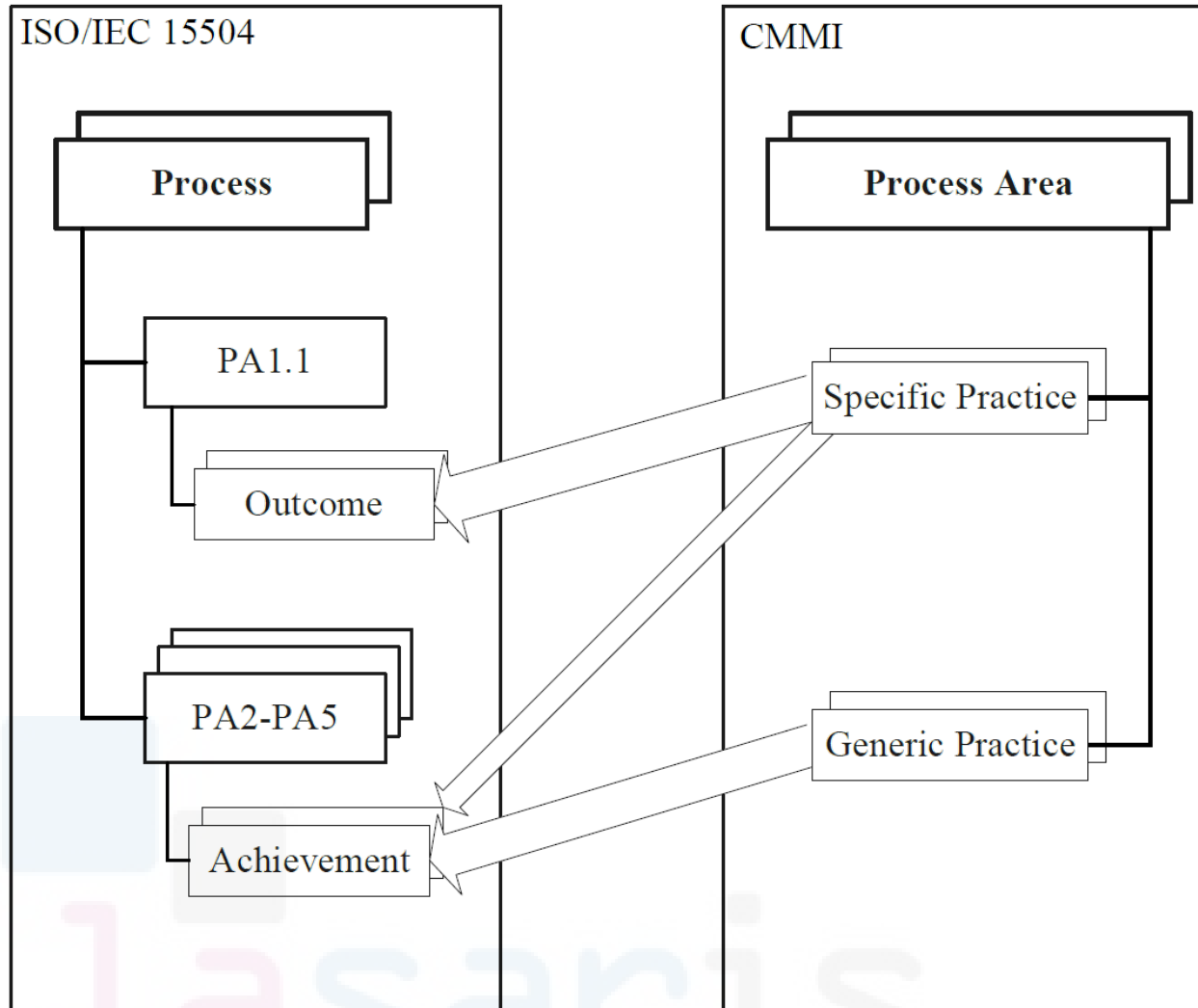
Continuous → assessment of the **capabilities of different Process Areas (PA)**

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 (SPICE)



Mapping CMMI to ISO/IEC 15504 (SPICE)

	ML2			ML3			ML4			ML5						
	CL1		CL2	CL1		CL2	CL3	CL1		CL2	CL3	CL1		CL2	CL3	
	N	P	L	F	N	P	L	F	N	P	L	F	N	P	L	F
ACQ.1 Acquisition preparation																
ACQ.2 Supplier selection																
ACQ.3 Contract agreement																
ACQ.4 Supplier monitoring																
ACQ.5 Customer acceptance																
SPL.1 Supplier tendering																
SPL.2 Product release																
SPL.3 Product acceptance support																
OPE.1 Operational use																
OPE.2 Customer support																
ENG.1 Requirements elicitation																
ENG.2 System requirements analysis																
ENG.3 System architectural design																
ENG.4 Software requirements analysis																
ENG.5 Software design																
ENG.6 Software construction																
.....																
SUP.1 Quality assurance																
SUP.2 Verification																
SUP.3 Validation																
SUP.4 Joint review																
SUP.5 Audit																
SUP.6 Product evaluation																
SUP.7 Documentation																
SUP.8 Configuration management																
SUP.9 Problem resolution management																
SUP.10 Change request management																

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

ISO/IEC 15939 - Example

- Defining the measurement construct

Information Need	Estimate productivity of future project
Measurable Concept	Project productivity
Relevant Entities	<ol style="list-style-type: none"> Code produced by past projects Effort expended by past projects
Attributes	<ol style="list-style-type: none"> C++ language statements (in code) Timecard entries (recording effort)
Base Measures	<ol style="list-style-type: none"> Project X Lines of code Project X Hours of effort
Measurement Method	<ol style="list-style-type: none"> Count semicolons in Project X code Add timecard entries together for Project X
Type of Measurement Method	<ol style="list-style-type: none"> Objective Objective
Scale	<ol style="list-style-type: none"> Integers from zero to infinity Real numbers from zero to infinity
Type of Scale	<ol style="list-style-type: none"> Ratio Ratio
Unit of Measurement	<ol style="list-style-type: none"> Line 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.

ISO/IEC 15939:2007 defines the measurement process for software systems engineering

→ was discussed in the lecture about software metrics & measurement

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
<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • 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 	<ul style="list-style-type: none"> • Measured stage • Management Level Maturity • Innovating • On Demand Just in Time Release • Optimise • Insane • Ambient • Scaling • Coexistence with non-agile • Enterprise transformation • Matured

Agile Process Maturity

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

Process ID	AMP.1	
Process name	<u>Backlog management</u>	
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

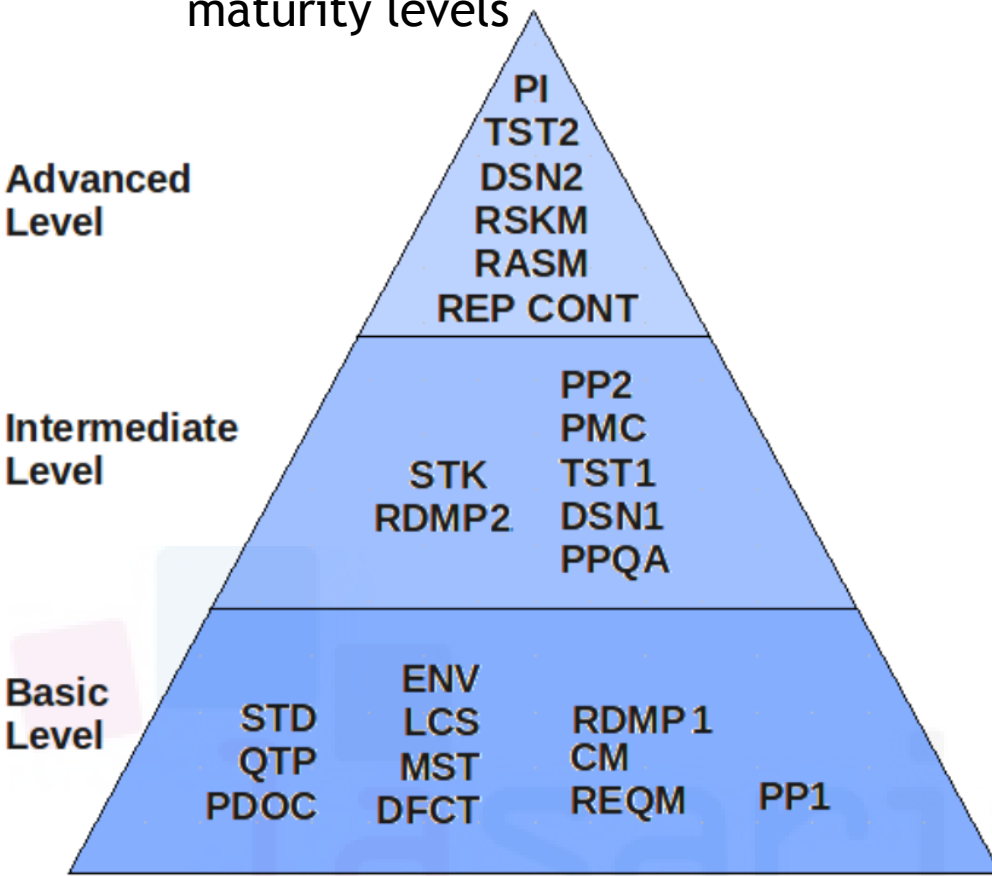
Agile Maturity Matrix

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



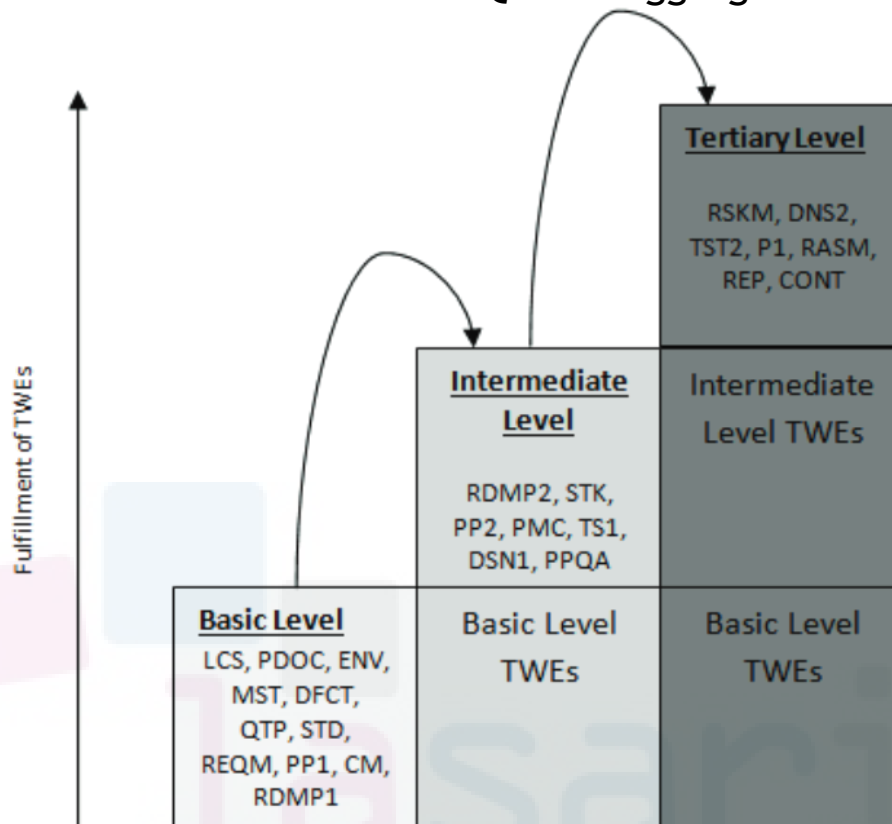
PDOC - Product Documentation
 STD - Use of Established and Widespread Standards
 QTP - Quality of Test Plan
 LCS - Licenses
 ENV - Technical Environment
 DFCT - Number of Commits and Bug Reports
 MST - Maintainability and Stability
 CM - Configuration Management
 PP1 - Project Planning Part 1
 REQM -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

- Each TWE is divided in goals (G) → practices (TWE) → checklists → metrics
- Example: PDOC (Product Documentation) → G1. Provide HQ documentation → P1. Create Development Documentation → C1. Is detailed architectural design documentation available? (yes/no) → C2.....
- 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:



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

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

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

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

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

P=practice, G=goal, ML=Maturity Level, R=rating

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.

