ETSN15 2024 Requirements Engineering

Agile RE [AGRE]

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

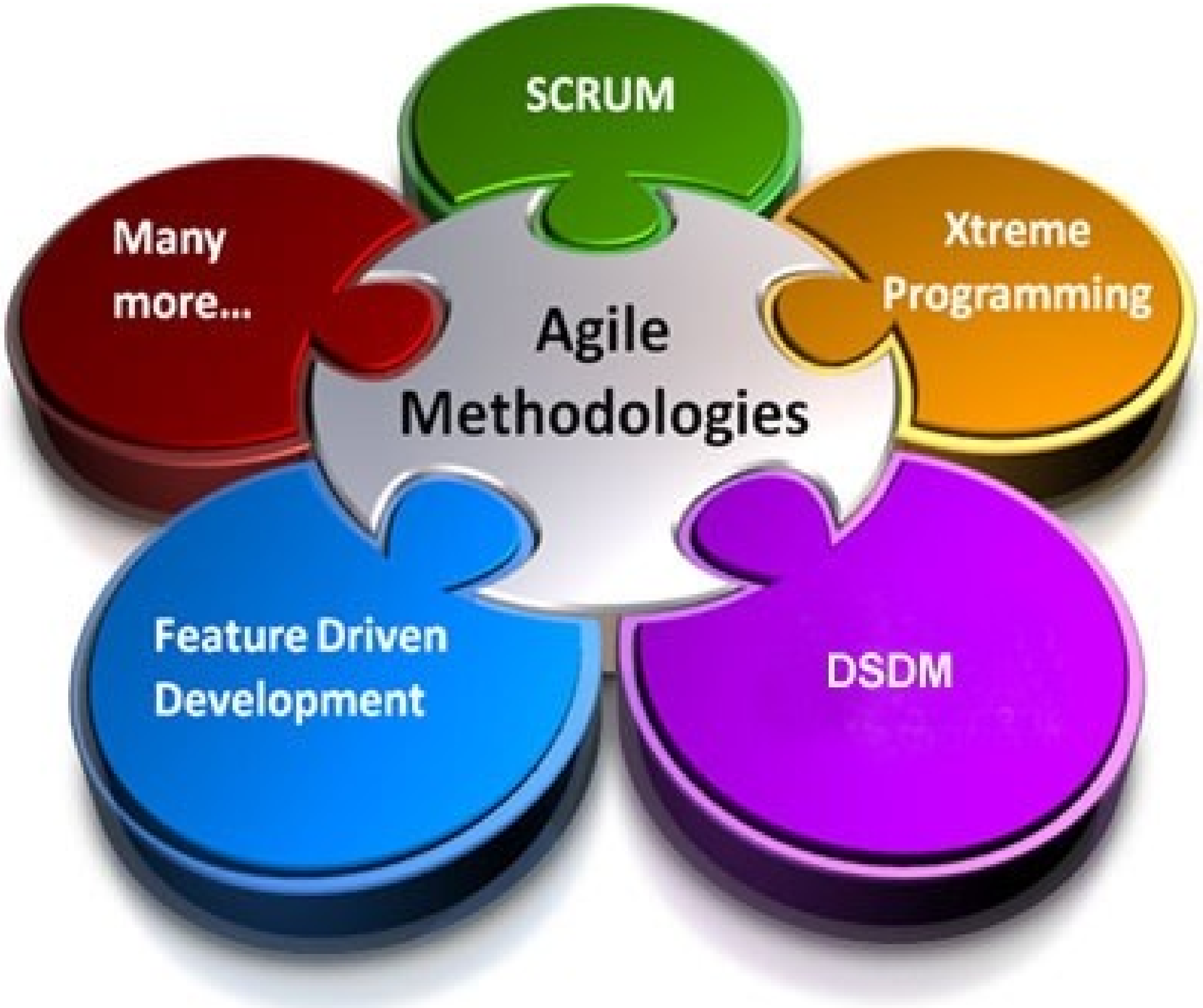
Requirements change

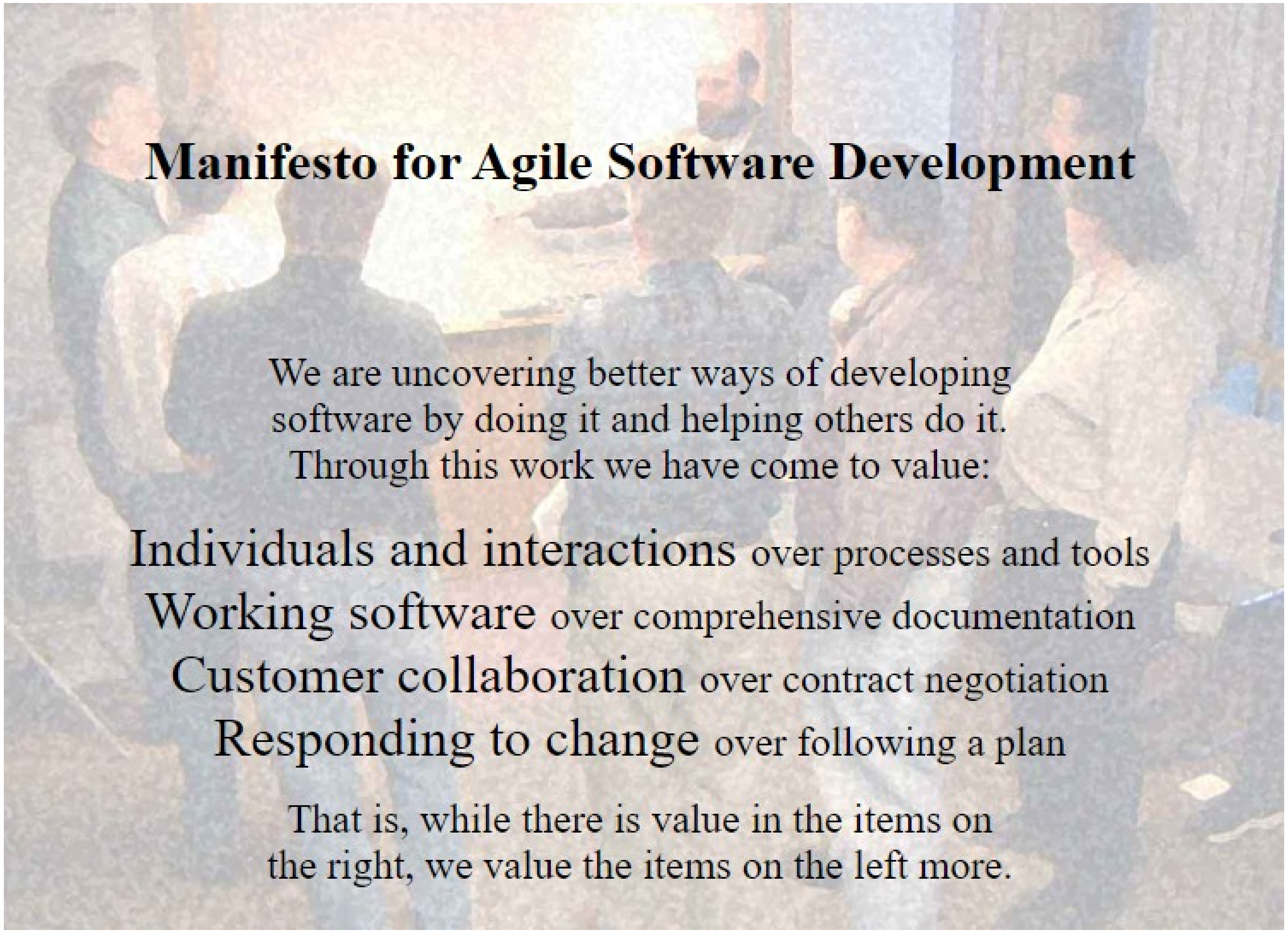
- Constantly
- Sometimes quickly!
- Why?
  - We learn
  - Changed needs & priorities by users & other stakeholders
  - Tech development
  - Competition
  - Time-to-market pressures

Agile Methodologies

See overview in wikipedia:
https://en.wikipedia.org/wiki/Agile_software_development

KANBAN





Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

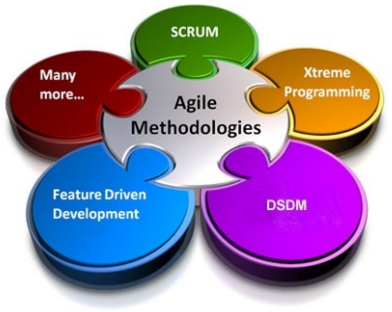
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

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

Principle-Driven Approach based on Agile Manifesto



More valuable	**Valuable**
Individuals & interactions	Processes and tools
Working software	Comprehensive documentation
Customer collaboration	Contract negotiation
Responding to change	Following a plan

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

Underlying assumptions → agile RE

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

Extensive documentation	Light-weight / agile RE
Costly	Cheaper - initially
Time consuming	Quicker - initially
RE competence **REQUIRED**	RE competence **nice-to-have**
Dev - stakeholder interaction **NOT** required	Dev – stakeholder interaction **REQUIRED**

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

Wrong! Exactly all projects need & have requirements ==
ideas/decisions of what the product should do

In Agile projects, some requirements ***are*** documented

- as traditional requirements
- as user stories & acceptance criteria
- as backlog entries
- as test cases
- combo of “requirements” and other artefacts

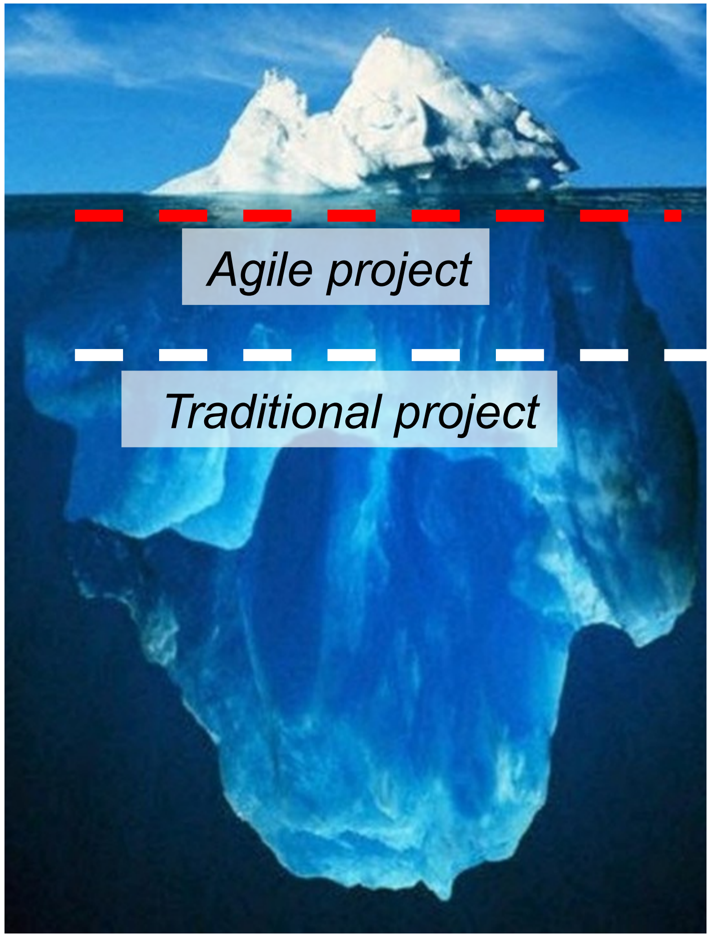
Many requirements are ***NOT*** documented (can be risky)

RE in Agile Projects

Practices [AGRE]

- *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



Paper [AGRE]

Agile Requirements Engineering Practices:
An Empirical Study

by Balasubramaniam Ramesh and Lan Cao

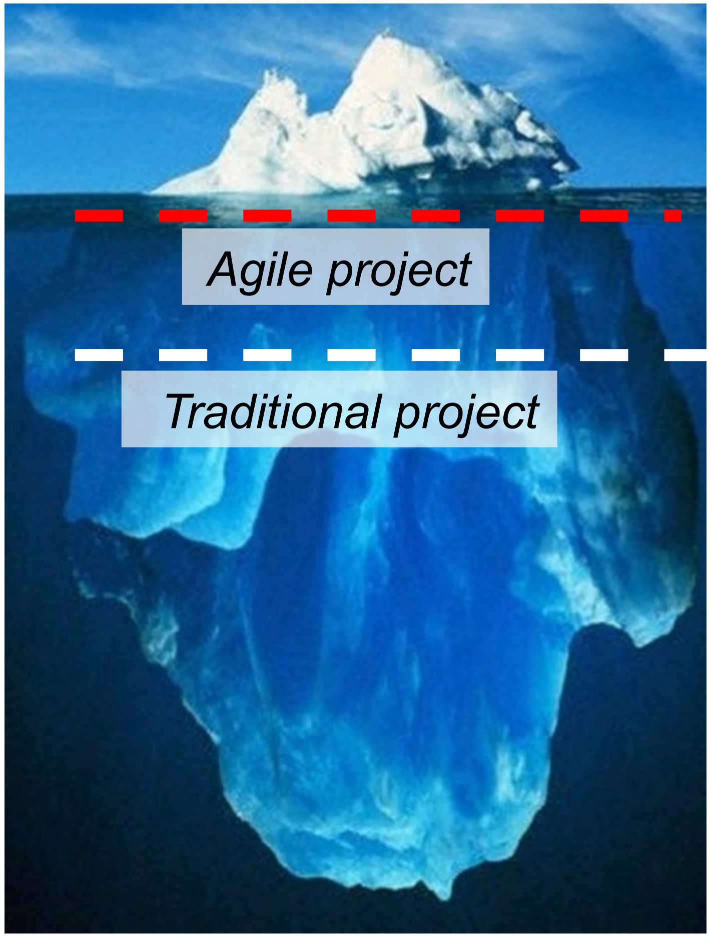
In: IEEE Software, pp. 60-67, January/February 2008

RE in Agile Projects [AGRE]

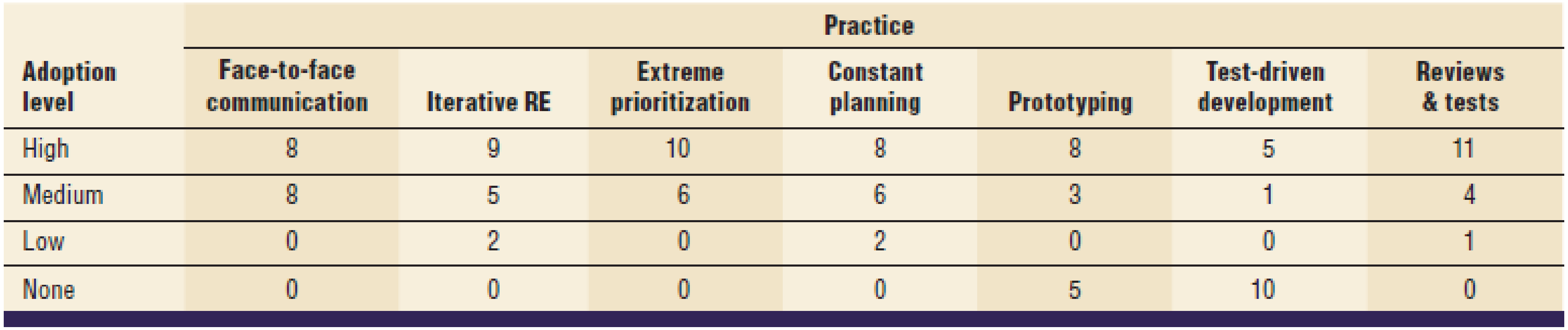
Practices

- *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*

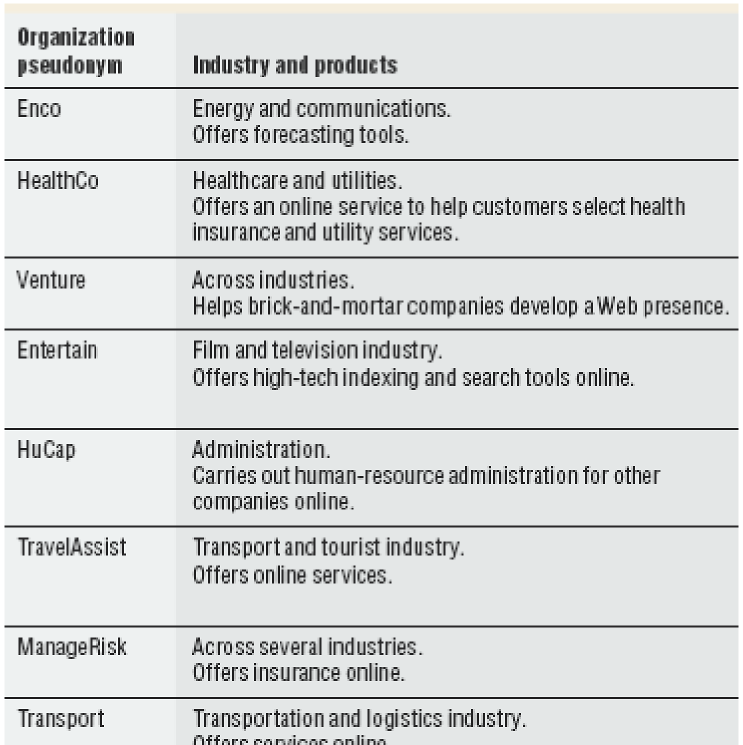
Level of detail at dev start



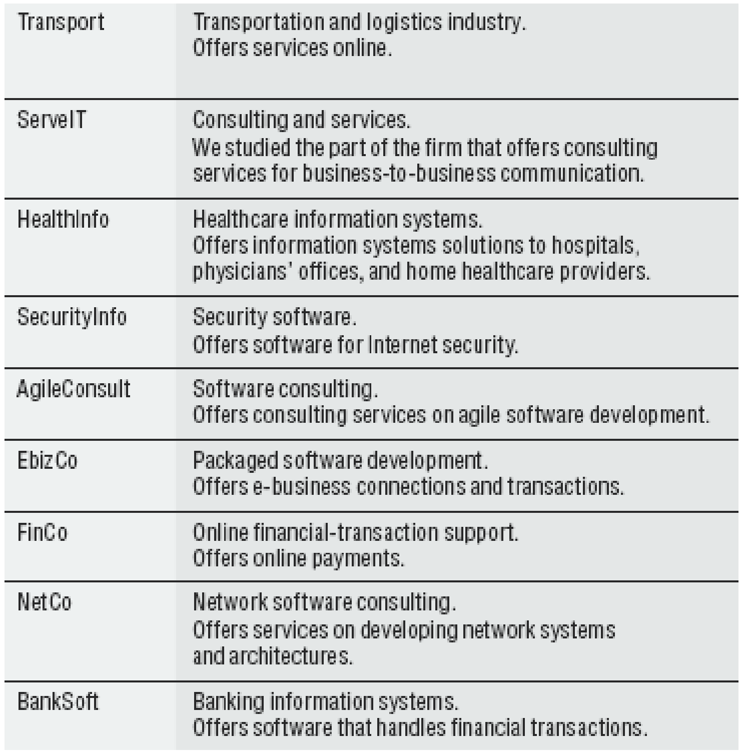
Agile RE practices in 16 companies



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

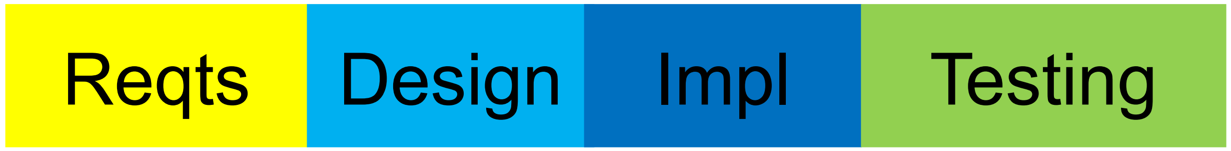


Organization pseudonym	Industry and products
Enco	Energy and communications. Offers forecasting tools.
HealthCo	Healthcare and utilities. Offers an online service to help customers select health insurance and utility services.
Venture	Across industries. Helps brick-and-mortar companies develop a Web 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

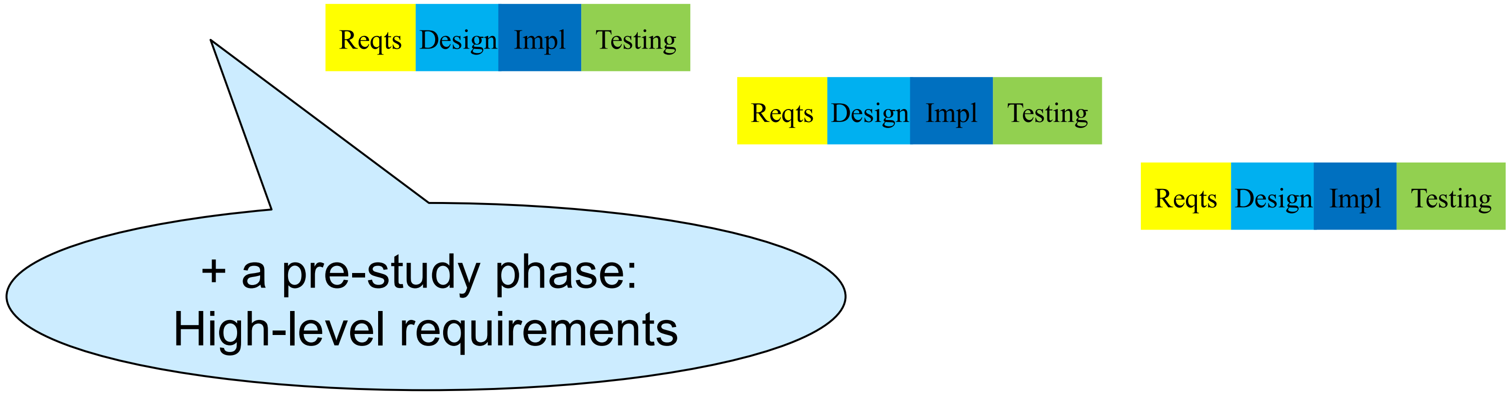


Transport	Transportation and logistics industry. Offers services online.
ServeIT	Consulting and services. We studied the part of the firm that offers consulting services for business-to-business communication.
HealthInfo	Healthcare information systems. Offers information systems solutions to hospitals, 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.

Traditional Development Process

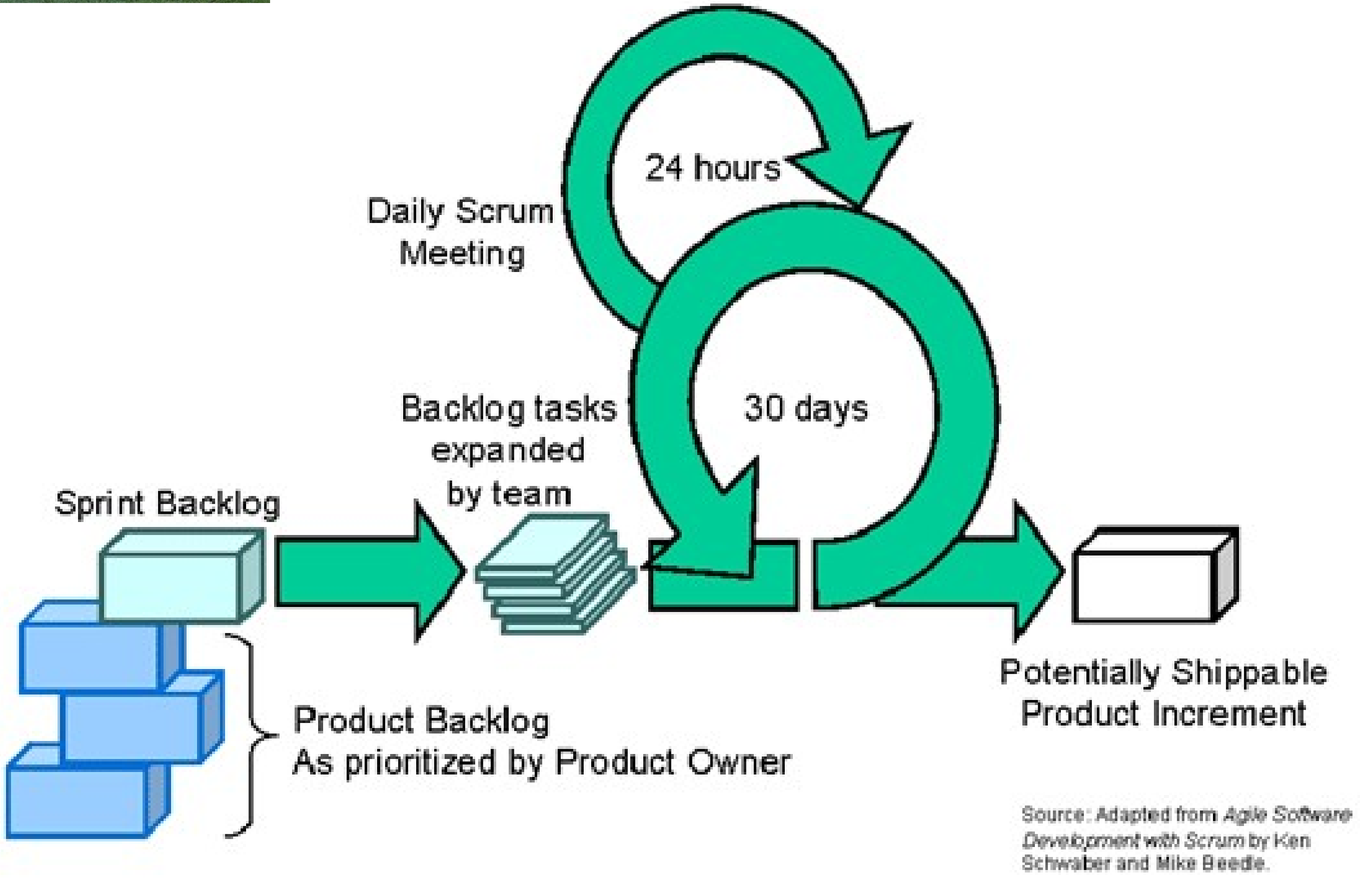


Agile Development Process – Integrated RE



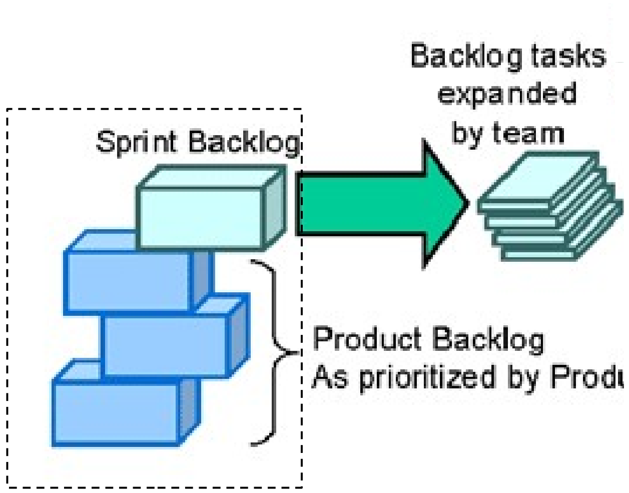
- Same activities, different sizing and timing
  - → Different principles and management approach
  - → Different people detailing requirements
  - → Different documentation formats

Scrum Development


24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Product Backlog
As prioritized by Product Owner
Potentially Shippable Product Increment
Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

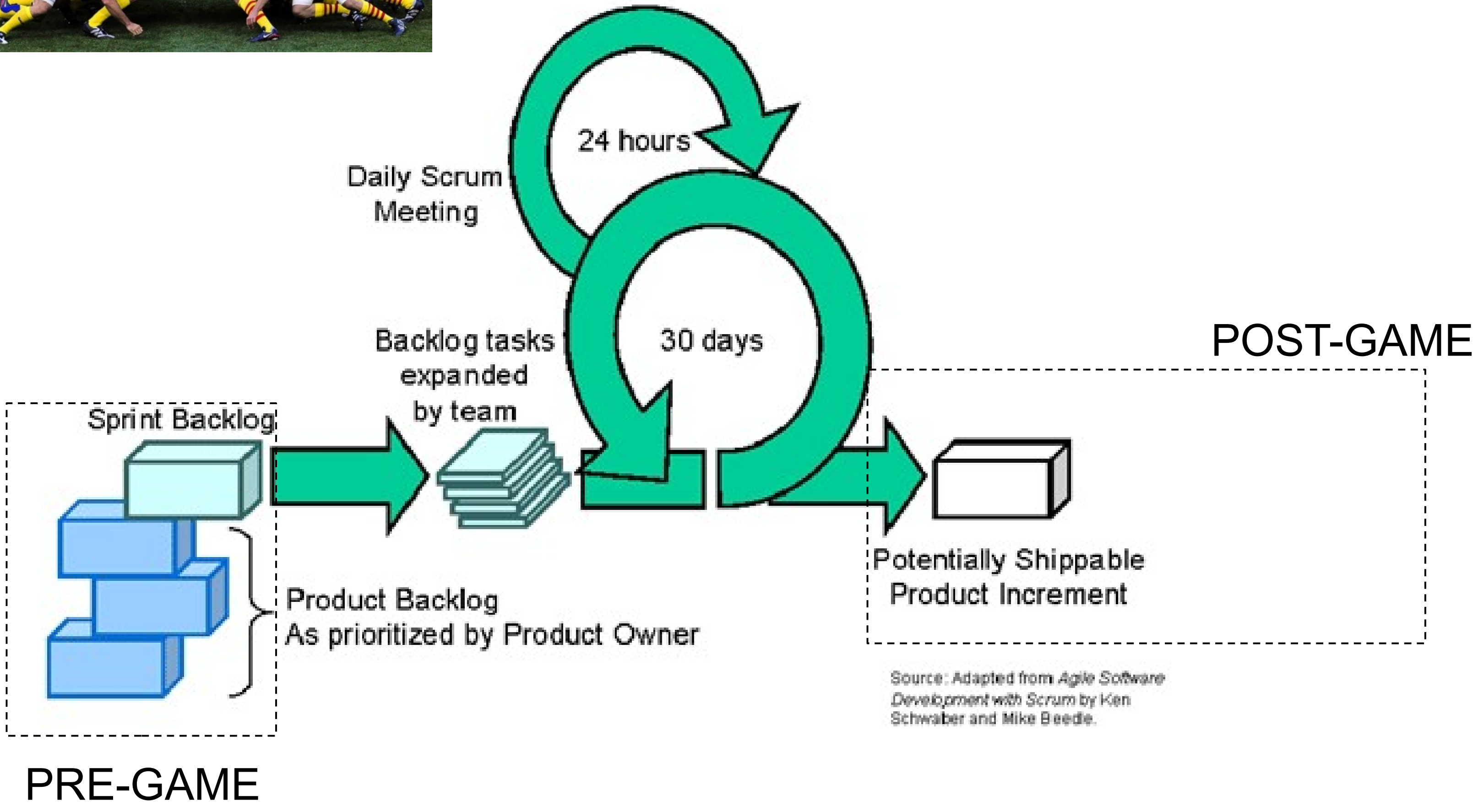
Requirements in Scrum

- Pre-game – Scrum planning & backlog grooming == RE
- Requirements in Backlogs


Backlog tasks expanded by team
Sprint Backlog
Product Backlog As prioritized by Prod

PRE-GAME

Scrum sprints - Time boxed iterations



Daily Scrum Meeting
24 hours
Backlog tasks expanded by team
30 days
POST-GAME
Sprint Backlog
Product Backlog
As prioritized by Product Owner
Potentially Shippable Product Increment
Source: Adapted from Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.
PRE-GAME

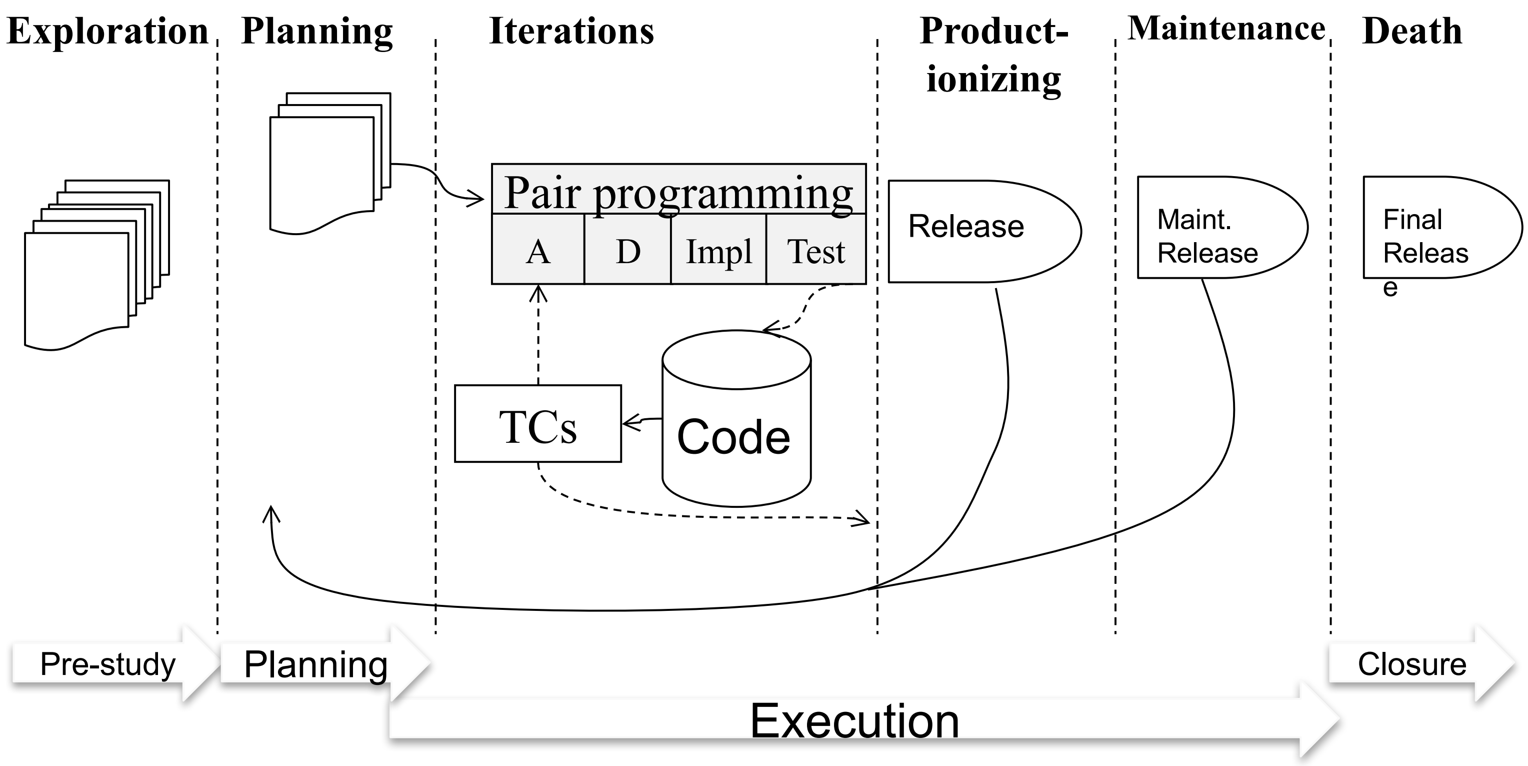
- Requirements INTEGRATED in backlog, test cases, design docs etc

Continuous Feedback & Transparency

Business, Management and Development roles involved in

- Sprint planning meeting
- Daily stand-up meetings
- End-of-sprint demo
- Sprint retrospective meetings

Phases of XP



Kent Beck, eXtreme Programming eXplained, Ch 21, Addison Wesley, 2000

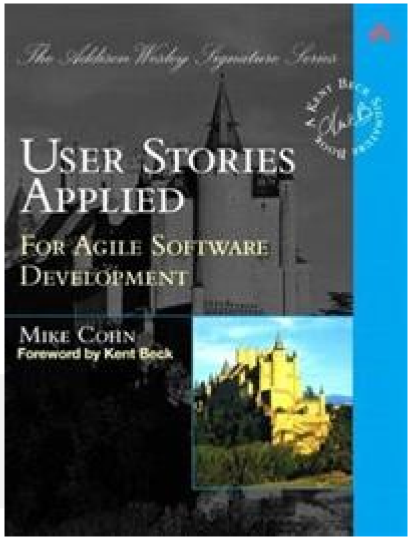
User story & Acceptance Criteria

User story:

As a passenger, I can cancel a flight reservation

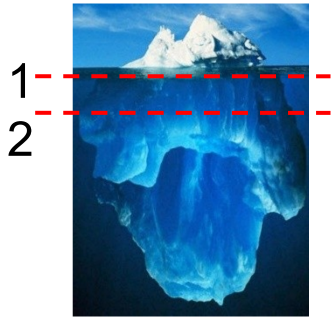
Acceptance criteria / test cases

- 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


The Addison-Wesley Signature Series
A Kent Beck Signature Book
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck

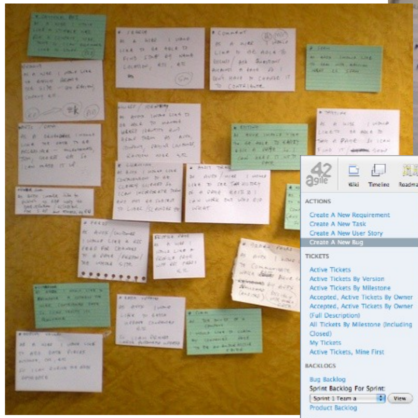
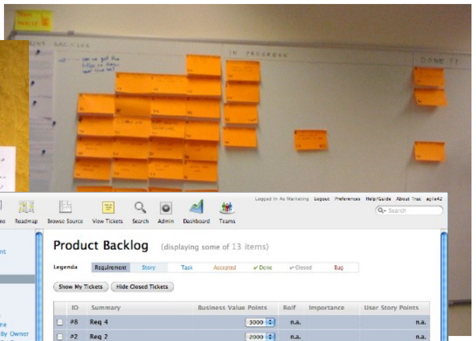
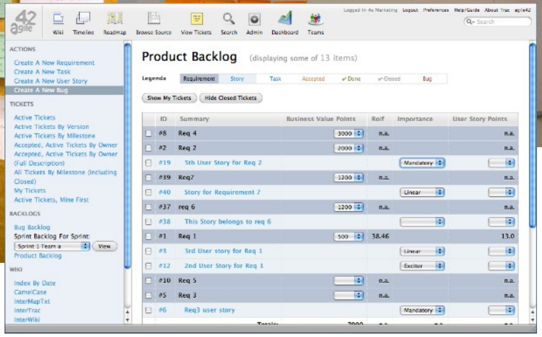
Cohn, Mike. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

Specification of user stories



1. Product Owner/Customer **defines** & **prioritizes** Epics/User stories in **product backlog**
2. Team **defines** details for each user story in **sprint backlog**
    1. Tasks
    2. Acceptance criteria & test cases

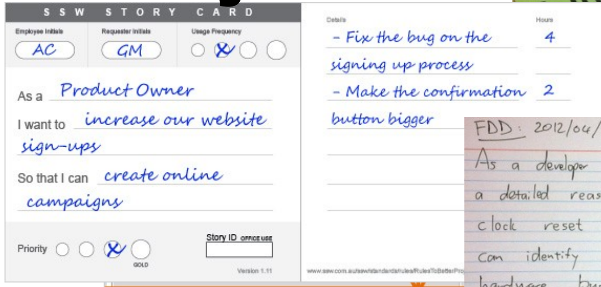
Backlogs

Product Backlog (displaying some of 13 items)

	ID	Summary	Business Value Points	RoIf	Importance	User Story Points
☐	#8	Req 4	3000	n.a.		n.a.
☐	#2	Req 2	2000	n.a.		n.a.
☐	#19	5th User Story for Req 2			Mandatory	
☐	#39	Req7	1200	n.a.		n.a.
☐	#40	Story for Requirement 7			Linear	
☐	#37	req 6	1200	n.a.		n.a.
☐	#38	This Story belongs to req 6				
☐	#1	Req 1	500	38.46		13.0
☐	#3	3rd User story for Req 1			Linear	
☐	#12	2nd User Story for Req 1			Exciter	
☐	#10	Req 5		n.a.		n.a.
☐	#5	Req 3		n.a.		n.a.
☐	#6	Req3 user story			Mandatory	

Story cards



S S W S T O R Y C A R D

Employee Initials	Requester Initials	Usage Frequency
AC	GM	○ ✗ ○ ○

As a Product Owner
I want to increase our website sign-ups
So that I can create online campaigns

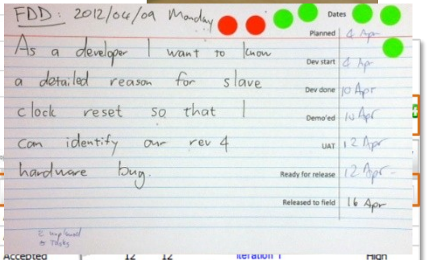
Priority ○ ○ ✗ ○ GOLD

Story ID (office use)

Version 1.11

Details	Hours
- Fix the bug on the signing up process	4
- Make the confirmation button bigger	2

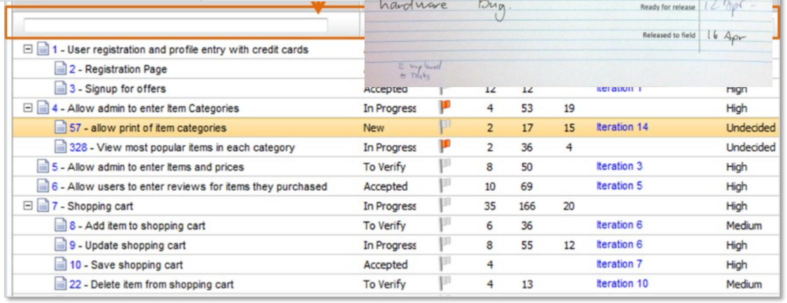
www.ssw.com.au/ssw/standards/rules/RulesToBetterPro



FDD 2012/04/09 Monday

As a developer I want to know a detailed reason for slave clock reset so that I can identify our rev 4 hardware bug.

Dates	
Planned	4 Apr
Dev start	4 Apr
Dev done	10 Apr
Demo'ed	11 Apr
UAT	12 Apr
Ready for release	12 Apr
Released to field	16 Apr



1 - User registration and profile entry with credit cards							
2 - Registration Page							
3 - Signup for offers	Accepted		12	12		Iteration 1	High
4 - Allow admin to enter Item Categories	In Progress		4	53	19		High
57 - allow print of item categories	New		2	17	15	Iteration 14	Undecided
328 - View most popular items in each category	In Progress		2	36	4		Undecided
5 - Allow admin to enter Items and prices	To Verify		8	50		Iteration 3	High
6 - Allow users to enter reviews for items they purchased	Accepted		10	69		Iteration 5	High
7 - Shopping cart	In Progress		35	166	20		High
8 - Add item to shopping cart	To Verify		6	36		Iteration 6	Medium
9 - Update shopping cart	In Progress		8	55	12	Iteration 6	High
10 - Save shopping cart	Accepted		4			Iteration 7	High
22 - Delete item from shopping cart	To Verify		4	13		Iteration 10	Medium

Test cases as Requirements

De facto practice

- Company A: Medium-sized, Networking equipment

Tool-supported **Behaviour-driven development**

- Company B: Small, Consultants

Story-test driven / Stand-alone with manual test cases

- Company C: Large, Telecom

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.

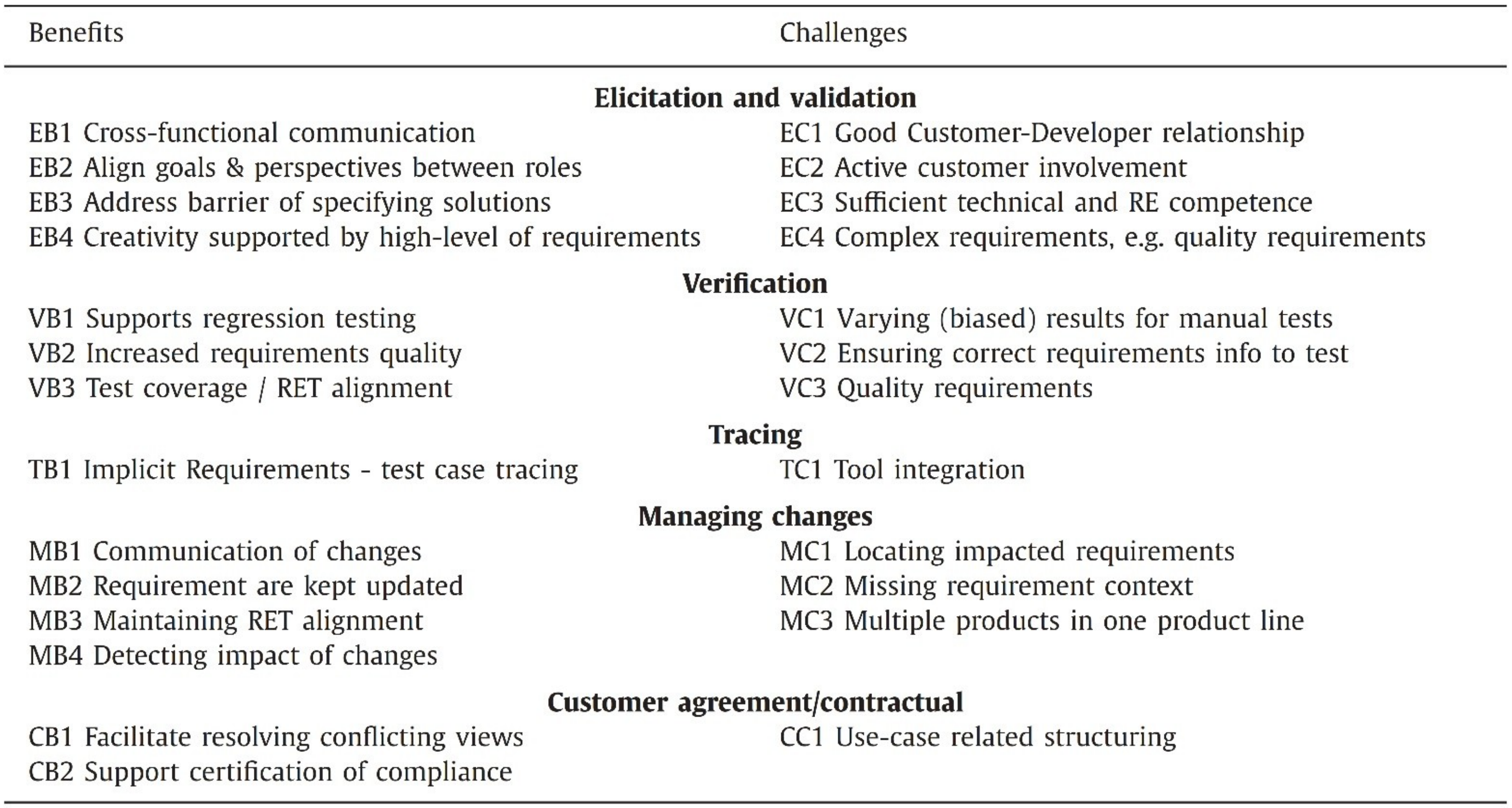
Test cases as Reqts: Variation points

- **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**

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.

Test Cases as Requirements in Agile practice



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	

Table 7 in ATCR.pdf (optional paper in zip not included in exam)]

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.

Face-to-face communication

Direct communication between customer and development

- Techniques
  - User Stories == high-level requirements spec
  - Complemented by other artifacts, 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

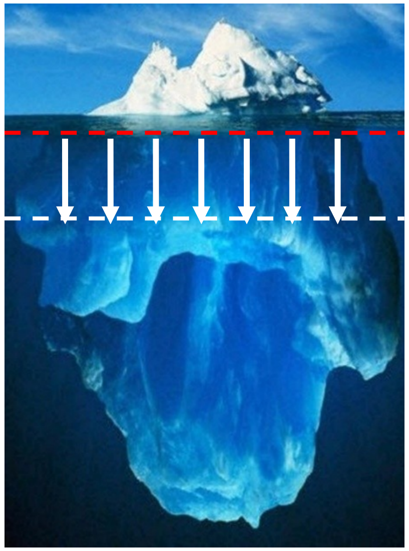
Face-to-face Communication

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

- High-level requirement analysis is carried out at beginning of project.
- **development team acquires a high-level understanding of the critical features of the application**, rather than creating a detailed specification of requirements.
- Requirements are **not intended to be complete or cover all features**.
- Requirements serve as a **starting point to plan the initial release cycle**.
- **As more is getting known, more** features/user stories are added.
- Techniques:
  - **Start of development cycle**, the customer sits down with the development team to provide detailed information on a set of features that need to be implemented. During this process, requirements are discussed at a greater level of detail.
  - Requirements analysis is intertwined with the design activity.

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

- Implement the features with the highest priority early in the project so that the **customers can realize most of the potential business value**.
- During development, the customer's as well as the developer's **understanding of the project improves and new requirements are added**, or existing requirements are modified.
- To **keep priorities up to date**, prioritization & Planning **is repeated frequently** during the entire development process.

Constant Planning

- A core principle - adapt / react quickly to changes
- a way to better satisfy customer needs.
- Before implementing a feature, conversation with customer about needs == requirements

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

- way to communicate with their customers.
- Validation and elicitation (of details)
- Risks of using production software
  - **Push to deploy** prototypes rather than **release quality code**

Reviews and Acceptance Tests

- review meetings / interaction for **requirements validation**.
- Before and during implementation.
- End of sprint/iteration: Sign-off with all stakeholders
  - demo with feedback
  - Show project status (schedule & quality), builds trust

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

Pros & Cons of Agile Development

Strengths

- quickly delivers working increments
- avoids unnecessary overhead
- short communication paths
- feedback from early stages used in developing latter stages

Weaknesses

- weak long-term and overall perspective
- weak / missing documentation
- weaker specialist competence
- less structure/guidance for weaker engineers

https://en.wikipedia.org/wiki/Agile_software_development#Criticism

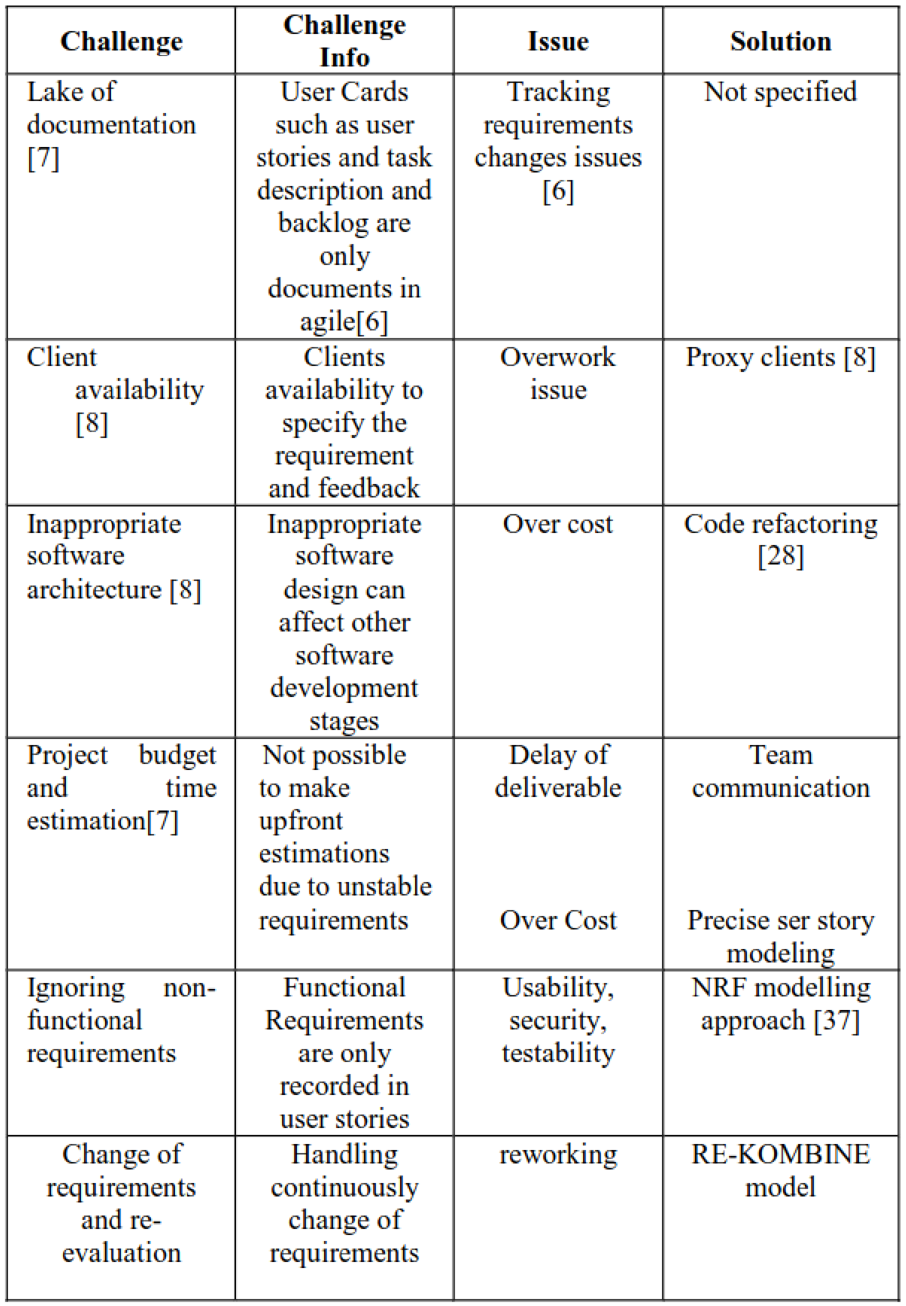
AGILE REQUIREMENTS ENGINEERING CHALLENGES

Elghariani, Kaiss, and Nazri Kama.

"Review on Agile requirements engineering challenges."

2016 3rd International conference on computer and information sciences (ICCOINS). IEEE, 2016.

TABLE 2 SUMMARY OF AGILE REQUIREMENTS ENGINEERING CHALLENGES



Challenge	Challenge Info	Issue	Solution
Lake of documentation [7]	User Cards such as user stories and task description and backlog are only documents in agile[6]	Tracking requirements changes issues [6]	Not specified
Client availability [8]	Clients availability to specify the requirement and feedback	Overwork issue	Proxy clients [8]
Inappropriate software architecture [8]	Inappropriate software design can affect other software development stages	Over cost	Code refactoring [28]
Project budget and time estimation[7]	Not possible to make upfront estimations due to unstable requirements	Delay of deliverable Over Cost	Team communication Precise ser story modeling
Ignoring non-functional requirements	Functional Requirements are only recorded in user stories	Usability, security, testability	NRF modelling approach [37]
Change of requirements and re-evaluation	Handling continuously change of requirements	reworking	RE-KOMBINE model

Computers in Human Behavior 51 (2015) 915–929

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Review

A systematic literature review on agile requirements engineering practices and challenges

Irum Inayat [a,*], Siti Salwah Salim [a], Sabrina Marczak [b], Maya Daneva [c], Shahaboddin Shamshirband [d,e]

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 (NFRs)	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)

Some more papers on Agile RE challenges

Alsaqaf, Wasim, Maya Daneva, and Roel Wieringa. "Agile quality requirements engineering challenges: First results from a case study." 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, 2017.

Bjarnason, Elizabeth, Krzysztof Wnuk, and Björn Regnell. "A case study on benefits and side-effects of agile practices in large-scale requirements engineering." proceedings of the 1st workshop on agile requirements engineering. 2011.

Schön, Eva-Maria, et al. "Key challenges in agile requirements engineering." Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings 18. Springer International Publishing, 2017.

Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., & de Oliveira Neto, F. G. (2021). Requirements engineering challenges and practices in large-scale agile system development. Journal of Systems and Software, 172, 110851.

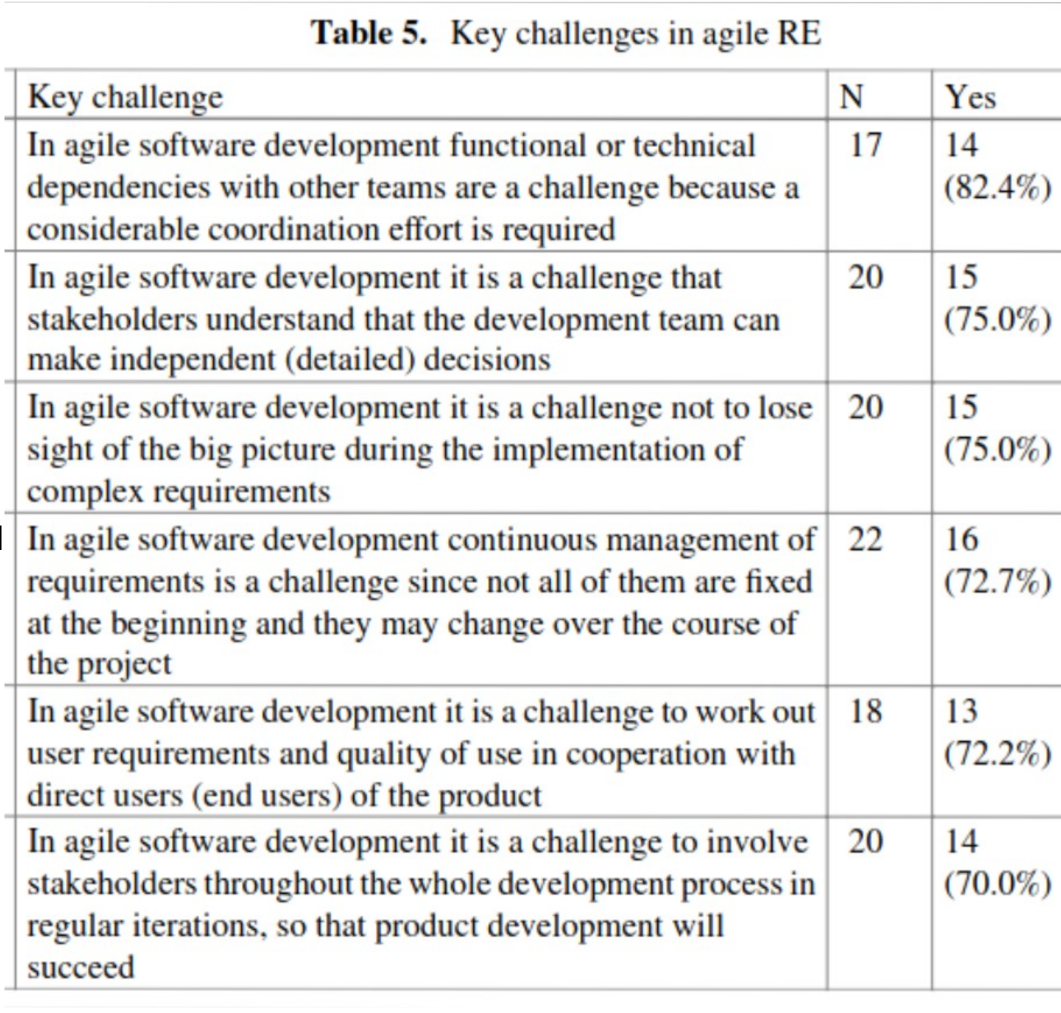


Table 5. Key challenges in agile RE

Key challenge	N	Yes
In agile software development functional or technical dependencies with other teams are a challenge because a considerable coordination effort is required	17	14 (82.4%)
In agile software development it is a challenge that stakeholders understand that the development team can make independent (detailed) decisions	20	15 (75.0%)
In agile software development it is a challenge not to lose sight of the big picture during the implementation of complex requirements	20	15 (75.0%)
In agile software development continuous management of requirements is a challenge since not all of them are fixed at the beginning and they may change over the course of the project	22	16 (72.7%)
In agile software development it is a challenge to work out user requirements and quality of use in cooperation with direct users (end users) of the product	18	13 (72.2%)
In agile software development it is a challenge to involve stakeholders throughout the whole development process in regular iterations, so that product development will succeed	20	14 (70.0%)