20 JAHRE ITERATEC 1996-2016

iteratec

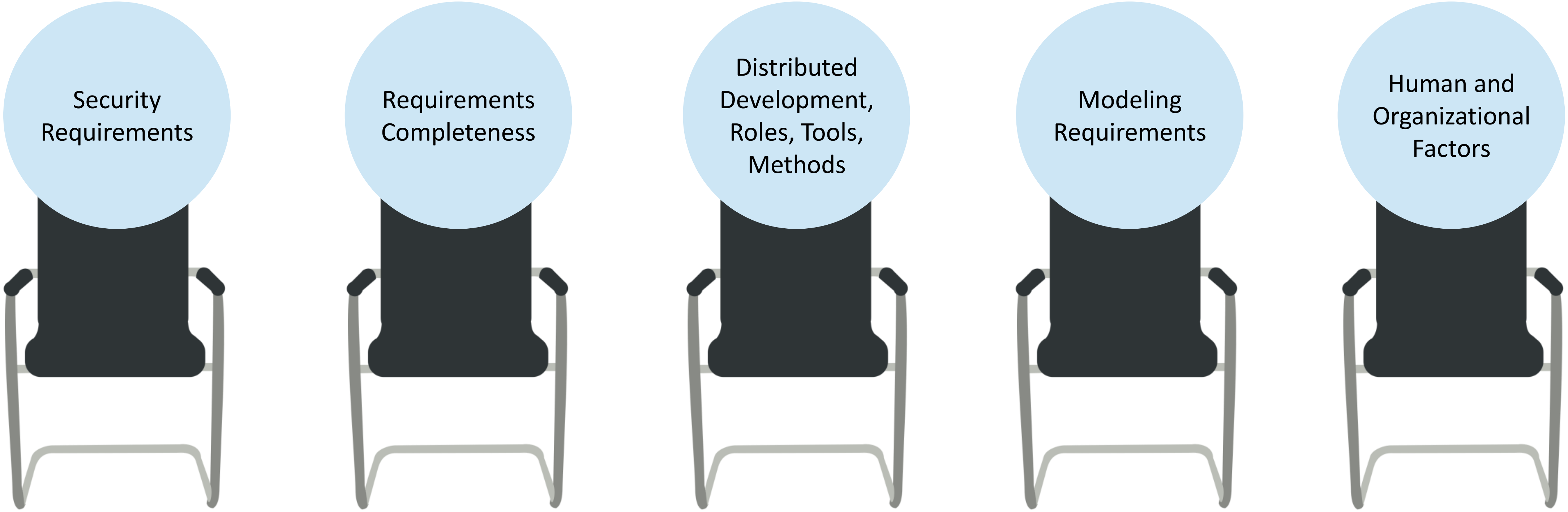
KOMPETENZ, DIE ENTLASTET

Panel on ICSEA/Req&Dev

Validating Products versus Requirements; Dis(covering) the Gaps

Michael Gebhart

Our Panelists



Security Requirements

Requirements Completeness

Distributed Development, Roles, Tools, Methods

Modeling Requirements

Human and Organizational Factors

Jon Geater
Thales e-Security-Ltd., UK

Heidar Pirzadeh
SAP SE, Canada

Mira Kajko-Mattsson
KTH Royal Institute of Technology, Sweden

Radek Koci
Brno University of Technology, Czech Republic

Luis Fernandez Sanz
Universidad de Alcalá, Spain

http://www.iceberg-sqa.eu

Human and organizational factors: impact on software quality

Panel: Validating Products versus Requirements; Dis(covering) the Gaps

Luis Fernández-Sanz, Universidad de Alcalá, Spain

Universidad de Alcalá

luis.fernandezs@uah.es

This work has been partially funded by European Commission for project ICEBERG no. 324356 (7h Framework Programme IAPP Marie Curie program).

SEVENTH FRAMEWORK PROGRAMME

ICSEA 2016, August 21 - 25, 2016 - Rome, Italy

Human and organizational factors

- Software projects are a social activity
- Addressed up to some extent in software engineering research, less than technical topics: e.g. project estimation
- Connection to software quality: neglected in research
  - Open to real practice and lack of data
  - Different research methods, qualitative and quantitative
  - Multidisciplinarity and exploring hybrid fields not understood by traditional researchers

Example

- Work in software testing:
  - How training impact effectiveness of test case design, 71 professionals
    - Less training, more duplicated/useless cases
    - Unsystematic design (<50% coverage)
    - Only 30-35% of software professionals trained in testing (3 surveys)

Example in requirements: analyzing multicultural teamwork

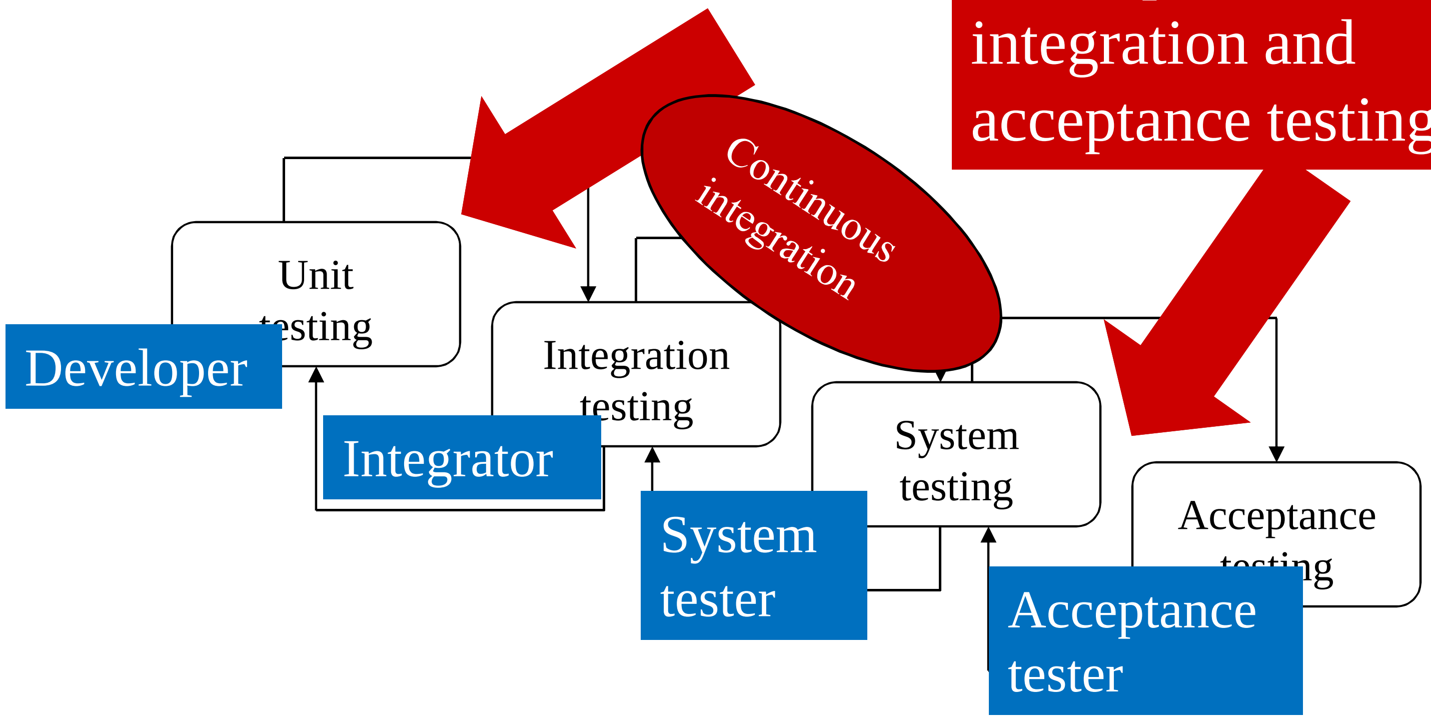
- Effects of teamwork in requirements analysis, real case for experiment
- Discovering reqs thru answers to questions (368 people, 6 countries)
    - Individually and then looking for team consensus
- Analysis of results: promotion of teamwork spirit
- But, analyzing results of multinational settings (Hofstede's indicators)
    - Yes, attitude's trends match with Hofstede's numbers
    - Higher IDV (individualism), poorer teamwork results
    - Higher UAI (uncertainty avoidance), better reqs. analysis results

Validating Products vs Reqs

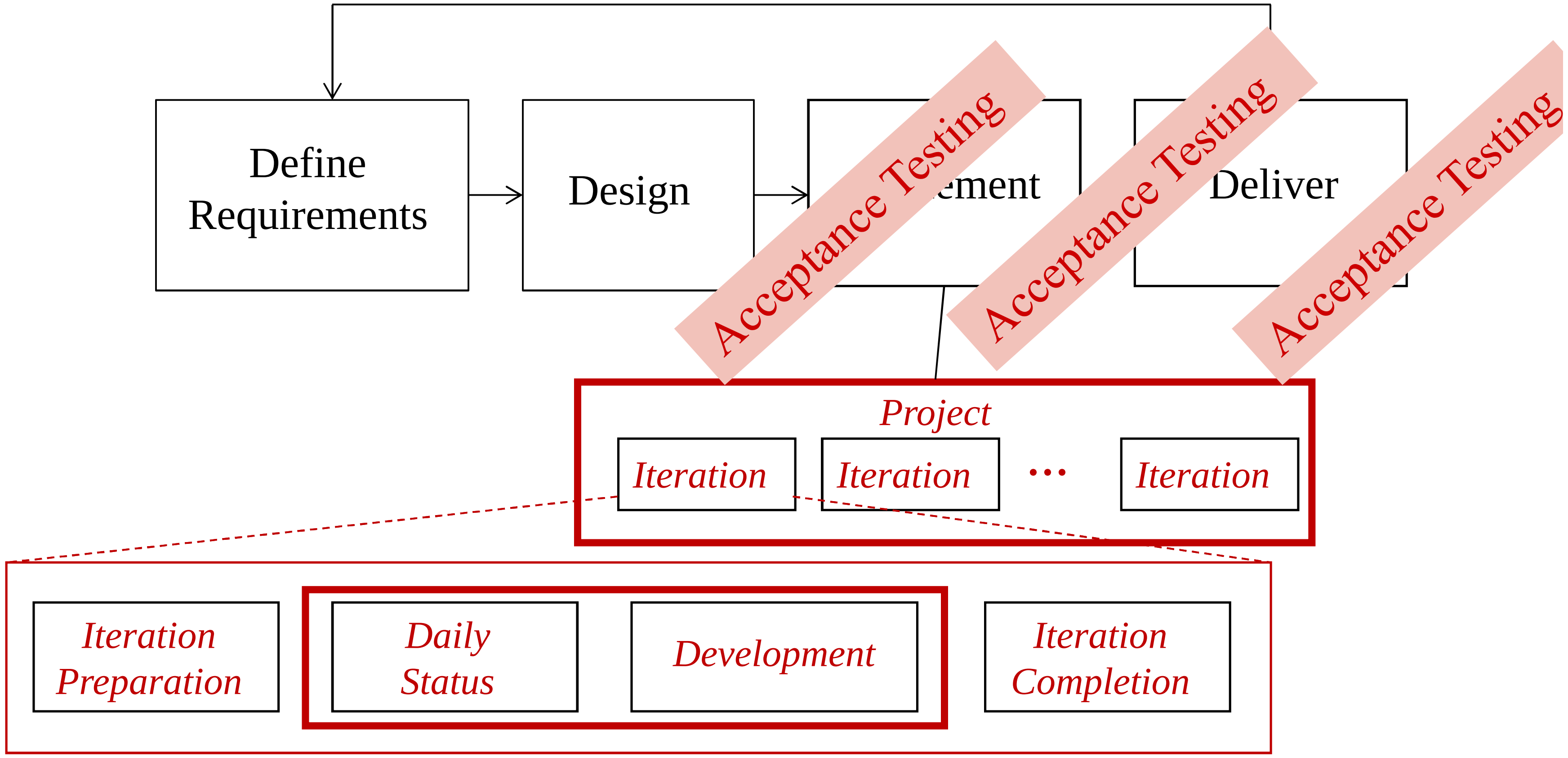
Mira Kajko-Mattsson
KTH Royal Institute of Technology
Sweden

Software testing phases

Agile mainly concentrates on developer's continuous integration and acceptance testing



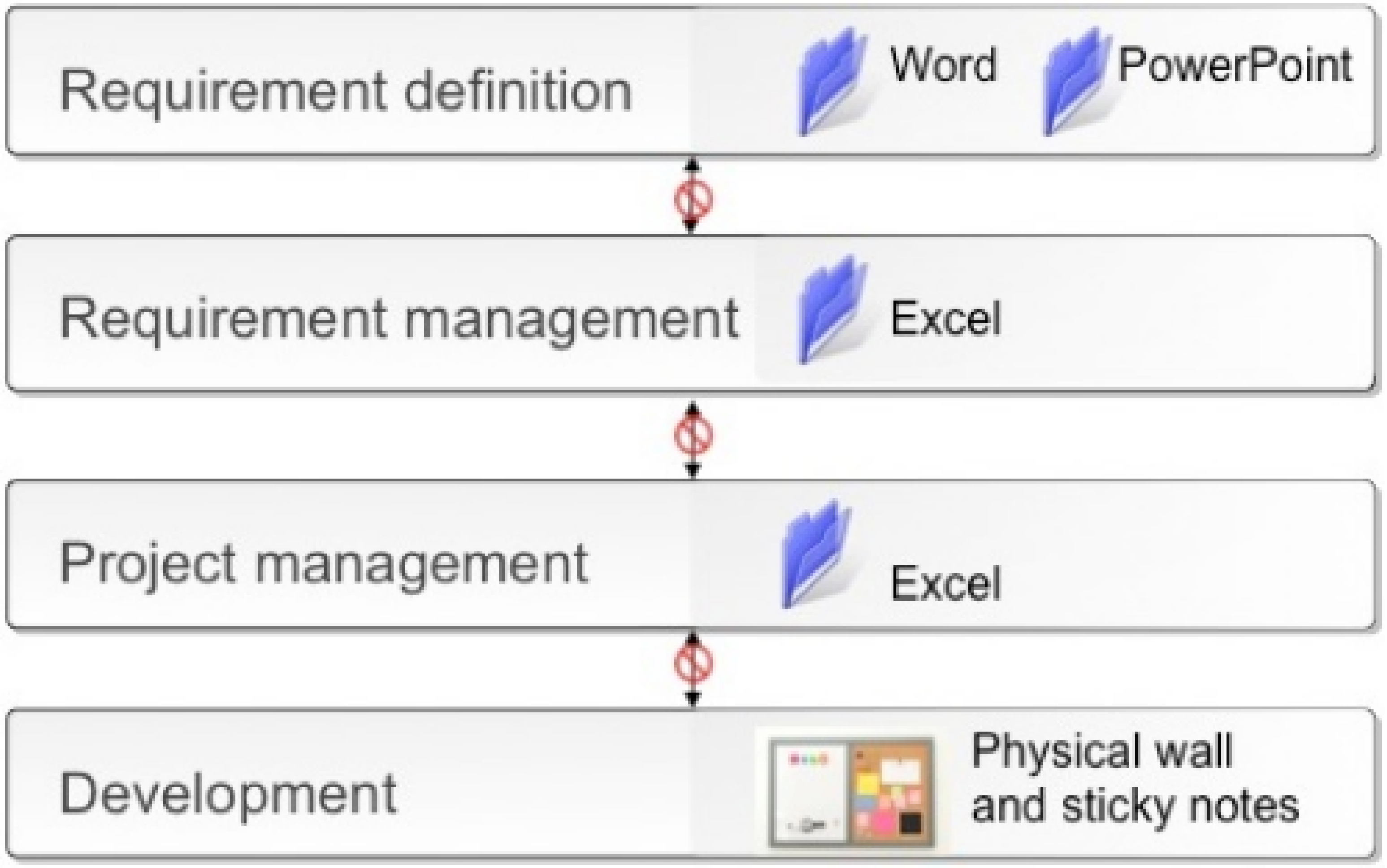
Agile Software Development Lifecycle, variant 1



Challenges

- Humans
- Supporting tools
- Developers’ process

Status within one company



Requirement definition

Word PowerPoint

Requirement management

Excel

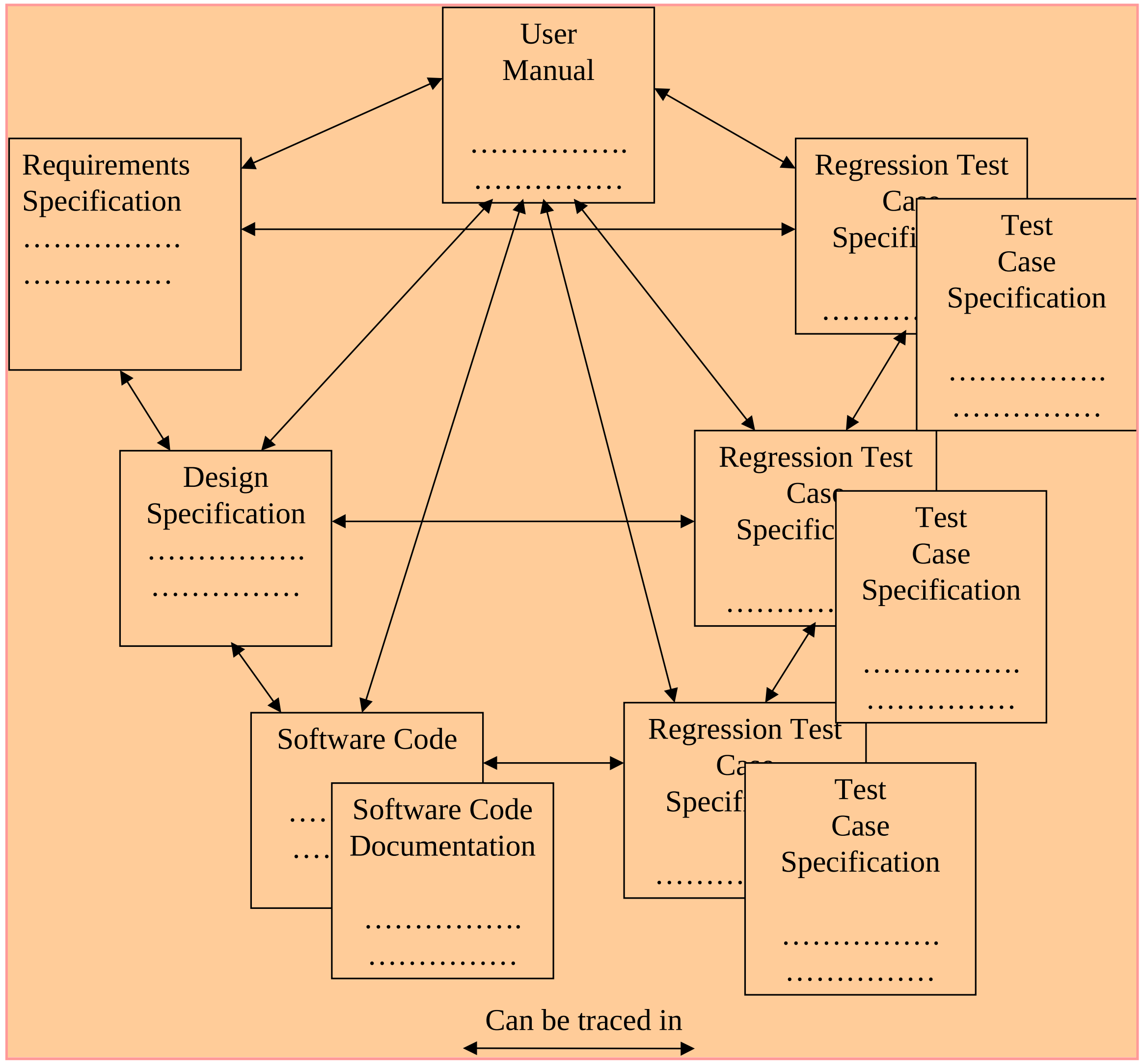
Project management

Excel

Development

Physical wall and sticky notes

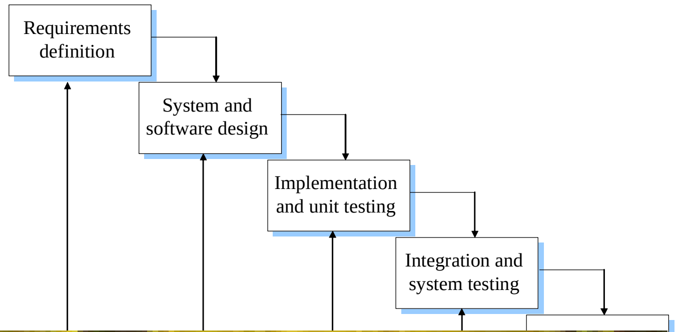
Traceability should be better supported by the tools



Developer's individual process should be improved

- Unit testing is the heart of agile methods
  - No modification or refactoring of code is complete until 100% of unit tests have run successfully.
  - No story is complete until all its acceptance tests have passed successfully.
- Is this enough?

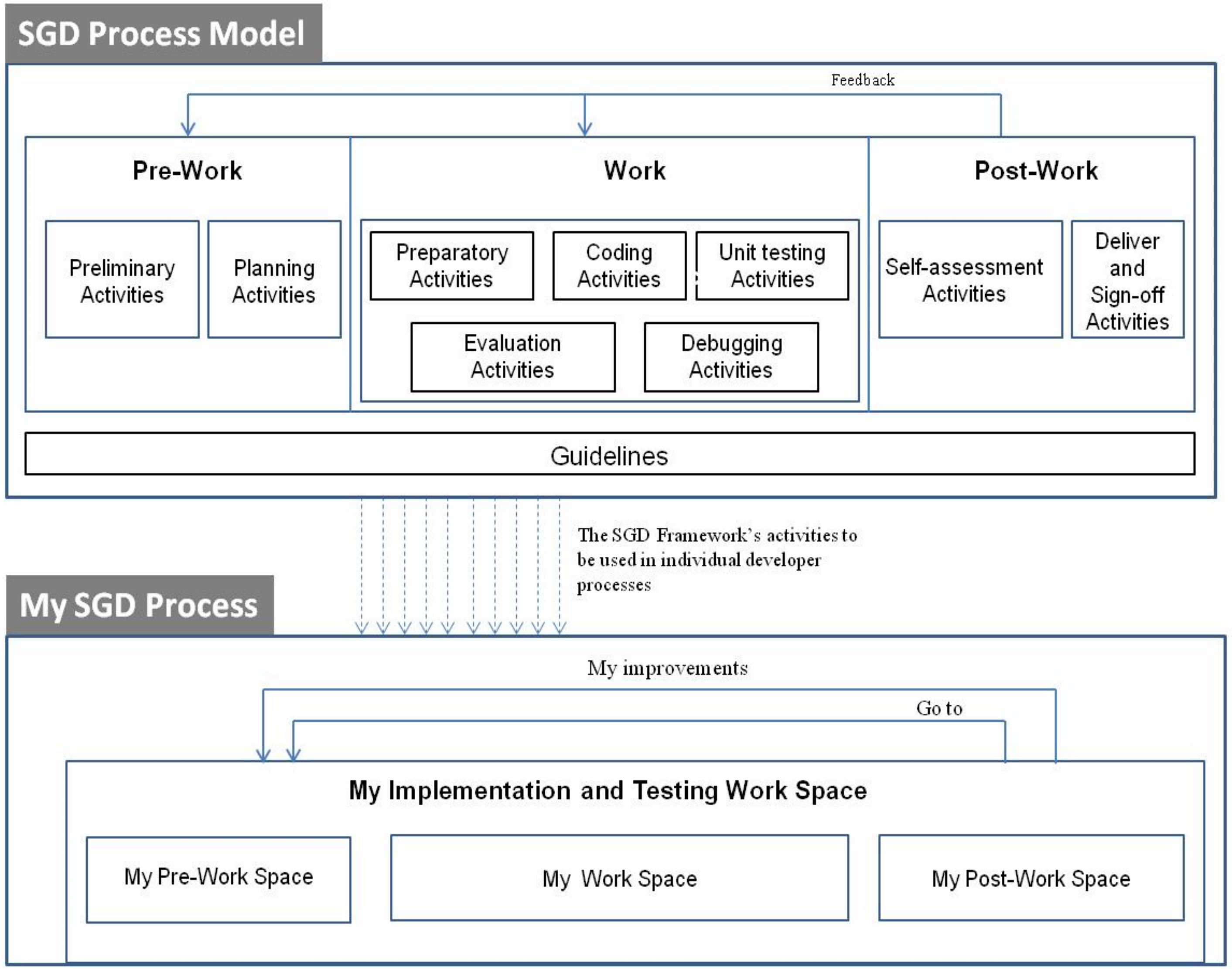
Methods


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing

Incremental developr

Managers wind up (clockwork) developers to follow the methods

Solution


SGD Process Model
Feedback
Pre-Work
Work
Post-Work
Preliminary Activities
Planning Activities
Preparatory Activities
Coding Activities
Unit testing Activities
Evaluation Activities
Debugging Activities
Self-assessment Activities
Deliver and Sign-off Activities
Guidelines
The SGD Framework's activities to be used in individual developer processes
My SGD Process
My improvements
Go to
My Implementation and Testing Work Space
My Pre-Work Space
My Work Space
My Post-Work Space

Validating Products versus Requirements; Dis(covering) the Gaps

Heidar Pirzadeh

Gap in Implementation

- Developers misunderstand the requirements, make implementation mistakes, or the requirements change during or after development.
  - Validation Testing
  - Release Testing
  - Requirement Based Testing
  - Freeze Requirements During an Increment

Gap in Requirements

- Requirements are incomplete or incorrect
  - Lots of space for interpretation
  - Partial market research
  - Literal translations of customer needs
  - Outdated

Gap in Identification of Complexity

- Processes for gathering requirements (as an initial step of problem solving) might not work depending of the complexity of the problem.
  - Simple Problems
  - Complicated Problems
  - Complex Problems [1]
  - Wicked [2] or Chaotic Problems [3]

References

1. C. F. Kurtz and D. J. Snowden. 2003. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Syst. J.* 42, 3 (July 2003), 462-483. DOI=http://dx.doi.org/10.1147/sj.423.0462
2. Rittel, H.W. and Webber, M.M., 1973. Dilemmas in a general theory of planning. *Policy sciences*, *4*(2), pp.155-169.
3. Lorenz, E.N., 1963. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, *20*(2), pp.130-141.

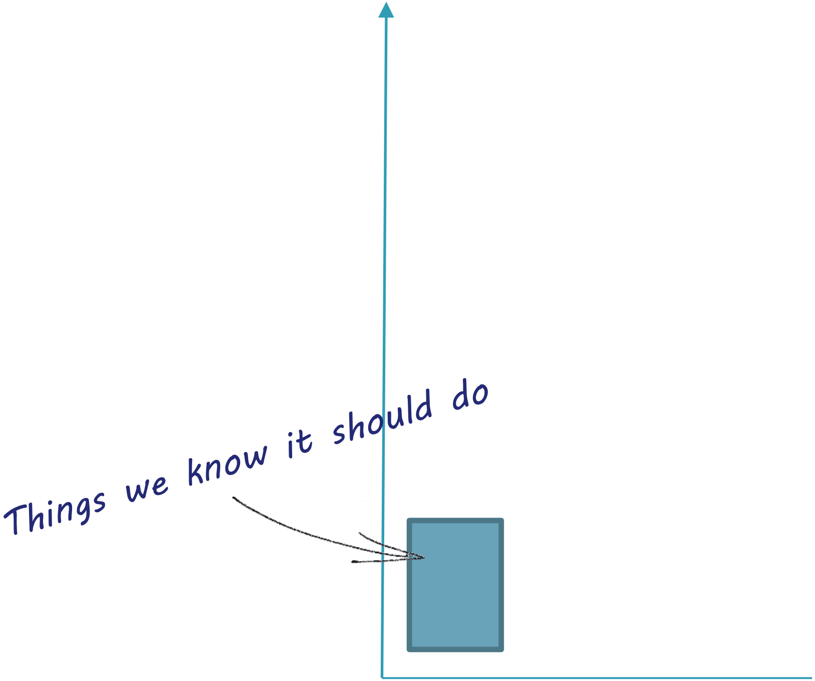
Panel discussion: Mind the gap!

Validating Products versus Requirements; Dis(covering) the Gaps

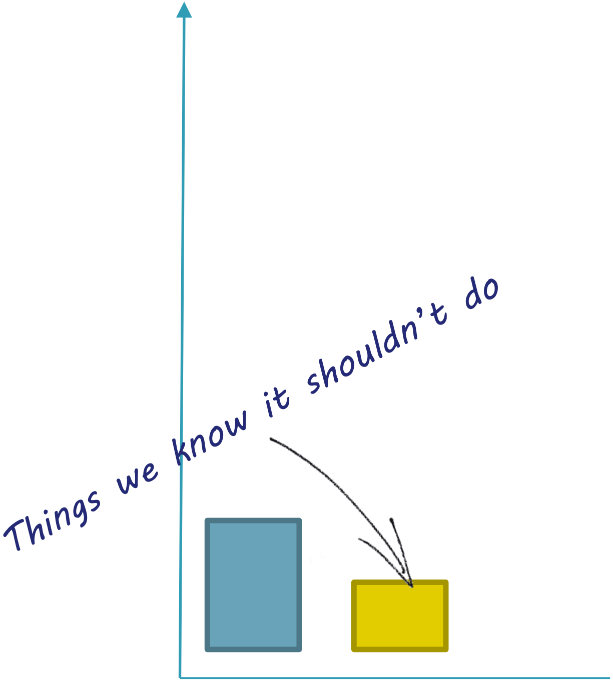
The problem with defining requirements

The problem with defining requirements

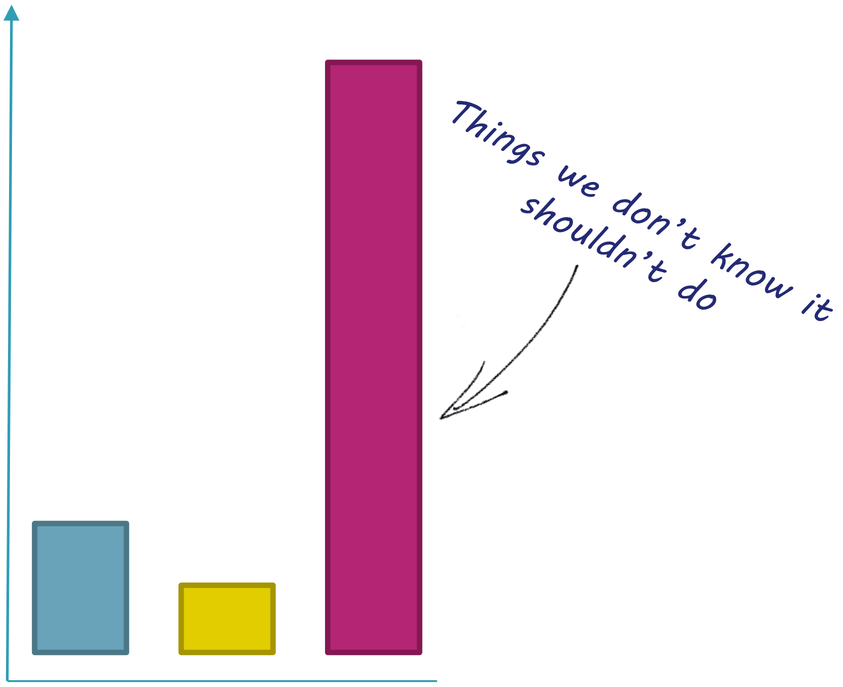
The problem with defining requirements



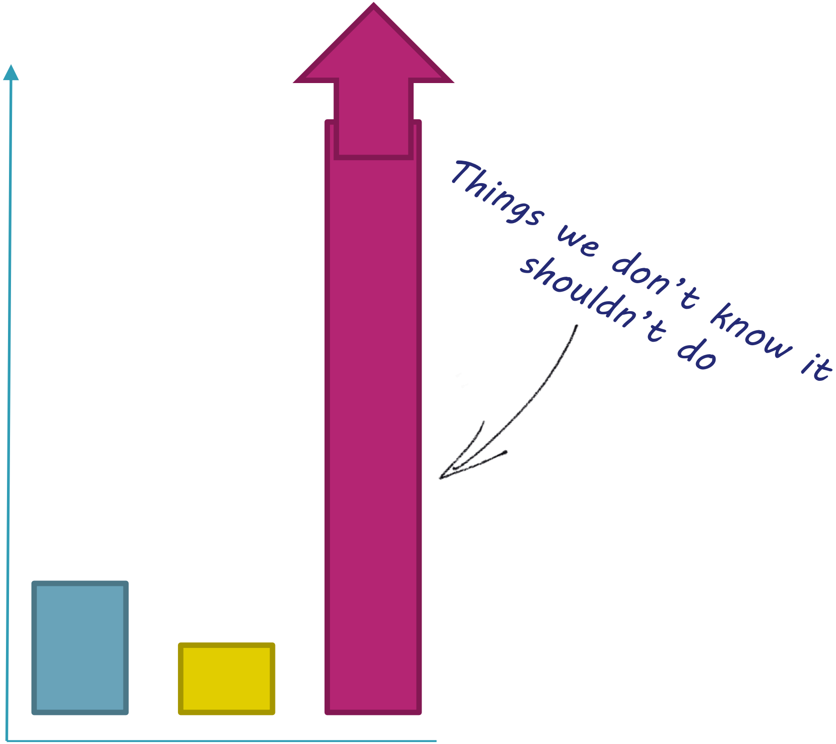
The problem with defining requirements



The problem with defining requirements



The problem with defining requirements



Unintended consequences

Unintended consequences

Unintended consequences

Seeing this does not always mean you are at risk.
You are at Risk!

The search for perfection

The search for perfection

The search for perfection

“The perfect is the enemy of the good”

The search for perfection

"Better a diamond with a flaw than a pebble without." - Confucius

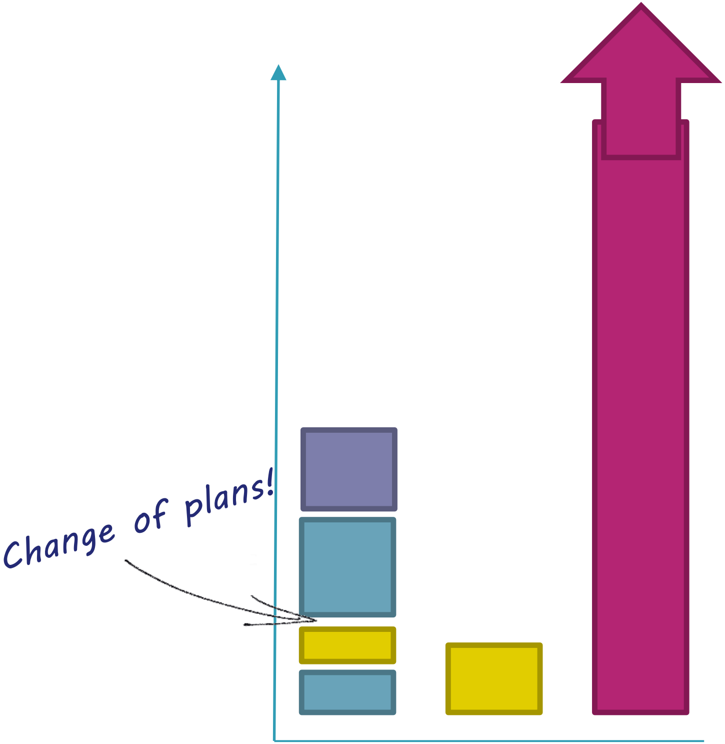
"The best is the enemy of the good." - Voltaire

"Striving to better, oft we mar what's well." - Shakespeare

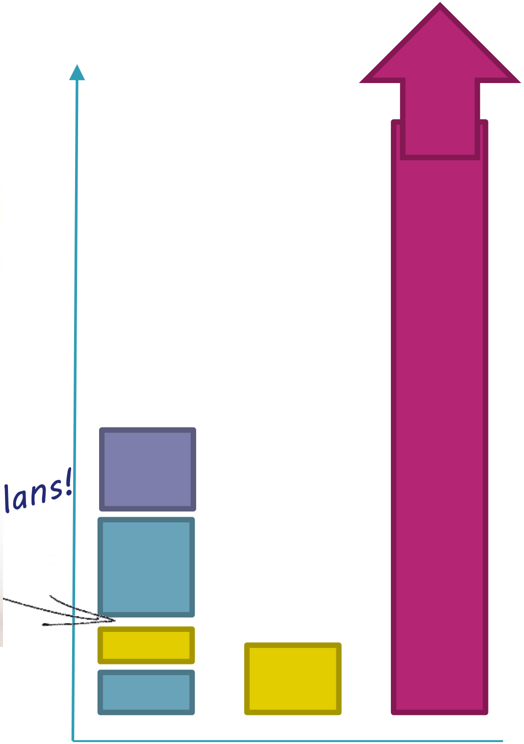
The commercial ‘creative process’

The commercial ‘creative process’

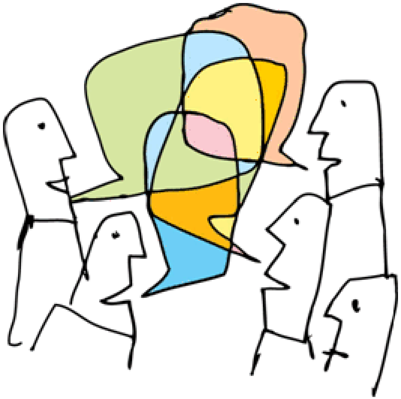
The commercial ‘creative process’



The commercial ‘creative process’



How can we possibly win?



How can we possibly win?

Can models implement software requirements?

Radek Kočí

Brno University of Technology, Faculty of Information Technology
Czech Republic

koci@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

ICSEA 2016, 21.-25.8.2016, Roma, Italy

Topic

Questions on modeling and implementation (maybe) of software functional requirements using formal methods.

Questions on the Requirements Specification

How to specify functional requirements?

- unrestricted natural language
- structured natural language
- predefined statement templates
- semi-formal specification language (ERD, DFD, UML, . . .)

What the requirements specification has to meet?

- it has to be readable and understandable for users
- the requirements has to be specified exactly (?)
- the specification has to be valid (how to do it?)

Valid Specification?

How to validate the requirements specification?

- inspections and reviews, evaluation at review meetings, . . .
- an animation of specifications
  ⇒ the need of executable form of the specification,
  e.g., Petri nets, state machines, Executable UML, . . .
- requirements verification through formal methods
  ⇒ the need of the formal specification,
  e.g., Petri nets, temporal logic, . . .

Formal methods

- provide higher precision and richer forms of analysis
- (but) are usually harder to use and less widely applicable

Executable/Formal Models?

Does the model adequately reflect the original specification or the developed system?

- how to create valid formal models from the specification?
- is it possible to specify requirements using formal models directly? (but it has to be still readable and understandable for users)
- is it possible to develop the system using models?

Answers (?)

How to create valid formal models from the specification?

- it is difficult
- model transformations are too complicated

Is it possible to specify requirements using formal models directly?

- yes
- formalisms with clear syntax and semantics
- these formalisms have to be usable by developers having no power mathematical backgroud, e.g., some kinds of Petri nets

⇒ it is possible use simulation or formal methods to verify specifications

⇒ it is possible to validate the requirements immediately they are specified

Answers (?)

Is it possible to develop the system using models?

- yes (partially)
- it is needed to combine specification models with other ones including programming language ⇒ the code is part of models ⇒ models implement requirements
- for instance, use cases, Petri nets, DEVS, Smalltalk, Java, . . .
- it can be a problem for time-critical systems, the transformation or final implementation would be needed

Answers (?)

Tool support needed

- Renew (Hamburg): a combination of Petri nets and Java
- PNtalk (Brno): a combination of Petri nets, DEVS formalism, and Smalltalk (so far the experimental version only, the new release is awaited this year)
- both concepts are able to run Petri nets on embedded system as a control software

Thank you for your attention!