Enhancing the designing and development of large-scale and complex softwares

Philippe GUEMKAM SIMO
MIKS 2018 - 2019

Supervisor:
Manuele Kirsch Pinheiro

Summary

Introduction & Research Problem

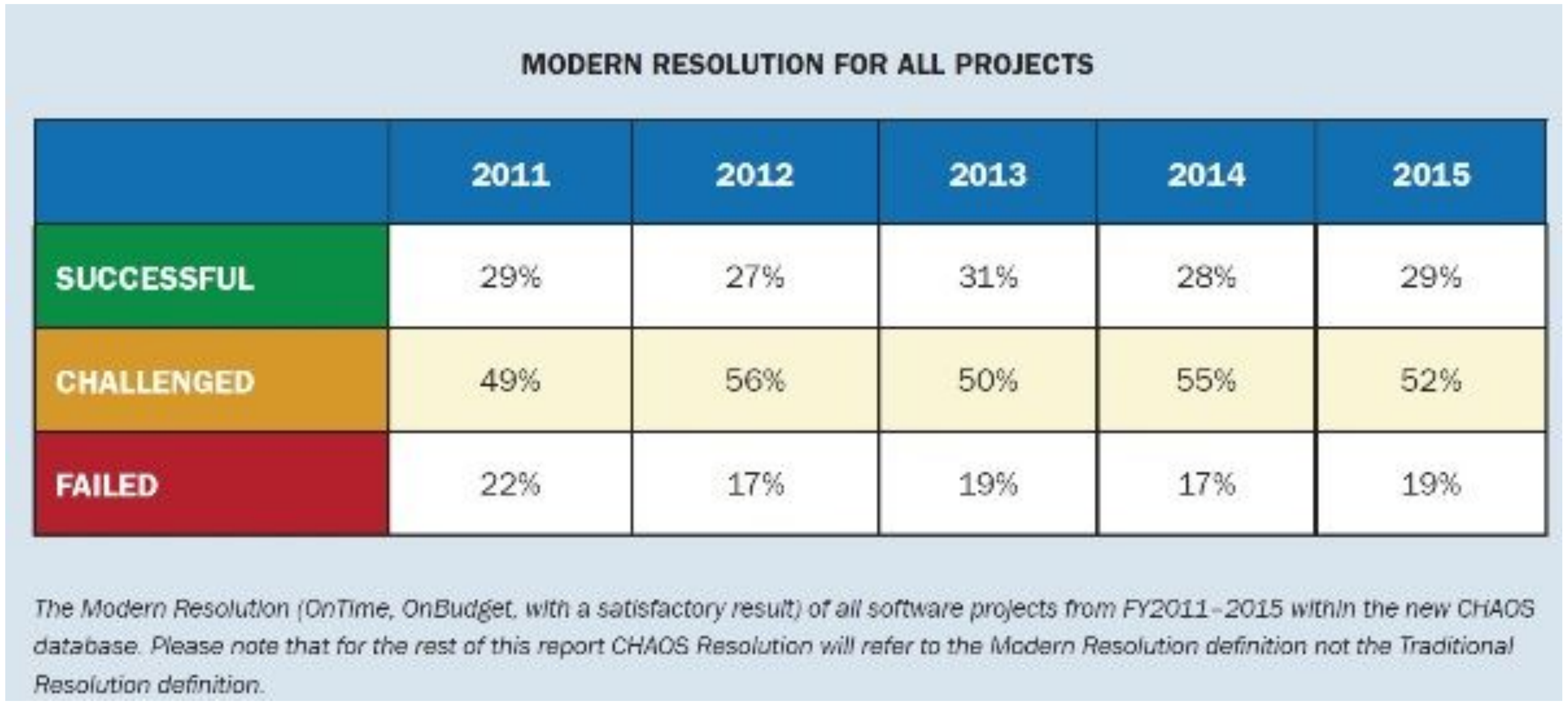
Large-scale and complex software

Critical business domain

Distributed architecture

Millions of users 24/7

Introduction & Research Problem



MODERN RESOLUTION FOR ALL PROJECTS

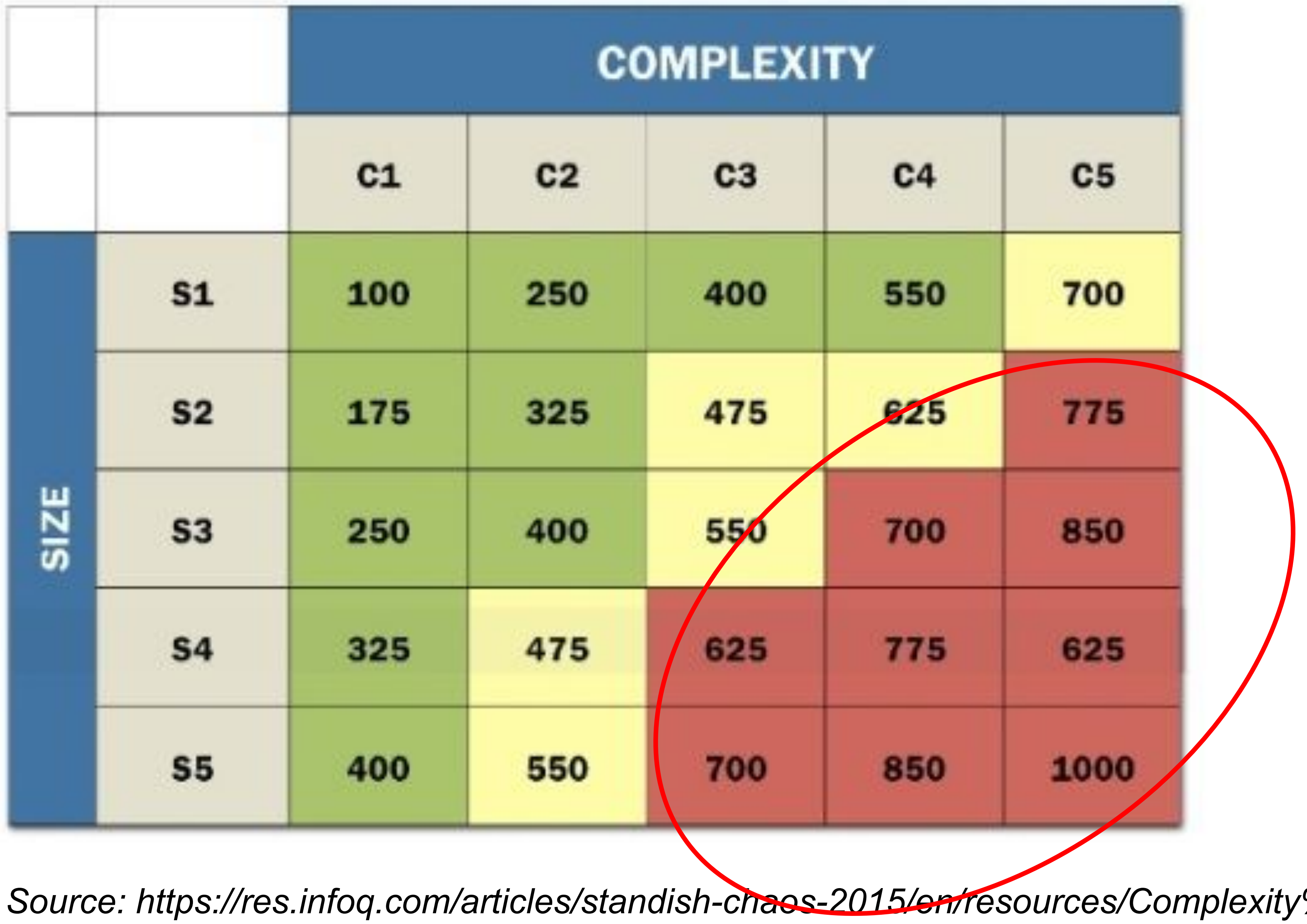
	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Projects still fail or are challenged

Source: https://res.infoq.com/articles/standish-chaos-2015/en/resources/Modern%20Resolution.jpg

Introduction & Research Problem



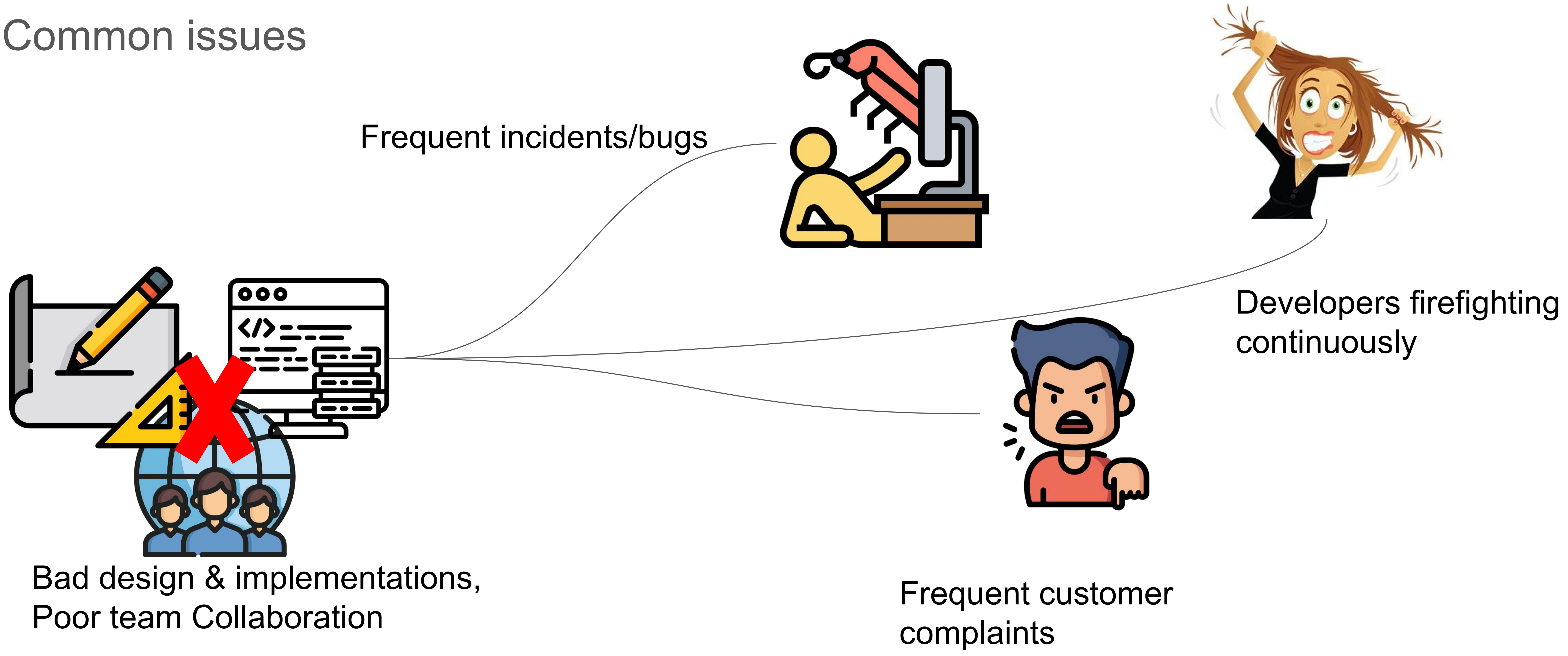
		COMPLEXITY				
		C1	C2	C3	C4	C5
SIZE	S1	100	250	400	550	700
	S2	175	325	475	625	775
	S3	250	400	550	700	850
	S4	325	475	625	775	625
	S5	400	550	700	850	1000

Source: https://res.infoq.com/articles/standish-chaos-2015/en/resources/Complexity%20Matrix.jpg

The greater the complexity and scale, the greater the risk of failure.

Introduction & Research Problem

Common issues


Frequent incidents/bugs
Developers firefighting continuously
Bad design & implementations, Poor team Collaboration
Frequent customer complaints

Introduction & Research Problem

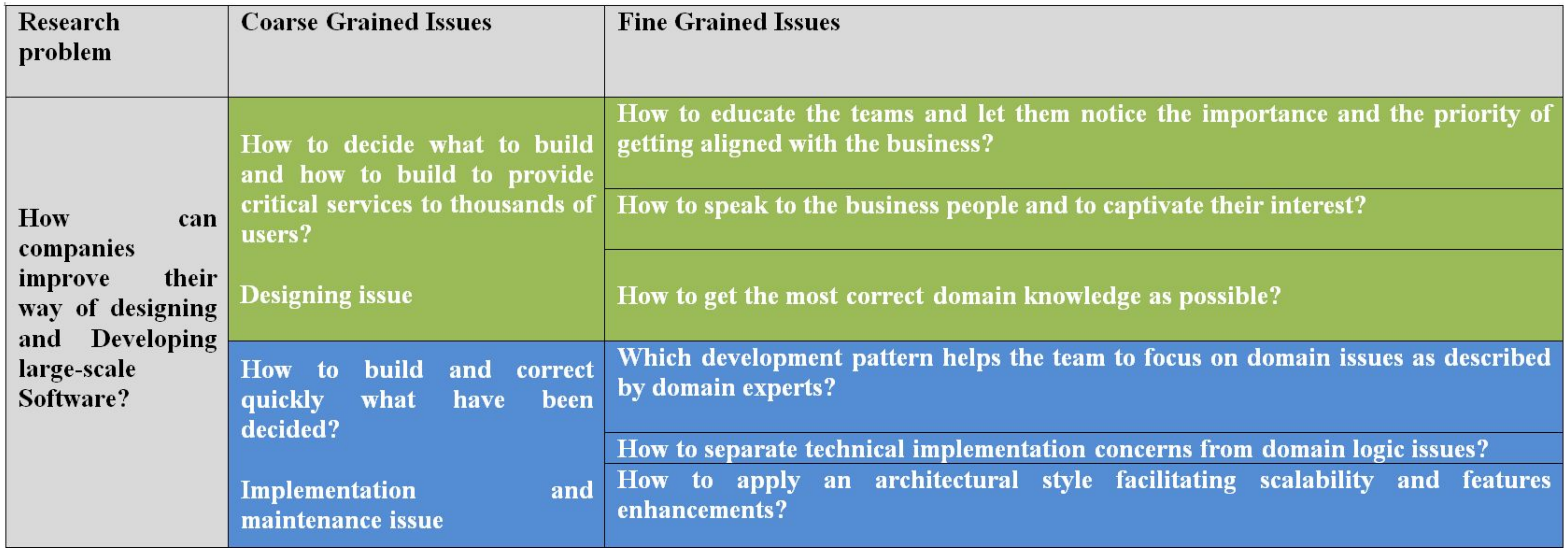
Coarse-grained research questions :

How to **improve the designing and the development** of large-scale and complex softwares?

- **Designing issue:** How to **decide what to build and how to build** to provide critical services to thousands of users?

- **Implementation and maintenance issue:** How to **build and correct quickly** what have been decided?

Introduction & Research Problem

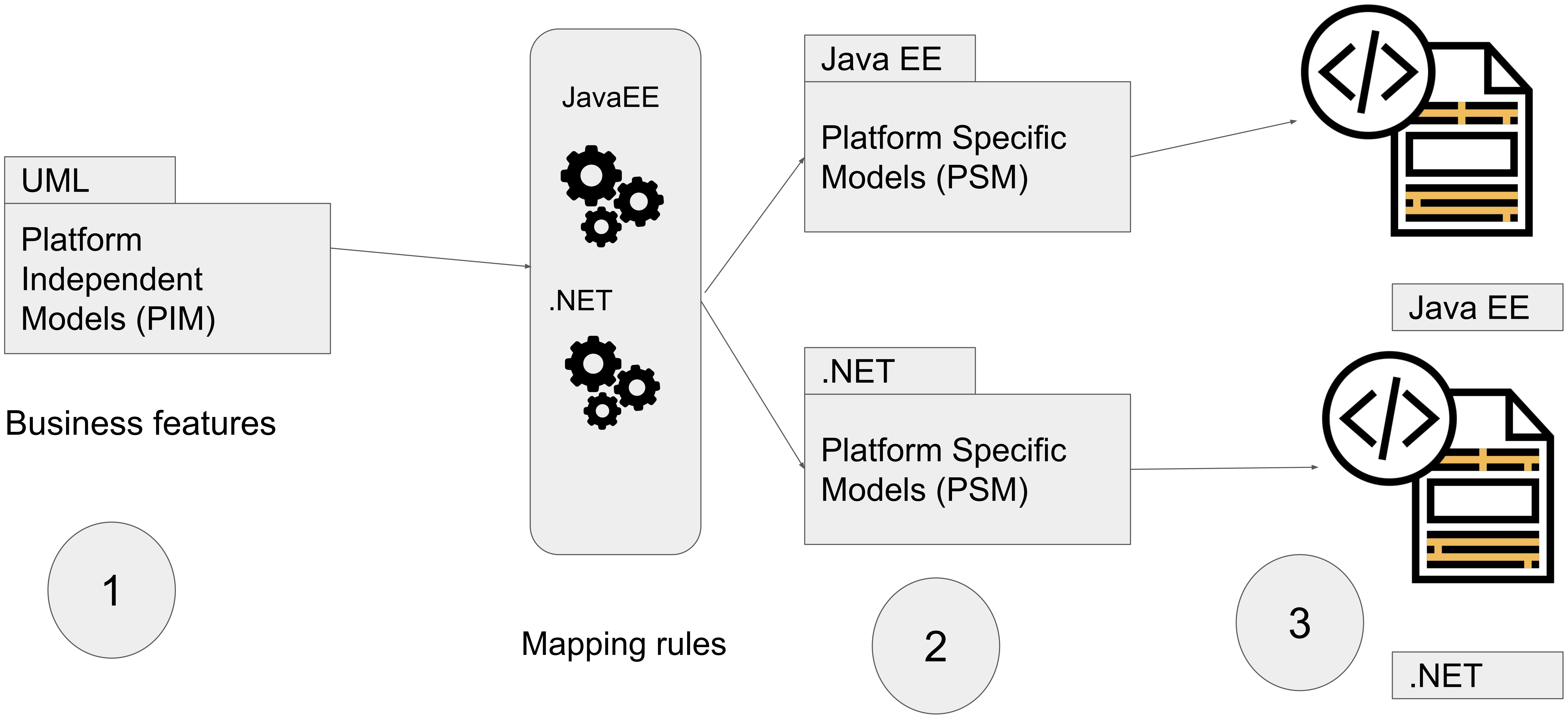


Research problem	Coarse Grained Issues	Fine Grained Issues
How can companies improve their way of designing and Developing large-scale Software?	How to decide what to build and how to build to provide critical services to thousands of users? Designing issue	How to educate the teams and let them notice the importance and the priority of getting aligned with the business?
		How to speak to the business people and to captivate their interest?
		How to get the most correct domain knowledge as possible?
	How to build and correct quickly what have been decided? Implementation and maintenance issue	Which development pattern helps the team to focus on domain issues as described by domain experts?
		How to separate technical implementation concerns from domain logic issues?
		How to apply an architectural style facilitating scalability and features enhancements?

State of art : Designing and development methodologies

- Model Driven Architecture (MDA)
- Behavior Driven Design (BDD)
- Domain Driven Design (DDD)

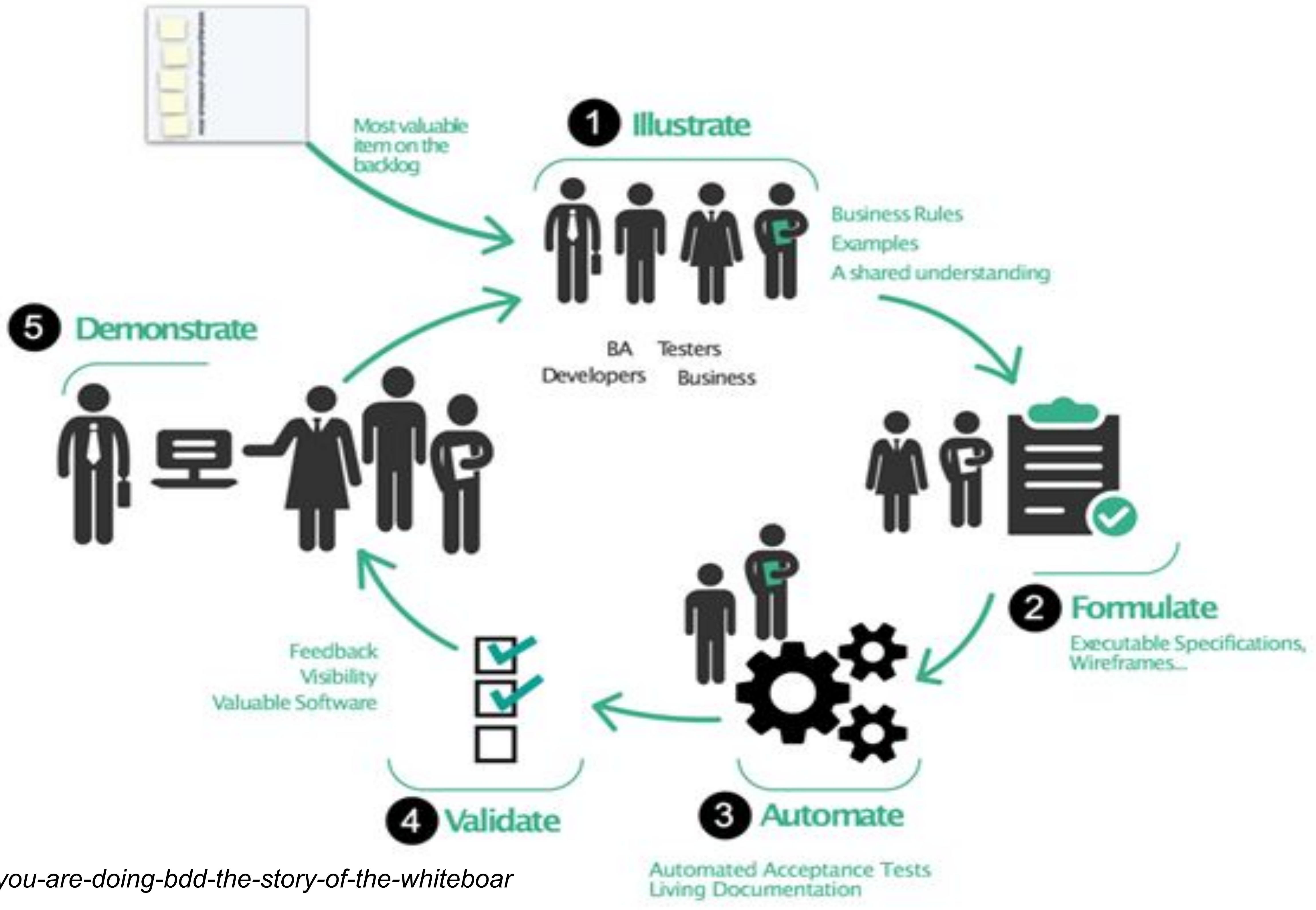
State of art : MDA


UML
Platform Independent Models (PIM)
Business features
1
JavaEE
.NET
Mapping rules
Java EE
Platform Specific Models (PSM)
.NET
Platform Specific Models (PSM)
2
Java EE
3
.NET

State of art : MDA

- Domain experts focus on specifying domain issues
- Developers focus on transformation and implementation issues
- Clear separation between technical concerns and domain logic in application code
- Difficulties when customizing the generated code, leading to inconsistency with the models

State of art : BDD

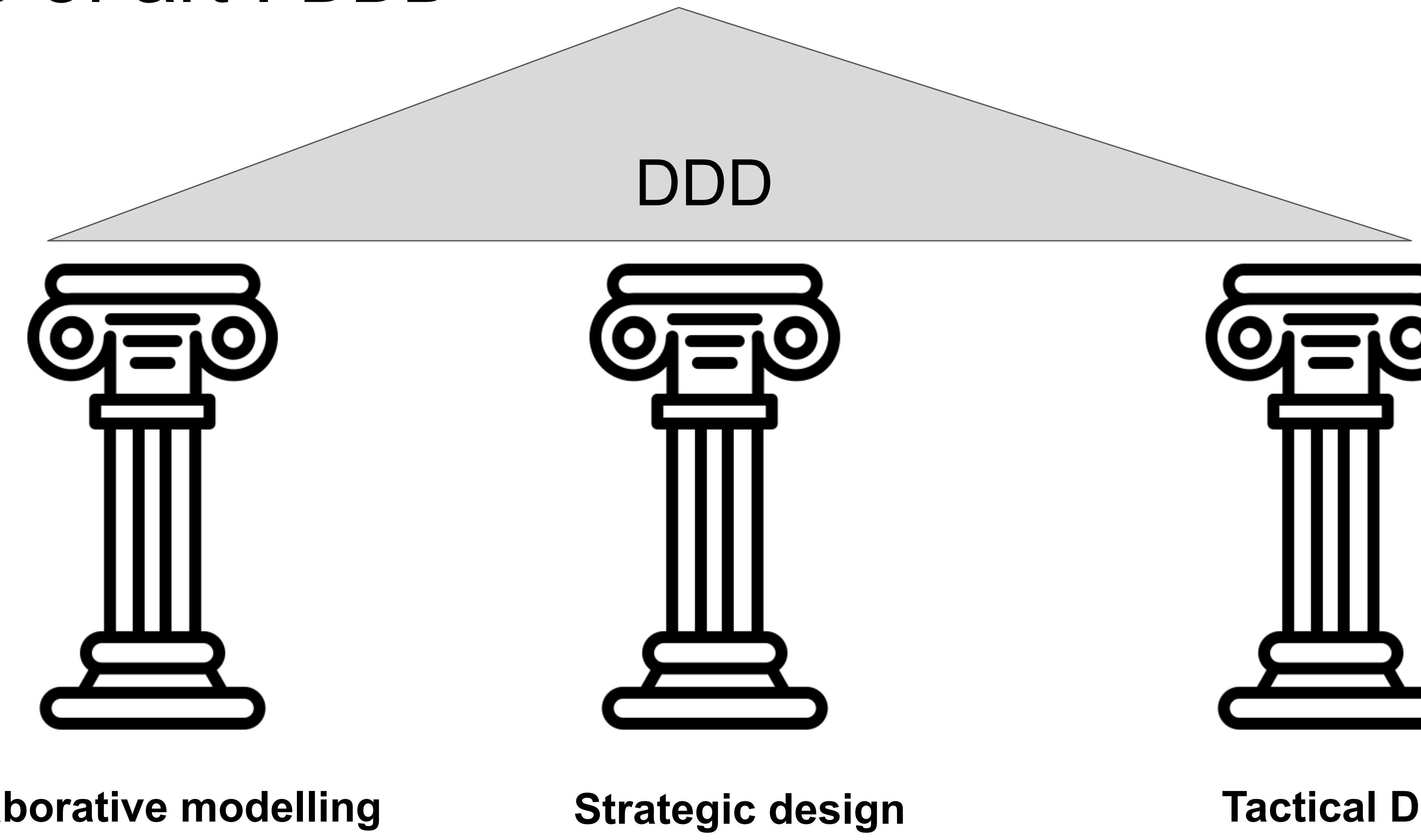


Source : https://johnfergusonsmart.com/so-you-say-you-are-doing-bdd-the-story-of-the-whiteboard-and-the-nail-gun/

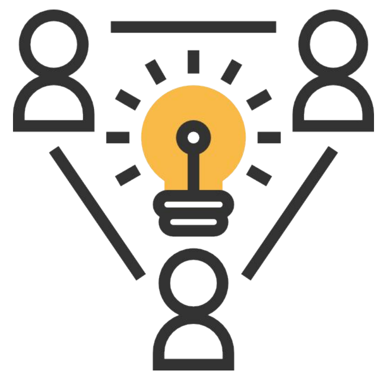
State of art : BDD

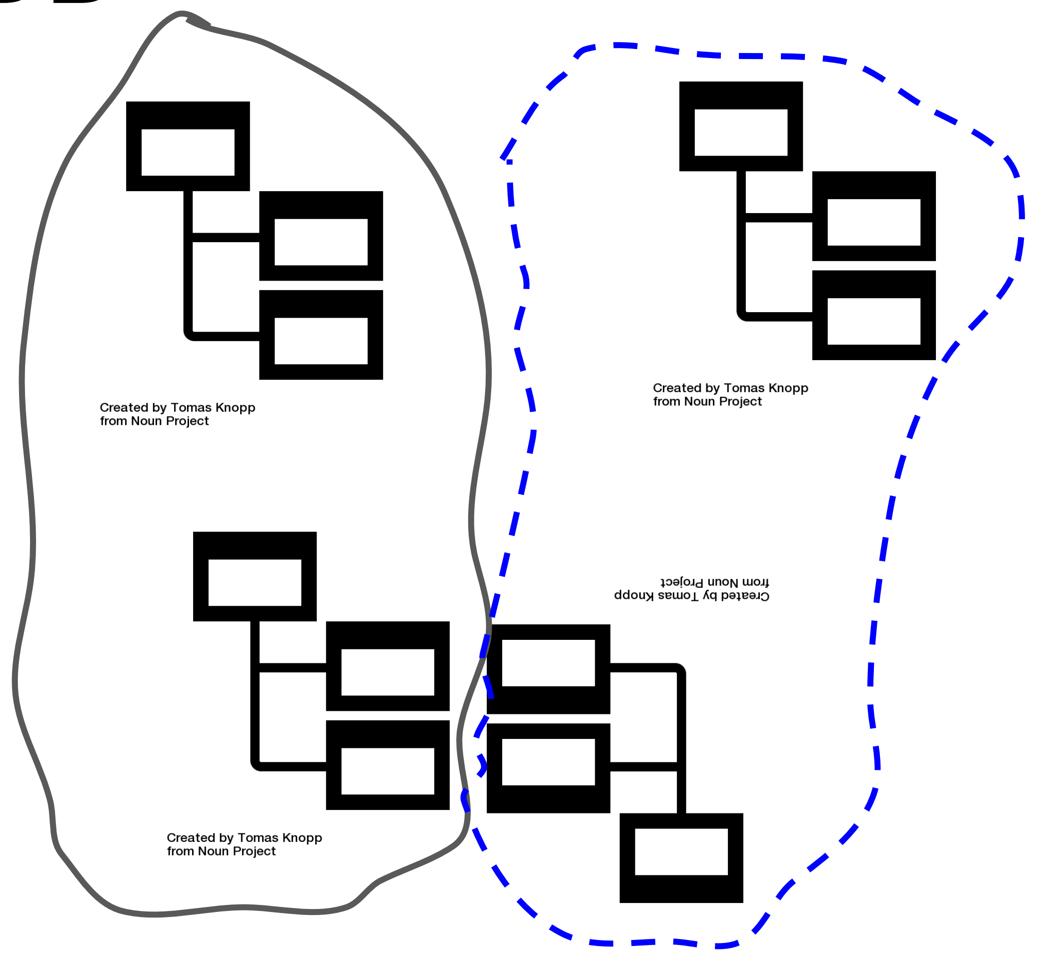
- Enhances the collaboration between developers team and stakeholders
- Improve software quality with respect to meeting requirements
- Editing scenario files for automated tests is time consuming
- Requires intensive communication between people editing features and test code writers

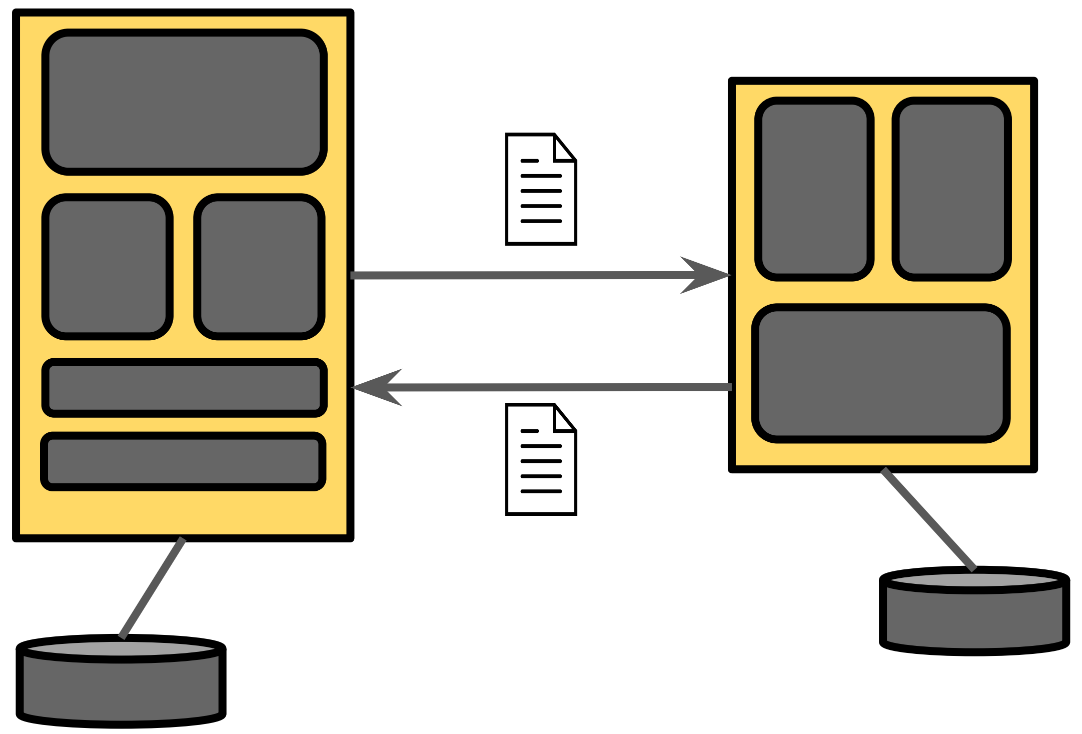
State of art : DDD



State of art : DDD




Created by Tomas Knopp from Noun Project
Created by Tomas Knopp from Noun Project
Created by Tomas Knopp from Noun Project
Created by Tomas Knopp from Noun Project



Collaborative modelling:

Event storming

Ubiquitous Language

Strategic design:

Bounded contexts

UL based domain models

Tactical Design:

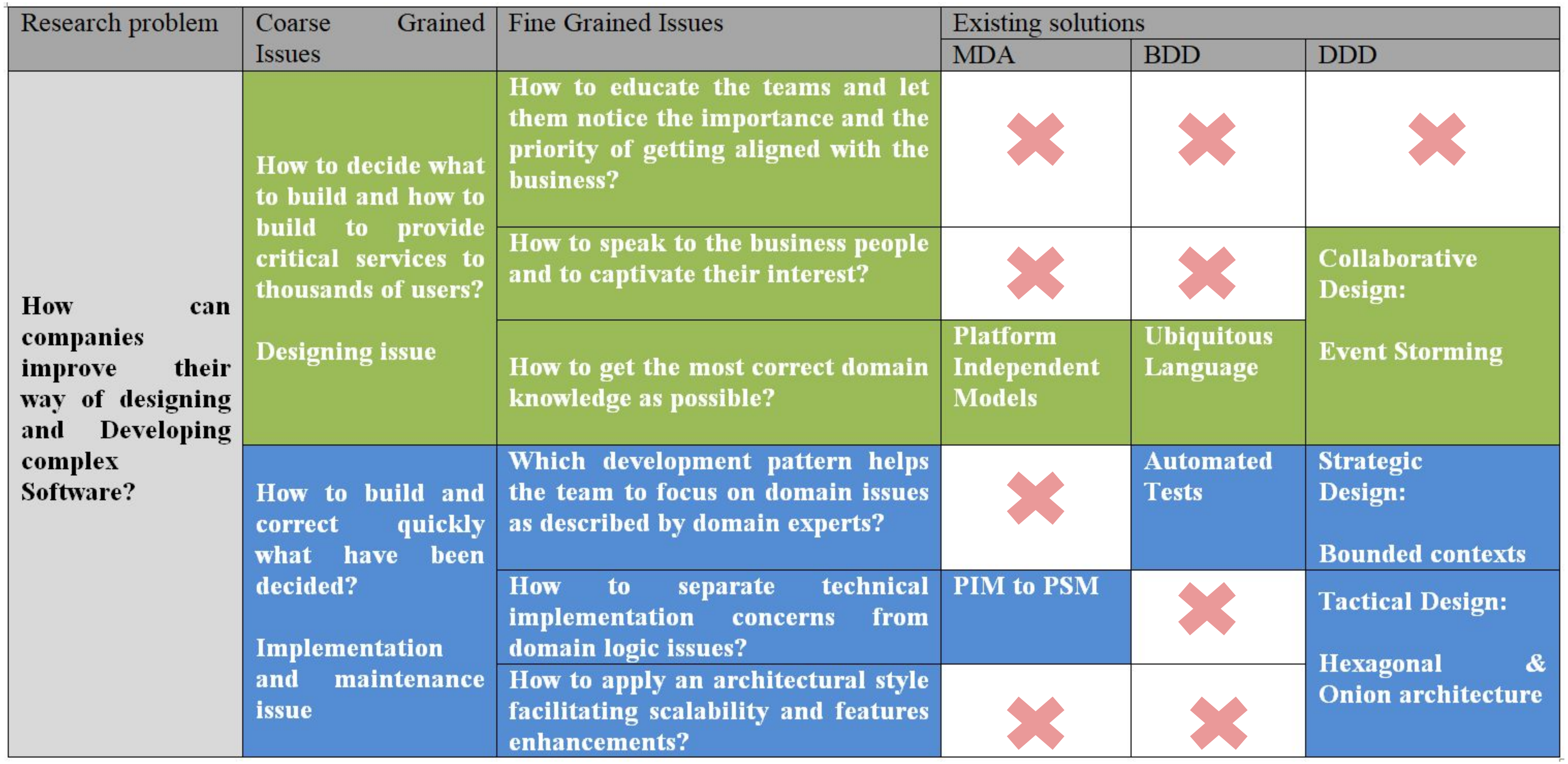
Onion architecture

Dependency Inversion Principle

State of art : DDD

- Focus on the knowledge of the domain
- Give useful ways of understanding the subject matter
- Suggest useful ways of implementing scalable solution
- Lot of efforts to implement and time consuming
- Very easy to do it wrong
- Domain experts expensive to hire

State of art: Gaps to be filled



Research problem	Coarse Grained Issues	Fine Grained Issues	Existing solutions		
			MDA	BDD	DDD
How can companies improve their way of designing and Developing complex Software?	**How to decide what to build and how to build to provide critical services to thousands of users?** **Designing issue**	**How to educate the teams and let them notice the importance and the priority of getting aligned with the business?**	✗	✗	✗
		How to speak to the business people and to captivate their interest?	✗	✗	**Collaborative Design:** **Event Storming**
		How to get the most correct domain knowledge as possible?	**Platform Independent Models**	**Ubiquitous Language**	
	How to build and correct quickly what have been decided? **Implementation and maintenance issue**	**Which development pattern helps the team to focus on domain issues as described by domain experts?**	✗	**Automated Tests**	**Strategic Design:** **Bounded contexts**
		How to separate technical implementation concerns from domain logic issues?	**PIM to PSM**	✗	**Tactical Design:** **Hexagonal & Onion architecture**
		How to apply an architectural style facilitating scalability and features enhancements?	✗	✗	

Research Methodology

Semi-structured interviews about **previous and current experiences**

- 1 Developer (exp: > 3)
- 2 Tech leaders (exp: > 10)
- 1 Architect (exp > 10)
- 2 business analyst (exp > 5)
- 1 domain expert (exp > 5)

Qualitative horizontal analysis

Aim: Discover **complementary insights** in order to **suggest guidelines**

Results

Activities vs. Roles:

Roles not corresponding to expected activities

Responsibilities are not quite defined nor applied

Usage of approaches:

BDD is the most used

DDD is acknowledged as necessary but not used

MDA is used implicitly

Knowledge on existing approaches:

Ad Hoc Knowledge

MDA, BDD, TDD, DDD

Feedback on approaches:

BDD enhances requirements elicitations

DDD enhances the value delivered to the client

Lack of knowledge management about previous experiences

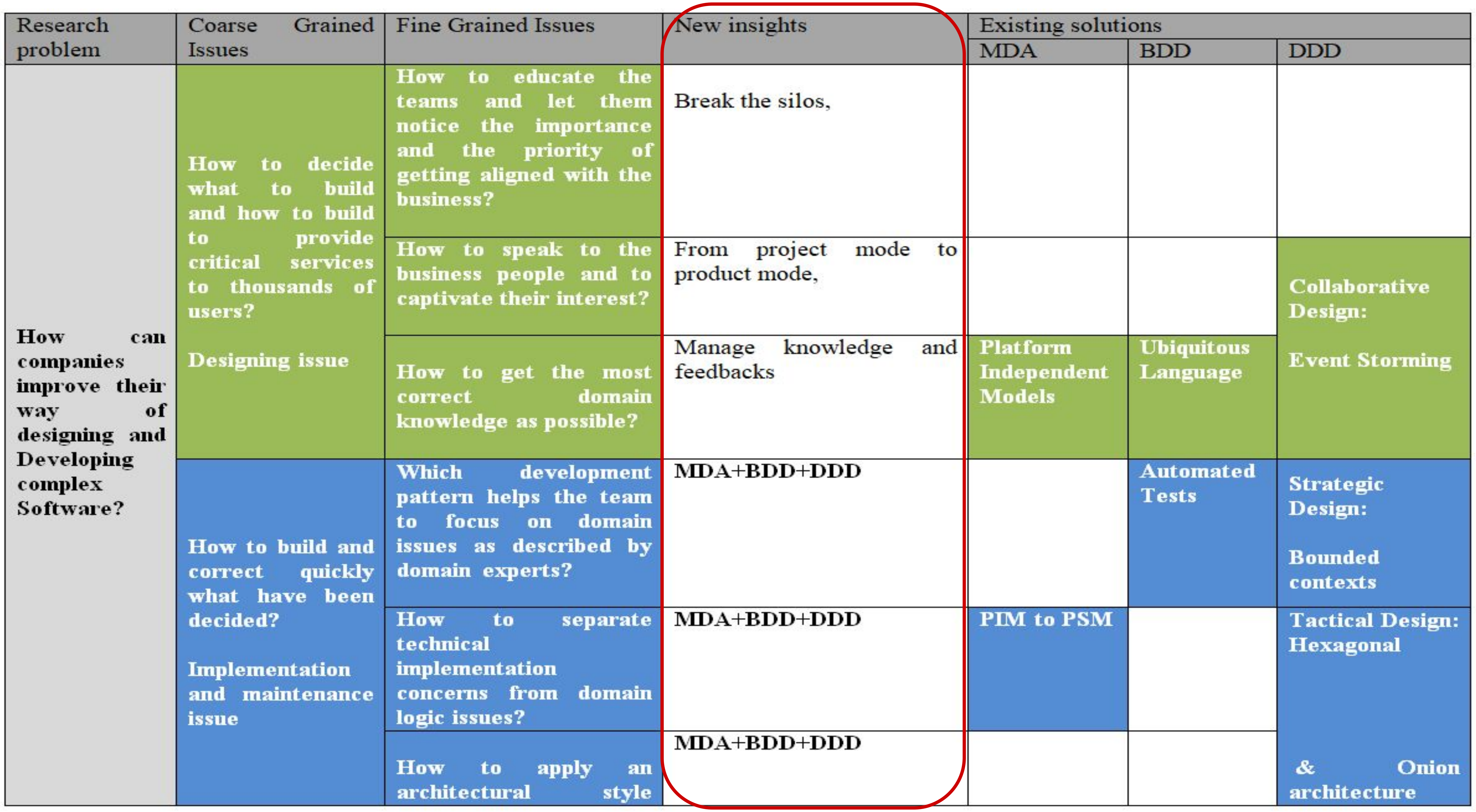
Results

Expectations:

Better interaction between business team and developer teams

Clear separation of responsibilities in the whole chain is necessary

Discussion



Research problem	Coarse Grained Issues	Fine Grained Issues	New insights	Existing solutions		
				MDA	BDD	DDD
How can companies improve their way of designing and Developing complex Software?	How to decide what to build and how to build to provide critical services to thousands of users? Designing issue	How to educate the teams and let them notice the importance and the priority of getting aligned with the business?	Break the silos,			
		How to speak to the business people and to captivate their interest?	From project mode to product mode,			Collaborative Design: Event Storming
		How to get the most correct domain knowledge as possible?	Manage knowledge and feedbacks	Platform Independent Models	Ubiquitous Language	
	How to build and correct quickly what have been decided? Implementation and maintenance issue	Which development pattern helps the team to focus on domain issues as described by domain experts?	MDA+BDD+DDD		Automated Tests	Strategic Design: Bounded contexts
		How to separate technical implementation concerns from domain logic issues?	MDA+BDD+DDD	PIM to PSM		Tactical Design: Hexagonal
		How to apply an architectural style	MDA+BDD+DDD			& Onion architecture

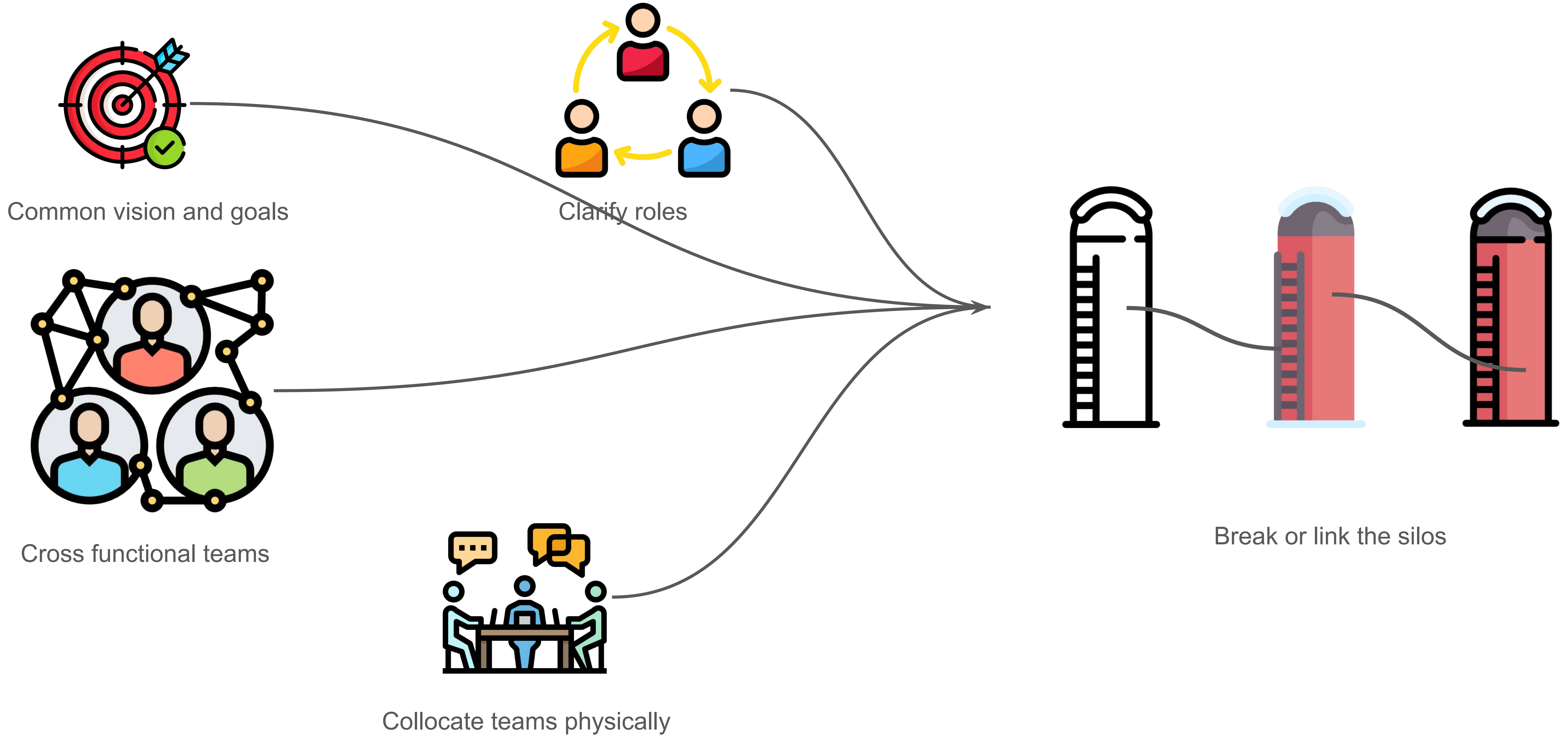
Discussion: Guidelines proposal

- Preparing the Organization

- Set up the Knowledge Management (KM) System

- **Designing and developing**

Discussion: Guidelines proposal - Preparing the Organization



Discussion: Guidelines proposal - Set up KM System

- Gather the knowledge management infrastructure informations
- Check KM mechanisms and technologies
- Enhance KM mechanisms and technologies

Discussion: Guidelines proposal - Design and development

- **Designing with respect to the requirements**

- Create the Ubiquitous Language
- Train the team on event storming
- Do event storming
- Define bounded contexts
- Construct PIM, translate to PSM

Discussion: Guidelines proposal - Design and development

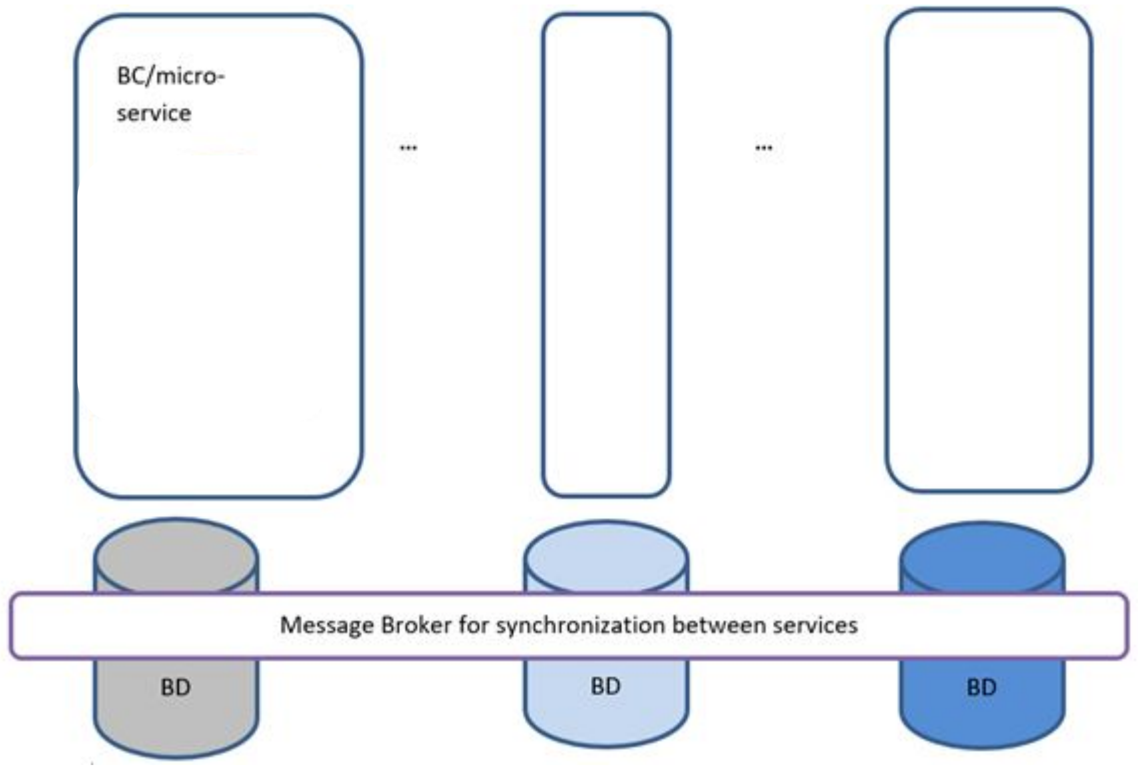
- **Implementing the requirements**

- Architecture: **Microservice over Onion**

- Assign developer to specific parts of the architecture with respect to complexity and working experience and skills

- Automated tests to synchronize PIM and PSM

- Master classes for architecture, conducted by an experienced architect



Assessment: AXA Bank IS Department

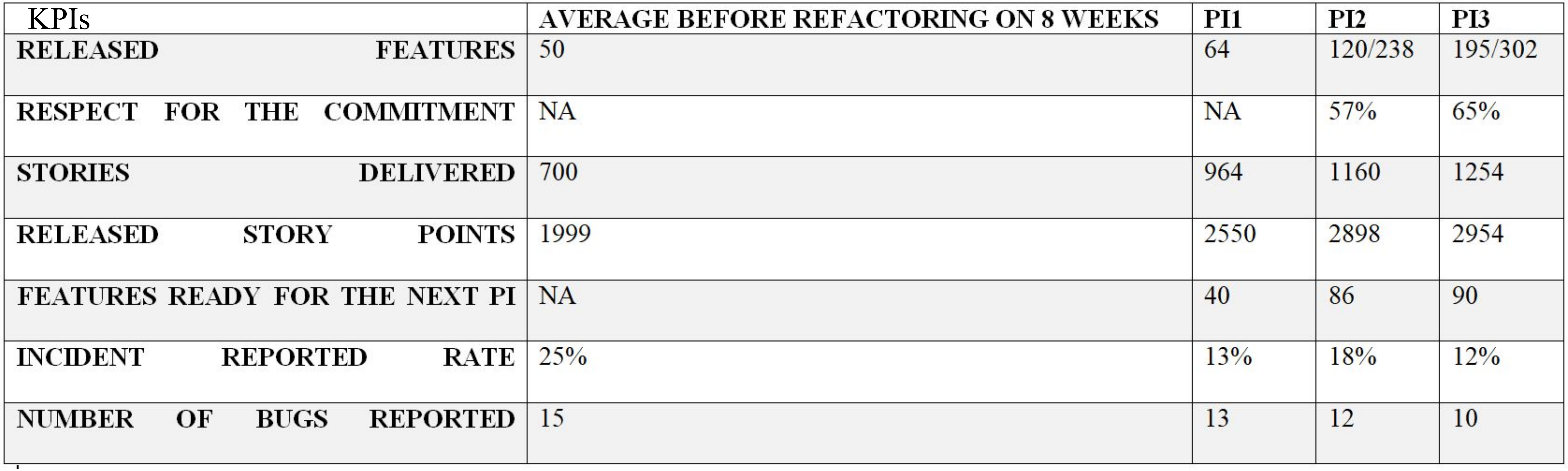
- Refactoring within the IS Department:
  From project mode to product mode with SAfe

- Reviewing KM Systems:
  Adding more mechanism for knowledge capture and sharing

- Training around event stormings

- Applying onion architecture on projects
  “Socle Applicatif de microservices chez Axa Banque: SAMA”

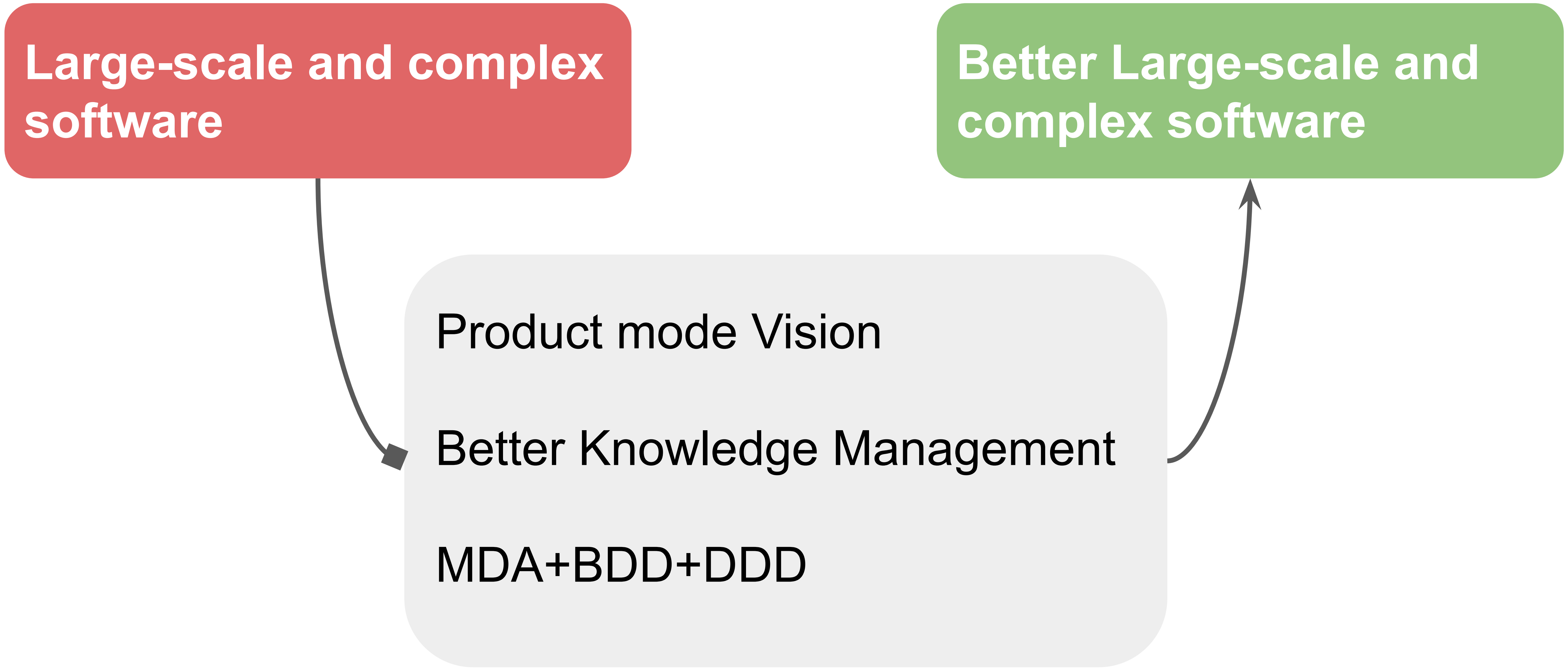
Assessment: AXA Bank IS Department



KPIs	AVERAGE BEFORE REFACTORING ON 8 WEEKS	PI1	PI2	PI3
RELEASED FEATURES	50	64	120/238	195/302
RESPECT FOR THE COMMITMENT	NA	NA	57%	65%
STORIES DELIVERED	700	964	1160	1254
RELEASED STORY POINTS	1999	2550	2898	2954
FEATURES READY FOR THE NEXT PI	NA	40	86	90
INCIDENT REPORTED RATE	25%	13%	18%	12%
NUMBER OF BUGS REPORTED	15	13	12	10

PI = One refactoring Program Increment (8 weeks)

Conclusion


Large-scale and complex software
Product mode Vision
Better Knowledge Management
MDA+BDD+DDD
Better Large-scale and complex software

⚠ To be assessed in other companies