An Introduction to ALISA and an Experience Report on its Usage for an Industrial Railway System Case Study

Dominique Blouin[1], Paolo Crisafulli[2], Françoise Caron[3] and Cristian Maxime[2]
[1]LTCI Lab, Telecom Paris, Institut Polytechnique de Paris, France
[2]IRT SystemX, France
[3]Eiris Conseil, France
dominique.blouin@telecom-paris.fr

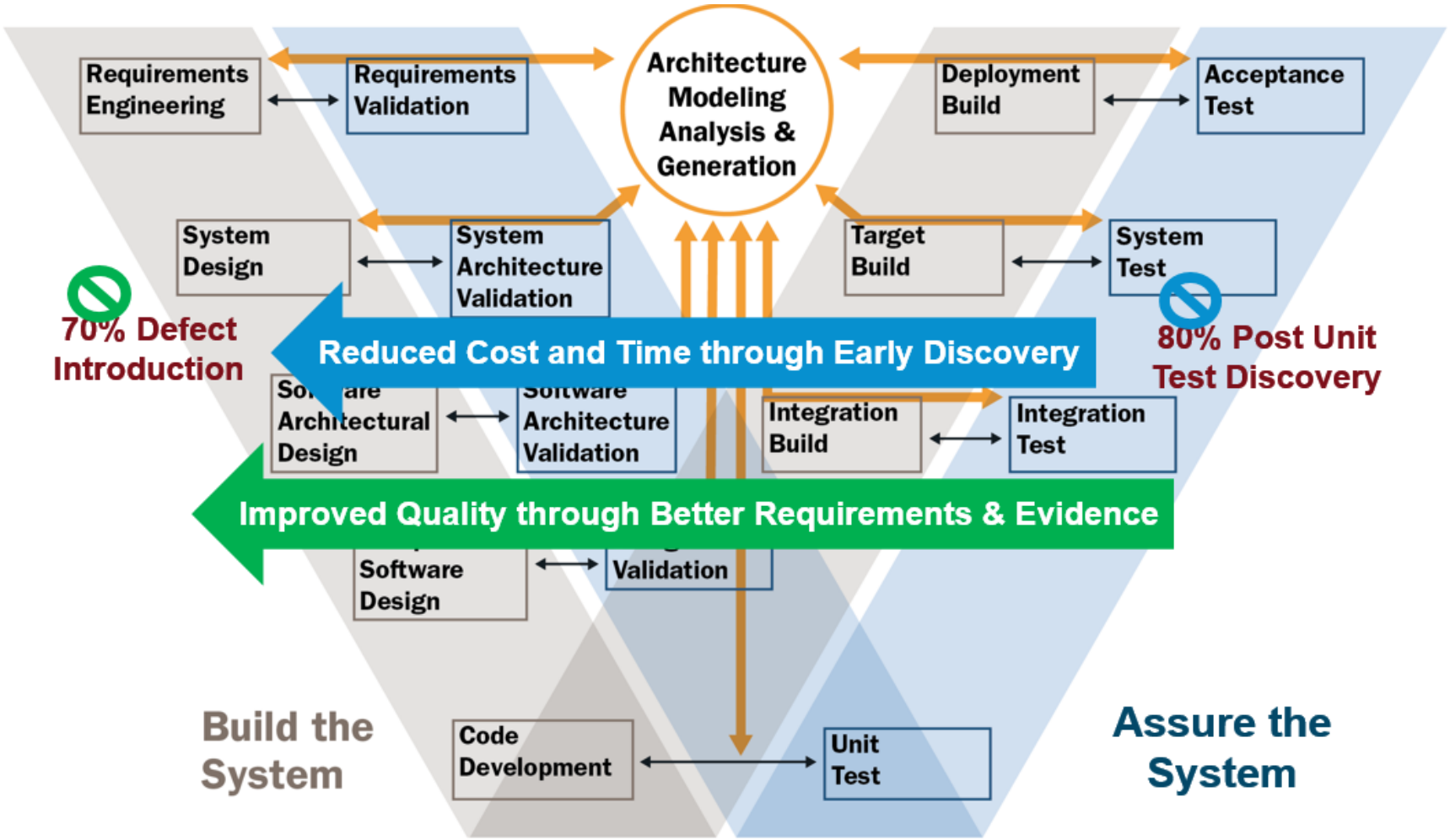
AADL at Telecom Paris

- **Members of AADL standards from the early days**
- **Several AADL-related projects**
  - https://mem4csd.telecom-paristech.fr/

Outline

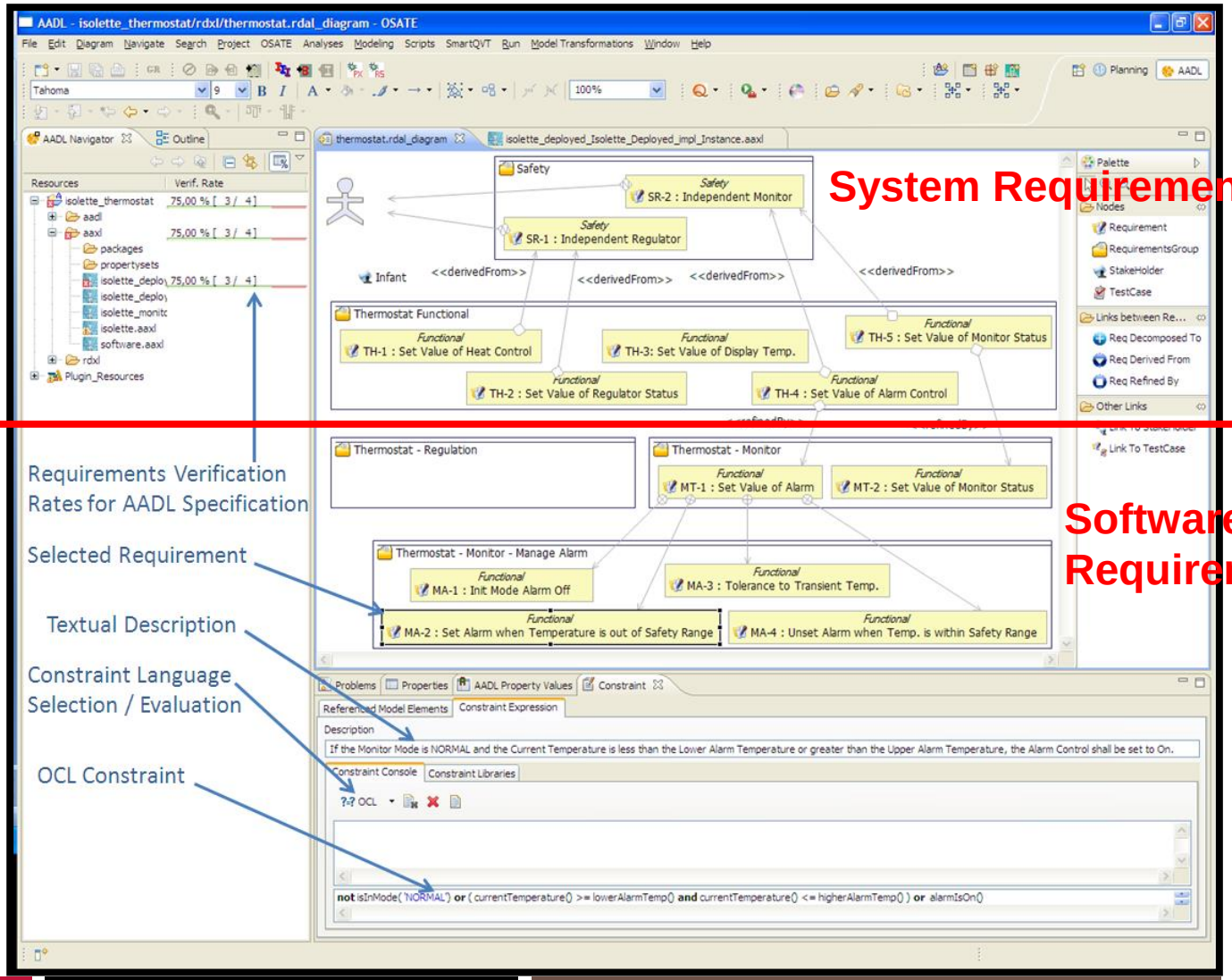
- **Context**
- **ALISA**
- **Experience Report on Railway Domain**
- **Conclusion and Perspectives**

ACVIP(Architecture-Centric Virtual Integration Process)



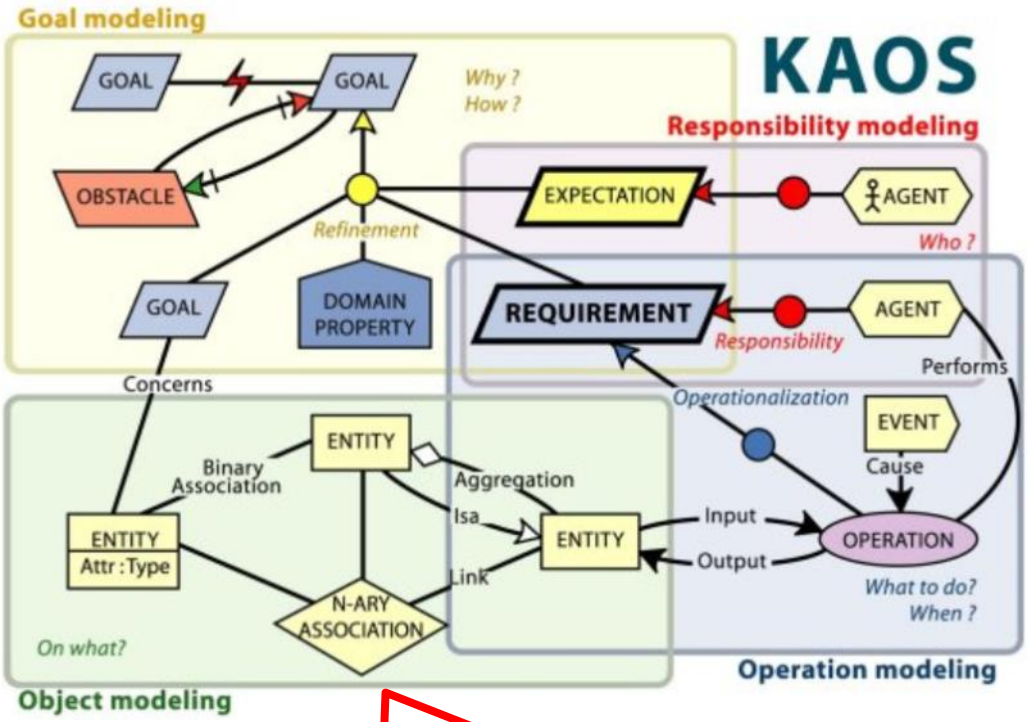
Source: McGregor, Gluch, and Feiler, *Analysis and Design of Safety-critical, Cyber-physical Systems*, 2017.

RDAL (Requirements Definition and Analysis Language) coupled with AADL



Inspired from Goal-Oriented Requirements Engineering (GORE)

- **4 complementary and interrelated views on the system:**
  - Goals (owners, users, business managers, regulations, etc.)
  - Responsible agents
    - Human and automated
    - System or environment
  - Problem domain
    - Concepts and their relationships
  - Behaviors
    - In order to achieve goals



Missing Domain-Specific Vocabulary

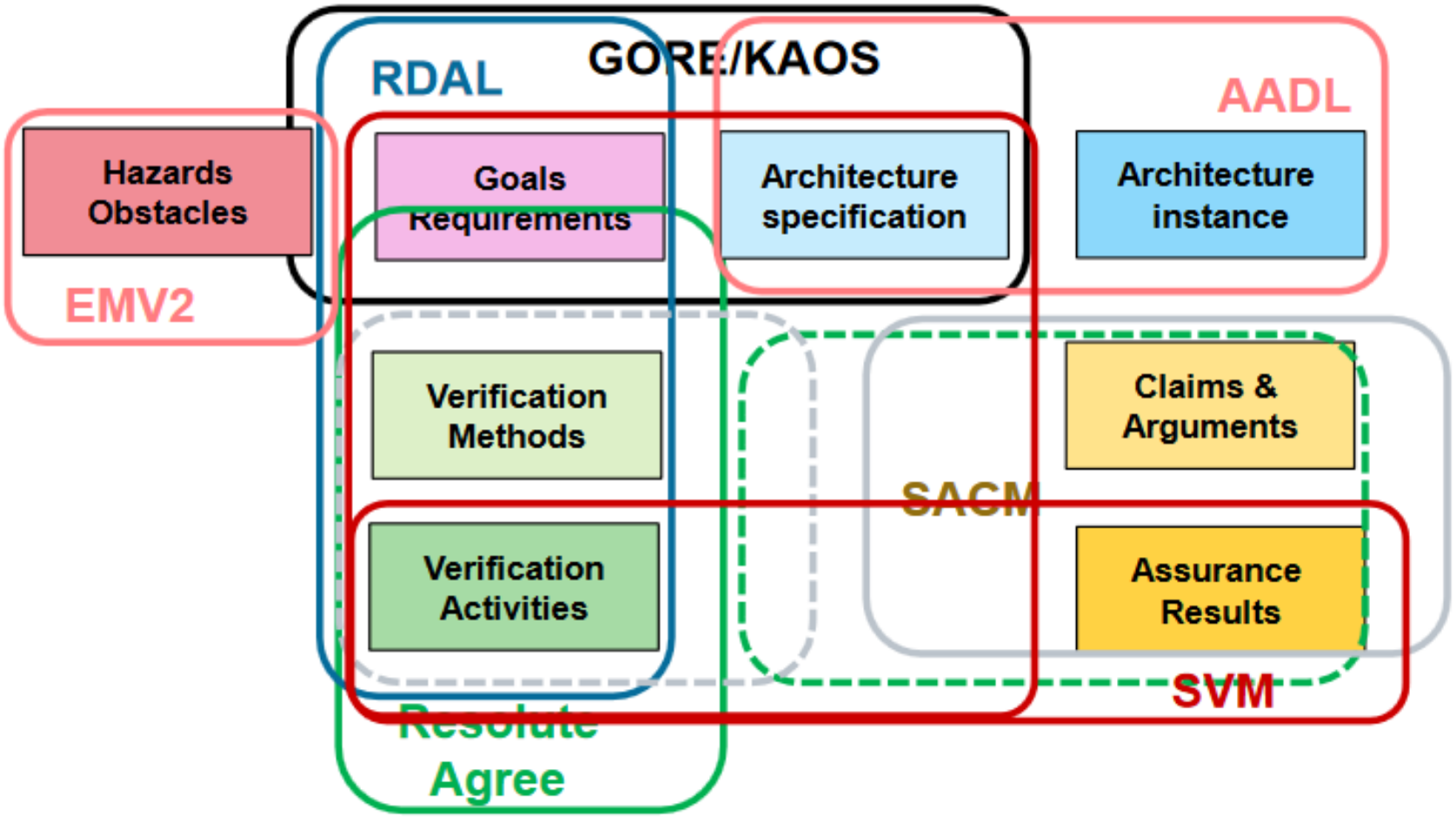
Outline

- Context
- ALISA
- Experience Report on Railway Domain
- Conclusion and Perspectives

ALISA (Architecture-Led Incremental System Assurance)

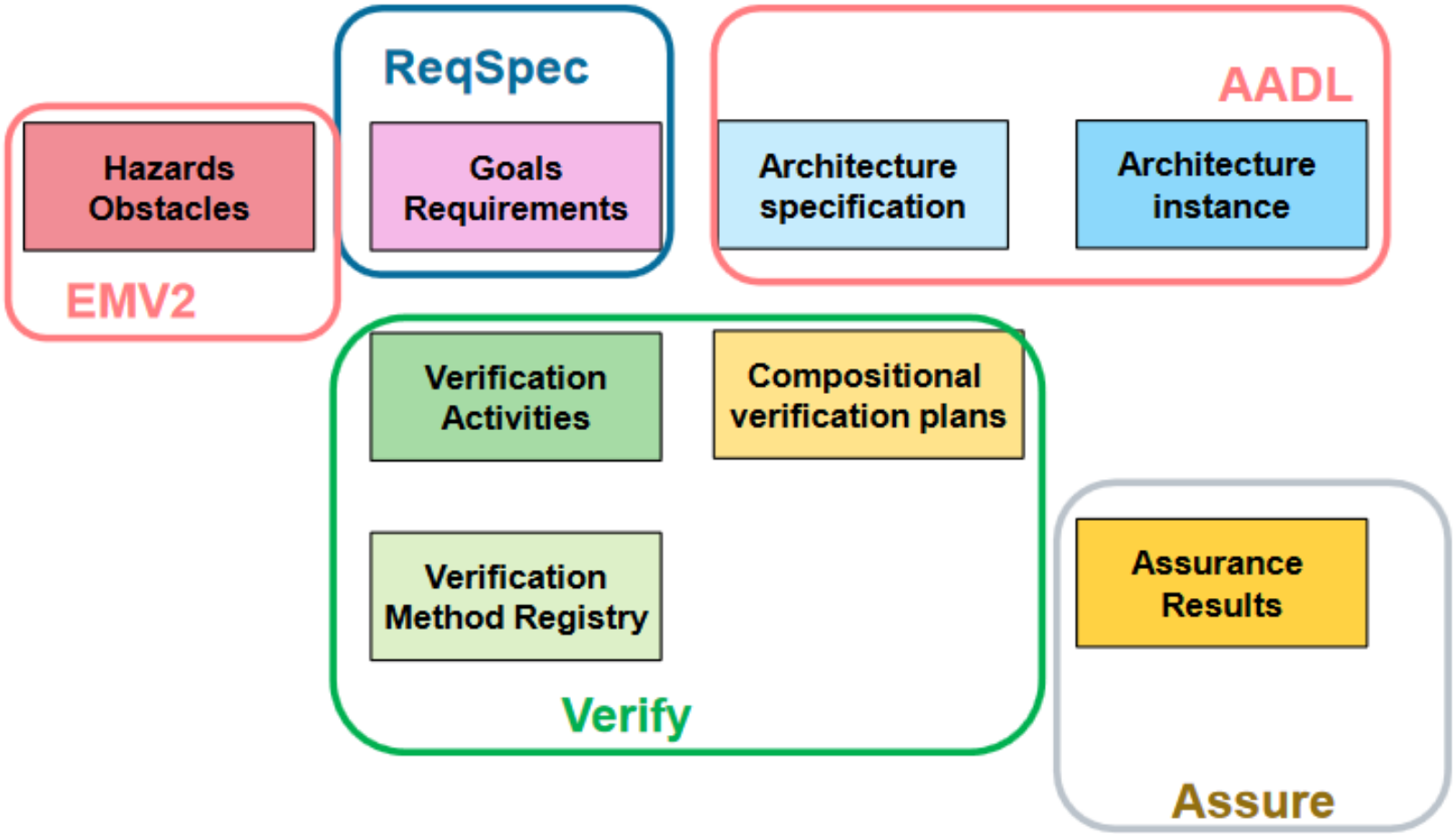
- Reimplementation of RDAL by SEI (Peter Feiler)
- Grammar based like AADL / OSATE
- Strongly coupled with AADL (extension of AADL grammar)
- Development of verification activity concepts
- Addition of assurance case modeling
- Link with error model annex (*mitigates* construct)

Languages and Concepts Basis for ALISA



Source: Peter Feiler, *ALISA Tutorial*, 2018.

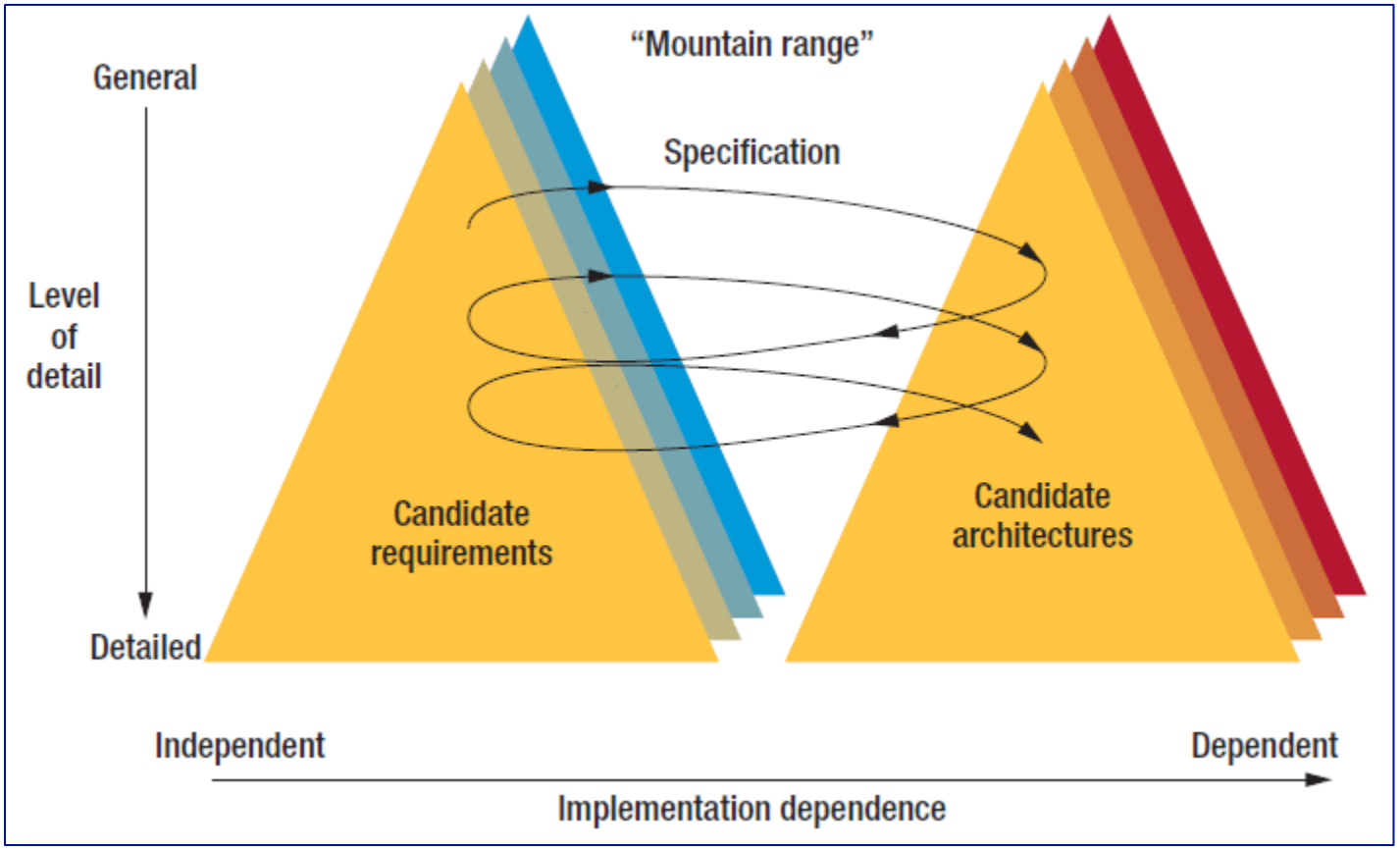
ALISA Unified Concepts



Source: Peter Feiler, *ALISA Tutorial*, 2018.

ALISA Incremental Process (Architecture-Led Incremental System Assurance)

- **Requirements and architecture "Twin Peaks" model**



Source: Cleland-Huang et al., *The Twin Peaks of Requirements and Architecture*, IEEE Software, 2013

ALISA Sub-Languages

- **Organization**: Defines the stakeholders of a project
- **ReqSpec: Stakeholder goals, system requirements**
  - Architecture-led
  - Verifiable
  - Coverage and uncertainty
- **Verify: Verification plans of verification activities**
  - Reasoning on how verification activities satisfy requirement
  - Verification methods (manual, automated)
  - Assumptions, preconditions on verification method
- **Alisa: Composition of verification plans into assurance cases**
  - Verification of AADL model artifacts across system architecture
  - Assurance tasks as filtered views of assurance plans
- **Assure: Manage assurance case instance execution and results**
  - Multi-valued logic evaluation of verification action & results
  - Acceptable risk factors (e.g., design assurance levels)
  - Filtered execution of assurance plans (based on category tags)

ReqSpec: Stakeholder Goals

```
stakeholder goals Line_Follower_Robot_Perf for Line_Follower_Robot_Cps::Line_Follower_Robot_Cps [

    goal G_Perf_1 : "Minimal Cost" [
        description "The cost of producing the robot should be minimal."
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer Tartempion_Warehouse_Equipments_Ltd.Marketing
        rationale "The robot should be cheap so that it is competitive on the market."
        category Quality.Cost
    ]

    goal G_Perf_2 : "Minimal_Transportation_Time" [
        description "The time taken to carry objects should be minimal."
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer Tartempion_Warehouse_Equipments_Ltd.Customer
        rationale "The robot should be fast to meet the needs of customers."
        category Quality.Performance
    ]
]
```

ReqSpec: Requirements

- **Decomposition: reference(s) to requirements of an enclosing system that this requirement is derived from**

```
requirement R_Behav_1 : "Carry_Object_Function" [
    description
            "The robot shall carry an object between two specified points by following a predefined trajectory in the warehouse."
    see goal Line_Follower_Robot_Behavior.G_Behav_1
    category Quality.Behavior
]

    requirement R_Behav_1_1 : "Pick_Up_Object_Function" for pick_up_object [
        description "At the beginning of the path, the robot shall pick up an object on the floor."
        category Quality.Behavior
        decomposes R_Behav_1
    ]

    requirement R_Behav_1_2 : "Follow_Line_Function" for follow_line [
        description "The robot shall follow a line on the floor of the warehouse."
        category Quality.Behavior
        decomposes R_Behav_1
    ]
```

- **Refinement: provides a more detailed specification for the same system. Requirements are refined until they become verifiable.**

```
requirement ETCS_OB01_evc_2oo3_design_redundancy : "All CPUs of the EVC shall execute the same functions" [
  refines ETCS_OB01_evc_2oo3_design
  category Quality.Safety
]
```

ReqSpec: Verifiable Requirements

Via predicate

```
requirement ERA_5_2_1_1_evc : "EVC response time" for message_processing_flow [

   decomposes ERA_5_2_1_1

   compute SystemOperationMode: string
   compute MinLatency : Time
   compute MaxLatency : Time
   val MaxLatencyInMs : real = (MaxLatency%us / (1 us)) / 1000
   val PipelinePeriod : real = (#Middleware_Properties::Default_Hyper_Period%us / (1 us)) / 1000
   description "Delay between reception of an input data message and output command dispatch."
   "Delay shall be < " MaxEVCResponseTime
   value predicate MaxLatency < MaxEVCResponseTime
   category Quality.Latency
]
```

Via verification activity

```
requirement ETCS_OB01_evc_2oo3_design_redundancy : "All CPUs of the EVC shall execute the same functions" [
   refines ETCS_OB01_evc_2oo3_design
   category Quality.Safety
]
claim ETCS_OB01_evc_2oo3_design_redundancy [
   activities
   redundancy : Resolute.allFunctionsAreRedounded ( )
]
```

ReqSpec: Verifiable Environmental Assumptions

- **Requirements that constrain the environment of the system**

```
system requirements Line_Follower_Robot_Env_Assumptions for Line_Follower_Robot::Warehouse_Robots.normal [

    requirement EA_1: "Minimum Warehouse Luminosity" for light_source [
        description "The power of the light source shall not be less than the Minimum Illuminance value"
        rationale "Otherwise the light sensor of the robot will not be able to give proper readings given its sensitivi
        category Kind.Assumption
        val Minimum_Illuminance = 100.0 lx
        value predicate #Physics_Properties::Illuminance >= Minimum_Illuminance
    ]

    requirement EA_2: "Minimum Curvature Radius" for line [
        description "The curvature radius of the line to be followed by the robot shall not be lower than TODO"
        rationale "Otherwise the robot given its speed, mass and response time will not be able to follow the line."
        category Kind.Assumption
        val Minimum_Curvature_Radius = 100.0 mm
        value predicate #Physics_Properties::Curvature_Radius >= Minimum_Curvature_Radius
    ]
]
```

- **Organization of requirements:**
  - Requirements and goals can be declared into sets (packages)
  - Global sets can be declared that can be reused across systems

Verify: Methods and Registry

- **Reusable verification methods on models and other artifacts**
  - OSATE Analysis plugins, Java methods, Python Scripts, Resolute claim functions, JUnit-based code tests

```
verification methods KPIs [
    method Save(component, value: real): "Save a KPI, name and value" [
        java fr.irt_systemx.pst.alisa.kpi.Persistence.save(double value)
        description "Save a KPI, name and value"
    ]

    method Compute(component, value: real) returns (value: real): "Compute a KPI" [
        java fr.irt_systemx.pst.alisa.kpi.Persistence.compute(double value)
        description "Compute a KPI"
    ]
```

```
verification methods Resolute [

    method allFunctionsAreRedounded ( ) : "Functions shall be redounded" [
        description "Check that all functions running on the CCP2 are the same" resolute
                Middleware_2oo3.Design_2oo3_Verification.Middleware_2oo3.Design_2oo3_Verification_publi
    ]

    method cpusAreOfSameType ( ) : "CPUs shall be of same type" [
        description "Check that CPUs are of same type" resolute
                Middleware_2oo3.Design_2oo3_Verification.Middleware_2oo3.Design_2oo3_Verification_publi
    ]
```

Verify: Verification Plans

- Made of claims aligned with system requirements
  - Contain verification activities invoking verification methods of registry

```
verification plan ETCS_OnBoard_Safety_Verification for ETCS_OnBoard_Safety_Requirements [

    claim ETCS_OB01 [
        activities
        persistence: KPIs.Save(kpi)
    ]

    claim ETCS_OB01_evc [
        claim ETCS_OB01_evc_2oo3_design [
            claim ETCS_OB01_evc_2oo3_design_redundancy [
                activities
                redundancy : Resolute.allFunctionsAreRedounded ( )
            ]

            claim ETCS_OB01_evc_2oo3_design_cpus_make_and_model [
                activities
                consistency : Resolute.cpusAreOfSameType ( )
            ]
        ]
    ]
]
```

Verify: Multi-Valued Verification Activity Results

- Verification activity result states
  - Success, fail, error, tbd
- Compositional argument expressions
  - All: a collection of independent Vas
  - Va1 then Va2: Execution of Va2 dependent on success of Va1
  - Va1 else Va2: Execute Va2 only if Va1 produces negative result
  - Va1 else [fail: Va21 timeout: Va22 error: Va23]
- Mode specific verification activities
- Parameterized verification activities (data sets as input)

Assure: Assurance case, plans and tasks

- Assurance plan: configuration of assurance case:
  - Which part(s) of architecture to be verified using which verification plans
- Assurance plan instantiation & execution
  - Automated verification activity execution
  - Tracking of result state and reports
- Assurance Task
  - Filtered assurance plan instances based on requirement, verification, and verification activity selection categories

```
assurance case ETCS_OnBoard_Case for Middleware_2oo3::Integrated [

    assurance plan ETCS_OnBoard_Middleware_Plan for Middleware_2oo3::Integrated.basic [
        description "Assurance plan for ETCS on board functions and software"
        assure subsystem all
        assure ETCS_OnBoard_Design_Rules_Verification ETCS_OnBoard_Performance_Verification ETCS_OnBoard_Safet
    ]

]
```

Assure: Assurance Case Execution and Results

- Assurance view in OSATE to execute assurance cases

```
ystem requirements ETCS_OnBoard_Performance_Requirements for Functions_2oo3::Integrated.basic [
   description "These are some ERA performance requirements for the ETCS on board system"

   requirement ERA_5_2_1_1 :
          "Emergency break delay" [
      val MaxEVCResponseTime = 300 ms
      val MaxUpstreamResponseTime = 350 ms
      val MaxDownhillResponseTime = 350 ms
      val MaxEmergencyBreakDelay = MaxUpstreamResponseTime + MaxEVCResponseTime + MaxDownhillResponseTime
      value predicate MaxEmergencyBreakDelay <= 1 sec
      description "Delay between receiving of a balise message and applying the emergency brake."
             "StartEvent: The reference mark of the on board antenna leaving the side lobe zone of the last balise in
             "Sto
             MaxE
      category Qua
   ]

   requirement ERA_
      decomposes E
      compute Syst
      compute MinL
      compute MaxL
      description
             MaxEVCRespons
      value predicate MaxL
      category Quality.Late

   ]
```

```
verification plan ETCS_OnBoard_Performance_Verification for ETCS_OnBoard_Performance_Requirements [

   claim ERA_5_2_1_1 [
      // Just check the predicate defined in the requirement itself
   ]

   clai

   ]
]
```

rmation | Assurance View | Progress | Search | Error Log | Assurance Case | Console | History

Evidence	Pass	Fail	Err	Todo	Description
Case ETCS_OnBoard_Case	5	1	1		
Plan ETCS_OnBoard_Middleware_Plan(Integrated.basic)	5	1	1		
Claim engineering_design_rules_full_connect		1			As a design good practice, all components in a model shall have all
Claim engineering_design_rules_perdiodic_threads	1				All threads shall be periodic
Claim ERA_5_2_1_1	1				Delay between receiving of a balise message and applying the emer
Predicate	1				(MaxEmergencyBreakDelay <= 1 sec)
Claim ERA_5_2_1_1_evc(message_processing_flow)	1				Delay between reception of an input data message and output com
Evidence responsetime	1				Analysis of all end-to-end flows in a system instance or for a specific
Success: message_processing_flow					Latency results for message_processing_flow
i Value:					
i Value: 23.338					
i Value: 146.18					

Outline

- **AADL**
- **ALISA**
- **Experience Report on Railway Domain**
- **Conclusion and Perspectives**

Using ALISA for Railway Software Systems

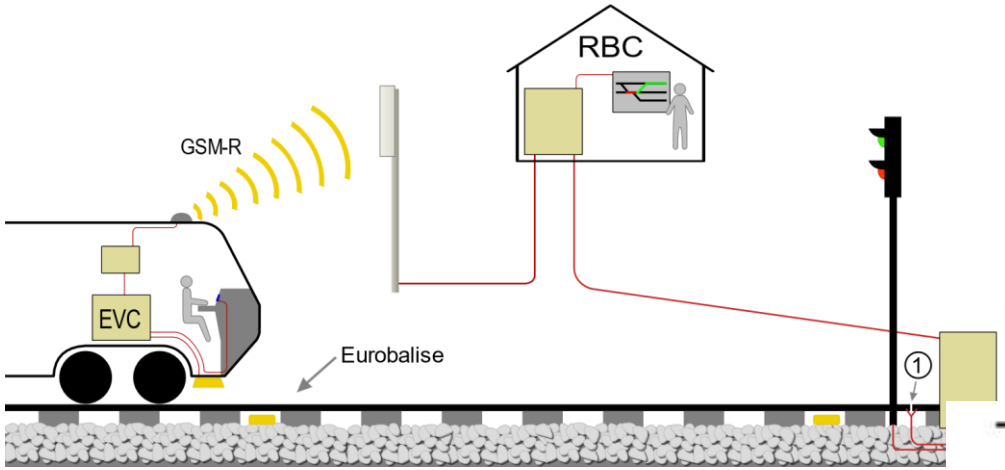
Engineering Railway Systems with an Architecture-Centric Process Supported by AADL and ALISA: an Experience Report

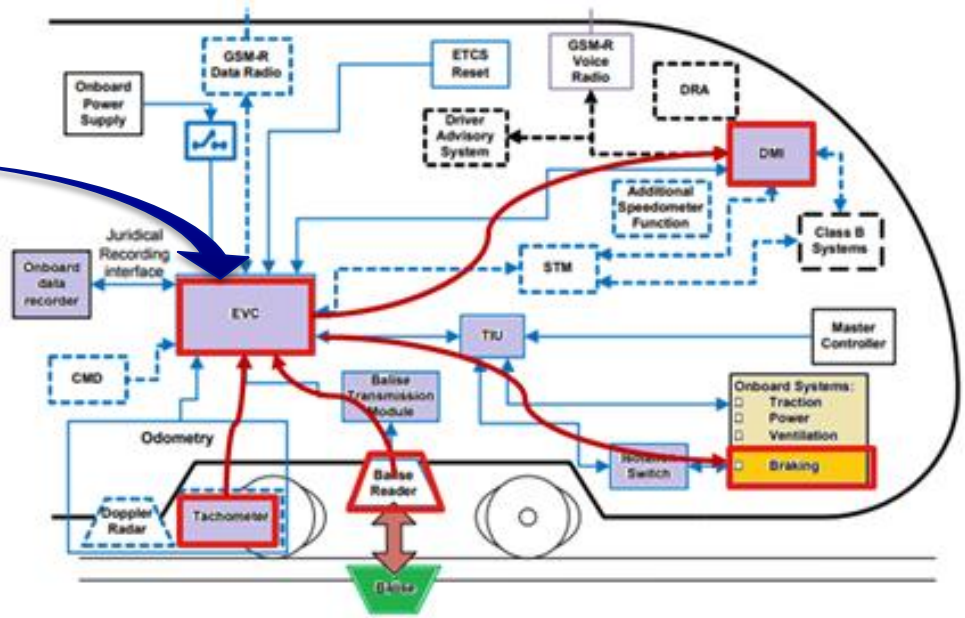
Paolo Crisafulli[1], Dominique Blouin[2], Françoise Caron[3], and Cristian Maxim[1]

[1] IRT SystemX, Paris, France firstname.lastname@irt-systemx.fr
[2] LTCI, Telecom Paris, Institut Polytechnique de Paris, France dominique.blouin@telecom-paris.fr
[3] Eiris Conseil, France francoise.caron@eiris.fr

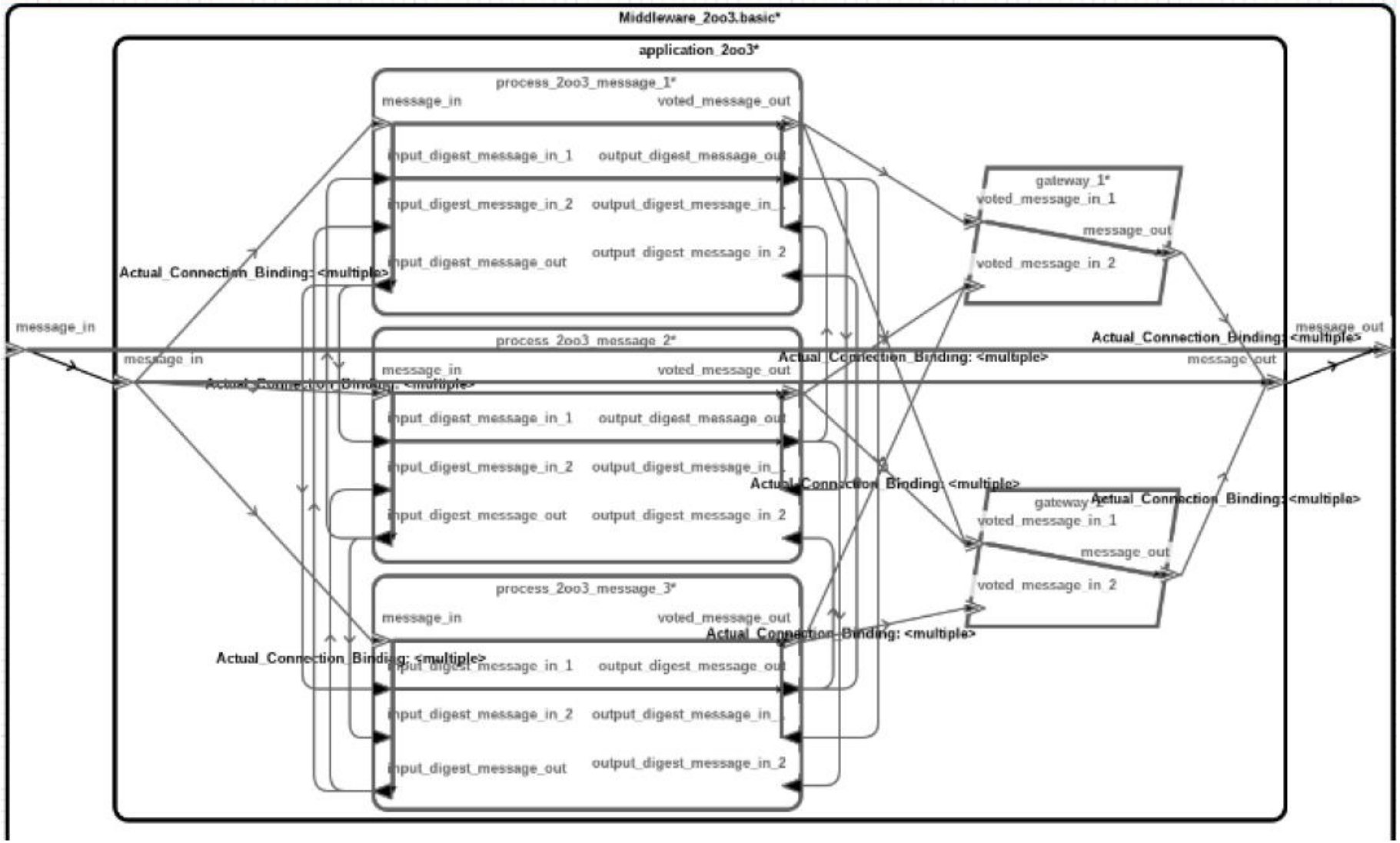
- Many of the ALISA examples taken from this work

European Train Control System Case Study: EVC (European Vital Computer)

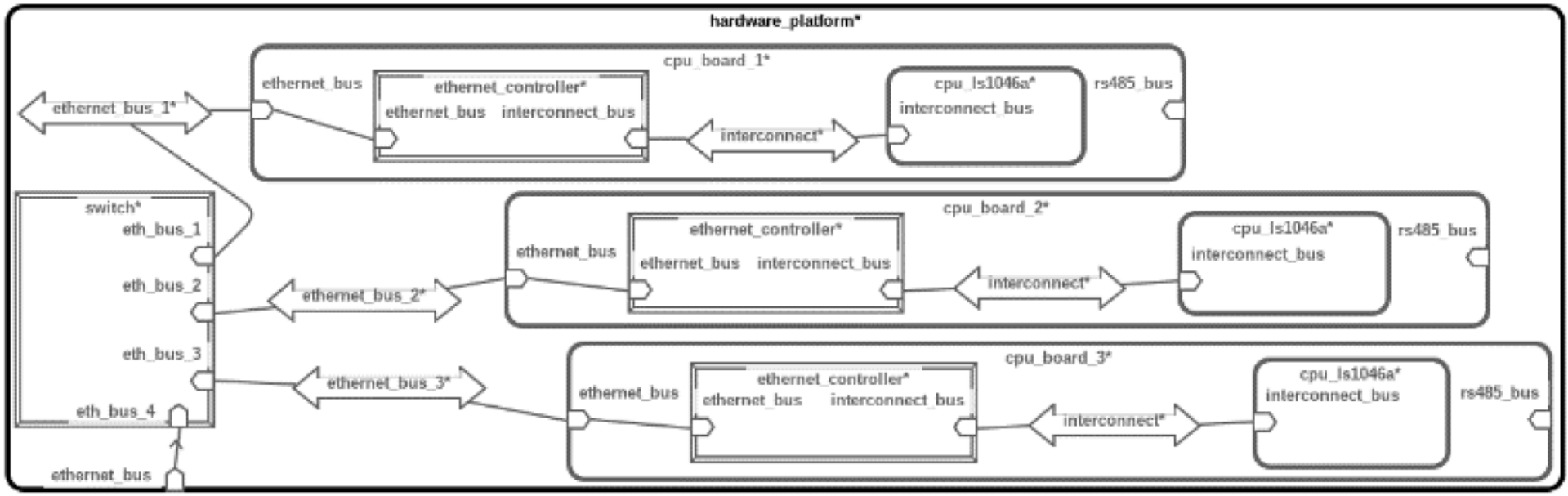




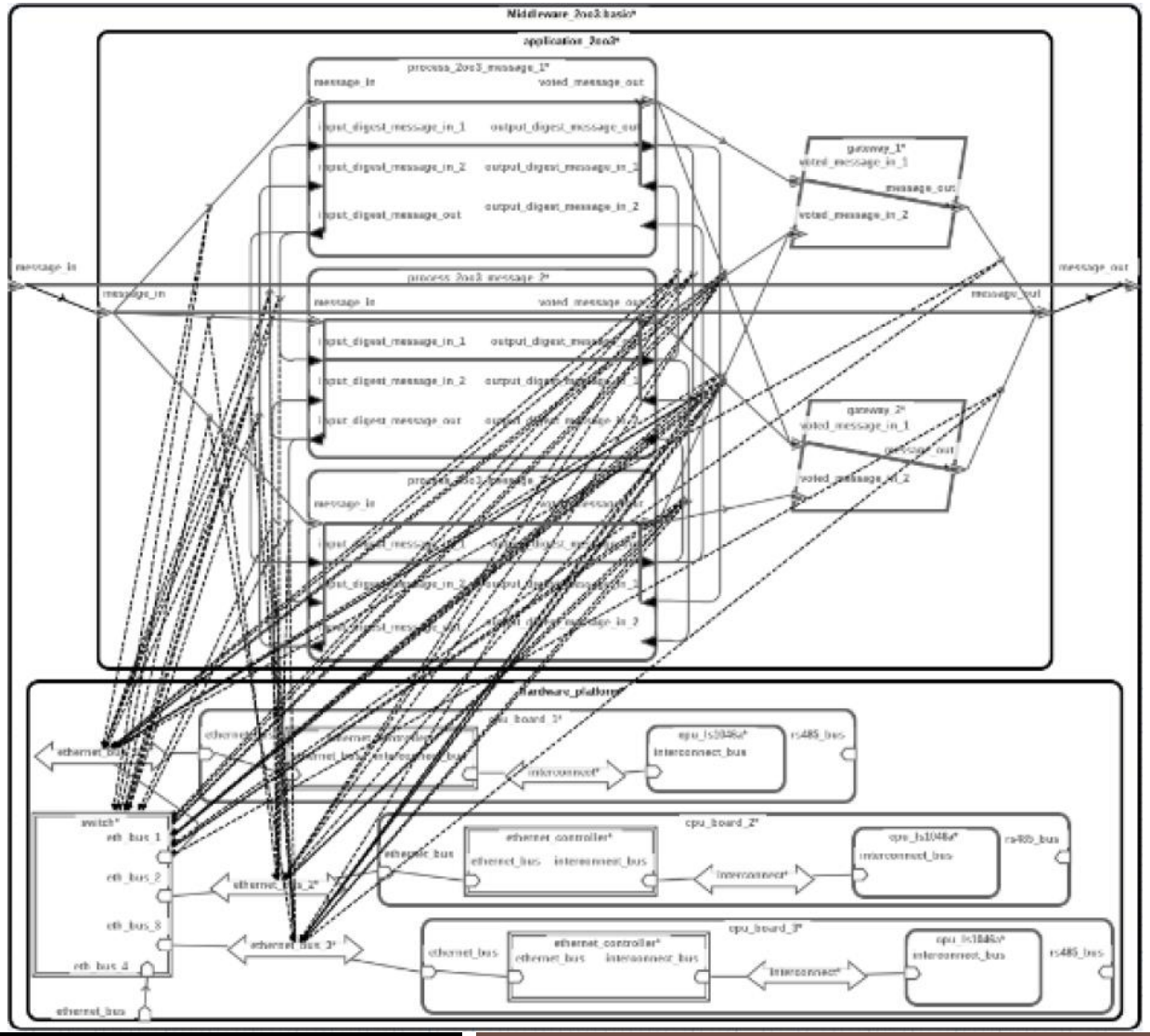
EVC (European Vital Computer) Railway System Case Study: Software View



Railway System Case Study: Execution Platform View



Railway System Case Study: Execution Platform View



Some Requirements for the EVC

- **Focus on performance requirements verification:**
  - Latency < 300 ms
  - Incoming messages <= 1000 msg/s
  - Safety and availability:
    - THR of 0.67 x 10^{-9} dangerous failures/hour
    - 2oo3 (akaTMR) design
      - Verify some 2oo3 design constraints
      - Same threads shall run on each board
      - Boards shall be of the same model

- **Design rules (reusable good practices)**
  - All input and output ports, physical or logical, shall be connected
  - All threads shall be periodic

Towards an Agile Engineering Process

- Reuse the continuous integration paradigm from software development
- Define ALISA requirements for major design choices
- Implement continuous requirements verification to maintain design within the solution space shaped by the set of requirements
- KPIs computation and charting to qualify in terms of performance the evolution of design and alternatives over time
  - Provide KPI charts
  - Design goals not working

Building Blocks

Osate/ALISA/AADL Inspector

AADL parsing, analysis and verification platform

Git/Repo

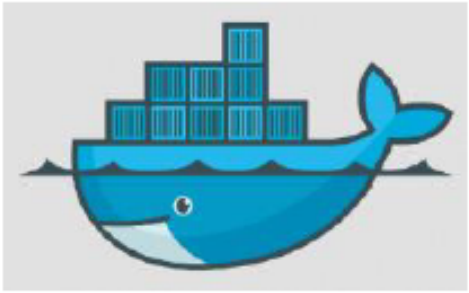
Versioning system for the comprehensive source of all artifacts:

- Requirements
- Models and Code
- Verification activities
- Dockerfiles

Jenkins

Continuous integration

Triggers verification check on any change to the artifacts

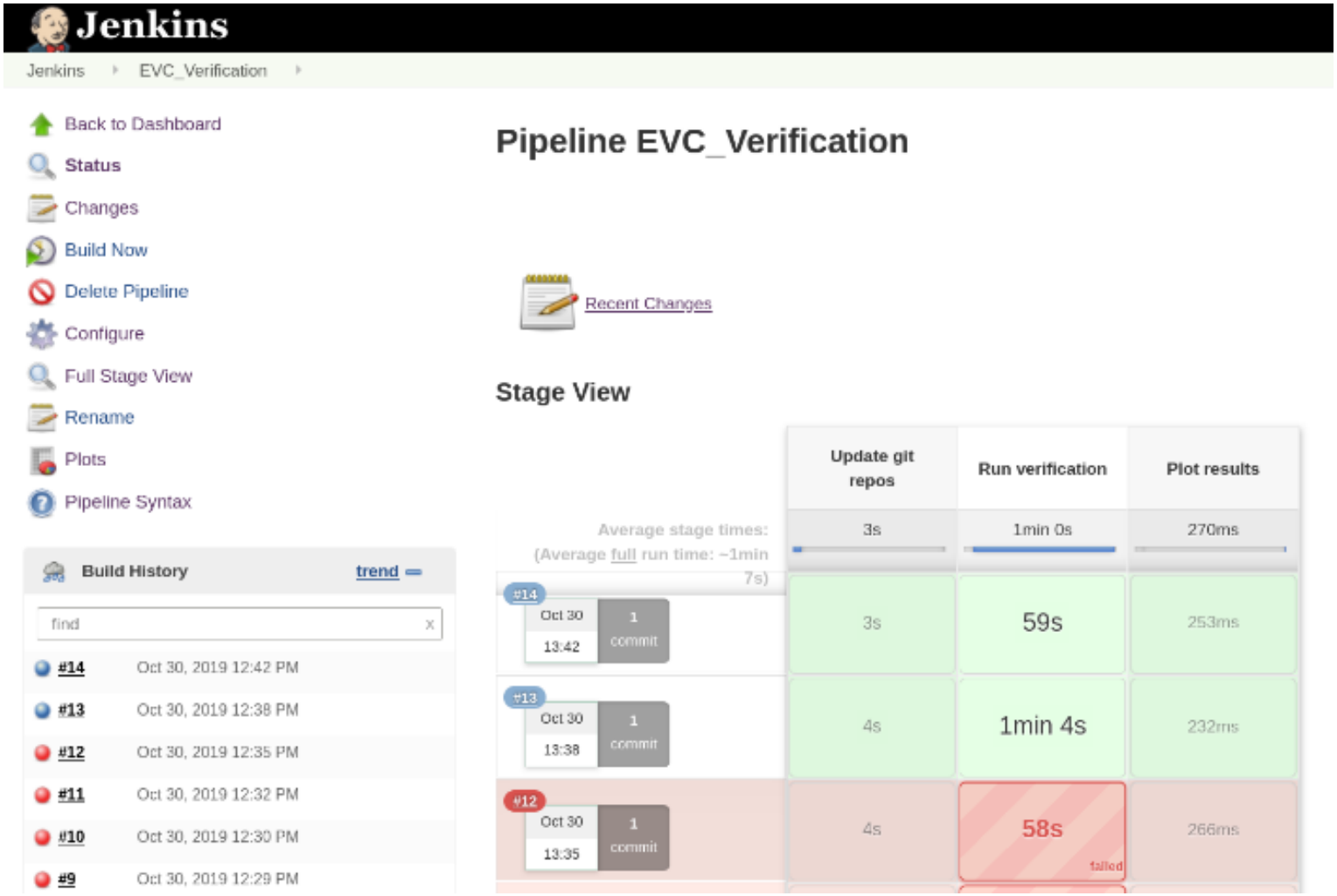


Docker

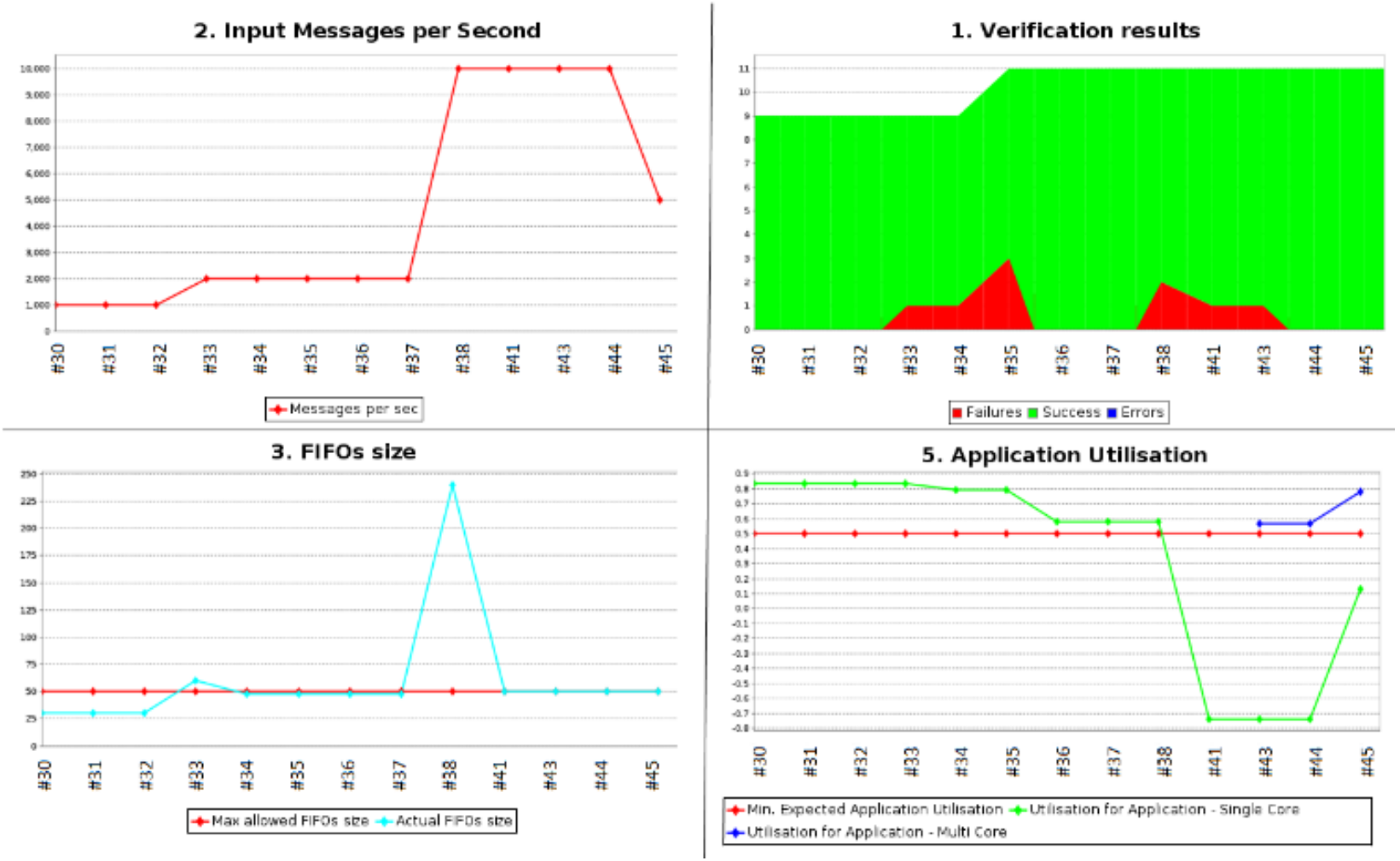
Container platform

Configuration management of the development, build and test environments

Build History



Build History



Outline

- **AADL**
- **ALISA**
- **Experience Report on Railway Domain**
- **Conclusion and Perspectives**

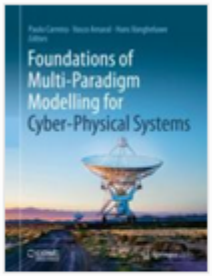
Conclusion and Perspectives

- Demonstrated how AADL and ALISA can support agile architecture-centric engineering processes:
  - The continuous verification maintains the design within the solution space shaped by the set of requirements
  - The KPIs computation and charting qualify system performance evolution of design alternatives over time
- ALISA is not yet completely mature:
  - Scalability and multi-organization issues to be addressed
  - Design goals
- AADL ecosystem of companion languages and development environments opens the way to agile engineering of highly constrained systems requiring a certification process
- Additional work on the overall system engineering process published at SysCon 2020

More Info on ALISA

- **AADL Tutorial by Peter Feiler: https://apps.dtic.mil/sti/pdfs/AD1084078.pdf**

- **OSATE online help**



Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems pp 209-258 | Cite as

AADL: A Language to Specify the Architecture of Cyber-Physical Systems

Authors Authors and affiliations

Dominique Blouin, Etienne Borde

Integration of ALISA into the PST Process

Seamless Integration between Real-time Analyses and Systems Engineering with the PST Approach

Françoise Caron
EIRIS Conseil
Jouy-en-Joas, France
francoise.caron@eiris.fr

Dominique Blouin
LTCI, *Telecom Paris*
Institut Polytechnique de Paris
Palaiseau, France
dominique.blouin@telecom-paris.fr

Paolo Crisafulli
IRT SystemX
Palaiseau, France
paolo.crisafulli@irt-systemx.fr

Cristian Maxim
IRT SystemX
Palaiseau, France
cristian.maxim@irt-systemx.fr