



Model-Based System Engineering in Practice: Document Generation - MegaM@Rt Project Experience

Presented by **Andrey Sadovykh**, Innopolis / Softeam



Andrey's Background



- Experience

- Project Manager, Coordinator – 14 years

- Worked for

- SWsoft, AIRBUS, SOFTEAM

- Worked with

- European Space Agency
- Thalès, Scania, Volvo, Nokia, SAP, Bombardier, ATOS, IBM, SIEMENS, EDF

MSc MIPT

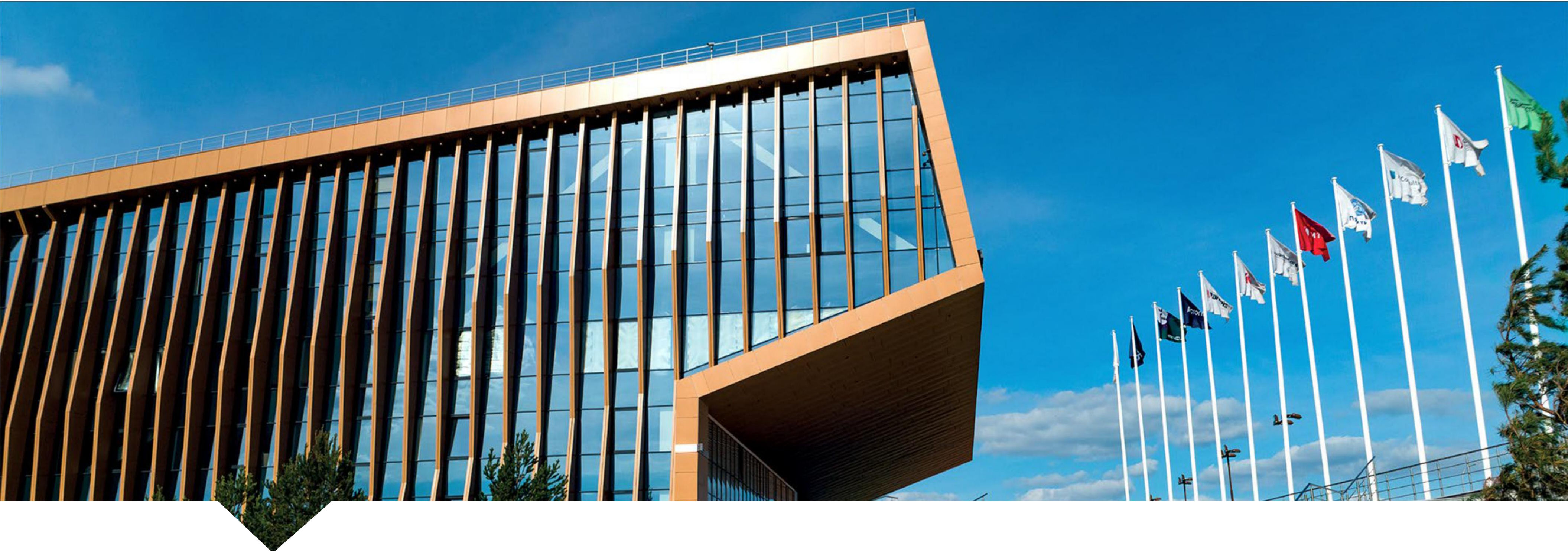
PhD AIRBUS / Sorbonne Uni.

MBA HEC Paris

- Areas of Interest

- Distributed Systems
- Model-driven Engineering applied ...
 - Software and services
 - Cyber-physical systems
- Digital Innovation

- Why documentation?
- MegaM@Rt2 project
- Documentation for Requirements, Architecture, Roadmap and Traceability
- Modelio Demo
- Conclusion and discussion
- Backup
 - Requirement Management Approach
 - Architecture Management Approach



Introduction

How much (%) of project time do you spend on documenting?

Why documentation?

What are the main difficulties in documenting your project?

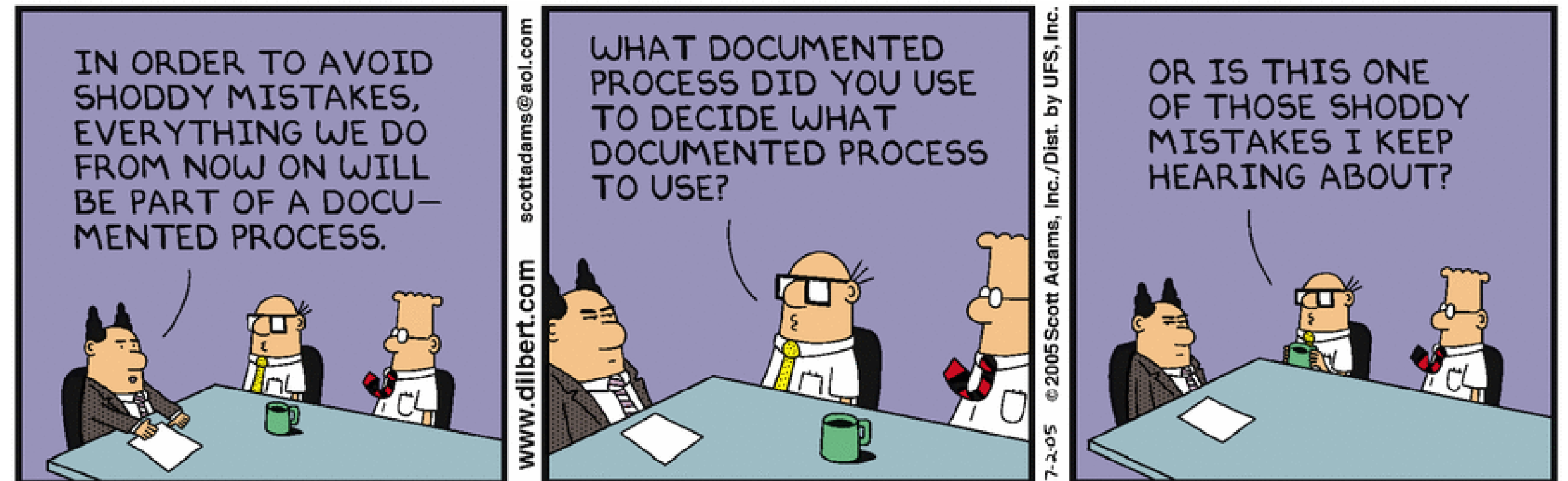
Why documentation?

Purpose

- Communication
 - Requirements
 - Architecture
 - Design
 - Test plans
 - Reviews and reports
- Contractual, legal means
- Required by process to instill quality

Properties

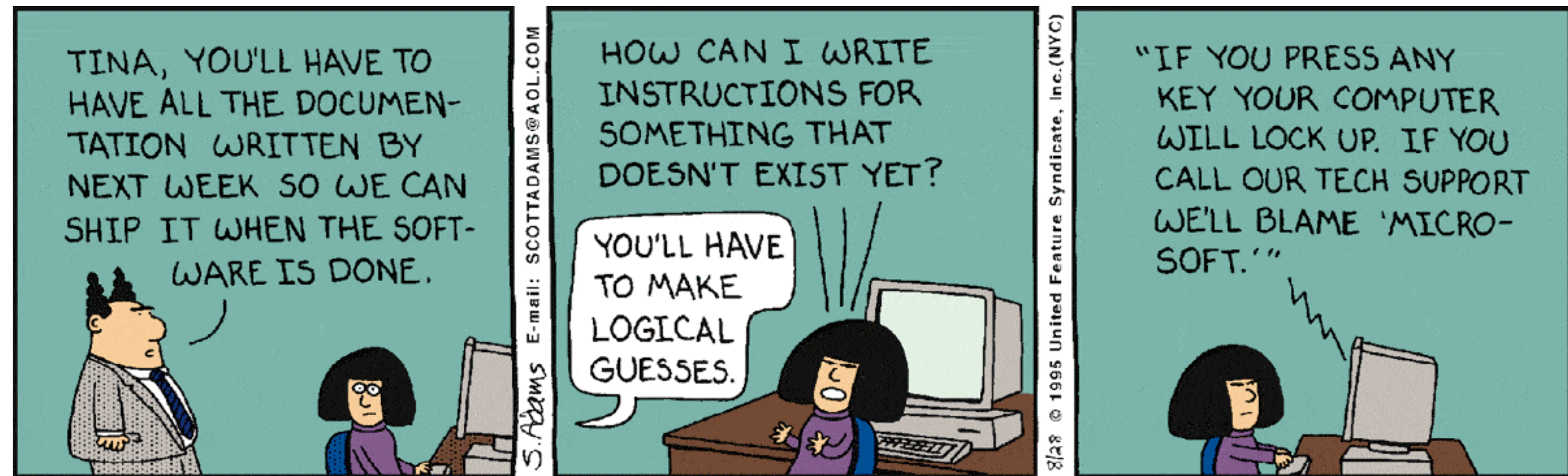
- Readers / Stakeholders
- Reading style – story or dictionary
- Traces and references, diagrams
- Formats



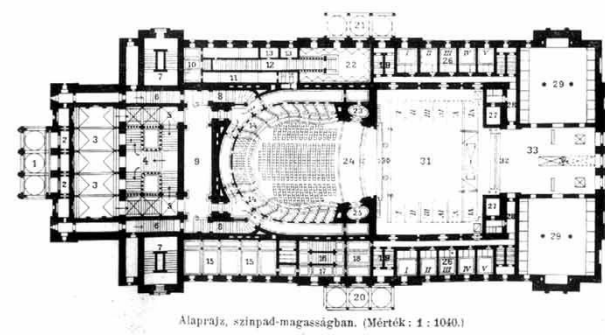
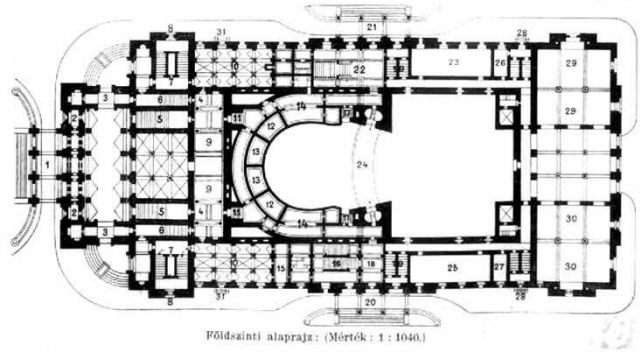
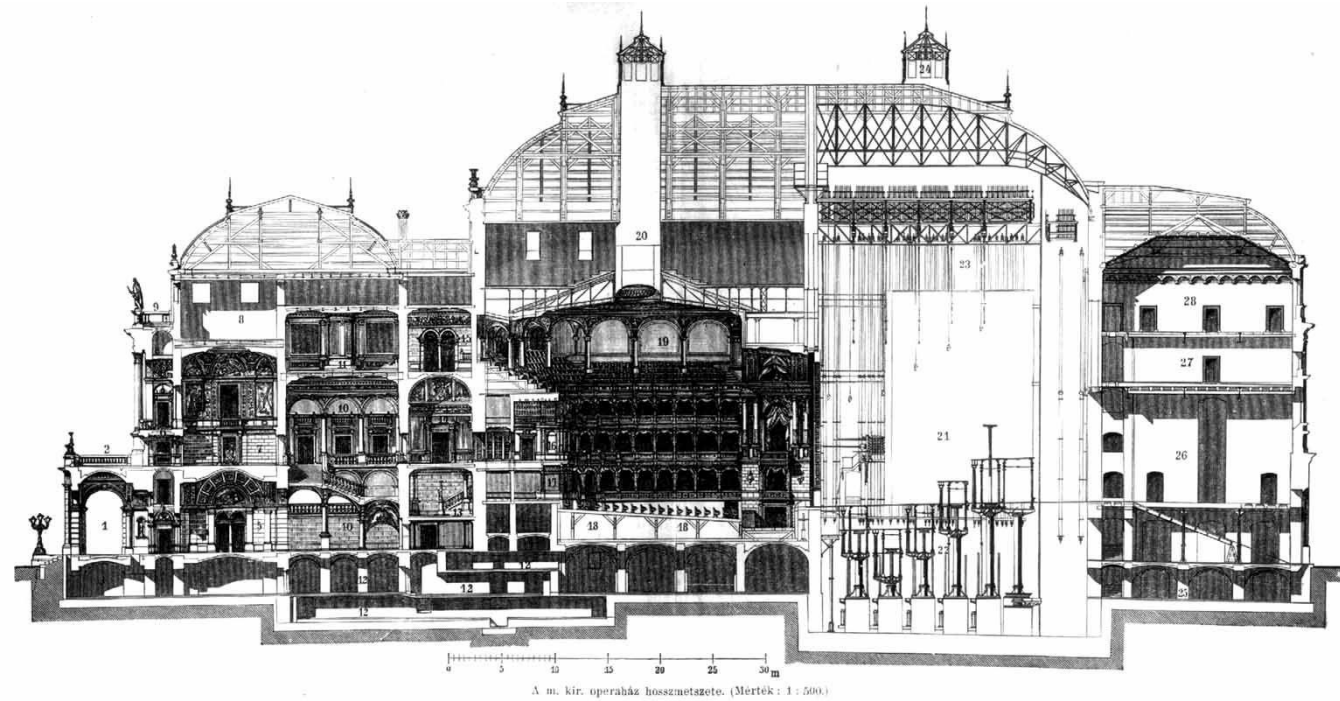
Problems with the documenting

Takes productive time/cost!

- Consistency and synchronization
- Outdating
- Maintaining references
- Formatting and presenting
- Reusing



Model as a blueprint to produce things



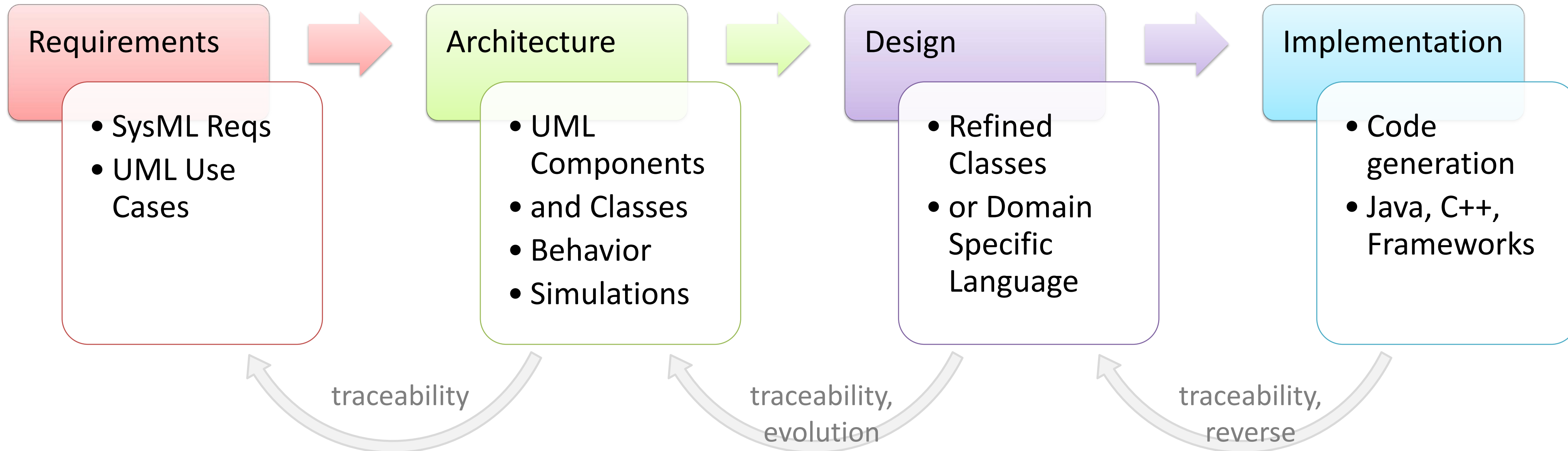
- Models
 - Descriptive
 - Prescriptive
- Fowler distinguish models as
 - Sketches – for communication
 - Blueprints – for development
 - Programs – for execution
- Why modelling in software engineering?
 - Increase in productivity
 - Less errors
 - Cut coding
- Drivers
 - More complex software to be developed including
 - Increasing need to evolve the software
 - Lack of software engineering skills
 - Need to understand the domain problem to support with soft

We can produce documents from models too

Mapping,
Model transformations
and refinements

Mapping,
Model transformations
and refinements

Code generation,
Automated resource allocation
Automated deployment

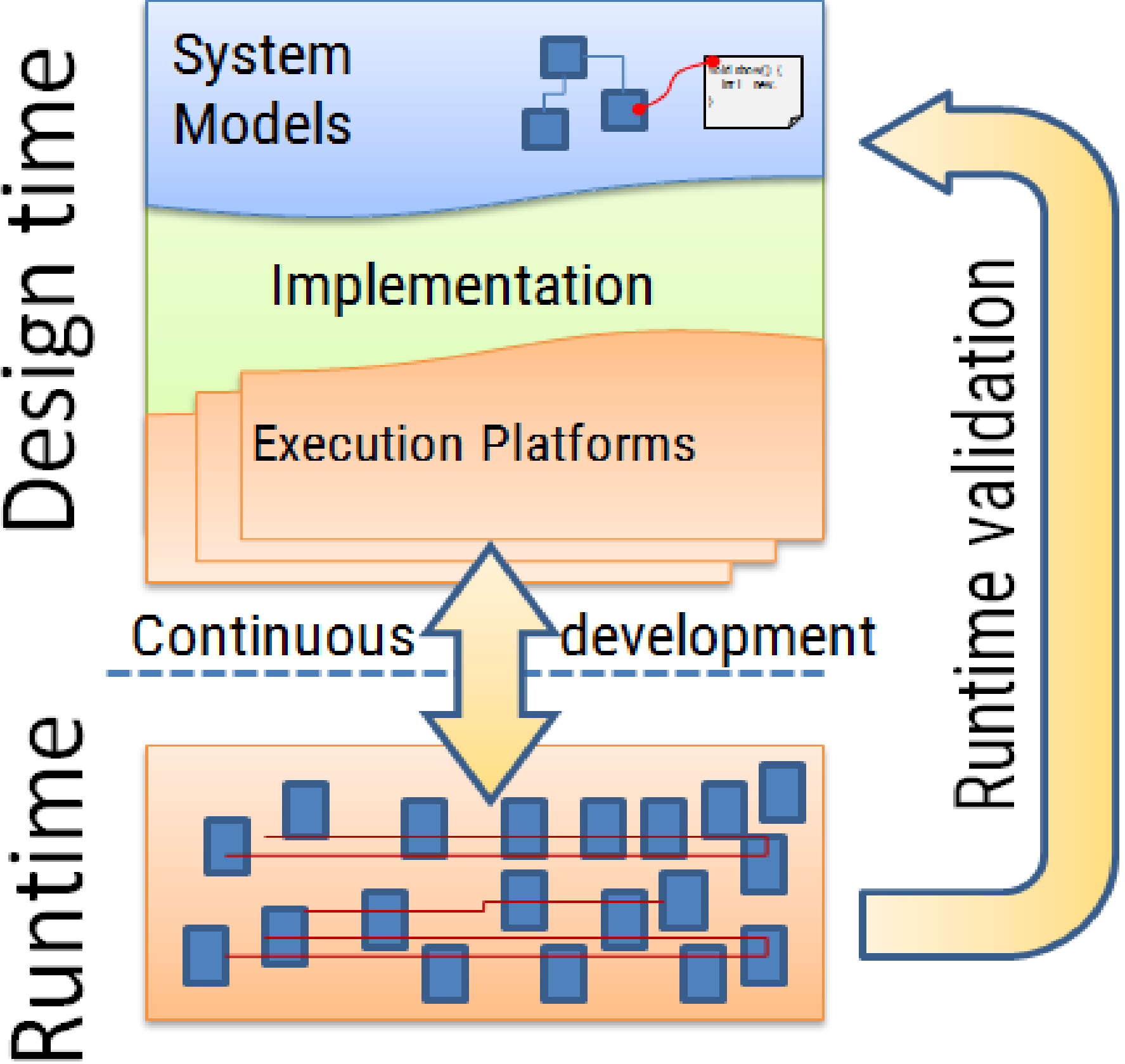




MegaM@Rt2 project and its challenges



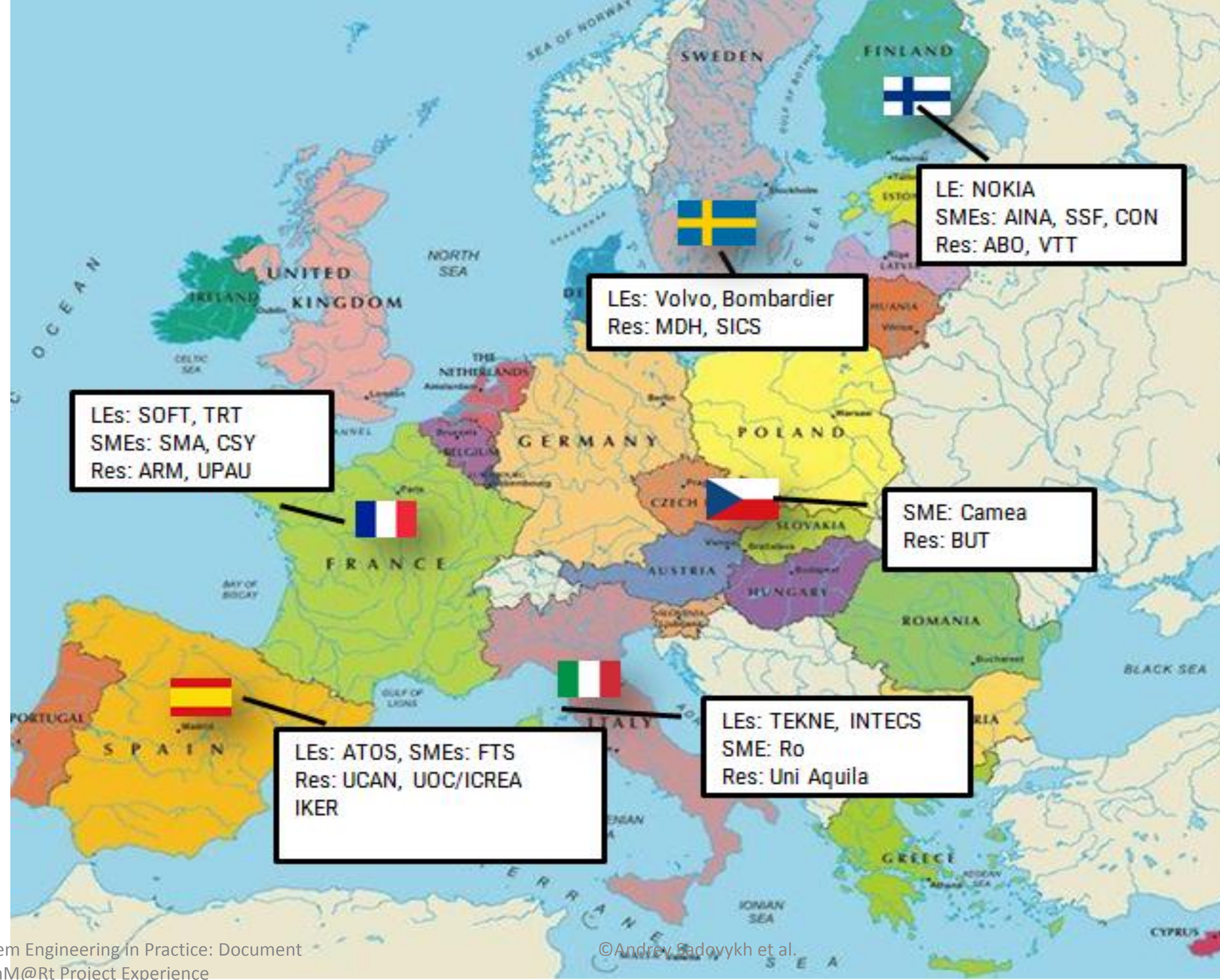
Scope - MegaM@Rt2 tool box:



- Design time:
 - Holistic system engineering
 - Team collaboration over distributed models
 - Global traceability
- Runtime:
 - Tracing / Monitoring
 - Models@Runtime



Size

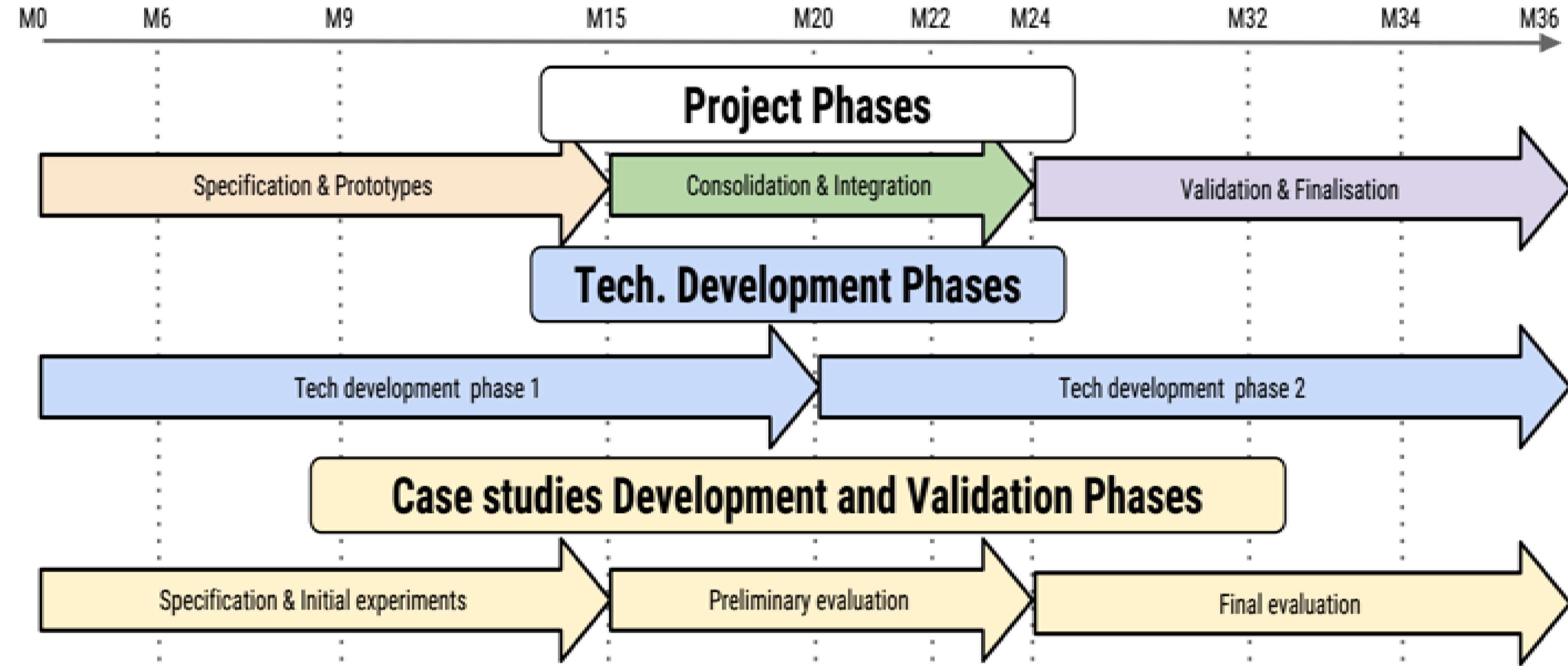


Collaboration, Complementarities and Differences

		Case study providers								Technology providers (Tools and Methods)																																							
		TRT	CSY	IKER	TEK	NOK	VCE	BT	CAM	AINA	SOFT	SMA	ARM	UPAU	ATOS	UCAN	UOC	FTS	UAQ	INT	RO	ABO	SSF	VTT	CON	MDH	SICS	BUT																					
Application Domain	Transportation	X	X				X	X	X		Modelio suite	Smartesting Tools (CertifyIt / MBeetle)	AM3, EMFViews, Neo4EMF, ATL	Model execution (PauWare, SCXML)	EMF, ATL, Acceleo, OCL, UML2, Profiles		EMFtoCSP, Collaboro			EMF, MOSES, ATL, Acceleo, MARTE, UML2	CHESS (Modeling) & MOSES (Performance Analysis)	Completeness/Consistency Requirements check		LIME toolset		rmig De																							
	Smart warehouse			X																																													
	Telecom				X	X				X																																							
	Industrial Control			X						X																																							
Innovation topics	Domain specific languages	X	X			X			X		X	X	X	X	X	X	X					X							X																				
	Requirements modeling	X			X	X	X	X			X					X		X		X	X					X																							
	Aspect oriented modeling	X									X				X							X							X																				
	MB Verification	X	X			X	X	X		X					X	X	X	X		X	X	X				X	X	X																					
	MB Performance Analysis				X					X						X			X	X																													
	Simulation	X		X	X	X		X						X		X											X																						
	MB Validation	X	X			X	X	X		X		X			X		X	X	X				X			X	X	X																					
	(MB) Runtime verification	X		X	X	X	X	X	X	X	X			X									X	X	X		X	X	X																				
	MB testing (online &offline)			X	X	X	X	X	X	X	X		X			X							X	X		X	X	X	X																				
	Requirements/system traceability			X		X	X	X			X		X			X		X		X			X			X	X																						
	MB collaboration and governance						X	X			X		X				X	X																															
	Continuous Development			X	X	X				X	X	X	X	X	X	X		X					X	X	X					X																			
	Anti-pattern detection	X													X		X		X																														
	Root-cause analysis					X	X	X		X														X	X		X	X																					
Model Management & Storage					X					X		X						X	X																														

Process

Timeline:

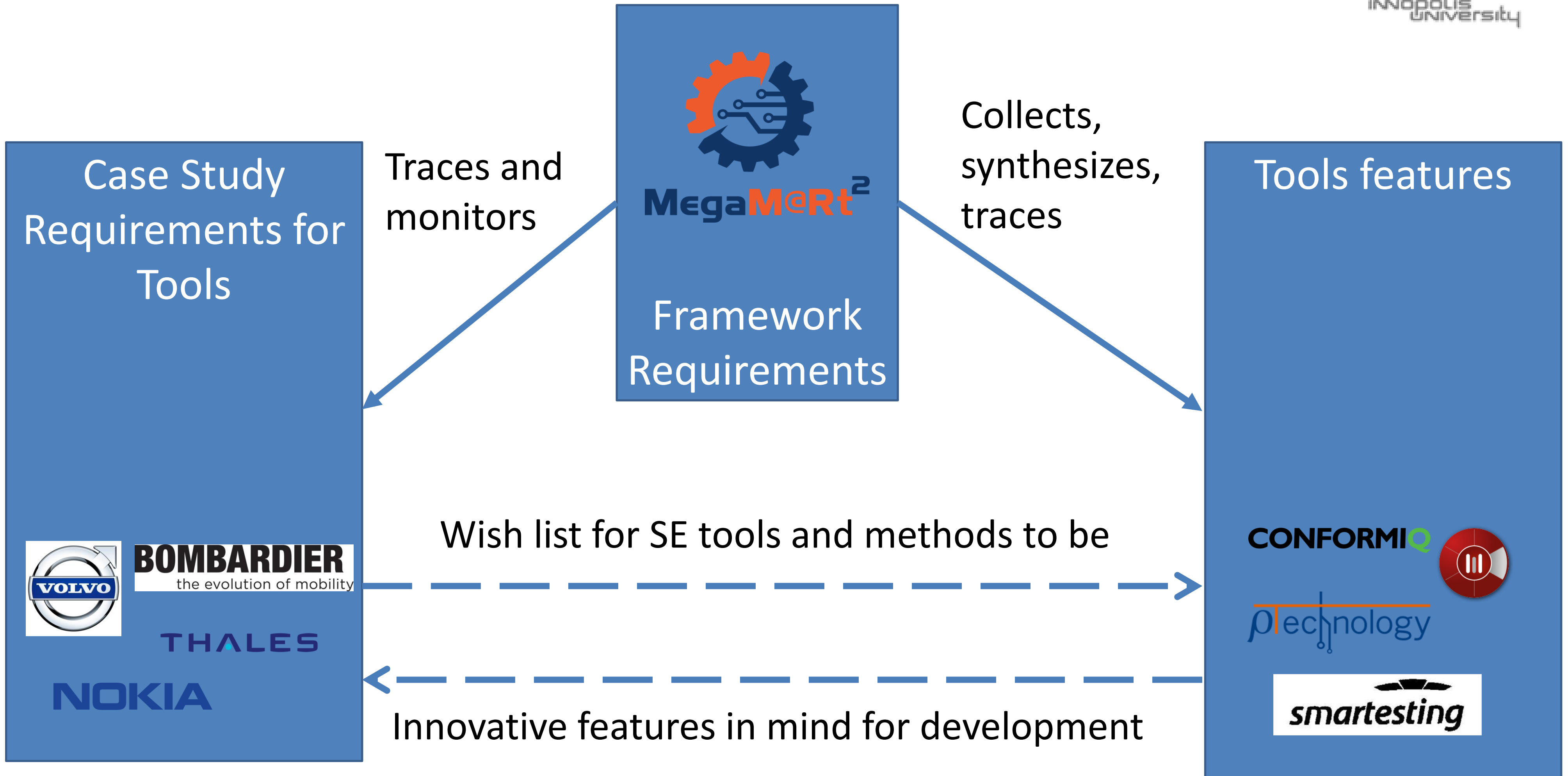




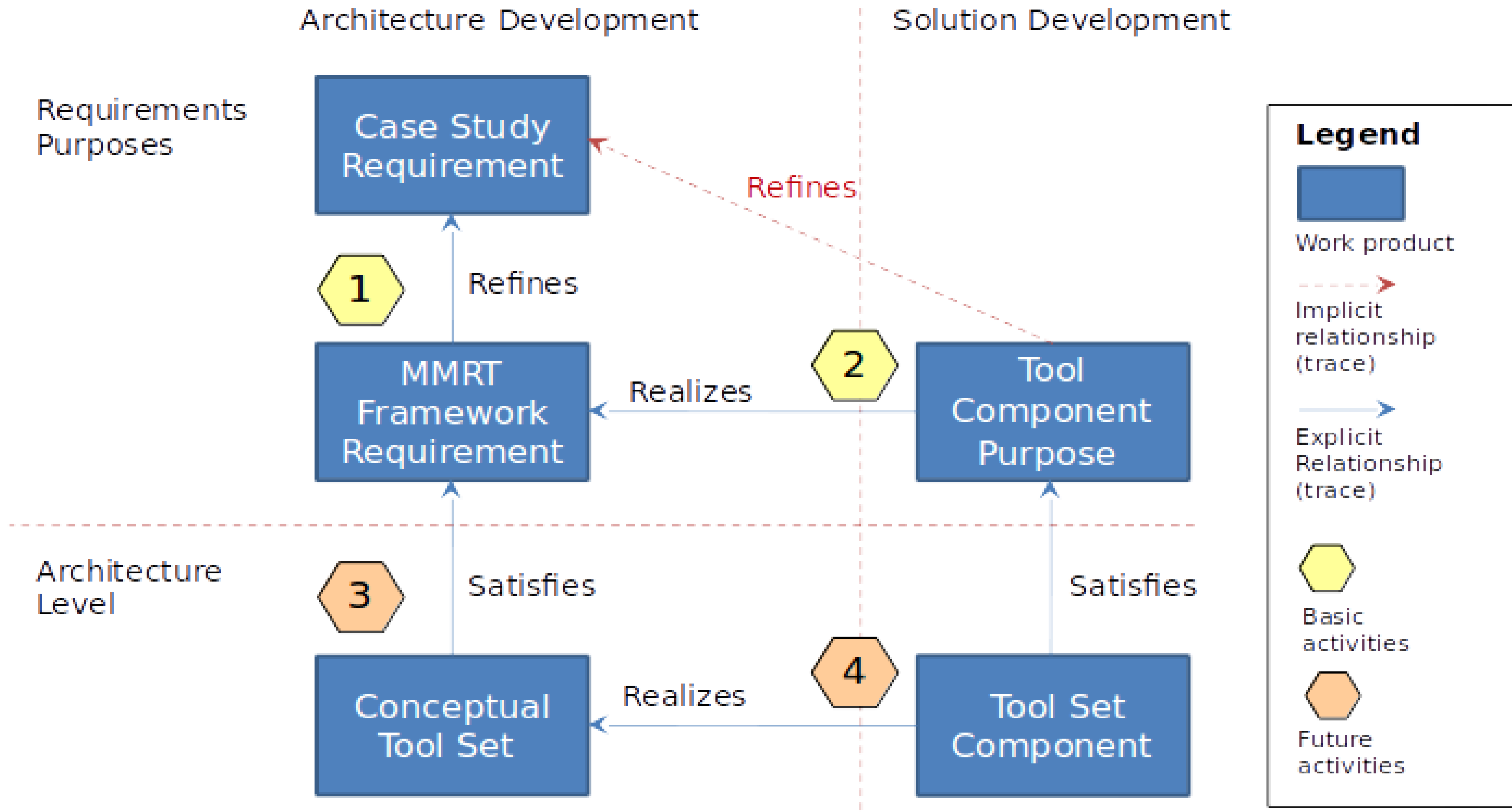
Automating Documentation in MegaM@Rt

Requirement, Architecture, Traceability

MegaM@Rt2 Approach

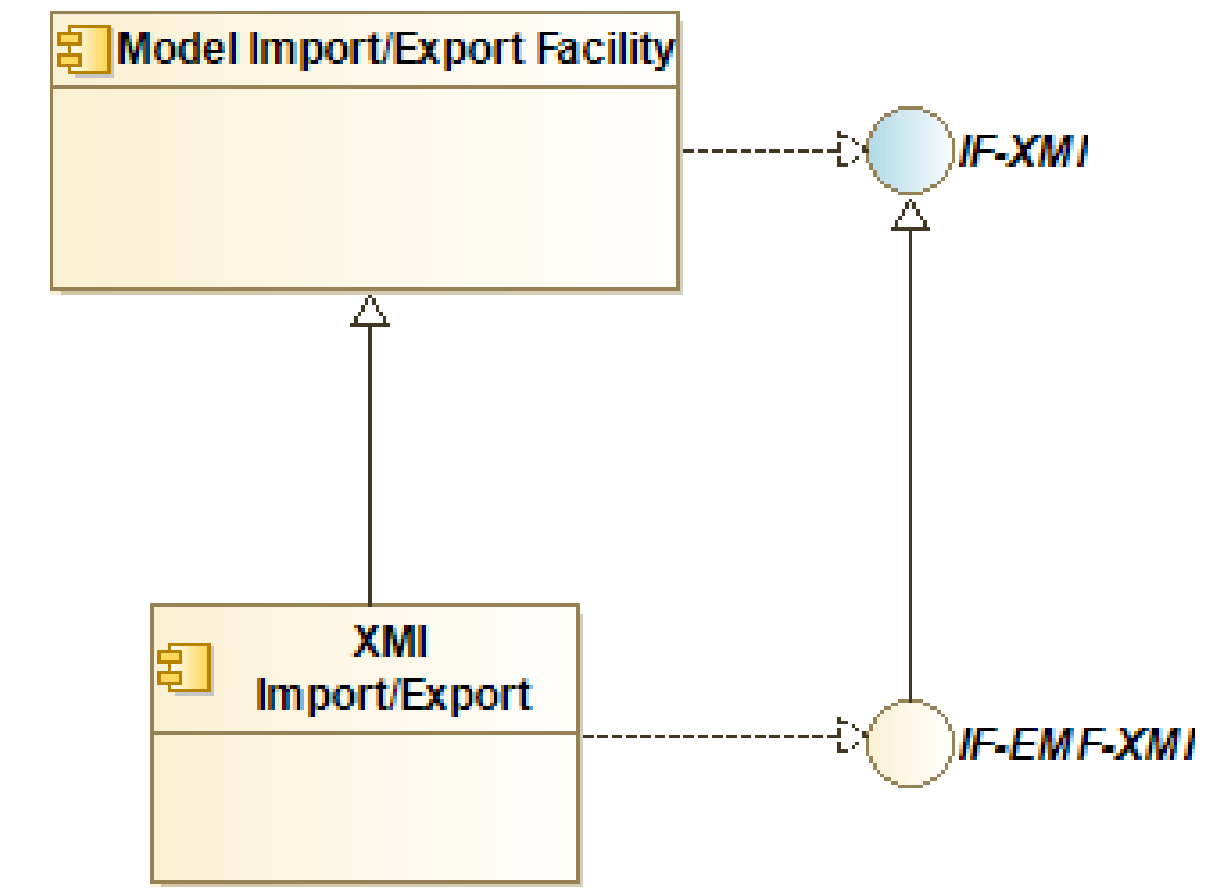
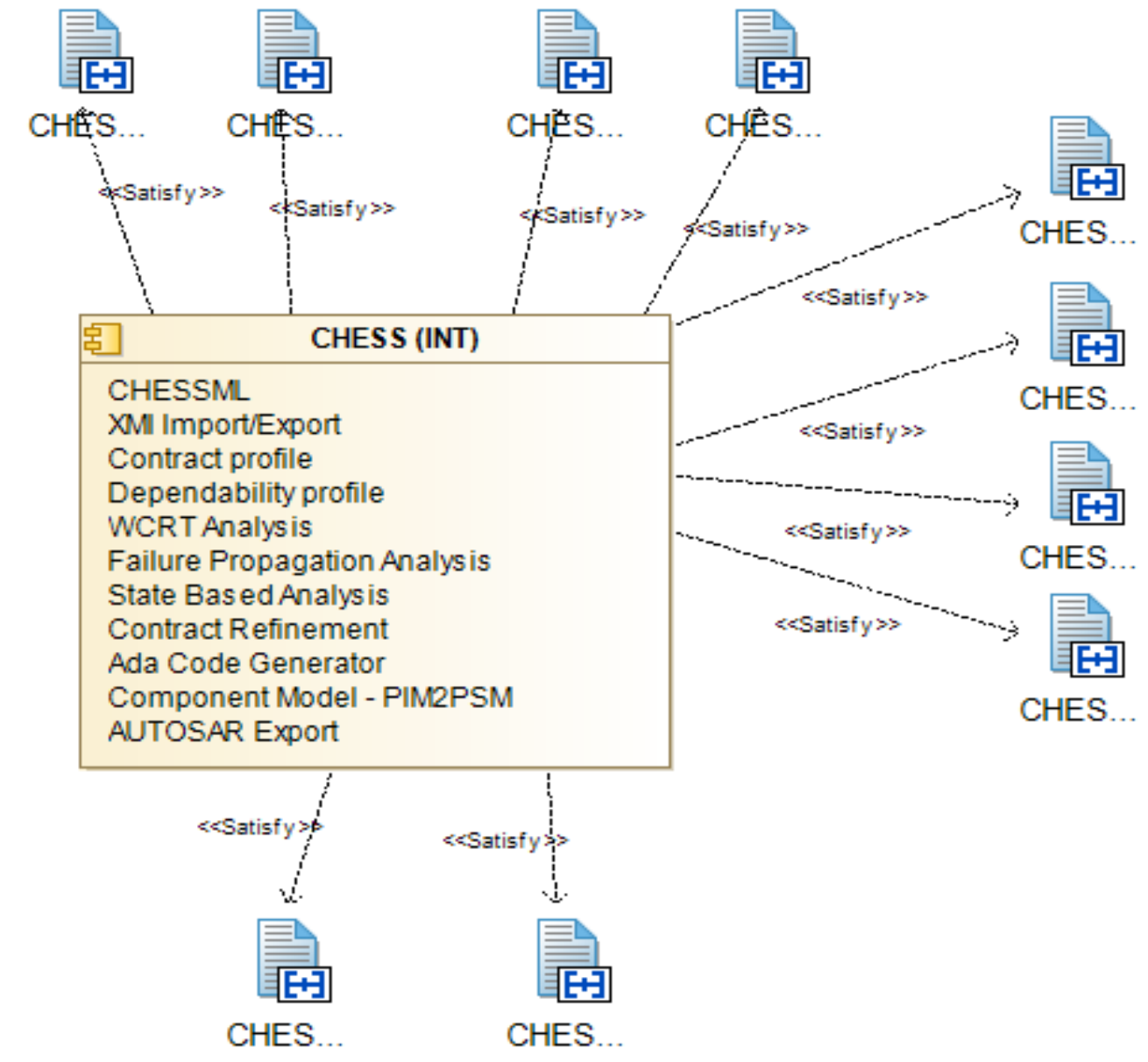
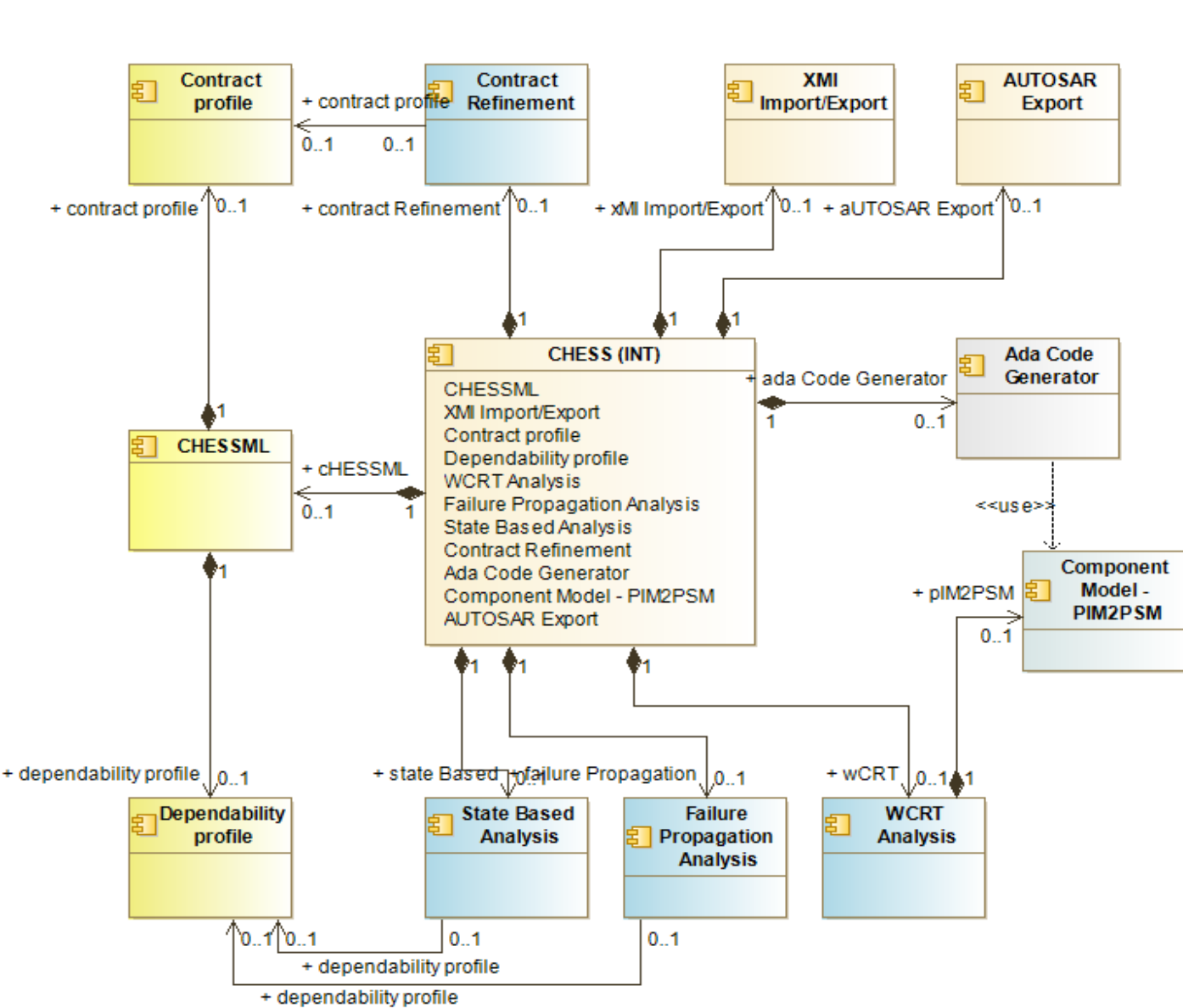


MegaM@Rt2 Approach



Individual tools modelling

- Each conceptual tool set sub-component and relevant interfaces have been refined to better satisfy the refined framework architecture and requirements



Document generation

The screenshot displays the MegaMaRt2ArchitectureD1.2 - Modelio 3.6 environment. On the left, a project tree shows the hierarchy: MegaMaRt2ArchitectureD1.2 > Architecture Level > MegaM@Rt Architecture > Conceptual Tool Set > MegaM@Rt Framework > Common Framework. A context menu is open over the 'Common Framework' folder, listing actions such as 'Create diagram...', 'Create matrix...', 'Create element', 'Subversion', 'Document Publisher', 'Excel Exchange', 'MegaMaRt', 'Modeler Module', 'SysML Architect by Modeliosoft', 'Add stereotype', 'Create stereotype...', 'Delete element', 'Cut element', 'Copy element', 'Paste element', 'Edit element...', 'Related diagrams...', 'Macros', 'Patterns', 'Import/Export', 'Check model', 'Administration', and 'Refinement'. The 'MegaMaRt' option is selected, with a sub-menu showing 'Full spec' and 'Frameworks section'. The right pane shows a document page with a table of requirements (Table 19) and a diagram (Table 20) illustrating the functional interfaces of the Modelio (SOFT) component.

Table 19 Deployment infrastructure

3.2 Modelio (SOFT)

Modelio (SOFT) is an open-source modeling environment supporting industry standards like UML and BPMN. Modelio provides a central repository for the local model, which allows various languages (UML2 profiles such as SysML and MARTE) to be combined in the same model, enabling abstraction layers to be managed and traceability between different model elements to be established. Modelio proposes various extension modules and can be used as a platform for building new Model-Driven Engineering (MDE) features such as code generation and reverse engineering of Java and C++. The environment enables users to build UML2 Profiles, and to combine them with a rich graphical interface for dedicated diagrams, model element property editors and action command controls.

3.2.1 Purpose of Modelio (SOFT) component

Properties	Purpose
Criticality: High Release: Baseline References:	MODELIO-010: Modelio shall provide system modeling capabilities in SysML.
Criticality: High Release: Initial References:	MODELIO-020: Modelio shall support holistic system engineering practices
Criticality: High Release: Baseline References:	MODELIO-030: Modelio shall support functional properties modeling on system level
Criticality: High Release: Baseline References:	MODELIO-040: Modelio shall support extra-functional properties modeling with MARTE
Criticality: High Release: Baseline References:	MODELIO-050: Modelio shall support functional properties in holistic system engineering approach
Criticality: High Release: Baseline References:	MODELIO-060: Modelio shall support extra-functional properties in holistic system engineering approach
Criticality: High Release: Baseline References:	MODELIO-070: Modelio shall manage traceability on Modelio project level
Criticality: High Release: Final References:	MODELIO-080: Modelio shall manage traceability on holistic system engineering level
Criticality: High Release: Final References:	MODELIO-090: Modelio shall visualize results of the runtime analysis on the system level

Table 20 Modelio (SOFT) component purpose

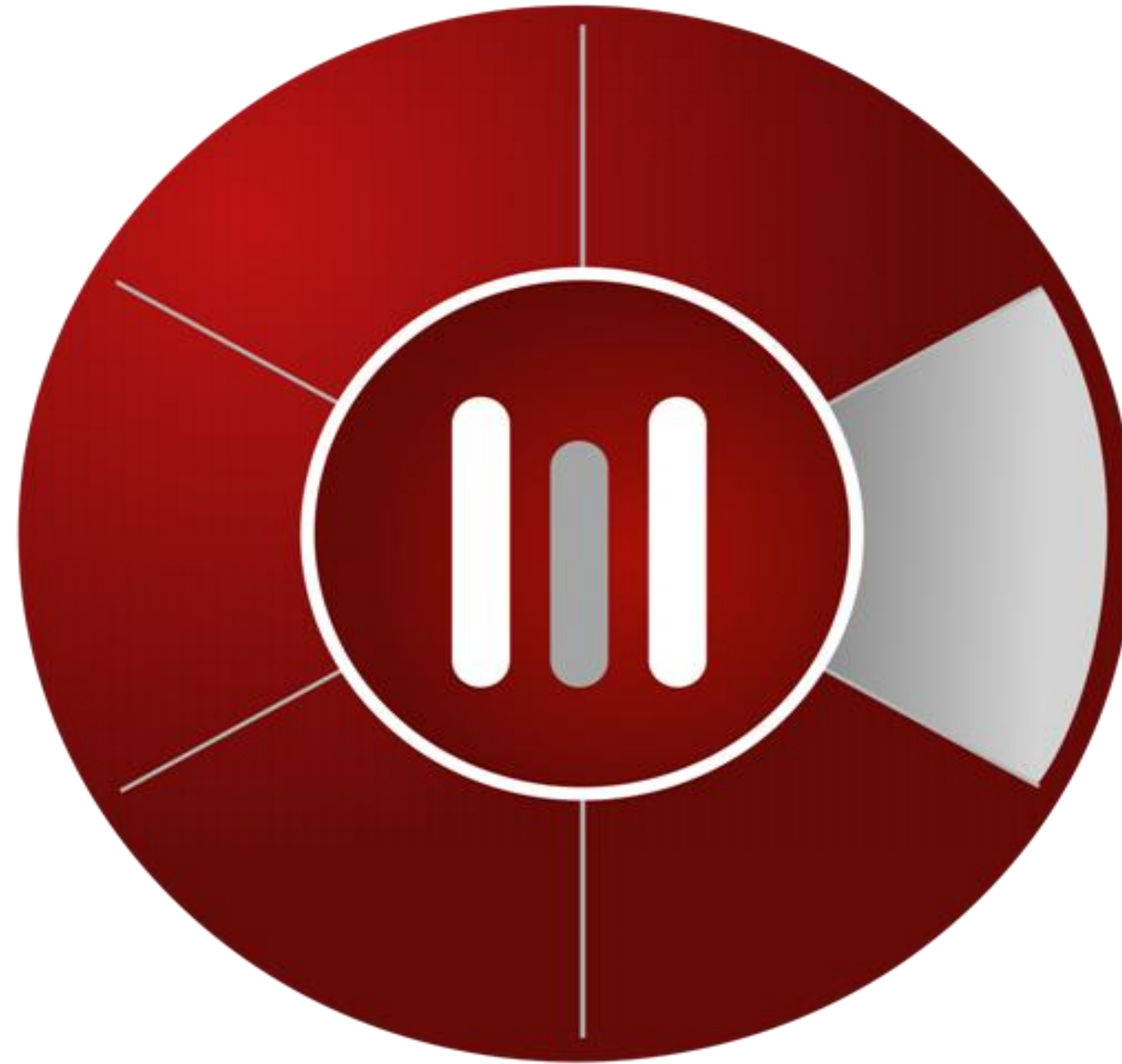
3.2.2 Functional Interfaces

```

classDiagram
    class Modelio_SOFT["Modelio (SOFT)"]
    Modelio_SOFT <|-- #EMF-XML
    Modelio_SOFT <|-- #OWG-XML
    Modelio_SOFT <|-- #Eclipse
    Modelio_SOFT <|-- #MODEL-EDIT-G.
    Modelio_SOFT <|-- #MODELIO-JAVA-API
    Modelio_SOFT <|-- #WORD-DOCUMENT
    Modelio_SOFT <|-- #HTML-DOCUMENT
    Modelio_SOFT <|-- #JAVA-CODE
    Modelio_SOFT <|-- #C++-CODE
    
```

Page 23 of 105

Demo with Modelio





Conclusions and discussion



Discussion of the MegaM@Rt approach

Advantages

- Technical coordination support
 - Requirements traceability
 - Live architecture document
 - Single model for everything
- Managing integrity of the project
 - Collaboration support
 - Common understanding
 - Common tool
 - Single reference
- Synchronization among WPs
 - Sharing specification approaches
 - Sharing document generators
 - Sharing document structures
- Useful tools
 - Document generation
 - Document templates
 - Traceability live view

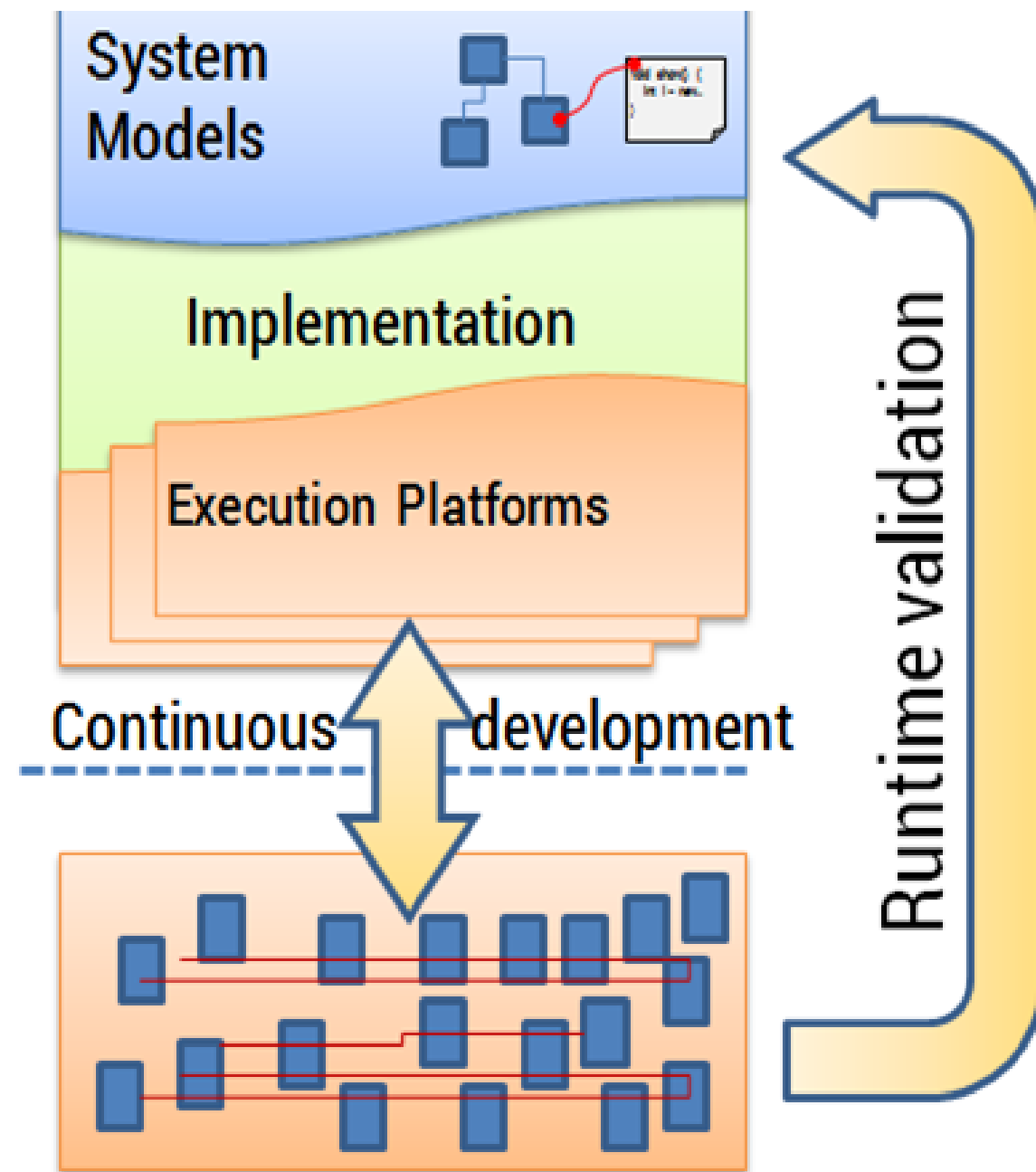
Limitations

- Very high-level modelling
- Learning curve
- Synchronizing contributions
- Styling of documents
- Still manual effort needed

Conclusions: Overall useful

- Can be used in many other similar projects
 - Uniformity, consistency for many contributors
- Approach is implementable in many tools

Thank you



- Contact info:
 - andrey.sadovykh@softeam.fr
 - a.sadovykh@innopolis.ru

- Useful links:
 - <https://megamart2-ecsel.eu/>

- More about tool
 - <http://www.ModelioSoft.com>
 - <http://www.modelo.org>

Conclusions: Overall useful

- Can be used in many other similar projects
 - Uniformity, consistency for many contributors
- Approach is implementable in many tools

Future work

- Specifying integration means
- Specifying tool chains for validation scenarios (eg BPMN)



MegaM@Rt²



ECSEL JU



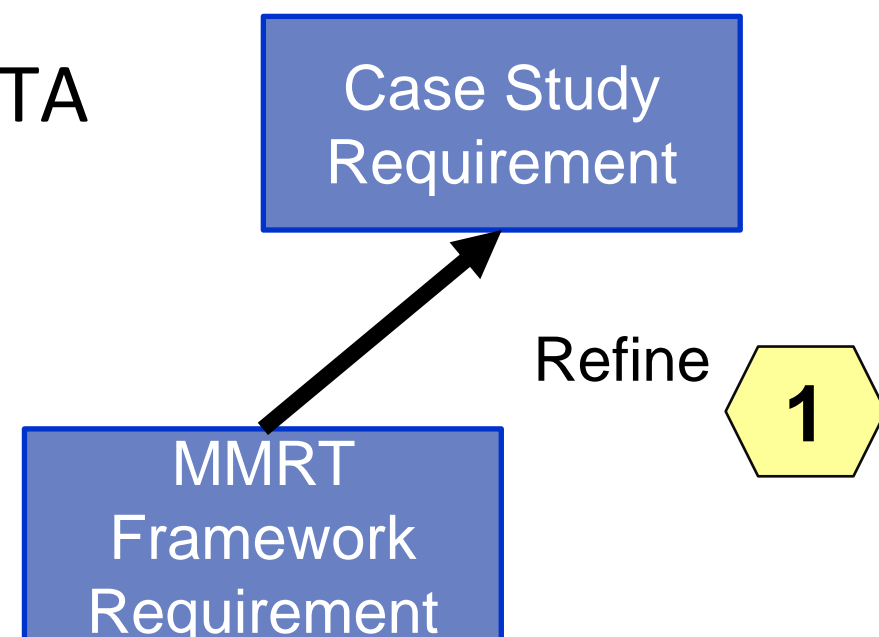


Backup

MMRT Framework Requirements Specification

A Top-Down approach

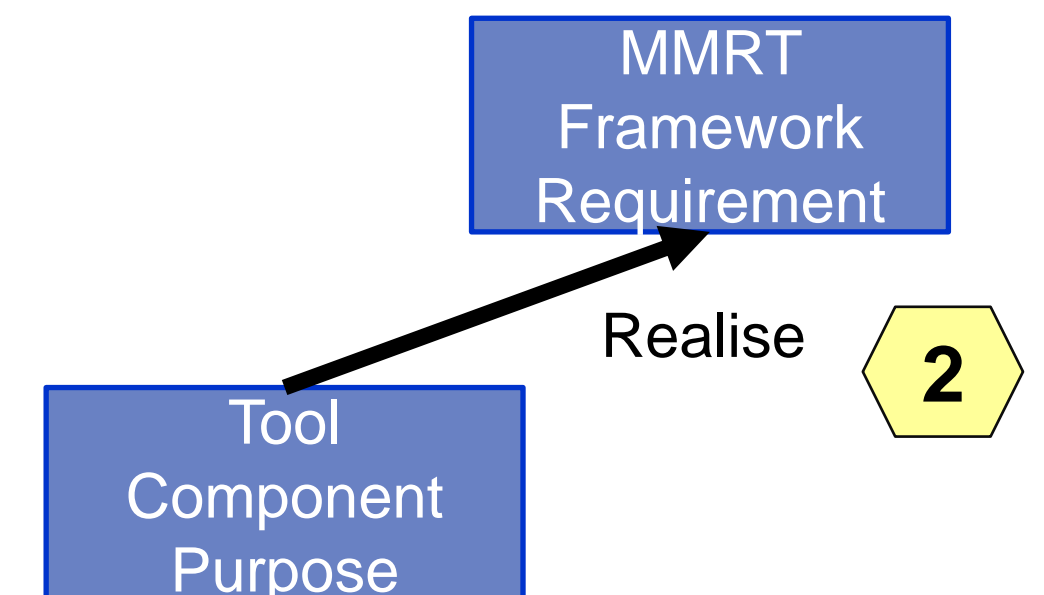
- From Case Studies Requirements
(a total of 106 CSRs
from 9 case study providers)
- To MMRT Framework Requirements
- Additional input form:
initial proposal, SOTA



A Bottom-up approach

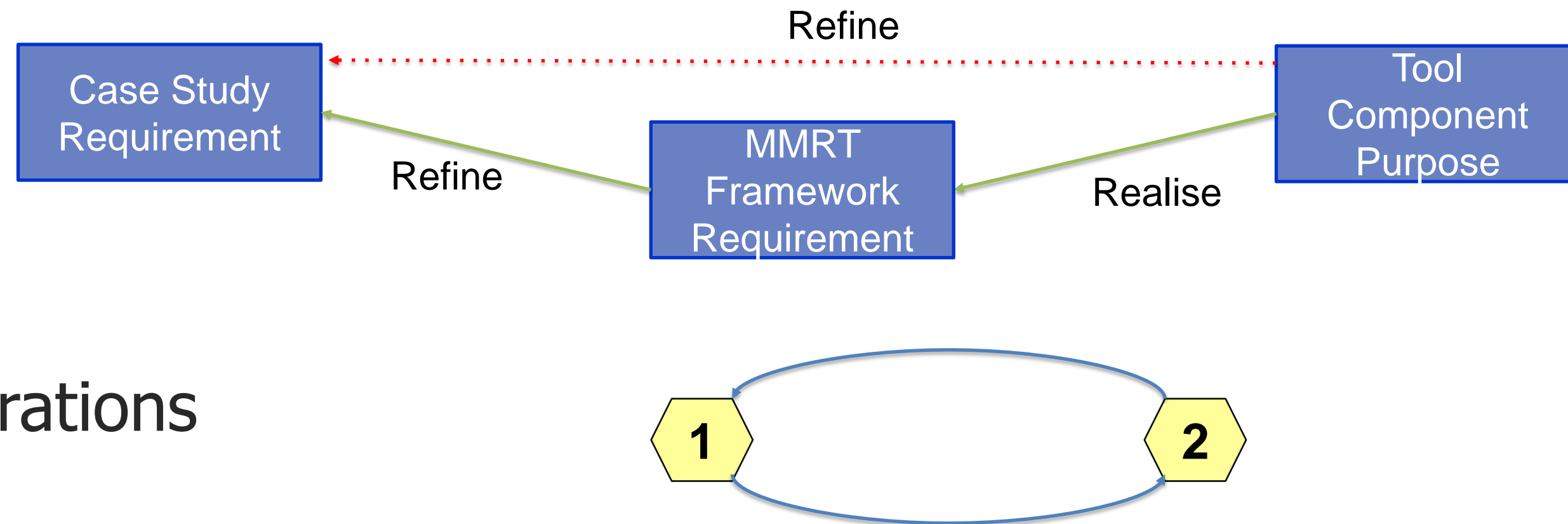
- Tools Components Purposes
(28 tools with a total of 223 different
TPs)
- Realise MMRT Framework
Requirements

- Framework requirements (FRs) bridge between
CSRs and TPs



Meet in the Middle

- Framework requirements (FRs) bridge between CSRs and TPs Purposes



- Refinement iterations

- Tool support:

- Modelio modeling tool to collect CSRs, TPs, create traceability matrices, and generate documentation

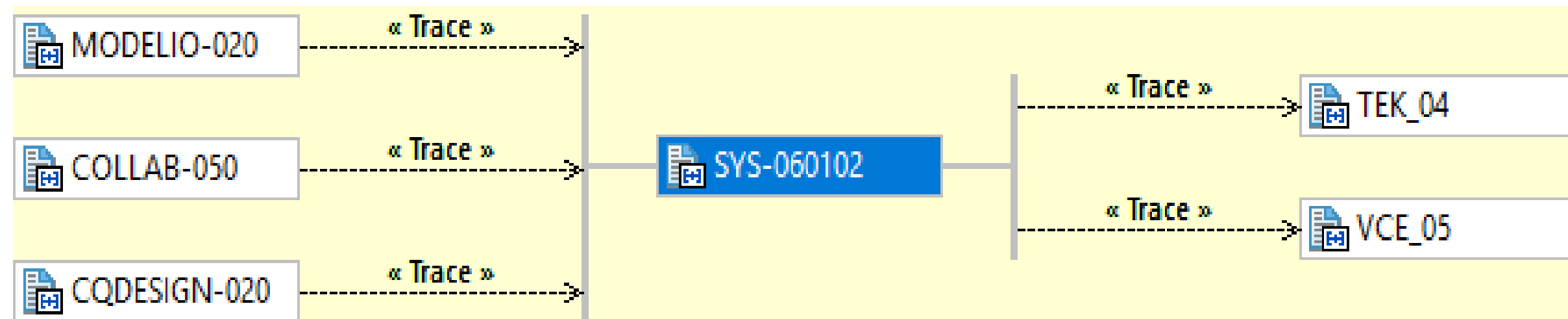
Mapping requirements by traceability matrix

	RTA-00004	RTA-00005	RTA-00006	RTA-00007	RTA-00008	RTA-00009	RTA-00010
NeoEMF-040							
AIPHS-010				↗	↗		
AIPHS-020				↗	↗		
AIPHS-030				↗	↗		
AIPHS-040							↗
AIPHS-050							
AIPHS-060				↗			
HepsyCode-010							
HepsyCode-020							
HepsyCode-030							
HepsyCode-040							
JTL-010	↗						
JTL-020	↗						
JTL-030			↗				
JTL-040			↗				
JTL-050			↗				
JTL-060						↗	↗

Gap Analysis

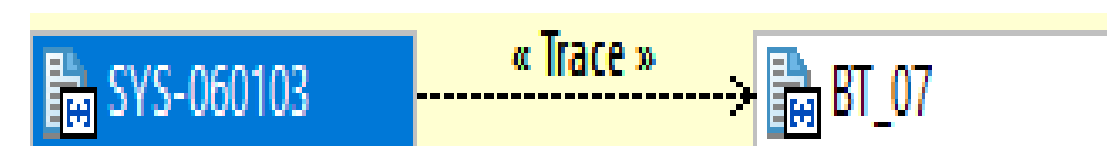
- Looking for:

- CSRs coverage

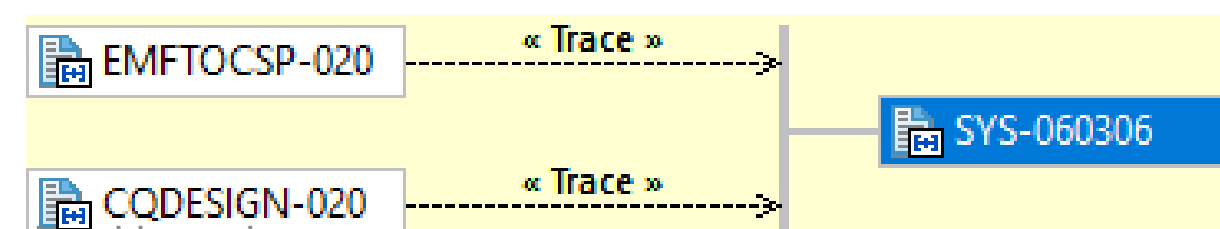


- Multiple tools satisfying a requirement: alternative available

- Unsatisfied CSR to define mitigation actions (e.g. sharing results from other projects, including additional tools, etc..)



- Added values by additional TPs unrelated to CSRs (i.e. suggestion to industrial partners for process improvement)



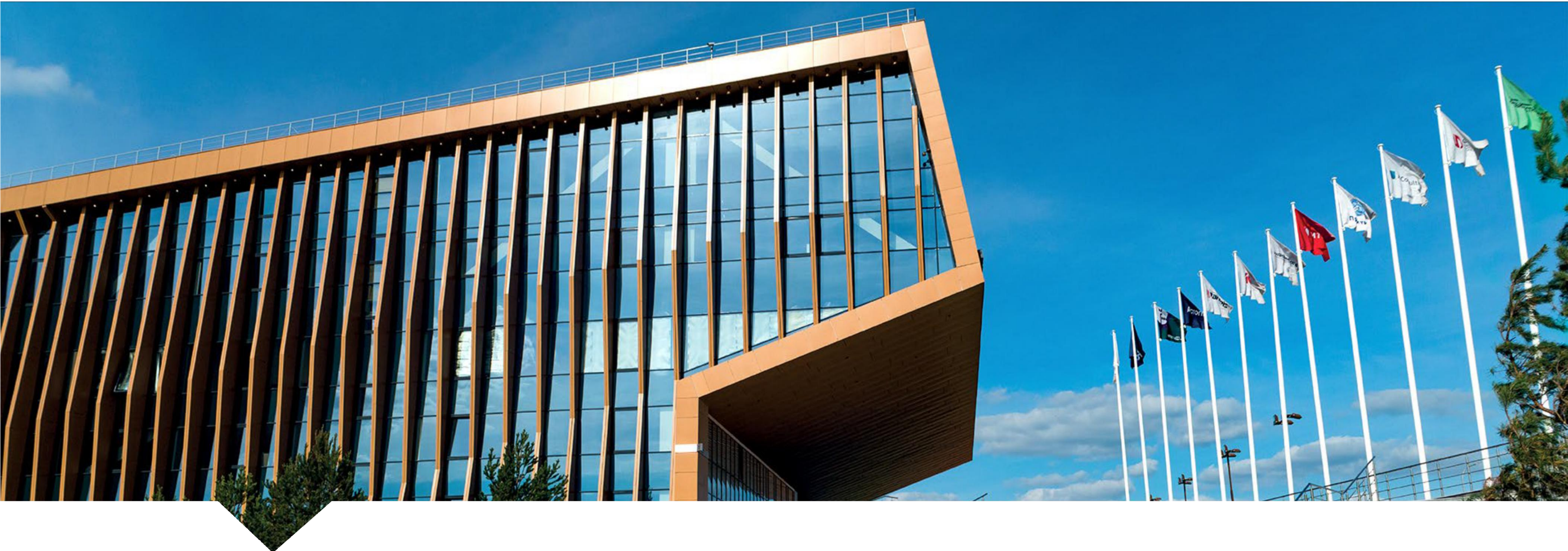
Gap Analysis Results - Overview

	#FRs	FRs not mapped to any CSRs	CSRs not satisfied by any TPs	TPs not mapping to any FR
WP2	37	3	1	0
WP3	39	4	1	10
WP4	15	0	0	2

Roadmap Analysis

- Project management can have a global vision over the tool sets
- Tool providers can plan their developments in the project
- Case study providers can plan the evaluation of tools

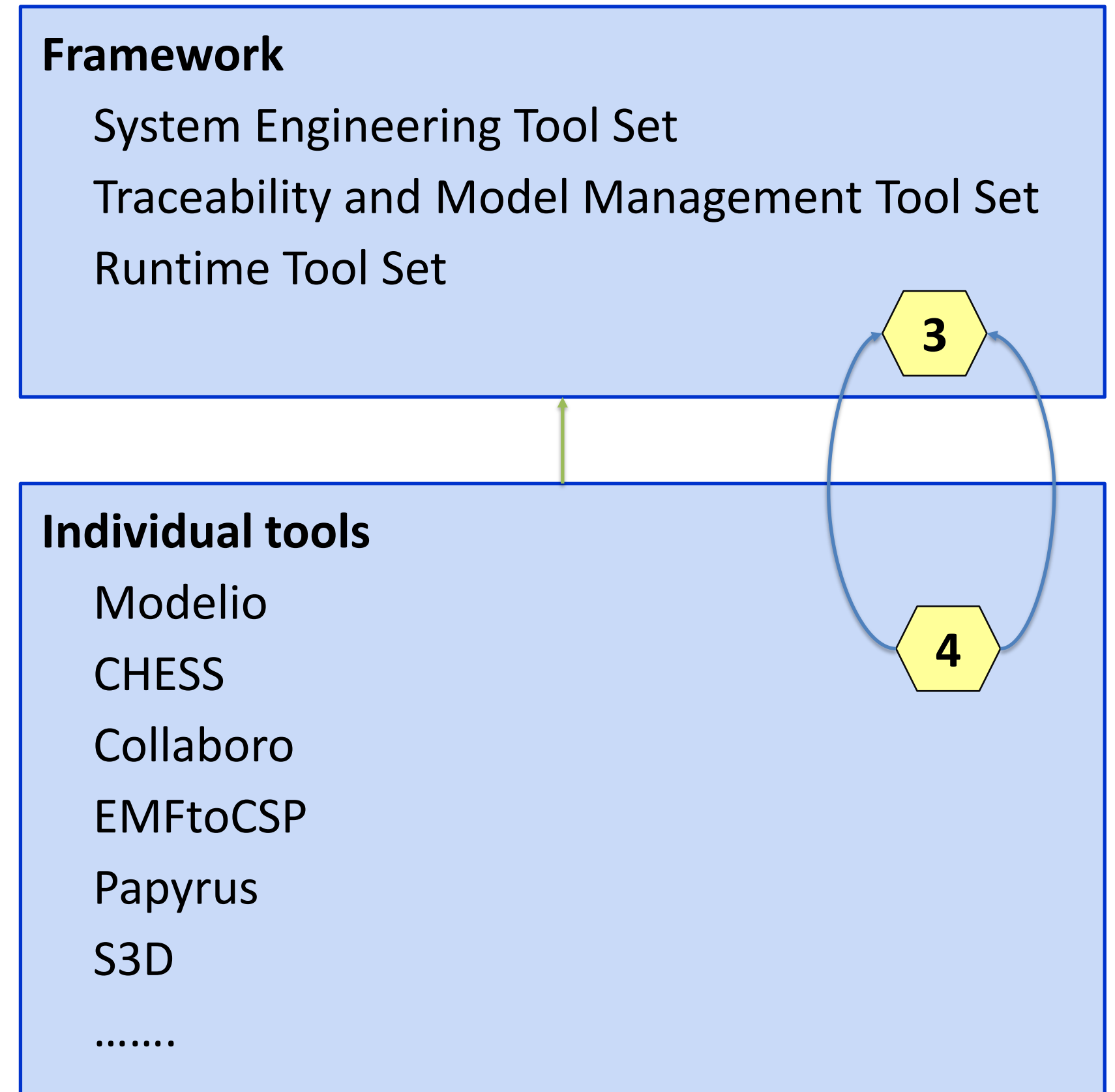
ID	Baseline (M0)	Initial (M15)	Intermediate (M20)	Final (M32)
RTA-00001	MODELIO-150		PAPYRUS-170, MODELIO-130	PAU-020, PAPYRUS-180, MODELIO-140
RTA-00002	CERTIFYIT-070, CQDESIGN-020, CQDESIGN-110			MBEETLE-010, PAU-020
RTA-00003	CERTIFYIT-010, RCRS-010, RCRS-020, RCRS-030, RCRS-040	LIME-010, LIME-020	LIME-030, RCRS-050, RCRS-080	MBEETLE-010, MODELIO-100, LIME-040, RCRS-060, RCRS-070
Model-Based System Engineering in Practice: Document Generation - MegaM@Rt Project Experience ©Andrey Sadovykh et al.



Architecture Management Approach

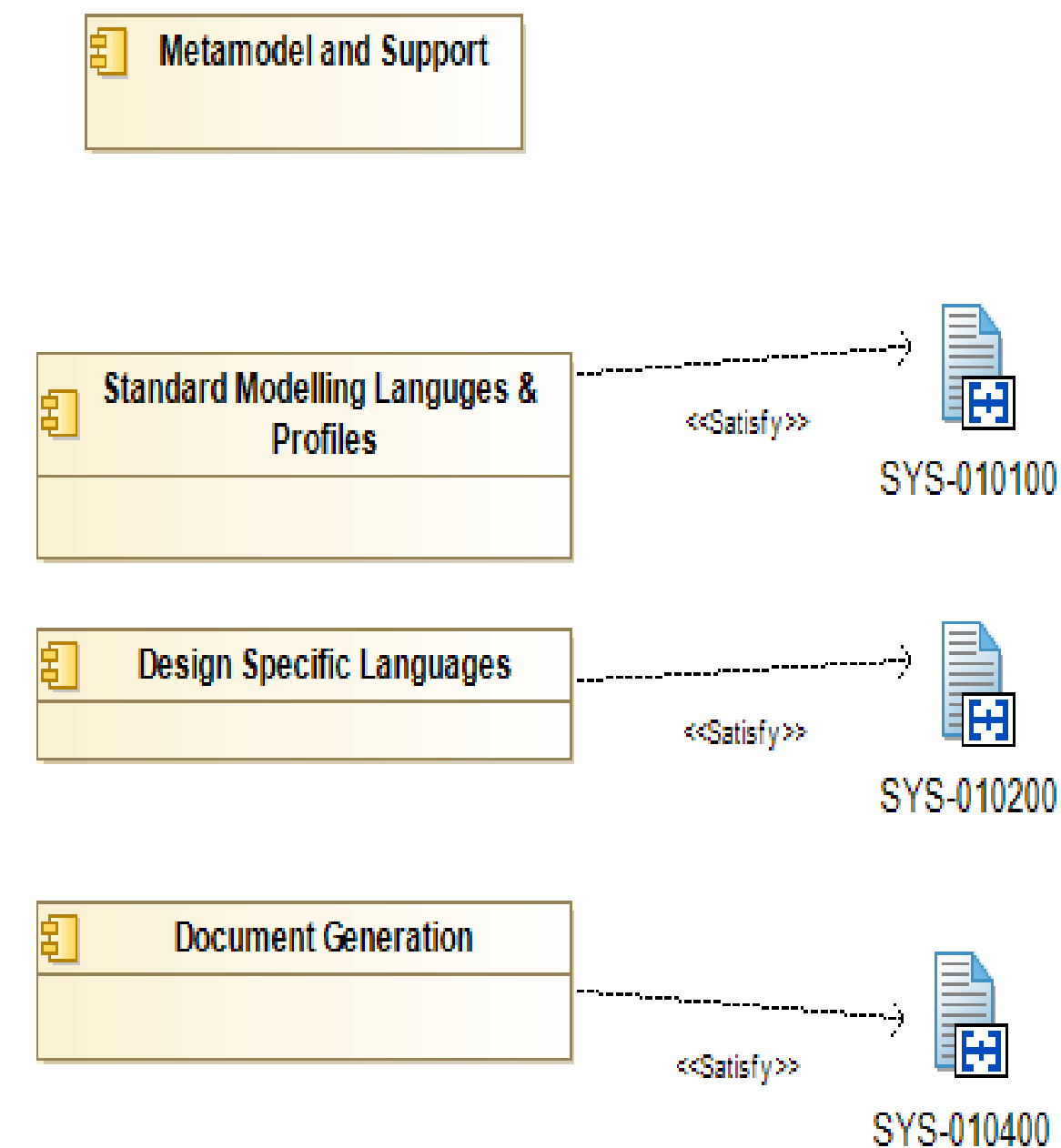
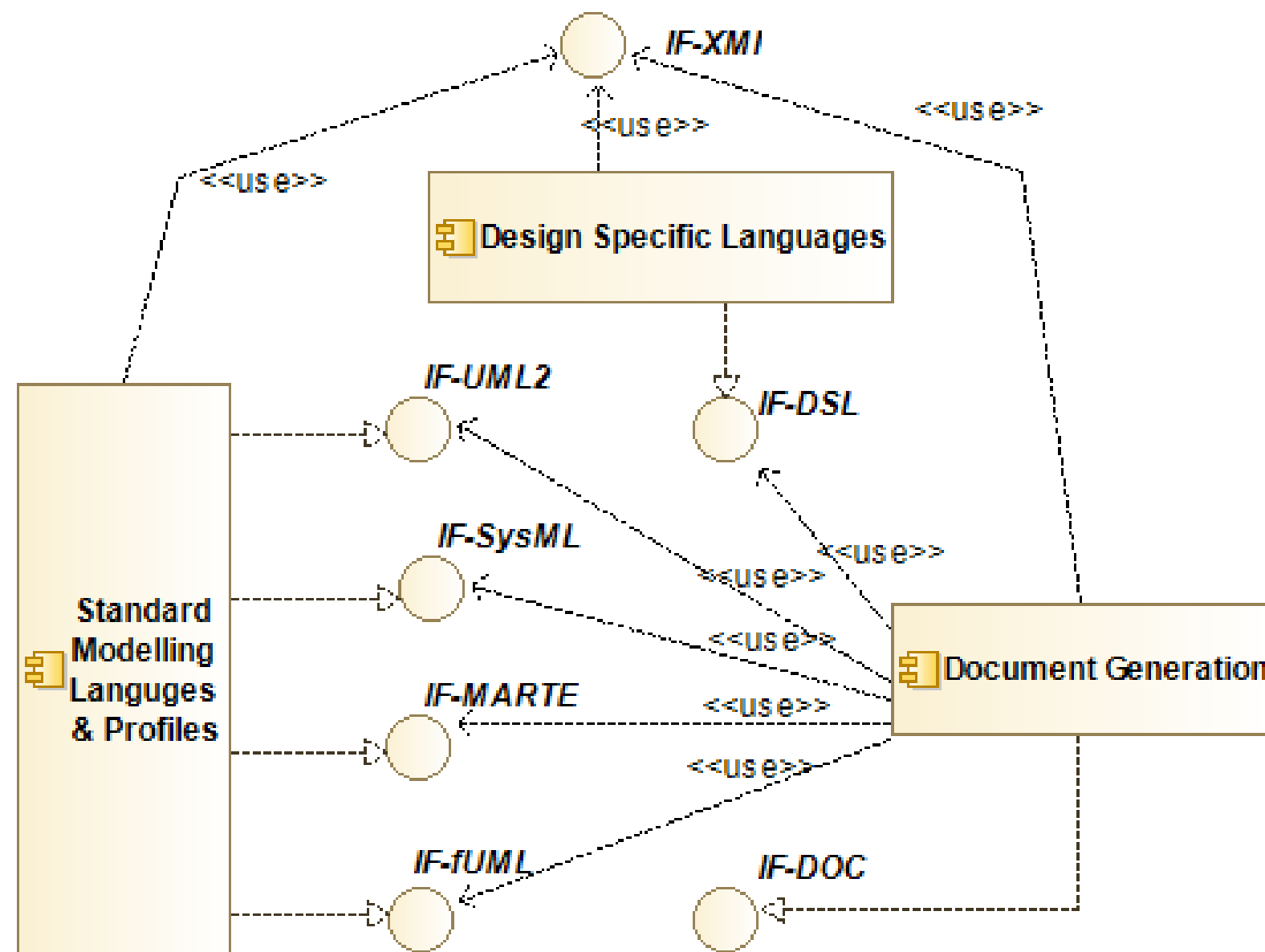
Approach for high-level architecture

- Major element = Tool component
- Services
 - Purposes
 - Functional interfaces Realises
 - Subordinate components
- Integration means and Deployment
 - Interfaces / data exchange
 - Deployment
- Relation to the Framework



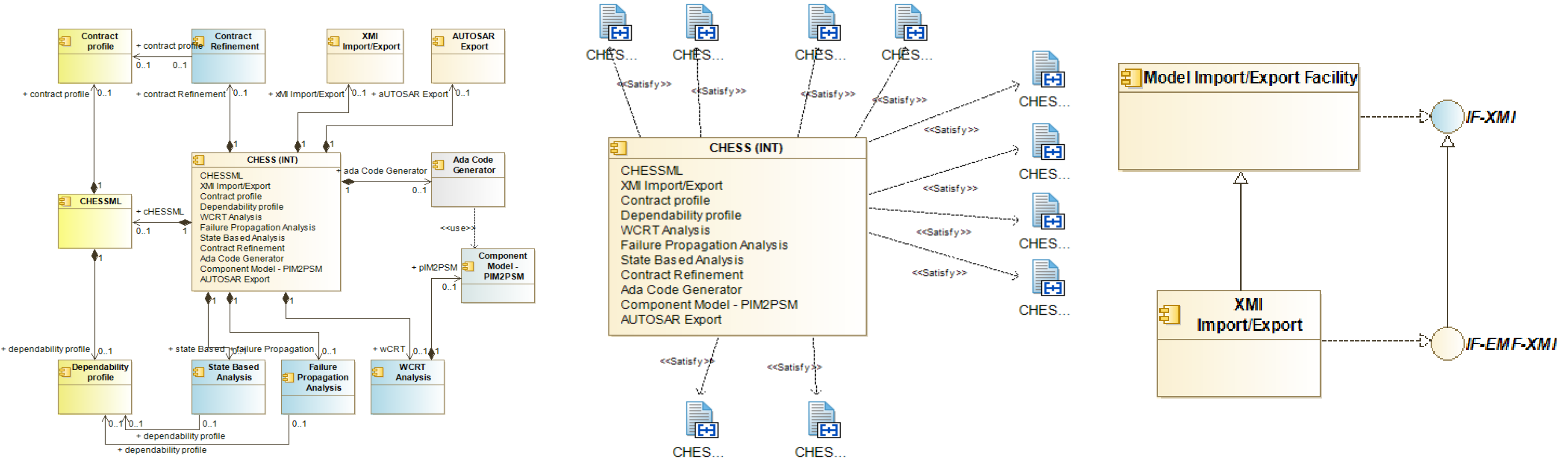
Framework Modelling

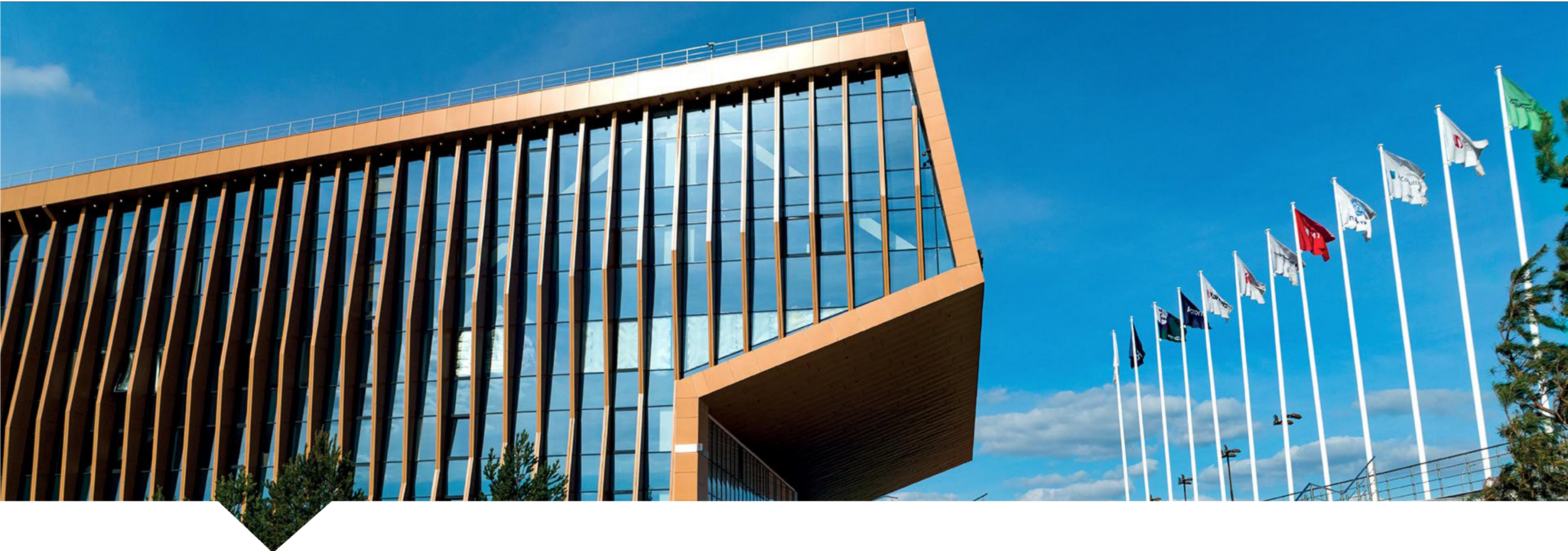
- MegaM@Rt Framework highlight interfaces to support tools integration and traceability to requirements to guarantee and allow checking model consistency



Individual tools modelling

- Each conceptual tool set sub-component and relevant interfaces have been refined to better satisfy the refined framework architecture and requirements

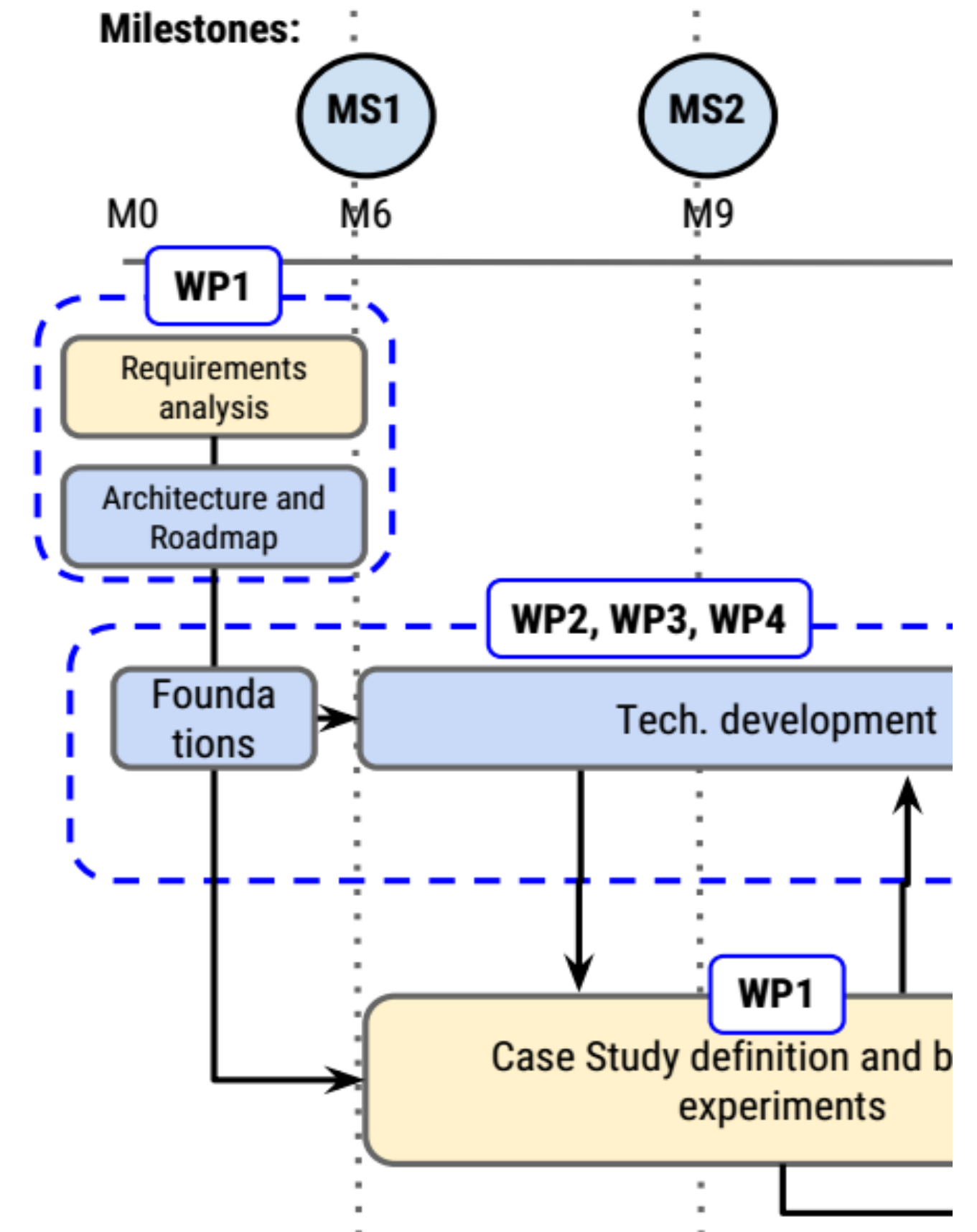




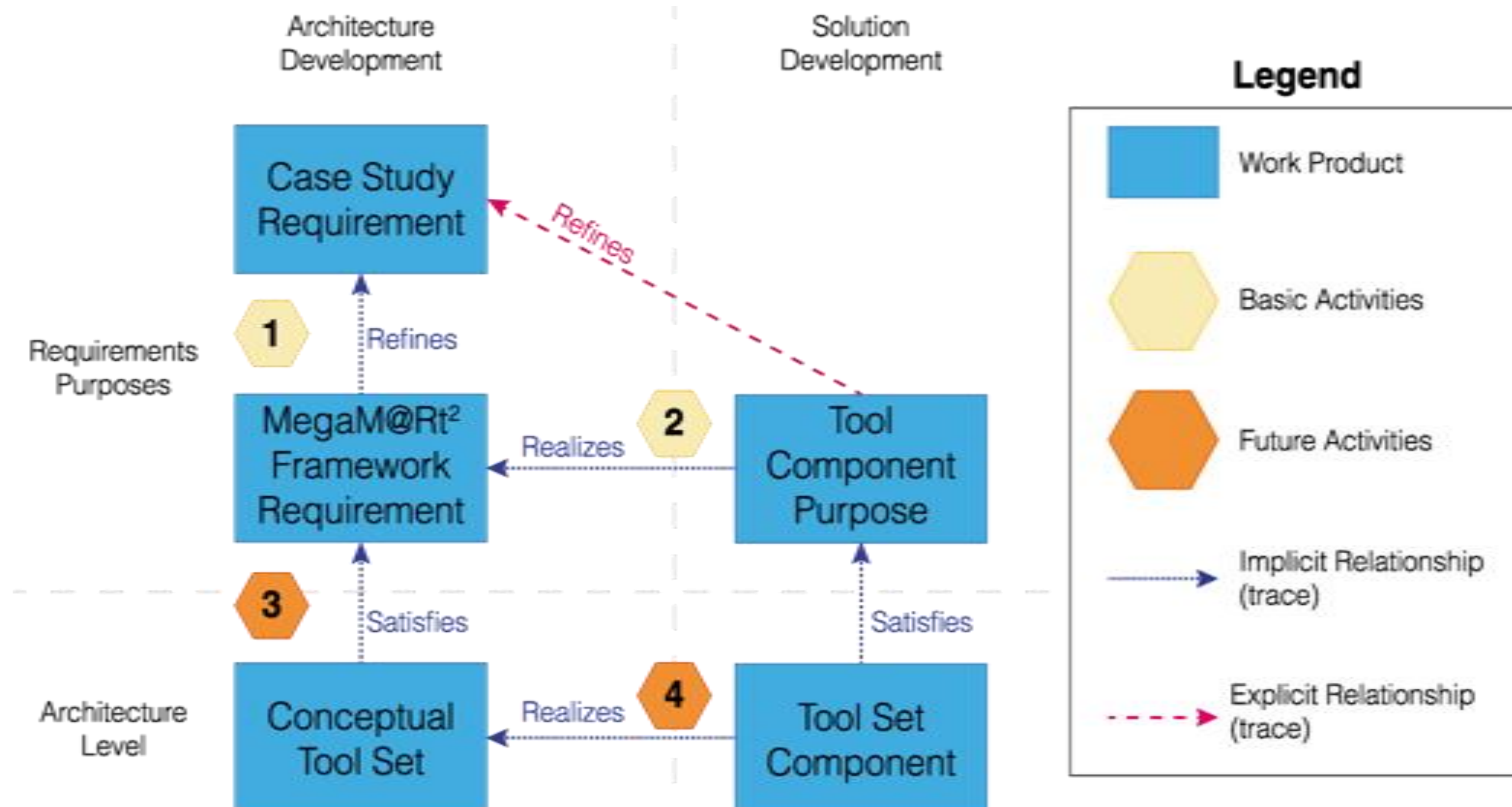
Questions?

D1.2 Architecture specification and roadmap - initial version

- Framework
 - Conceptual tools as described in the FPP
 - Individual tools by partners
- Properties
 - High-level requirements help to identify the features, goals and objectives
 - Functional interfaces and services
 - Subordinates
 - Deployment



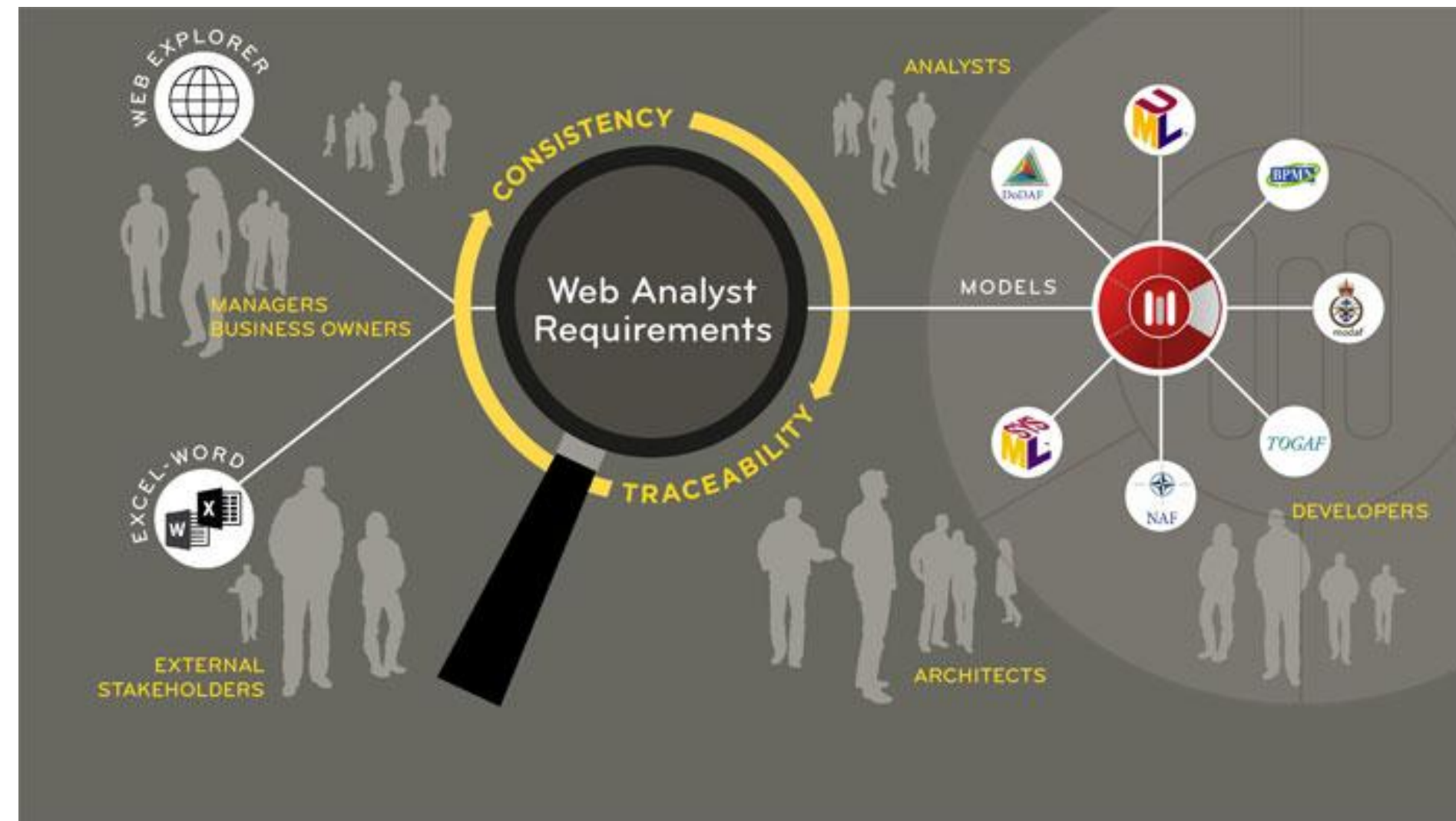
Specification process



Tooling

- Modelio
- Analyst
- Document publisher

- Constellation
- Collaborative modelling
- Configuration management

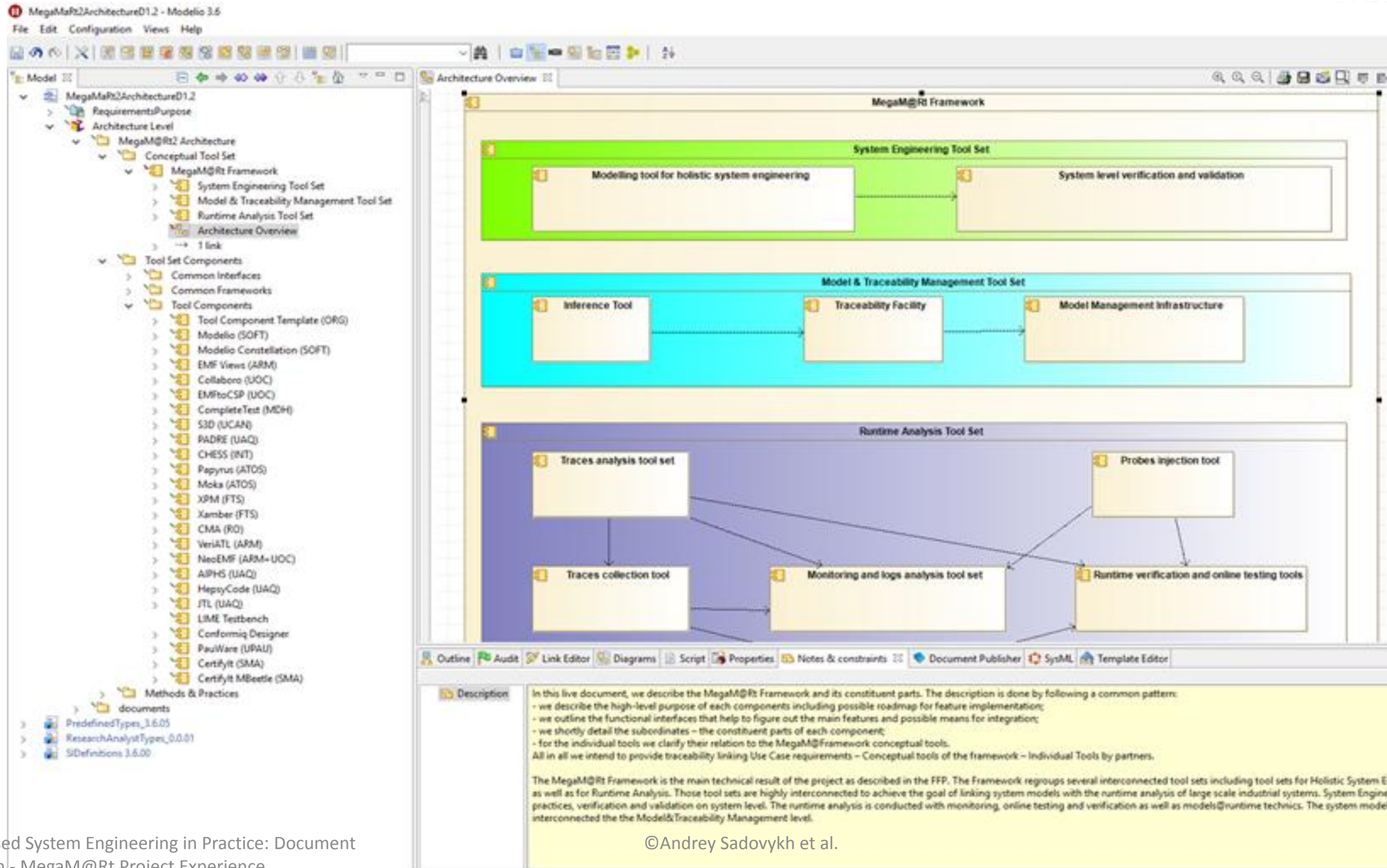


Requirements editing

The screenshot displays the MegaMaRt2Architecture - Modelio 3.6 application. The interface includes a menu bar (File, Edit, Configuration, Views, Help), a toolbar with various icons, and a main workspace. On the left, a 'Model' tree view shows a hierarchical structure of requirements, including 'RequirementsPurpose', 'Case Study Scenarios', 'Case Study Requirements', 'MegaM@Rt Tool Set Requirin', 'MMRT Framework Rec', 'System Enginee', 'Runtime Analysi', 'System Enginee', 'Model & Tracea', 'Tool component purp', 'Goals', 'Dictionary', 'Architecture Level', 'MegaM@Rt2 Architecture', 'Conceptual Tool Set', 'MegaM@Rt Fran', 'Tool Set Components', 'Common Interfa', and 'Common Frame'. The main workspace shows a table of requirements with columns for Id, Definition, Criticality, Release, and References. The table contains 20 rows of requirements, each with a unique ID and a detailed definition. The 'Criticality' column is set to 'High' for all entries, and the 'Release' column is set to 'Final' or 'Baseline'. The 'References' column is empty for all entries. The table is currently in 'Table' view, as indicated by the 'Table Form' tabs at the bottom.

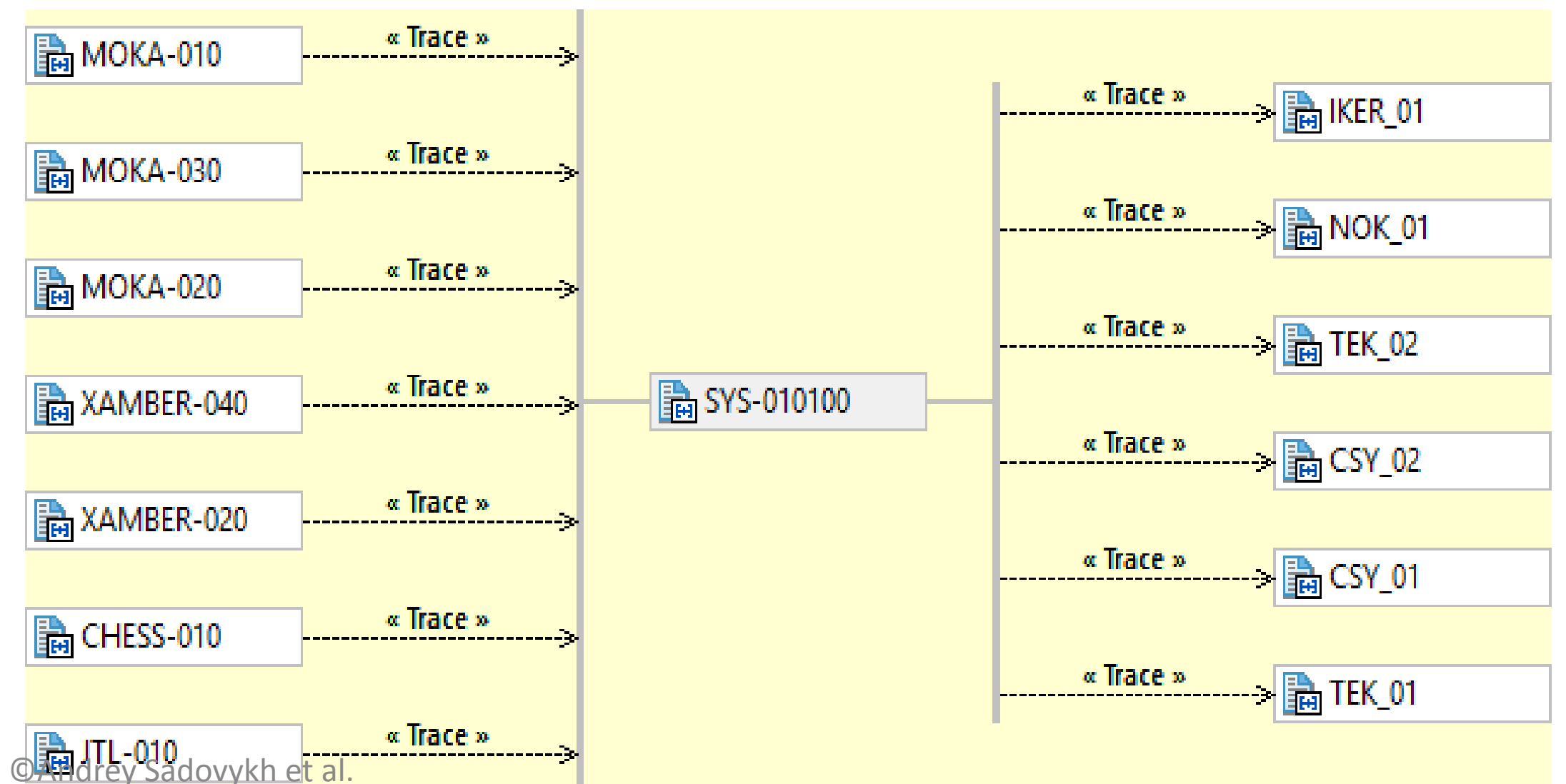
	Id	Definition	Criticality	Release	References
5	SYS-000001	The SE must support a teamwork collaboration environment.	High	Final	
6	SYS-000002	The SE must provide an advanced graphical user interface (GUI) to simplify the	High	Final	
7	SYS-000004	The SE must support the HW/SW co-design.	High	Final	
8	SYS-000005	The SE must provide smarter design exploration techniques based on a pre-analysis	High	Final	
9	SYS-010100	The SE must support standard modelling languages, standard profiles (i.e. AADL, UML, SysML, MARTE, FUMML, UTP) and profile customisation capability.	High	Baseline	
10	SYS-010200	The SE must support domain specific modelling languages (i.e. EAST-ADL, FDB).	High	Baseline	
11	SYS-010400	The SE must provide the mean to generate project and design documents from the	High	Baseline	
12	SYS-020201	The SE must allow modelling the functional requirements based on stakeholder	High	Final	
13	SYS-020202	The SE must allow modelling the non-functional/extra-functional requirements	High	Final	
14	SYS-020300	The SE must extend the "intra-model" requirements traceability across the whole	High	Baseline	
15	SYS-030101	The SE must support the architectural views definition and modelling.	High	Final	
16	SYS-030301	The SE must provide modelling adopting separation of concerns principle.	High	Baseline	
17	SYS-030401	The SE must support the system variability modelling	High	Baseline	
18	SYS-030402	The SE must support the reuse of existing models or patterns	High	Baseline	
19	SYS-040201	The SE must provide the means to model non-functional/extra-functional	High	Baseline	
20	SYS-060000	The SE must provide advanced test design capabilities to support the verification	High	Baseline	

Architecture



Traceability

SYS-CSR	SYS-010100	SYS-010200	SYS-010400	SYS-020201
CSY_01	✓			
CSY_02	✓	✓		
CSY_03				
CSY_04				
CSY_05				
IKER_01	✓	✓		
IKER_04		✓		
IKER_16				
IKER_18				
NOK_01	✓	✓		
NOK_02				✓
NOK_03		✓		
NOK_04				



Document generation

The screenshot displays the MegaMaRt2ArchitectureD1.2 - Modelio 3.6 environment. On the left, a hierarchical project tree shows the structure of the architecture, including 'MegaMaRt2ArchitectureD1.2', 'RequirementsPurpose', 'Architecture Level', and 'MegaM@Rt2 Architecture'. A context menu is open over the 'MegaMaRt' component, listing actions such as 'Create diagram...', 'Create matrix...', 'Create element', 'Subversion', 'Document Publisher', 'Excel Exchange', 'MegaMaRt', 'Modeler Module', 'SysML Architect by Modeliosoft', 'Add stereotype', 'Create stereotype...', 'Delete element', 'Cut element', 'Copy element', 'Paste element', 'Edit element...', 'Related diagrams...', 'Macros', 'Patterns', 'Import/Export', 'Check model', and 'Administration'. The right pane shows a document viewer displaying 'Table 19 Deployment infrastructure' and '3.2 Modelio.(SOFT)'. Below the text, there is a table of properties and purposes for the Modelio component, and a diagram showing functional interfaces for Modelio.(SOFT).

Table 19 Deployment infrastructure

3.2 Modelio.(SOFT)

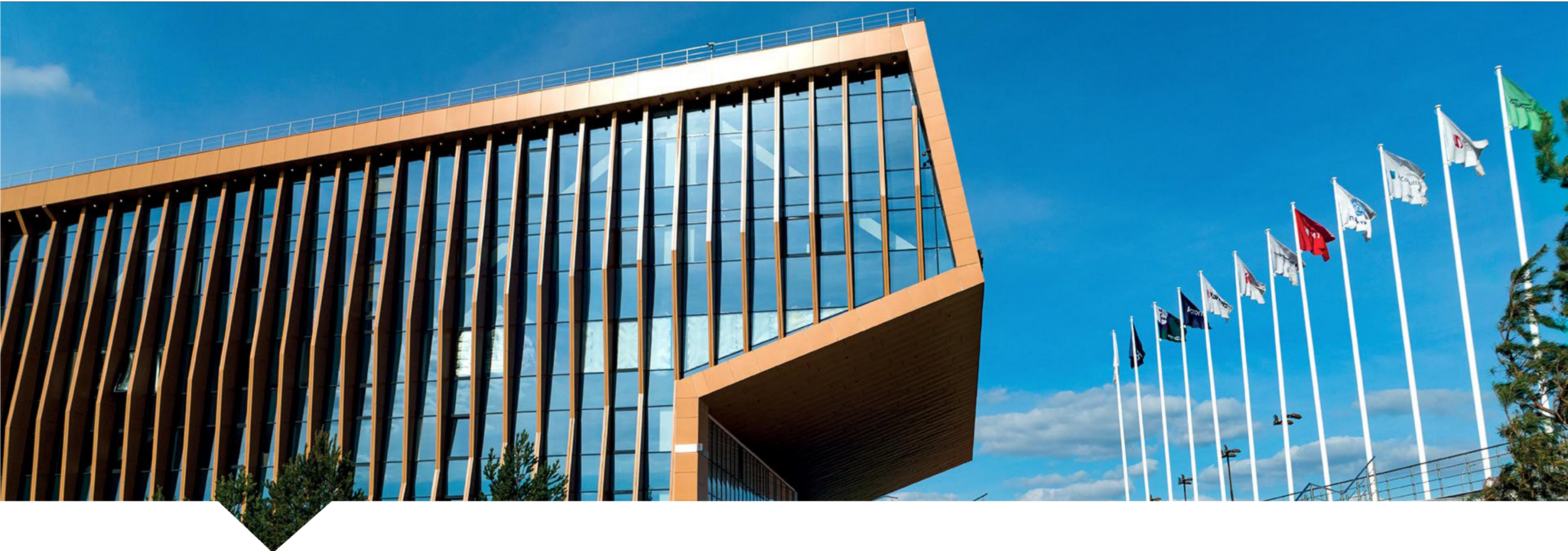
Modelio.(SOFT) is an open-source modeling environment supporting industry standards like UML and BPMN. Modelio provides a central repository for the local model, which allows various languages (UML2 profiles such as SysML and MARTE) to be combined in the same model, enabling abstraction layers to be managed and traceability between different model elements to be established. Modelio proposes various extension modules and can be used as a platform for building new Model-Driven Engineering (MDE) features such as code generation and reverse engineering of Java and C++. The environment enables users to build UML2 Profiles, and to combine them with a rich graphical interface for dedicated diagrams, model element property editors and action command controls.

3.2.1 Purpose of Modelio.(SOFT) component

Properties	Purpose
Criticality: High Release: Baseline References:	MODELIO-010: Modelio shall provide system modeling capabilities in SysML.
Criticality: High Release: Initial References:	MODELIO-020: Modelio shall support holistic system engineering practices
Criticality: High Release: Baseline References:	MODELIO-030: Modelio shall support functional properties modeling on system level
Criticality: High Release: Baseline References:	MODELIO-040: Modelio shall support extra-functional properties modeling with MARTE
Criticality: High Release: Baseline References:	MODELIO-050: Modelio shall support functional properties in holistic system engineering approach
Criticality: High Release: Baseline References:	MODELIO-060: Modelio shall support extra-functional properties in holistic system engineering approach
Criticality: High Release: Baseline References:	MODELIO-070: Modelio shall manage traceability on Modelio project level
Criticality: High Release: Final References:	MODELIO-080: Modelio shall manage traceability on holistic system engineering level
Criticality: High Release: Final References:	MODELIO-090: Modelio shall visualise results of the runtime analysis on the system level

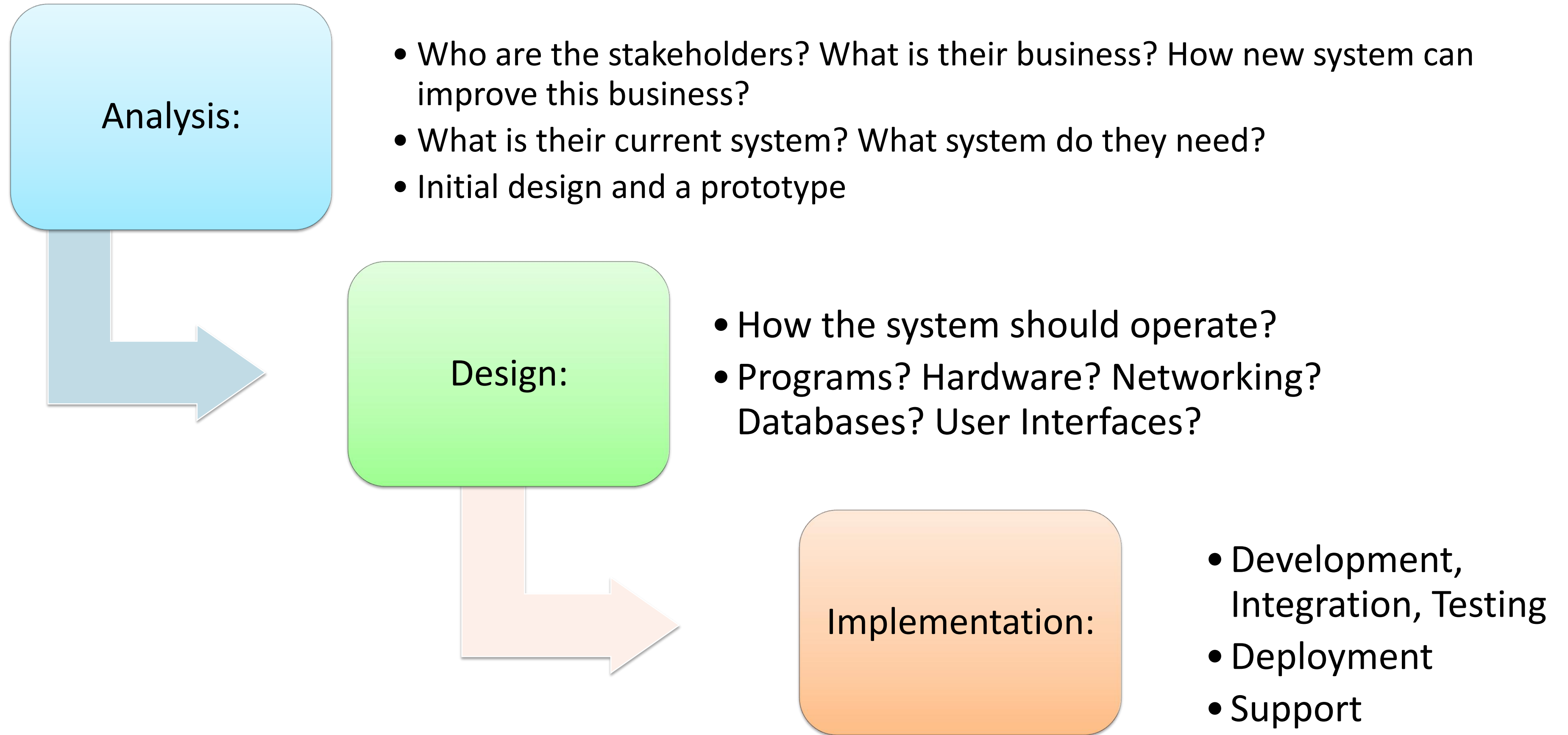
Table 20 Modelio.(SOFT) component purpose

3.2.2 Functional Interfaces

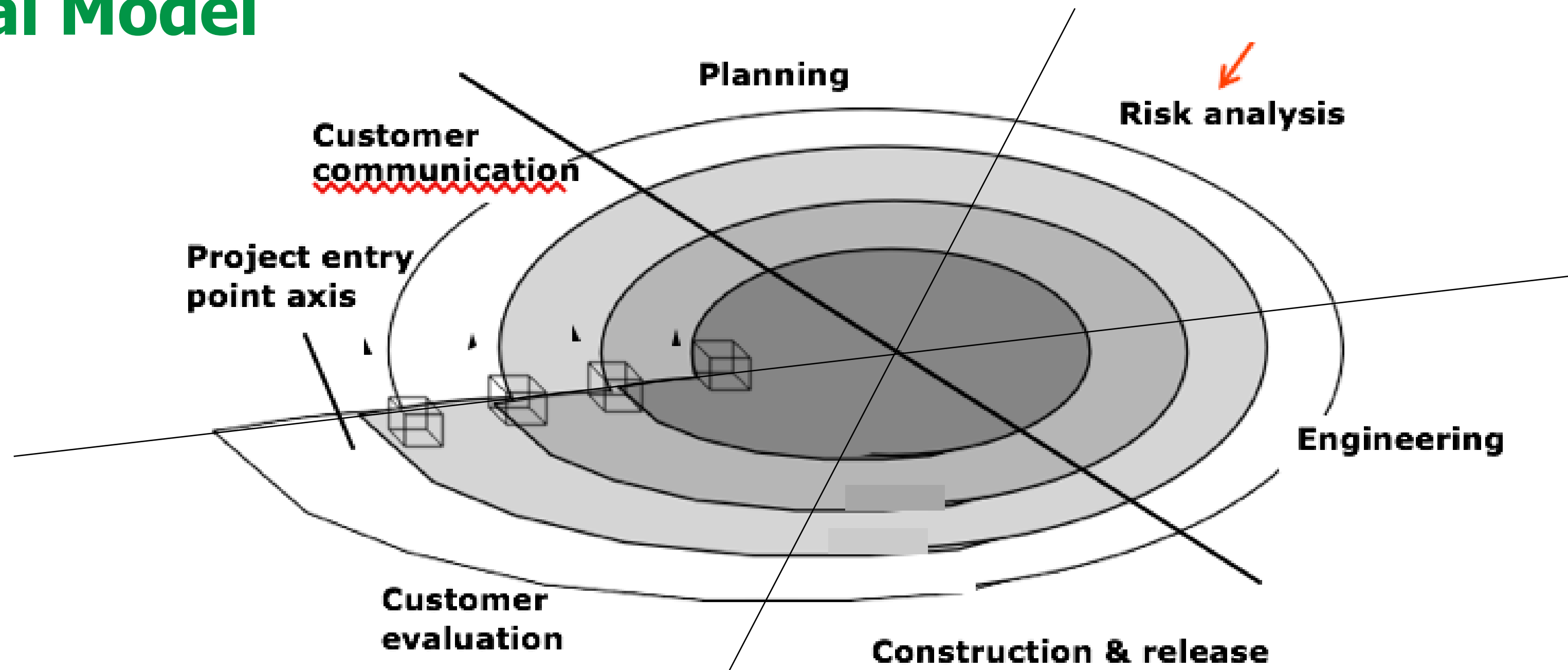


Additional slides

Example: System Development Lifecycle



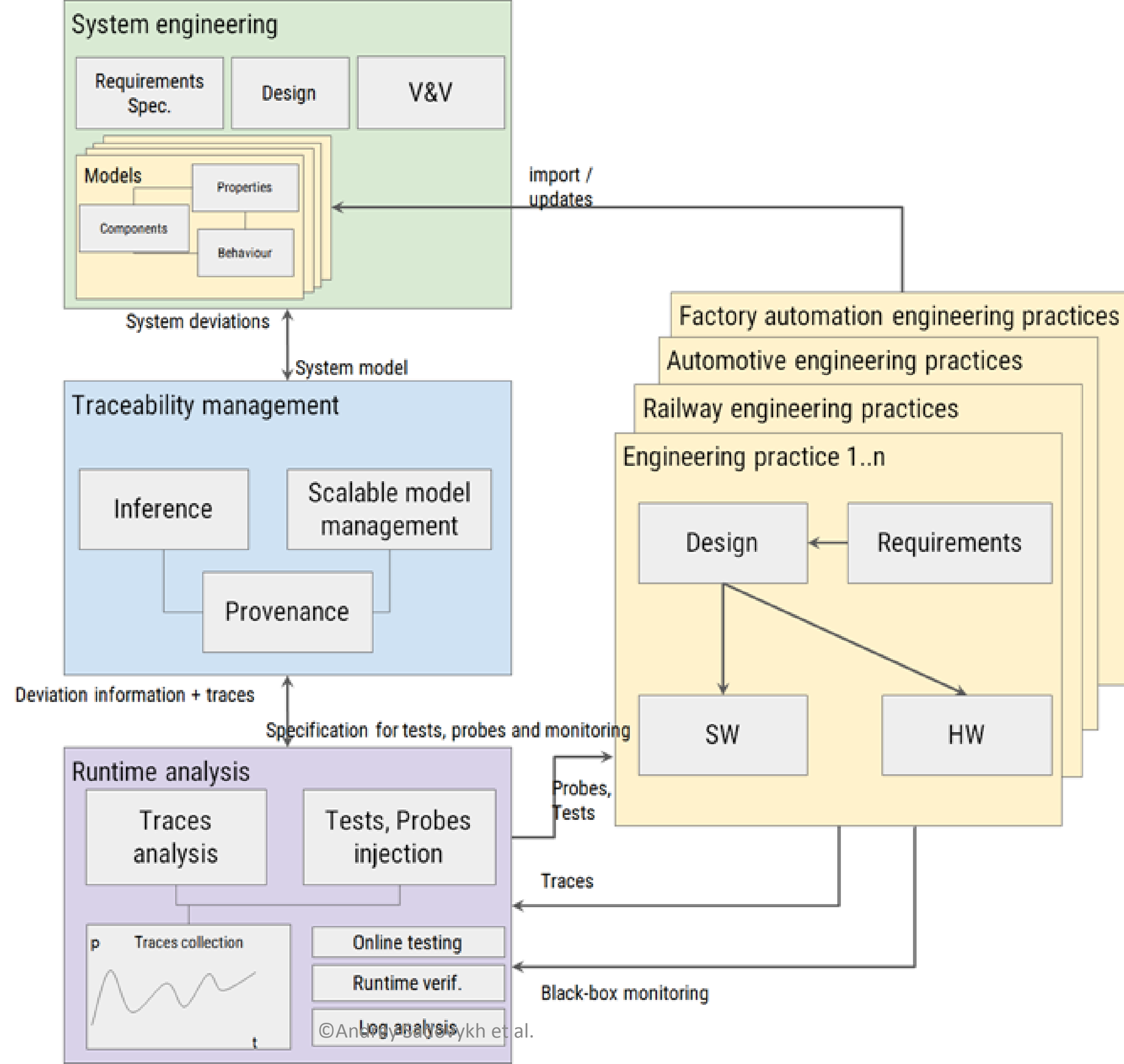
Spiral Model



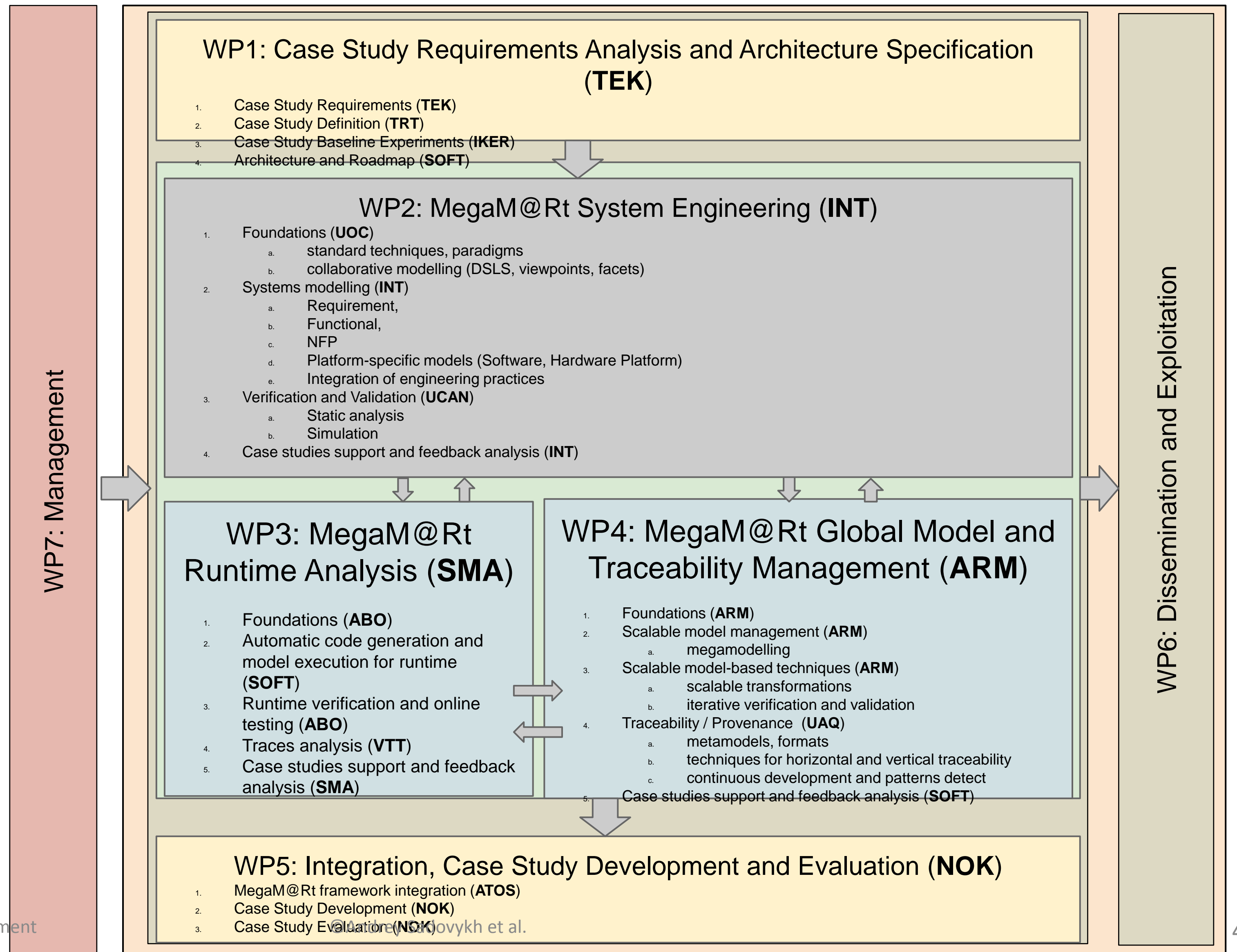
-  Product maintenance projects
-  Product enhancement projects
-  New product development projects
-  Concept development projects

Roger S. Pressman's "Software Engineering, a Practitioners Approach"

Overall approach

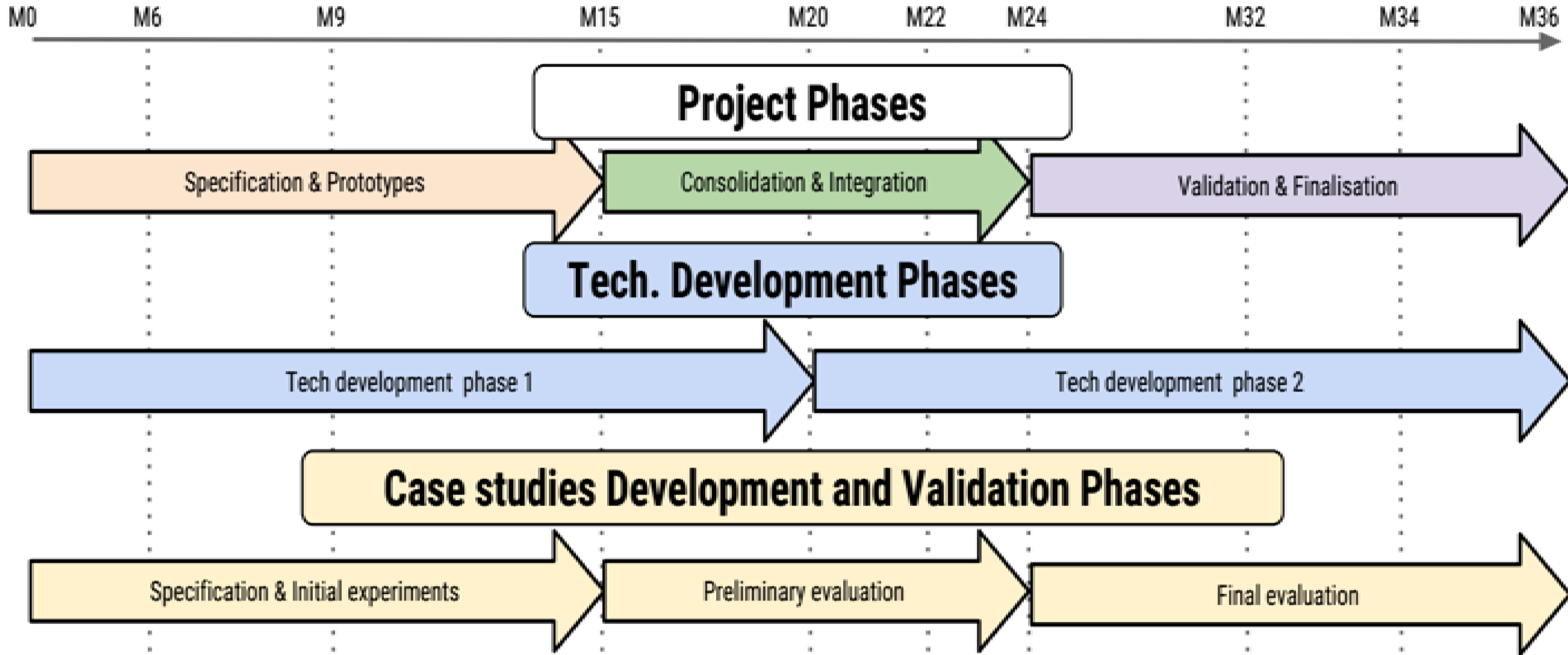


Work packages

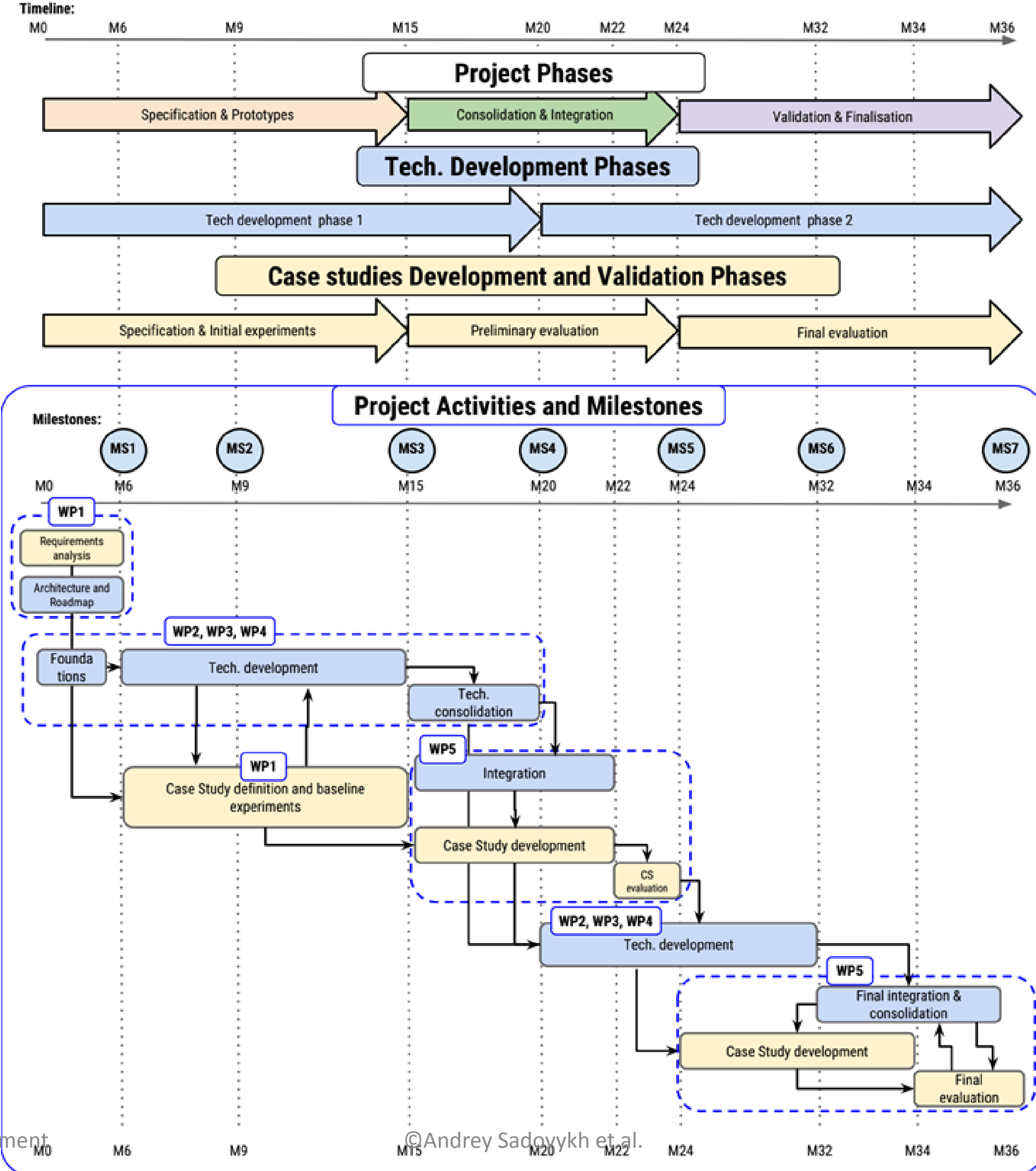


Phases

Timeline:



Process



Next steps for MegaM@Rt modelling approach

- Matchmaking for Case Studies and Tools
- Baseline experiments and refinement of requirements, purposes, roadmap and architecture
- Tracing status on features delivery
- Planning integration
- Case study requirements coverage monitoring

Hackathon session

- Boost collaboration
- Speed-up baseline experiments and prototyping
- Base for validation scenarios activities
- Early evaluation and results
- Feedbacks to design

