fortiss

Integrated Systems and Safety Engineering

Towards Meaningful Assurance Cases

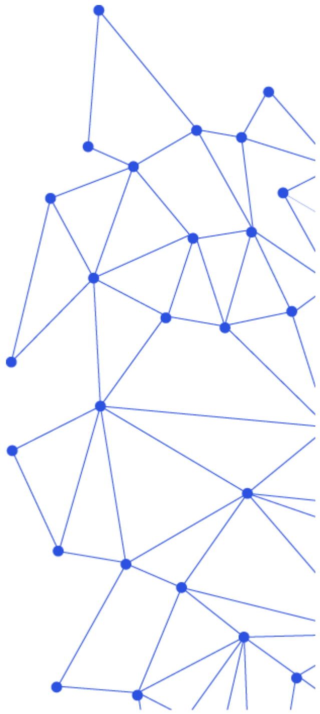
Carmen Cârlan

Harald Ruess

Sebastian Voss

Supported by D-MILS (d-mils.org)

fortiss GmbH
An-Institut Technische Universität München



Assurance Cases

State-of-the-Practice (I)

Implicit assurance/safety cases are mainly supported by standard-mandated evidence (i.e. IEC 61508, ISO26262, DO178C)

- **Checkmark-based approach to safety engineering is encouraged**, since the role/purpose of standard-mandated evidence often remains unclear

  Example: Section B.30 of the IEC 61508 recommends the use of „formal methods for example CCS, CSP, HOL, LOTOS, OBJ, VDM, Z, B“ for SIL2 and beyond; and highly recommended for SIL4). These phrases were copied into the tender document for a drive-by-wire development, and relegated to a TIER2 supplier of a wheel angle sensor

- **Tailoring according to the specific safety-needs of the product difficult:** unclear how to be best use avilable resources for increased assurance; also considerable impact on development costs
- **Not all design decisions necessarily explicated**, as current certification regimes focus on traceability

Assurance Cases

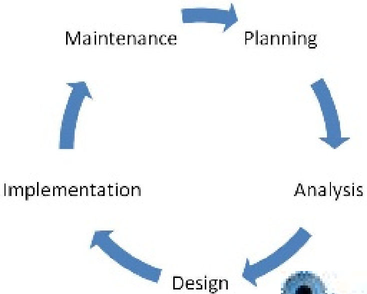
State-of-the-Practice (II)

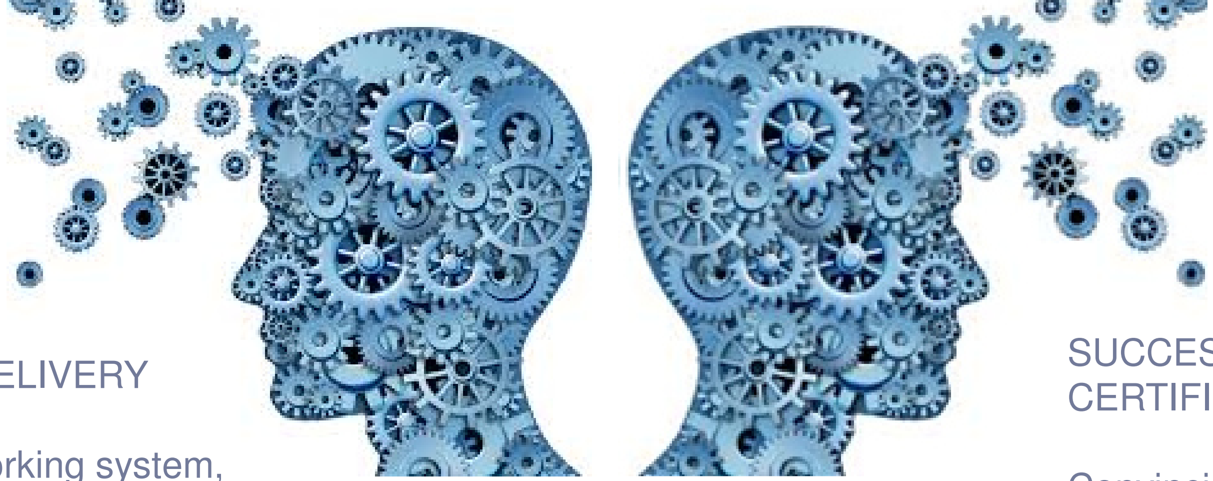
Explicit assurance cases (goals, arguments, evidence) not state-of-the-practice for developing safety-critical systems

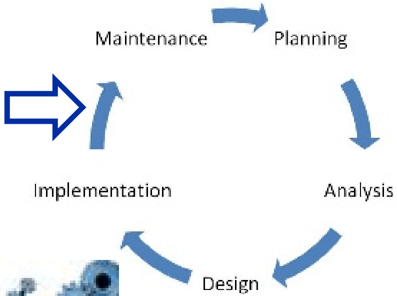
- Assurance cases with the purpose of certification, but **not well-integrated into product design** and development
- Sometimes considered to be an **extra document,** if not extraneous from the point-of-view of the design team and the safety team.
- What other uses are there for an assurance case?

Assurance Cases

State-of-the-Practice (III)







SUCCESFUL DELIVERY

Developing a working system, which complies with the client‘s requirements

Systems & Software Engineers

Safety Engineers

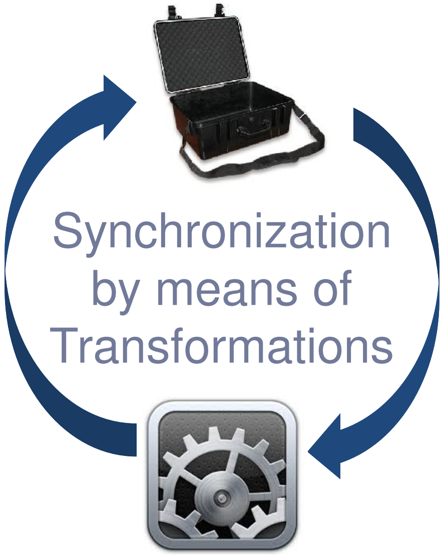
SUCCESFUL CERTIFICATION

Convincing regulators that the system is safe in the given context

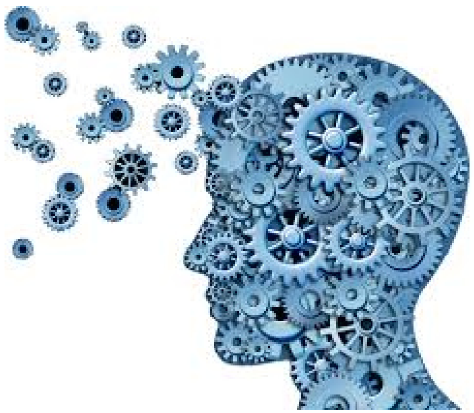
Our Approach

Integrated Model-Based Development of Product and Assurance Case

Systems & Software Engineers



Synchronization by means of Transformations



Safety engineers

I. ***Model-based development approach*** with integrating views for a *modular construction systems;*

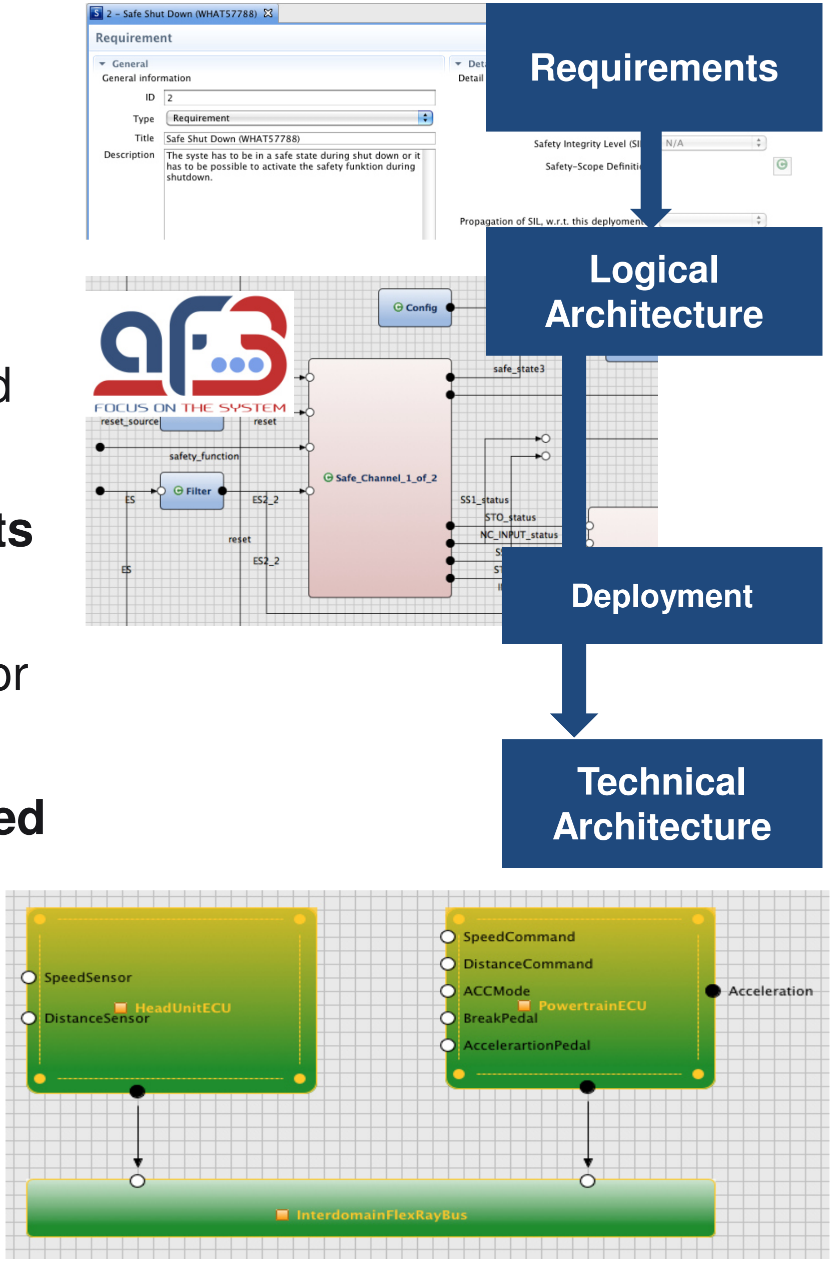
II. ***Modular construction and argumentation principles*** within these views, based on safety standards;

III. H***igh-level design decisions*** *and their* ***documentation*** by means of ***safety case patterns.***

Model-based Development

AF3 Framework

- □ Supports concept phase and product development at **system**, **hardware** and **software** Level
- □ Explicates **allocations** and **refinements** between different abstractions
- □ Provides modular, hierarchic concept for **networks of components**
- □ Can be **simulated and formally verified**
- □ Supports **automated verification** (e.g., contracts)
- □ Supports **automated generation** (e.g., test cases, code, platform configurations, schedules)



2 - Safe Shut Down (WHAT57788)

Requirement

General

General information

ID: 2

Type: Requirement

Title: Safe Shut Down (WHAT57788)

Description: The syste has to be in a safe state during shut down or it has to be possible to activate the safety funktion during shutdown.

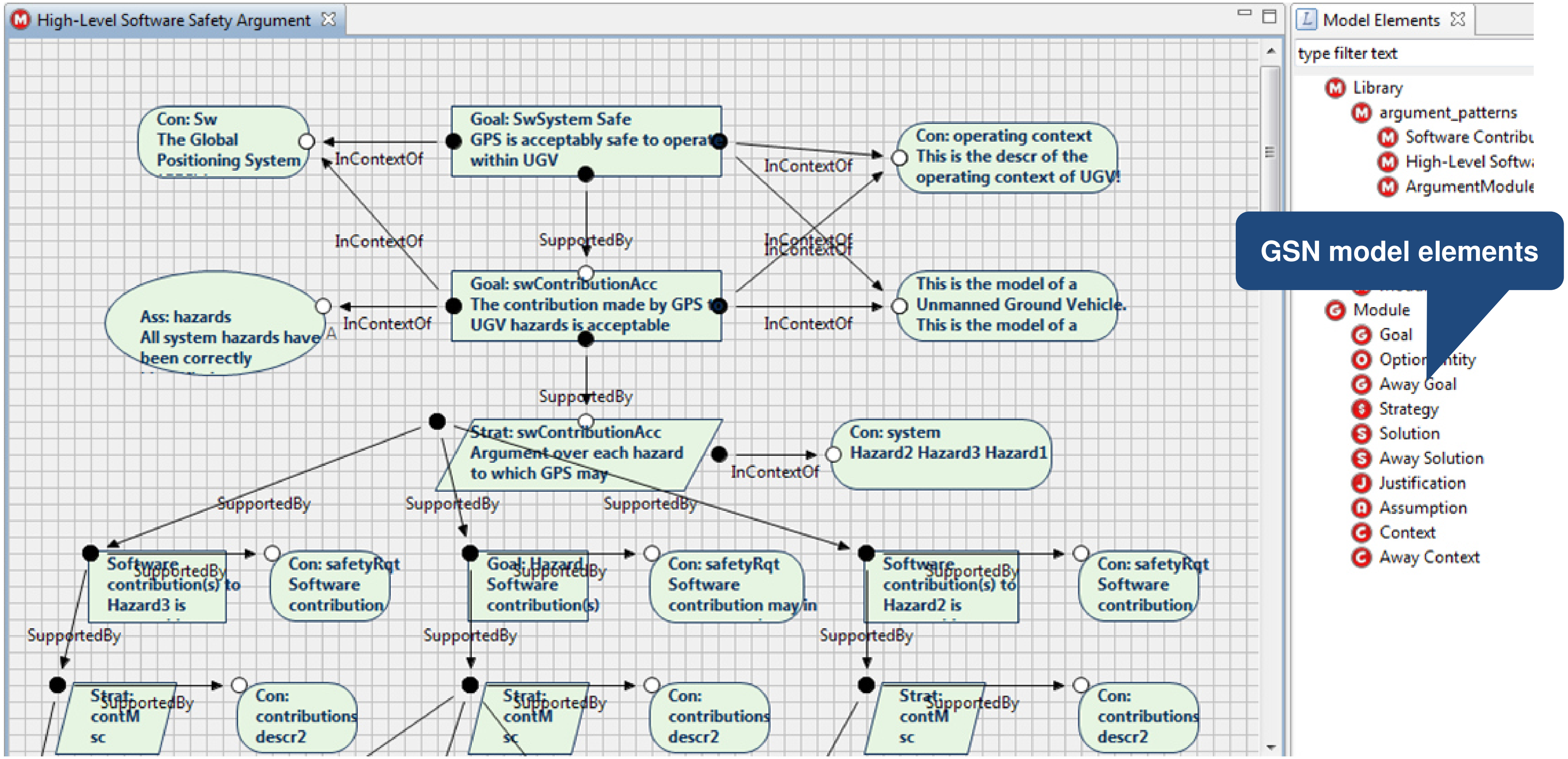
Safety Integrity Level (SI: N/A

Safety-Scope Definiti

Propagation of SIL, w.r.t. this deplyoment

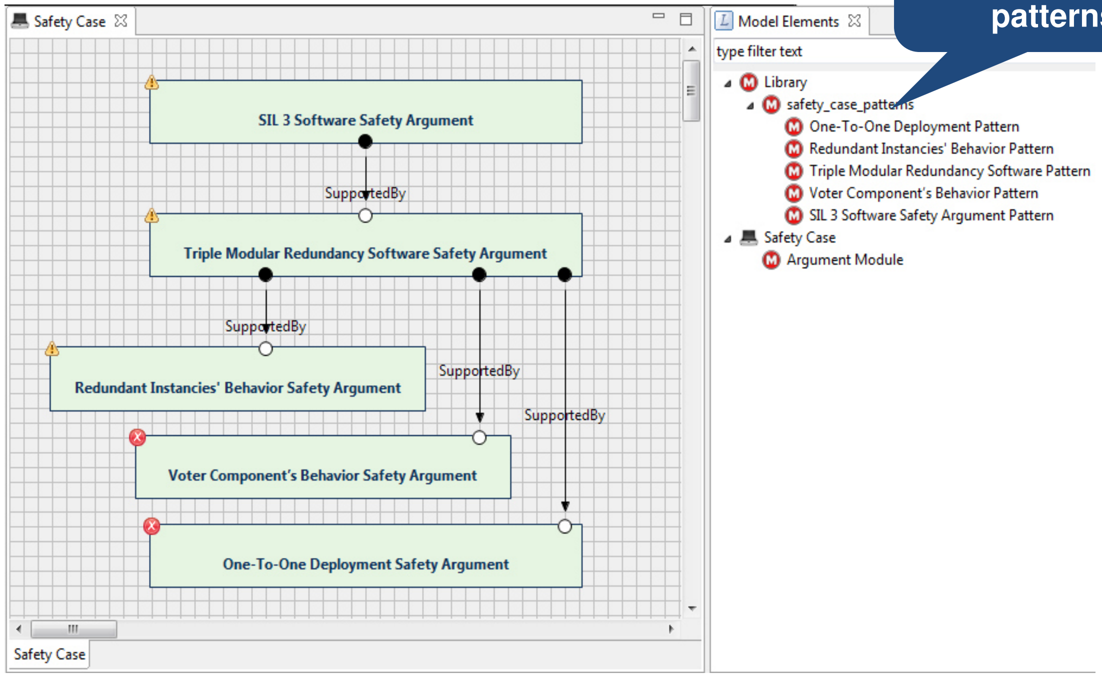
GSN-based Assurance Cases in AF3

The Argument Structure View



GSN model elements

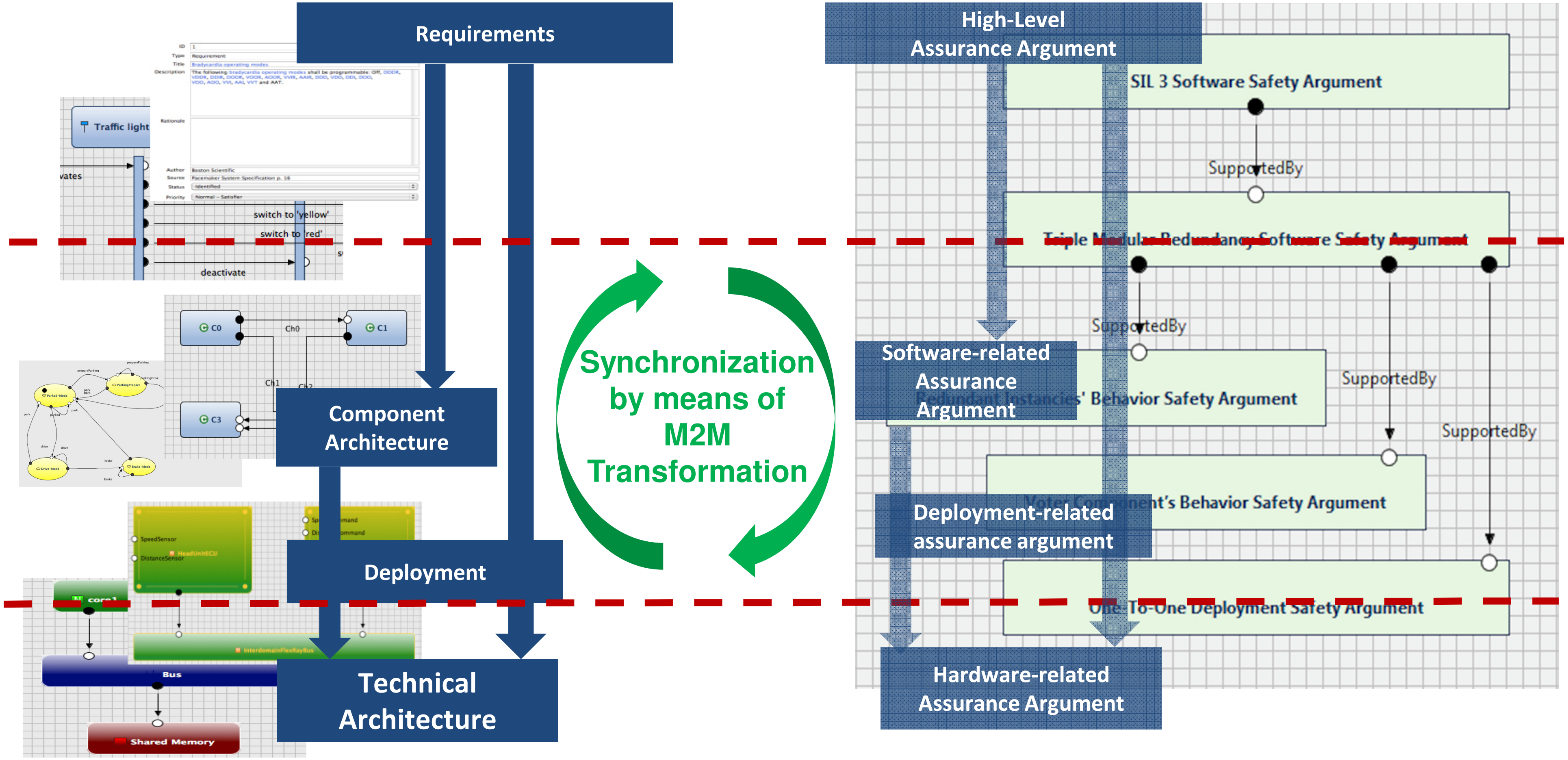
Modular Assurance Case Patterns


Safety Case
SIL 3 Software Safety Argument
SupportedBy
Triple Modular Redundancy Software Safety Argument
SupportedBy
Redundant Instancies' Behavior Safety Argument
SupportedBy
SupportedBy
Voter Component's Behavior Safety Argument
One-To-One Deployment Safety Argument
Safety Case
Model Elements
type filter text
Library
safety_case_patterns
One-To-One Deployment Pattern
Redundant Instancies' Behavior Pattern
Triple Modular Redundancy Software Pattern
Voter Component's Behavior Pattern
SIL 3 Software Safety Argument Pattern
Safety Case
Argument Module

Library of Assurance case patterns

- Pattern instantiation provides references in assurance cases to corresponding system artefacts
- … as the basis for integrated views for the design of a system and the argumentation about its functional safety

Integrated Development of System and Assurance Case



ID: 1
Type: Requirement
Title: Bradycardia operating modes
Description: The following bradycardia operating modes shall be programmable: Off, DDDR, VDDR, DDIR, DOOR, VOOR, AOOR, VVIR, AAIR, DDD, VDD, DDI, DOO, VOO, AOO, VVI, AAI, VVT and AAT.
Rationale:
Author: Boston Scientific
Source: Pacemaker System Specification p. 16
Status: Identified
Priority: Normal – Satisfier

Requirements
Component Architecture
Deployment
Technical Architecture
Synchronization by means of M2M Transformation
High-Level Assurance Argument
Software-related Assurance Argument
Deployment-related assurance argument
Hardware-related Assurance Argument
SIL 3 Software Safety Argument
SupportedBy
Triple Modular Redundancy Software Safety Argument
SupportedBy
Redundant Instancies' Behavior Safety Argument
SupportedBy
SupportedBy
Voter Component's Behavior Safety Argument
One-To-One Deployment Safety Argument
Traffic light
vates
switch to 'yellow'
switch to 'red'
deactivate
C0
C1
C3
Ch0
Ch1
Ch2
SpeedSensor
DistanceSensor
HeadUnitECU
core1
InterdomainFlexRayBus
Bus
Shared Memory

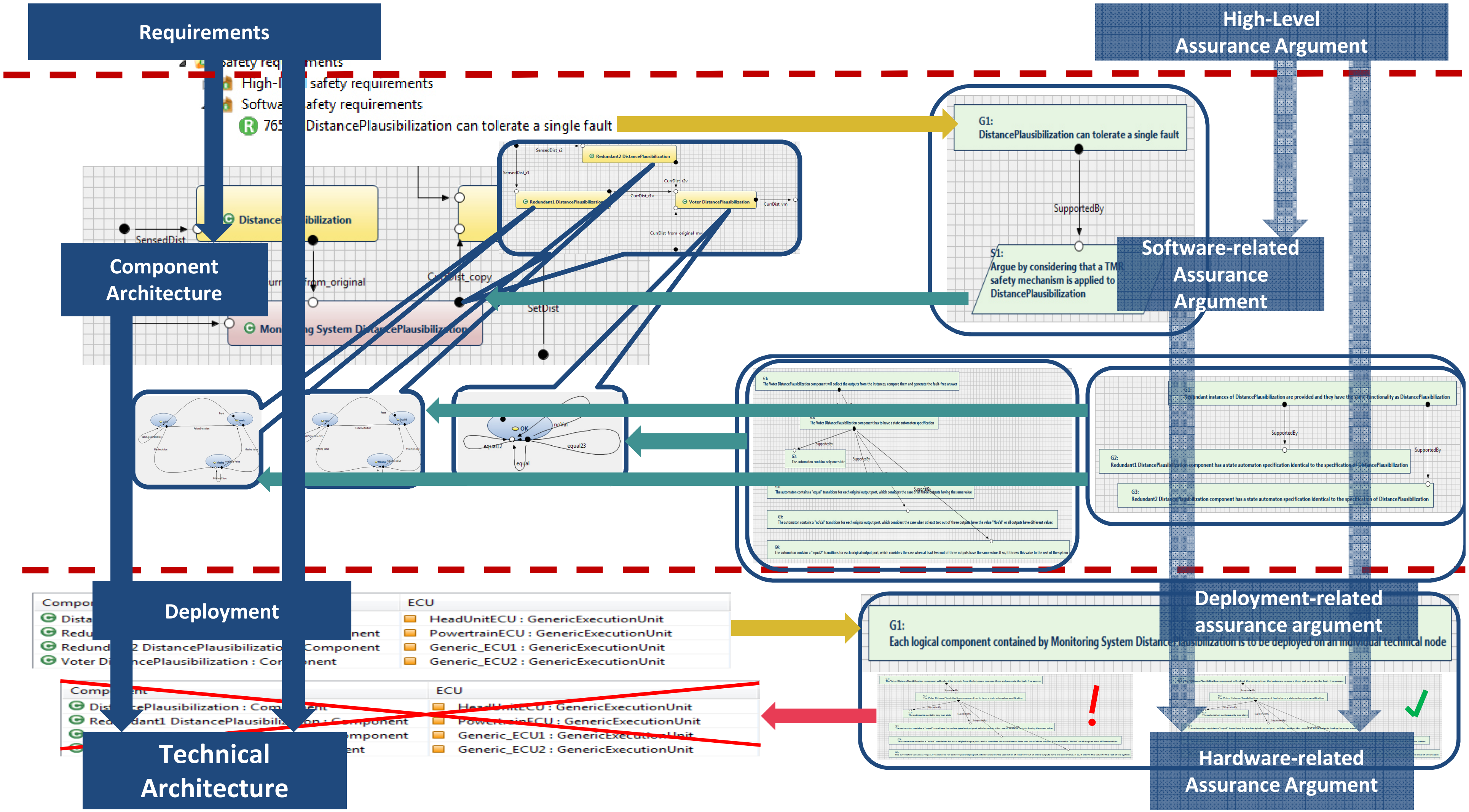
System Design Artefacts

Modular System Safety Case

[VCST15] S. Voss, C. Cârlan, B. Schätz, T. Kelly, **Safety Case Driven Model-Based Systems Construction, EITEC, CPS Week, April 2015, Seattle.**

Example 1

Deciding on Appropriate Architectural Design


Requirements
Safety requirements
High-level safety requirements
Software safety requirements
765 DistancePlausibilization can tolerate a single fault
Component Architecture
DistancePlausibilization
SensedDist
CurrDist_from_original
CurrDist_copy
SetDist
Monitoring System DistancePlausibilization
SensedDist_r2
SensedDist_r1
Redundant2 DistancePlausibilization
Redundant1 DistancePlausibilization
Voter DistancePlausibilization
CurrDist_r2v
CurrDist_r1v
CurrDist_vm
CurrDist_from_original_mv
OK
noVal
equal12
equal23
equal
Deployment
Technical Architecture
High-Level Assurance Argument
Software-related Assurance Argument
Deployment-related assurance argument
Hardware-related Assurance Argument
SupportedBy
!
✓

G1:
DistancePlausibilization can tolerate a single fault

S1:
Argue by considering that a TMR safety mechanism is applied to DistancePlausibilization

G1:
The Voter DistancePlausibilization component will collect the outputs from the instances, compare them and generate the fault-free answer

G2:
The Voter DistancePlausibilization component has to have a state automaton specification

G3:
The automaton contains only one state

G4:
The automaton contains a "equal" transitions for each original output port, which considers the case of all three outputs having the same value

G5:
The automaton contains a "noVal" transitions for each original output port, which considers the case when at least two out of three outputs have the value "NoVal" or all outputs have different values

G6:
The automaton contains a "equal2" transitions for each original output port, which considers the case when at least two out of three outputs have the same value. If so, it throws this value to the rest of the system

G1:
Redundant instances of DistancePlausibilization are provided and they have the same functionality as DistancePlausibilization

G2:
Redundant1 DistancePlausibilization component has a state automaton specification identical to the specification of DistancePlausibilization

G3:
Redundant2 DistancePlausibilization component has a state automaton specification identical to the specification of DistancePlausibilization

Compo	ECU
Dista	HeadUnitECU : GenericExecutionUnit
Redu nent	PowertrainECU : GenericExecutionUnit
Redund 2 DistancePlausibilizatio Component	Generic_ECU1 : GenericExecutionUnit
Voter Di ncePlausibilization : Co nent	Generic_ECU2 : GenericExecutionUnit

Comp nt	ECU
~~Dis cePlausibilization : Com ent~~	~~HeadUnitECU : GenericExecutionUnit~~
~~Re dant1 DistancePlausibili on : Component~~	~~PowertrainECU : GenericExecutionUnit~~
~~mponent~~	~~Generic_ECU1 : GenericExecutionUnit~~
~~ent~~	~~Generic_ECU2 : GenericExecutionUnit~~

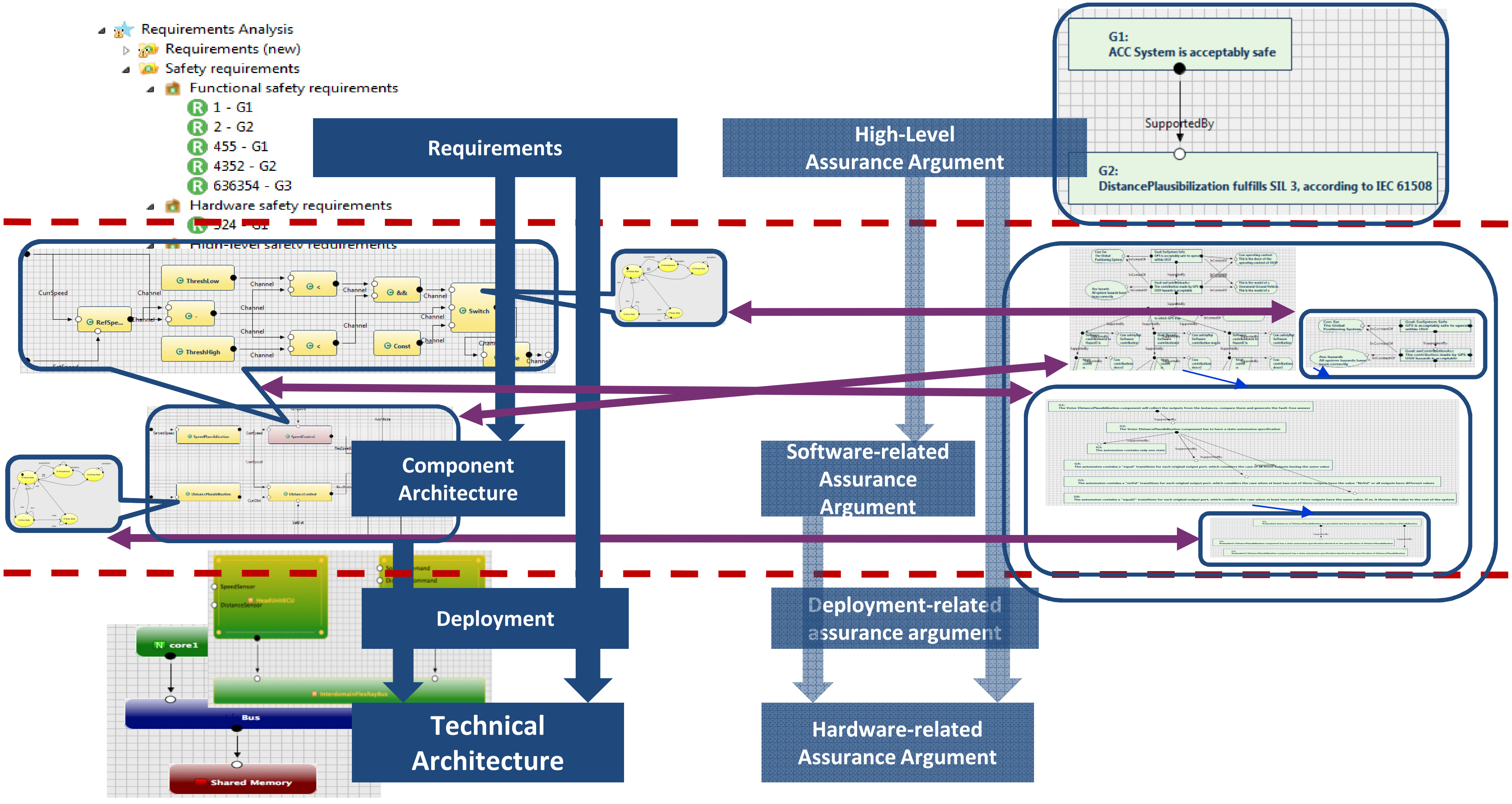
G1:
Each logical component contained by Monitoring System DistancePlausibilization is to be deployed on an individual technical node

System Design Artefacts

Modular System Assurance Case

Example 2

Architectural Refinement by Means of TMR Transformation



Requirements Analysis
- Requirements (new)
- Safety requirements
  - Functional safety requirements
    - R 1 - G1
    - R 2 - G2
    - R 455 - G1
    - R 4352 - G2
    - R 636354 - G3
  - Hardware safety requirements
    - R 324 - G1
  - High-level safety requirements

G1:
ACC System is acceptably safe

SupportedBy

G2:
DistancePlausibilization fulfills SIL 3, according to IEC 61508

Requirements

High-Level Assurance Argument

Component Architecture

Software-related Assurance Argument

Deployment

Deployment-related assurance argument

Technical Architecture

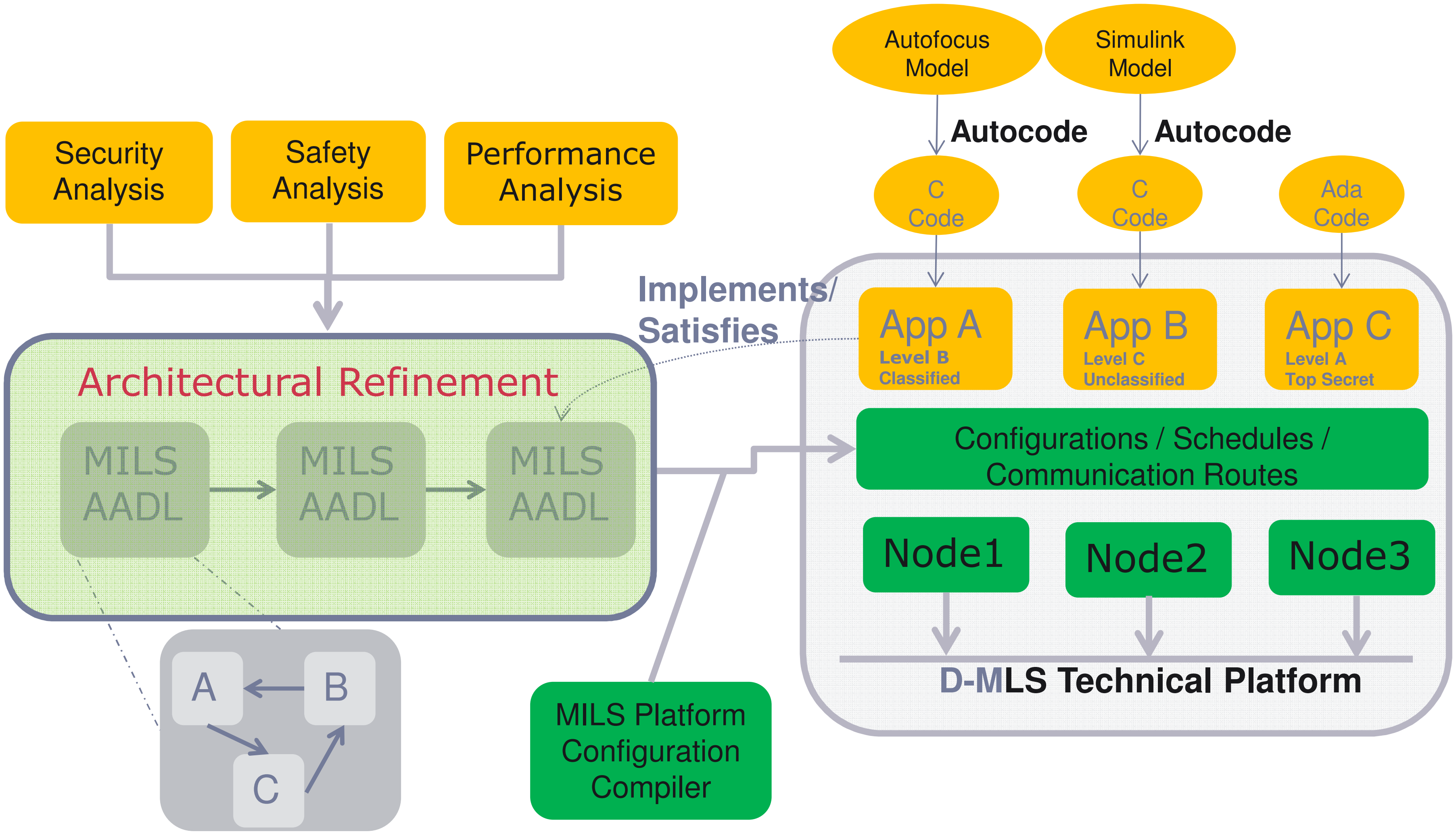
Hardware-related Assurance Argument

System Design Artifacts

Modular System Safety Case

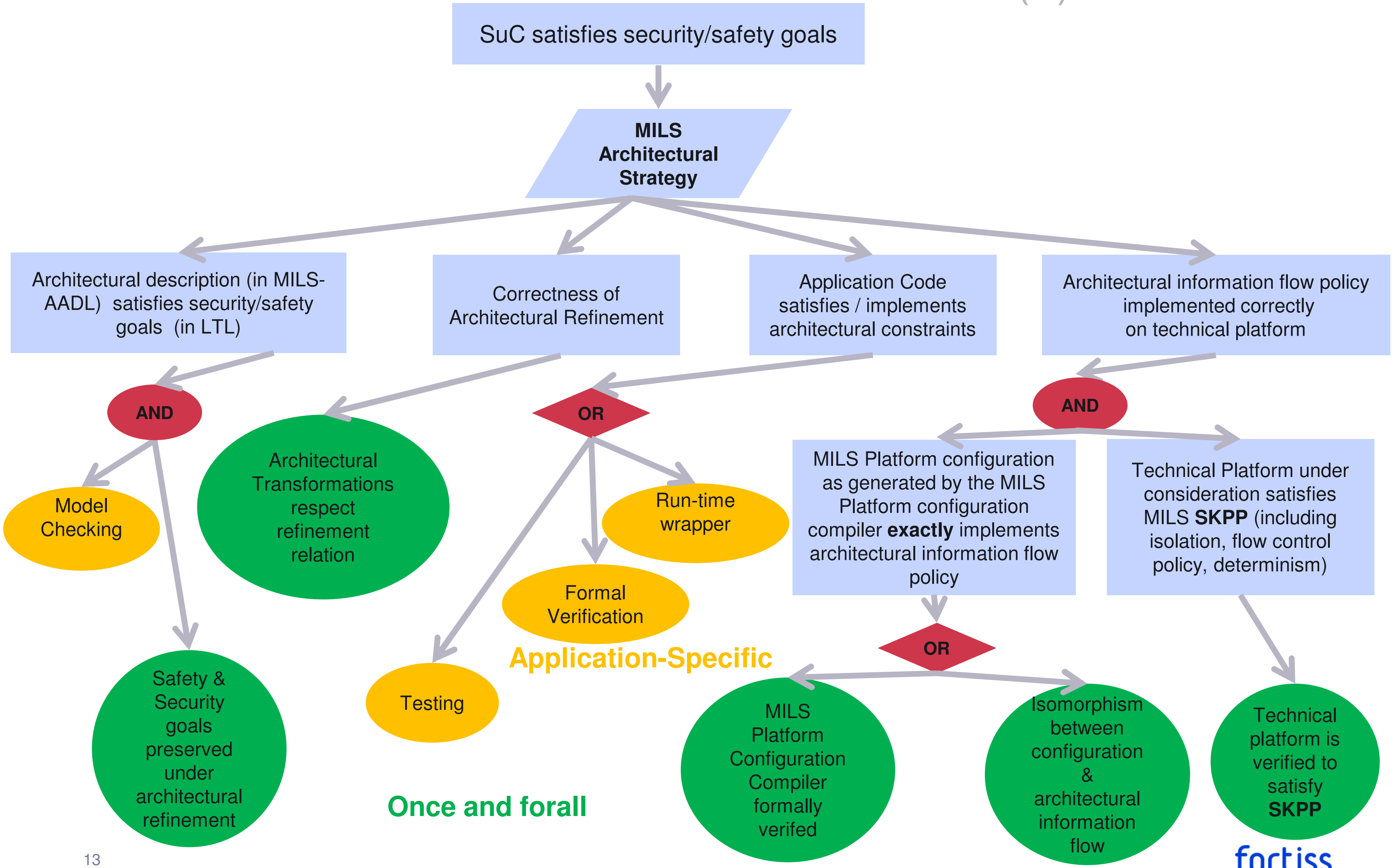
Example 3

MILS Architectural Assurance Case Pattern



Example 3

MILS Architectural Assurance Case Pattern (II)



SuC satisfies security/safety goals

MILS Architectural Strategy

Architectural description (in MILS-AADL) satisfies security/safety goals (in LTL)

Correctness of Architectural Refinement

Application Code satisfies / implements architectural constraints

Architectural information flow policy implemented correctly on technical platform

AND
Model Checking
Safety & Security goals preserved under architectural refinement
Architectural Transformations respect refinement relation
OR
Testing
Formal Verification
Run-time wrapper
AND

MILS Platform configuration as generated by the MILS Platform configuration compiler **exactly** implements architectural information flow policy

Technical Platform under consideration satisfies MILS **SKPP** (including isolation, flow control policy, determinism)

OR
MILS Platform Configuration Compiler formally verifed
Isomorphism between configuration & architectural information flow
Technical platform is verified to satisfy **SKPP**

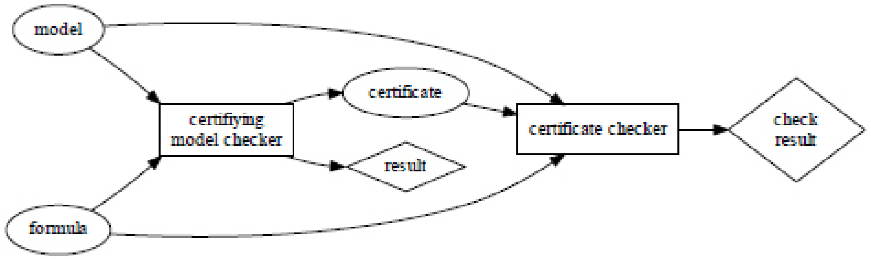
Application-Specific

Once and forall

Example 4

Certifying Model Checker For Building Assurance Cases

- **Model Checkers** (MC) usually only output counterexamples on failed proof attempts.
- Counterexamples have been used to construct FTA and FMEA in an automated fashion.
- **Certifying MC** produce **independably checkable certificate**



- **Certifying MC for mu-calculus** (including CTL, CTL*, LTL,…) with **winning strategies** for corresponding games **as certificates** [HNR15]
  - Certificates may be computed for both safety and liveness properties from MC
  - Winning strategies are checkable in low polynomial time
  - Winning strategies may be used to **scrutinise** safety arguments a la **interactive proofs**
  - **Challenger** suggests a move, to which **Prover** responds with a move according to strategy, and so on, …

[HNR15] M. Hofmann, C. Neukirchen, H. Rueß, **Certification for mu-calculus with winning strategies**, submitted to **ICTAC 2015.**

Integrated System and Assurance Case Development

Potential Benefits

- Assurance cases decompose along **vertical and horizontal structure** of system design artefacts
- Assurance case may **guide safe and efficient system development**
- Architecture-centric approach provides opportunity for **high-level assurance patterns** (e.g. MILS) for reducing the effort of building up safety cases
- Certifying model checkers for automatically generating **formally checkable evidence** in **assurance case**s
- Assurance case may extend, and even replace, the traditional syntactic tracing *(„depends-on")* with a **semantic tracing** *(„why?")* capability
- System may **safely (self-) evolve**/**adapt** within the limits of the capability of adapting corresponding safety case(s)

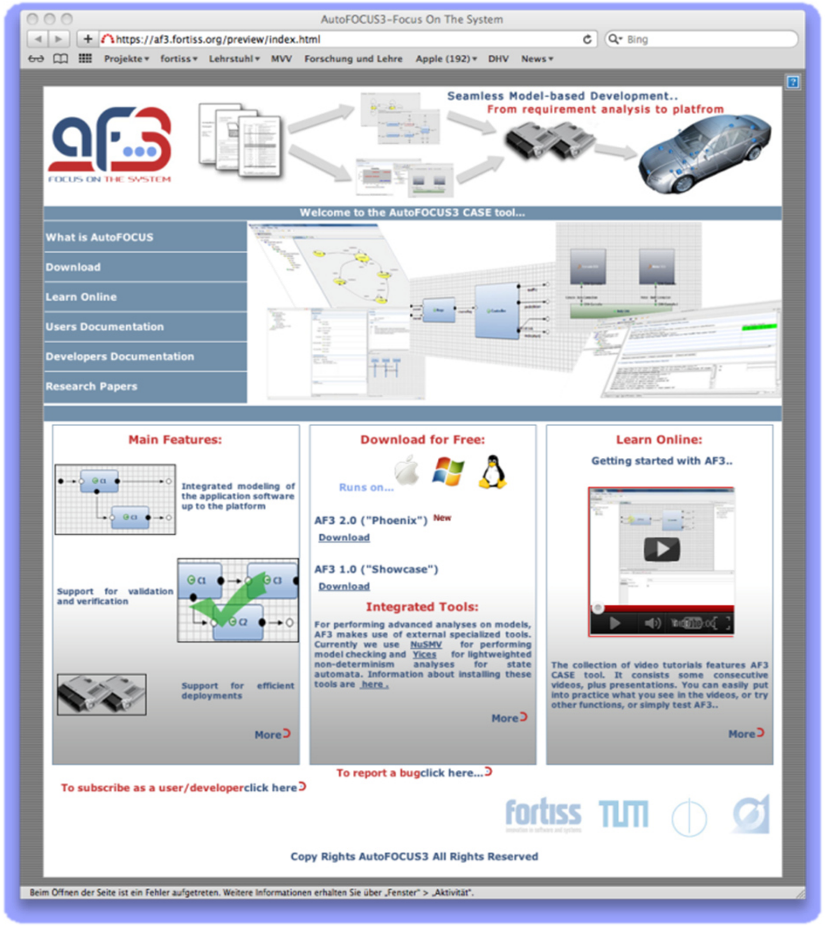
Conclusions

Presented first steps towards realizing **integrated system development and its corresponding safety case** in the AF3 model-based framework

- Approach needs to be formalized with the goal of having M2M **transformations and also deployment formally verified** (e.g. PVS)
- More complete **catalogue of transformations** (e.g. architectural refinement by means of fault-tolerance patterns) needed
- Refine **MILS architecture-specific assurance case patterns** and implement as transforimation in AF3
- Approach needs to be validated by means of **realistic case studies**

AF3 – Try it out!

Eclipse Public License



Seamless Model-based Development..
From requirement analysis to platfrom

Welcome to the AutoFOCUS3 CASE tool...

Main Features:

Integrated modeling of the application software up to the platform

Support for validation and verification

Support for efficient deployments

More

Download for Free:

Runs on...

AF3 2.0 ("Phoenix") New
Download

AF3 1.0 ("Showcase")
Download

Integrated Tools:

For performing advanced analyses on models, AF3 makes use of external specialized tools. Currently we use NuSMV for performing model checking and Yices for lightweighted non-determinism analyses for state automata. Information about installing these tools are here .

More

Learn Online:

Getting started with AF3..

The collection of video tutorials features AF3 CASE tool. It consists some consecutive videos, plus presentations. You can easily put into practice what you see in the videos, or try other functions, or simply test AF3..

More

To report a bugclick here...

To subscribe as a user/developerclick here

Copy Rights AutoFOCUS3 All Rights Reserved

Beim Öffnen der Seite ist ein Fehler aufgetreten. Weitere Informationen erhalten Sie über „Fenster" > „Aktivität".

af3.fortiss.org

„… how much better will it be to bring under mathematical laws human reasoning, which is the most excellent and useful thing we have“. (Leibniz)

Harald Ruess

ruess@fortiss.org

fortiss GmbH
An-Institut Technische Universität München
Guerickestraße 25 · 80805 München · Germany

tel +49 89 3603522 33 **fax** +49 89 3603522 50

www.fortiss.org