ETSN15
Requirements Engineering

Lecture 8:

Validation [Lau:9] & Inspections [INSP]

Tentafrågeupplägg

Agile RE [AGRE + ATCR]

This lecture is input to your current project task:
To develop your Validation Checklist for the 'customer' validation efforts during next week.
Work on this at exercise session.

Elizabeth Bjarnason
Björn Regnell
http://www.cs.lth.se/ETSN15

How will you do requirements validation in your project?

- Inspections [INSP]?
- Tests: usability testing, prototyping, model-based simulations?

Requirements validation

Purpose

- To make sure that we have elicited and documented the right requirements in a good way

”Will we build the right system with these requirements?”

Methods

- Inspections [INSP]
- Tests, e.g. usability testing, prototypes, model-based simulations
- Mathematical proofs

Requirements Validation through tests

Different types of dynamic validation:

- **Manual ”simulation” (walk-through) based on scenarios/use cases/task descriptions**
- **Paper prototypes or “mock-ups”**
- **Executable prototypes**
- **Pilot tests**

Important steps:

- **Choose suitable test approach, environment, etc.**
- **Choose who will do the testing**
- **Create & Run test cases**
- **Document problems**
- **Fix problems**
- **Consider: How to avoid problems in the future?**

Inspections [INSP]

Described already by
M.E. Fagan, IBM, early 70-ies

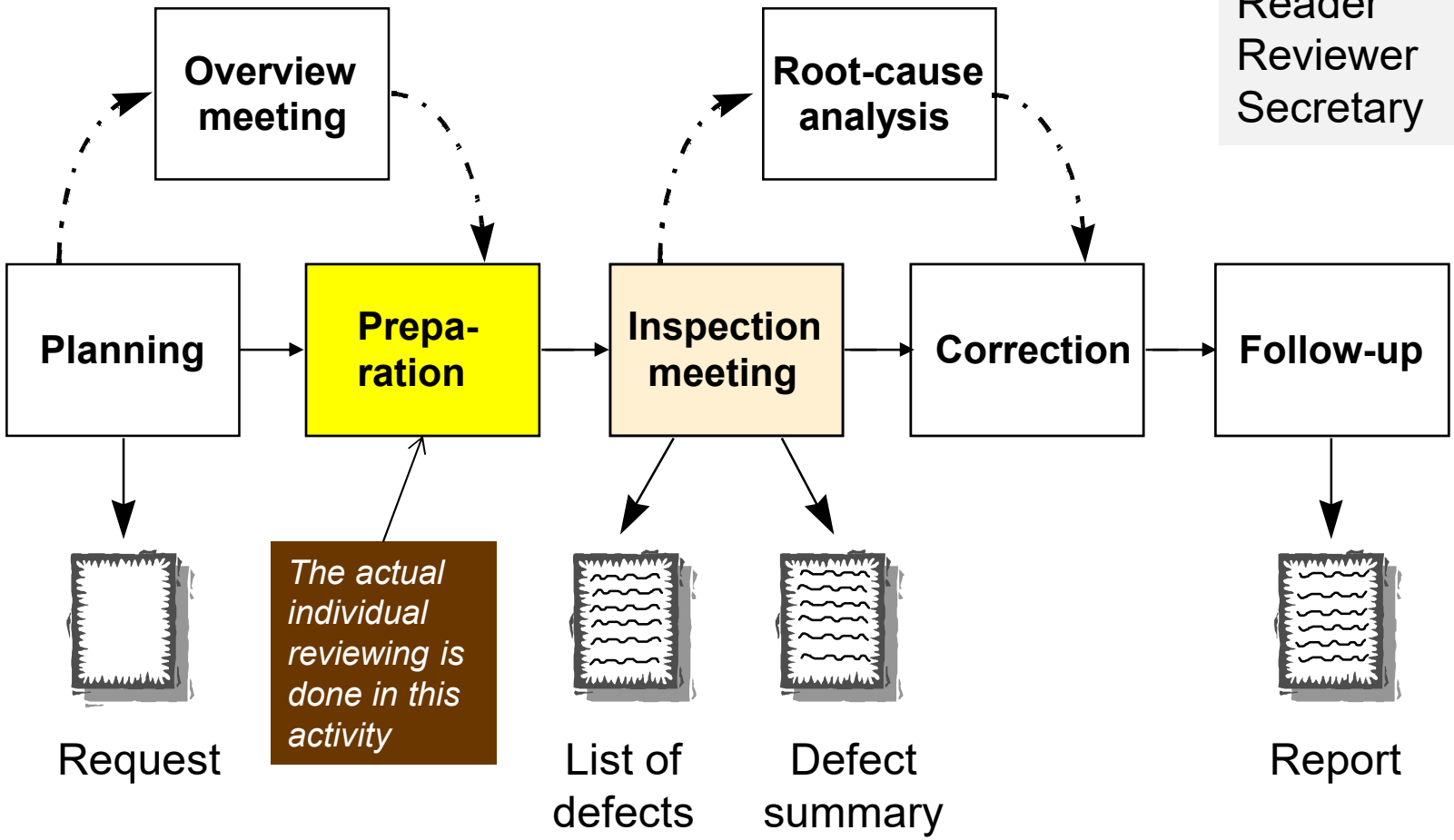
- systematic assessment
- documents inspected by others to **detect defects**

General objectives of inspection methods:

- Defect detection
- Knowledge dissemination
- Team building
- Decision-making

The inspection process [INSP]

Roles:
Author
Moderator
Reader
Reviewer
Secretary



Different methods to detect defects (reading techniques)

Ad hoc

- To your best ability (no specific guidelines)

Checklist

- A list of questions or check items direct the review

Perspective-based reading

- Different reviewers inspect from different perspectives and their findings are combined:
  e.g. user, designer, tester – perspectives,
  or from the perspective of different tasks/use cases

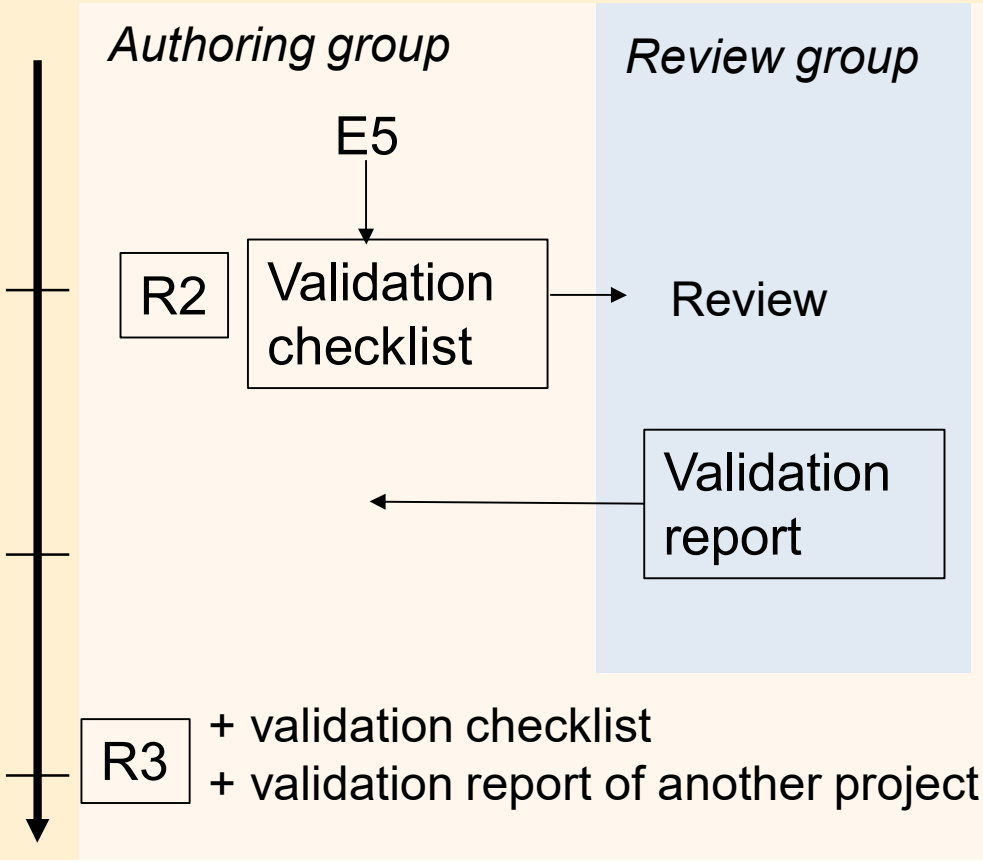
N-fold inspection

- N independent groups run inspection process in parallel

Course Project: Validation of R2 (in W6)

- Consider how to maximize value of review
- Prepare by providing the review group with a **Validation Checklist** suitable for your project **(Exercise 5!)**
- **Validation Report (**by review group) should contain relevant and useful issues ranked by criticality

See project description



Your two roles in validation

- As **author** make a useful **checklist**
- As **reviewer** make a useful **validation report**

2021 VT/Spring

Home

Modules

Assignments

People

Validation check-list

Due Sunday by 23:59 **Points** 0

Submit your validation check list here (in Canvas) AND

Send validation checklist + SRS for R2 to the group that is to validate your requirements.

NOTE: The reviewing group is also responsible for asking Qs on the Authoring group's oral presentation at the final project conference. For example, about choice of RE techniques, experienced RE challenges & solutions during the project. **Keep this in mind while reviewing their SRS!**

Authoring group	Reviewing group
A1	A2
A2	A3
A3	B1
B1	B2
B2	A1

For example, group A1 sends their SRS and checklist to group A2, and group A2 then reviews A1's SRS and writes a validation report reporting their review findings.

Send via Canvas Inbox: Compose new message - In To field, select course "ETSN15..." then "Student groups" then the group listed above.

We will work with the validation checklist on exercise 5.

A2 reviews A1's R2 SRS etc

Also look at grading criteria for Validation

Different kinds of checks

- Content of spec
- Structure of spec
- Consistency of spec

Fig 9.2A Contents check

Does the spec contain:

- Customer, sponsor, background
- Business goals + evidence of tracing

- Data requirements
  (database, i/o formats, comm. state, initialize)

- System boundaries & interfaces
- Domain-level reqts (events & tasks)
- Product-level reqts (events & features)
- Design-level reqts (prototype or comm. protocol)
- Specification of non-trivial functions
- Stress cases & special events & task failures

- Quality reqts (performance, usability, security . . .)

- Other deliverables (documentation, training . . .)
- Glossary (definition of domain terms . . .)

From: Soren Lauesen: Software Requirements

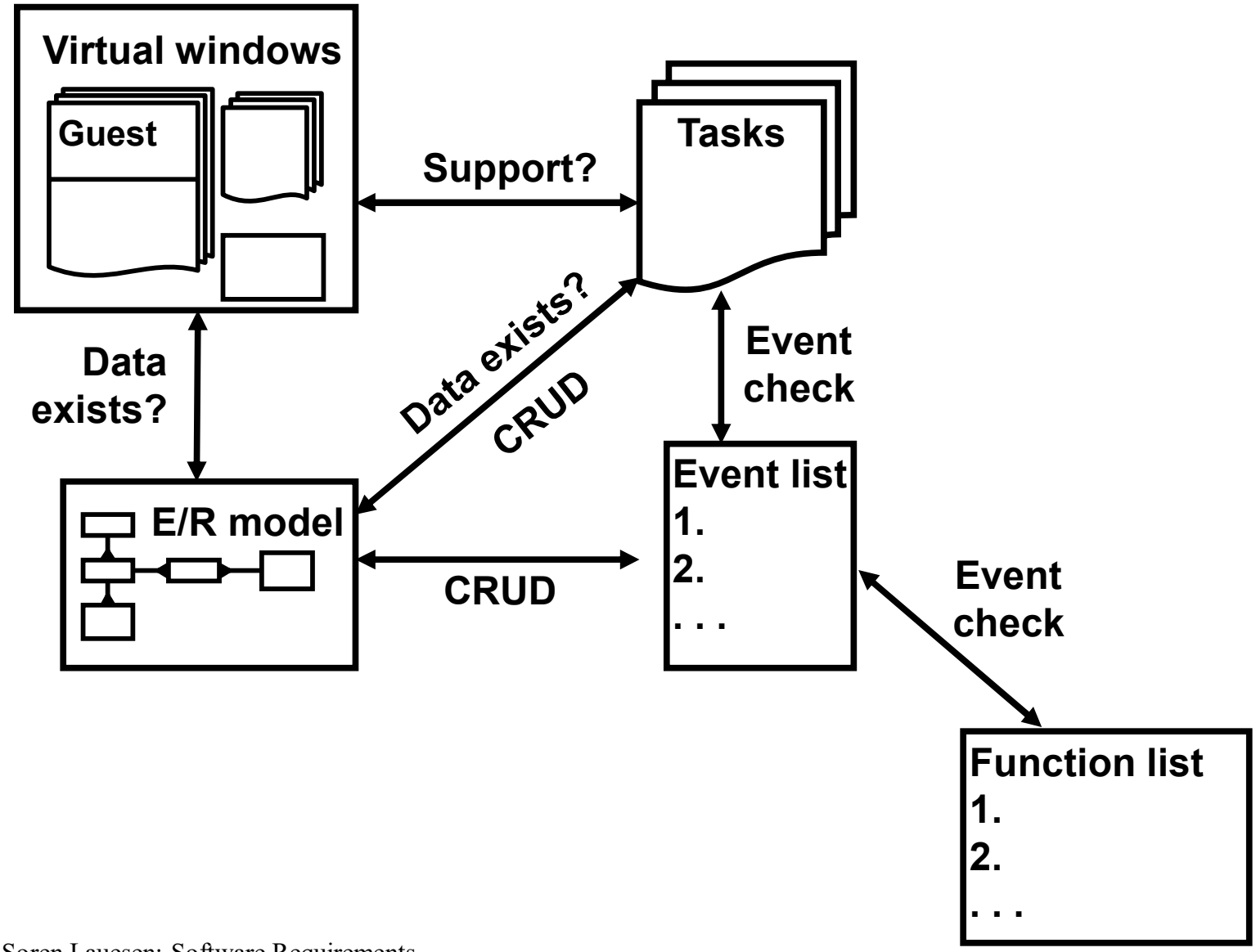
Fig 9.2B Structure check

Does the spec contain:

- Number or Id for each requirement
- Verifiable requirements
- Purpose of each requirement
- Examples of ways to meet requirement
- Plain-text explanation of diagrams, etc.
- Importance and stability for each requirement
- Cross refs rather than duplicate information
- Index
- An electronic version

From: Soren Lauesen: Software Requirements

Fig 9.2C Consistency checks



From: Soren Lauesen: Software Requirements

Fig 9.2D CRUD+O matrix

Create, Read, Update, Delete + Overview

Task \ Entity	Guest	Stay	Room	RoomState	Service	ServiceType
Book	C U O	C	O	U O		
CheckinBooked	RU	U O	O	U O		
CheckinNonbkd	C U O	C	O	U O		
Checkout	U	U O	R	U		
ChangeRoom	R	R	O	U O		
RecordService			O		C	R
PriceChange			C UDO			C UDO
Missing?	D	D		C?UD?	UD	

SLUT+Ö

Skapa
Läsa
Uppdatera
Ta bort
Översikt

From: Soren Lauesen: Software Requirements

Fig 9.3 Checks against surroundings

Reviews

Review:

Developers and customer review all parts.

Goal-means analysis:

Goals and critical issues covered?
Requirements justified?

Risk assessment:

Customer assesses his risk.
Developers assess their risk.
High-risk areas improved.

Tests

Simulation and walk-through

Follow task descriptions. Correct?
Supported?

Prototype test (experiment with prototypes):

Requirements meaningful and realistic?
Prototype used as requirement?

Pilot test (install and operate parts of system):

Cost/benefit?
Requirements meaningful and realistic?

From: Soren Lauesen: Software Requirements

Fig 9.4(A) Check list

Project:	Noise Source Location, NSL vers. X	**Date, who:** 99-03-15, JPV
Contents check	**Observations - found & missing**	**Problem?**
Customer & sponsor	Missing, OK	
. . .		
Data: Database contents	Class model as intermediate work product	
. . .		
Initial data & states	Missing	Seems innocent, but caused many problems particularly when screen windows were opened.
Functional reqs: Limits & interfaces		
Product-level events and functions	Mostly as features	
. . .		
Special cases: Stress cases		
Power failure, HW failure, config.	Missing	**Problem.** Front-end caused many problems

From: Soren Lauesen: Software Requirements

Project:	Noise Source Location, NSL vers. X	**Date, who:** 99-03-15, JPV
Contents check (2)	**Observations - found & missing**	**Problem?**
Quality reqs: Performance	Missing, also in parts not shown here.	**Problem.** Response time became important.
Capacity, accuracy	Missing, also in parts not shown here.	**Problem.** Data volume, etc. became important.
Usability	Missing	Would have been useful
Interoperability	Missing	External dataformats, robot role, etc. caused problems
. . .		
Other deliverables: Documentation	Missing	Unimportant. Company standards exist.
. . .		

Structure check	**Observations - found & missing**	**Problem?**
ID for each req.	OK	
Purpose of each requirement	Good. Domain described.	

Consistency checks	**Observations - found & missing**	**Problem?**
CRUD check: Create, read, update, delete all data?	Have been made	

Tests	**Observations - found & missing**	**Problem?**
Prototype test	Not done, nor during development.	**Should have been done**. Caused many problems later.

[INSP] Check list

Checklist för krav		
Dokument	**Krav**	**Språk**
Finns sammanfattning?	Beskriver kravet design eller ger förslag till lösningar?	Är alla syftningar entydiga (kolla alla "den", "det", "deras" och "dess")?
Finns författare?	Beskriver flera krav samma eller liknande behov?	Är alla komparative precisa och förståeliga (kolla alla "före", "innan", "snabbare", "efter")?
Finns datum?	Kan några krav grupperas ihop?	Har alla ord samma betydelse för utvecklare och användare (kolla alla: "samtidigt", "kompletthet", "minst", "normalt", "i medeltal", "ofta")
Finns innehållsförteckning?	Kan något krav delas upp i flera krav?	Innehåller något krav ord som gör kravet svårt att verifiera (kolla alla: "snabbt", "effektivt, "lagom", "minst", "mest")
Finns alla klasser av krav?	Är det möjligt att uppfylla kravet med tillgänglig teknik?	Finns vaga ord (kolla alla "några", "ibland", "ofta", "vanligen")
Finns definition av termer och begrepp?	Är kravet unikt identifierat?	Finns ofullständiga uppräkningar (kolla alla "osv.", "etc." och "till exempel")
Finns index?	Är kravet testbart?	
	Är termer och begrepp definierade?	
	Är kravet självständigt eller måste du undersöka andra krav för att förstå det?	
	Kan olika personer tolka kravet på olika sätt?	
	Har andra (liknande) krav utvärderats?	
	Är någon information redundant?	
	Saknas någon information?	

Figur 28. Checklista för att inspektera krav.

Discussion

- What are the quality criteria for a requirements specification?
  - For contractual purposes
  - For planning purposes
  - For development
  - For testing

Criteria for Good Requirements (IEEE Standard)

Correct

Incorrect requirements are useless and potentially dangerous!

If the requirements are not correct, we risk spreading mis-information within project and to customers.

Complete

Spec covers all necessary requirements to describe the full scope incl. exceptions, error handling etc

Unambiguous

Everyone understands it the same way.

Can everyone read, discuss + agree on what it means?

Clear & Concise

Simply and clearly stated. Makes it easier for others (incl pure readers) to understand.

Consistent

Are there requirements that contradict each other?

Modifiable

Modifications are easy to make, maintaining consistency of the whole specification

Verifiable

If a requirement is not verifiable, determining whether it was correctly implemented is a matter of opinion.

Design independent

Requirement describes functionality from user perspective, not how to implement

Ranked for importance and stability

Info needed to handle changes; why is req important (reqts motivation / prio / stakeholder), likely to change?

Traceable

What motivates this reqt? Indicates if it is needed. Useful when discussing scope &/ reqts changes.

Del 1 på Tentan: Påstående-anledning-frågor

För varje par av påstående/anledning svara med ett av följande alternativ:

A: Både påståendet och anledningen är **korrekta** uttalanden OCH anledningen **förklarar** påståendet på ett **korrekt** sätt.

B: Både påståendet och anledningen är **korrekta** uttalanden, men anledningen **förklarar inte** påståendet.

C: Påståendet är **korrekt**, men anledningen är ett **felaktigt** uttalande.

D: Påståendet är **felaktigt**, men anledningen är ett **korrekt** uttalande.

E: Både påståendet och anledningen är **felaktiga** uttalanden.

Påstående	Anledning	Svar
Virtuella fönster passar bra för att beskriva icke-funktionella krav.	Virtuella fönster är en bra hjälp vid validering av fullständighet av datakrav.	D
Kontextdiagram är en bra hjälp för att upptäcka saknade gränssnitt och diskutera vad som ska levereras.	Ett kontextdiagram ger en lättbegriplig översikt av systemets avgränsning och dess aktörer.	A

Extentor finns på kurswebben!

LÄS kursmaterialet i god tid!!!

Tentamensperioder och öppettider för anmälan

Läsår	Tentamens-period	Typ	Anmälan öppnar	Anmälan stänger	Tentamens-period börjar	Tentamens-period slutar
2020/21	1	Tenta	2020-10-05	2020-10-19	2020-10-23	2020-10-30
2020/21	2	Tenta	2020-11-30	2020-12-14	2021-01-04	2021-01-16
2020/21	3	Tenta	2021-02-22	2021-03-08	2021-03-15	2021-03-20

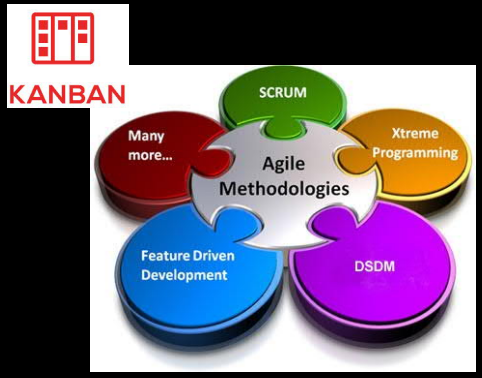
Project conference
Tue W7 come 15:10 latest

- Submit presentation material in Canvas by **Tuesday at 12.00 hrs**
- Exactly!! **8 minutes** presentation; will be interrupted!
- Contents
  - ~ 1 minute about project mission
  - ~ 3 minutes overview of project results
  - ~ 4 minutes about methods and experiences
- Questions by discussant group (same as for Validation report) – 2-4 mins
  - E.g. choice of RE techniques, experienced RE challenges & solutions during the project
- Max 1 minute for switching to next group
- One or max 2 presenters (not too much time on switching)
- **Practice before** to keep time and focus on the most important!
- If you want to practice English this is a good chance!
  (Swedish is also Ok)

Order of presentation at Project Conference

Mandatory attendance!

Presenter	Discussant
A1	A2
A2	A3
A3	B1
15 min break	
B1	B2
B2	A1



Agile Requirements Engineering

[AGRE] [ATCR]

Requirements change

Extensive documenting is costly & time consuming

”We don’t do requirements. We are agile.”

Cheaper to manage changes gradually

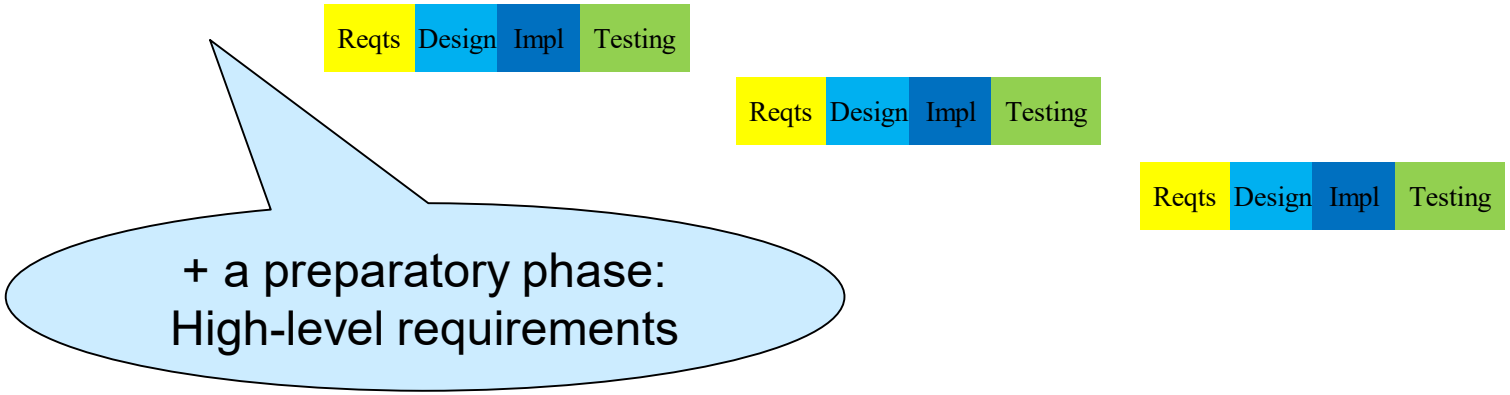
Customer can tell us

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

The Agile Manifesto, http://agilemanifesto.org/, 2001

Traditional Development Process

Agile Development Process – Integrated RE



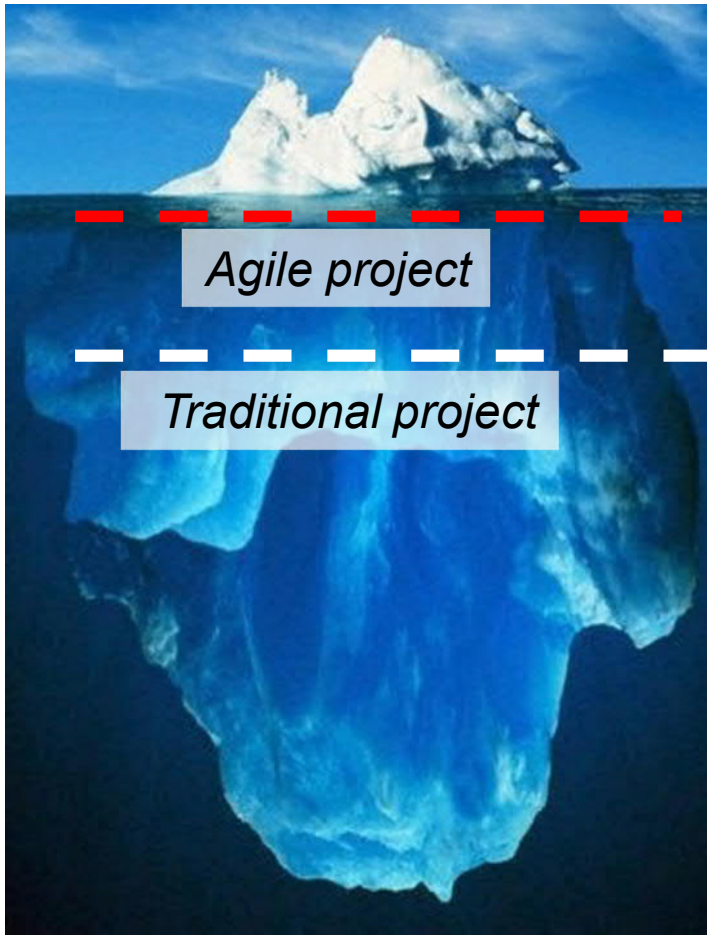
- Same activities, different sizing and timing
  - → Different principles and management approach
  - → Different people detailing requirements
  - → Different documentation formats

RE in Agile Projects [AGRE]

Practices

- *Iterative RE*: Gradual detailing
- Work order
  - *Extreme prioritization*: Just-in-time
  - *Constant planning*
- Integrated RE:
  Dev roles more involved in RE
  - *Face-to-face communication*
  - *Reviews & tests*
  - *Prototyping*
  - *Test-driven development*

Level of detail at dev start



”We don’t do requirements. We are agile.”

All projects need & have
requirements ==
ideas/decisions of what product should do

In Agile projects, some reqts are documented
- as traditional requirements
- as user stories & acceptance criteria
- as backlog entries
- as test cases
- combo of “requirements” and other artefacts

Many requirements are NOT documented

User story & Acceptance Criteria (TCs)

Cohn, Mike. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
Good book on hands-on agile requirements!

User story

As a passenger, I can cancel a flight reservation

Acceptance criteria / test cases

- Verify that a premium member can cancel the same day without a fee
- Verify that a non-premium member is charged 10% for a same-day cancellation
- Verify that an email confirmation is sent
- Verify that the hotel is notified of any cancellation

Test cases as Requirements

Paper [ATCR]

Bjarnason, Unterkalmsteiner, Borg, & Engström (2016). *A multi-case study of agile requirements engineering and the use of test cases as requirements.* Information and Software Technology, 77, 61-79.

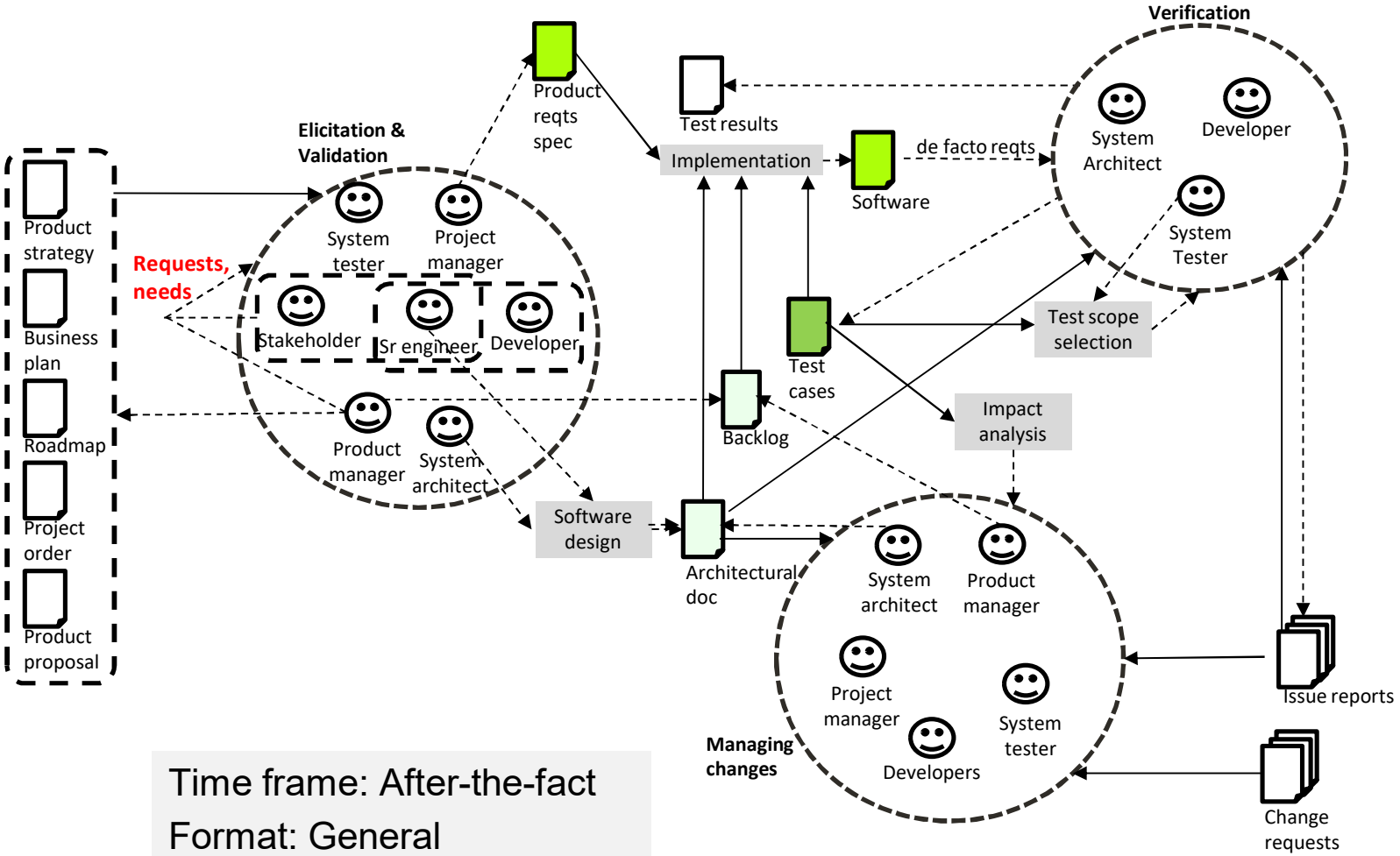
Case study of 3 companies

- Company A: Medium-sized, Networking equipment
  - **De facto practice**
- Company B: Small, Consultants
  - **Tool-supported Behaviour-driven development**
- Company C: Large, Telecom
  - **Story-test driven for manual test cases**
  - **Stand-alone strict and manual**

Variation points of TCR [ATCR]

- **Documentation time frame**
  upfront or after-the-fact (during testing)
- **Requirements format**
  ranging from natural language to structured
- **Machine executable specification**
  automated tests
- **Tool support for TCR**

De facto TCR [ATCR: Company A]



Time frame: After-the-fact
Format: General
Executable: Partly
Specific tooling: No

De facto TCR

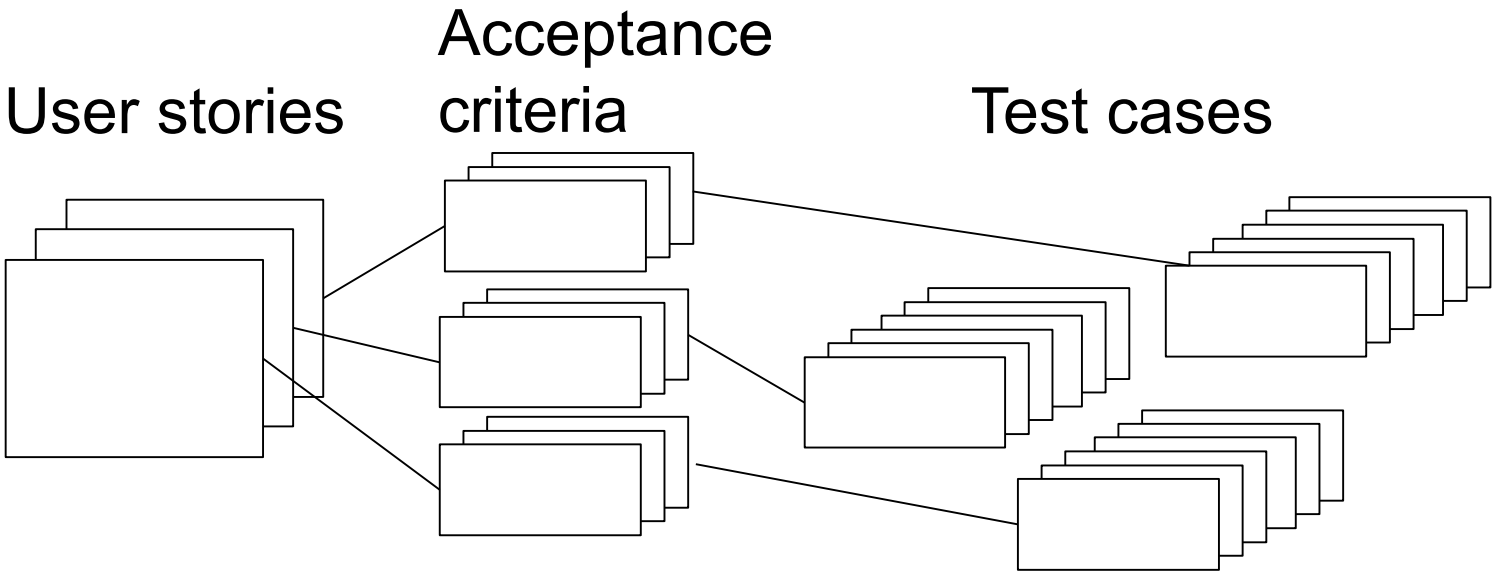
Benefits	Challenges
Elicitation & Validation	
EB1 Cross-functional communication	EC2 Active customer involvement
EB2 Aligning goals & perspectives	EC3 RE competence
EB4 Creativity supported	
Verification	
VB1 Supports regression testing	VC1 Varying (biased) results for man test
	VC2 Correct reqts info for testing
Managing changes	
	MC2 Missing reqts context info
	MC3 Multiple products in product line
Tooling	
	TC1 Tool integration

Story-test driven TCR
[ATCR: Planned for Company C]

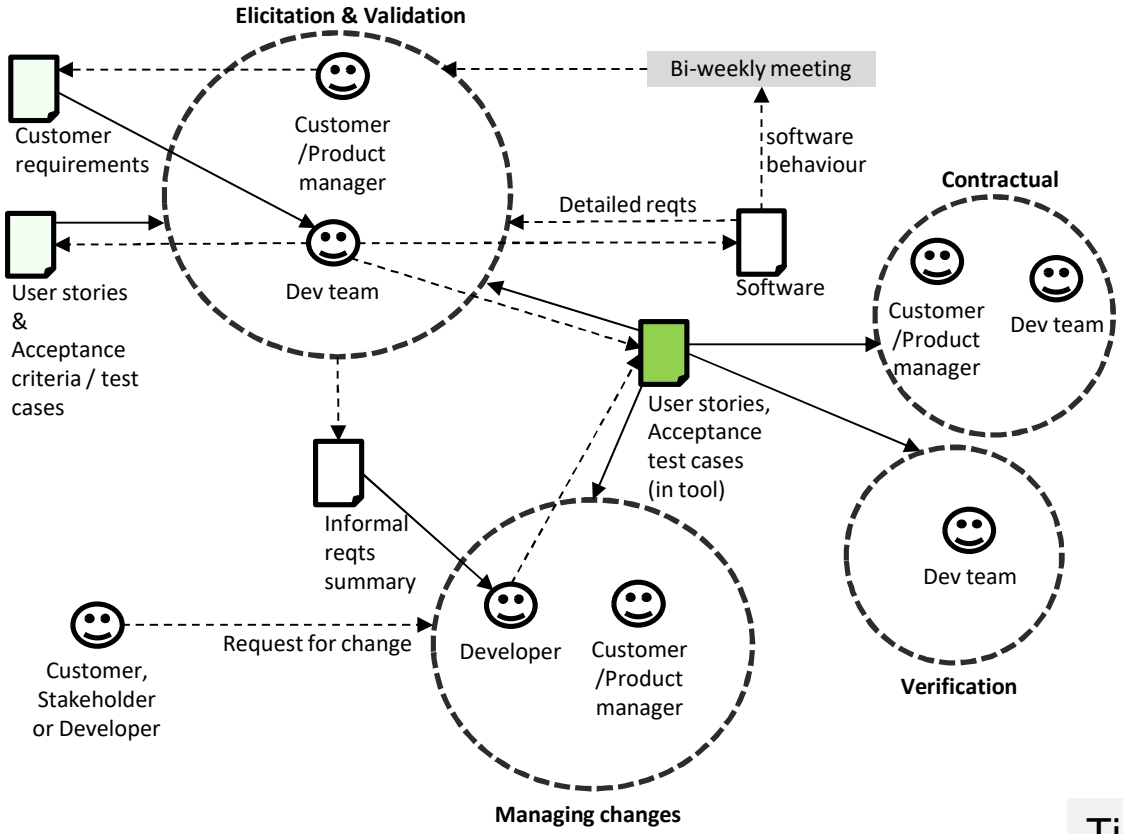
Requirements are documented as

- User stories
- Acceptance criteria

Time frame: Upfront
Format: Semi structured
Executable: Partly
Specific tooling: Yes



Behaviour-Driven TCR
[ATCR: Company B]

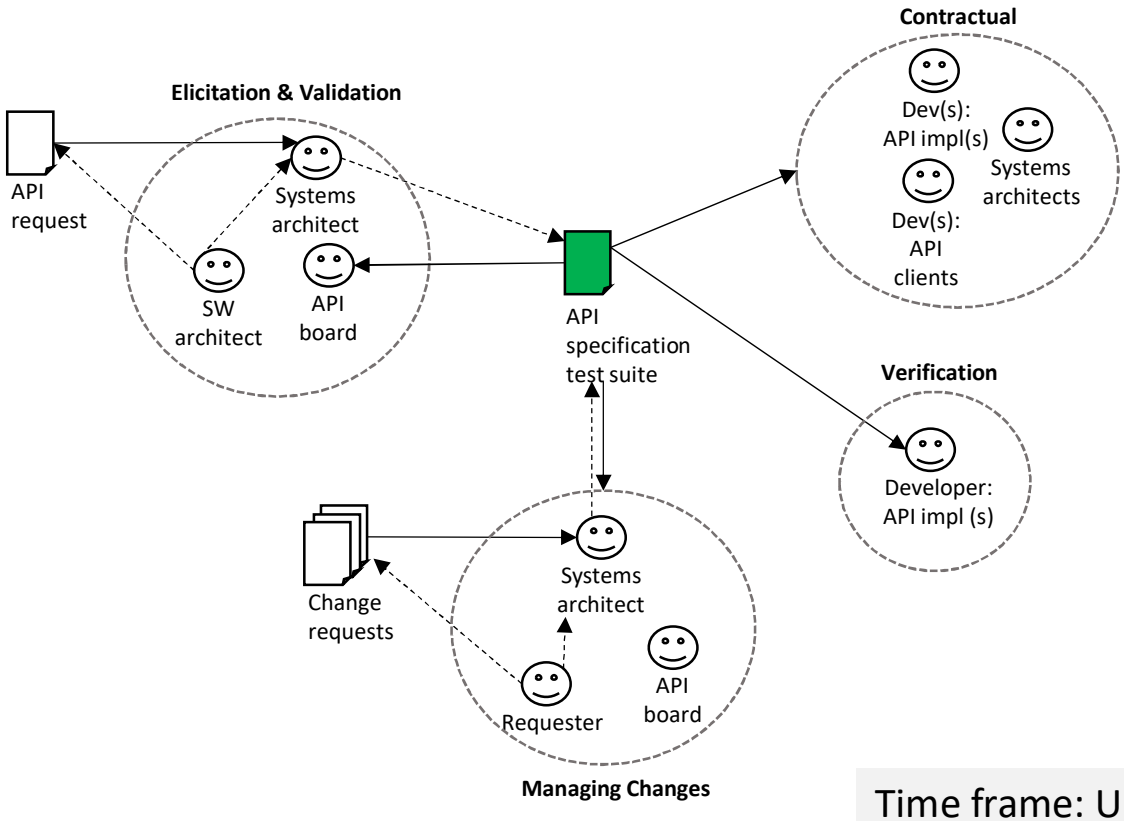


Time frame: Upfront
Format: Structured
Executable: Yes
Specific tooling: Yes

Behaviour-Driven TCR

Benefits	Challenges
Elicitation & Validation	
EB1 Cross-functional communication	EC3 Technical & testing competence
EB3 Addresses barrier of ‘solutions’	EC4 Complex reqts, e.g. quality, ui
EB4 Creativity supported	
Verification	
VB1 Supports regression testing	VC1 (biased) results for HL tests
	VC2 Correct reqts info for testing (low coverage and rate of update + outside of team)
Managing changes	
MB2 Keeping reqts up to date	MC3 Multiple products (HWs) in product line
MB4 Detecting impact of change	

Stand-alone strict [ATCR: Company C]



Time frame: Upfront

Format: Semi--structured

Executable: Yes

Specific tooling: No

Stand-alone Strict TCR

Benefits	Challenges
Elicitation & Validation	
EB1 Cross-functional communication	EC3 Ensuring similar competence
Verification	
VB1 Supports regression testing	
Maintaining Changes	
MB1 Communication of changes	
MB2 Keeps reqts up to date	
MB3 Maintaining reqts-test alignment	
MB4 Detecting impact of change (post-fact)	
Customer agreement / contractual	
CB1 Resolving conflicting views	
CB2 Certification of compliance	

TCR: Affect on RE process

[ATCR Table 7]

Benefits	Challenges
Elicitation and validation	
EB1 Cross-functional communication	EC1 Good Customer-Developer relationship
EB2 Align goals & perspectives between roles	EC2 Active customer involvement
EB3 Address barrier of specifying solutions	EC3 Sufficient technical and RE competence
EB4 Creativity supported by high-level of requirements	EC4 Complex requirements, e.g. quality requirements
Verification	
VB1 Supports regression testing	VC1 Varying (biased) results for manual tests
VB2 Increased requirements quality	VC2 Ensuring correct requirements info to test
VB3 Test coverage / RET alignment	VC3 Quality requirements
Tracing	
TB1 Implicit Requirements - test case tracing	TC1 Tool integration
Managing changes	
MB1 Communication of changes	MC1 Locating impacted requirements
MB2 Requirement are kept updated	MC2 Missing requirement context
MB3 Maintaining RET alignment	MC3 Multiple products in one product line
MB4 Detecting impact of changes	
Customer agreement/contractual	
CB1 Facilitate resolving conflicting views	CC1 Use-case related structuring
CB2 Support certification of compliance	

Paper [AGRE]

Agile Requirements Engineering Practices: An Empirical Study
by Balasubramaniam Ramesh and Lan Cao
In: IEEE Software, pp. 60-67, January/February 2008

Agile RE practices in 16 companies

Adoption level	Practice						
	Face-to-face communication	Iterative RE	Extreme prioritization	Constant planning	Prototyping	Test-driven development	Reviews & tests
High	8	9	10	8	8	5	11
Medium	8	5	6	6	3	1	4
Low	0	2	0	2	0	0	1
None	0	0	0	0	5	10	0

Organization pseudonym	Industry and products
Enco	Energy and communications. Offers forecasting tools.
HealthCo	Healthcare and utilities. Offers an online service to help customers select health insurance and utility services.
Venture	Across industries. Helps brick-and-mortar companies develop a Web presence.
Entertain	Film and television industry. Offers high-tech indexing and search tools online.
HuCap	Administration. Carries out human-resource administration for other companies online.
TravelAssist	Transport and tourist industry. Offers online services.
ManageRisk	Across several industries. Offers insurance online.
Transport	Transportation and logistics industry. Offers services online.
Transport	Transportation and logistics industry. Offers services online.
ServeIT	Consulting and services. We studied the part of the firm that offers consulting services for business-to-business communication.
HealthInfo	Healthcare information systems. Offers information systems solutions to hospitals, physicians' offices, and home healthcare providers.
SecurityInfo	Security software. Offers software for Internet security.
AgileConsult	Software consulting. Offers consulting services on agile software development.
EbizCo	Packaged software development. Offers e-business connections and transactions.
FinCo	Online financial-transaction support. Offers online payments.
NetCo	Network software consulting. Offers services on developing network systems and architectures.
BankSoft	Banking information systems. Offers software that handles financial transactions.

Face-to-face communication

Direct communication between customer and development

- Techniques
  - User Stories == high-level requirements spec
  - Complemented by other artefacts, e.g. "backlog"
- Prerequisites
  - Active involvement of (knowledgeable) customers

Customers can steer project

Avoids time-consuming documentation

Risk of inadequate requirements

On-site customer rep is challenging

Handling more than one customer

Relies on trust rather than agreed requirements

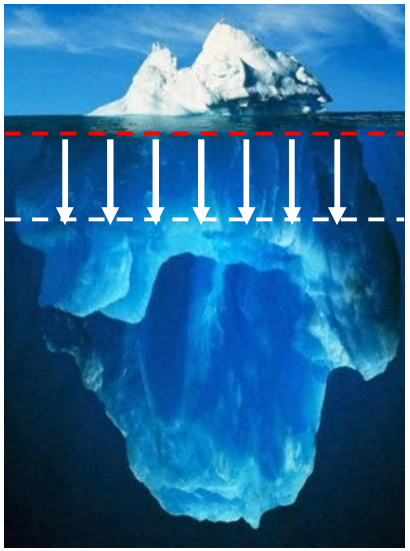
Face-to-face Communication

Perceived Benefits

- **Customers can steer** the project in unanticipated directions, especially when their requirements **evolve** owing to changes in the **environment** or their own **understanding** of the software solution.
- Informal communication **obviates the need for time-consuming documentation** and approval processes, which are perceived as unnecessary, especially with evolving requirements.

Perceived Challenges

- If **intensive interaction** between **customers and developers cannot** be established, this approach poses the **risk of wrong or inadequate requirements**.
- Achieving **on-site customer representation is difficult** (even in the form of a surrogate product manager).
- When **more than one customer group** is involved, achieving **consensus/compromise** in the short development cycle is challenging.
- **Customers** used to a traditional development process **might not understand or trust** the agile RE **process**, which doesn't produce detailed requirements.



Iterative RE

Requirements **emerge during development** based on **initial high-level requirements**

- Techniques
  - Requirements analysis and detailing for each development cycle
  - Requirements intertwined with design

Good customer relationship

Clearer and understandable requirements due to direct customer interaction

Accurate cost and scheduling of project

Neglect of quality requirements

Lack of documentation beyond dev team

Iterative RE

Perceived Benefits

- Iterative RE creates a more **satisfactory relationship with the customer.**
- Requirements are clearer and more **understandable** because of the **immediate access to customers** and their involvement in the project when needed.

Perceived Challenges

- **Cost & Schedule Estimation** for entire project: Difficult, since the project scope is subject to constant change. Obtaining management support for such projects could be challenging.
- Minimal documentation: When a communication breakdown occurs the **lack of documentation** might cause a variety of problems (e.g., **scalability, evolution, introduction of new team members**).
- **Neglect of quality requirements**: Especially during early development cycles, **customers often focus** on core functionality and ignore quality reqts such as scalability, maintainability, portability, safety, or performance.

Extreme Prioritization & Constant Planning

Aim to deliver **most valuable features first**

Responsive to changes in customer demands

- Techniques
  - Work on most valuable features first
  - Continuously revise prioritisation & planning (for each iteration)
  - Constant feedback from customer

Customer provides business prio

Re-prioritization supported by dev process

Early validation minimizes need & cost for major changes

Other criteria suffer, e.g. quality

Instability in dev work

Inadequate architecture and increased costs

Refactoring requires time and experience

Extreme Prioritization

Perceived Benefits

- Involved customers can **provide business reasons** => **clear understanding** of the customer's priorities **helps the development team** better meet customer needs.
- agile RE **built and provides** numerous opportunities for **reprioritization**.

Perceived Challenges

- **Only business value prio** might cause major problems in the **long run (e.g., 'omitted' quality reqts**).
- Continuous reprioritization, when not practiced with caution, may lead to **instability**

Constant Planning

Perceived Benefits

- The early and constant **validation** of requirements largely **minimizes the need for major changes**.
- Thus, the **cost of change request decreases** dramatically compared to traditional software development.

Perceived Challenges

- Often, **architecture (early cycles) becomes inadequate** as **requirements change** and **redesign** of the architecture **adds significantly to project cost.**
- **Refactoring** depends on developers' **experience and schedule pressure**.
- Refactoring often doesn't fully address the problem of inadequate/inappropriate **architecture**.

Prototyping & Reviews & Acc Test

Communicate through prototypes and frequent review meetings

Involves customers, developers and testers

Requirements **validation** and **refinement** through feedback

- Techniques
  - End-of-sprint sign-off meeting

Efficient validation
Assess project status
Trust: Customer, Mgmt
Early problem identification

Risks with evolving prototypes in production
Unrealistic expections regarding leadtime
Weak formal validation, consistency checks
Dev of acc tests require access to customers

Prototyping

Perceived Benefits

- **Avoids** incurring **overhead of creating formal requirements** documents.

Perceived Challenges

- **Risk in production mode** may cause problems with features such as **scalability, security, and robustness**.
- **Quick deployment of prototypes** in the early stages may create **unrealistic expectations** among customers. unwilling to accept longer development cycles for more scalable and robust implementations

Reviews and Acceptance Tests

Perceived Benefits

- ascertain **project on target?**
- increase customer **trust and confidence**
- **identify problems early**.
- **obtain management support**

Perceived Challenges

- Weak validation due to lack of stringency: **formal modeling, consistency checking**
- acceptance testing requires **access to the customers**

Test-Driven Development

Developers **create test before writing new code**

Tests specify expected behaviour of code

Tests capture complete requirements

Traces to production code facility reqts changes

Requires competence in testing, requirements understanding and customer collaboration

Most organizations fail to implement this practice

Test-driven Development

Perceived Benefits

- **traceability** facilitates incorporating changes. **Tests** may be used to capture complete requirements and design documentation that are linked to production code. This.

Perceived Challenges

- **developers aren't accustomed to writing tests before coding**. Also, consistently following the practice demands a lot of discipline.
- Moreover, TDD **requires a thorough understanding of the requirements** and extensive **customer collaboration;** involves **refining low-level specifications** iteratively.
- most organizations reported that they're unable to implement this practice.

Summary of Benefits & Challenges of Agile RE

Practices	Benefits	Challenges
Face-to-face communication	• Customers can steer the project • No time-consuming documentation	• If no intensive interaction then bad reqts. • On-site customer representation is difficult
Iterative RE	• Better relationship with the customer • More understandable reqts	• Cost & Schedule Estimation • Lack of documentation • Neglect of non-functional requirements
Extreme prioritization	• Customers provide business reasons • Opportunities for reprioritization.	• Business value not enough • May lead to instability
Constant planning	• Minimizes the need for major changes • Cost of addressing a change decreases	• Early architecture becomes inadequate • Refactoring isn't always obvious
Prototyping	• Help communicate with customers to validate and refine requirements	• Risky to deploy prototypes into production • Create unrealistic expectations
Test-driven development	• Gives traceability that make changes easier	• Developers unused to test before coding • Requires a thorough understanding of reqts and extensive collaboration between the developer and the customer
Reviews & acceptance tests	• Help to know if project is on target • Increase customer trust and confidence • Identify problems early • Obtain management support	• No formal model or verification of reqts • Consistency checking or formal inspections seldom occur. • Difficult if lacking customer access

To do …

- Read [AGRE], [ATCR], Lau:9, [INSP]
- Exercise E5 Validation (project validation preparation, **bring** your **System Requirements specification** + **litterature**)

Week 6

- Project deliverables (see project descr / course programme):
  - Release R2 & **Validation checklist** (Sun 23.59)
  - **Validation Report** based on checklist from other group (Fri)
  - Handled in Canvas: as assignment submission and via Canvas mail (SRS R2+checklist+Validation Report)
- Project meeting with supervisor
- Sign-up for exam (**tentaanmälan** öppnar på måndag!)

Week 7

- Submit **Conf Presentation MATERIAL (CP) Tue W7 before 12.00** hrs
- Prepare Qs as “discussant” (review) group
- Tue W7 be there **15.10** latest for PROJECT CONFERENCE **Mandatory examination!**