SMT-based verification of Cyber Physical Systems

Alessandro Cimatti

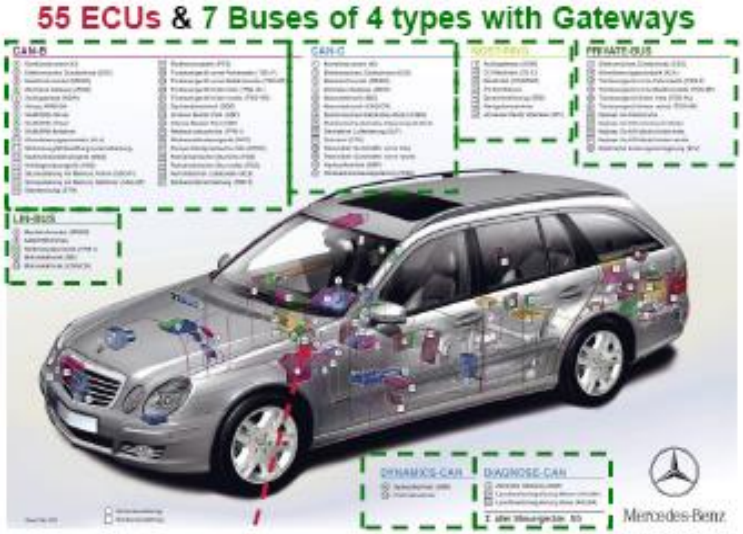
Fondazione Bruno Kessler (FBK), Trento, Italy

September 18, 2018

INTRODUCTION

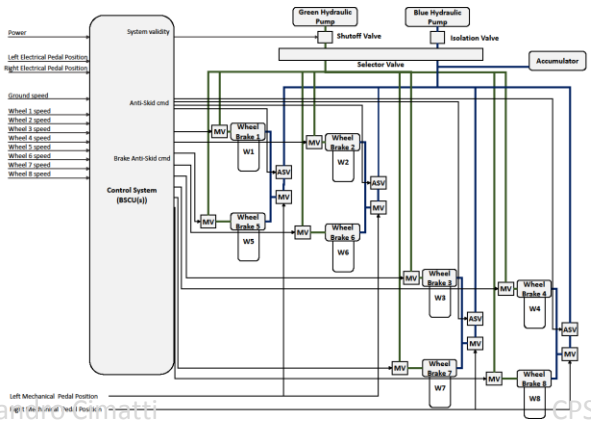
CPS: The Design Challenge

- Designing complex systems
  - Automotive
  - Railways
  - Aerospace
  - Industrial production
- Sources of complexity:
  - Hundreds of functions
  - Networked control
  - Real-time constraints
  - Complex execution model with mixture of real-time and event-based triggers
  - System composed of multiple heterogeneous subsystems
  - Critical Functions:
    - ABS, drive-by-wire
    - Operate switches, level crossings, lights
    - Manage on-board power production
  - Conflicting objectives:
    - Avoid crashes vs move trains

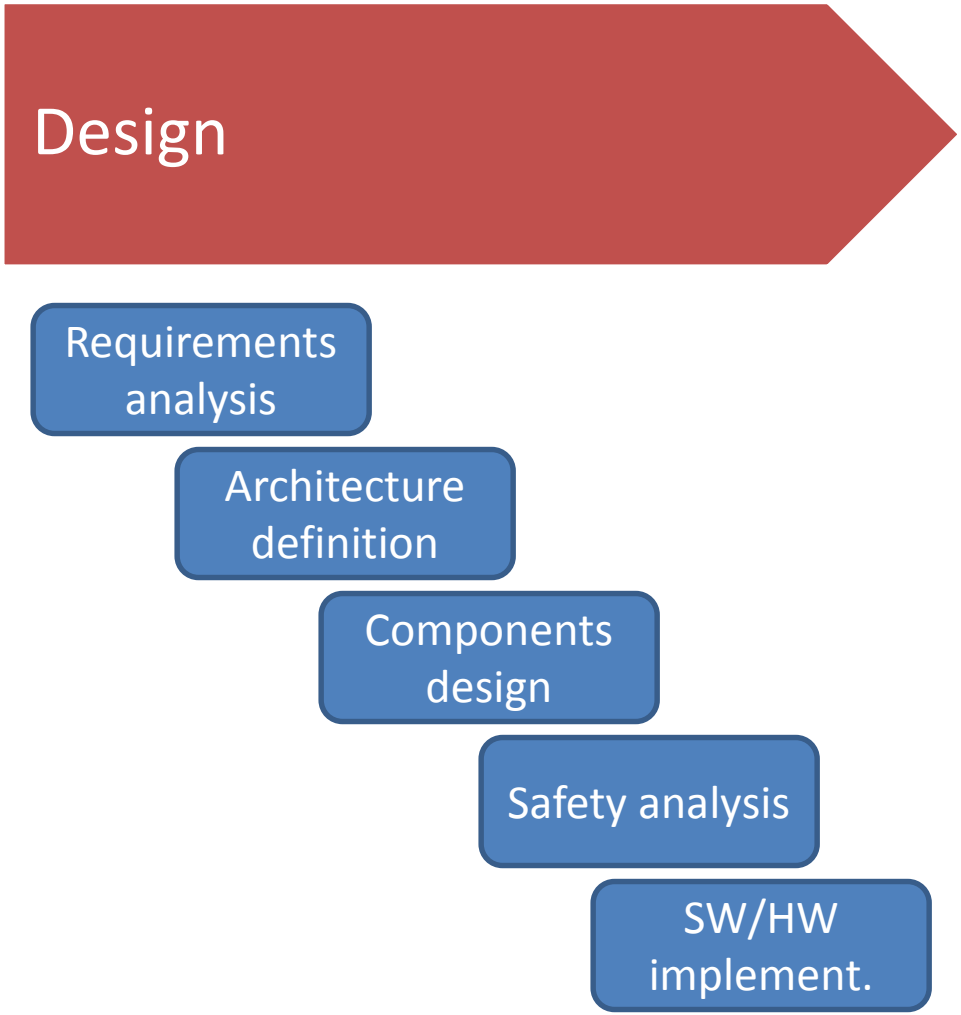


A Wheel Brake System

- Control brake for aircraft wheels
- Redundancy
  - Multiple BCSU
  - Hydraulic plants
- Functions
  - Asymmetrical braking
  - Antiskid
    - Single wheel/coupled
    - depending on control mode



Life Cycle of Complex Systems

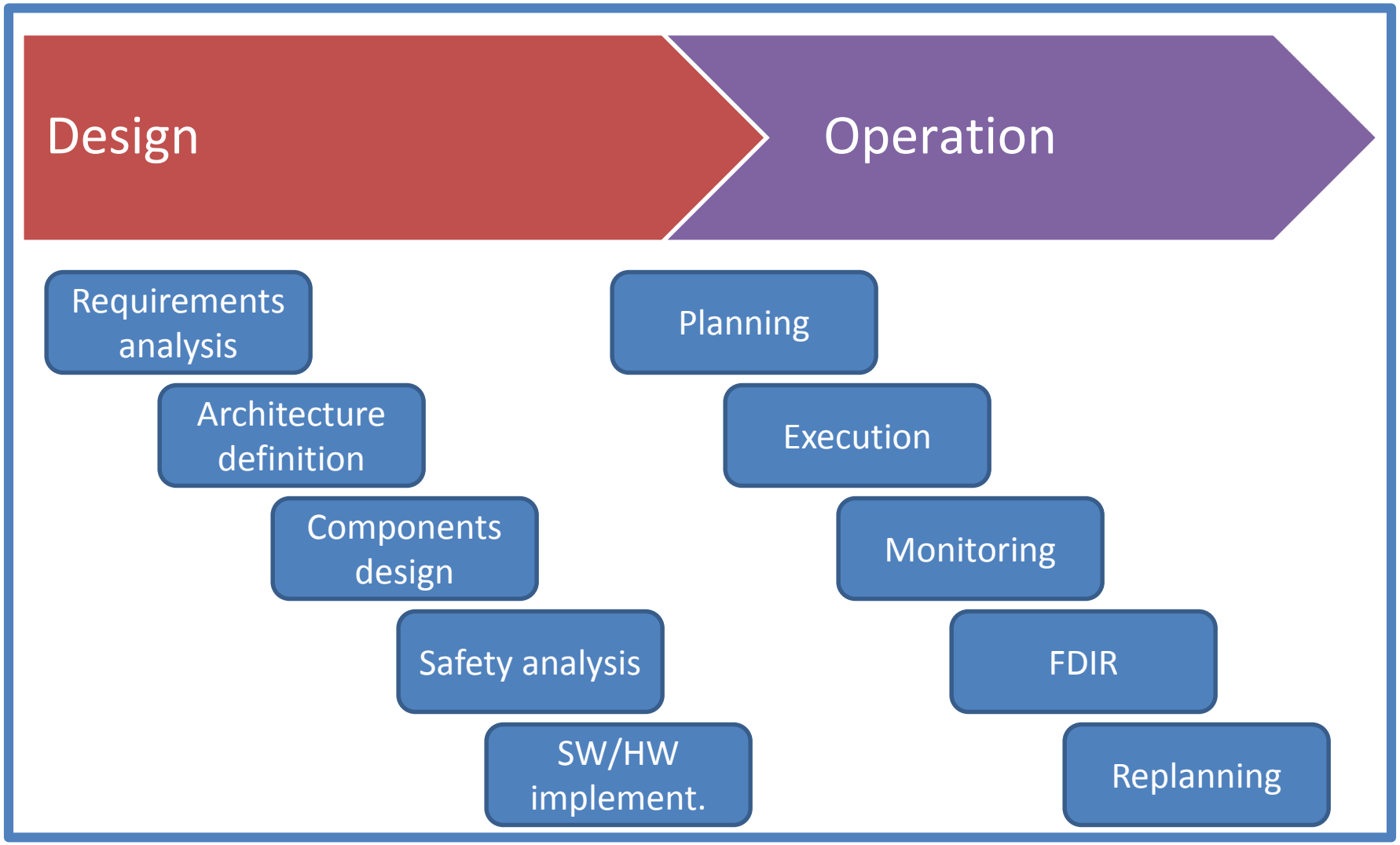


- Functional correctness
  - Does the system satisfy the requirements?
- Requirements validation:
  - Are the requirements flawed?
- Safety assessment
  - Is the system able to deal with faults?

Complex systems

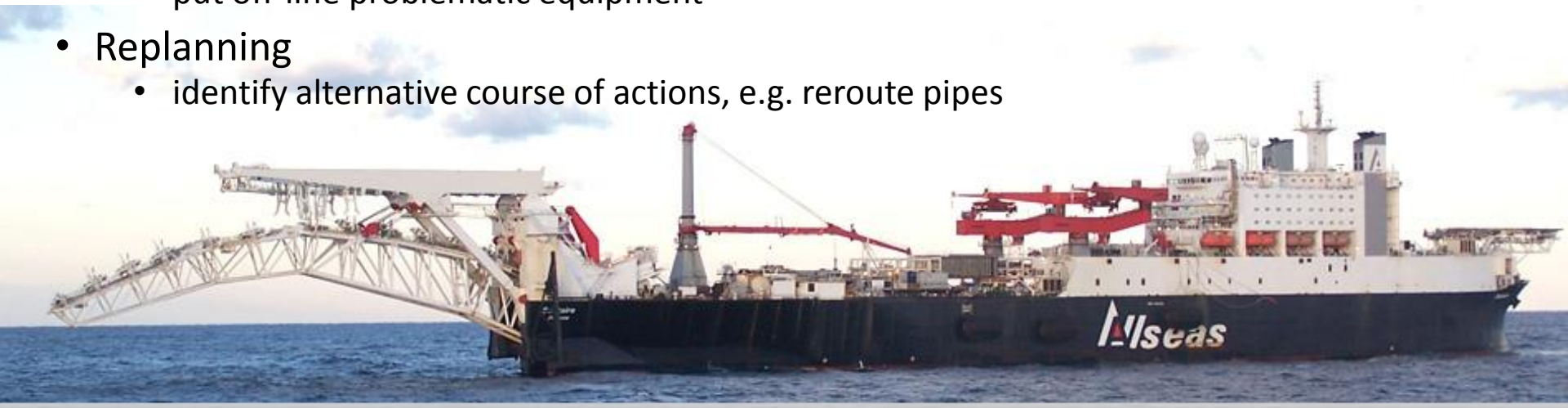
- Source of difficulty: critical systems
- Must provide reliable response to very wide range of adverse conditions
  - Redundancy, reconfiguration
- Examples:
  - Wheel brake system
  - Power supply on board of a large-sizes aircraft
- Key remark: operational conditions and response thoroughly analyzed *upfront*
- Validation of reconfiguration policies
  - As designed “off-line”

Life Cycle of **Adaptive** CPS



From design to operation…

- Planning
  - plan how to achieve desired “firing” sequence
  - retrieve pipes from holds, pre-weld, send to firing line, final weld
- Execution Monitoring
  - welding may fail, activities can take more time than expected
  - plant may fail
- Fault Detection, Fault Identification/Isolation
  - is there a problem? where is it?
- Fault Recovery
  - put off-line problematic equipment
- Replanning
  - identify alternative course of actions, e.g. reroute pipes



Factory automation projects

- Activity scheduling in galvanic coating factories
  - Execute precise “recipe”
  - Quick re-plan for production changes
  - Fault tolerance
- Estimation of expected costs
- Helping in design of flexible and efficient plants

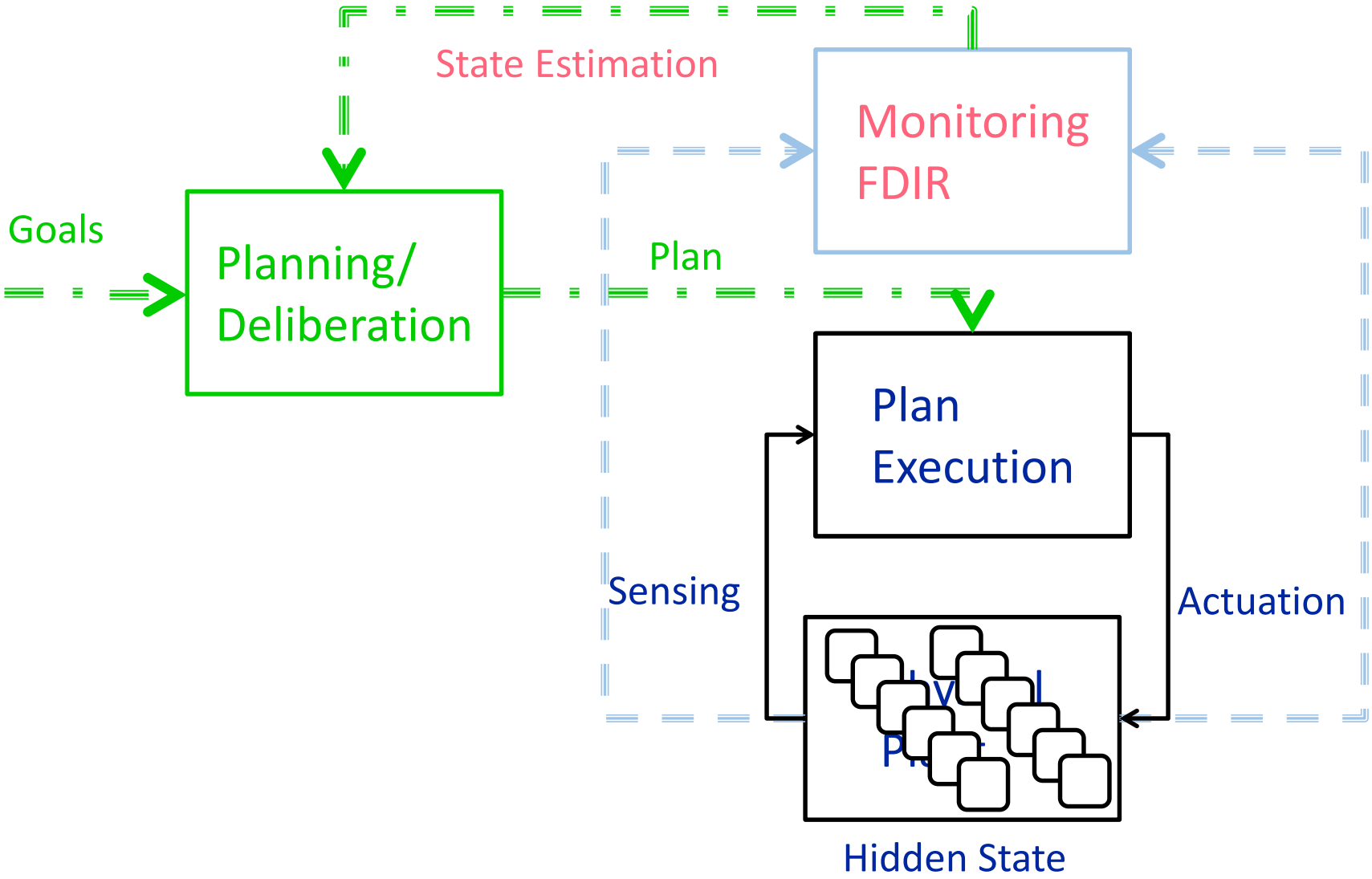
Galvanic processes and plants

- Sequence of chemical washes
- Timing is crucial
- Pieces moved in stocks by carriage-mounted forklifts
- Once started, cannot be interrupted without quality degradation

Adaptive/reconfigurable systems

- Highly optimized functions in controlled environments
- Unpredictable sequence of missions
  - Arrival of urgent production batch
  - Degraded operational conditions
- Advanced methods for
  - Automated programming
  - Simulation and cost estimation

Operation of adaptive systems

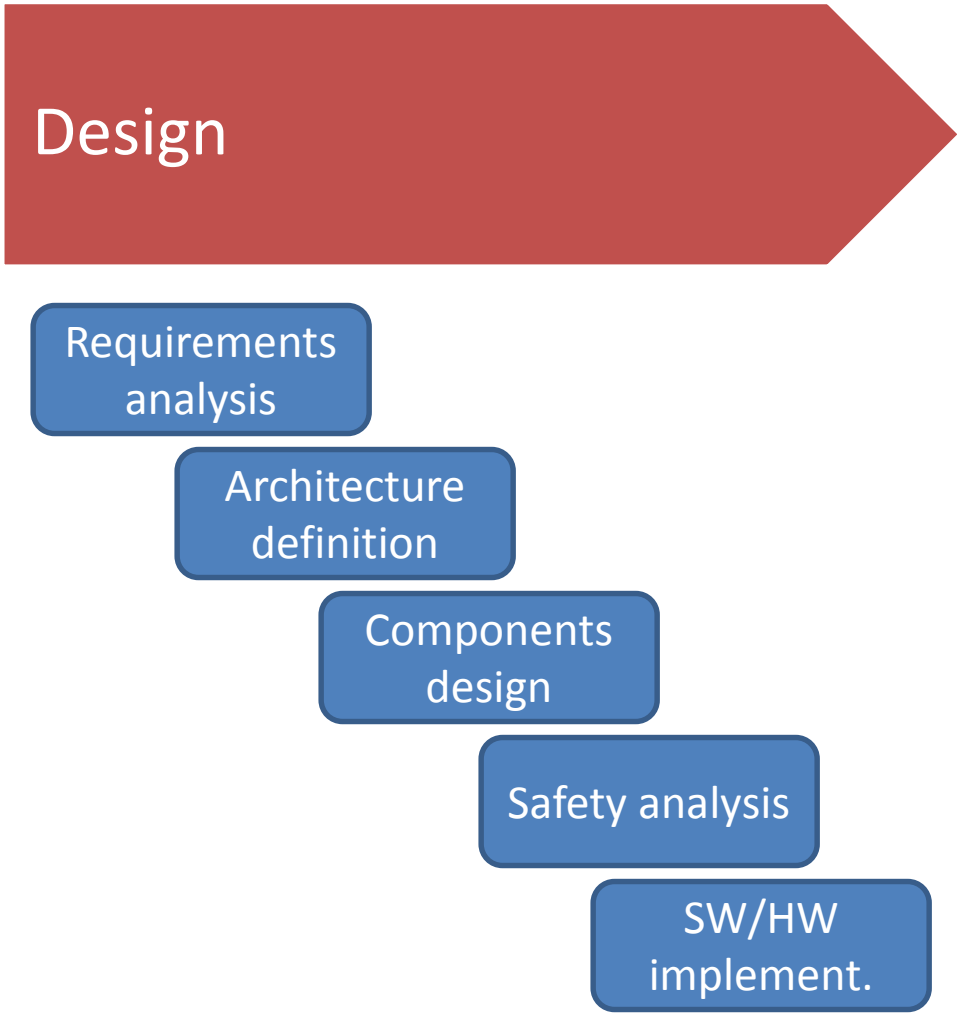


Automated planning and monitoring

- Plan validation
  - Does plan achieve required objectives?
  - Could be manually generated
- Planning as generation of suitable course of actions
  - Actions with possibly uncertain durations
  - Actions with different costs
- Execution Monitoring, FDI
  - Is execution proceedings as expected?
  - Fault detection and identification

- Can be reduced to analysis of transition systems
- Planning as model checking paradigm

Life Cycle of Complex Systems



- Functional correctness
  - Does the system satisfy the requirements?
- Requirements validation:
  - Are the requirements flawed?
- Safety assessment
  - Is the system able to deal with faults?

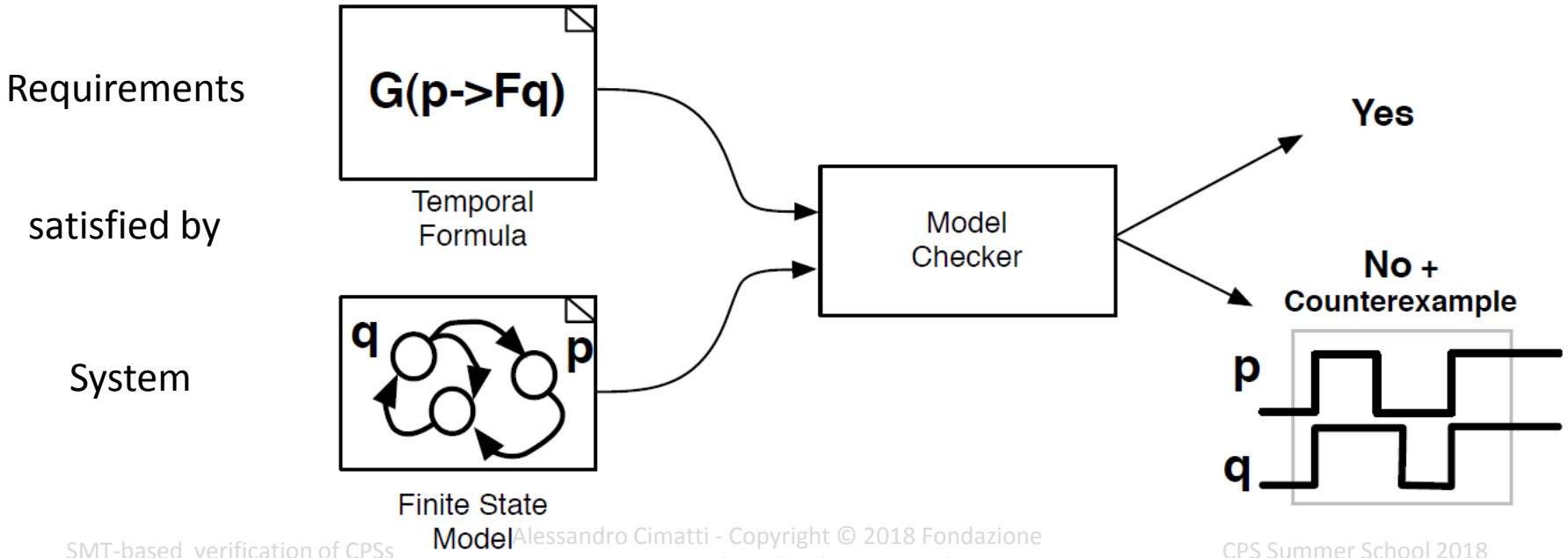
Formal verification

Formal methods

- From aeronautic standard DO-178C:
  - *"Descriptive notations and analytical methods used to construct, develop and reason about mathematical models of system behavior. A formal method is a formal analysis carried out on a formal model"*
- Increasing interests from industry
- Can be used to support or replace classical methods
- Examples of formal methods:
  - Model checking
  - Theorem proving
  - Abstract interpretation
  - …

“Old-fashioned” Model Checking

- Does system satisfy requirements?
- System as finite state model
- Requirements as temporal properties



The three main challenges in Formal Verification

- Scalability
- Scalability
- Scalability

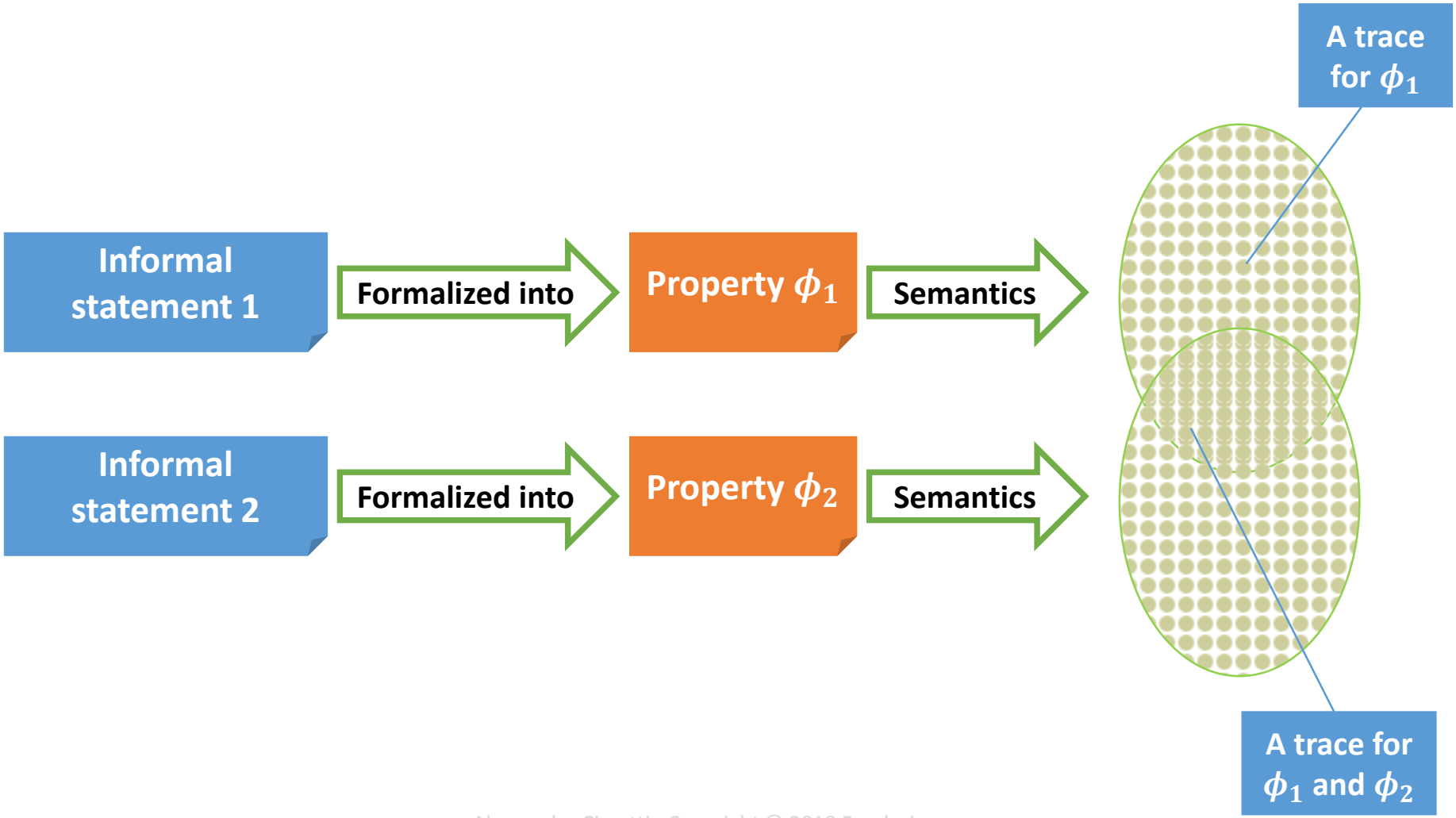
The ability to analyze large models automatically

Properties

- Properties are expressions in a mathematical logic using symbols of the system description.
- Used to formalize requirements.
  - Often closer to informal than behavioral descriptions
- Each property associated with set of system's behavior.

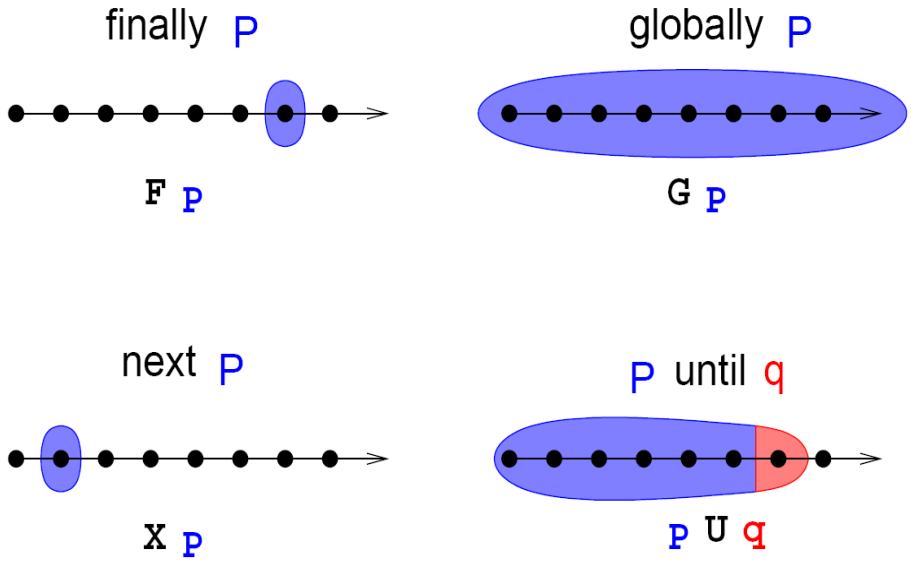
- Problems:
  - Specification: define the properties of a system.
  - Verification: check if the system satisfies the properties.
  - Validation: check if we are considering the right properties.
  - Synthesis: construct a system that satisfies the properties.

Properties, traces, and logic



Linear Temporal Logic

- Linear models
  - Traces as sequences of states
- Built over atomic propositions
- Using Boolean connectives
- And temporal operators



LTL examples

- Gp
  - “always p” – invariant
- G(p $\rightarrow$ Fq)
  - “p is always followed by q” – reaction
- G(p $\rightarrow$ Xq)
  - “whenever p holds, q is set to true in the next cycle”
  - immediate reaction
- GFp
  - “infinitely many times p” – fairness
- FGp
  - “eventually permanently p”
- G(p $\rightarrow$ (q U r))

Model Checking

- Model as transition system
  - Set of variables V
  - Initial states I(V)
  - Transition relation T(V,V')
- Ensuring the design is correct
  - Traces of model are subset of "good" traces

$$M \models \varphi$$

Models – where do they come from?

- Models are directly extracted from design languages
- Verilog, VHDL
- AADL, SysML, UML
- Altarica
- C
- Proprietary languages

Many levels of expressiveness

- Finite state transition systems
- Infinite state transition systems

- Timed automata
- Hybrid automata

- Software
- Concurrent software

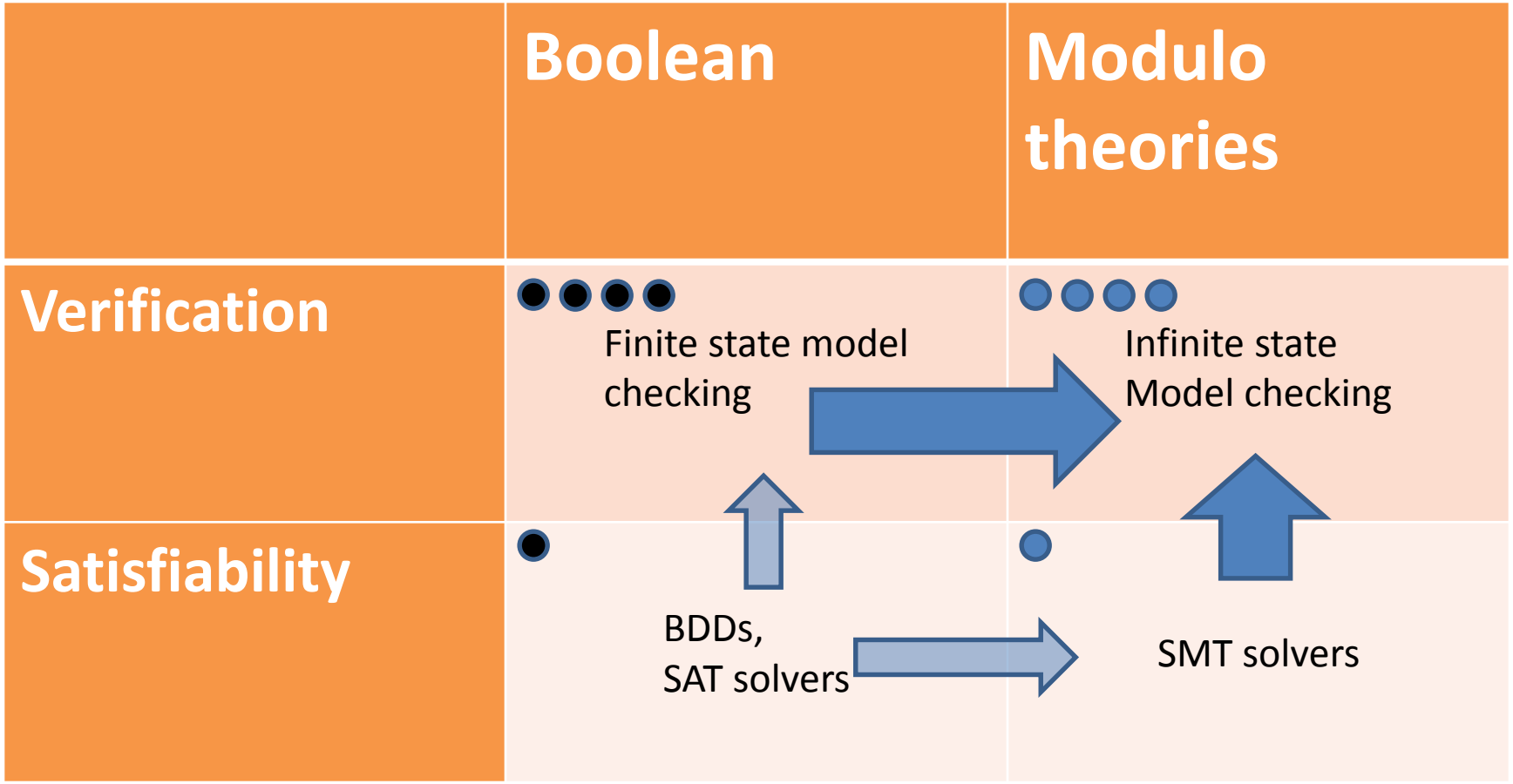
- Closed-loop software + hybrid plant

Formal verification engines

- From BDD-based engines…
  - Fix-point computation
- to SAT-based engines
  - Bounded model checking, induction, interpolation, IC3
- SMT: SAT + decision procedures
- Verification Modulo Theories
  - From finite-state…
    - Circuits, microcode
  - To infinite-state
    - Software, timed systems, hybrid systems, closed loop

Satisfiability vs Verification

(or, combinational vs sequential)



A “modern” view of FM

- Functional verification
- Safety analysis
  - Construct fault trees, FMEA tables
  - Timed Failure Propagation Graphs (TFPG)
- Contract-based design
  - Delegation of top-level requirements to subcomponents
  - Correctness by construction

- Tool chains:
  - COMPASS, CHESS
    - http://www.compass-toolset.org/, https://www.polarsys.org/chess/
  - nuXmv, xSAP, OCRA
    - http://nuxmv.fbk.eu/, http://xsap.fbk.eu, http://ocra.fbk.eu
- Applications:
  - AIR 6110 wheel brake system (https://es-static.fbk.eu/projects/air6110/)
  - NASA nextgen function allocation (https://es-static.fbk.eu/projects/nasa-aac/)

Dealing with faults

Beyond Model Checking

- Application of formal methods for design verification
  - ensuring the design is correct
    - Model-checking

$$M \models \varphi$$

NOT SUFFICIENT HERE

NEED TO ENSURE THE ROBUSTNESS AGAINST FAILURE CONDITIONS

Safety Assessment

- Safety Assessment
  - *"The safety assessment process provides a methodology to evaluate the design of systems, and to determine that the associated hazards have been properly addressed."*
  - In Aeronautics, process described in standards:
    - ARP4754A: Guidelines for Development of Civil Aircraft and Systems
    - ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

Safety Assessment

1. Fault extension

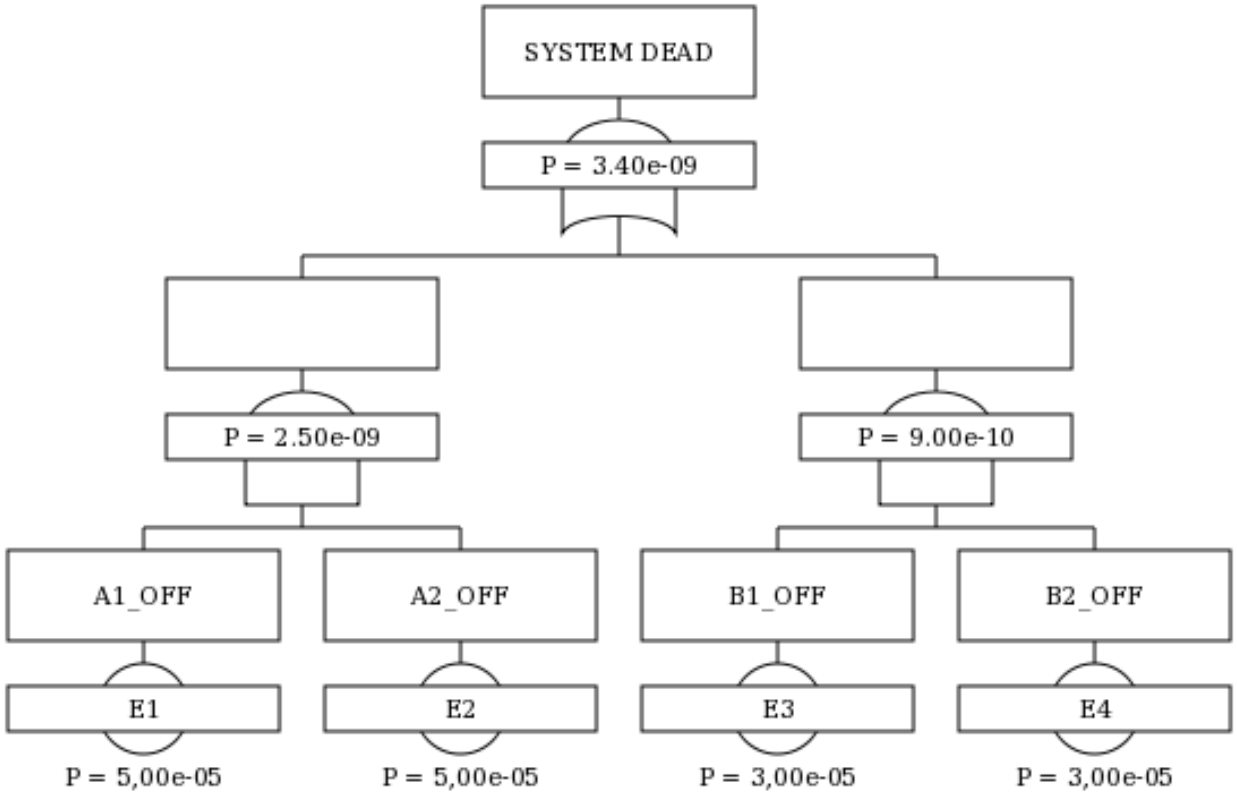
$$M \rightsquigarrow M_{[F]}$$

2. Model-Based Safety Assessment

$$\delta(F) : M_{[F]} \not\models \varphi$$

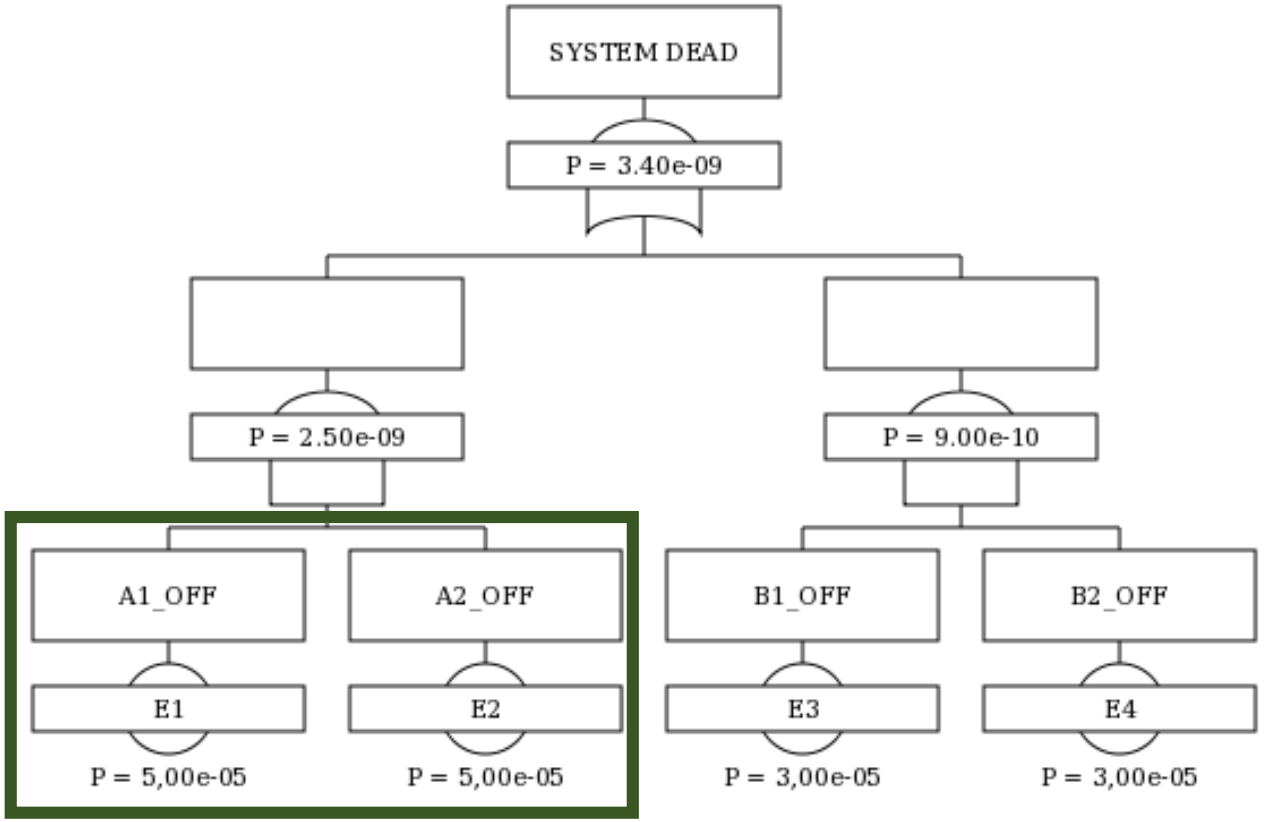
Safety Assessment

$\delta(F)$:



Safety Assessment

$\delta(F)$:



Minimal Cut Set (MCS)

From Minimal Cut Sets to reliability

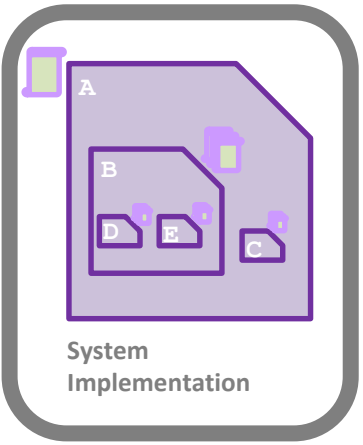
- Given a set of MCSs and a mapping P giving the probability for the basic faults, it is possible to compute the probability of the occurrence of the top-level event.
- Assumption: basic faults are independent.
- The probability of a single MCS σ is given by the product of the probabilities of its basic faults:

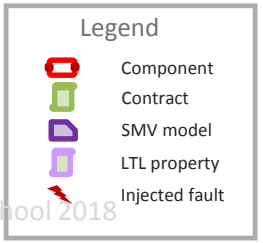
$$P(\sigma) = \prod_{f_i \in \sigma} P(f_i)$$

- For a set of MCSs S, the probability can be computed using the above and the following recursive formula:

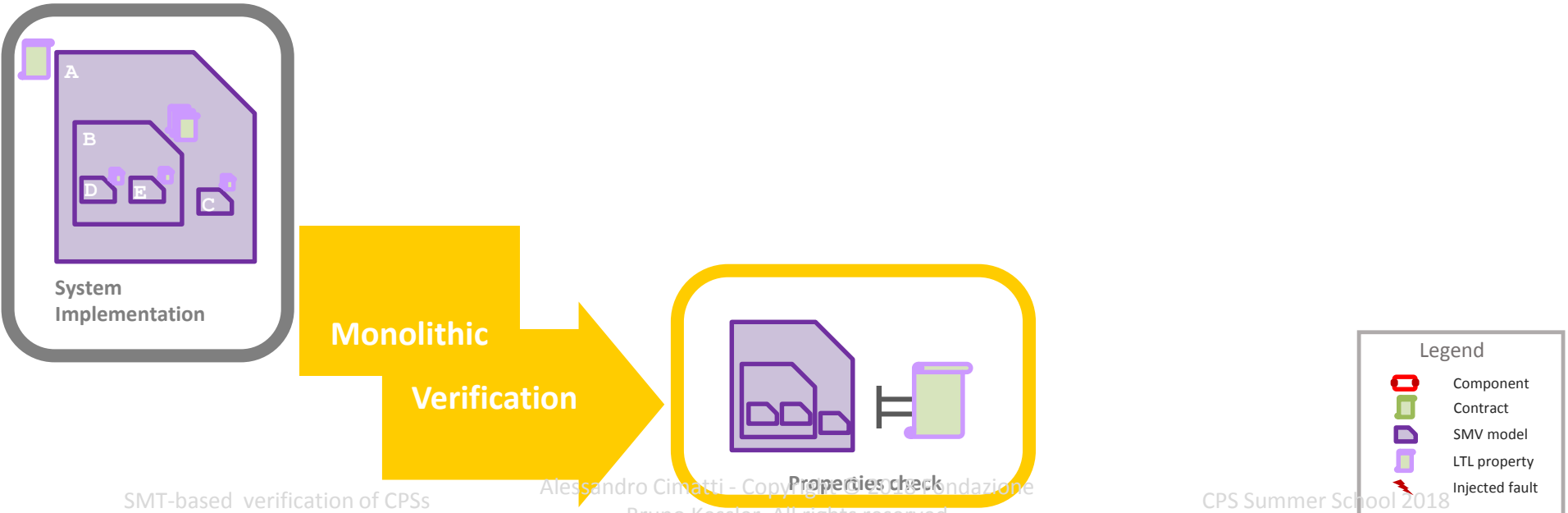
$$P(S_1 \cup S_2) = P(S_1) + P(S_2) - P(S_1 \cap S_2)$$

Monolithic Workflow

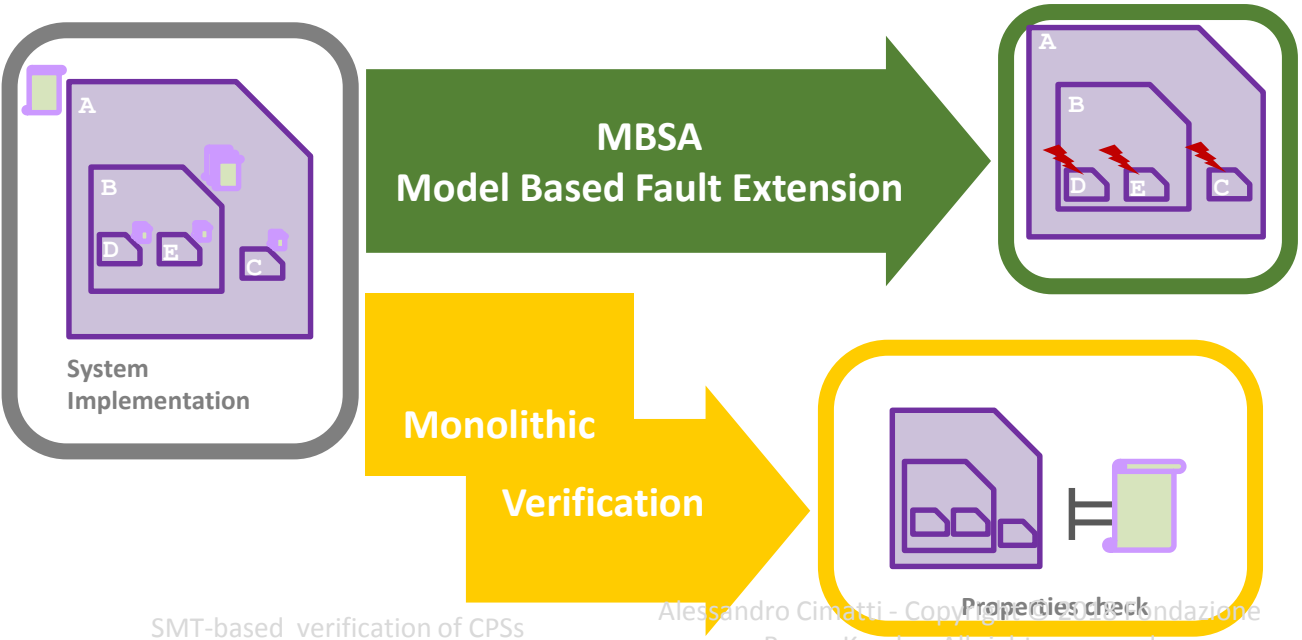




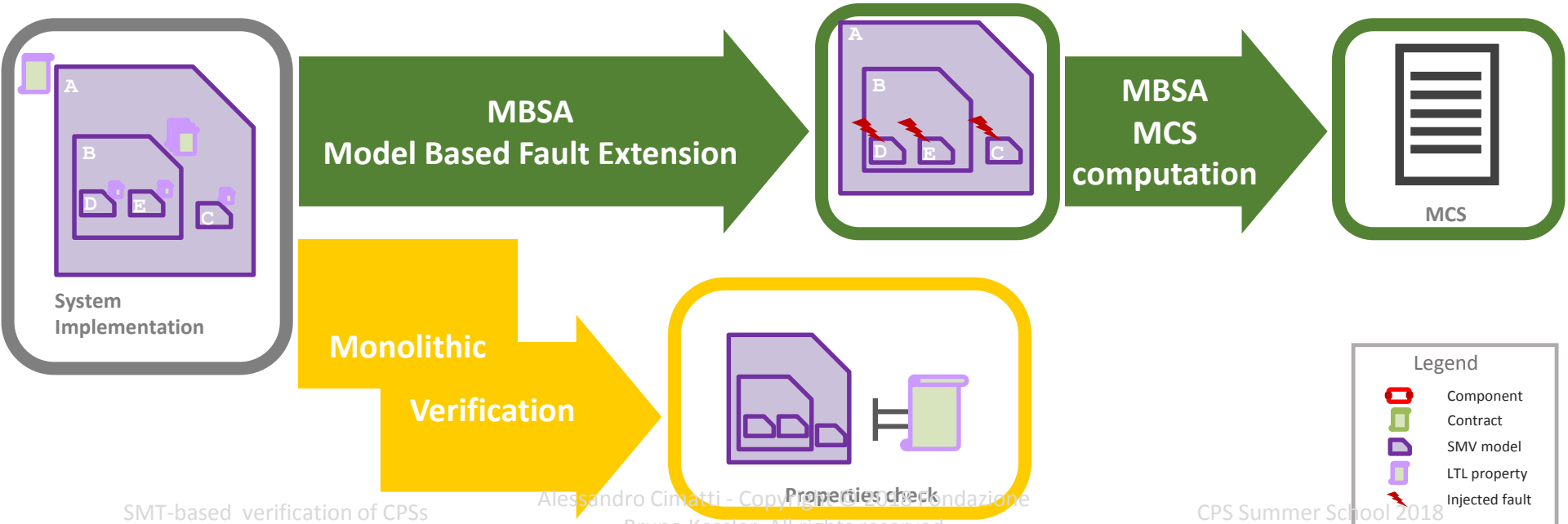
Monolithic Workflow



Monolithic Workflow



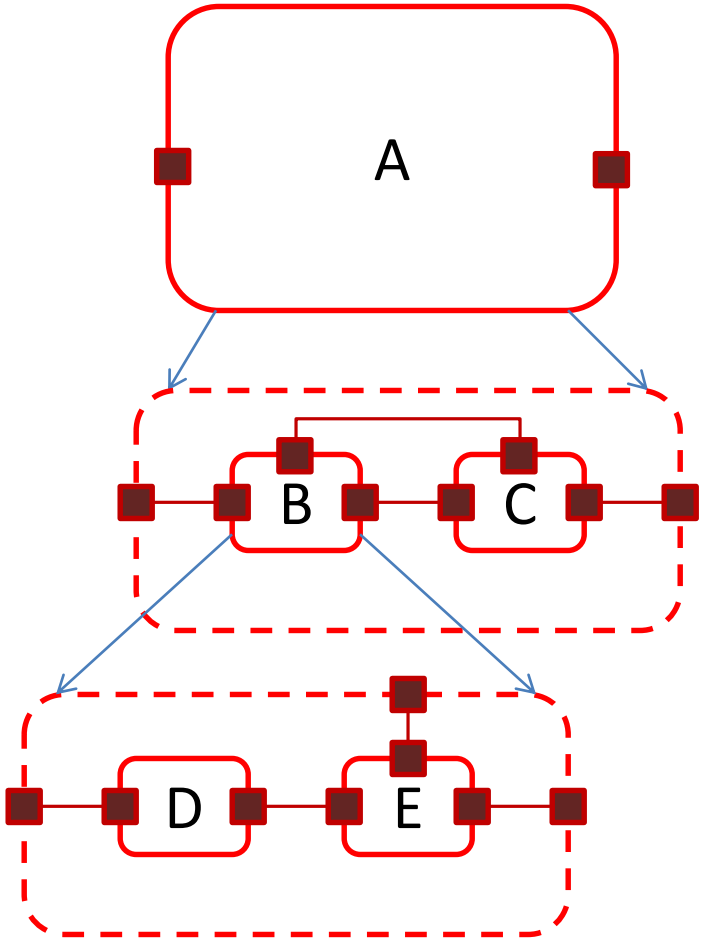
Monolithic Workflow



Requirements decomposition

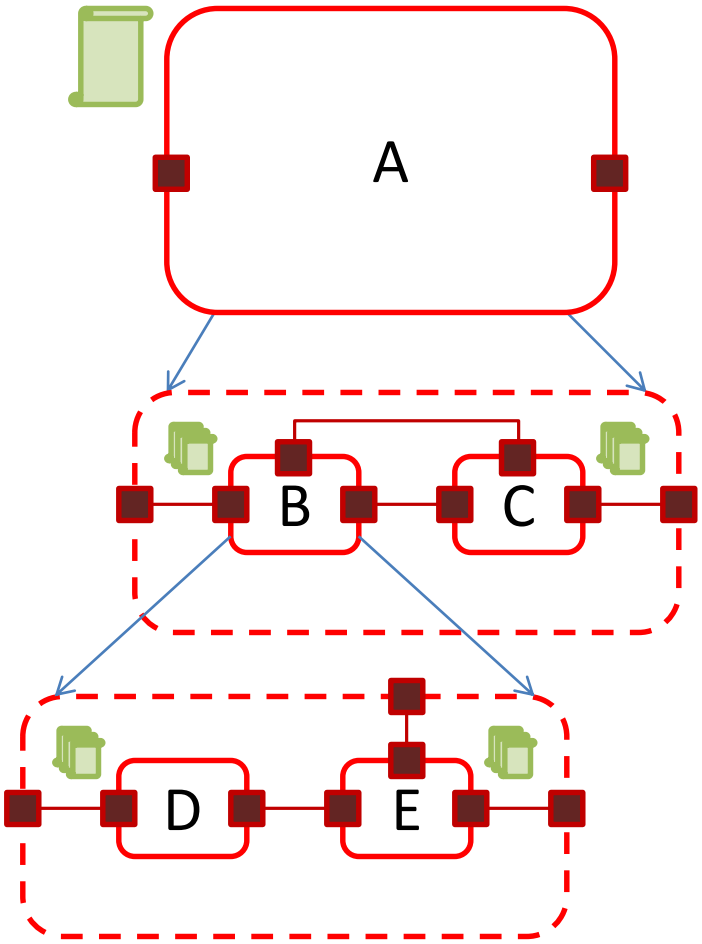
Contract-based Design

- Hierarchical decomposition
- Ex:
  - System A
  - System A decomposed into subsystems B and C
  - Subsystem B decomposed into equipments D and E
  - Hierarchical decomposition preserves ports



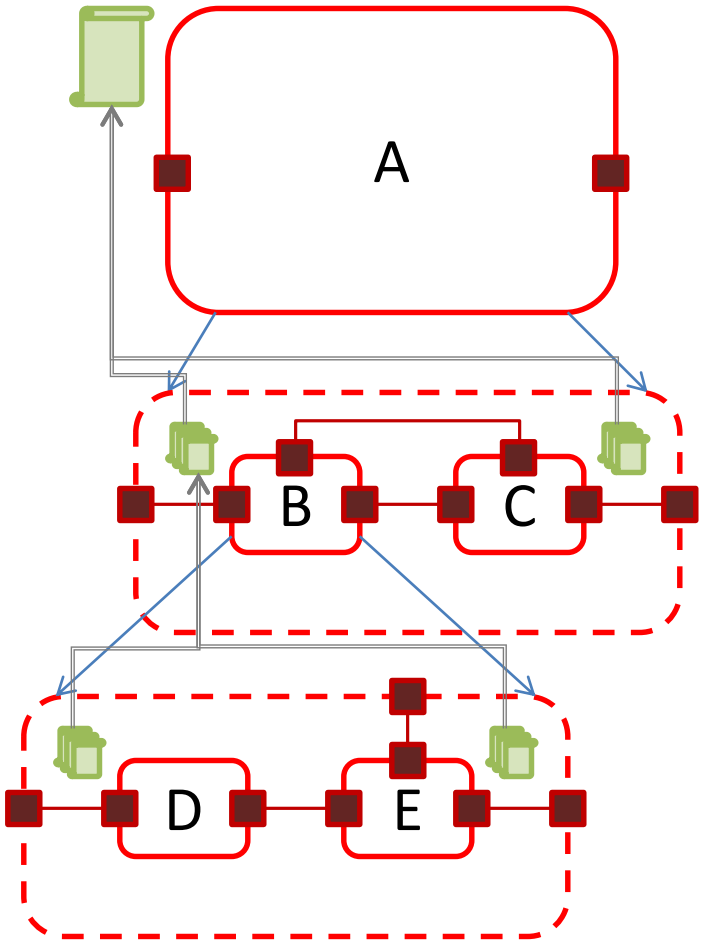
Contract-based Design

- A component is immersed in an environment
- Its behavior is specified by contracts
- Contract: assumptions + guarantees
  - Assumptions: what the environment of the component is supposed to do
  - Guarantee: what the component shall do



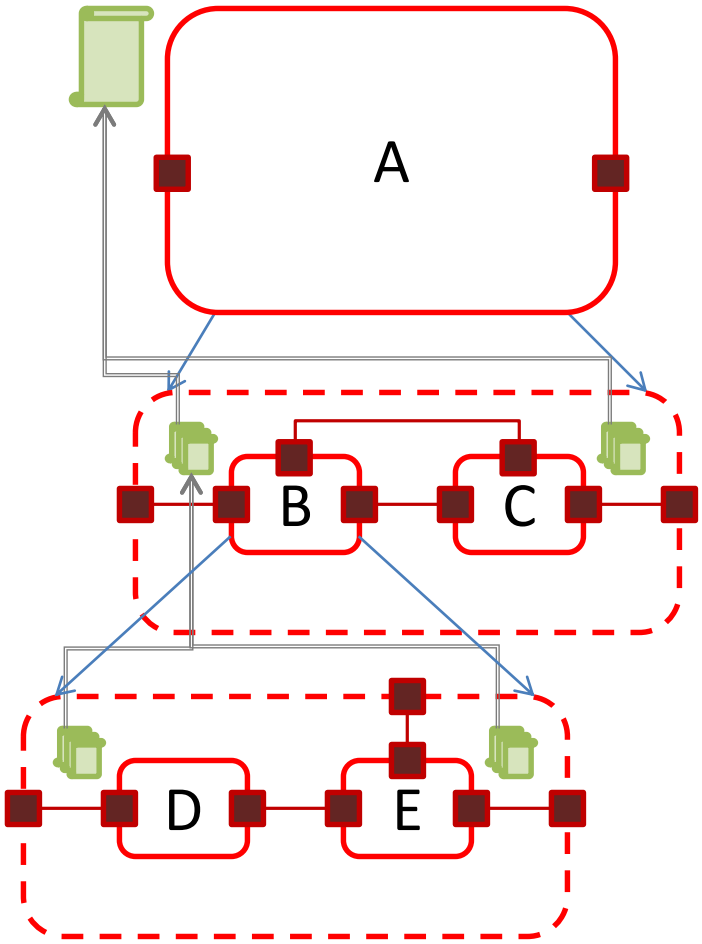
Contract-based Design

- Specify components while designing
  - decomposing the specification based on the decomposition of the architecture
- Ensure the correctness of the decomposition
  - Does the contract of A follow from the contracts of B and C?



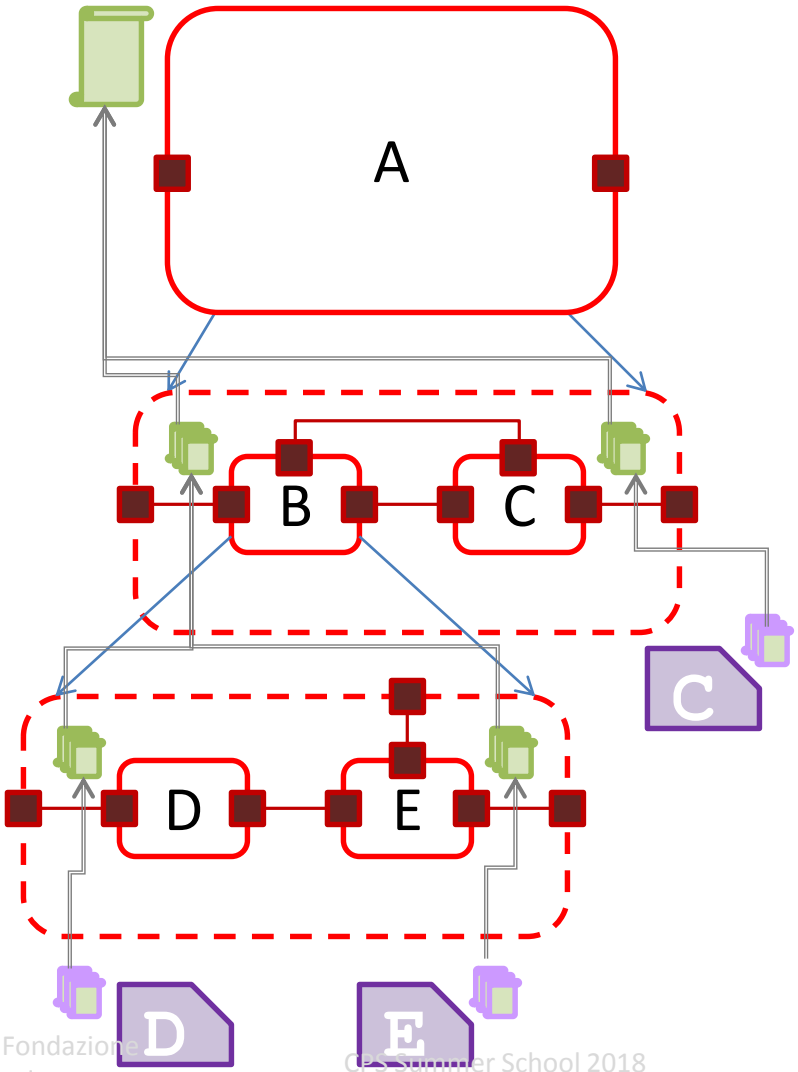
Contract-based Design

- A formal language to specify contracts
  - Temporal logics
- A framework for correct contract refinement
  - Proof obligations
  - Logical consequence of temporal logic formulae



Model Checking

- A formal language to specify implementation
  - Finite state machines
- Checking implementation
  - **Model checking**



Contracts

- Properties of the component and its environment.
  - Can be seen as assertions for component interfaces.
- Contracts used to characterize the correctness of component implementations and environments.
- Typically, properties for model checking have a “fully observable” view of the system internals.
- For components instead:
  - Limited to component interfaces.
  - Structure into assumptions and guarantees.
- Contracts for OO programming are pre-/post-conditions [Meyer, 82].
- For systems, assumptions correspond to pre-conditions, guarantees correspond to post-conditions.

Refinement: proof obligations

- Given C=<A,G> contract for component
- Given C_1=<A_1,G_1>, …, <A_n,G_n> contracts for subcomponents
- Proof obligations for "{ C_i } refines C":
  - $\{(A_1 \rightarrow G_1), \ldots, (A_n \rightarrow G_n)\} \models A \rightarrow G$
  - $\{(A_2 \rightarrow G_2), \ldots, (A_n \rightarrow G_n)\} \models A \rightarrow A_1$
  - …
  - $\{(A_1 \rightarrow G_1), \ldots, (A_{i-1} \rightarrow G_{i-1}), (A_{i+1} \rightarrow G_{i+1}), \ldots, (A_n \rightarrow G_n)\} \models A \rightarrow A_i$
  - …
  - $\{(A_1 \rightarrow G_1), \ldots, (A_{n-1} \rightarrow G_{n-1})\} \models A \rightarrow A_n$

What does it mean?

- Focus on properties of father component
- $\{(A_1 \rightarrow G_1), \ldots, (A_n \rightarrow G_n)\} \models A \rightarrow G$
- The contract of the father component $A \rightarrow G$ must follow from the contracts of the subcomponents

- Alternative view:
  $\{(A_1 \rightarrow G_1), \ldots, (A_n \rightarrow G_n), A\} \models G$

What does it mean?

- Focus on i-th subcomponent
- $\{(A_1 \rightarrow G_1)\ , \ldots, (A_{i-1} \rightarrow G_{i-1}),$
  $(A_{i+1} \rightarrow G_{i+1}), \ldots, (A_n \rightarrow G_n)\} \models A \rightarrow A_i$
- The assumptions of the i-th subcomponent must follow from the contracts of the other subcomponents plus the assumptions of the father component

Proof obligations

- PO's necessary and sufficient for correct contract refinement [CT12]
- Extension to deal with asynchronous composition

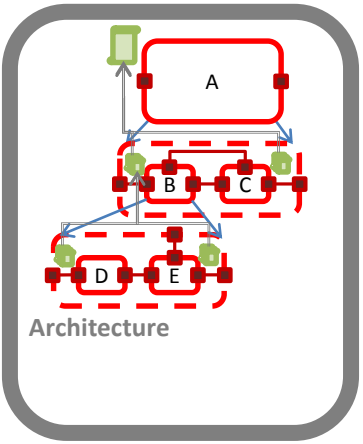
- Key issue: diagnostic information!
  - In case of violation, trace
  - Localization by means of proof-based methods
    - unsat core extraction

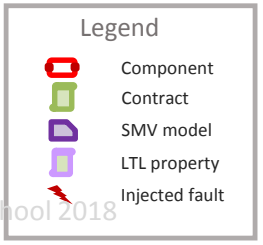
Weak vs. strong assumptions

- Weak vs. strong assumptions (both important):
  - Weak assumptions
    - Define the context in which the guarantee is ensured
    - As in assume-guarantee reasoning
    - Different assume-guarantee pairs may have inconsistent assumptions (if x>0 then …, if x<0 then …)
  - Strong assumptions
    - Define properties that must be satisfied by the environment.
    - Original idea of contract-based design.
    - If not satisfied, the environment can cause a failure (division by zero, out of power, collision).

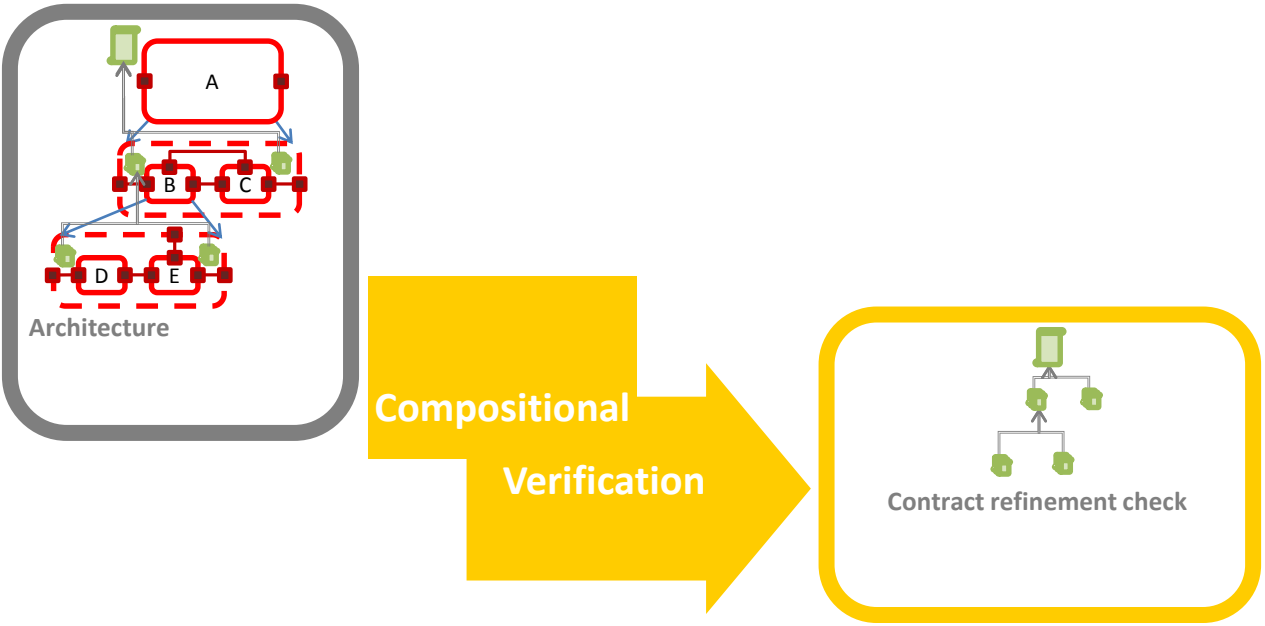
Overall Workflow

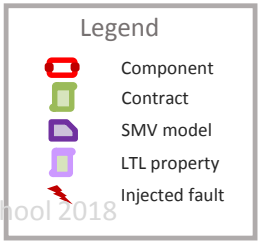
Overall workflow



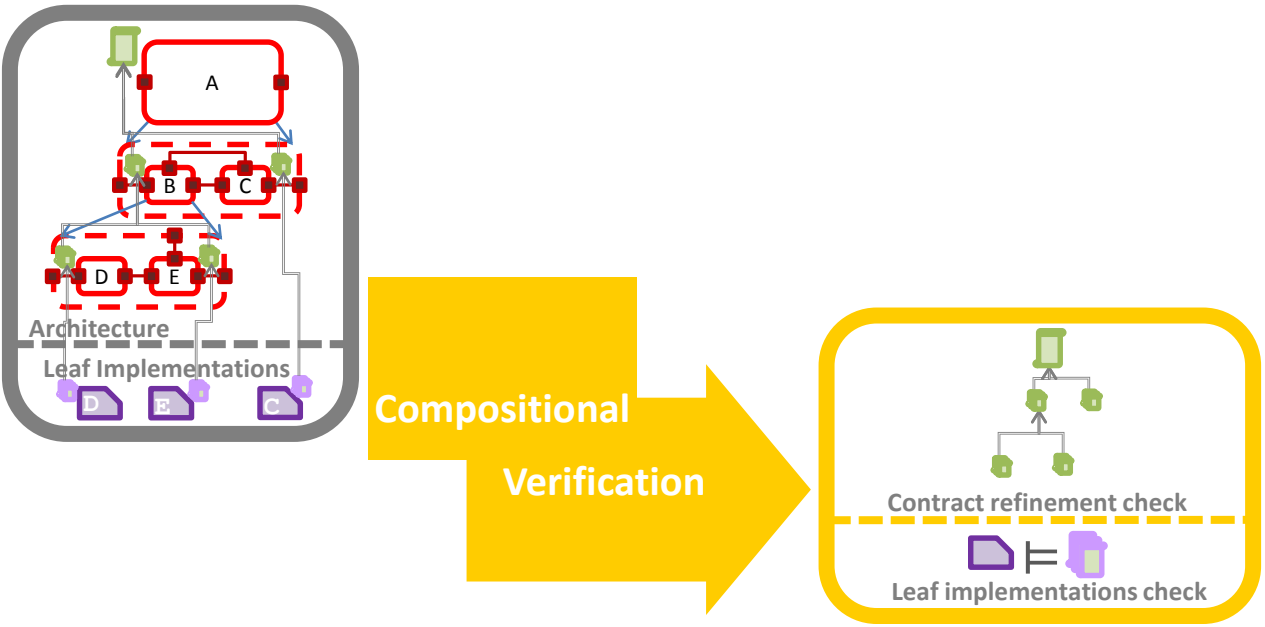


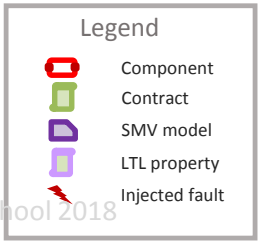
Overall workflow



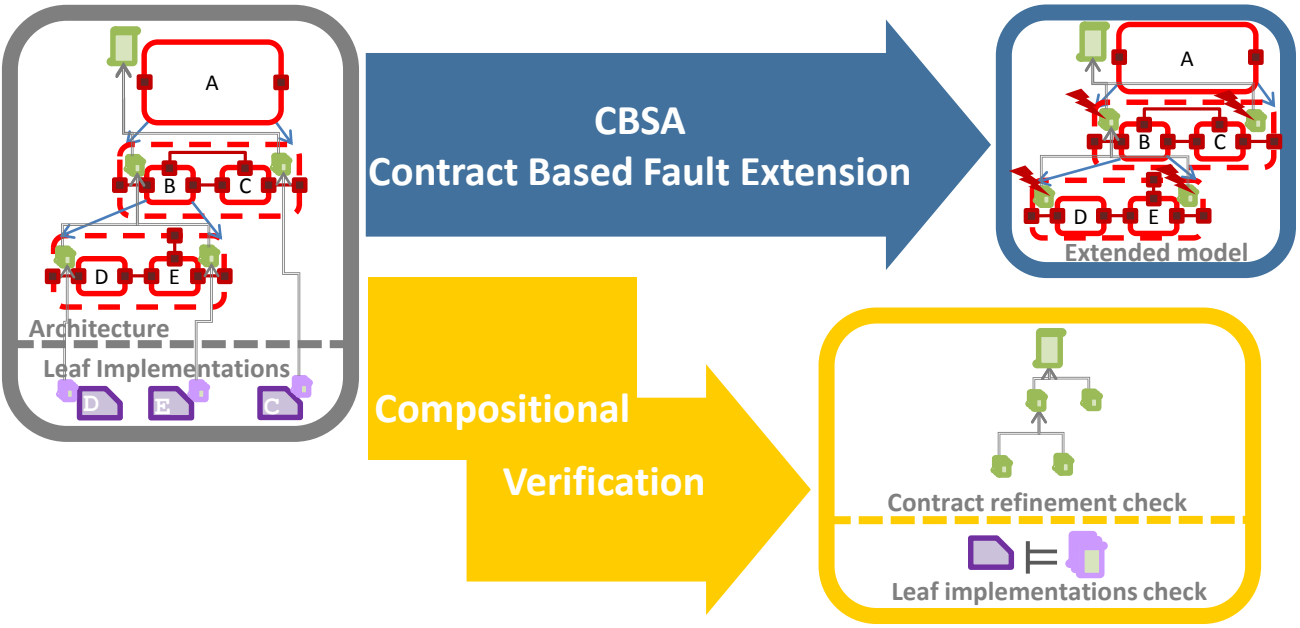


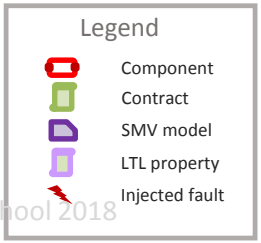
Overall workflow



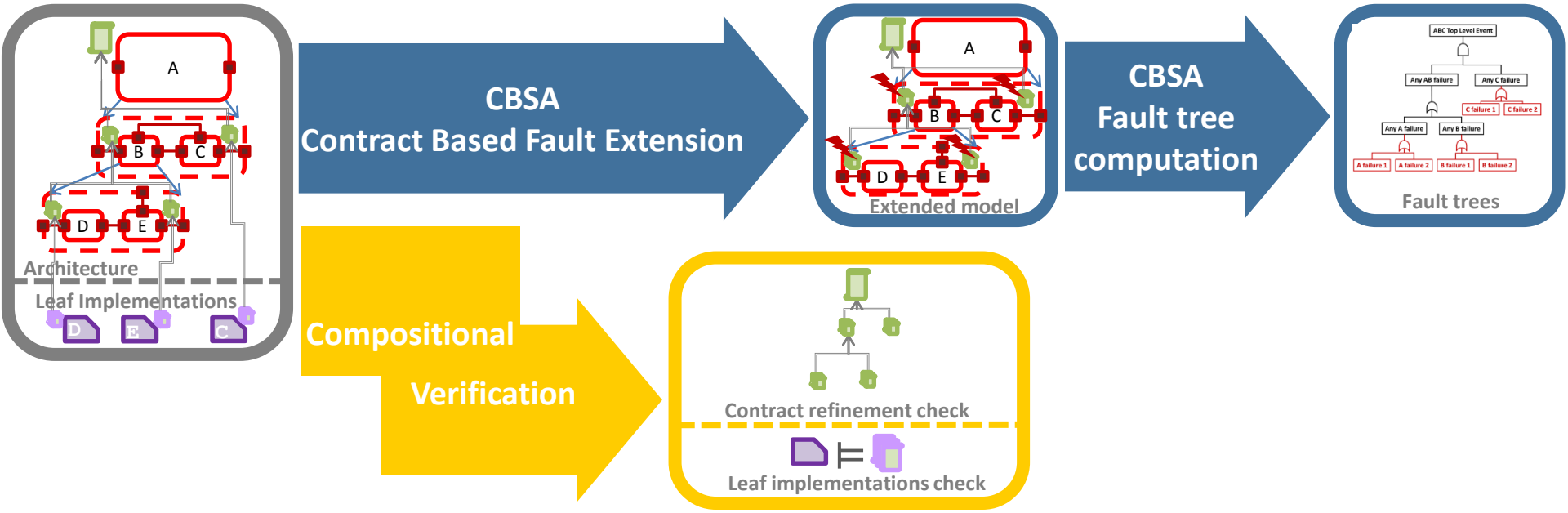


Overall workflow





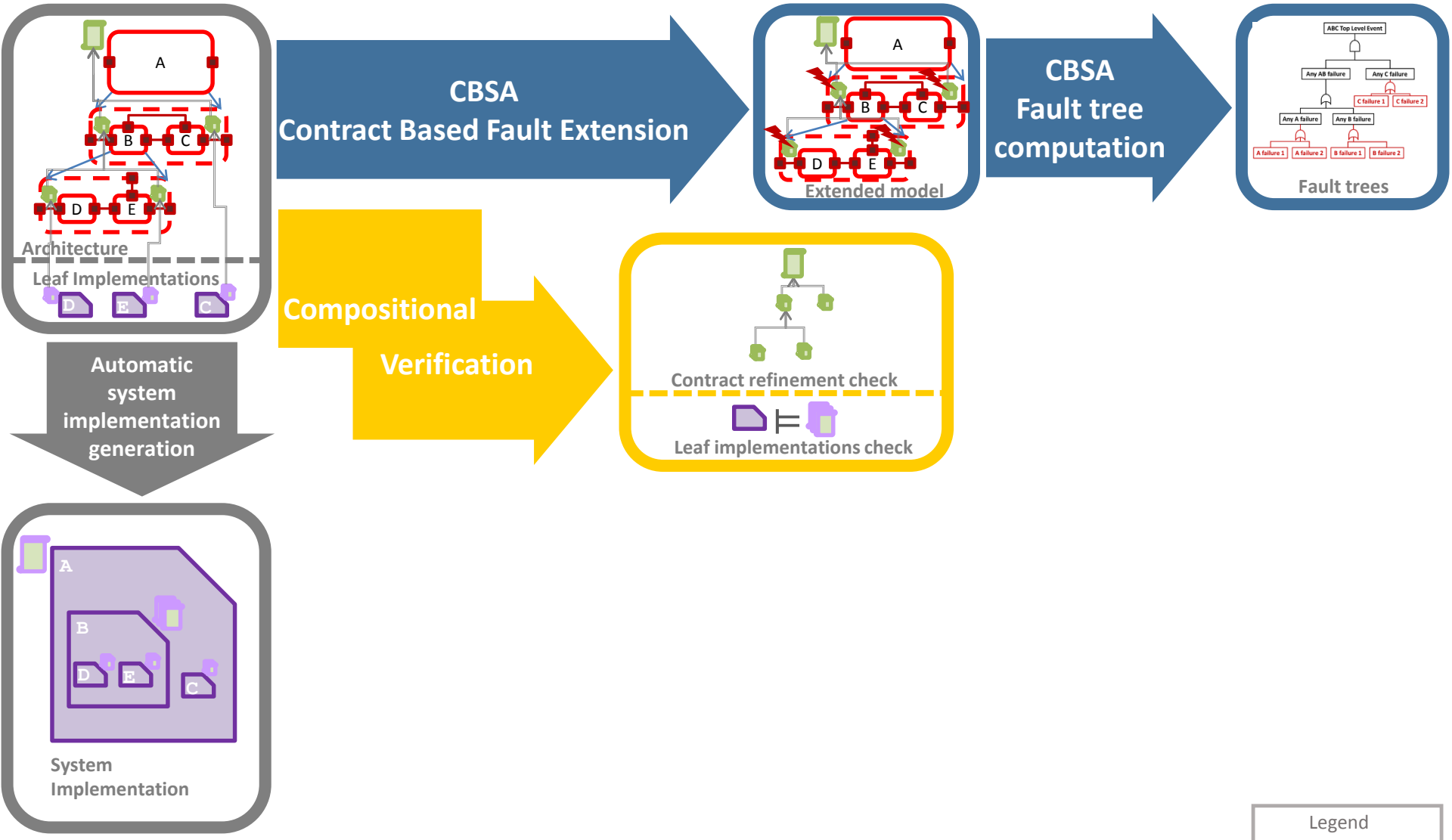
Overall workflow



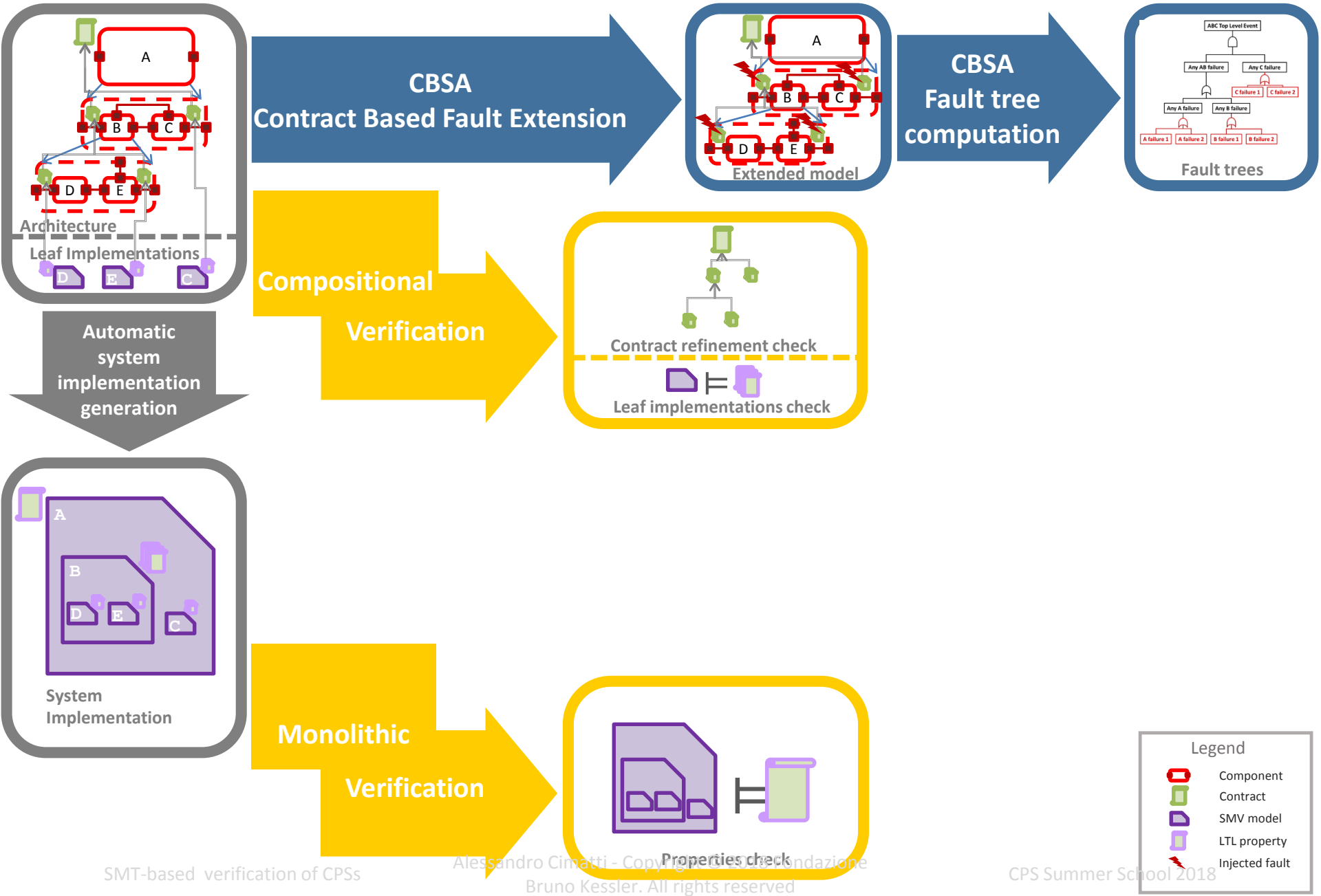
Legend

- Component
- Contract
- SMV model
- LTL property
- Injected fault

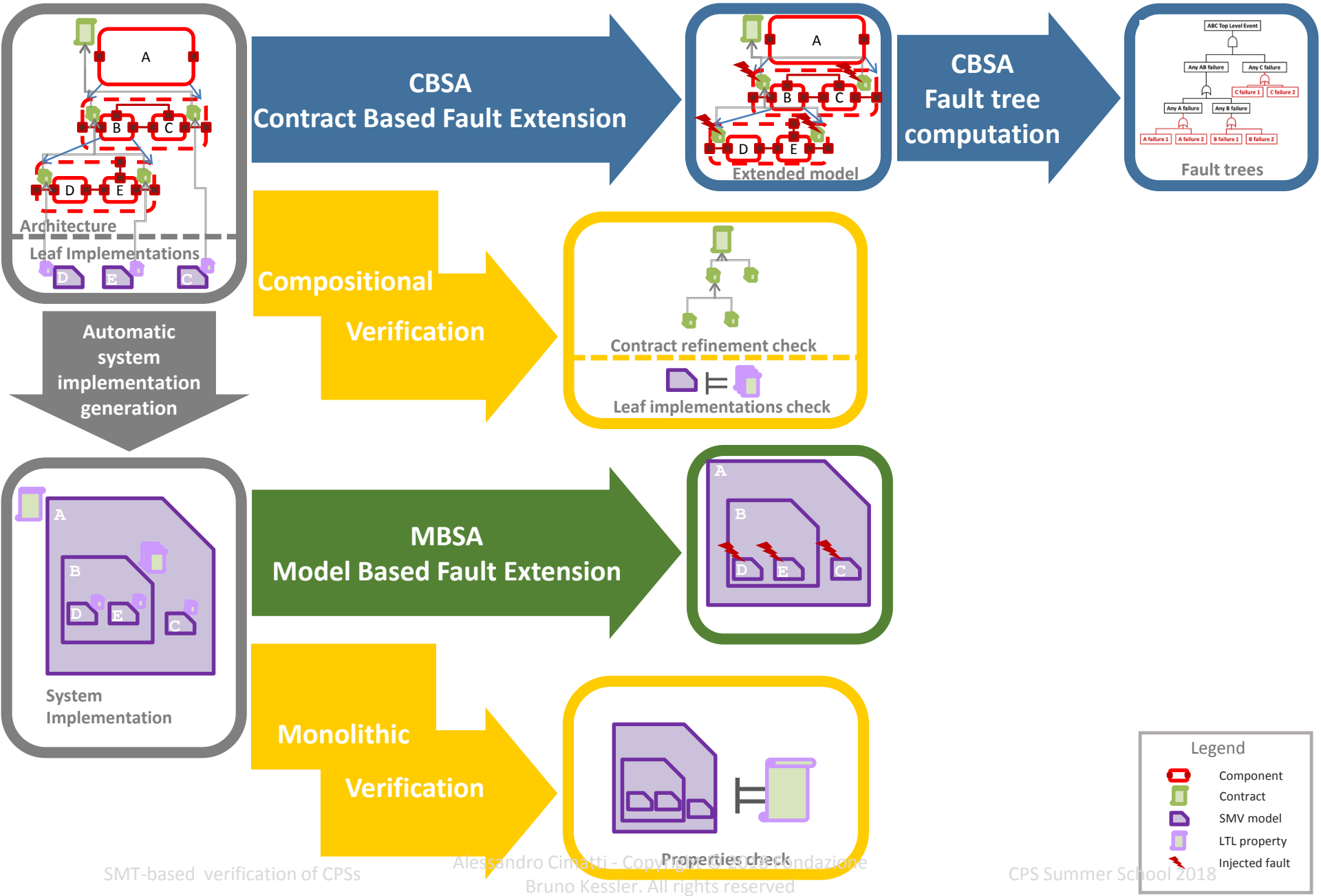
Overall workflow



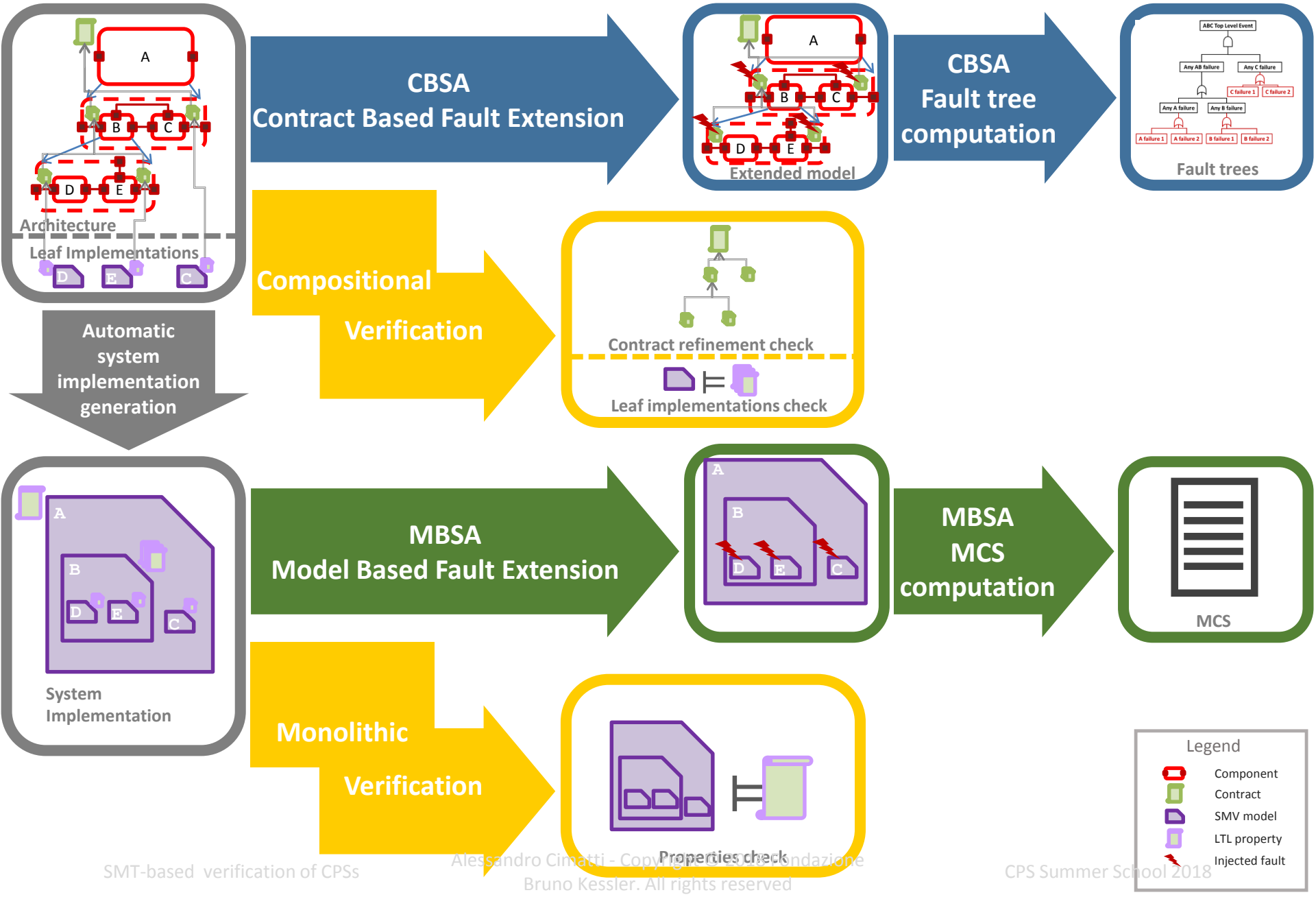
Overall workflow



Overall workflow



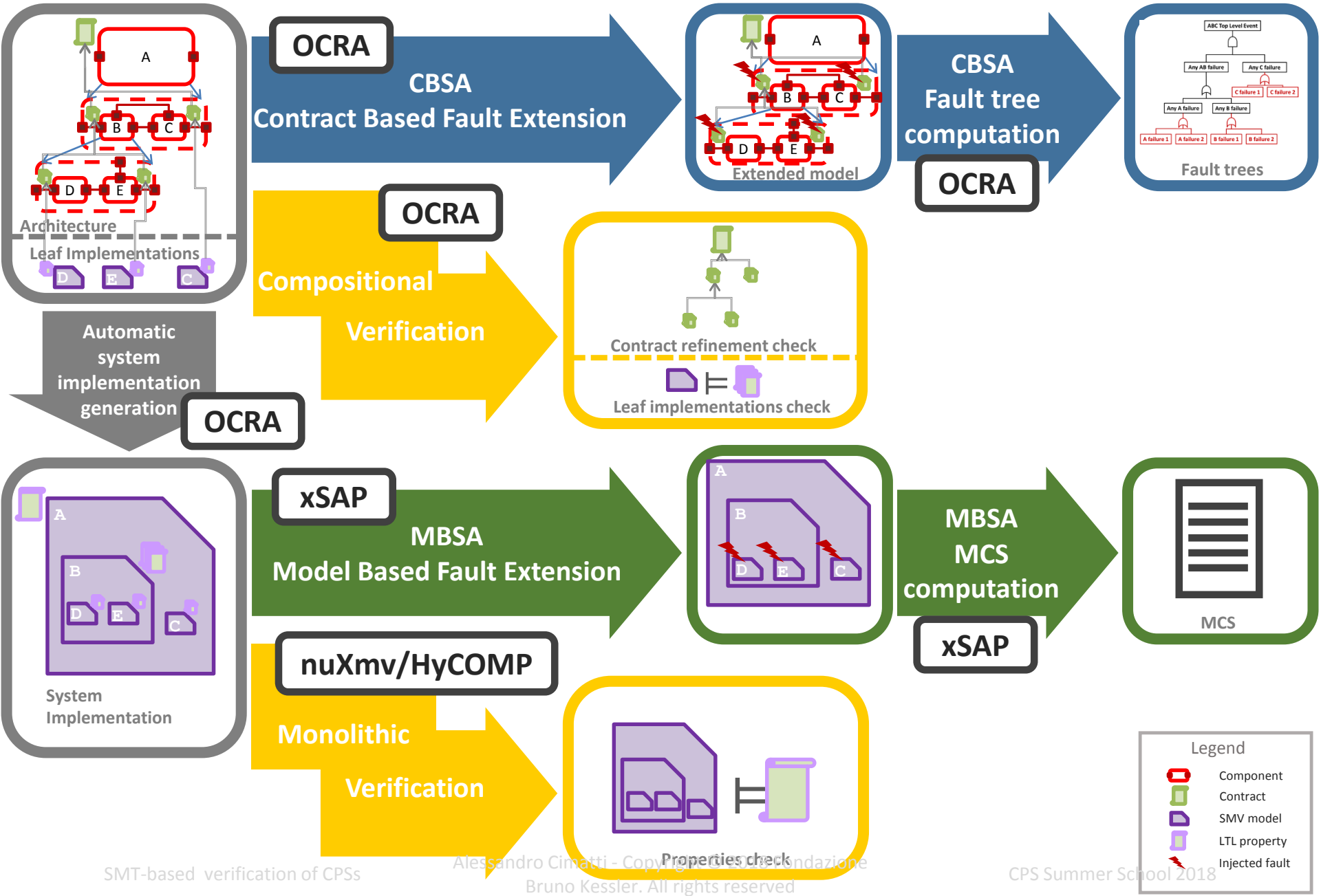
Overall workflow



Tool set

- nuXmv
  - model-checker for finite and infinite states systems in discrete time
- HyCOMP
  - model-checker for finite and infinite states systems in discrete time
- xSAP
  - Model-based Safety Analysis tool
- OCRA
  - Contract-based Design tool

Overall workflow



Boolean modeling

Wheel Brake System case study

Finite domain variables

Discrete time

AIR 6110 Wheel Brake System

- Aerospace Information Report 6110
  - Contiguous Aircraft/System Development Process Example
- Hypothetical dual-engine aircraft
  - 300-350 passengers
  - 5 hours of flight max
- Focus on the Wheel Brake System (WBS)
  - Braking function for the two main landing gears
    - 4-wheels landing gear
    - Independently controlled

AIR 6110 Wheel Brake System

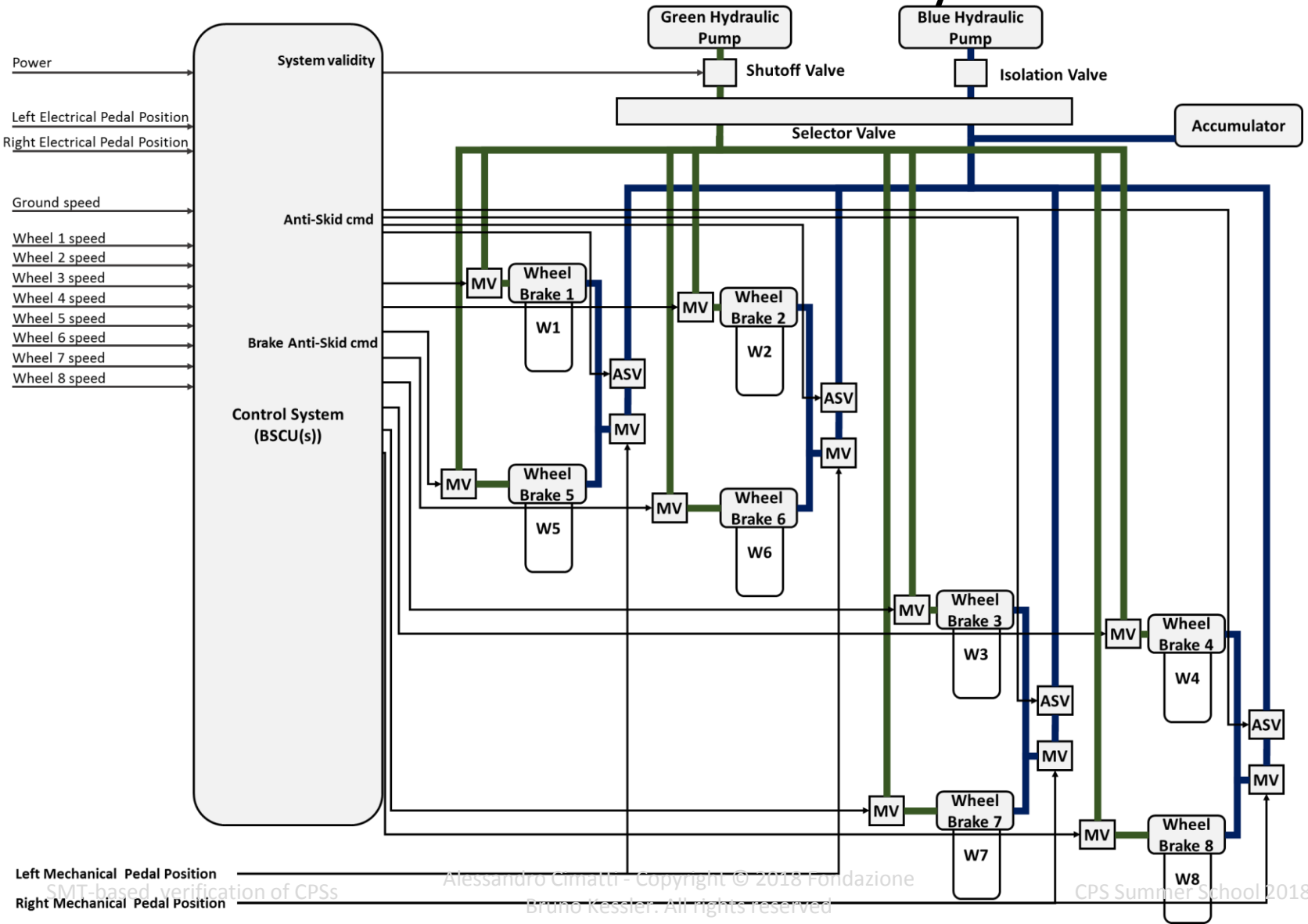
- Main features
  - Hydraulic brake electrically or mechanically controlled braking
  - Anti-skid function
  - Redundancy in the hydraulic and control system

AIR 6110 Wheel Brake System

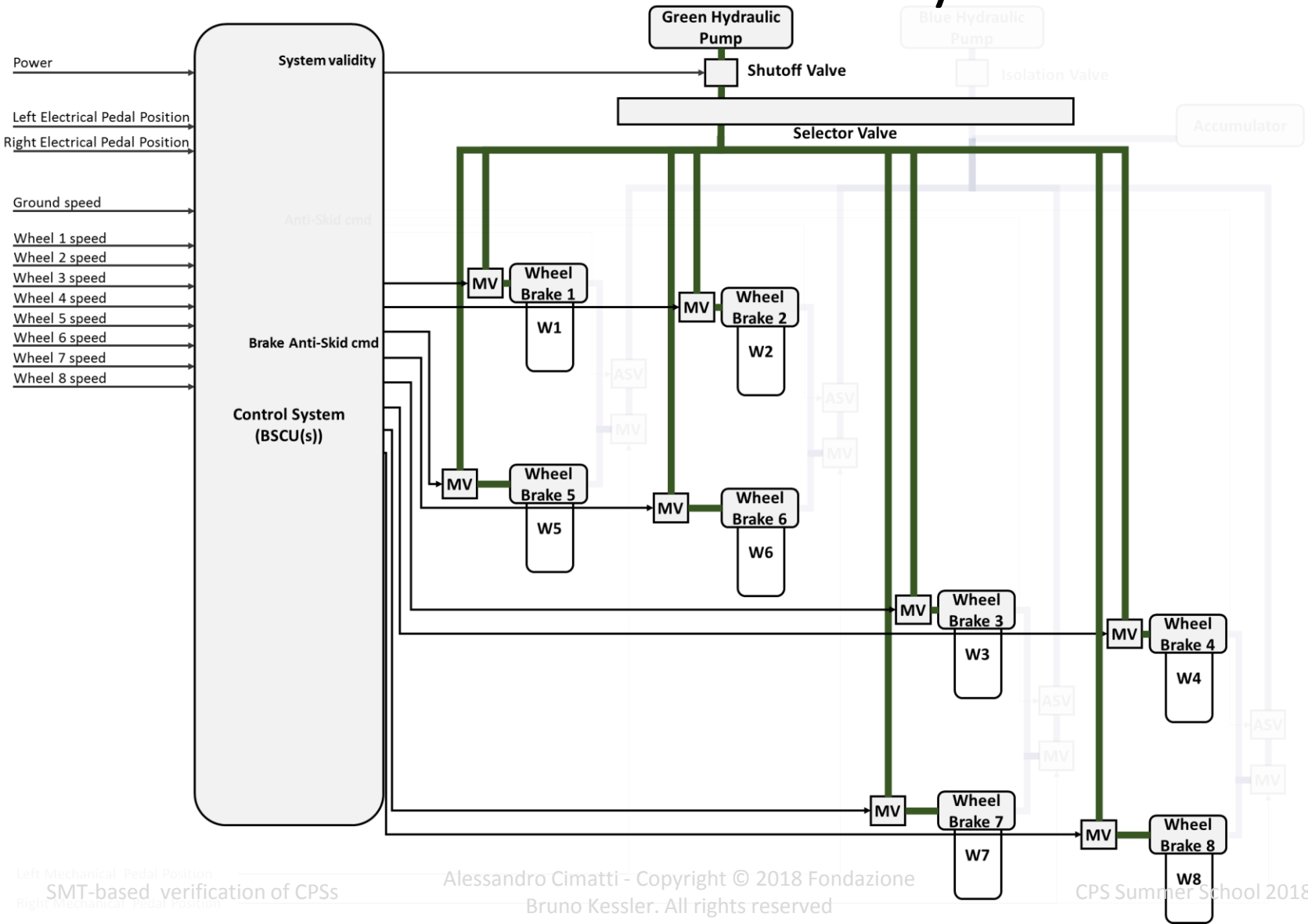
- Main features
  - Hydraulic brake electrically or mechanically controlled braking
  - Anti-skid function
  - Redundancy in the hydraulic and control system

		Wheel Brake System		
		Normal Mode *(Primary pressure source)*	**Alternate Mode** *(Secondary pressure source)*	**Emergency Mode** *(Finite-reserve accumulator)*
Control system	**Valid**	• $Brake_{Electrical}$ • Individual AntiSkid	• $Brake_{Mechanical}$ • Paired-AntiSkid	• $Brake_{Mechanical}$ • Paired-AntiSkid
	Invalid	N/A	$Brake_{Mechanical}$	$Brake_{Mechanical}$

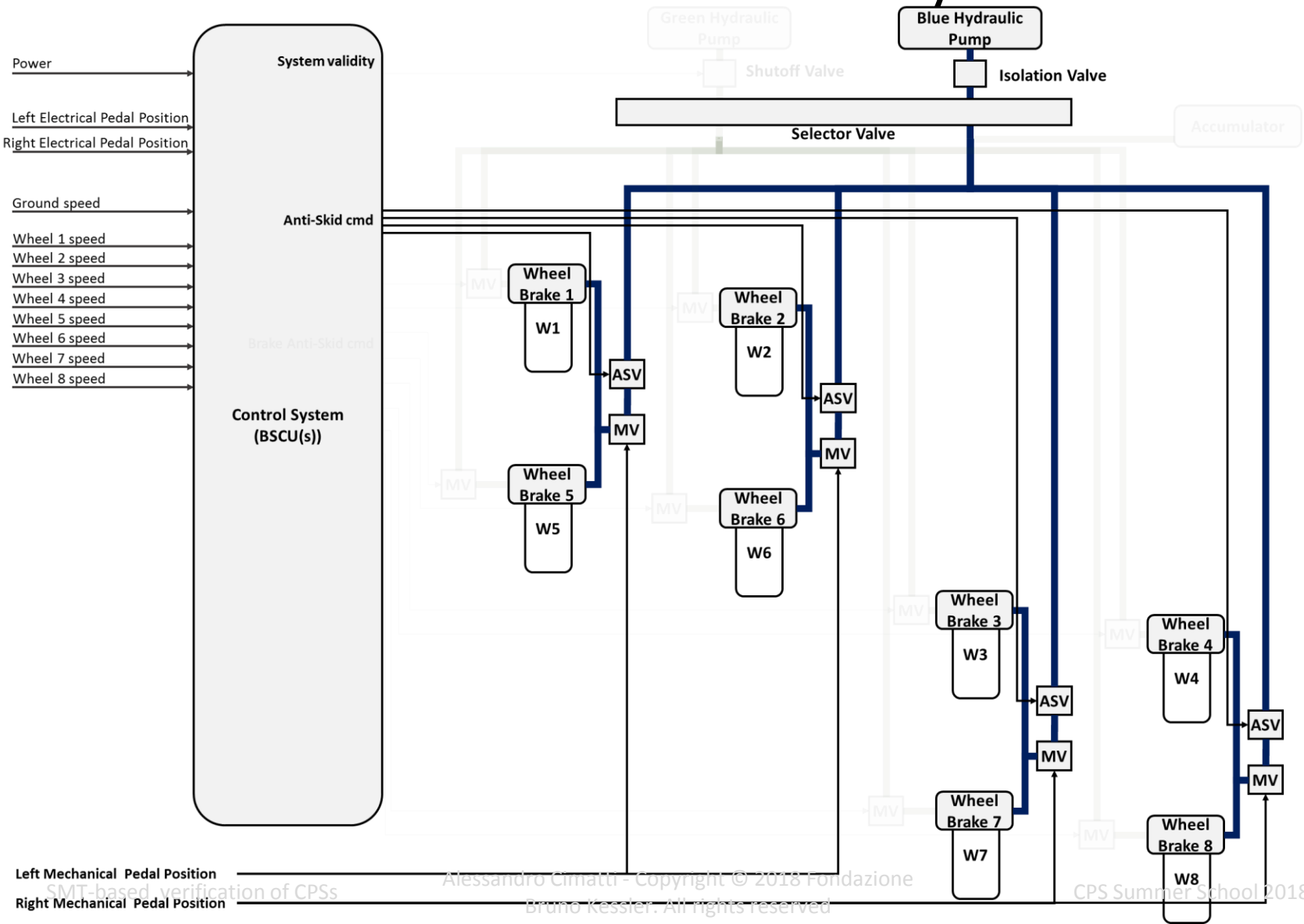
AIR 6110 Wheel Brake System



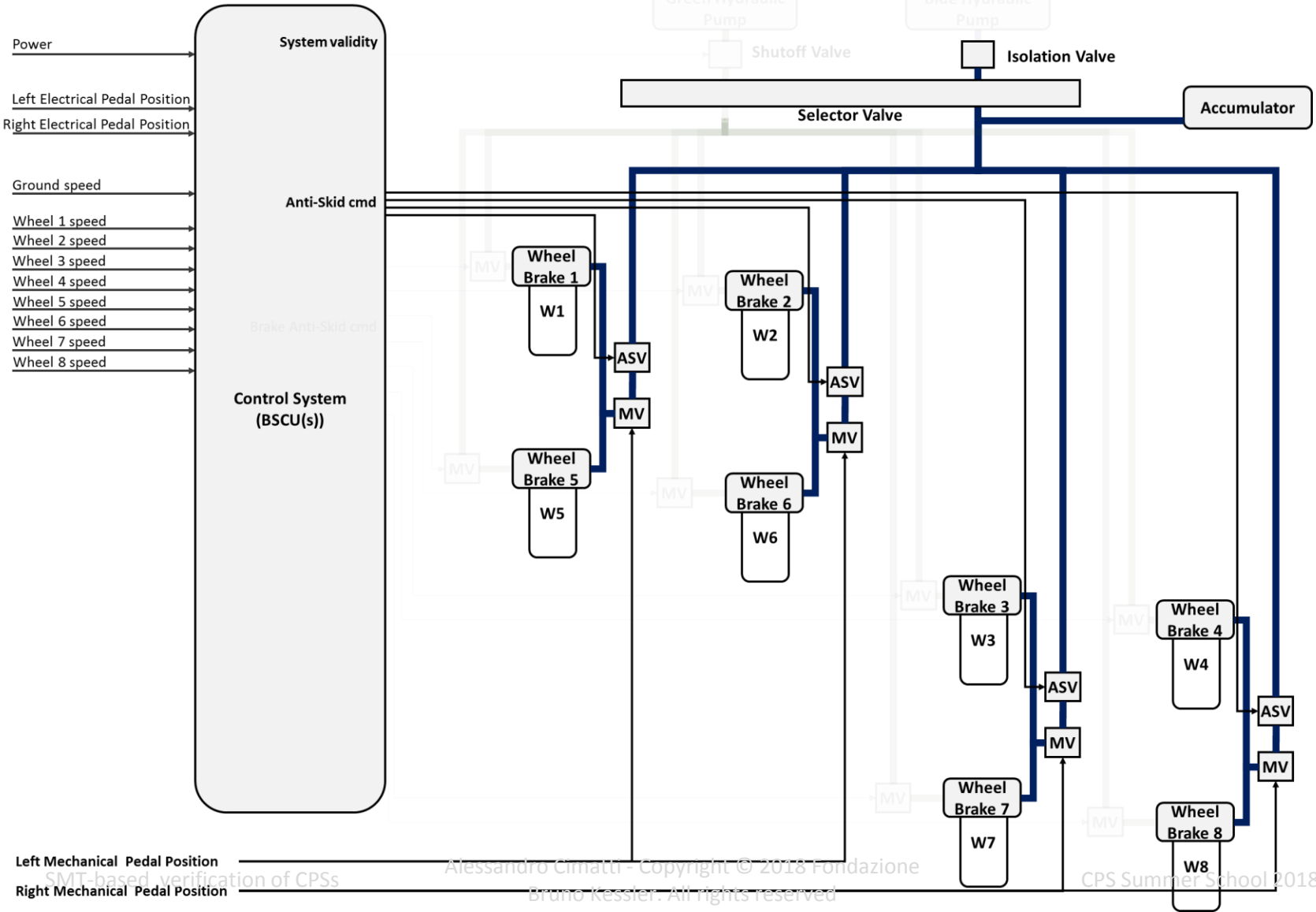
AIR 6110 Wheel Brake System



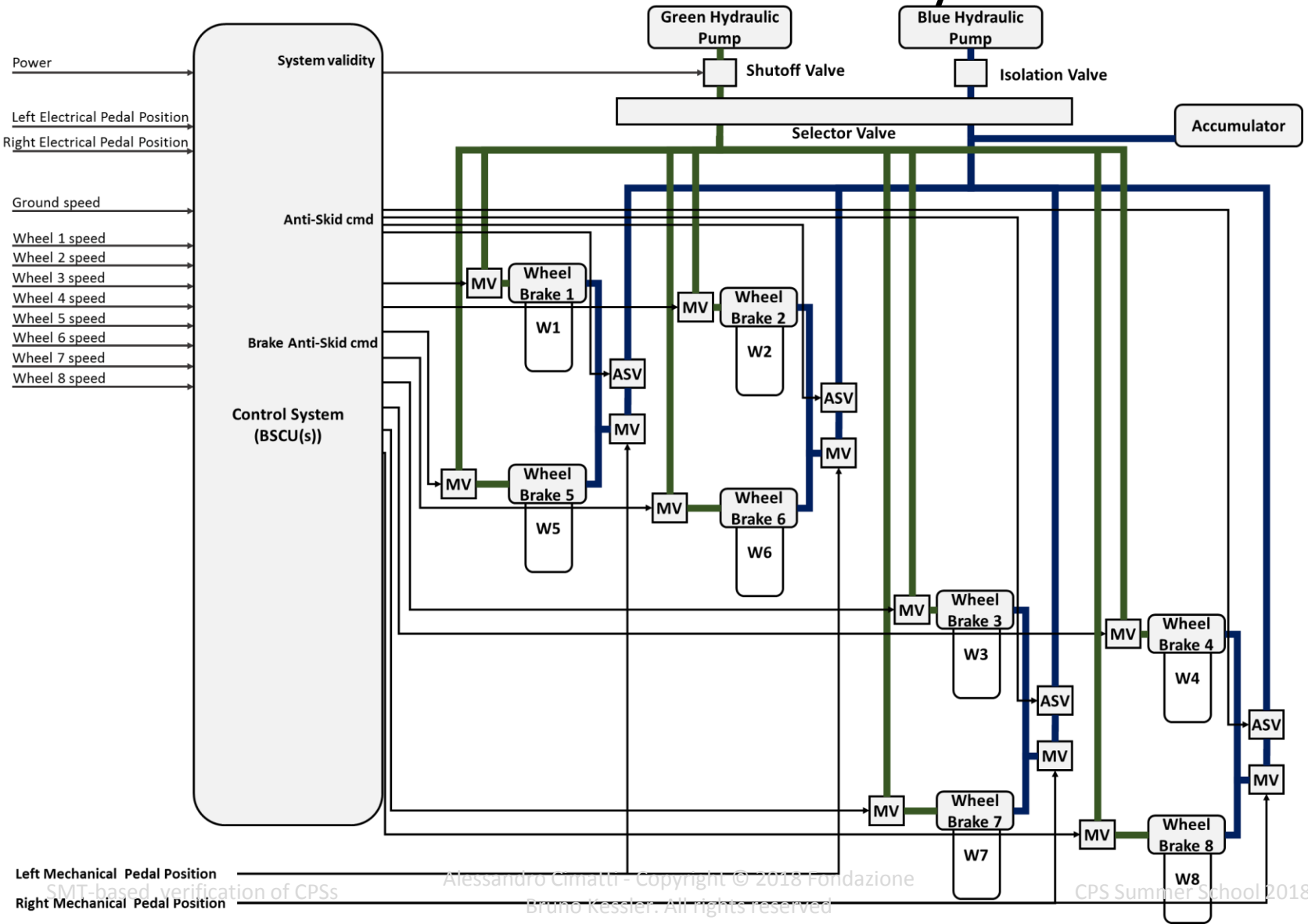
AIR 6110 Wheel Brake System



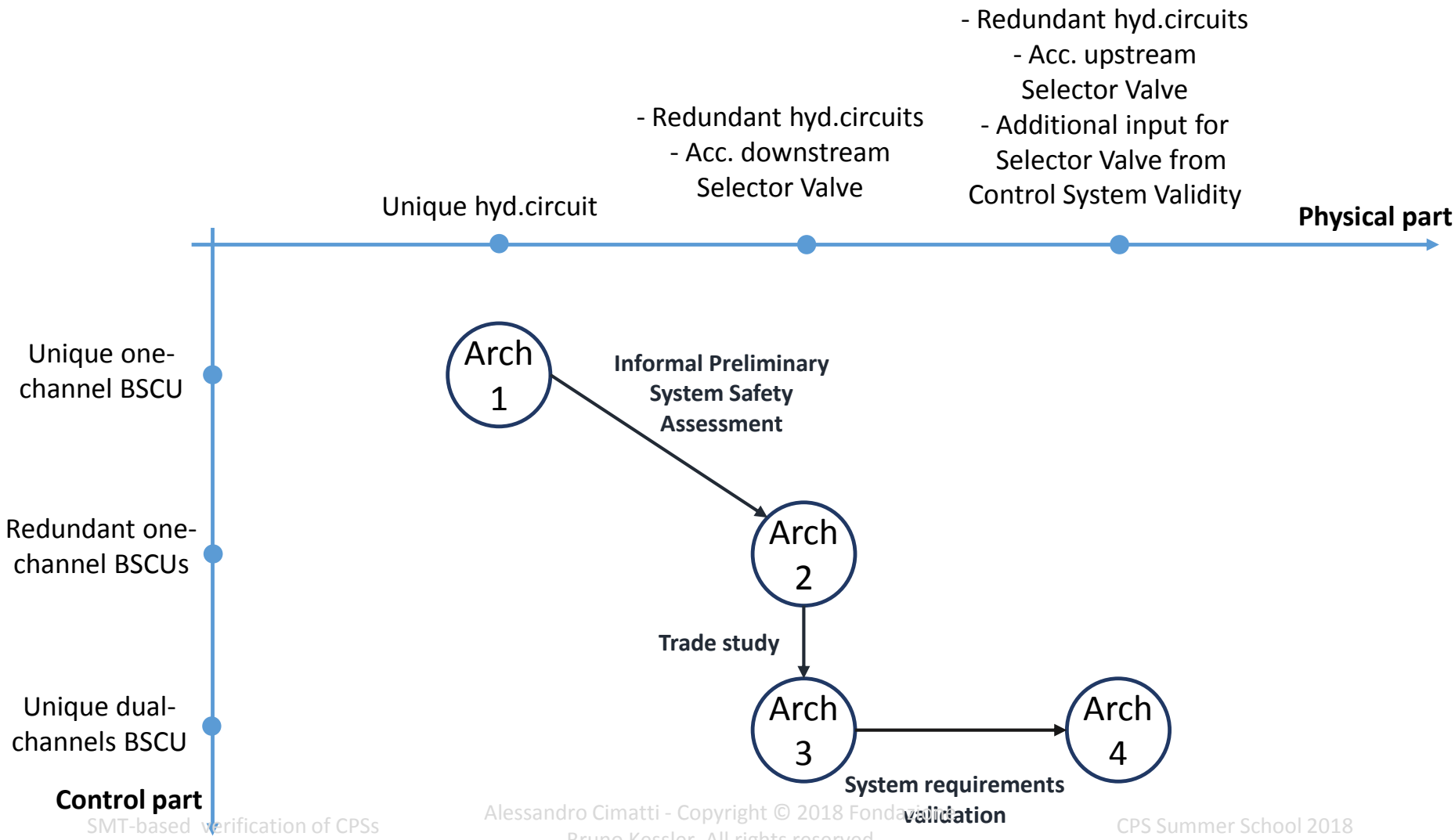
AIR 6110 Wheel Brake System



AIR 6110 Wheel Brake System



AIR 6110 Process



AIR 6110 WBS requirements

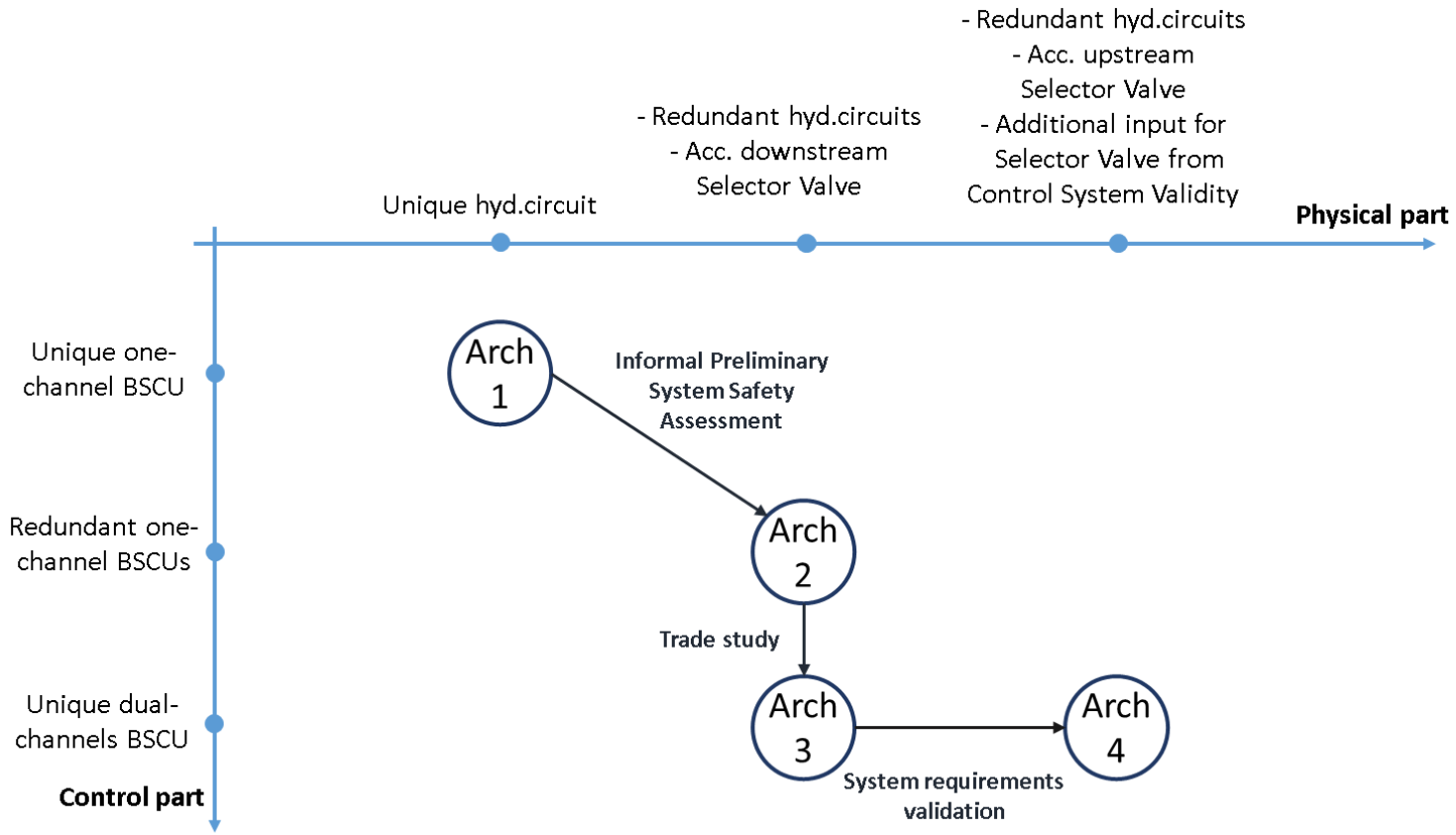
- Requirements sample:
- **S18-WBS-R-0321:** Loss of all wheel braking (unannunciated or annunciated) during landing or RTO shall be extremely remote
- **S18-WBS-R-0322:** Asymmetrical loss of wheel braking coupled with loss of rudder or nose wheel steering during landing or RTO shall be extremely remote
- **S18-WBS-0323:** Inadvertent wheel braking with all wheels locked during takeoff roll before V1 shall be extremely remote
- **S18-WBS-R-0324:** Inadvertent wheel braking of all wheels during takeoff roll after V1 shall be extremely improbable
- **S18-WBS-R-0325:** Undetected inadvertent wheel braking on one wheel w/o locking during takeoff shall be extremely improbable

Scope of the activity

- Review of the AIR6110 with:
  - Formal modeling
  - Formal Verification & Validation
  - Formal Safety Assessment
- Use of the presented workflow and tools

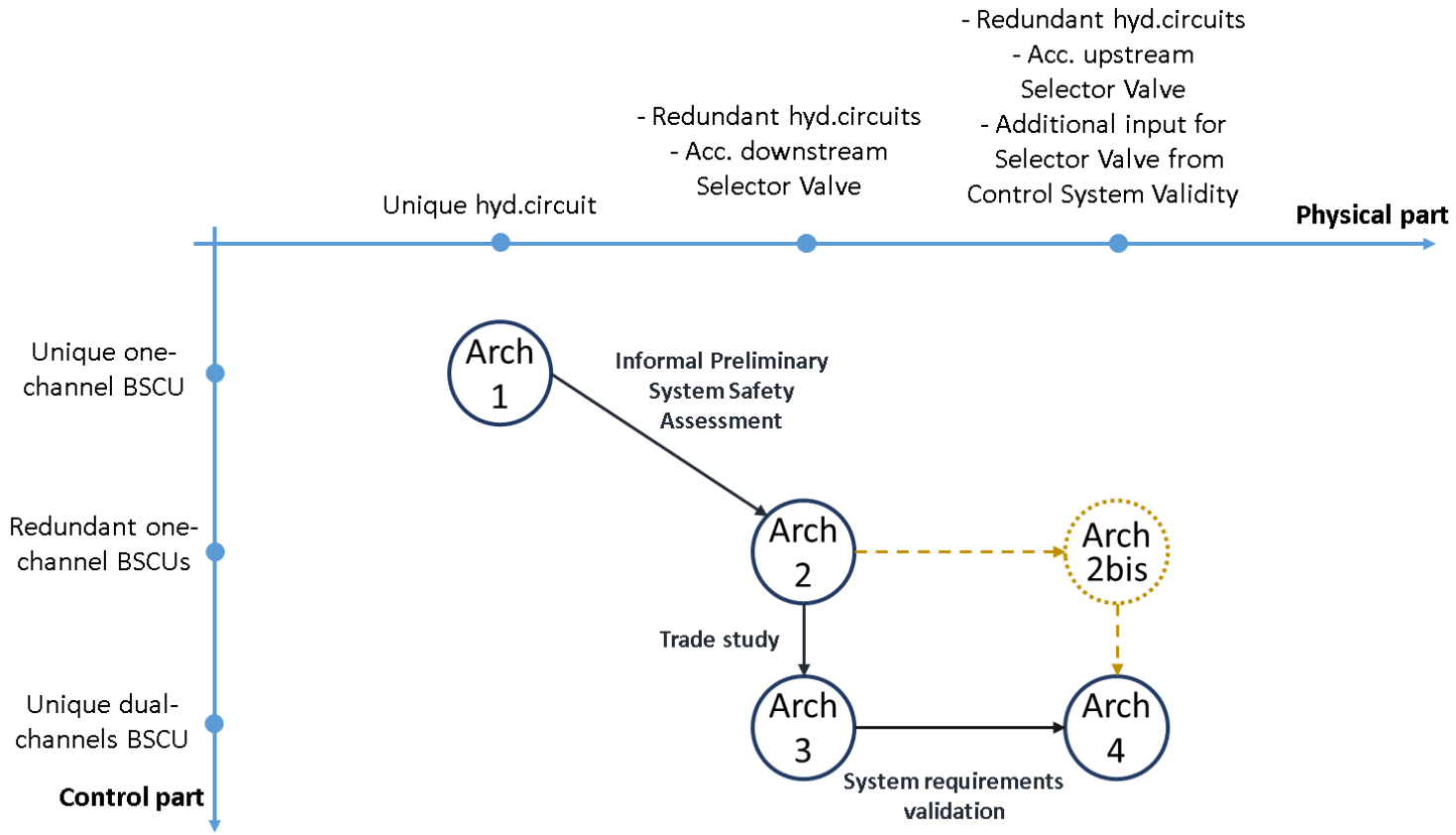
Application on AIR6110 WBS

- Application on 5 WBS architectures versions



Application on AIR6110 WBS

- Application on 5 WBS architectures versions



Formal modeling

- Hydraulic circuits are unidirectional
- Hydraulic pressures, braking force and ground speed are representing as bounded integer (0..10)
- Commands and power are Boolean
- Wheel speed becomes a wheel status: rolling or stopped
- Accumulator has an infinite reserve
- Discrete time
- All behaviors are instantaneous (except the wheel behavior)

Formal modeling

- Size of the formal models:
  - 30 component types for 169 instances
  - Max depth of 6 levels
  - 149 contracts for 304 property instances
  - 33 failure modes for 261 fault variables

- Translation of requirements:
  - Example:
    - **S18-WBS-R-0321: "***Loss of all wheel braking (unannunciated or annunciated) during landing or RTO shall be extremely remote***"**
    - **Becomes**: "*never loss of all wheel braking*"
    - *"Shall be extremely remote"* will be used for evaluating the reliability during MBSA

V & V: Compositional approach

- Contracts refinement (BDD algorithm) checked in 30-100s
- Detection of an unexpected flaw in Arch2
  - Preclusion of the operation modes: Normal VS Alternate
  - Arch2: the alternate circuit can be supplied by the accumulator while the normal circuit is operating
    - Detection of the problem in Arch3 which leads to Arch4! (AIR6110, p.67)
  - If application of the modification of Arch 4 concerning the placement of the accumulator
    - Creation of architecture Arch2bis
    - The previous property is verified

V & V: Monolithic approach

- BDD algorithm:
  - Build of the BDD model out of reach => Simplification needed
  - After simplification: All properties checked in ≈3000s
- IC3 algorithm:
  - No need of simplification
  - All properties checked in ≈150s

Model-Based Safety Analysis (MBSA)

- Conducted with xSAP
- Example of Safety requirements chosen as Top Level Events (TLE)
  - **S18-WBS-R-0321**: never loss of all wheel braking.
  - **S18-WBS-R-0322-left**: never asymmetrical loss of wheel braking (left side).
  - **S18-WBS-R-0322-right**: never asymmetrical loss of wheel braking (right side).
  - **S18-WBS-R-0323**: never inadvertent braking of all wheels with all wheels locked
  - **S18-WBS-R-0324**: never inadvertent braking of all wheels.
  - **S18-WBS-R-0325-wheelX**: never inadvertent braking of one wheel without locking. (duplicated for the 8 wheels)
- **3150 Analyses launched: 3089 succeeded, 61 timed out (10h)**

Model-Based Safety Analysis (MBSA)

- Arch1 is weaker than the other architectures
- Arch2 and Arch3 have the same results
  - It confirms the results of AIR6110: Modification due to trade study has no impact on the safety objectives.
- Arch4 is better than Arch3
  - same observation for Arch2bis and Arch2
- The computed probabilities for Arch2, Arch2bis, Arch3 and Arch4 are consistent with the expectations

Contract-Based Safety Analysis (CBSA)

- Conducted with OCRA
- Same property violations taken as Top Level Events (TLE)
- All hierarchical fault trees generated in a couple of minutes for one architecture
- For each property, the hierarchical fault tree produced is an over-approximation of the one produced with MBSA
  - Formally checked for the case study

Summing up

- Cover the process described in AIR6110 with formal methods
- Production of modular descriptions of 5 architectures
  - Analysis of their characteristics in terms of a set of requirements expressed as properties
  - Production of more than 3000 fault trees
  - Production of reliability measures
- Detection of an unexpected flaw in the process
  - Detection of the wrong position of the accumulator earlier in the process

Lessons learned

- Going from informal to formal allows highlighting the missing information of the AIR6110 to reproduce the process
- OCRA modular modeling allows a massive reuse of the design through architectures variant
- Automated and efficient engines as IC3 is a key factor
- MBSA is crucial in this context:
  - Automatic extension of the nominal model with faults
  - Automatic generation of artifacts eases the analysis and the architecture comparison in terms of safety

Technical report and all artifacts available at:

https://es.fbk.eu/projects/air6110

SMT modeling

Wheel Brake System case study

Infinite domain variables (SMT modeling)

Discrete time

From finite to infinite

- Use first-order predicates instead of propositions
  - $G(x \geq a \vee x \leq b)$
  - $GF(x = a) \wedge GF(x=b)$
- Predicates interpreted according to specific theory T (for example: Reals)
- Next operator to express changes/transitions:
  - $G(next(x) = x + 1)$
  - $G(next(a) – a \leq b)$

Contributions

- Same system used as in the Boolean modeling
  - Air6110 Wheel Brake System
- Same workflow and tools
- But more complex formal modeling:
  - Introduction of Real variables
  - Introduction of more fine-grained behaviors, including non-linear behaviors and parametric behaviors
- Re-run of the analyses:
  - Formal Verification & Validation
  - Formal Safety Assessment

Formal modeling

- What remains:
  - Hydraulic circuits are unidirectional
  - Wheel speed becomes a wheel status: rolling or stopped
  - Accumulator has an infinite reserve
  - Discrete time
  - All behaviors are instantaneous (except the wheel behavior)
- What changes:
  - Hydraulic pressures, braking force and ground speed are representing as real
  - Commands are represented as real, with a constrained range
  - Additional features:
    - Non-linear behavior
    - Parametric gain curve of the Meter Valve

Preliminary Conclusion

- More realistic model, gain of expressiveness for the designer
  - Less error prone!
- IC3 algorithm is still a key factor here
- V&V not that much impacted by the changes
- Safety Assessment is impacted in terms of performance

Hybrid modeling

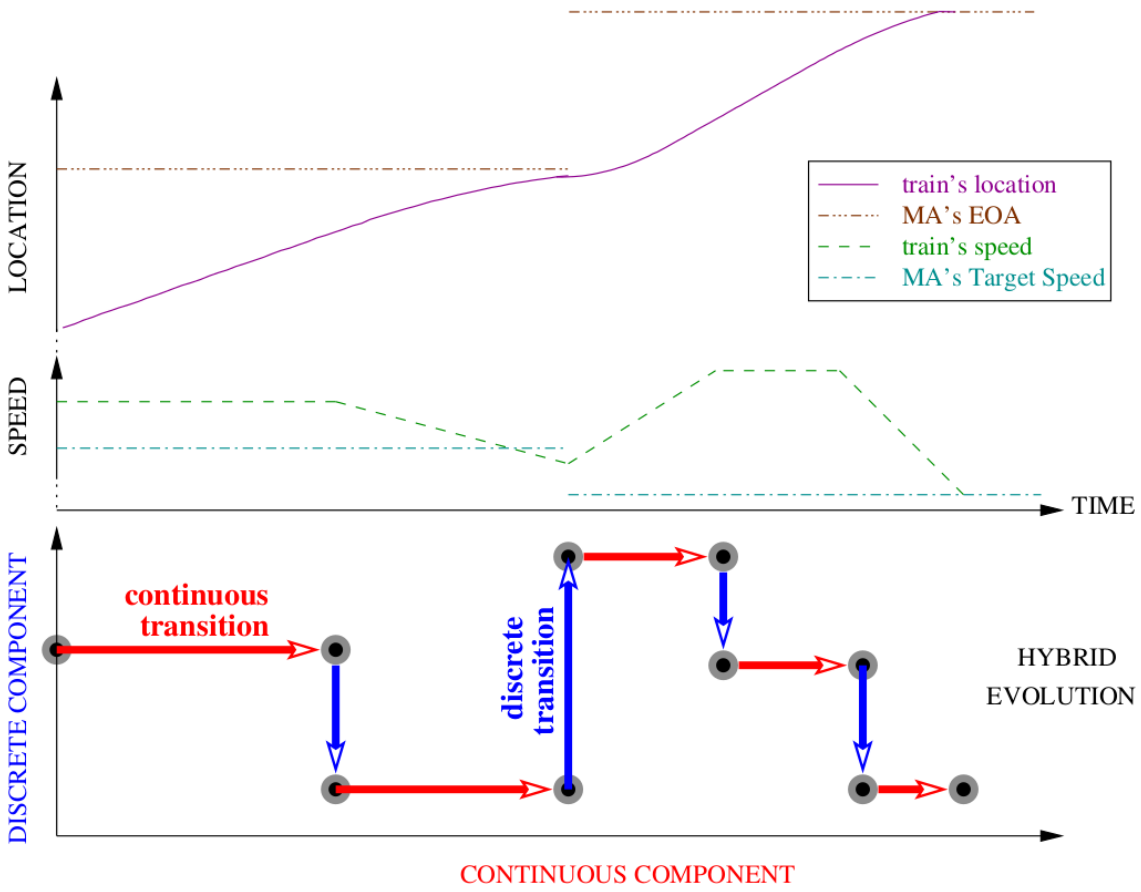
Landing Gear System case study

Infinite domains variables (SMT)

Hybrid time (discrete + continuous time)

From discrete to hybrid time

- Discrete transitions
  - Instantaneous change in (discrete) state of the system
- Continuous transitions
  - Continuous variables evolve over time according to specified laws
  - Discrete state does not change



Landing Gear System

- Industrial challenge from ABZ 2014 conference
- System in charge of the extension and retraction of the landing gears
- 3 different landing sets:
  - Front
  - Left
  - Right

Requirements

- 2 kinds of requirement:
  - Normal mode requirements
    - 9 requirements
    - Ex: *(R11) When the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then the gears will be locked down and the doors will be seen closed less than 15 seconds after the handle has been pushed.*
  - Failure mode requirements (detection of failures in the system i.e. failure of the Normal mode)
    - 10 requirements
    - Ex: *(R61) If one of the three doors is still seen locked in the closed position more than 7 seconds after stimulating the opening electro-valve, then the Boolean output normal mode is set to false*

Objectives

- Formal modeling
  - Use of Time Automata with Continuous time
  - Much more complex

- Application of the workflow
  - The only difference is the use of HyCOMP in place of nuXmv

Ongoing...

Work in Progress!

CONCLUSION & FUTURE WORK

Conclusion

- Workflow allows:
  - Massive reuse of component modeling through different variant of architectures
  - Automatic analyses and fault extension
  - Formal Verification & Validation
  - Formal Safety Assessment

- Tools and techniques used in various industrial collaboration

Future Work

- Front end to ease the modeling and analysis
  - CHESS tool
- Improvement of contract-based design: diagnostic info
- Improvement of scalability of MBSA
  - Improvement of fault tree computation
  - Computation a fault tree avoiding DNF composition is an important research challenge
- Improvement of the modeling capability
  - more realistic, more intuitive
- Analysis of redundant architectures
- Design of FDI and partial observability

THANK YOU