Quality Improvement Methods and Empirical Research in SE

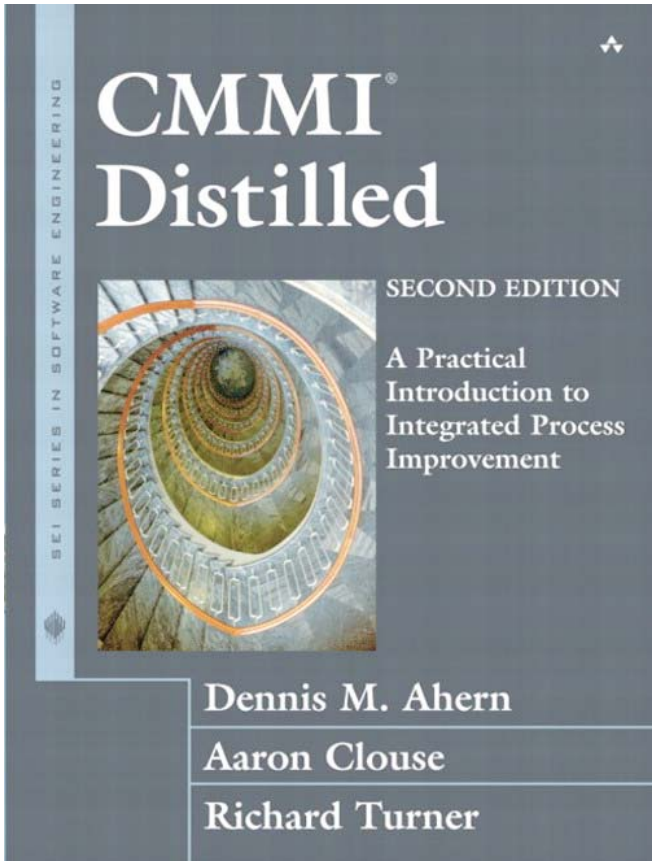
Michel Chaudron

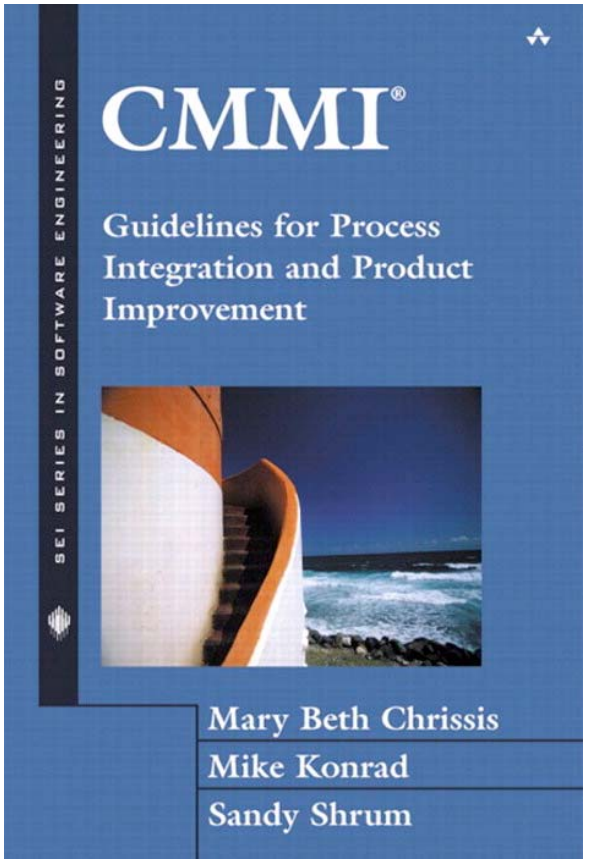
Leiden Institute of Advanced Computer Science

Agenda

- Quality Improvement
  - Software Process Improvement (CMMI)
  - Review and Inspection
  - Formal Methods
- Risk Management
- Empirical Research in Software Engineering
- Summary

CMM - Capability Maturity Model*

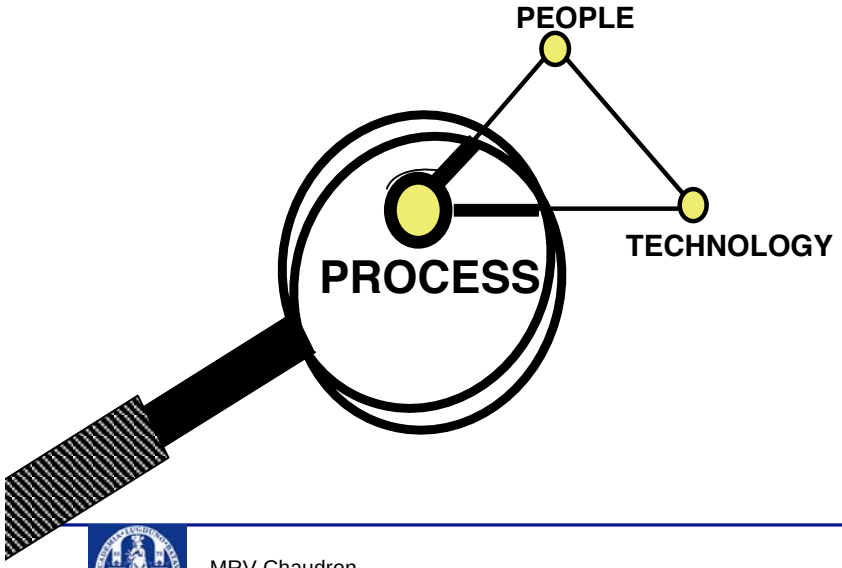




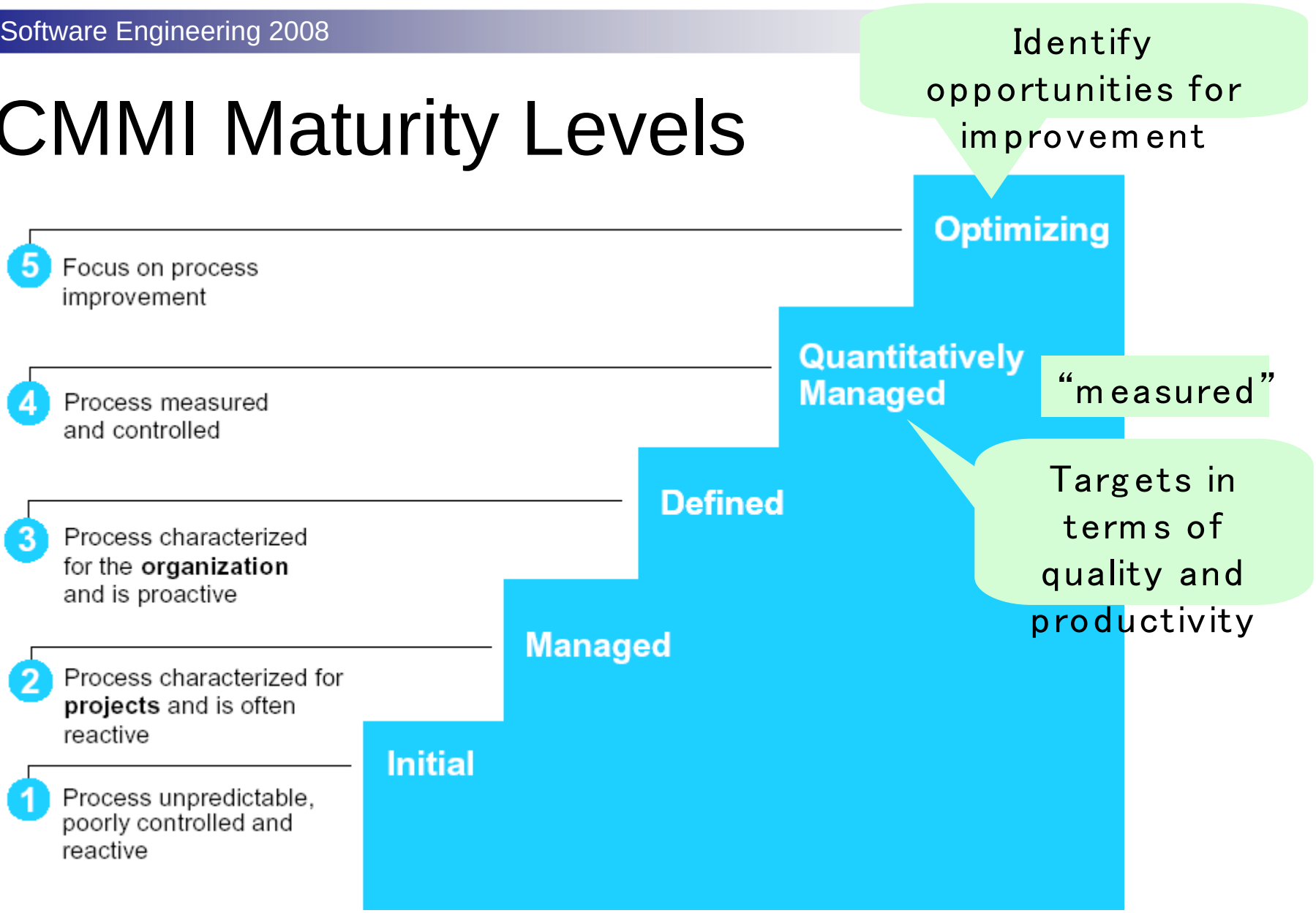
* a.k.a. Consultant Money Making Initiative

Premise of Software Process Improvement (SPI)

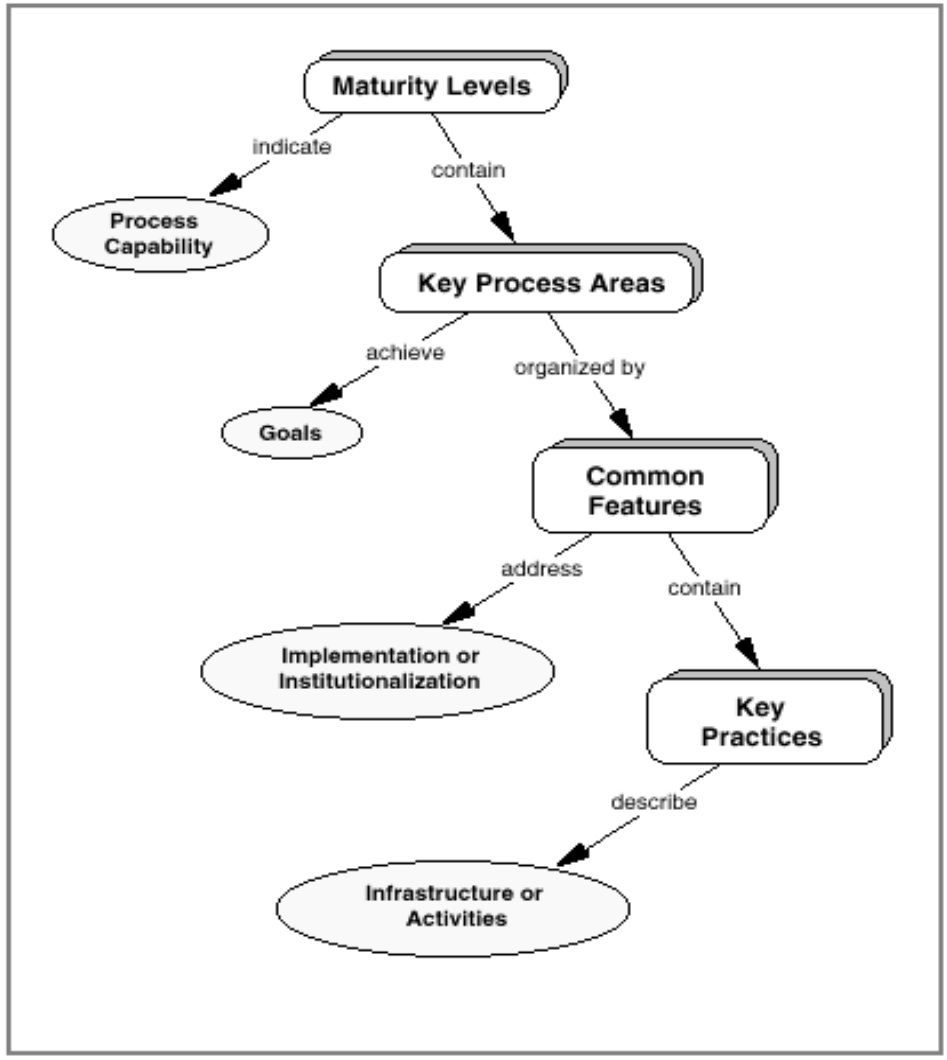
“The quality of a product is largely determined by the quality of the process that is used to develop and maintain it.”



CMMI Maturity Levels



The CMM Structure



Process Areas by Maturity Level

Level	Focus	Process Areas
5 Optimizing	*Continuous process improvement*	Organizational Innovation and Deployment Causal Analysis and Resolution
4 Quantitatively Managed	*Quantitative management*	Organizational Process Performance Quantitative Project Management
3 Defined	*Process standardization*	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Integrated Supplier Management (SS) Risk Management Decision Analysis and Resolution Organizational Environment for Integration (IPPD) Integrated Teaming (IPPD)
2 Managed	*Basic project management*	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management
1 Initial		

Is the premise true?

Process improvement should be done to help the business— not for its own sake.

Software Review and Inspection

Leiden Institute of Advanced Computer Science

Review

- A **Review** is a reading technique in which a software artifact is checked for defects by one or more persons other than the creator(s) of the document.
- Review can be applied to any type of document: code, design documents, test plans and requirements
- There are a number of types of review ranging in formality and effect.

Types of Review

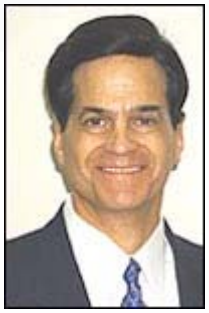
- Buddy Checking
  - having a person other than the author **informally** review a piece of work.
  - generally does not involve the use of checklists to guide inspection and is therefore **not repeatable**.
  - generally does not require collection of data
  - **difficult to put under managerial control**

Types of Review

- Walkthrough
  - the author of an artifact **presents** his document or program to an audience of peers
  - The audience asks **questions** and makes comments on the artifact being presented in an attempt to identify defects
  - often break down into arguments about an issue
  - usually involve **no prior preparation** on behalf of the audience
  - usually involve minimal documentation of the process and of the issues found
  - process improvement and defect tracking are therefore not easy

Types of Review

- Review by Circulation
  - similar in concept to a walkthrough
  - artifact to be reviewed is **circulated** to a group of the author(s) peers for comment
  - avoids potential arguments over issues, however it also **avoids the benefits of discussion**
  - reviewer may be able to spend **longer reviewing** the artifact
  - there is documentation of the issues found, enabling defect tracking
  - usually minimal data collection



Types of Review

- Inspection (Fagan 76)
  - formally structured and managed peer review processes
  - involve a review team with clearly defined roles
  - **specific data is collected** during inspections
  - inspections have quantitative goals set
  - reviewers check an artifact against an unambiguous set of **inspection criteria** for that type of artifact
  - The required data collection promotes process improvement, and subsequent improvements in quality.

Software Inspection

- The inspection process comprises three broad stages:
  - preparation
  - collection
  - repair

- Gilb and Graham [GilbGraham93] expand this three stage process into the inspection steps; Entry, Planning, Kickoff Meeting, Individual Checking, Logging Meeting, Root Cause Analysis Edit, Follow Up, Exit.

Principles of inspecting

- Choose an effective and efficient inspection team
  - between two and five people
  - Including **experienced** software engineers
- Require that participants **prepare** for inspections
  - They should **study** the documents **prior to the meeting** and come prepared with a list of defects
- Only inspect documents that are ready
  - Attempting to inspect a very poor document will result in defects being missed

Benefits of Inspection

believers edition

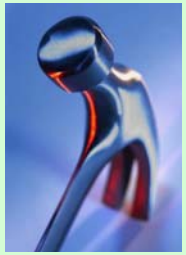
- 30% to 100% net productivity increases;
- Overall project time saving of 10% to 30%;
- 5 to 10 times reduction in test execution costs and time;
- Reduction in maintenance costs of up to one order of magnitude;
- Improvement in consequent product quality;
- Minimal defect correction backlash at systems integration time.
- In addition to these tangible benefits, less tangible benefits such as a training effect for inspectors are also evident.

Silver bullet?

Benefits of Inspection

Chaudron edition

- Helps creating common understanding and shared vision of the system
- Small investment in effort can have large benefits
  - Becomes better when staff is trained and checklists and reading guidelines are available
- Subjective
- Does not solve all problems
  - ➔ should be used in combination with other QA techniques



A peer-review

- Managers are normally not involved
  - This allows the participants to express their criticisms more openly, not fearing repercussions
  - The members of an inspection team should feel they are all working together to create a better document
  - Nobody should be blamed

Egoless-ness

- You are not your document/code
- Being open to improvement
- Seeing feedback as a learning opportunity

- Nobody is perfect

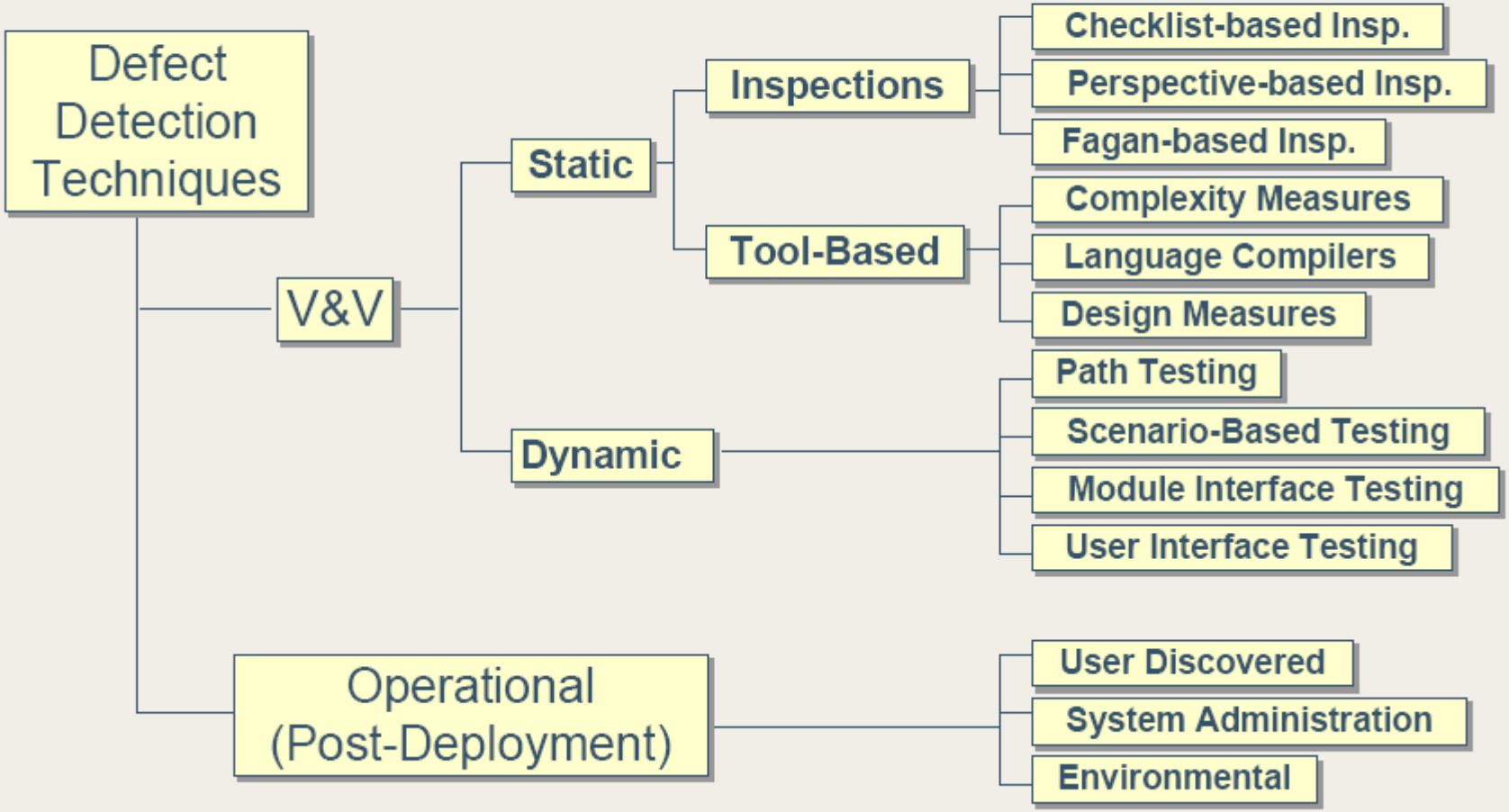
Quality Improvement Methods

- Structured processes
- Reviews and Inspections
- Metrics
- Testing
- Prototyping
- Mathematical proof of correctness / formal specification

Carnegie Mellon
Software Engineering Institute

Defect Discovery Sources

(how are the data generated)



Risk Management

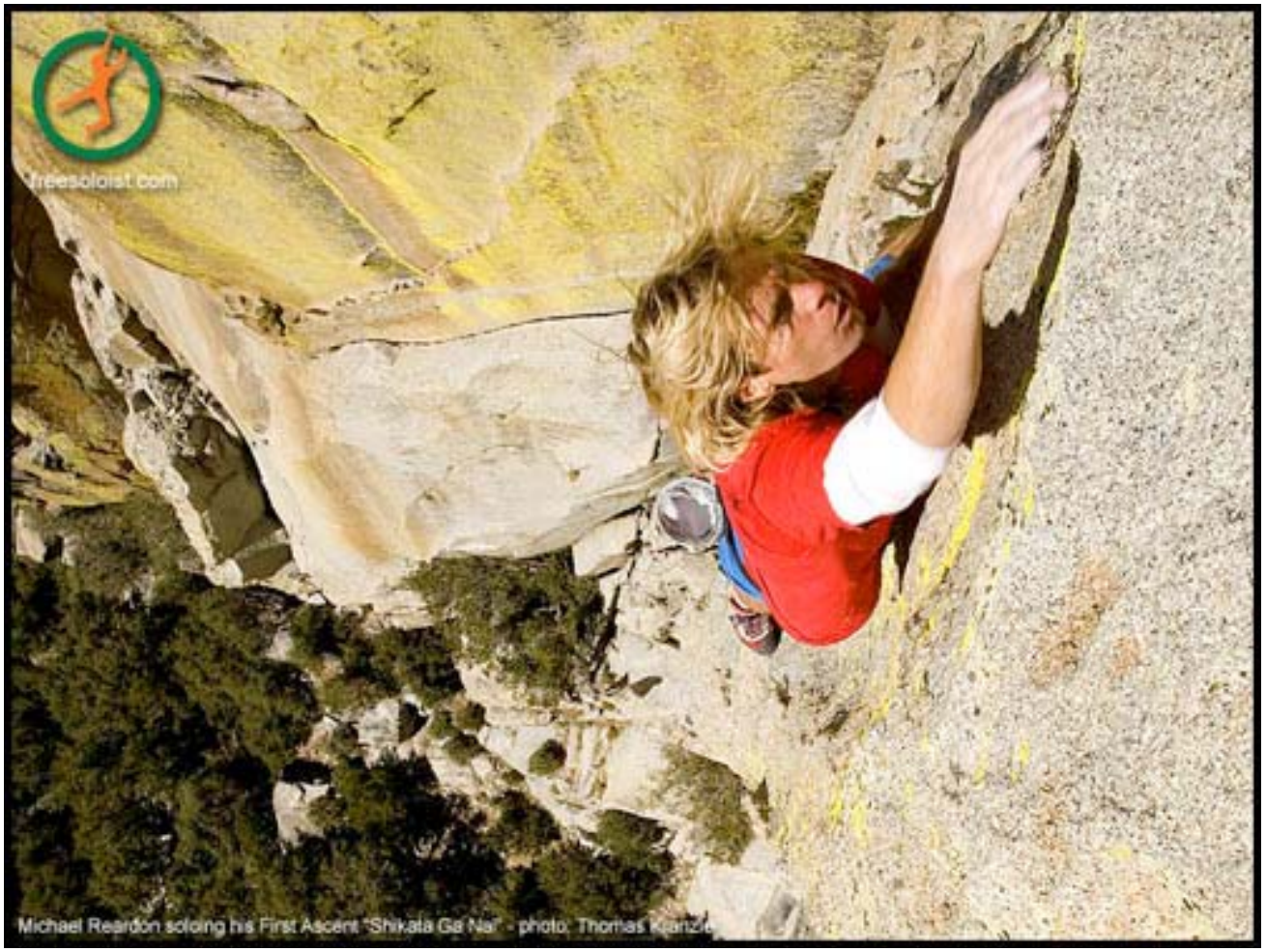
Leiden Institute of Advanced Computer Science

Risk Management

- Risk
  - Risk refers to uncertainty about the structure, outcomes or consequences of a decision or plan.

- Risk Management?
  - A Method for Dealing with Project Risks
    - Identification and Handling of Risks
  - On-Going Activity

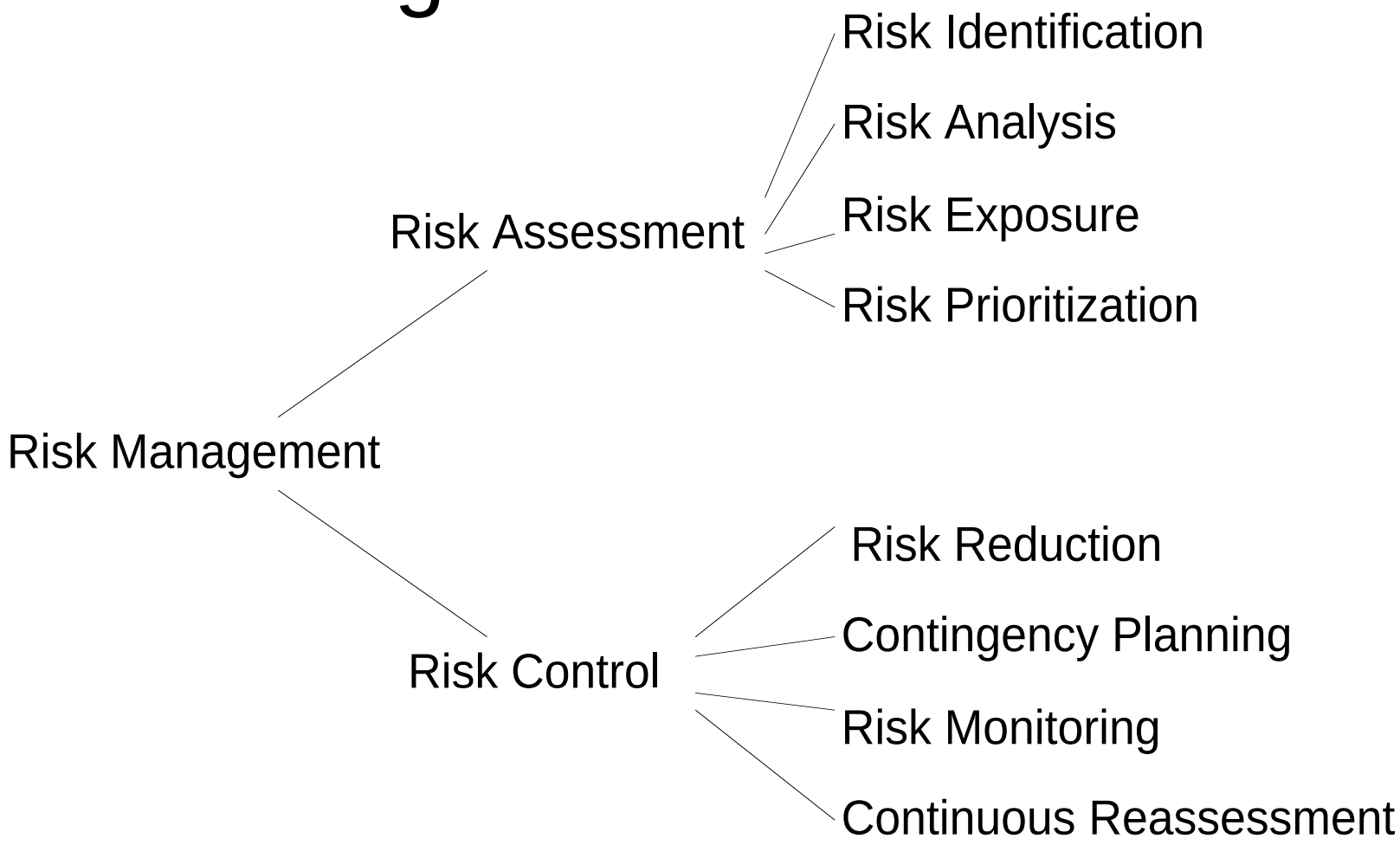
Risk?


freesoloist.com
Michael Reardon soloing his First Ascent "Shikata Ga Nai" - photo: Thomas Kranzle

Risk Management: Basic Approach

- Analysis of Project
  - Identification of Risks
- For Each Risk:
  - Impact and Probability Analysis
    - What is the Nature of the Risk?
  - Avoidance/Mitigation Plans
    - How Can We Minimize the Risk?
  - Contingency Plans
    - What Do We Do if it Occurs?

Risk Management



Risk Management: How to Identify Risks

- Start with a typical list of software risks
- Review development plan
  - Critical Paths
  - Critical Staff Members
  - Critical Vendor Deliveries
  - Critical Milestones
  - Training Requirements
- Review Requirements
- Review Technical Design
- Review Past Projects

Risk Management: How to Identify Risks (Continued)

- Conduct Risk Brainstorming Sessions with Staff, Users, Vendors, Customers, and Management
  - Try to assess the direction of thinking by third parties as they may give an indication of future requirements, expectations, or vendor changes.
  - If you are dependent on vendors, try to understand their business situation.
- Get as much input as possible!

Common Risks in IT Development

#	Name of the risk item	Description	Stakeholder concerned[11]	Software risk component related to
1	Personnel shortfalls	Lack of qualified personnel and their change	Customer, users, subordinates, maintainers, bosses, project manager	Personnel management risks
2	Unrealistic schedules and budgets	Development time and budget estimated incorrectly (too low)	Customers, bosses, project manager	Scheduling and timing
3	Developing wrong software functions	Development of software functions that are not needed or are wrongly specified	User, project manager	System functionality
4	Developing wrong user interface	Inadequate or difficult user interface	User, project manager	
5	Gold plating	Adding unnecessary features ("whistles and bells") to software because of professional interest or pride or user's demands	Sub-ordinates, users, project manager	Requirements management

Common Risks in IT Development

				Requirements management
6	Continuing stream of requirement changes	Uncontrolled and unpredictable change of system functions and features	Sub-ordinates, user, project manager	
7	Shortfalls in externally furnished components	Poor quality of system components that have been delivered externally	Customers, bosses, project manager	Sub-contracting
8	Shortfalls in externally performed tasks	Poor quality or unpredictable accomplishment of tasks that are performed outside the organization.	Customers, bosses, project manager	
9	Real-time performance shortfalls	Poor performance of the resulting system	Users, customer, maintainers, project manager	Resource usage and performance
10	Straining computer science capabilities	Inability to implement the system because of lacking technical solutions and computing power.	Sub-ordinates, users, customers, project manager	

Software Risk Management Techniques

Risk items	Risk management techniques
Personnel shortfalls	Staffing with top talent; job matching; team-building;
Unrealistic schedules & budgets	Detailed multisource cost & schedule estimation; incremental development; software reuse
Developing the wrong software functions	Organization analysis; mission analysis; user surveys; prototyping; early users manuals
Developing the wrong user interface	Prototyping; scenarios; task analysis; user characterization
Continuing stream of requirements changes	Information hiding; incremental development (defer changes to later increments)
Real-time performance shortfalls	Simulation; benchmarking; modeling; prototyping

Risk analysis

- estimating size of loss
  - how long it takes to “fix” the risk
- estimating probability of loss
  - most experienced estimates risks
  - delphi method vs. group consensus
  - betting on topic
  - adjective calibration
- risk exposure
  - probability of unexpected loss multiplied by the size of loss

Analysis, Exposure, & Prioritization

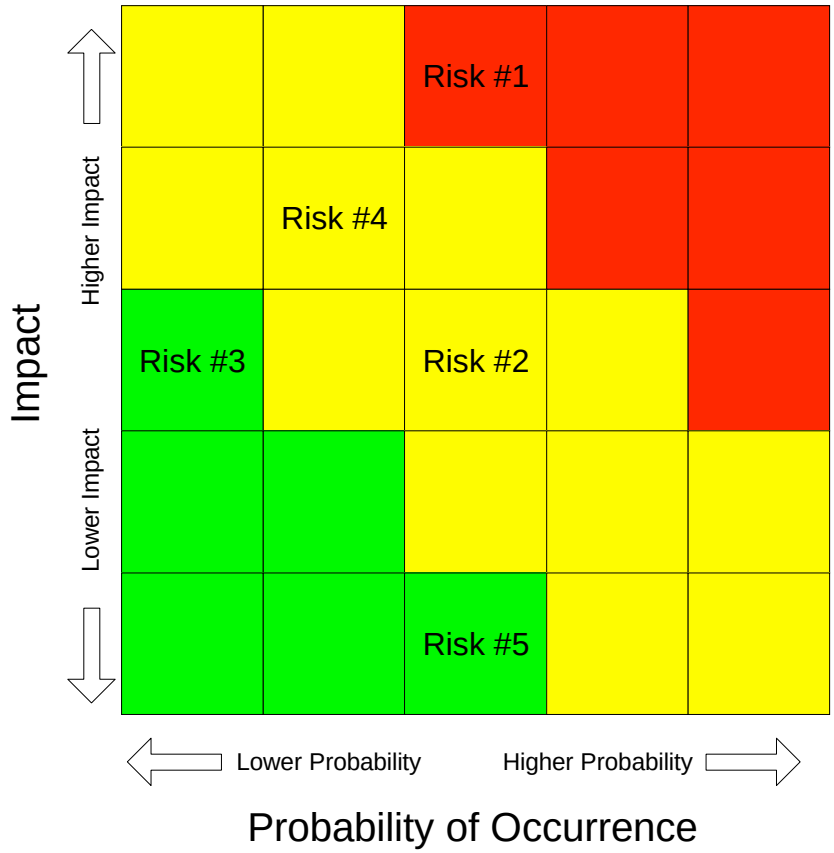
- For Each Risk:
  - Determine Probability of Occurrence
    - What is the likelyhood of occurrence?
  - Determine Impact
    - What is the impact if it occurres?
  - Determine Exposure
    - What will we lose if the risk occurs?
- For All Risks:
  - Prioritize
    - Where should we put our limited resources?

Analysis, Exposure, Prioritization: How?

- Various Techniques Available But Key is Experience
  - Individual
  - Organizational
- Don’t Rely on Just Yourself - Get lots of Inputs

Risk Assessment: A Simple Classification & Tracking Method

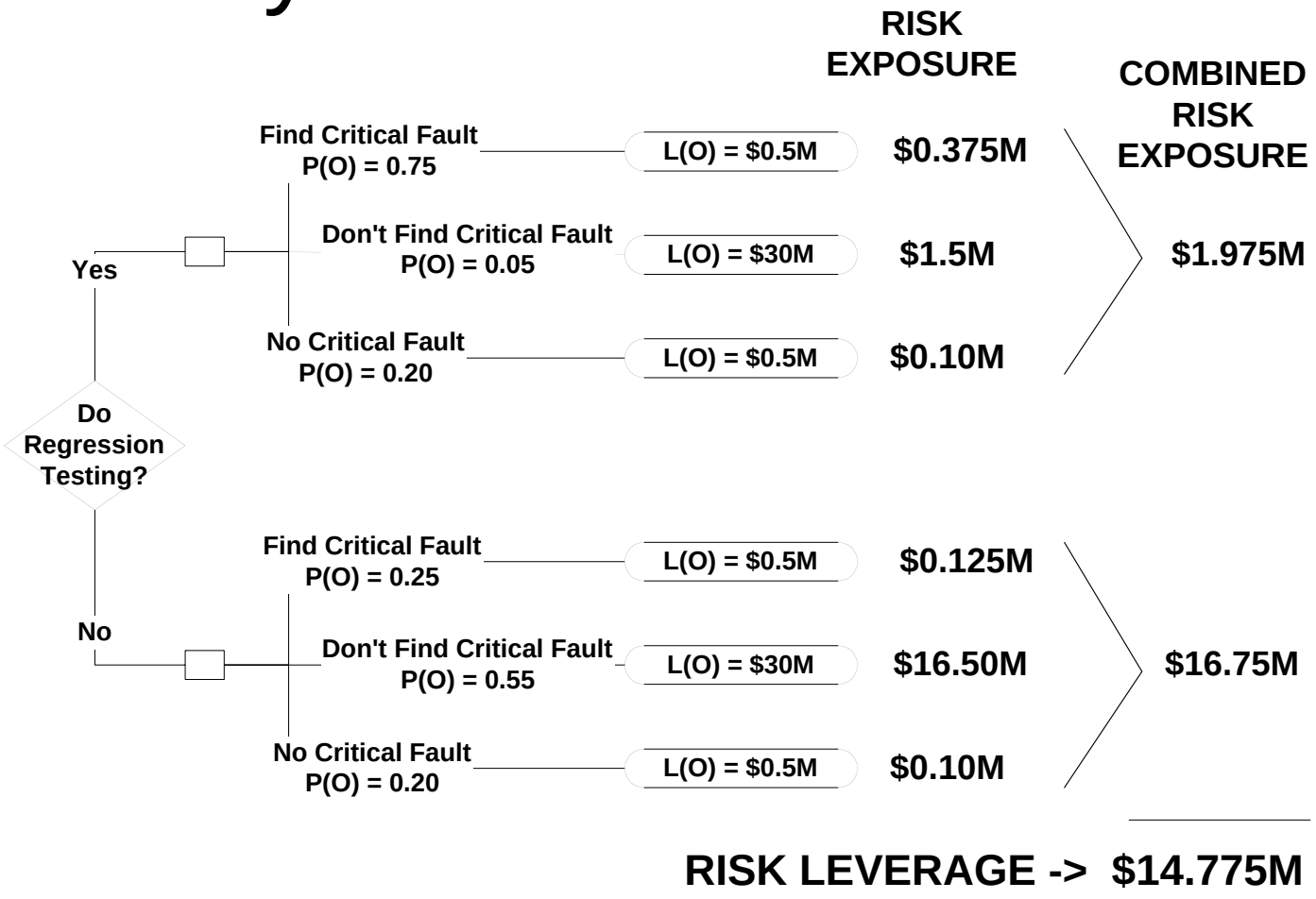
- Probability of Occurrence vs Impact
  - 1 to 5 Scale
- Priorities
  - Red - High
  - Yellow - Med
  - Green - Low
- Review/Present Chart Periodically



Risk Assessment: Probability Methods

- Can we quantitize the risk?
- For Each Risk:
  - For Each Possible Action:
    - Estimate Probability of an Given Outcome P(O)
    - Estimate $ Loss of an Given Outcome L(O)
    - Multiply the P(O) by L(O) to give $ exposure for the unwanted outcome
  - Sum all $ exposures for each Possible Action
  - Compare the $ exposures
  - Calculate Risk Leverage
    - (Risk Exposure Before Reduction - Risk Exposure After Reduction) / (Cost of Risk Reduction)

Example Risk Assessment Using Probability Method



Risk Reduction

- Avoiding Risk
  - Modifying project requirements
- Transferring the Risk
  - By allocation to other systems
  - Buying Insurance to cover financial loses
- Mitigating the Risk
  - Pre-Event Actions to:
    - Reduce Likelihood of Occurrence and/or
    - Minimize Impact, Fail-over, Repair, …
- Some risks cannot be reduced
  - Contingency Plan - how will you deal with the risk

Monitoring Risk

- Periodic Review of Risk Status
  - Changes in Probabilities or Impacts
  - Changes in Avoidance/Mitigation/Contingency Plans
- Periodic Review of Project to Identify New Risks
- Implementation of Risk Avoidance or Mitigation Plans
- Keep Management and Customers Informed!!!
  - Frequent Risk Reviews

Risk Management Process

From: http://www.ruleworks.co.uk/riskguide/manage-risk-nl.htm

Empirical Research in Software Engineering

Leiden Institute of Advanced Computer Science

Empirical Research

Empirical research

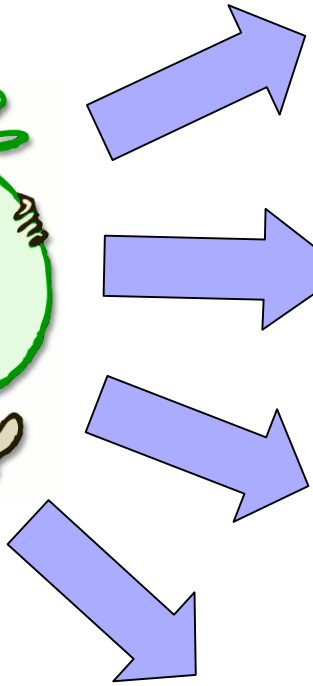
is research that bases its findings on direct or indirect observation as its test of reality.

Astronomy

Chemistry

Physics
Newton's apple

How to best allocate budget?



people

- Experienced .. Novice

technology

- Java vs .Net
- Code generation
- Automated testing, …

process

- In-house or offshore
- …

product

- Which features
- What level of quality?

We must understand the effect of our choices on productivity, quality, …

Examples

- The use of Object Oriented modeling and programming improves quality and productivity
  - True ?
  - Not True?
  - Don’t know

The Bottom Line

**“In God we trust,
all others bring data.”**

- W. Edwards Deming

What is ‘evidence’?

Use of RUP

Use of RUP leads to improvement of productivity and quality

Approaches:

- Measure
- Expert opinions (interviews)
- Simulation

Combination of the above (triangulation)

Empirical Cycle

Real World

Theory

$E = a.N^b$

Observation

Data / Results

Induction

Idea / Conclusion

Hypothesis

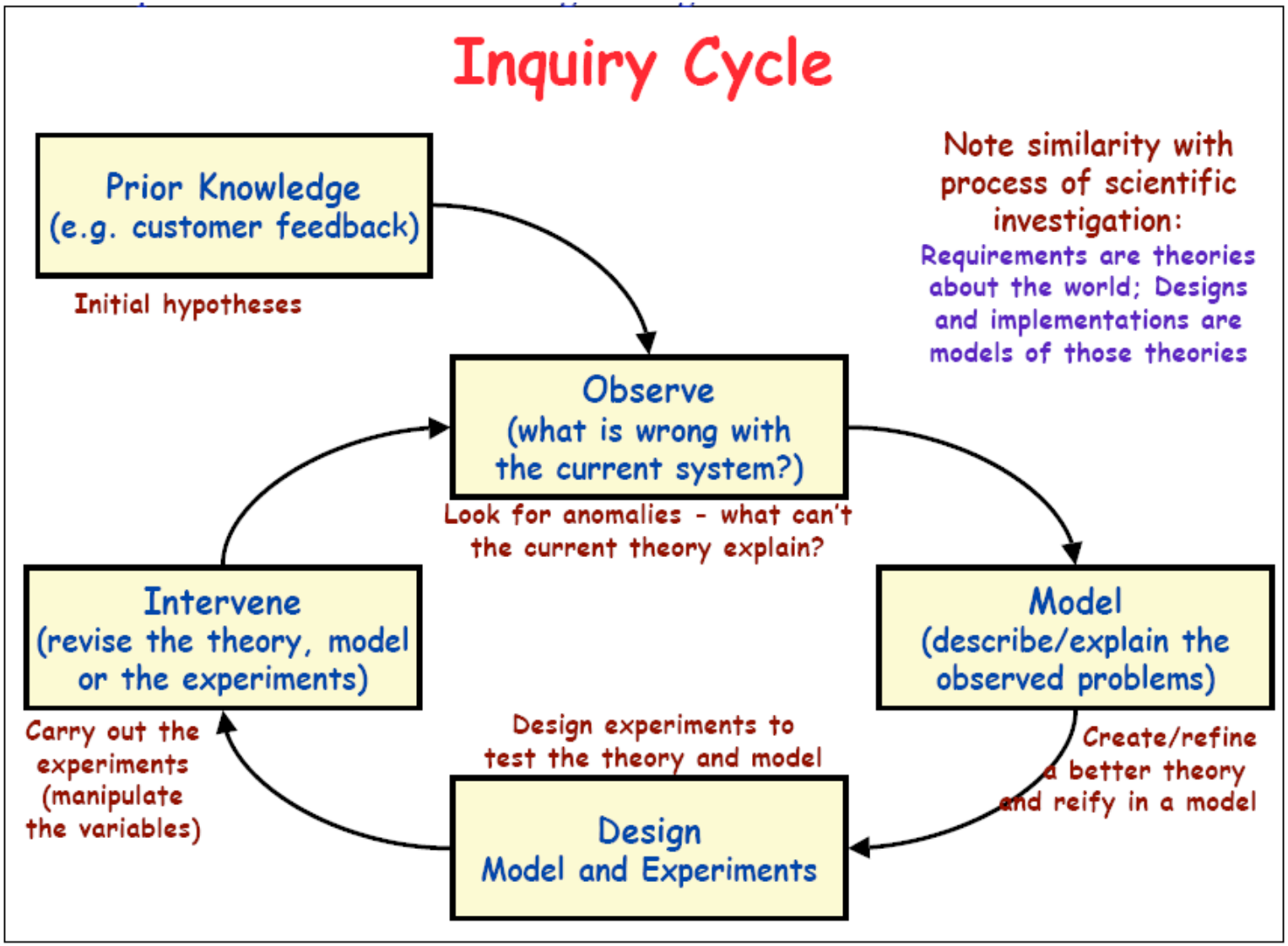
Model

Deduction

Prediction

Test

Experiment / Intervention


Inquiry Cycle
Prior Knowledge
(e.g. customer feedback)
Initial hypotheses
Note similarity with process of scientific investigation:
Requirements are theories about the world; Designs and implementations are models of those theories
Observe
(what is wrong with the current system?)
Look for anomalies - what can't the current theory explain?
Model
(describe/explain the observed problems)
Create/refine a better theory and reify in a model
Design
Model and Experiments
Design experiments to test the theory and model
Intervene
(revise the theory, model or the experiments)
Carry out the experiments (manipulate the variables)

Important characteristics of scientific research:

- rigor
- testability / falsifiability
- reproducibility
- precision
- objectivity
- parsimony
- generalisability (if possible)

Many Methods Available:

- Laboratory Experiments
- Field Studies
- Case Studies
- Pilot Studies
- Rational Reconstructions
- Exemplars
- Surveys
- Artifact/Archive Analysis ("mining"!)
- Ethnographies
- Action Research
- Simulations
- Benchmarks

Study - Examples

- Survey
  - After a new development process has been introduced: developers answer a questionnaire about their confidence in the new process.
- Experiment
  - Source code inspections: one group of participants uses inspection technique A, the other group uses inspection technique B. Compare the number of detected defects.
- Case study
  - Run a pilot project using a new tool (e.g. UML case tool) and compare productivity to company baseline

Experiment

- **When appropriate***:* control on who is using which technology, when, where and under which conditions. Investigation of self-standing tasks where results can be obtained immediately
- **Level of control***:* high
- **Data collection***:* process and product measurement, questionnaires
- **Data analysis***:* parametric and non-parametric statistics, compare central tendencies of treatments, groups
- **Pro's***:* help establishing causal relationships, confirm theories
- **Con's***:* representative? Challenging to plan in a real-world environment. Application in industrial context requires compromises

Case study

- **When appropriate***:* change (new technology) is wide-ranging throughout the development process, want to assess a change in a typical situation
- **Level of control***:* medium
- **Data collection***:* product and process measurement, questionnaires, interviews
- **Data analysis***:* compare case study results to a baseline (sister project, company baseline)
- **Pro's***:* applicable to real world projects, help answering why and how questions, provide qualitative insight
- **Con's***:* difficult to implement a case study design, confounding factors, analysis of results is subjective

Survey

- **When appropriate***:* for early exploratory analysis. Technology change implemented across a large number of projects, description of results, influence factors, differences and commonalities
- **Level of control***:* low
- **Data collection***:* questionnaires, interviews
- **Data analysis***:* comparing different populations among respondents, association and trend analysis, consistency of scores
- **Pro's***:* generalization of results is usually easier (than case study), applicable in practice
- **Con's***:* little control of variables, questionnaire design is difficult (validity, reliability), execution is often time consuming (interviews)

Empirical Life-cycle

Method Development

Initial Idea

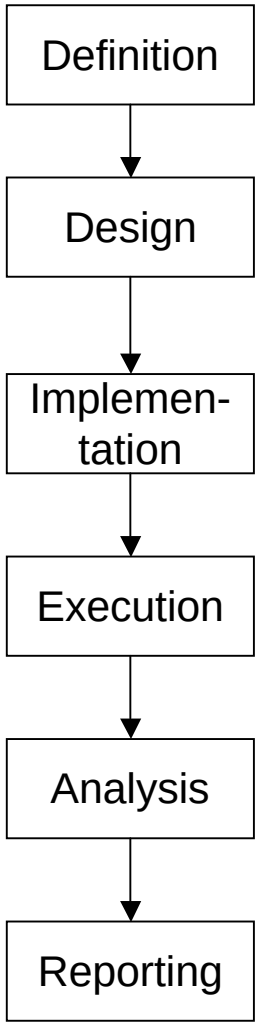
Exploratory Interviews

Survey

Industrial Case Studies

Experiment

A process for conducting empirical studies

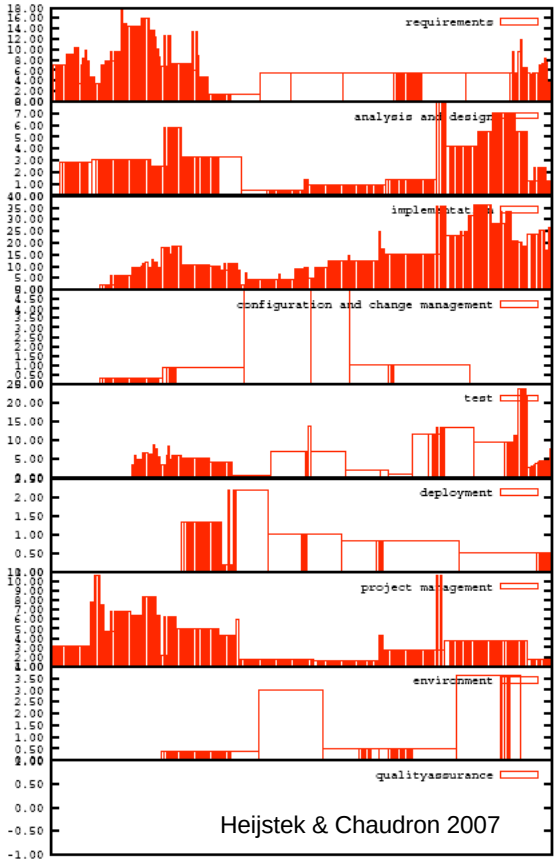


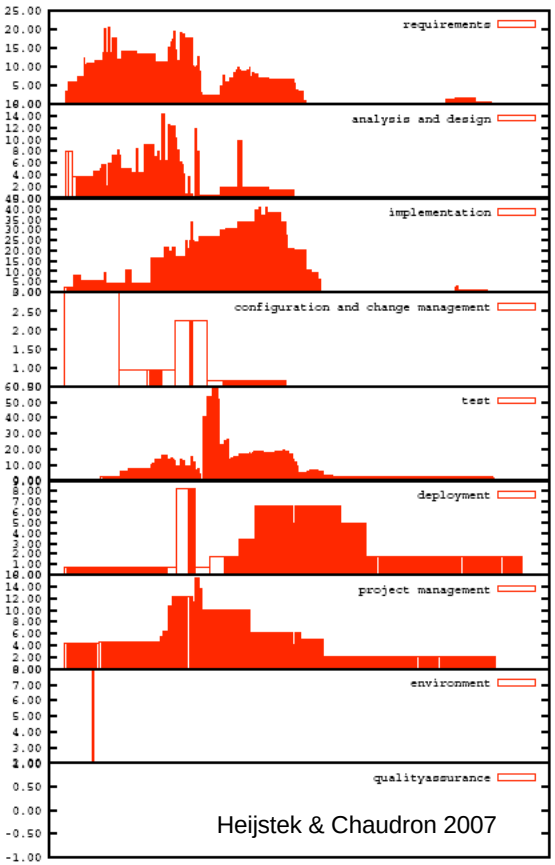
- Determine study goal and research hypothesis. Select type of empirical study to be employed.
- Operationalize study goal and hypothesis.
  Make study plan: what needs to be done by whom and when.
- Prepare material required to conduct the study.
- Run study according to plan and collect required data (data collection).
- Analyze collected data to answer operationalized study goal and hypotheses
- Report your study so that external parties are able to understand results and context of the study.

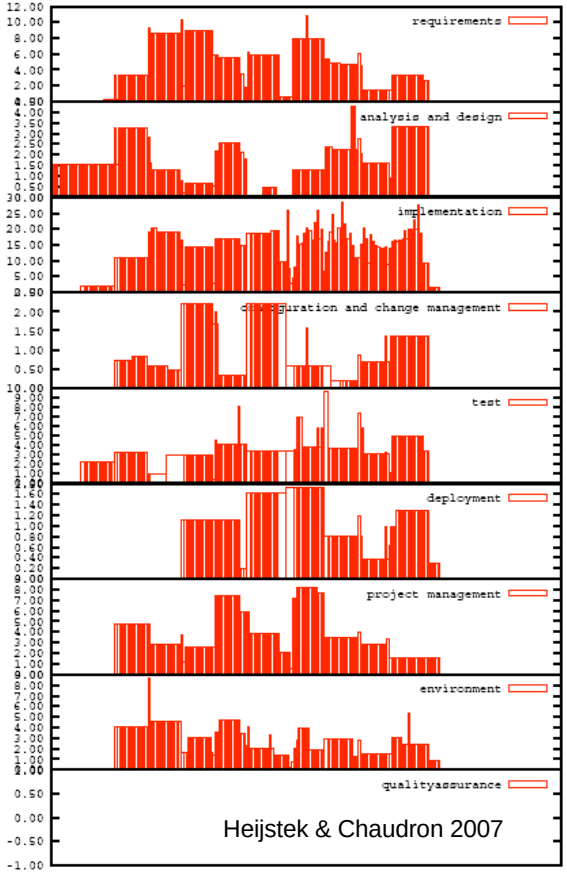
Validity

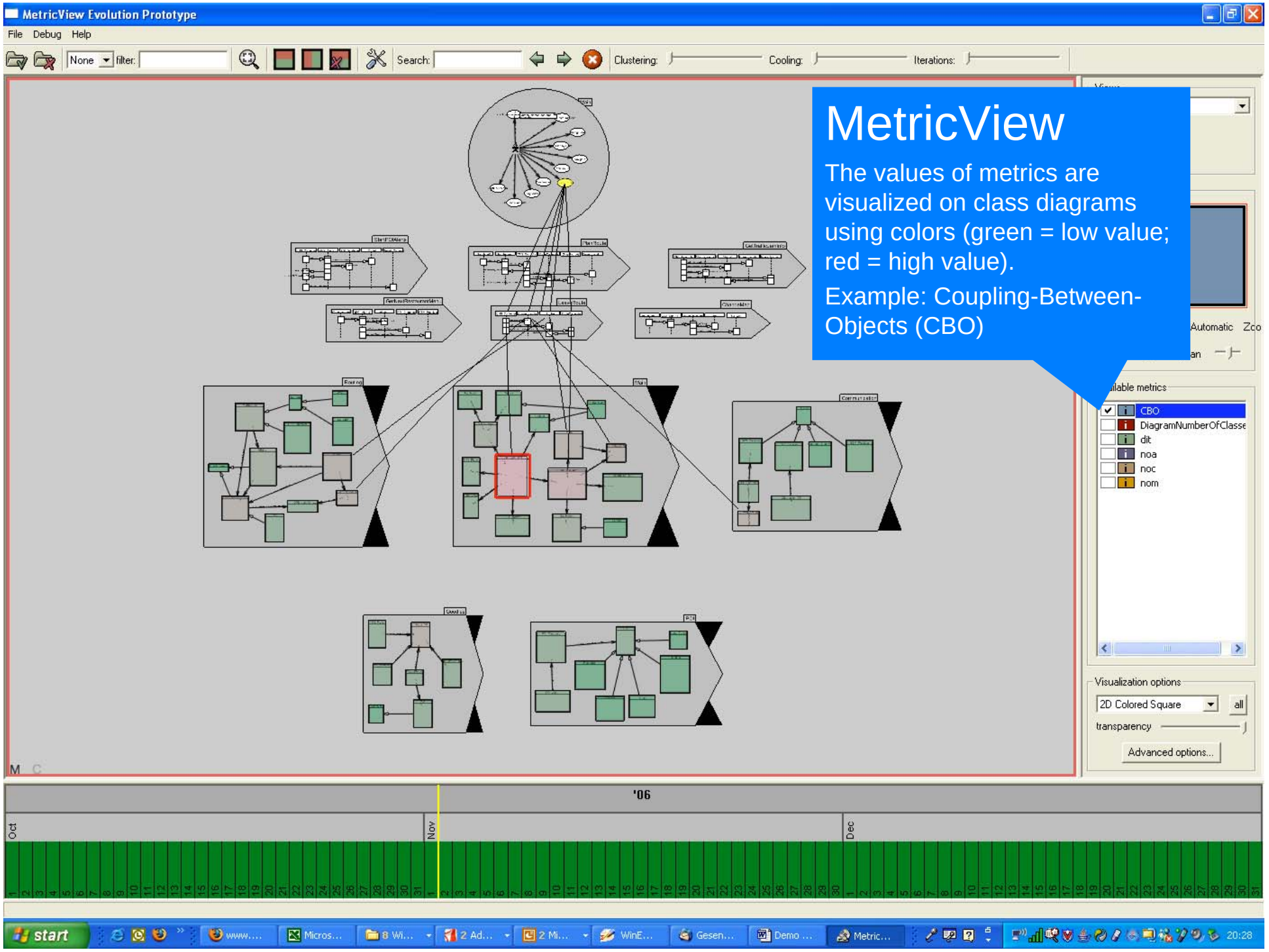
- Are the results valid for the sample population?
- Are the results valid for the population to which we would like to generalize?
- Threats to Validity
  - Conclusion validity
    - Relation between treatment and outcome
  - Internal validity
    - Treatment → outcome = causal relationship?
  - Construct validity
    - Relation between theory and observation
  - External validity
    - Generalizability of the result

RUP Humps from 3 (largish) projects








MetricView Evolution Prototype
File Debug Help
None filter:
Search:
Clustering:
Cooling:
Iterations:
MetricView
The values of metrics are visualized on class diagrams using colors (green = low value; red = high value).
Example: Coupling-Between-Objects (CBO)
Automatic
CBO
DiagramNumberOfClasse
dit
noa
noc
nom
Visualization options
2D Colored Square
all
transparency
Advanced options...
'06
Oct
Nov
Dec
start
20:28

Conclusions

- Empirical Research is essential for validation of methods/techniques/processes in practice;
  - Feedback for improvement
  - Collaboration between industry and academia is essential
- Different study-types (‘strategies’) are possible.
  - Depending on the goal and context
  - Good preparation is important
  - Good literature is available

References

[1] A. Endres, D. Rombach, A Handbook of Software and Systems Engineering – Empirical Observations, Laws and Theories, Pearson Addison Wesley, 2003.

[2] C. Wohlin, P. Runeson, M. Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslen, Experimentation in Software Engineering - An Introduction. The Kluwer International Series in Software Engineering, Kluwer Academic Publishers, 2000.

[3] R. van Solingen and E. Berghout, The Goal/Question/Metric Method. McGraw-Hill, 1999.

[4] N. Fenton and Shari L. Pfleeger, Software Metrics: A Rigorous Practical Approach. London: International Thompson Computer Press, 1996.

[5] B. Freimut, T. Punter, S. Biffl, M. Ciolkowski, State-of-the-art in Empirical studies, IESE-Report No. 017.02/E & ViSEK report No. 007/02, Kaiserslautern, Fraunhofer IESE, March 2002.

[6] T. Punter, M. Ciolkowski, B. Freimut, I. John, Conducting on-line surveys in software engineering, ACM IEEE Int. Symposium on Empirical Software Engineering (ISESE'03), Los Alamitos, IEEE, pp. 80-88.

[7] B. Kitchenham, Evaluating Software Engineering Methods and Tools - Part 9: Quantitative Case Study Methodology, ACM SIGSOFT Software Engineering Notes, vol. 23, pp. 24-26, Jan. 1998.

[8] M.V. Zelkowitz, D.R. Wallace, Experimental models for validating Technology, IEEE Computer, vol. 31 no. 5, pp. 23-31, May 1998.

M.Sc. Eindprojecten met …

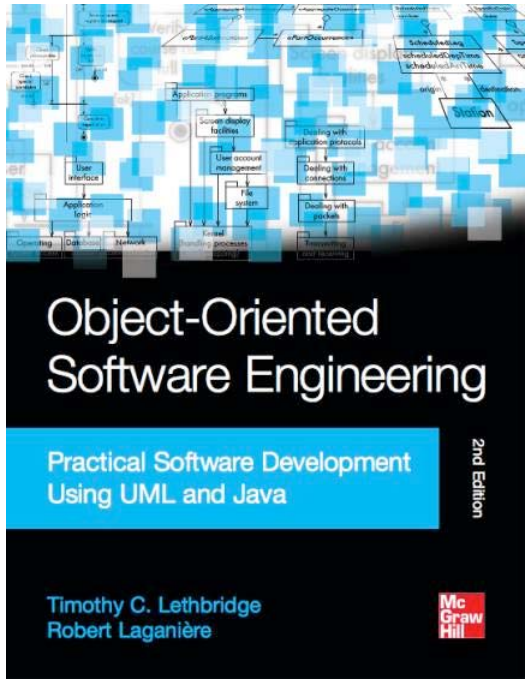
Guest lecture 24 april 11-13

And many more (including companies/universities) abroad …

Highlights SE

Leiden Institute of Advanced Computer Science

Book: Object-Oriented Software Engineering, Timothy C. Lethbridge, Robert Laganière (2nd Ed.)



- Ch 1: introduction to the subject
- Ch 2: OO-basics
- Ch 4: Requirements
- Ch 5 & Ch 8: Modeling using UML
- Ch 6: Design patterns
- Ch 9: Architecture & Designing
- Ch 10: Testing / Quality Assurance
- Ch 11: Management (Estimation, Risk)
- Websites: www.mhhe.com/lethbridge en www.llsoeng.com

Project Management

- People are key
  - Get good people, Make them happy, Set them loose
- Manage Risk Early and Frequently
- Anticipate changes

Requirements Engineering

- Understand the domain
- SMART
- Manage Change

Software Architecture

- Principle decisions about design of a system
- Describe using multiple views
- Validate architecture

  review, measure, prototype

- Mathematics is not a careful march down a well-cleared highway, but a journey into a strange wilderness, where the explorers often get lost.
  Rigour should be a signal to the historian that the maps have been made and the real explorers have gone elsewhere.

  [Anglin, W.S.]