CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Agile Requirements Engineering

Eric Knauss, 2017-11-08

Guest Lecture: DIT045 H17

! These slides were modified after presentation:
Pictures were removed to reduce file-size.

Hi, nice meeting you! (Short CV)

- Born in Hattingen, Germany
- University in Dortmund and Hanover
- PhD in Hanover: **Improving Requirements Documentation based on Heuristics and Experience**
- Postdoc in Hanover: **Global Software Development**
- Postdoc in Victoria BC: **RE in Distributed Large-Scale Software Projects**

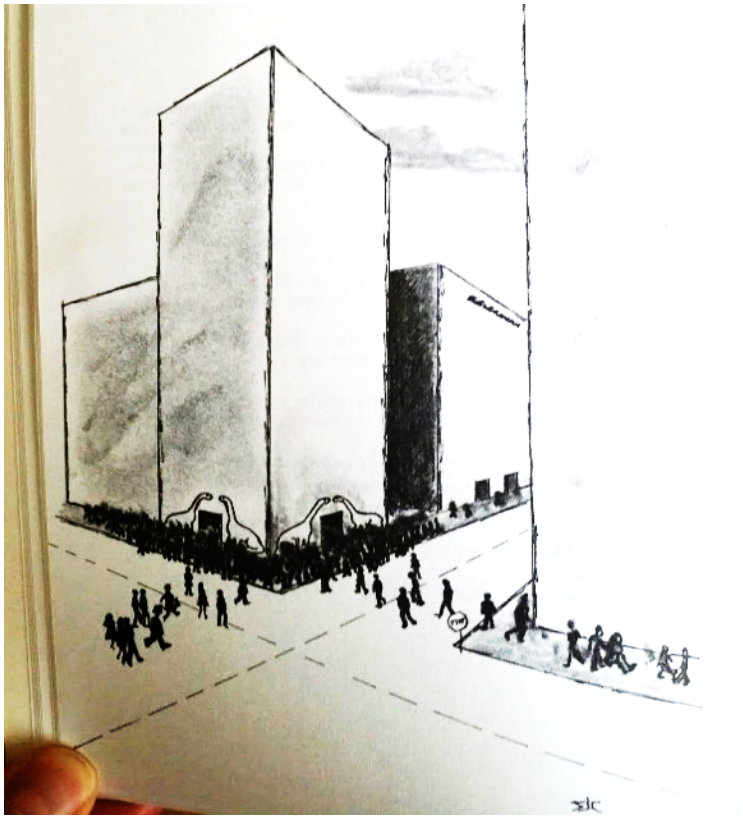
- **Topics**
  - **Requirements Engineering**
  - **Agile Methods**
  - **RE and Project Management**
  - **Experience and Knowledge Management**

Reminder

Requirements-writing exercise facilitated by the TAs in the afternoon.

- No, you don’t have to attend, and
- No, there’s no new material.
- But: it’s useful practice for your A1.

A Problem



At the heart of Gotham City's financial district stands the glistening new 73-story Brontosaurus Tower.

- Even though not yet fully occupied, the elevator service has been found woefully inadequate
- Some tenants have actually threatened to leave if the service is not improved, and quickly.

A few facts

1. Building primary houses offices (hours: weekdays, 9am to 5pm)
2. Nearly everyone using the building is associated in some way with the financial world
3. Occupants fairly uniformly distributed over the 73 floors – and so is the elavator traffic
4. The owner has invested heavily in advertising to fill the remaining office space.
5. Discouraging words spread like lightning in the tight little world of the financial district.

What is to be done about this situation?

Example taken from
Gause and Weinberg: Are your lights on? Dorset House Publ. 1982

A Problem (immediate ideas)

1. Speed up the elevator
2. Add elevators by cutting new shafts through the building
3. Add elevators by constructing outside shafts
4. Stagger working hours to spread the rush hour load over a longer period
5. Move occupants to different floors to reduce total passenger traffic within the building
6. Restrict the number of people entering the building
7. Replace existing elevators with bigger cars stretching two or three stories
8. Provide more services locally on each floor to reduce floor-to-floor traffic
9. Reschedule the elevators with special local and express arrangements, as needed

"The User"

1. Increase the rent, so fewer occupants will be needed to pay off the mortgage
2. Convince the occupants that Brontosaurus Tower is a terrific leisurely place to work because of the elevator situation
3. Convince the occupants that they need more exercise – which they could get by walking the stairs – by posting walking times and calorie consumption estimates over well-traveled routes
4. Burn down the building and collect the fire insurance
5. Sue the builder
6. Steal elevator time from the next-door neighbor

"The Landlord"

Not so good: rushed into solutions!
Instead: ask a few questions:
- Who has the problem?
- What is the problem?

A Problem

…is a difference between things as ***desired*** and things as ***perceived***.

A Requirement

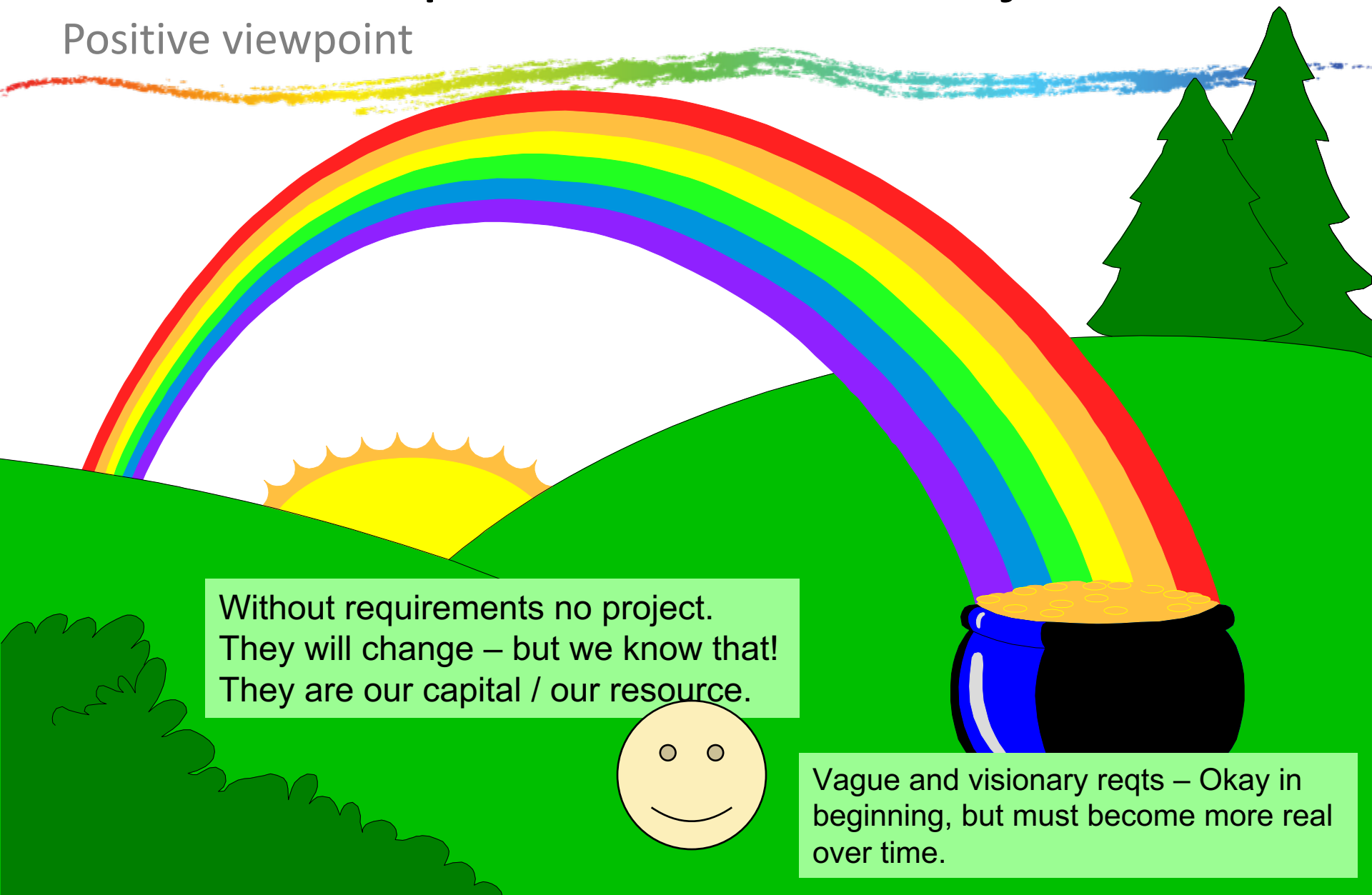
…is a condition or capability **needed** by a user to solve a ***problem*** or achieve an objective.

Requirement Engineering

…is a systematic approach to reduce the likelihood to develop the wrong solution (i.e., one that does not solve the problem).

Without requirements: No Project

Positive viewpoint



Without requirements no project.
They will change – but we know that!
They are our capital / our resource.

Vague and visionary reqts – Okay in beginning, but must become more real over time.

How do people develop software to solve problems?

Principle activities

Analysis

Understand the problem; derive requirements

Design

Sketch the key building blocks, decompose problem

Implementation

Develop software

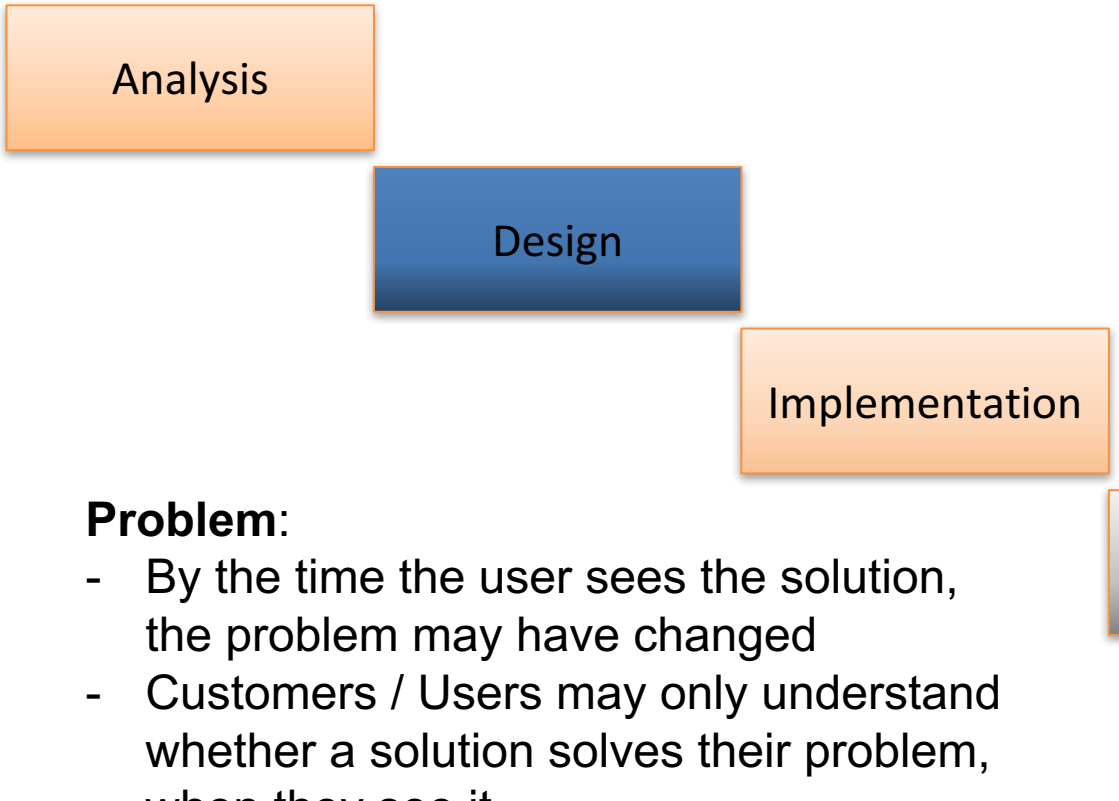
Testing

Make sure that software solves the problem

Deployment

Deliver software so that users can work with it

Classic way: Do it in this order

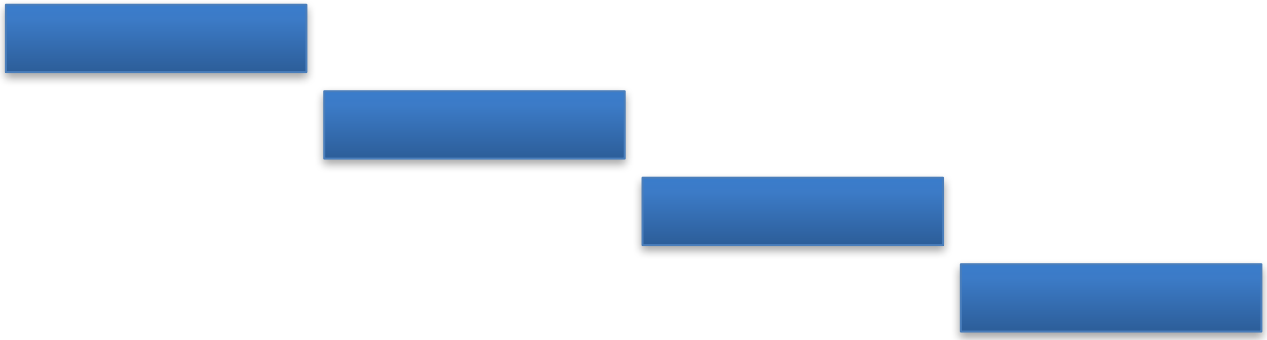


Waterfall
+ Simple
+ Clear handovers
- Hard to manage late change
- Long time-to-market

Problem:
- By the time the user sees the solution, the problem may have changed
- Customers / Users may only understand whether a solution solves their problem, when they see it

Systematic sequential development

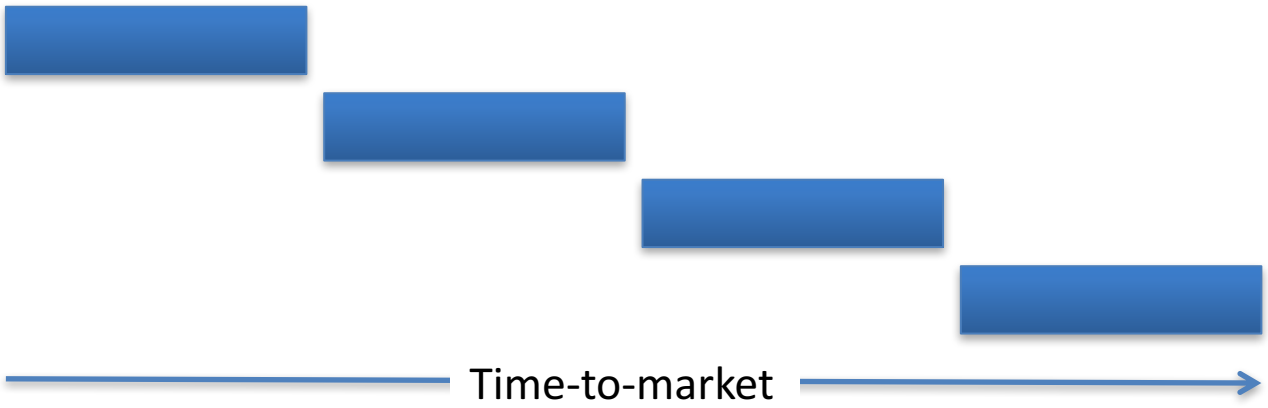
- Requirements
- Design
- Programming
- Test



- Advantages
  - Simple
  - Controllable
  - Cost efficient
- Problems
  - Time-to-market
  - What about change?

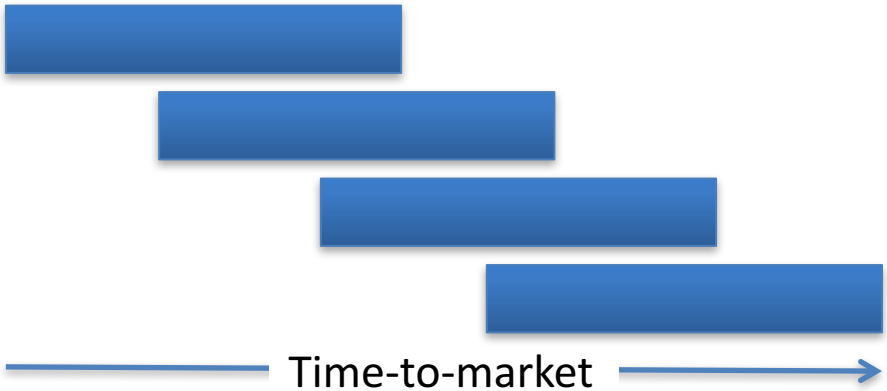
Towards concurrent development

- Requirements
- Design
- Programming
- Test



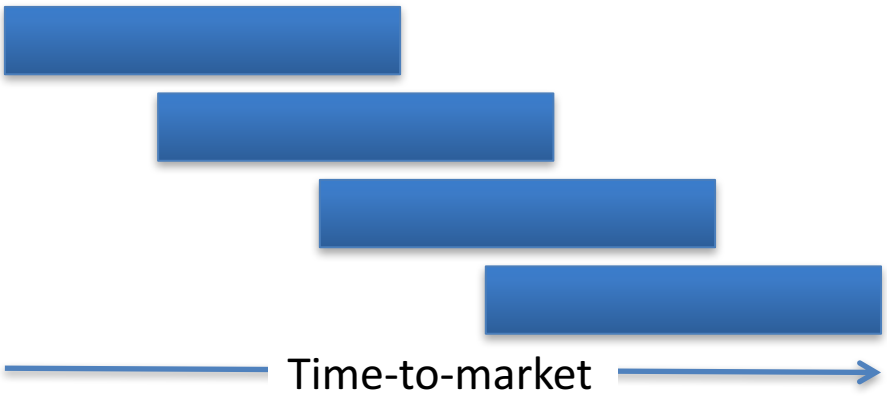
What can we do if time to market and robustness against late changes are more important than cost-efficiency?

- Requirements
- Design
- Programming
- Test



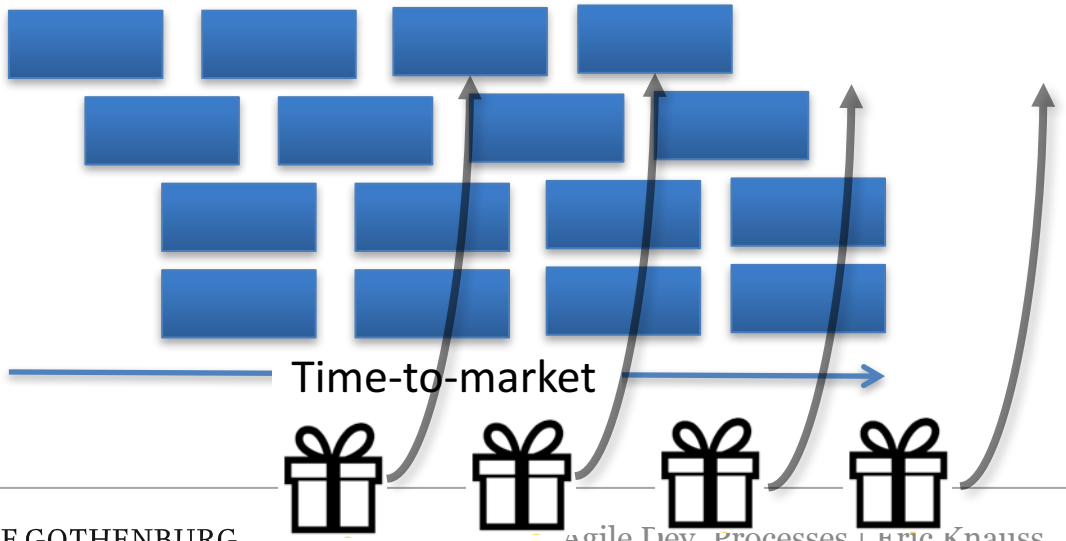
Towards Iterative/Incremental development

- Requirements
- Design
- Programming
- Test



Goal: Minimize risk, optimize learning from customer

- Requirements
- Design
- Programming
- Test



Agile Manifesto

http://agilemanifesto.org

- Began as a provocation: Plan-driven development did not safe the Software world…
- Now a very serious movement, well adapted in industry.
- There are a couple of established agile methods: How to integrate these values in everyday software development

What is agile? What is not?

- Agile – a compendium of ideas
  - Applied by number of methods
    (incl. XP, Scrum, Kanban, Lean Software Development)

- Core characteristics defined through
  - **Values**: General assumptions framing the agile view of the world
  - **Principles**: Core agile rules, organizational and technical
  - **Roles**: responsibilities and privileges of the various actors in an agile process
  - **Practices**: specific activities practiced by agile teams
  - **Artifacts**: tools, both virtual and material, that support the practices

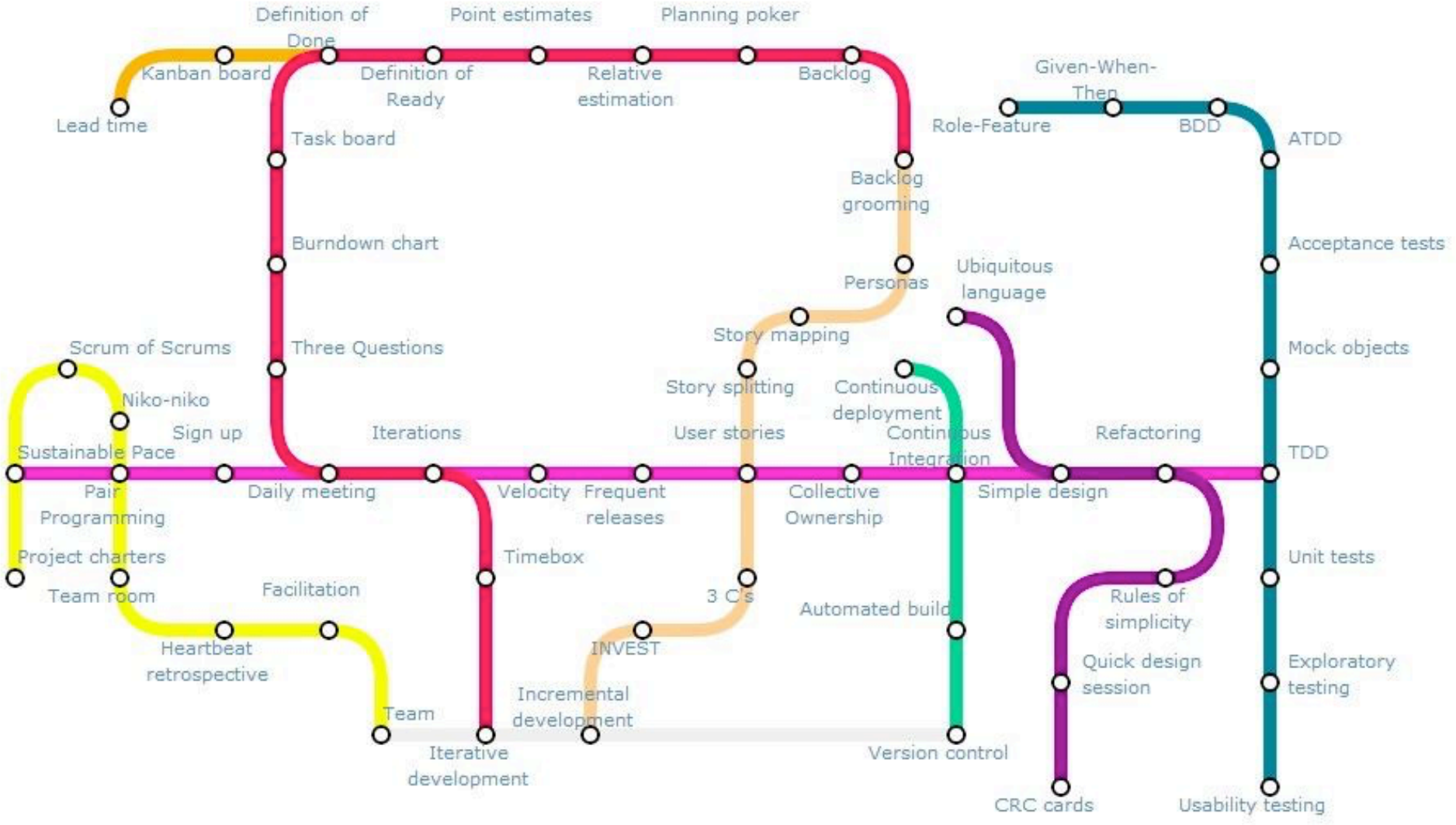
[Mey2014]

Agile Values

1. Redefined roles for developers, managers, and customers
2. No ”Big Upfront” steps
3. Iterative development
4. Limited, negotiated functionality
5. Focus on quality, understood as achieved through testing

[Mey2014]

Agile Practices



Lines represent practices from the various Agile "tribes" or areas of concern:

- Extreme Programming
- Teams
- Lean
- Scrum
- Product management
- Devops
- Design
- Testing
- Fundamentals

https://techblog.betclicgroup.com/wp-content/uploads/2013/12/agileSubwayM

Also check: http://guide.agilealliance.org

https://en.wikipedia.org/wiki/Scrum_(rugby)

Overview: Scrum

Scrum Principles

- Reflection
  - Stop and review product & process
- Self-correction
  - Based on reflection
- Visibility
  - Everything is visible (=known) for all stakeholders, e.g. plans, schedules, issues, …

Scrum practices

- Product backlog vs. Sprint backlog
- Sprint planning
  - Planning poker: estimate cost
  - ROI (Return on Investment): cost vs. benefit
- Retrospective
- Fixed sprint length
- Burn-down charts
- Daily scrum: no longer than 15min

SCRUM Practices

Hints

- SCRUM Meetings
  - SCRUM Master minds the time (2-3 min./Person)
    - Standup-meeting: faster
    - Replace status meetings – safe time
  - Important: Always same time and place!
    - (not important: where)
    - Daily / frequent meetings avoid long/quiet crisis
  - Content
    - What was done since the last meeting?
    - What is planned to be done before the next meeting?
    - Found obstacles? Write on whiteboard!
  - Useful: Share information and facilitate social aspects
  - Schedule further meetings to follow up on things (e.g. obstacles)

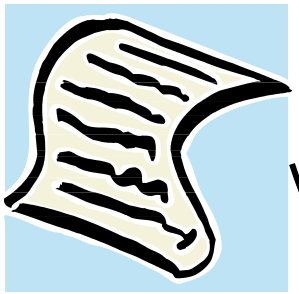
SCRUM Practices

Hints

- Sprint
  - During sprint: Autonomous team: *Pioneers*
    - No new requirements / no changes
    - No external influences
    - Only sprint goal
      - Fixed: Time (approx. 30 Tage), Cost (Developers etc.), Quality
      - Variable: Functionality
        - » Team can adjust details and scope of functionality based on the time-cost-quality frame and with respect to the sprint goal
  - Sprint can be cancelled
  - After Sprint: 4h Sprint-Meeting
    - Avoid long preparation (max. 2h)
    - Avoid slides
    - Often very informal
- Adjust SCRUM (longer Sprints, other Meetings…)
  - Okay, after being successful with the traditional setup
  - Only based on experiences – never without experience

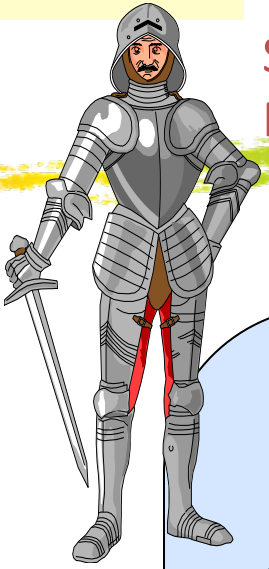
Scrum

Product Owner



Product Backlog (prioritized)

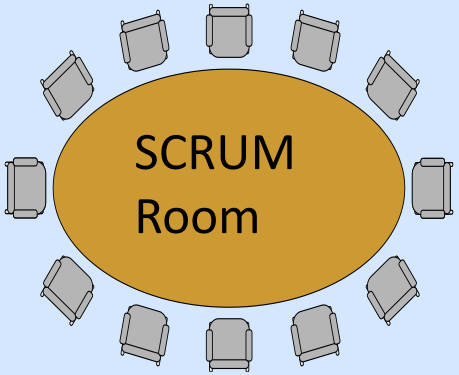
SCRUM Master



Others

SCRUM team 7+/-2

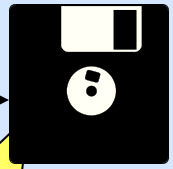
SCRUM Room



SPRINT Backlog

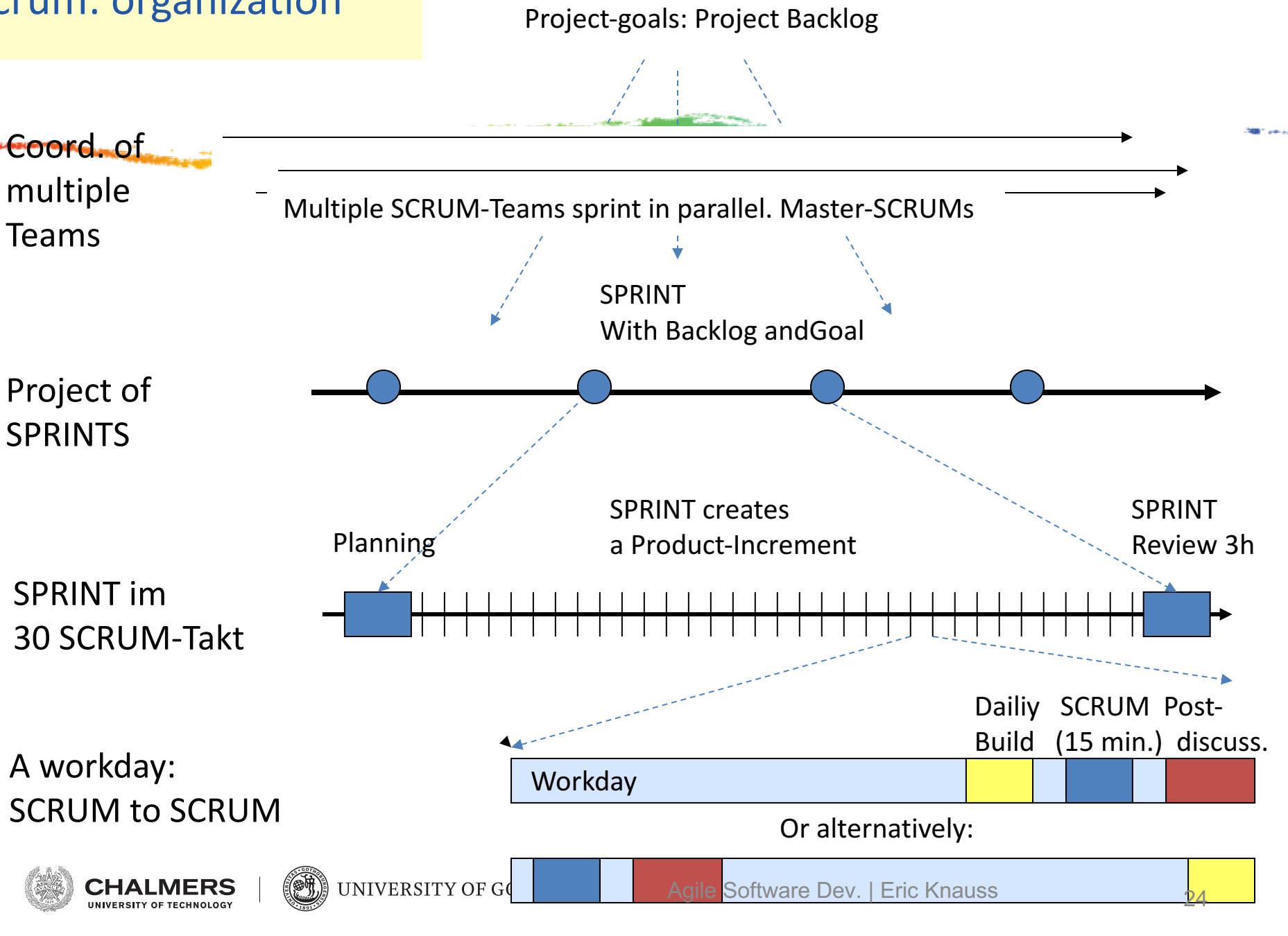
SPRINT: 30 days

Increment



Daily SCRUM 15 min.

Scrum: organization



RE in Scrum?

		Challenges
Who?		
When?		
What?		

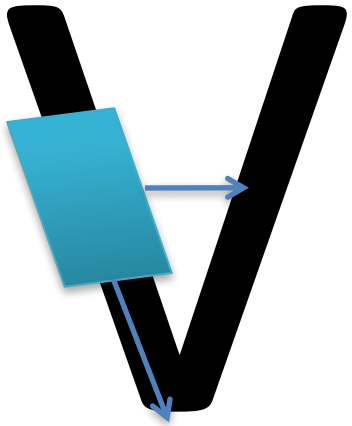
RE in Scrum?

		Challenges
Who?	Product owner, Team	How to manage distributed knowledge?
When?	PO: Breadth-First to prioritize Product Backlog	No guidelines (in agile) on how to do that. RE!
	Team: Just-in-time during Sprint	First contact for that is the PO who (hopefully) establishes contact to customer/user where possible
What?	Mainly user stories	Where do user stories come from? What about quality requirements?

Requirements now and then

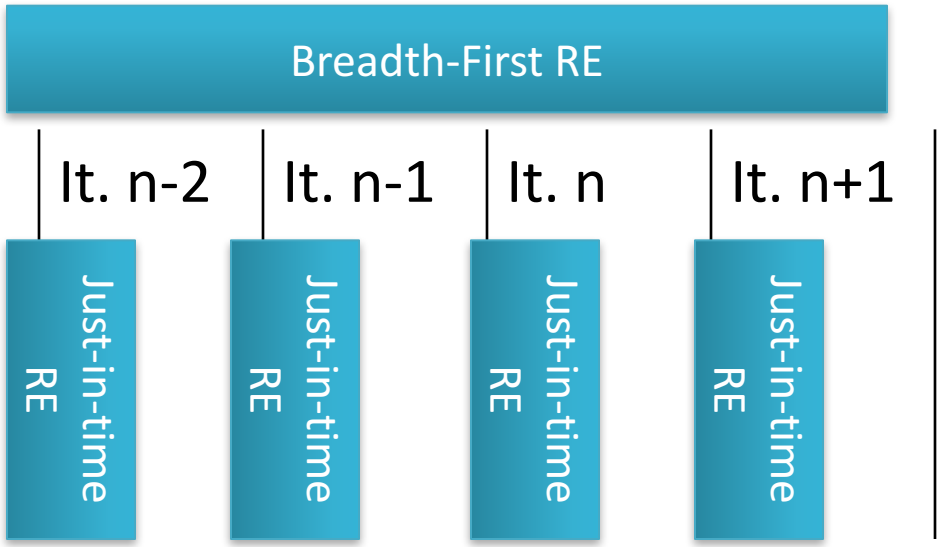
= “Stakeholder needs/properties a system should fulfill/exhibit”

Then



Requirements a “waterfall phase” with specialists

Now



Requirements are everybody’s responsibility

→ Knowledge Management Problem

Overview of Agile RE from Research Perspective

The term ‘‘agile RE’’ is used to define the ‘‘agile way’’ of planning, executing and reasoning about requirements engineering activities.

[RCB2010]

Agile RE practices from literature

- **Face-to-face communication**
- User stories
- Iterative requirements
- **Extreme Requirements prioritization**
- Change management
- Cross-functional teams
- Prototyping
- Test Driven Development
- Review meetings and acceptance tests
- **Managing requirements change through constant planning**

Overview of Agile RE from Research Perspective

The term "agile RE" is used to define the "agile way" of planning, executing and reasoning about requirements engineering activities.

[RCB2010]

Agile RE practices from literature

- **Face-to-face communication**
- User stories
- Iterative requ
- **Extreme Requirements prioritization**
- Change management
- Cross-functional team
- Prototyping
- Test Driven Developm
- Review meetings and
- **Managing requirements change through constant planning**

Facilitates customer involvement
Challenges in establishing trust between customer and developer

Done at each step of the cycle
Unstable architecture

Minimal documentation
Inadequate architecture

Table 7
Summary of challenges of traditional RE resolved by agile RE practices.

Source: [ISM+2015]

No.	Challenge	Practice	Description
1.	Communication issues (Bjarnason et al., 2011a; Carlson & Matuzic, 2010)	Frequent face-to-face meetings (Bang, 2007; Sillitti et al., 2005)	Agile RE promotes regular interaction with customer and among teams. It is the predominant method for eradicating communication gaps
		Collocated teams (Highsmith & Fowler, 2001)	Agile principles prefers collocated teams for better communication and collaboration
		Onsite customer (Cao & Ramesh, 2008; Lundh & Sandberg, 2002; Pichler et al., 2006)	Customer and development teams should be located in the same place to enhance informal communication, to enable timely feedback, to facilitate agreement, to develop ownership, and to create a sense of responsibility
		Alternate customer representations (Bjarnason et al., 2011a; Fraser, Mellon, Dunsmore, & Lundh, 2001; Hoda, Noble, & Marshall, 2011)	In industry, having business representatives to be present at the development site is expensive and impossible at times. RE alternatives such as proxy customers can serve the same purpose
		Cross-functional agile teams (Bjarnason et al., 2011a)	Cross-functional agile teams aid in the clarification and understanding of requirements
		Integrated RE process (Bjarnason et al., 2011a)	Locating RE process closer to development activities enhances developer's understanding and reduces communication lapses
2.	Overscoping (Bjarnason et al., 2011a)	One continuous scope flow (Bjarnason et al., 2011a)	In agile methods, developers receive a list of features that are constantly prioritised by the customer. Thus, the chance of having to repeat allocation in projects is reduced
		Gradual detailing (Bjarnason et al., 2011a)	Gradual detailing of requirements helps to reduce overscoping and contributes to a feasible scope
		Cross-functional teams (Bjarnason et al., 2011a)	The team can focus more on important features when sharing responsibilities and working closely together in a cross-functional structure
3.	Requirements validation (Carlson & Matuzic, 2010)	Requirements prioritisation (Racheva et al., 2010)	Customer continues with prioritisation requirements in every iteration; thus, less important requirements remain on hold
		Prototyping (Cao & Ramesh, 2008; Ramesh et al., 2010)	Prototyping helps in providing the customer with a blueprint of the product, and therefore helps in validating the requirements
4.	Requirements documentation (Bjarnason et al., 2011a)	User stories (Bjarnason et al., 2011a; Carlson & Matuzic, 2010)	User stories are precise and provide to-the-point explanation of user demands, prevent the need for maintaining long SRS documents as well as constant updating and traceability
		Face-to-face communication (Cao & Ramesh, 2008; Ramesh et al., 2010)	More face-to-face communication reduces ambiguities and the need for maintaining long documents
5.	Rare customer involvement (Carlson & Matuzic, 2010)	Requirements prioritisation by the customer (Racheva et al., 2010)	Prioritisation of the requirements for all iterations ensures, to a large extent that the customer goals will be met

Source: [ISM+2015]

Table 8
Summary of challenges of agile RE.

Challenge	Description	Impact	Solutions
Minimal documentation (Cao & Ramesh, 2008)	User stories and product backlogs are the only documents in agile methods (Zhu, 2009)	Traceability issues (Zhu, 2009)	
Customer availability (Ramesh et al., 2010)	Availability of customer for requirements negotiation, clarification and feedback	Increase in rework	Surrogate customers (Ramesh et al., 2010)
Inappropriate architecture (Ramesh et al., 2010)	Inadequate infrastructure can cause problems during later project stages	Increase in cost	Code refactoring (Berry, 2002)
Budget and time estimation (Cao & Ramesh, 2008)	Initial estimates of time and cost are changed substantially by a change in requirements in subsequent stages	Project delays	Frequent communication
		Over-budgeting	Accurate modelling of user story
Neglecting non-functional requirements (NRFs)	User stories only satisfy system/product features	System security, usability, performance at stake	NRF modelling approach (Farid & Mitropoulos, 2012b). The NORMATIC tool (Farid & Mitropoulos, 2012a)
Customer inability and agreement (Daneva et al., 2013; Ramesh et al., 2010)	Incomplete domain knowledge and in consensus among customer groups	Increase in rework	Creation of delivery stories to accompany user stories (Daneva et al., 2013)
		Increase in cost	Frequent communication Iterative RE (Ramesh et al., 2010)
Contractual limitations (Cao & Ramesh, 2008)	Fixed-price contracts do not allow changes	Increase in cost	
Requirements change and its evaluation	To find the consequences of requirements change	Increase in work delay	RE-KOMBINE framework (Ernst et al., 2013)

State of the Art

	Message	Reference
-	Difficult to manage reqts. in large-scale agile.	[SKV2010]
-	Hard to implement efficient RE	[LSA2011, WS+2013, CC2008]
+	Agile practices support RE	[RCB2010,HP+2017, BWR2011]
-	Lack of empirical works on *solutions* and *RE in relation to agile*	[HD+2015,HP+2017, IS+2015]
+	Agile practices resonate very well with developers	[SA2008]
+	Success in scaling them up	[SA2008]
+	Possible to implement in large organizations	[LM+2004]
+	Good support for knowledge sharing	[LS+2013]
-	Challenging to coordinate and to integrate into system development	[EHS2014]

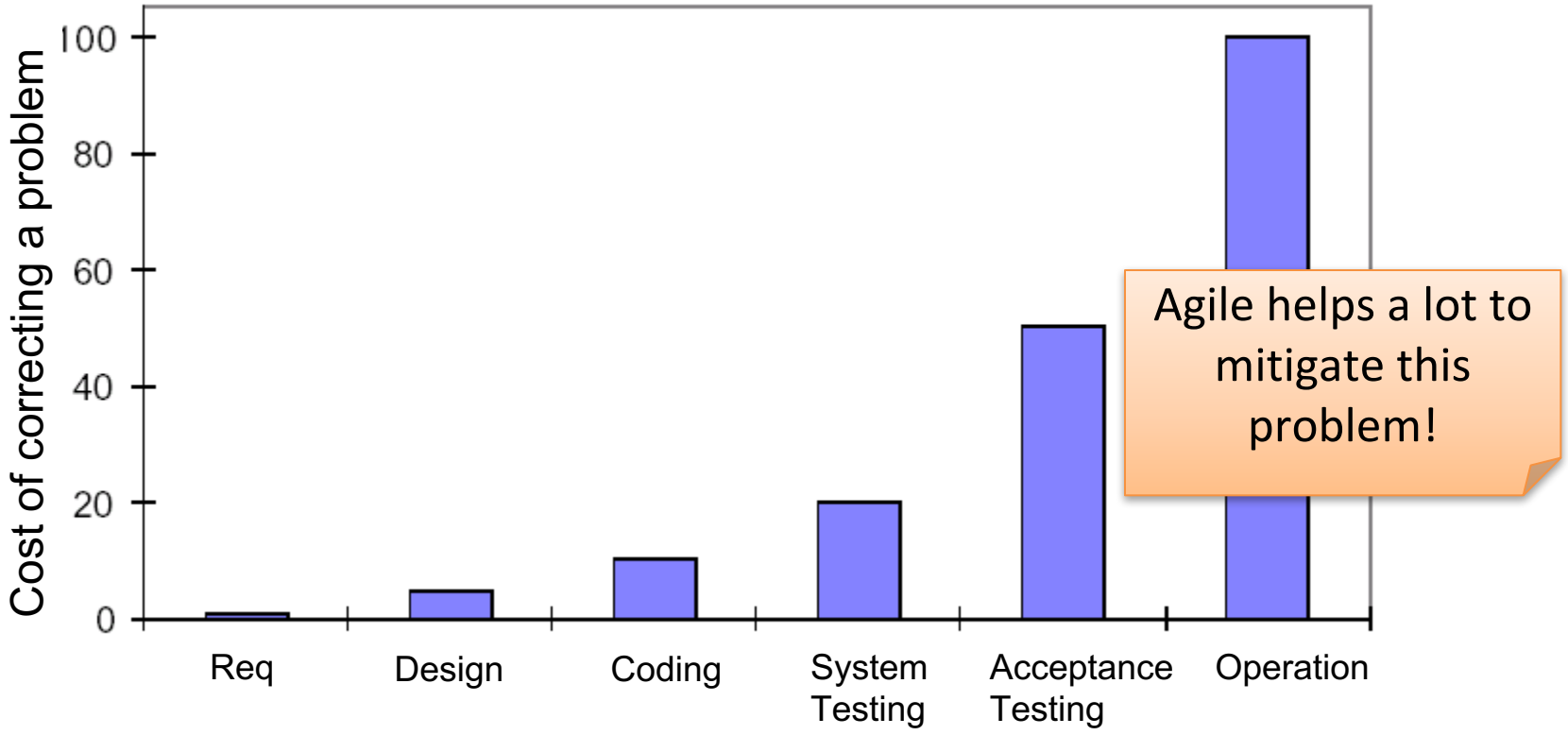
State of the Art 2

	Message	Reference
-	Definition of "Agile RE" is weak. Challenges: **Customer representatives**, **prioritiz., techn. debt**	[PH2017]
+ -	Requirements flow in large-scale agile: Increased **flexibility**, planning **efficiency**, communication efficiency. Problems with **over commitment**, **organizing system-level work**, and growing **technical debt**.	[HP+2017]
+	Agile methods can address some classic communication challenges in RE, but hard to ensure sufficient competence in XFT	[BWR2011]
-	Agile RE risks**: neglect quality reqts, customer inability**	[RCB2010]
±	Agile RE: 8 new challenges, 17 challenges of traditional RE solved	[IS+2015]
+	Traditional RE and agile **go well together**. Only difference: opinion about amount of documentation needed	[PEM2003]
-	RE and Agile **do not go well together**: Scenario focus and rejection of upfront X biggest damage that agile has caused to world	[Mey2014]

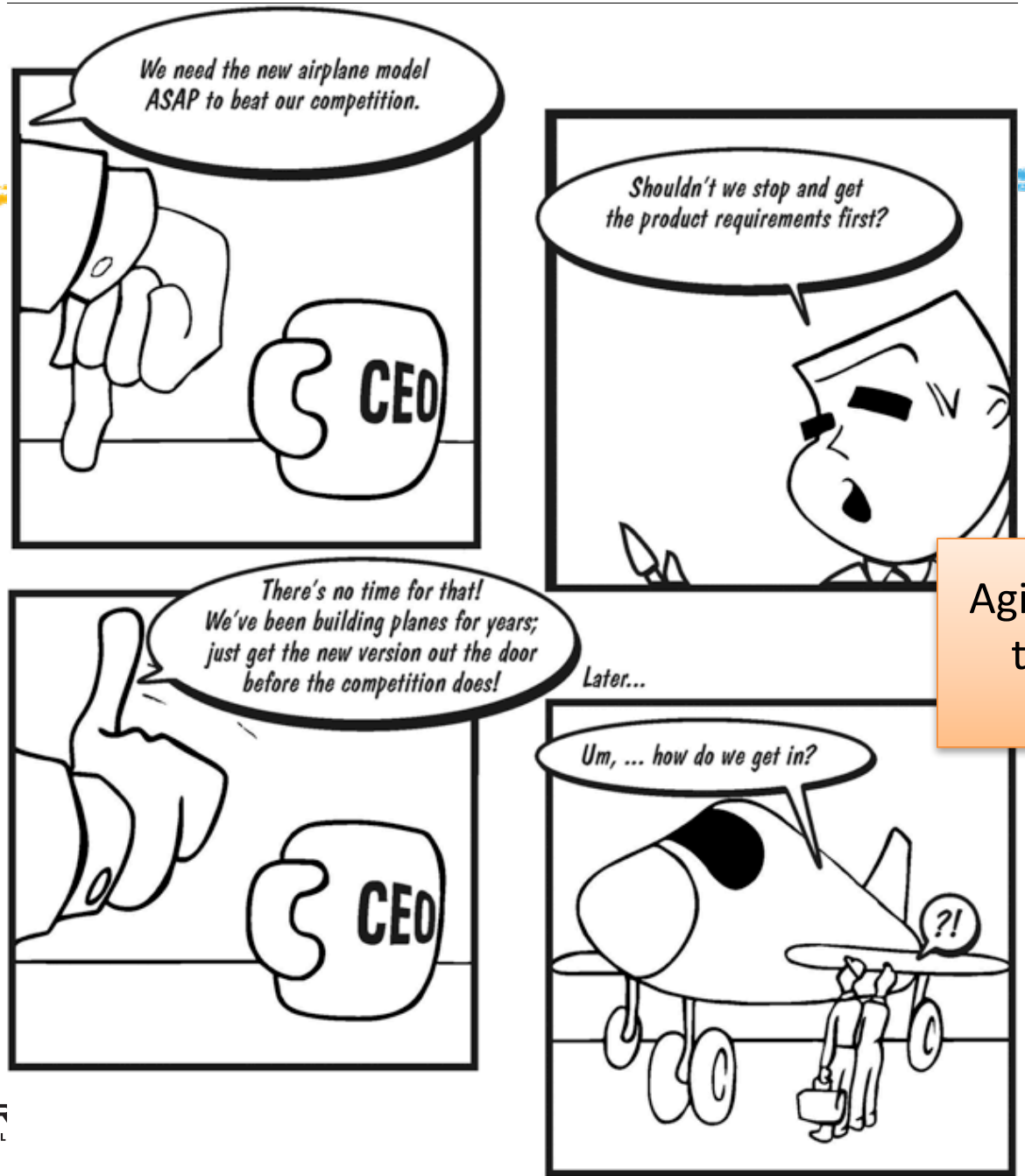
Useful, if applied in a lightweight way

RE FROM AN AGILE PERSPECTIVE

Economic consequences of requirements engineering problems

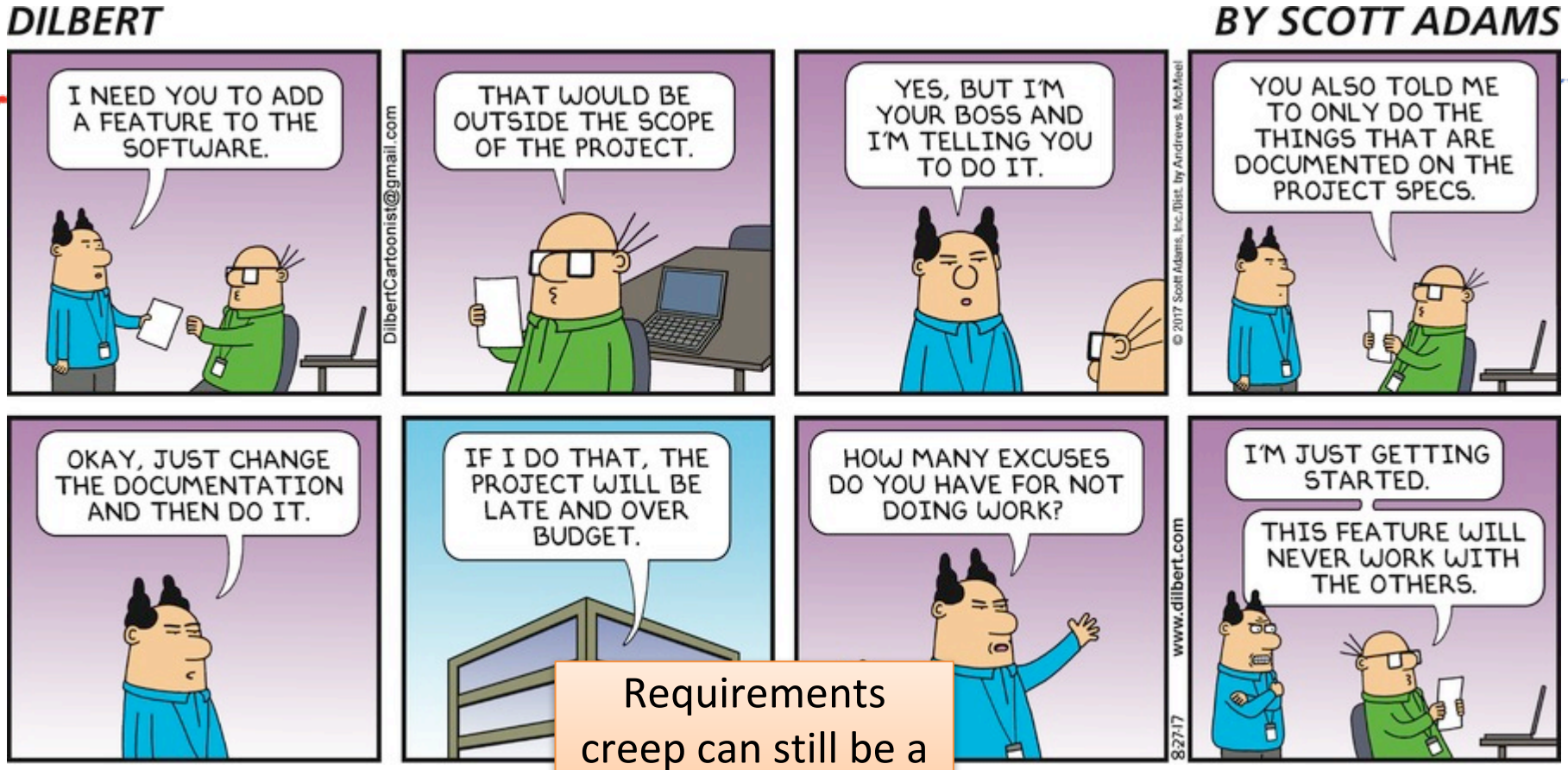


[Davis, 1992]



Agile can increase this problem!

Rushing into solutions

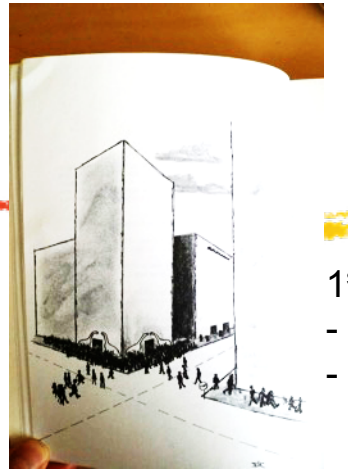


http://dilbert.com/strip/2017-08-27

Requirements creep can still be a problem in Agile.

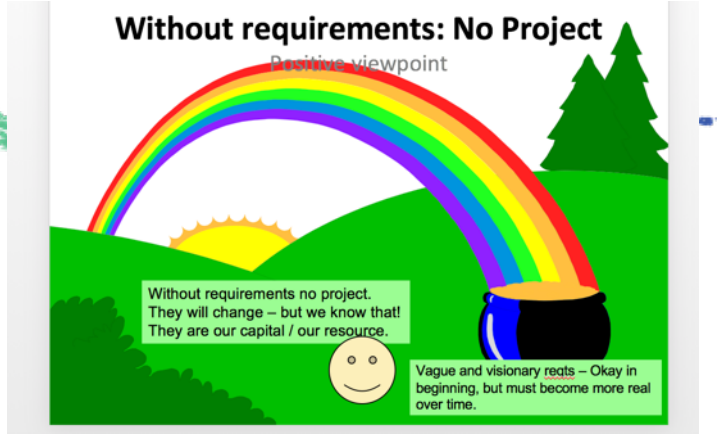
Lack of specification may increase feature interaction problem for complex systems

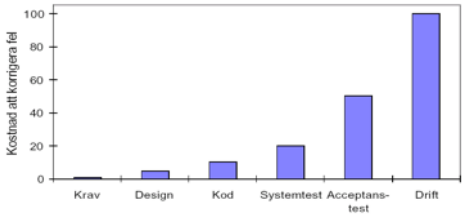
Requirements creep and Feature interaction



1st step of solving the problem:

- Understand what is the problem
- Understand who has the problem





Problems become more expensive the longer they are hidden

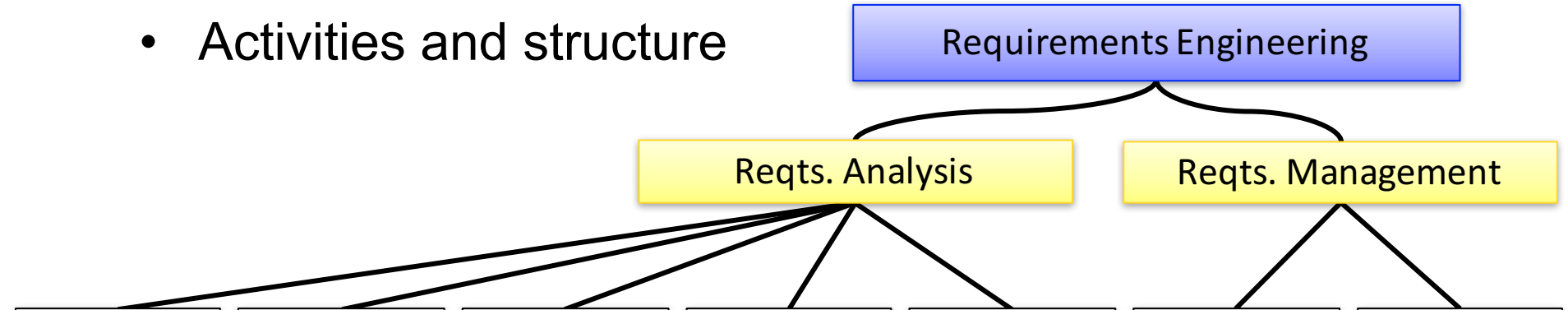
Rushing into solutions usually a bad idea

Requirements creep!
Feature interaction!

And many more…

What is Requirements Engineering?

- Activities and structure



- Identify Stakeholders and other Reqt src
- Capture needs, ideas, etc.
- Goal analysis

- Identify requirements
- Structure / refine
- Operationalize and make concrete

- Identify dependencies
- Identify + solve inconsistencies
- Prioritize

- Document reqts, current state, assumpt.
- Create baseline
- Administrative attributes

- Correct requirements?
- Requirements correct?

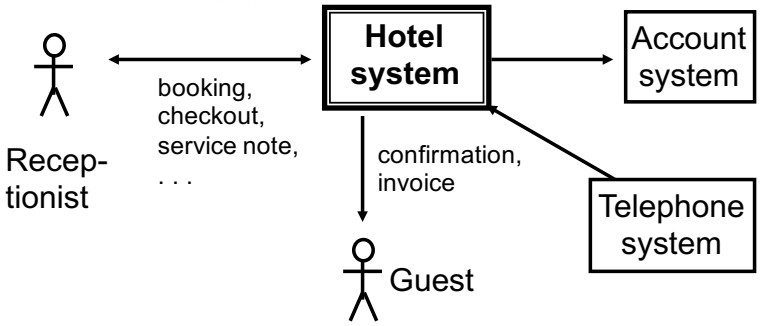
- Manage change requests
- Manage versions of reqts
- Propagate changes

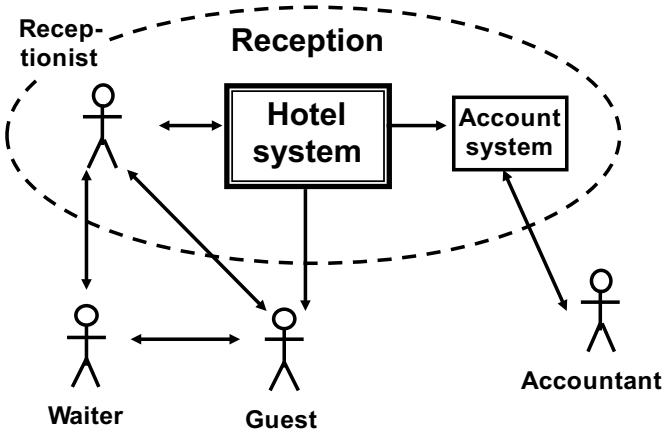
- Within reqts
- Pre-Tracing (reqt to src)
- Post-Tracing (reqt to change set, test)

Src: DaimlerChrysler, Dagstuhl-Seminar 1998

What is requirements engineering (2)?

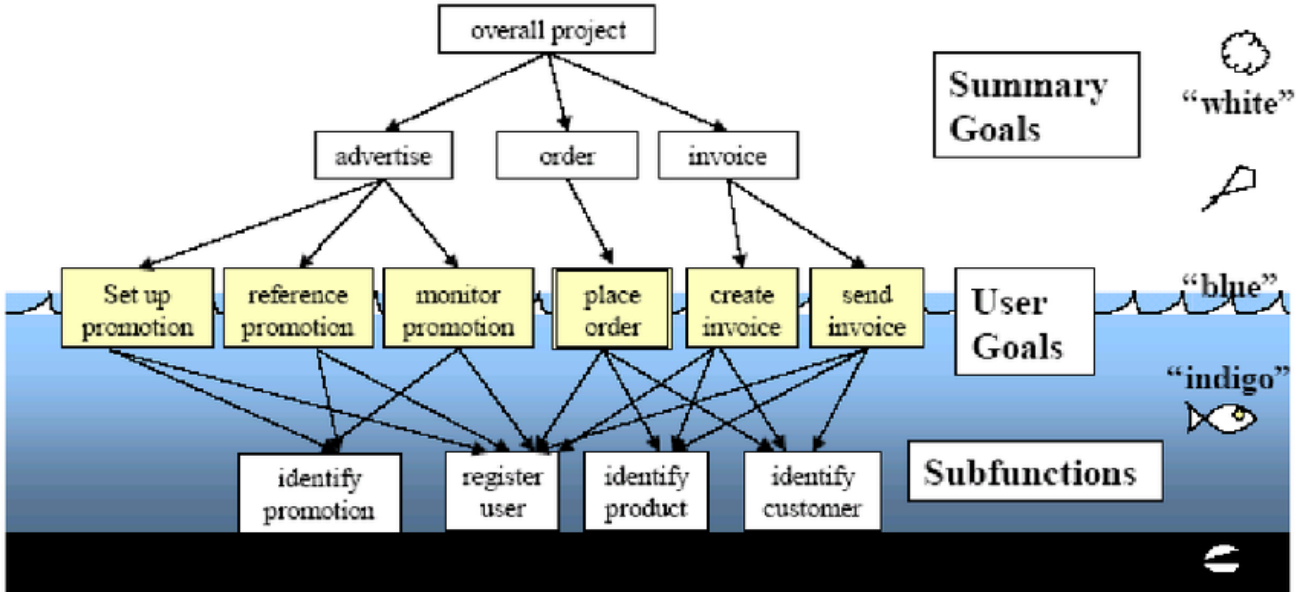
- Domain requirements
- User requirements
- System requirements





What is requirements engineering (3)?

- Abstraction levels



[Coc2001]

[GW2006]

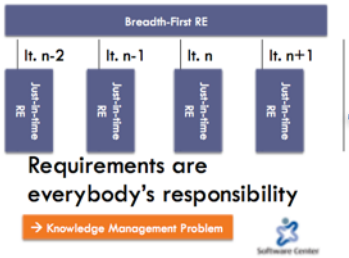
RE

When?

Then

Requirements a "waterfall phase" with specialists

Now



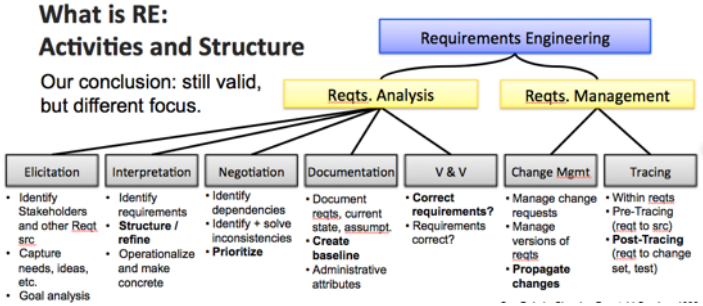
Requirements are everybody's responsibility

→ Knowledge Management Problem

What?

What is RE: Activities and Structure

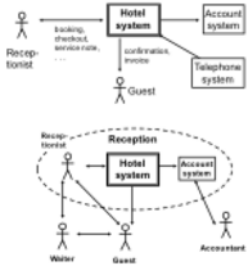
Our conclusion: still valid, but different focus.



Perspective?

What is RE 2:

- Domain requirements
- User requirements
- System requirements



Abstraction?

What is RE 3: Abstraction Levels

Do you need requirements engineering?

Requirements Eng (2006) 11: 1–3
DOI 10.1007/s00766-004-0206-4

VIEWPOINTS

Alan M. Davis · Didar Zowghi

Good requirements practices are neither necessary nor sufficient

Do you need requirements engineering?

Context	Agile	RE
Startup	(-) MVP and Learning over process!	(+) No. 1 problem: No clear problem to solve
Small mobile/web app		
Large-scale software dev		
Large-scale system dev		

Requirements Eng (2006) 11: 1–3
DOI 10.1007/s00766-004-0206-4

VIEWPOINTS

Alan M. Davis · Didar Zowghi

Good requirements practices are neither necessary nor sufficient

Do you need requirements engineering?

Context	Agile	RE
Startup	(-) MVP and Learning over process!	(+) No. 1 problem: No clear problem to solve
Small mobile/web app	(+) Ideal case	(-) Agile RE practices sufficient
Large-scale software dev	(+) But hard to implement, see SAFe and LESS framework	(+) Need to align work of several teams
Large-scale system dev	(?) Very hard to implement, but promises huge gains. Hardware, mechanics cannot be developed agile	(+) Need to align work of different domains, suppliers, etc.

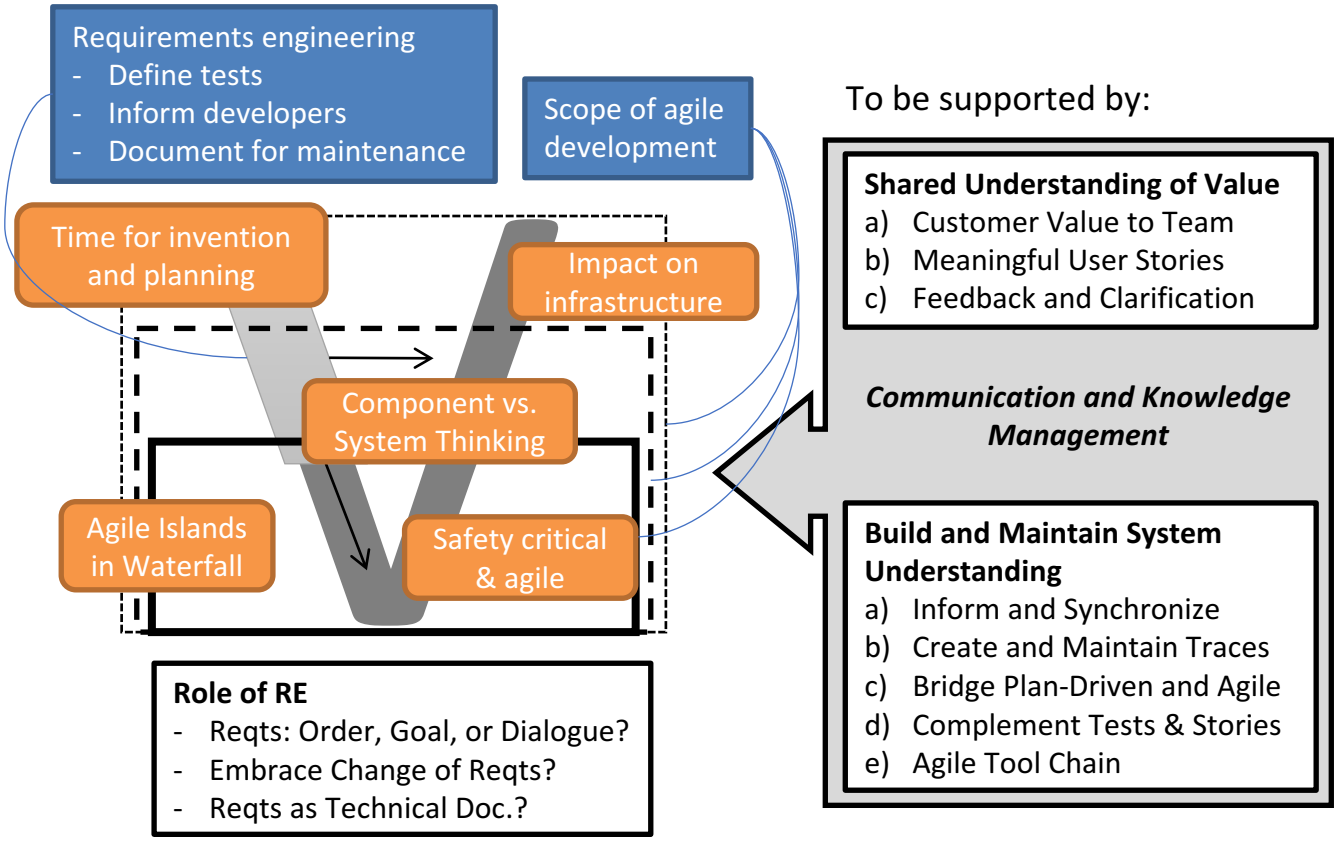
Requirements Eng (2006) 11: 1–3
DOI 10.1007/s00766-004-0206-4

VIEWPOINTS

Alan M. Davis · Didar Zowghi

Good requirements practices are neither necessary nor sufficient

Challenges with RE in Large-Scale Agile



Kasauli, R.; Liebel, G.; Knauss, E.; Gopakumar, S.; Kanagwa, B.: Requirements Engineering Challenges in Large-Scale Agile System Development. Submitted to RE conference, 2017

Reminder

Requirements-writing exercise facilitated by the Tas in the afternoon.

- No, you don’t have to attend, and
- No, there’s no new material.
- But: it’s useful practice for your A1.

Thank you!

- Interested in these topics?
  - Bachelor thesis?
  - Projects?
  - Collaboration?

- Want to learn more?
  - eric.knauss@cse.gu.se
  - @oerich
  - oerich.wordpress.com

Thanks!
Eric Knauss
eric.knauss@cse.gu.se
@oerich
oerich.wordpress.com