Empirical Studies to Identify Best Practices for Addressing Recurring Concerns of Enterprise Architects and Solution Architects in Large-Scale Agile Development

Niklas Reiter, 02.09.2019, Guided Research - Final Presentation

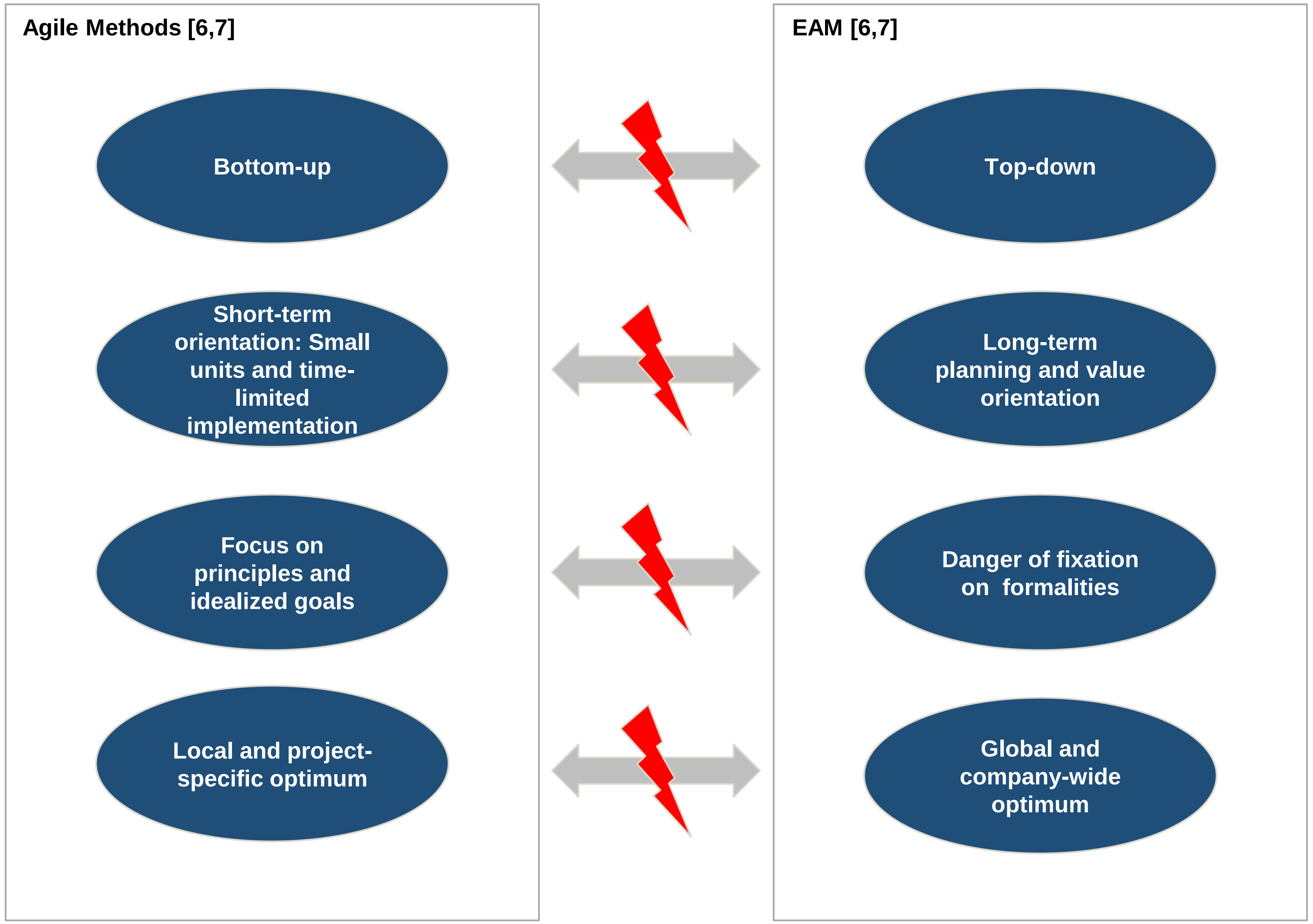
Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline

Motivation

Agile methods were originally designed for working at team level

Applying agile methods on **large-scale projects** leads to several **concerns** [8].

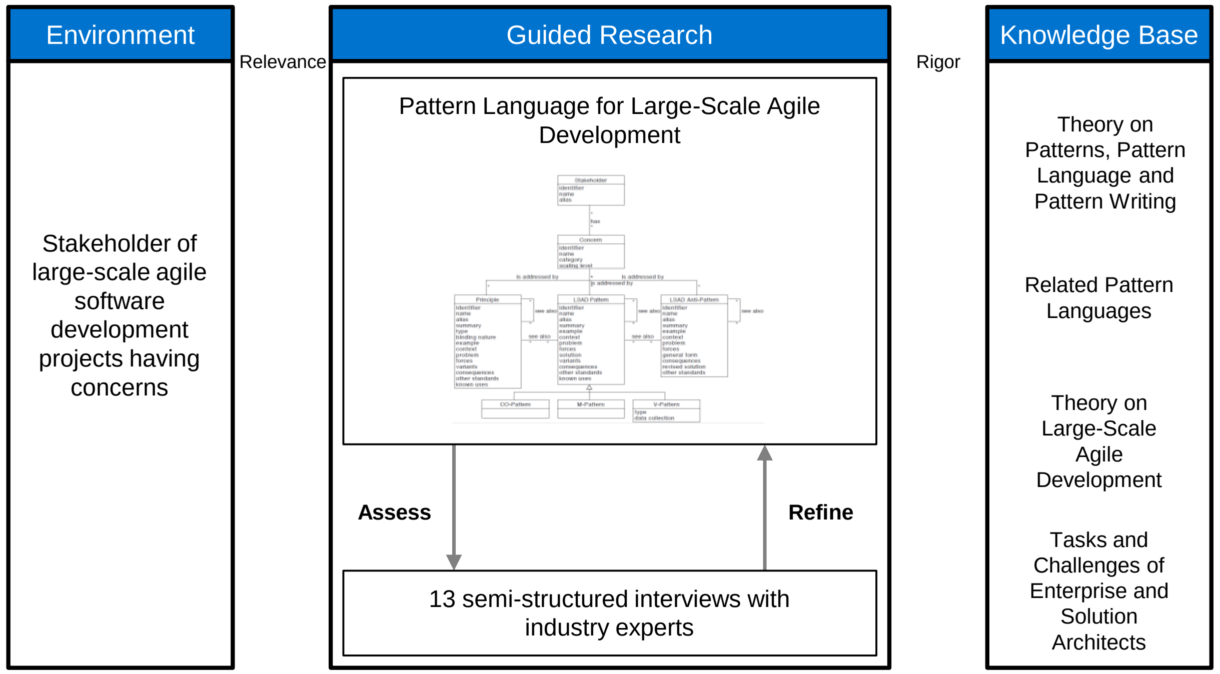


Agile Methods [6,7]
EAM [6,7]
Bottom-up
Top-down
Short-term orientation: Small units and time-limited implementation
Long-term planning and value orientation
Focus on principles and idealized goals
Danger of fixation on formalities
Local and project-specific optimum
Global and company-wide optimum

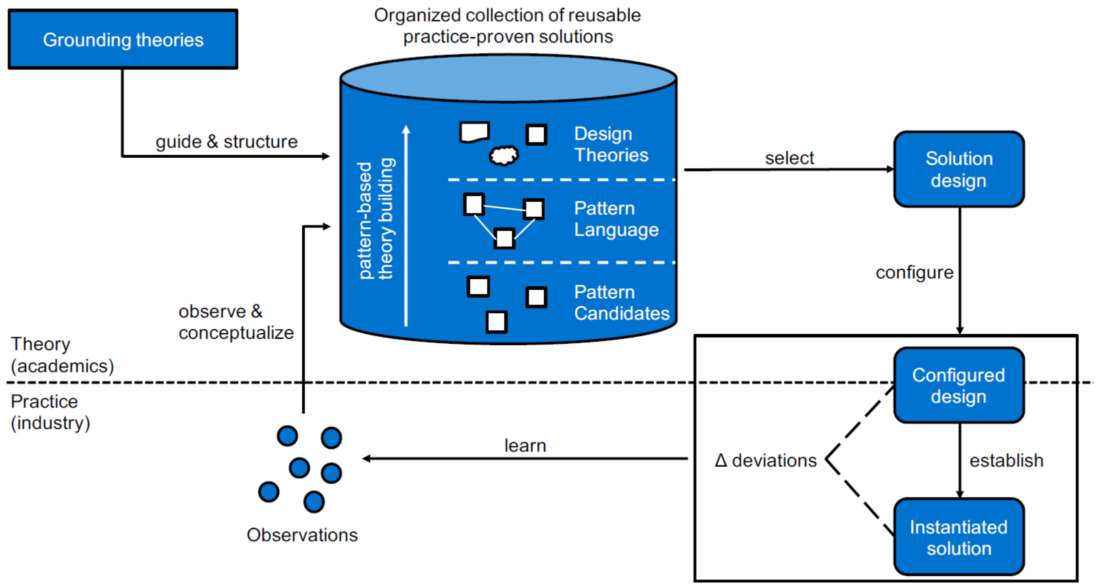
Outline

Research Methodology – Mixed-Methods Research Design

Design Science Approach [1]


Environment
Stakeholder of large-scale agile software development projects having concerns
Relevance
Guided Research
Pattern Language for Large-Scale Agile Development
Assess
Refine
13 semi-structured interviews with industry experts
Rigor
Knowledge Base
Theory on Patterns, Pattern Language and Pattern Writing
Related Pattern Languages
Theory on Large-Scale Agile Development
Tasks and Challenges of Enterprise and Solution Architects

Pattern-Based Research Design [3]



Structured Literature Review [12]
Deliverable: Identifying recurring concerns of stakeholders in large-scale-agile development

Case Study [10,13]
Deliverable: Investigating the role of enterprise architects in large-scale agile development and their collaboration with agile teams

Expert Interviews [2]
Deliverable: Identifying recurring concerns and best practices of enterprise and solution architects in large-scale agile development

Data Analysis
Deliverable: Analysis and validation of identified recurring concerns and good of enterprise and solution architects in large-scale agile development

Report
Deliverable: 35 recurring concerns, 43 patterns, 22 pattern candidates

Outline

Pattern Language for Large-Scale Agile Development [9]

Stakeholder

are defined as persons who have an interest in the project and/or are actively involved in the large-scale agile development.

Concern

describe challenges of stakeholders. They can be categorized as different topics such as risks or responsibilities and addressed by different Patterns, Anti-Patterns or Principles.

Principle

provide a common direction for action with the help of rules and guidelines to address specific concerns.

Coordination Pattern

define coordination mechanisms that are proven solutions for recurring coordination problems such as dependencies between activities or the management of tasks or resources.

Methodology Pattern

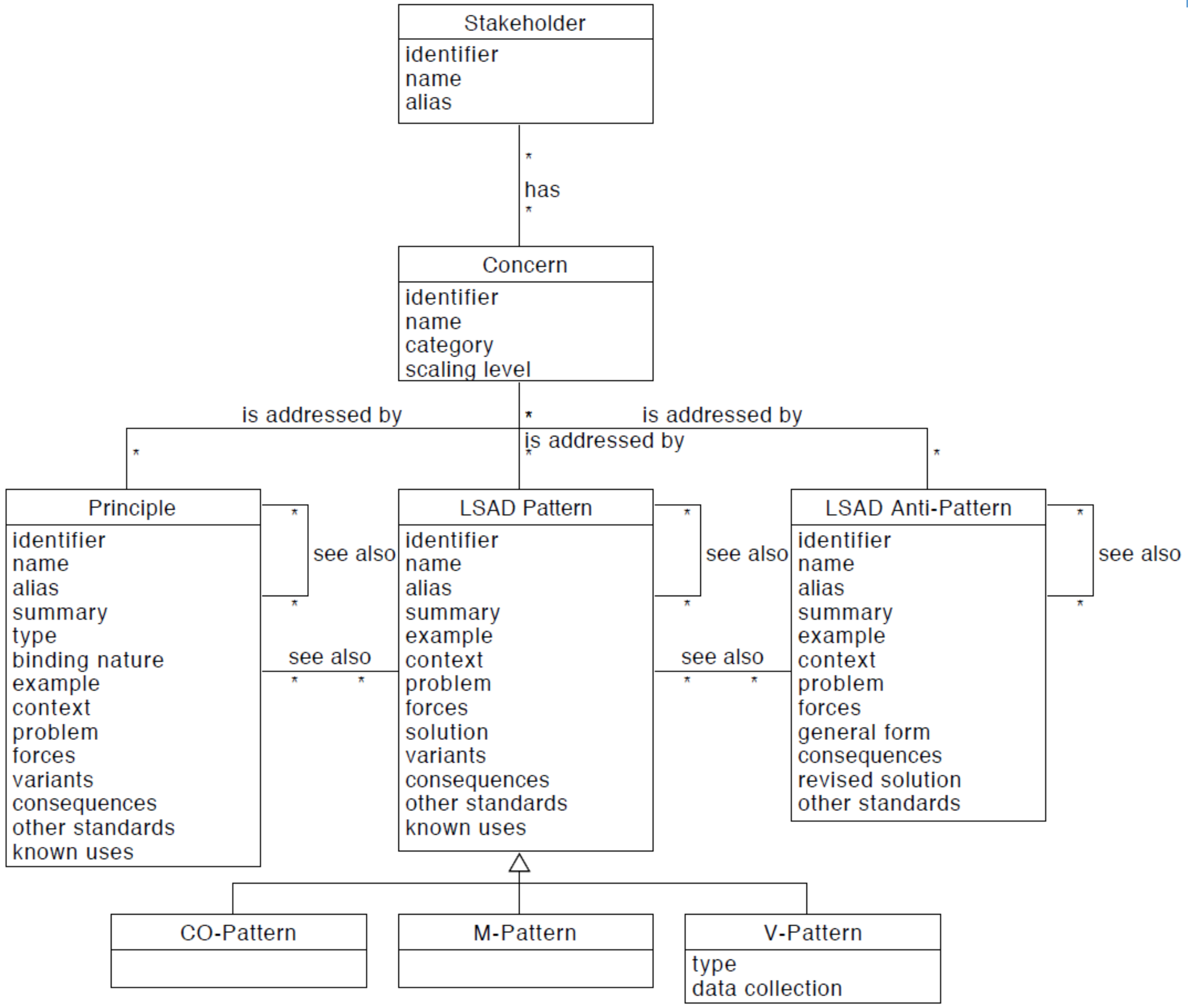
define concrete steps that are proven solutions to a problem.

Viewpoint Pattern

define proven solutions for visualizing information such as documents, boards, metrics, models, and reports.

Anti Pattern

define solutions that are unfavourable or harmful to the success of a software project. Anti-patterns represent the counterpart to patterns.



Outline

Recurring Concerns and Best Practices - Case Study & Expert Interviews

Case Study & Expert Interviews (Overview)			
Organization	No. Case Study Interviews	No. Expert Interviews	Roles
CarCo	20	3	Chief Technology Officer, Enterprise Architect, Group Lead IT, Product Owner, Requirements Engineer, Solution Architect, Scrum Master
ConsultCo	-	1	Solution Architect
GlobalInsureCo	12	-	Agile Developer, Chapter Lead, Agile Coaching, Enterprise Architect
ITCo	4	-	Enterprise Architect, Product Owner
PublicIncureCo	4	-	Agile Developer, Enterprise Architect, Head of IT Governance, Head of IT Governance Department
RetailCo	5	3	Chapter Lead Business Process Architecture, Chief Scrum Master, Enterprise Architect, Product Owner, Solution Architect, Scrum Master
SoftCo	-	3	Enterprise Architect, Solution Architect
TechCo	-	3	Enterprise Architect, Solution Architect
Sum	**45**	**13**	**15**

Expert Interviews			
ID	Role	Own Experience	Organization’s Experience
1	Enterprise Architect	3 - 6 years	1 – 3 years
2	Enterprise Architect	> 6 years	> 6 years
3	Enterprise Architect	> 6 years	3 - 6 years
4	Enterprise Architect	> 6 years	> 6 years
5	Enterprise Architect	> 6 years	1 - 3 years
6	Enterprise Architect	> 6 years	1 - 3 years
7	Solution Architect	> 6 years	1 - 3 years
8	Enterprise Architect	> 6 years	1 - 3 years
9	Solution Architect	> 6 years	> 6 years
10	Solution Architect	> 6 years	> 6 years
11	Solution Architect	> 6 years	3 - 6 years
12	Solution Architect	> 6 years	3 - 6 years
13	Enterprise Architect	> 6 years	3 - 6 years

Recurring Concerns and Best Practices - Recurring Concerns

🟩 Identified in Expert Interviews

🟦 Identified in Case Study

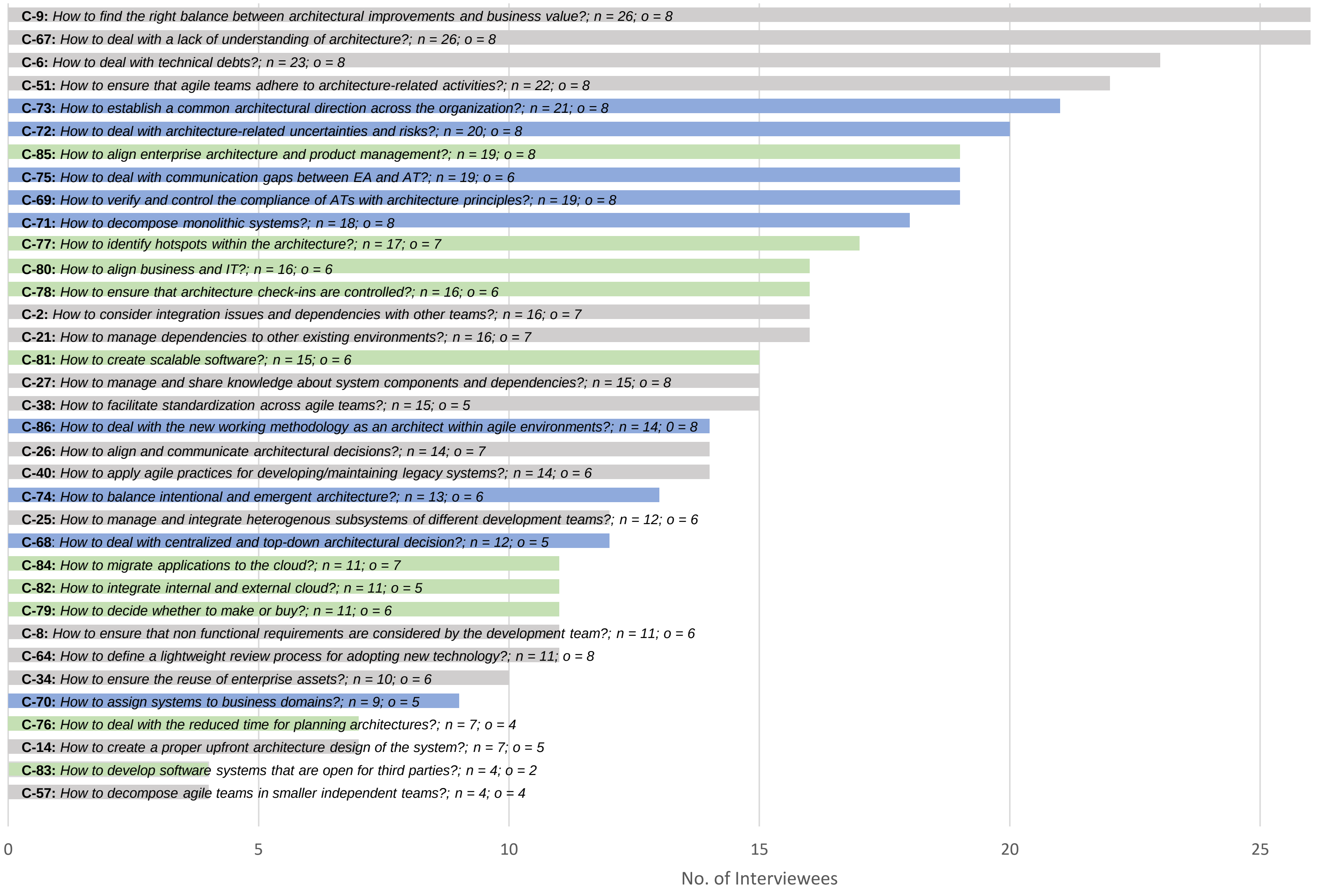
⬜ Identified in Literature

n = No. interviewees; o = No. organizations

🟩 10

🟦 9

⬜ 16



- ⬜ **C-9:** *How to find the right balance between architectural improvements and business value?; n = 26; o = 8*
- ⬜ **C-67:** *How to deal with a lack of understanding of architecture?; n = 26; o = 8*
- ⬜ **C-6:** *How to deal with technical debts?; n = 23; o = 8*
- ⬜ **C-51:** *How to ensure that agile teams adhere to architecture-related activities?; n = 22; o = 8*
- 🟦 **C-73:** *How to establish a common architectural direction across the organization?; n = 21; o = 8*
- 🟦 **C-72:** *How to deal with architecture-related uncertainties and risks?; n = 20; o = 8*
- 🟩 **C-85:** *How to align enterprise architecture and product management?; n = 19; o = 8*
- 🟦 **C-75:** *How to deal with communication gaps between EA and AT?; n = 19; o = 6*
- 🟦 **C-69:** *How to verify and control the compliance of ATs with architecture principles?; n = 19; o = 8*
- 🟦 **C-71:** *How to decompose monolithic systems?; n = 18; o = 8*
- 🟩 **C-77:** *How to identify hotspots within the architecture?; n = 17; o = 7*
- 🟩 **C-80:** *How to align business and IT?; n = 16; o = 6*
- 🟩 **C-78:** *How to ensure that architecture check-ins are controlled?; n = 16; o = 6*
- ⬜ **C-2:** *How to consider integration issues and dependencies with other teams?; n = 16; o = 7*
- ⬜ **C-21:** *How to manage dependencies to other existing environments?; n = 16; o = 7*
- 🟩 **C-81:** *How to create scalable software?; n = 15; o = 6*
- ⬜ **C-27:** *How to manage and share knowledge about system components and dependencies?; n = 15; o = 8*
- ⬜ **C-38:** *How to facilitate standardization across agile teams?; n = 15; o = 5*
- 🟦 **C-86:** *How to deal with the new working methodology as an architect within agile environments?; n = 14; 0 = 8*
- ⬜ **C-26:** *How to align and communicate architectural decisions?; n = 14; o = 7*
- ⬜ **C-40:** *How to apply agile practices for developing/maintaining legacy systems?; n = 14; o = 6*
- 🟦 **C-74:** *How to balance intentional and emergent architecture?; n = 13; o = 6*
- ⬜ **C-25:** *How to manage and integrate heterogenous subsystems of different development teams?; n = 12; o = 6*
- 🟦 **C-68**: *How to deal with centralized and top-down architectural decision?; n = 12; o = 5*
- 🟩 **C-84:** *How to migrate applications to the cloud?; n = 11; o = 7*
- 🟩 **C-82:** *How to integrate internal and external cloud?; n = 11; o = 5*
- 🟩 **C-79:** *How to decide whether to make or buy?; n = 11; o = 6*
- ⬜ **C-8:** *How to ensure that non functional requirements are considered by the development team?; n = 11; o = 6*
- ⬜ **C-64:** *How to define a lightweight review process for adopting new technology?; n = 11; o = 8*
- ⬜ **C-34:** *How to ensure the reuse of enterprise assets?; n = 10; o = 6*
- 🟦 **C-70:** *How to assign systems to business domains?; n = 9; o = 5*
- 🟩 **C-76:** *How to deal with the reduced time for planning architectures?; n = 7; o = 4*
- ⬜ **C-14:** *How to create a proper upfront architecture design of the system?; n = 7; o = 5*
- 🟩 **C-83:** *How to develop software systems that are open for third parties?; n = 4; o = 2*
- ⬜ **C-57:** *How to decompose agile teams in smaller independent teams?; n = 4; o = 4*

Recurring Concerns and Best Practices - Patterns and Pattern Candidates

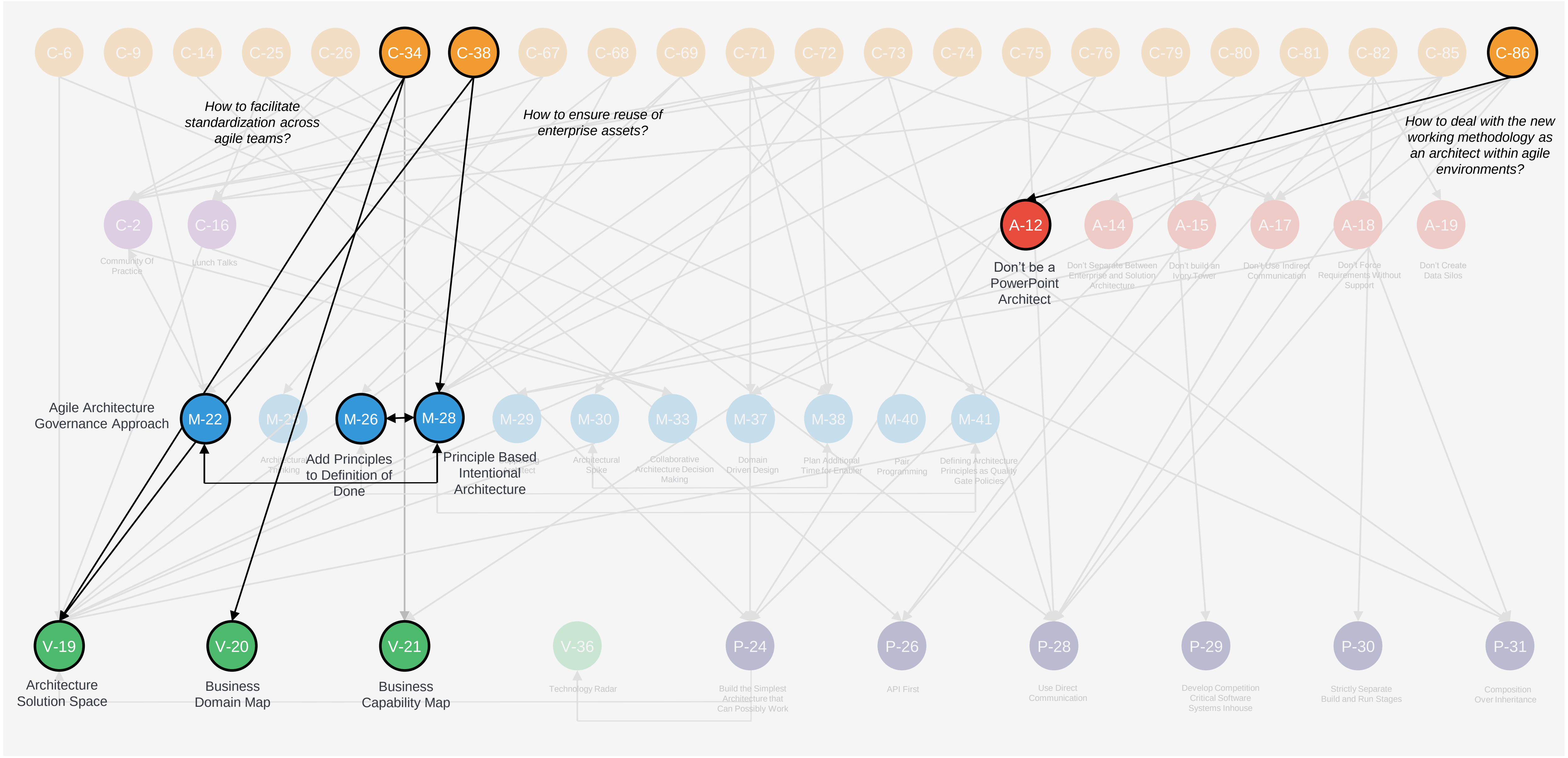
🟧	Anti Pattern
🟪	Coordination Pattern
🟦	Methodology Pattern
🟩	Viewpoint Pattern
⬜	Principle
□	Pattern

ID
Name
Occurrence

P - 18 Collocate Architects with Agile Teams **	P - 25 Use Microservices **	CO - 2 Community of Practice ******	M - 27 Quality Gate **	M - 35 Empowered Community of Practice ***	V - 21 Business Capability Map ***	V - 29 Number of Consulting Requests **	A - 11 Don't Use Best of Breed **	A - 19 Don't Create Data Silos ****
P - 19 IT Systems Communicate through Services ***	P - 26 API First *****	CO - 15 Center of Excellence ***	M - 28 Principle based Intentional Architecture *******	M - 36 Plan Additional Time for Enablers ***	V - 22 Technical Debt Backlog **	V - 30 Number of Dependencies ***	A -12 Don't be a PowerPoint Architect *****	
P - 20 End to End Responsibility **	P - 27 Cloud First *****	CO - 16 Lunch Talk ***	M - 29 Supporting Architect ******	M - 37 Domain Driven Design ***	V - 23 Communication Diagram ***	V - 31 Number of Releases **	A - 13 Don't Use Big Design Up Front ***	
P - 21 Loose Coupling of Systems ***	P - 28 Use Direct Communication *****	M - 22 Agile Architecture Governance Approach ******	M - 30 Architectural Spike ***	M - 38 Business Capability Centric Teams **	V - 24 Context Map *	V - 32 Number of Version Skippings ***	A - 14 Don't Separate between Enterprise and Software Architecture ***	
P - 21 Reuse is Preferable to Buy, which is Preferable to Make **	P - 29 Develop Competition Critical Software Systems Inhouse ***	M - 23 Enterprise Architecture Governance Service **	M - 31 Agile Collaboration Environment **	M - 39 Pair Programming *****	V - 25 Data Diagram **	V - 33 System Diagram ***	A - 15 Don't Build an Ivory Tower *****	
P - 22 Reuse Redundancy **	P - 30 Strictly Separate Build and Run Stages ****	M - 24 Architecture Governance through Institutional Pressure **	M - 32 Architecture Gate **	M - 40 Defining Architecture Principles as Quality Gate Policies *****	V - 26 Weighted Shortest Job First ***	V - 34 Time to Feature Delivery **	A - 16 Don't Overshoot Coordination Meetings **	
P - 23 Applications rely on One Technology Stack ***	P - 31 Composition Over Inheritance ****	M - 25 Architectural Thinking *****	M - 33 Collaborative Architecture Decision Making ******	V - 19 Architecture Solution Space ******	V - 27 Number of API Calls **	V - 35 Total Cost of Ownership **	A - 17 Don't Use Indirect Communication ****	
P - 24 Build the Simplest Architecture that Can Possibly Work ****	P - 32 Systems Communicate through Services Only ****	M - 26 Add Principles to DoD ****	M - 34 Architectural Runway **	V - 20 Business Domain Map ***	V - 28 Number of Changes of Architecture Models ****	V - 36 Technology Radar ****	A - 18 Don't Force Requirements Without Support ******	

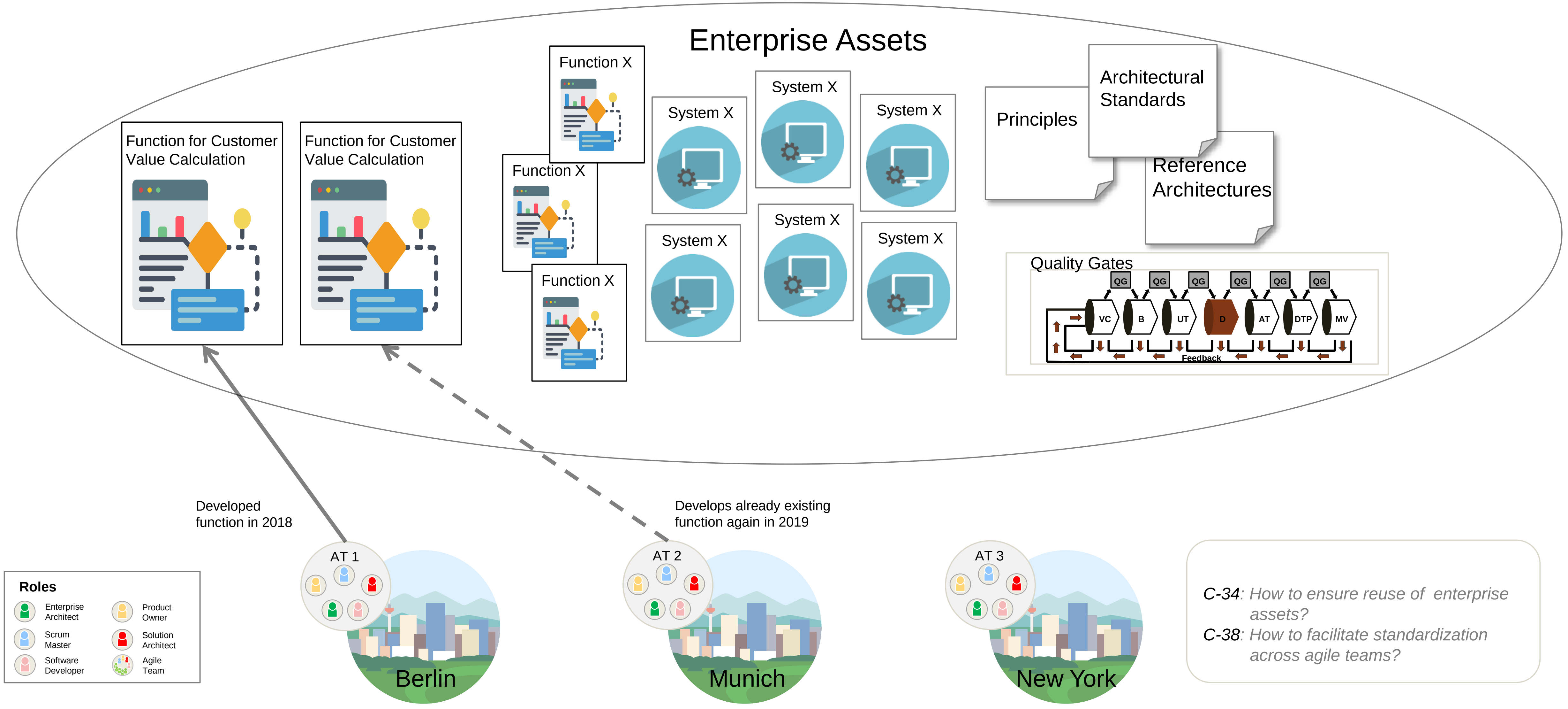
No. of Patterns	No. of P. Candidates
11	5
3	0
12	7
10	8
7	2

Recurring Concerns and Best Practices - Relationship


C-6
C-9
C-14
C-25
C-26
C-34
C-38
C-67
C-68
C-69
C-71
C-72
C-73
C-74
C-75
C-76
C-79
C-80
C-81
C-82
C-85
C-86
How to facilitate standardization across agile teams?
How to ensure reuse of enterprise assets?
How to deal with the new working methodology as an architect within agile environments?
C-2
Community Of Practice
C-16
Lunch Talks
A-12
Don't be a PowerPoint Architect
A-14
Don't Separate Between Enterprise and Solution Architecture
A-15
Don't build an Ivory Tower
A-17
Don't Use Indirect Communication
A-18
Don't Force Requirements Without Support
A-19
Don't Create Data Silos
Agile Architecture Governance Approach
M-22
M-25
Architectural Thinking
M-26
Add Principles to Definition of Done
M-28
Principle Based Intentional Architecture
M-29
Supporting Architect
M-30
Architectural Spike
M-33
Collaborative Architecture Decision Making
M-37
Domain Driven Design
M-38
Plan Additional Time for Enabler
M-40
Pair Programming
M-41
Defining Architecture Principles as Quality Gate Policies
V-19
Architecture Solution Space
V-20
Business Domain Map
V-21
Business Capability Map
V-36
Technology Radar
P-24
Build the Simplest Architecture that Can Possibly Work
P-26
API First
P-28
Use Direct Communication
P-29
Develop Competition Critical Software Systems Inhouse
P-30
Strictly Separate Build and Run Stages
P-31
Composition Over Inheritance

Outline

Exemplary V-Pattern: Architecture Solution Space

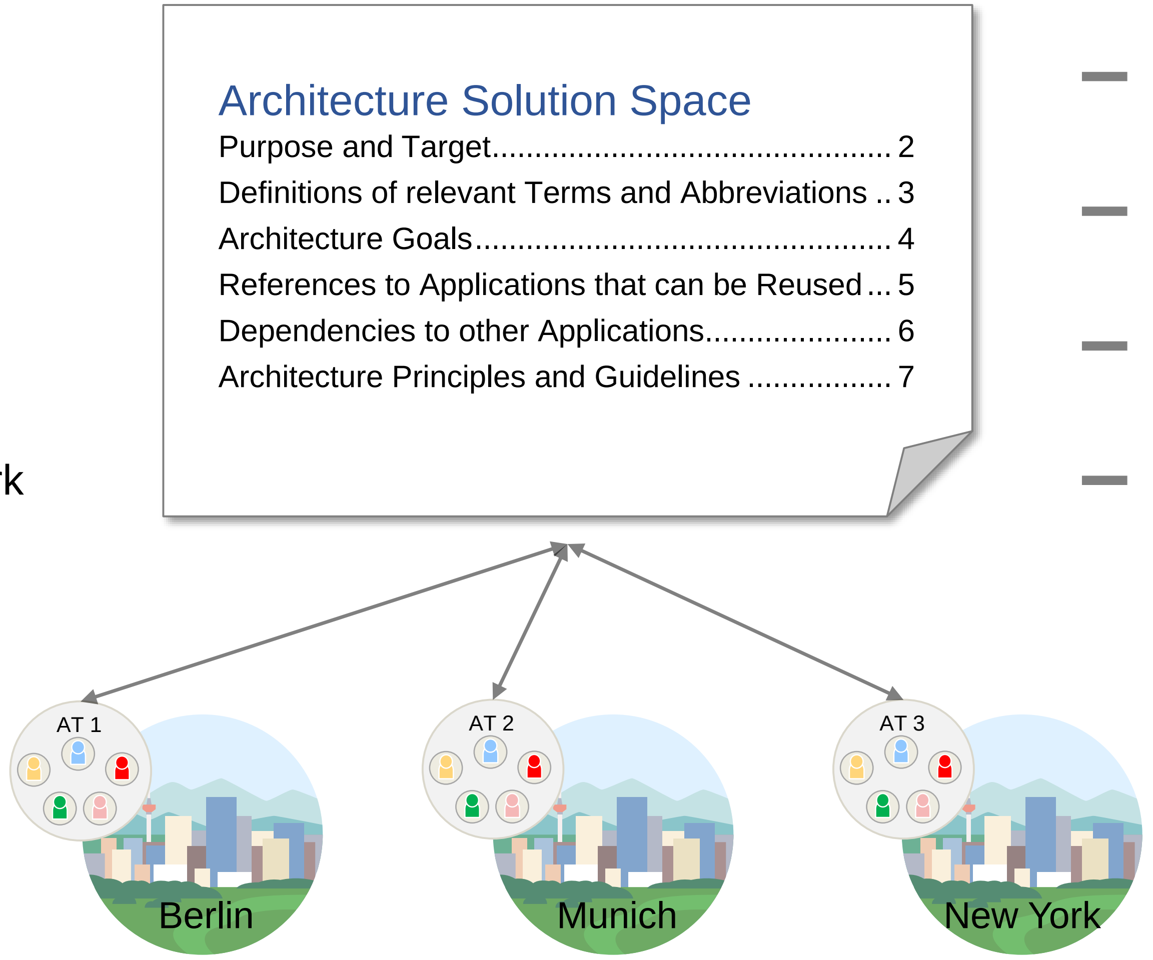


C-34: How to ensure reuse of enterprise assets?

C-38: How to facilitate standardization across agile teams?

Exemplary V-Pattern: Architecture Solution Space

+ Provides guidance
+ Helps to identify risks and dependencies in advance
+ Does not restrict freedom of developers
+ Provides a clean framework



Architecture Solution Space
Purpose and Target.. 2
Definitions of relevant Terms and Abbreviations .. 3
Architecture Goals.. 4
References to Applications that can be Reused ... 5
Dependencies to other Applications...................... 6
Architecture Principles and Guidelines 7
AT 1
Berlin
AT 2
Munich
AT 3
New York

– Only effective if created collaboratively
– Less effective if not maintained properly
– Control of adherence is difficult
– No consequences in case of non-compliance

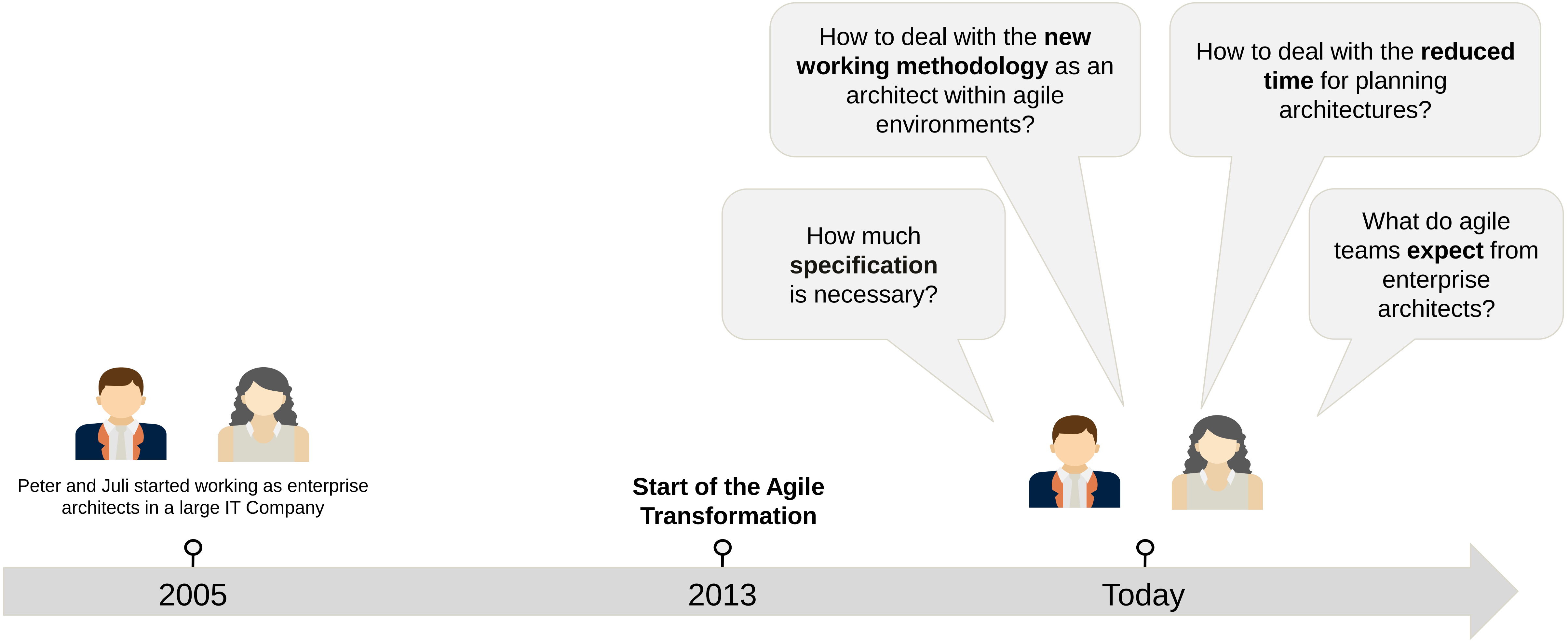
Context

Agile transformation forces organizations to create and design architectures differently
*→ **Maximal autonomy vs. EAM specifications***

Problem

C-34: How to ensure reuse of enterprise assets?
C-38: How to facilitate standardization across agile teams?

Exemplary Anti-Pattern: Don‘t be a PowerPoint Architect



How to deal with the **new working methodology** as an architect within agile environments?

How to deal with the **reduced time** for planning architectures?

How much **specification** is necessary?

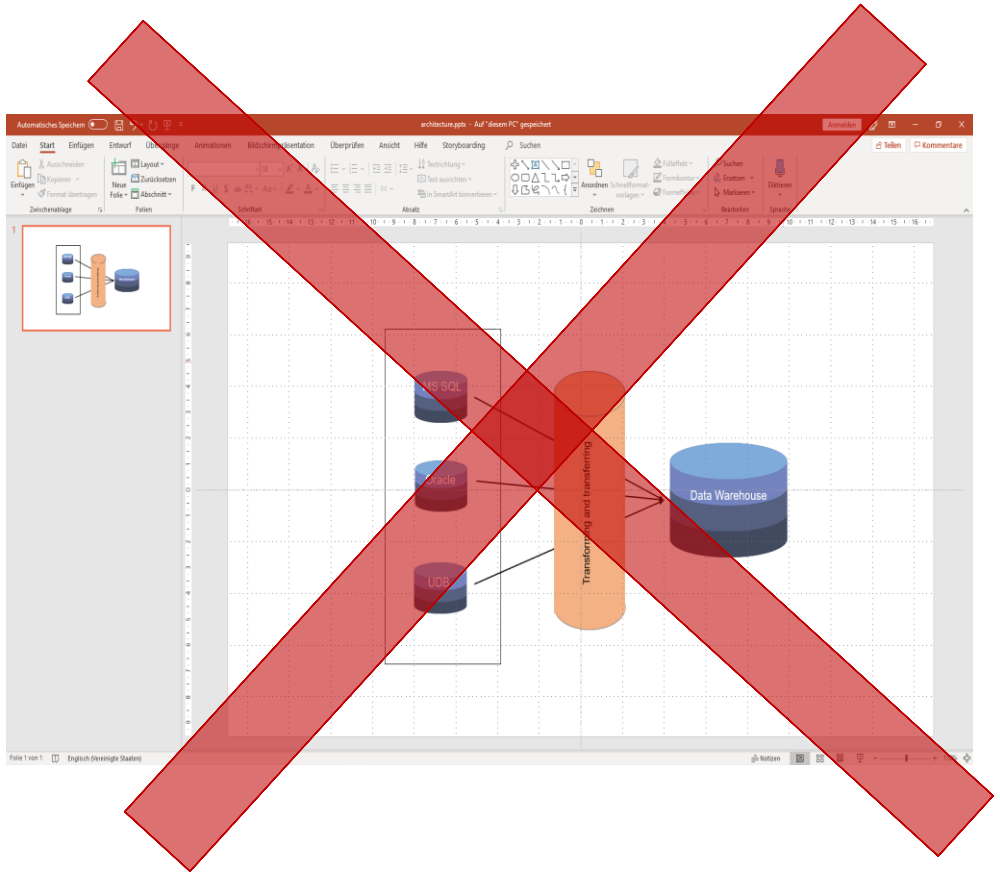
What do agile teams **expect** from enterprise architects?

Peter and Juli started working as enterprise architects in a large IT Company

Start of the Agile Transformation

Exemplary Anti-Pattern: Don‘t be a PowerPoint Architect

- + Provides technical guidance
- + Increases acceptance of architects by agile teams
- + Increases understanding of architecture
- + Increases intrinsic motivation to adhere to architecture principles



- – Requires a broad and deep skill set which is rare
- – Architects need to have enough capacity
- – Enabling takes a lot of time

Context

Working methodology has changed in an agile environment
*→ **More technical support is required***

Problem

C-86: How to deal with the new working methodology as an architect within agile environments?

Outline

Conclusion

Key Findings

- Agile transformation changes working methodology of architects leading do Anti-Patterns
- Patterns and Principles provide a way to balance intentional and emergent architecture
- Role of the supporting architect is increasingly important and requires deep technical know-how
- Feedback mechanisms and automated testing needs to be implemented for compliance

Future Work

- Identification of new patterns by conducting similar projects at other organizations
- Validation of identified patterns and pattern candidates in other organizations
- Long-time studies on the progress of concerns within agile transformations

References

[1] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. MIS Quarterly, 28(1):75–105, March 2004.

[2] Bogner, A., Littig, B., & Menz, W. (2009). Introduction: Expert interviews—An introduction to a new methodological debate. In Interviewing experts (pp. 1-13). Palgrave Macmillan, London.

[3] Buckl S., Matthes F., Schneider A.W., & Schweda C.M. (2013) Pattern-Based Design Research – An Iterative Research Method Balancing Rigor and Relevance.

[4] Dingsøyr , T., Nerur , S., Balijepally , V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software , 85 (6), 1213 1221.

[5] D. Greefhorst, E. Proper (2011): Architecture Principles: The Cornerstones of Enterprise Architecture. Springer-Verlag Berlin Heidelberg.

[6] Hanschke, S., Ernsting, J., & Kuchen, H. (2015, January). Integrating agile software development and enterprise architecture management. In System Sciences (HICSS), 2015 48th Hawaii International Conference on (pp. 4099-4108). IEEE.

[7] Hauder, M., Roth, S., Schulz, C., & Matthes, F. (2014). Agile enterprise architecture management: an analysis on the application of agile principles. In International Symposium on Business Modeling and Software Design BMSD.

[8] Uludağ , Ö., Kleehaus , M., Caprano , C., & Matthes , F. (2018). Identifying and Structuring Challenges in Large Scale Agile Development Based on a Structured Literature Review. 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC) EDOC), 191 197.

[9] Uludağ, Ö., Harders, N.-M., & Matthes, F. (2019). Documenting Recurring Concerns and Patterns in Large-Scale Agile Development.

[10] Runeson and Höst (2009): Guidelines for conducting and reporting case study research in software engineering;

[11] Tashakkori, A., & Teddlie, C. (Eds.). (2010). Sage handbook of mixed methods in social & behavioral research. sage.

[12] Vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., & Cleven, A. (2009, June). Reconstructing the giant: on the importance of rigour in documenting the literature search process. In Ecis (Vol. 9, pp. 2206-2217).

[13] Yin (2008): Case Study Research: Design and Methods (Applied Social Research Methods);

Thank you for your attention!