





# Requirements Engineering in Agile Software Development

Author: Andrea De Lucia and Abdallah Qusef  
Presenter: Tyler, Tyler, Krishna, Patrick, Hunter



# Problem

- Traditional structures begin with documentation
- Requirements change too fast
- Customers don't know their needs
- Traceability in Agile is ill defined

Has Your Organization Adopted One or More Agile Techniques?

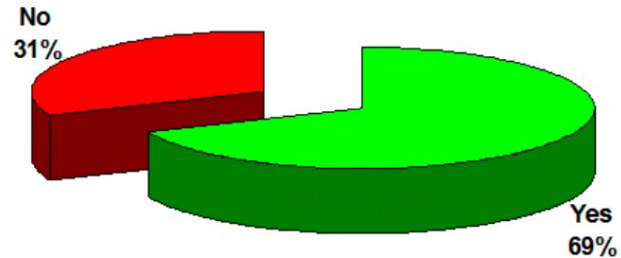


Figure 1 Agile Development Adoption

# Addressing the Problem

Agile Manifesto:

1. Individuals and interaction over processes and tools
2. Working software over documentation
3. Customer collaboration over contract negotiation
4. Responding to changes over following a plan

Requirements Engineer:

1. Focus on the process of requirements

# Agile Definition

James Highsmith describes anything being Agile is being able to “Deliver quickly. Change quickly. Change often.”

There are 11 key principles of the Agile process:

- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Customer satisfaction by rapid, continuous delivery of useful software
- Even late changes in requirements are welcomed
- Close daily cooperation between business people

# Agile Definition cont.

- Face-to-face conversation is the best form of communication
- Projects are built around motivated individuals who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

# Guidelines to Improve RE and Agile Relationship:

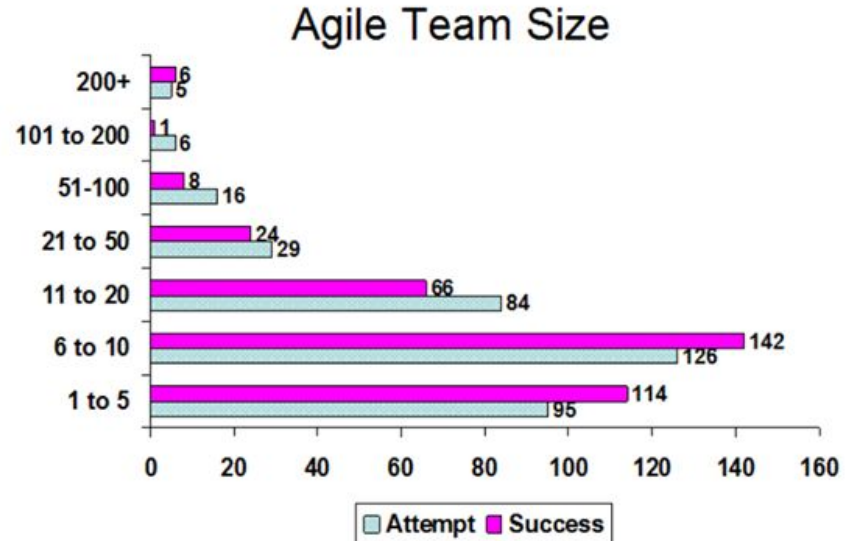
1. Customer Involvement
2. Agile Project Contracts
3. **Frequent Releases**
4. Requirements Elicitation Language
5. Non-functional Requirements
6. **Smaller Agile Teams are Flexible**
7. Evolutionary Requirements
8. **No Early Documentation**
9. Requirements Splitting
10. **Requirements Traceability**

# Interesting Part - Frequent Releases

- Faster releases speed up incremental process
- Get user feedback faster -> resolve feedback faster
- Better working product by end

## Interesting Part - Smaller Agile Teams are Flexible

- Teams are more flexible
- Continuous communication between team and stakeholder
- Have more successful results





## Interesting Part - No Early Documentation

- Quickly become irrelevant
- 5%-15% of the resources spent on requirements
- Address shortcomings in agile development while complying with the agile principles in general.



**PAPERWORK**

You'll never be able to avoid it,  
even if you're reincarnated as a cat.

# Advantage

Clear bulleted points in improving the relationship between requirements engineering and agile environment.

(Customer Involvement, Project Contracts, Frequent releases, Requirements Elicitation Language, Non-Functional Requirements, Smaller Teams, Evolutionary Requirements, Not Documenting Early, Splitting Requirements, and Requirement Traceability)

Sampled different sizes of agile teams to confirm successes on projects

Used a variety of methods to evaluate requirements in terms of ambiguity, clarity, etc

Elicitation -> Analysis -> Documentation -> Validation -> Management.

# Disadvantage

Lacks industrial case studies that support the ideas

Lacks empirical evidence supporting the distinction between functional and non-functional requirements

# Limitations

Agile only really works for smaller to mid sized teams, not larger teams

Development methods under agile do not scale (many iterations blurs overall project status)

Agile thrives on fast and motivated developers

Documentation is not easy due to the rapidly changing nature of agile

# Relevance

- Background into agile process
- Agile is a widely used process mode
- Become better developers
- Applicable to our group project

# Source

<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=49a4dc3491cc9917f5957473b5db2775a532d780>