Technische Universität Braunschweig

Institut für Verkehrssicherheit und Automatisierungstechnik iVA
Prof. Dr.-Ing. Dr. h.c. mult. E. Schnieder

openETCS D2.1: Report on existing methodologies

Work-Package 2: "Requirements for Open Proofs“
Task 2.1.1 – State of the Art

Jan Welte and Hansjörg Manz

22.11.2012

Report on existing methodologies
Agenda

1. Introduction
2. Interviews
3. Means of Description
4. Methodologies
5. Tools
6. Requirements
7. Summary

1. Introduction

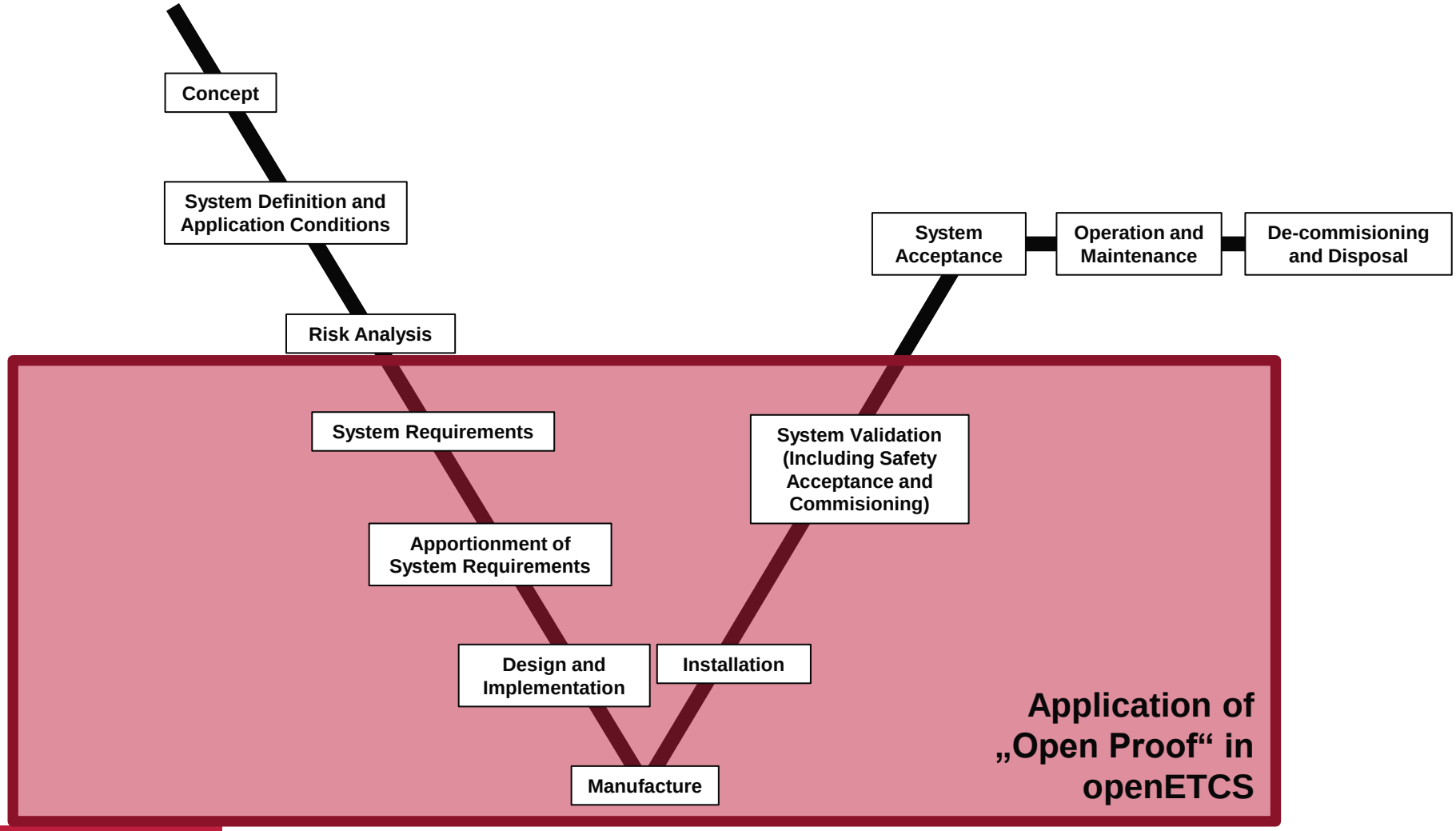
Functional steps for software development

The EN 50128 requires the following functional steps for software development:

- Define the Software Requirements Specification and in parallel consider the software architecture (software architecture is where the basic safety strategy is developed for the software and the software safety integrity level);
- Design, develop and test the software according to the Software Quality Assurance Plan, software safety integrity level and the software lifecycle;
- Integrate the software on the target hardware;
- Validate the software;
- Maintain software during the operational life.

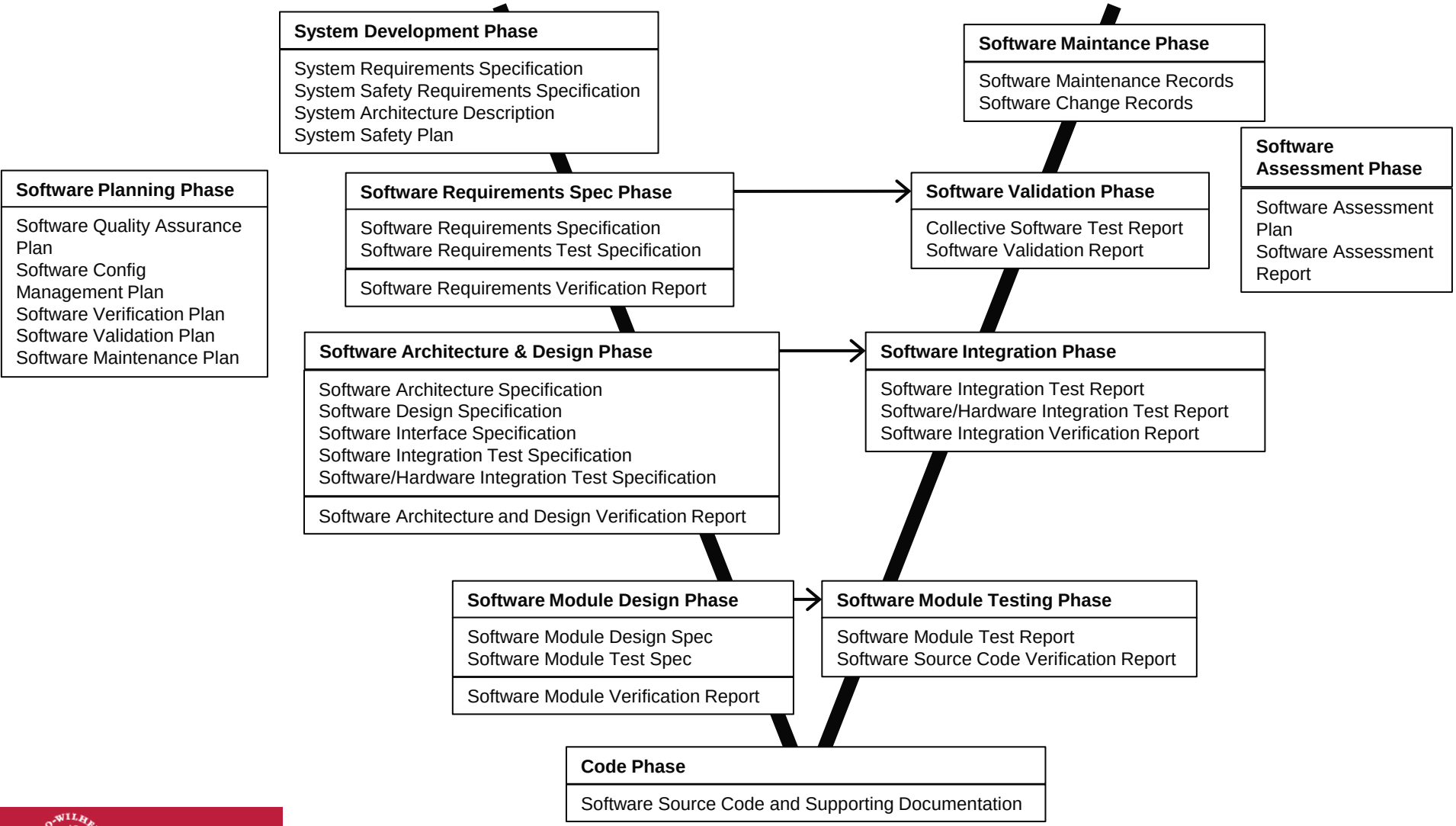
1. Introduction

System Lifecycle according to EN 50126



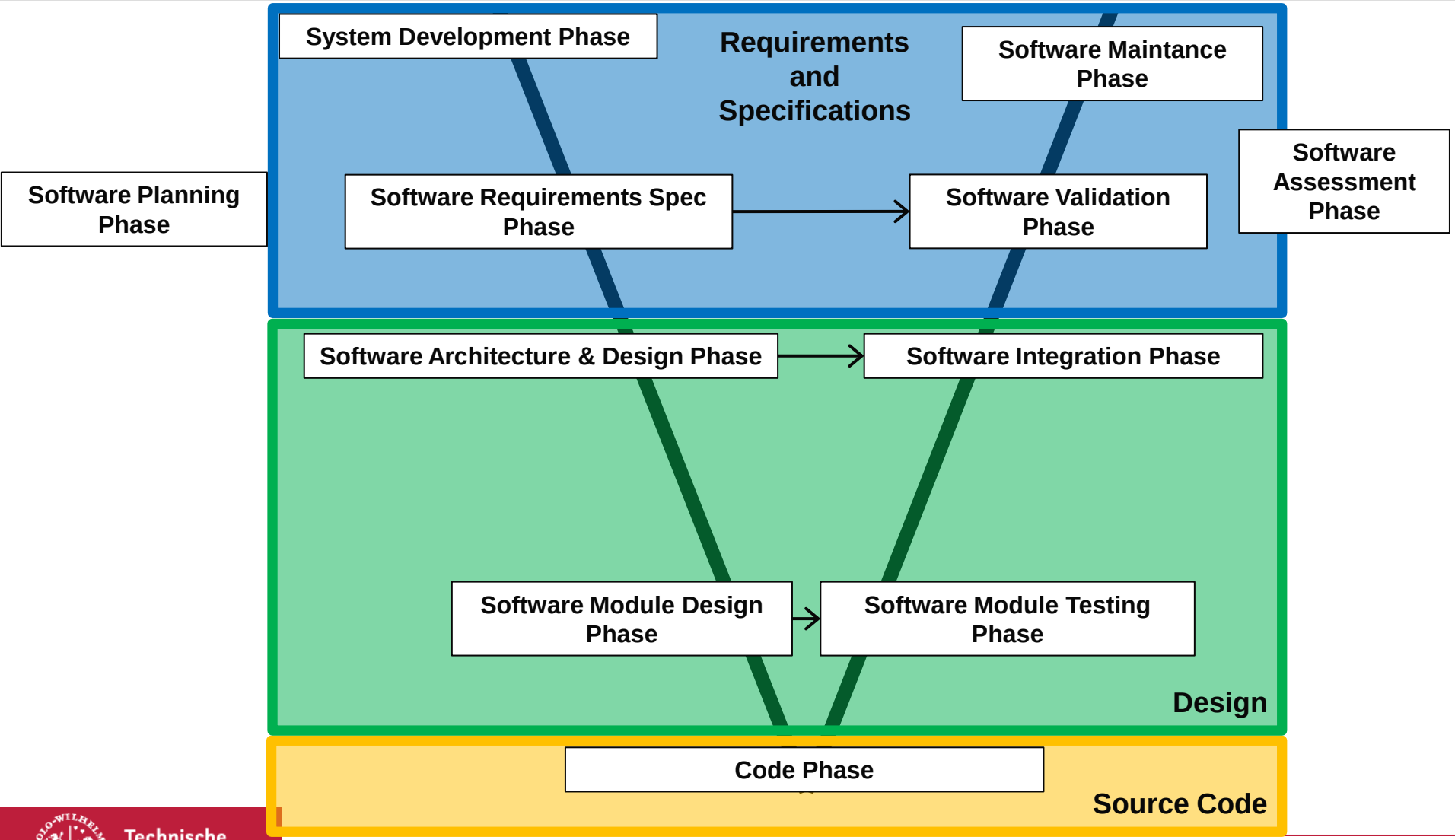
1. Introduction

Software Development Lifecycle according to EN 50128



1. Introduction

Software Development Lifecycle according to EN 50128



CENELEC standards: recommend semi-formal/ formal methods

Safe system (ERTMS) → requires formal proof → requires formal methods

Formal methods: generic used term for mathematical based procedures

Clear structure of used components is necessary

→ seperation according to BMW - Concept

- **B**eschreibungsmittel (ger.): Means of Description (eng.)
- **M**ethoden (ger.): Methodologies (eng.)
- **W**erkzeuge (ger.): Tools (eng.)

→necessary for handling development process in each step

All three components have to be “open” to allow open proofs.

2. Interviews
Structure of questions

Interviews with experts from the railway sector and other industries concerning their use of formal development processes and formal methods (for safety relevant and non-safety relevant software and hardware)

Categories of questions

A. Business and organisation of the company
B. Methodologies, means of description and tools for their software system development process
C. Methodologies, means of description and tools used for source code generation
D. Verification and validation processes for models and code
E. Experiences with different methodologies, means of description and tools
F. Requirements on future methodologies, means of description and tools

2. Interviews

Overview of conducted interviews

15 Interviews, with 21 experts conducted

- AEbt (Germany)
- Alstom (France)
- CERTIFER (Paris)
- DB Fernverkehr (Germany)
- DB Netz (Germany)
- ERTMS Solutions (Belgium)
- Formalmind (Germany)
- RATP (France)
- SNCF (France)
- Systerel (France)
- Siemens (France and Germany)
- SRE (Switzerland)

Enquired but not realised:

- Airbus (France)
- BMW (Germany)
- Bombardier (Germany)
- SBB (Switzerland)
- BAV (Switzerland)

3. Means of Description

Relevant means of description (properties based on VDI/VDE 3681:2005)

	Citeria									Level of abstraction			
	MoD/Technique	Formal basis	Representation	Description of structure	Description of behaviour	Explicit time representation	No expertise required	Level of standardization	Tool support	System development	Software Requirements Specification	Software architecture and design	Source code
ACSL (ANSI/ISO C Specification Language) and C	MoD	o	T	+	+	-	-	+	+		+	+	+
Ada and Spark	MoD	o	T	+	+	-	-	+	+			+	+
Alloy	MoD	+	M	+	+	-	-	-	+	o	+	+	
CNL (Controlled Natural Language)	MoD	+	T, M	-	o	-	o	-	o	o	+		
HOL (High Order Logic)	MoD	+	M	+	+	+	-	-	o	o	+		
Lustre/ Textual Scade	MoD	+	T	+	+	-	-	o	+	o	+	+	
OBJ	MoD	+	M	+	o	-							
Timed Petri Nets	MoD	+	M, G	+	+	+	-	+	o	+	+		
Process Calculi (CCS, CSP, LOTOS)	MoD	+	M	-	o								
State Machines	MoD	+	M, G	-	+	-	-	-	+	+	+		
TL (Temporal Logic)	MoD	+	M	-	+	-	-	-	o			+	
UML 2.0 (Unified Modelling Language) and SysML (System Modelling Language)	MoD	o	T, G	+	+	+	o	+	+	+	+	o	
VDM (Vienna Development Method)	Tech	+	M	+	o	-	-	+	+	o	+	+	
Z, B - Method and Event B	Tech	+	M, G	+	+	o	-	+	+	o	+	+	

„+“- fulfils criteria in full (can be used) , „o“ – fulfils criteria partially (can be used to some extent), „-“does not fulfils criteria (cannot be used)
T – textual, M - mathematical-symbolic, G - graphical

4. Methodologies

Mandatory Methodologies by EN 50128 for SIL3 and 4

Software Requirements Specification

- Natural language and
- Formal/ semi-formal description (if necessary)

Verification and testing

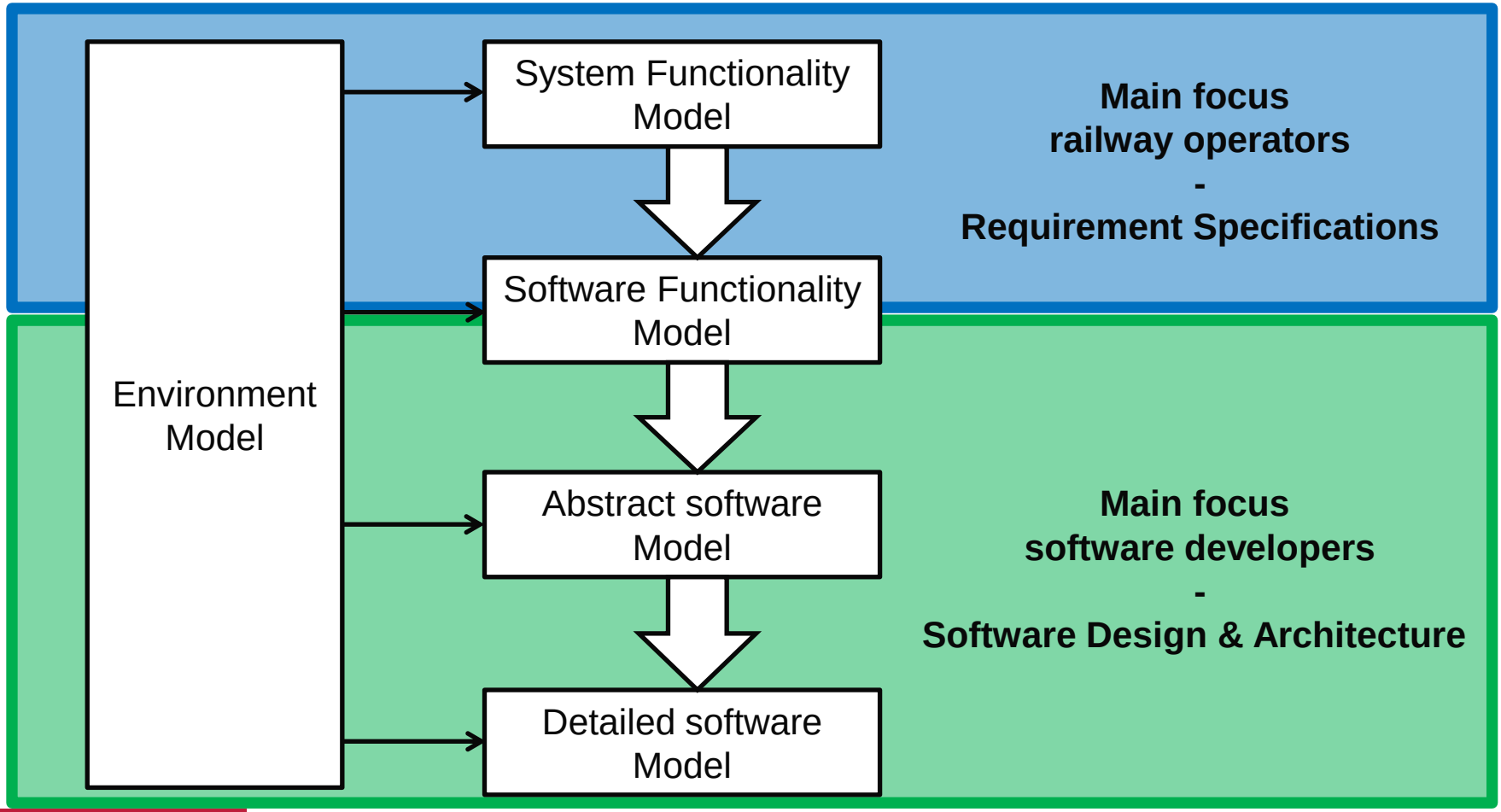
- Traceability for functional and non-functional Requirements
- Functional and Black Box tests
- Software performance tests

Source Code

- Coding standards have to be applied
- Compiler or translator have to be validated

4. Methodologies
Model-based Development

- Structured model-based approach is generally used



4. Methodologies

Methodologies for Verification and Validation

Verification

- Testing (including static and dynamic analysis)
  - Functional and Black Box Tests
  - Model-based Tests
- Formal Proofs

Formal proofs can substitute verification tests

Validation

- Testing
  - Test case and scenario catalogue
  - Testing against functional model
  - Simulation
- Formal Proof of functional requirements

Combination of Tests and Proof is needed for validation

4. Methodologies
Methodologies for Formal Proof

Properties of a system:

- **Structural properties**
- **Behavioural properties**

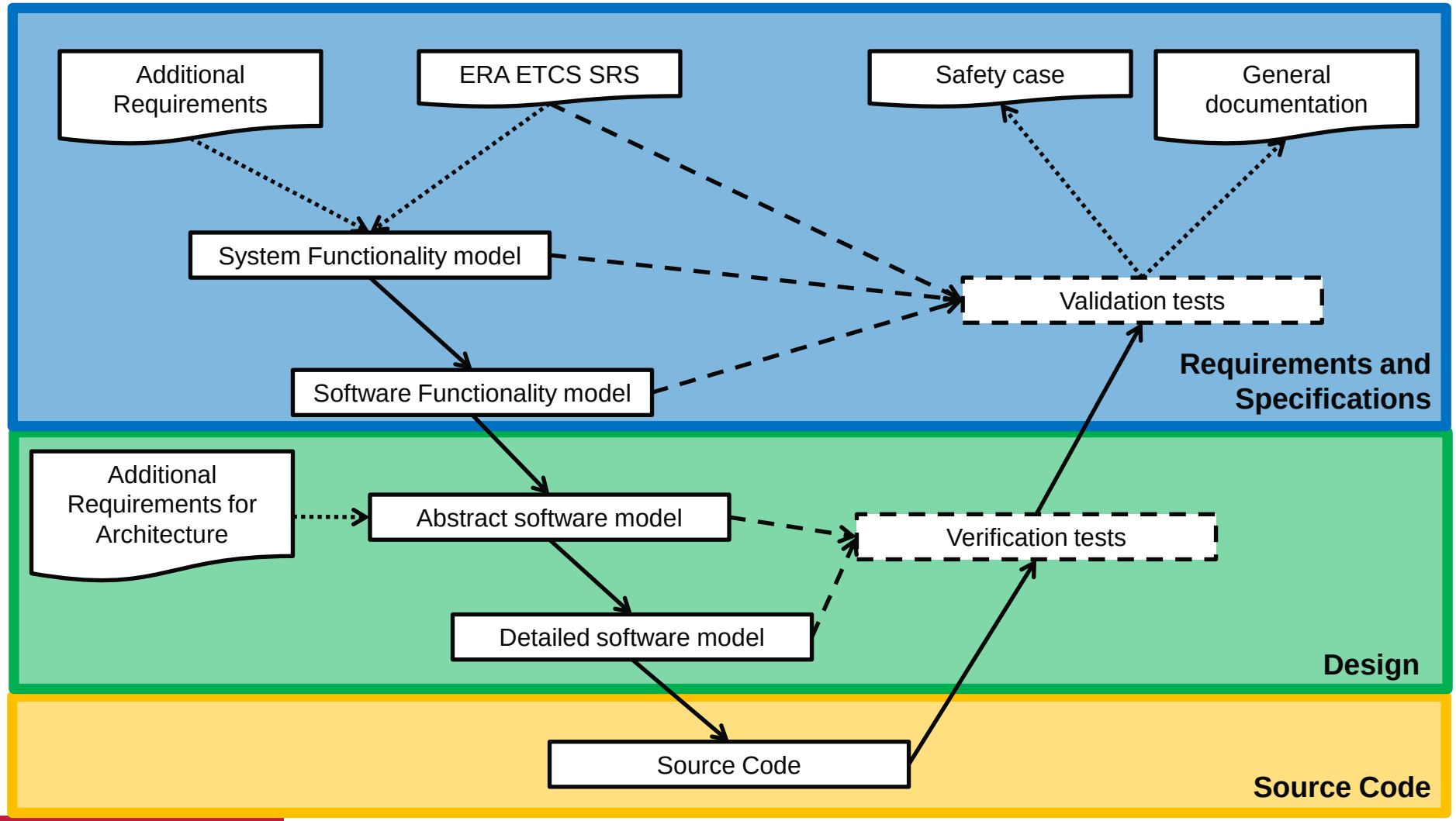
Two methodologies to provide formal proof

- **Theorem Proving**
  Verifying system‘s properties also for infinite systems
- **Model Checking**
  Verifying properties of finite state systems (negative results provide counterexample)

All relevant properties have to be identified and stated in a formal way.

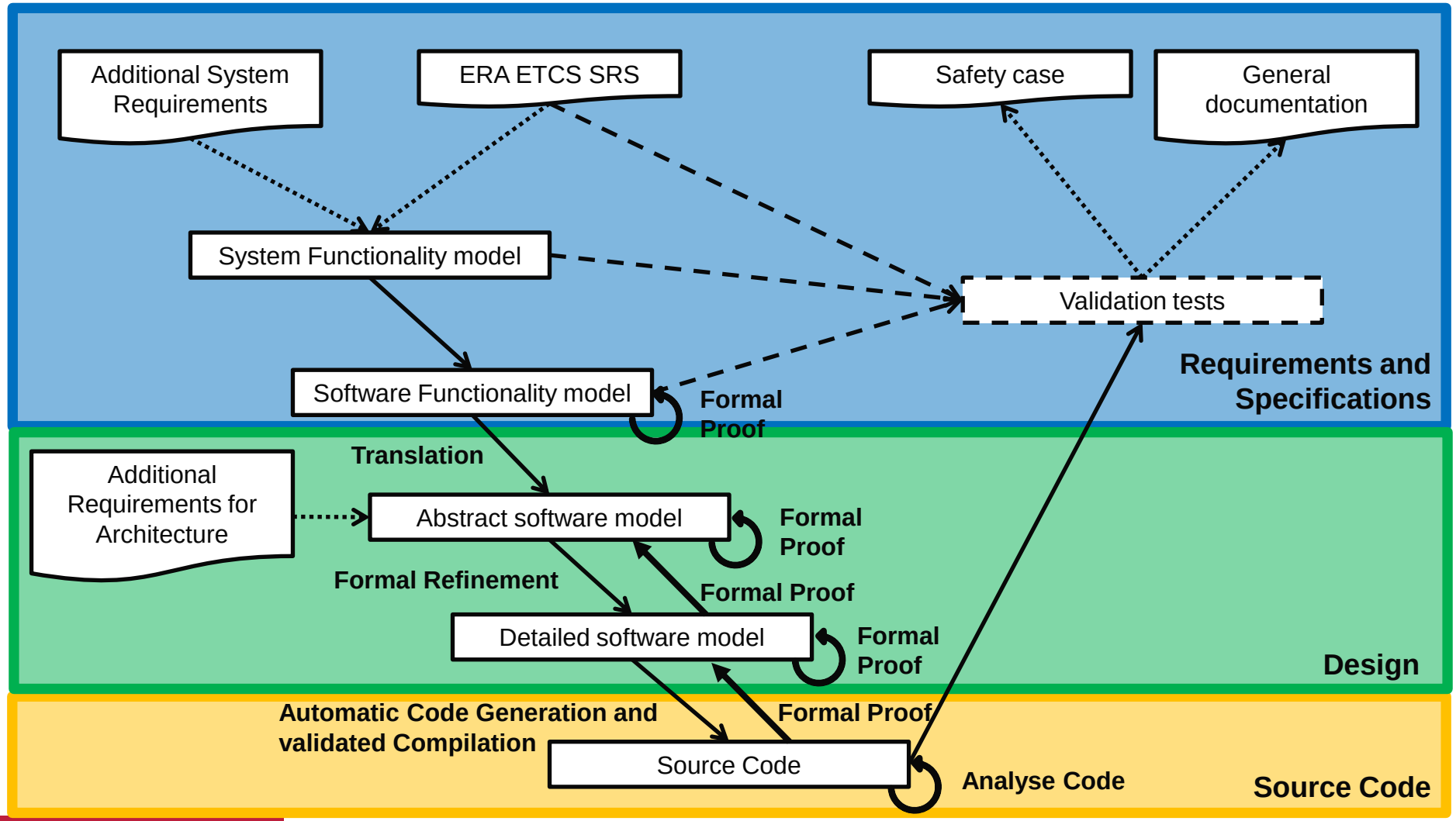
4. Methodologies

State of the Art Development Process – Test cases



4. Methodologies

State of the Art Development Process – Formal Proof



5. Tools

Tool support for different processes

Depending on the methodologies the tools have to support the following processes:

- Formalisation of textual requirements (i. a. ETCS SRS)
- Modelling
  - System/software structure
  - System/software behaviour
  - Software architecture & design
- Formal Refinement
- Model translation
- (Automatic) Code generation and compilation
- Formal Proof
  - Model checking (different levels)
  - Theorem proving
- Analysis of source code
- Testing
  - Test environment (for model and software tests)
  - Test case generation and test case database
- Simulation
- Traceability of requirements
- Versioning and configuration management
- Inteligent glossary
- Documentation

5. Tools

Overview tools (1)

Tool	Developer	License status	Means of Discription	Formalisation of textual reements	Modelling	Formal Refinement	Model translation	Code generation and compilation	Formal Proof	Analysis of source code	Testing	Simulation	Traceability of requirements	Versioning and configuration management	Inteligent glossary	Documentation
Alloy 4	Software Design Group at MIT	free	Alloy		+	+		+	+					+		o
Artisan Studio	Atego	commorical	SysML & UML 2.0	o	+		+	+					+	+	o	+
Atelier B	ClearSy System Engineering	free	B, Event B		+	+	+	+	+					+		o
CompoSys	ClearSy System Engineering	commorical	Event B	o	+				+		+	+		+	o	+
Control Build	Geensoft	commorical			+			+			+	+		+		o
CPN	Eindhoven University of Technology	open source	Petri Nets		+				+		+	+		+		o
Enterprise Architect	Sparx System	commorical	SysML & UML 2.0		+			+			+	+	+	+		+
ERTMS FormalSpecs	ERTMS Solutions	open source	DSL		+		+	+			+	+	+	+		o
Frama-C	CEA–LIST and INRIA-Saclay	open source	C						+	+						o
iglos	TU Braunschweig, iVA	free/commerial		+									+	+	+	+
iLock/iCertifier	Prover Technology AB	commorical			+		+	+	+	+	+	+		+		+
KNOW Enterprise	KnowGravity Inc.	commorical	UML 2.0 (xUML)	o	+		+	+			+	+	+	+	o	+
MagicDraw	No Magic	commorical	SysML & UML 2.0	o	+			+			+	+	+	+		o
mCRL2	Technische Universiteit Eindhoven	free	mCRL2		+	+			+					+		o
Modelio	Modeliosoft	open source	SysML & UML 2.0	o	+			+			o	o	+	+		o
NuSMV	Fondazione Bruno Kessler	open source	BDD and SAT						+							o

„+“- can be used , „o“ – can be used to some extent

5. Tools

Overview tools (2)

Tool	Developer	License status	Means of Discription	Formalisation of textual reements	Modelling	Formal Refinement	Model translation	Code generation and compilation	Formal Proof	Analysis of source code	Testing	Simulation	Traceability of requirements	Versioning and configuration management	Inteligent glossary	Documentation
Papyrus	CEA	open source	SysML & UML 2.0	+	+						+	+	+	+		o
Perfect Developer	Escher Technologies Ltd.	commorical			+			+	+		+			+		o
PRISM	University of Oxford	open source	BDD and MTBDD						+							o
ProB	Formal Mind GmbH	open source	B						+		+	+		+		o
ProR	Formal Mind GmbH	open source	ReqIF 1.0.1	+										+		+
Rational Software	IBM	commercial	SysML & UML 2.0	+	+		+	+			+	+	+	+		+
SCADE/ Suite Design Verifier	Esterel Technologies S.A.	commercial	Lustre/ State machines		+	+	+	+	+		+	+		+		o
Simulink/ Design Verifier	MathWorks	commercial			+	+	+	+	o		+	+		+		o
SPARK GPL/ GNATprove	AdaCore	open source	Spark ADA		+			+	+	+	+			+		o
SPIN	Bell Labs	open source	PROMELA						+	+	+	+				o
π-Tool	IQST	commercial	Petri Nets		+	+	+		+		+	+		+		o

„+“- can be used , „o“ – can be used to some extent

- A combination of different tools is usually used
- Data exchange between tools has to be coordinated (software platform)

Means of Description

- Well documented (better standardised)
- Understandable for domain experts (communication with modelling experts)
- Capturing structure and behaviour of all ETCS specifications
- Support formal proof
- Support graphical modelling
- Support model execution (especially for high level models)
- Support specification of test cases

Methodologies

- Conform to CENELEC standards
- Easy to apply in practice
- Clear traceability of changes and their influences
- Provide mainly automatic theorem proving and model checking

Tools

- Robust and validated
- Downwards compatible
- Independent of operating system
- Good performance on standard systems
- High level support and easy maintenance
- Good usability (visualisation)
- Open interfaces for data (import and export data)
- Integration in tool chain (data exchange and visual interface)
- Allow multi user work (avoidance of conflicts)
- Compatible with existing configuration management systems
- Allow automatic generation of textual documents from models
- Automatic link generation between glossary and models

7. Summary

Means of Description

- Various means of description are used to capture different abstraction levels and system properties

  → probably collection of means of description need to handle all development processes

Methodology

- Formal proof for software common practice, but not functional system requirements

  → continuous formal modelling is needed

Tools

- Various tools are available, but the common one are mostly commercial

  → platform is needed to integrate tools for different processes

Technische Universität Braunschweig

Institut für Verkehrssicherheit und Automatisierungstechnik

Prof. Dr.-Ing. Dr. h.c. mult. E. Schnieder

Questions and Discussion

Work-Package 2: "Requirements for Open Proofs“
Task 2.1.1 – State of the Art