Automatic Architecture Hardening Using Safety Patterns

November 4, 2014

Authors: Kevin Delmas, Rémi Delmas, Claire Pagetti
e-mail: prenom.nom@onera.fr

Context

Application The typical application is

- Control command application
- I/O connected through a network
- Executed on a many-core platform

Criticality The application provides Cat, Haz and Maj functions, following the ARP4754 criticality classification

Case Study

Initial work from ROSACE [PSG+12]

- Longitudinal Flight Controller: hybrid controller
- Critical software
- Model-level description in SIMULINK
- Multi-rate application
- No fault-tolerance mechanisms

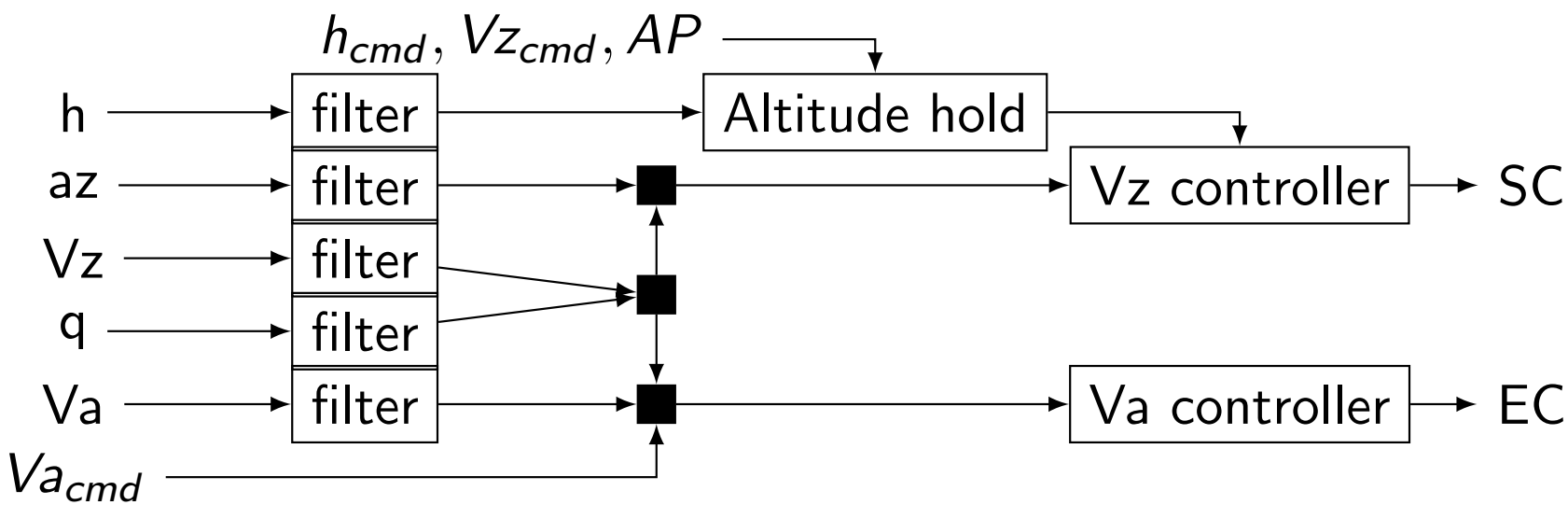


Figure: Controller Block Description

Design Process Overview

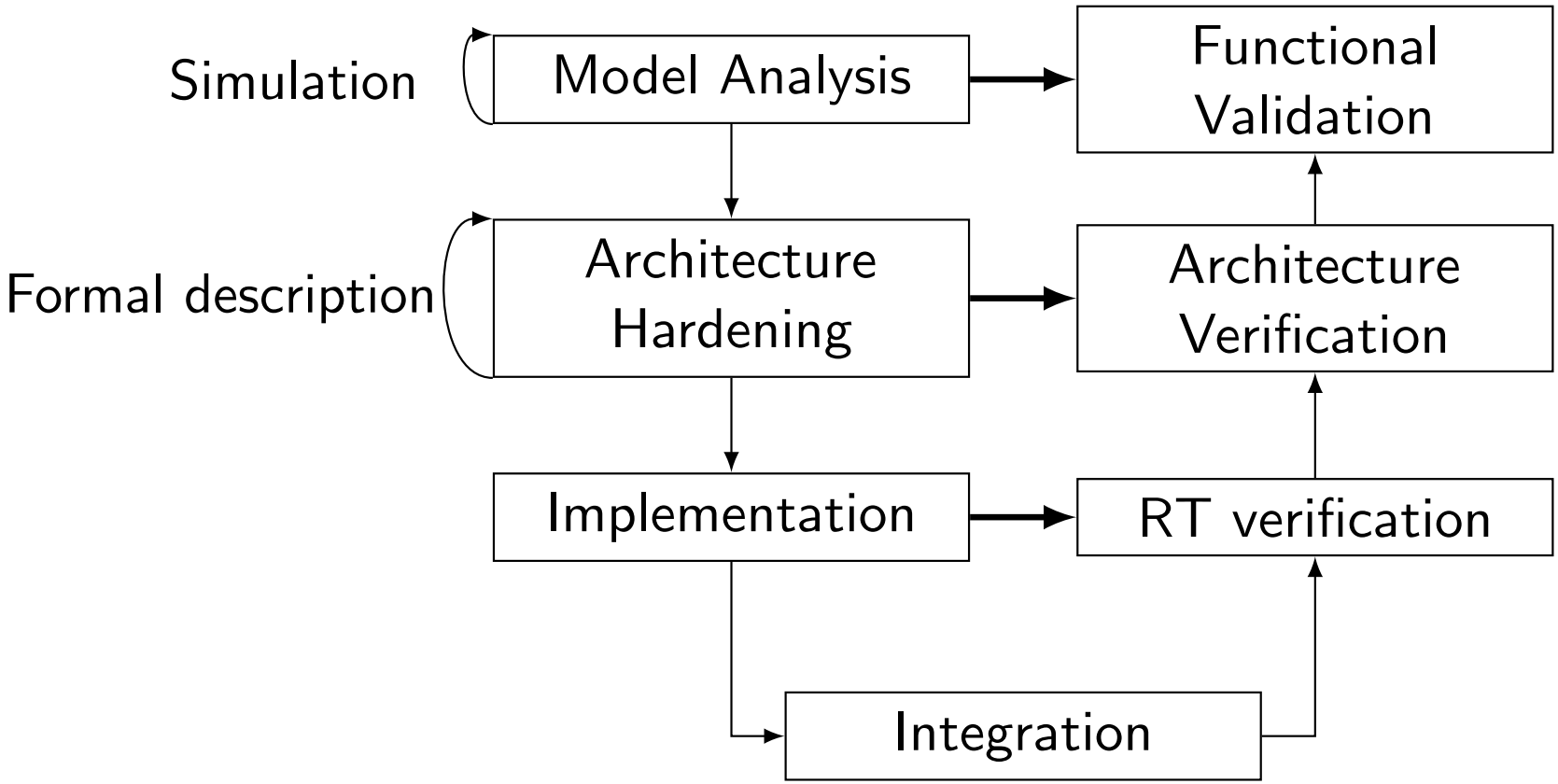


Figure: Hardening Design Process

Hardening process goal Enhance functional system model with redundancy to satisfy fault tolerance requirements. Focus on Single Upset Events induced faults [HOU06].

Table of Contents

1. Related Works

2. Proposed Hardening Process
   - Model Analysis
   - Application Hardening wrt. Platform-Induced Errors

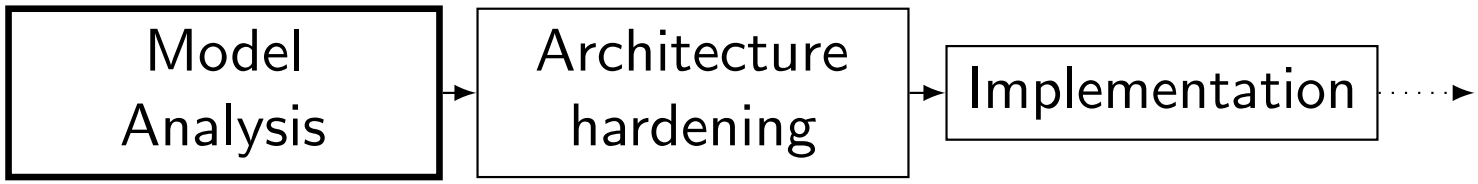
Safety Analysis Methods Model-based safety assessment using Altarica models and tools;

Single Event Upset SEU fault characterization and fault-tolerance means at silicon level and software level [HOU06];

Safety Design Patterns Redundancy-based safety patterns [KEH05];

Evolutionary Algorithms System enhancement with genetic algorithms [APS+11],[GO11],[WRP+13]. .

Phase 1: Simulink Model Analysis



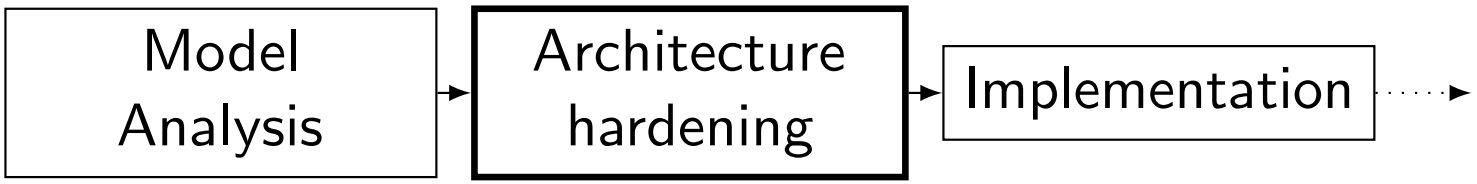
Phase Goal Modify SIMULINK model to make it tolerant to external faults:

- numerical errors on input values (sensor failures, network failures);
- temporal errors (delayed inputs).

Design Patterns range monitoring, offset filtering, integrator holding, Kalman filters, etc.

Model Assessment Fault injection simulation for typical error scenarios.

Ensure tolerance to execution platform faults (SEU)



Goal Suggest redundancy patterns for tolerance to platform faults (SEU)

Local Analysis Model platform-induced failure modes and failure propagation rules for individual Simulink components in AltaRica.

Global Analysis Assemble AltaRica components into a full system model and perform *model-based safety assessment* (MCS generation, sequence generation).

Safety Patterns Many-core platform $\rightsquigarrow$ heavy software redundancy. How to introduce redundancy **where needed**, at **lesser cost** ?

Component level modelling

Characterize effects of SEU-induced faults on software components. For each (SIMULINK) component:

FMECA Analyze effects and propagation to outputs of input or local data corruption.

Altarica component model Encode FMECA results in an ALTARICA components.

Example (Local description)

Component	Failure Mode	Local effect	Type
Va controller	state corruption	inconsistent EC	e^p
⋮	⋮	⋮	⋮

Table: Component Description

Dysfunctional application modelling with ALTARICA

Describe Assemble altarica components to obtain a model of safety effects of platform faults on the application.

Analyze Express and classify failure conditions, run MCS generator, evaluate MCS against safety requirements.

Modify Based on MCS, automatically identify components needing redundancy, select appropriate design pattern.

Safety Design Patterns

We used safety patterns based on hot redundancy:

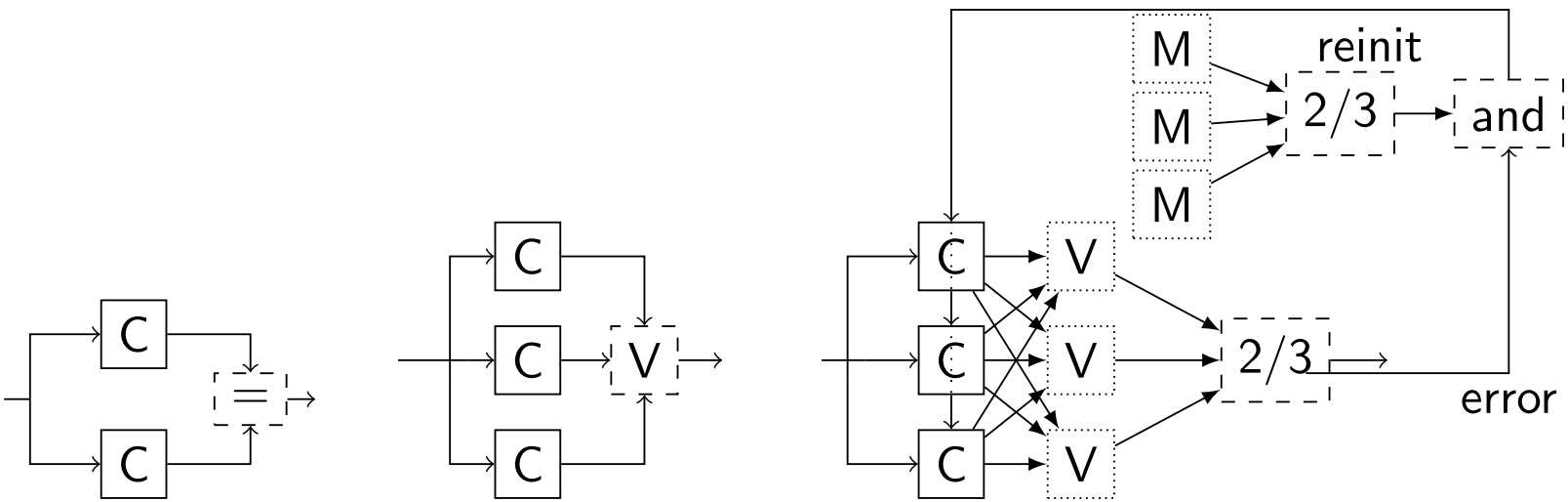


Figure: Design patterns used in the Rosace case study.

Iterative Hardening

The following iterative scheme is used:

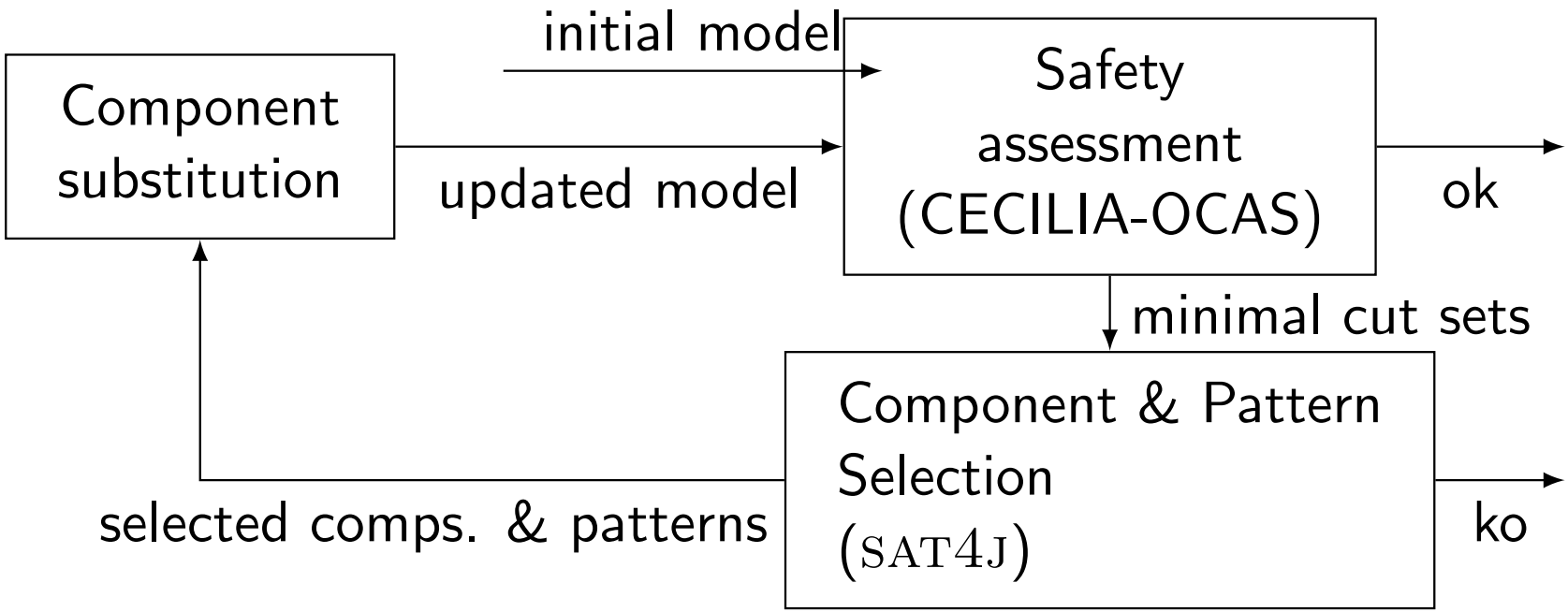


Figure: Iterative Hardening

MCS generation + assessment always last $\rightsquigarrow$ reduced assurance level on tools used for model modification.

Ensuring Fault Tolerance Increase

Goal Ensure actual increase of minimal cut sets cardinality by pattern application

Means Define and compute appropriate ordering relation of patterns

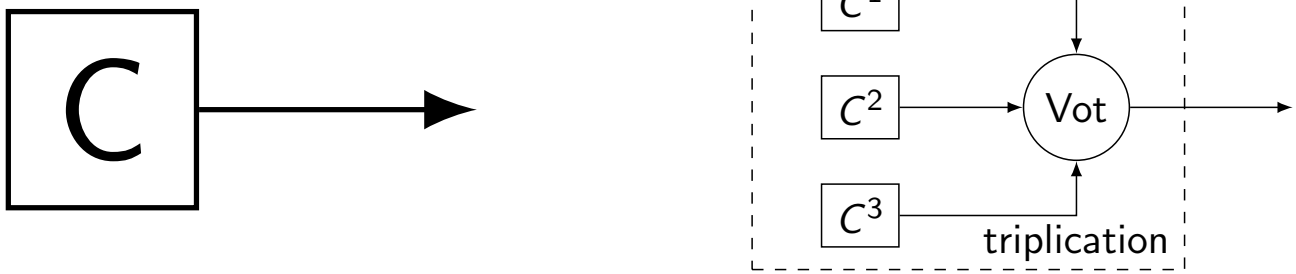
Let $C \in \mathrm{Comps}$ be a component, and $\mathrm{Pat}_C = \{P_1, \ldots P_n\}$ the set of patterns applicable to C.

Hyp. Output interface preservation:
$\mathrm{OutFlows}(P\langle C\rangle) = \mathrm{OutFlows}(C)$.

Def. Minimal number of events needed to corrupt at least one output of $P\langle C\rangle$:

$$\mathrm{Faulty}(P) : \bigvee_{o \in \mathrm{OutFlows}(P\langle C\rangle} (o \neq ok)$$

Ensure minimal cuts cardinalities increase



$$MCS_{\text{Faulty}(P)} = \begin{matrix}\{C.a\}\\ \{C.b\}\end{matrix} \qquad MCS_{\text{Faulty}(P')} = \begin{matrix}\{C^1.a, & C^2.b\}\\ \{C^1.a, & C^3.a\}\\ \{C^3.d, & C^2.b\}\end{matrix}$$

$$\underset{mcs \in MCS_{\text{Faulty}(P)}}{MIN} (card(mcs)) = 1 \qquad \underset{mcs \in MCS_{\text{Faulty}(P')}}{MIN} (card(mcs)) = 2$$

$$P < P'$$

So, we define the transitive & reflexive relation $<_C$ such that, for $(P, P') \in \mathrm{Pat}_C^2$:

$$P > P' \equiv \underset{mcs \in MCS_{\mathrm{Faulty}(P)}}{MIN} card(mcs) > \underset{mcs \in MCS_{\mathrm{Faulty}(P')}}{MIN} card(mcs)$$

Automating component and pattern selection

Q. Which components should be modified?

A. The smallest set of components involved in minimal cut sets of too low cardinality wrt. some failure condition.

Express component and pattern selection as a pseudo-Boolean optimization problem.

Pseudo-Boolean encoding

Component selection variables and constraints:

Vars $\{\mathrm{SelectComp}(C) | C \in \mathrm{Comps}\}$, where $\mathrm{SelectComp}(C) = \top$ means C is selected for modification;

Ctrs At least one component selected in each problematic *mcs*:

$$\mathrm{SelectCompCtr}(mcs) \equiv \sum_{e \in mcs} (\mathrm{SelectComp}(\mathrm{Evt2Comp}(e))) \geq 1$$

Pseudo-Boolean encoding

Pattern selection variables and constraints:

Vars $\{\mathrm{SelectPat}(C, P) | C \in \mathrm{Comps}, P \in \mathrm{CompPat}(C)\}$, where $\mathrm{SelectPat}(C, P) = \top$ means P selected for C.

Ctrs At most one pattern selected for each selected component:

$$\mathrm{AtMostOnePatternCtr}(C) \equiv \sum_{P \in \mathrm{CompPat}(C)} \mathrm{SelectPat}(C, P) \leq 1$$

Pseudo-Boolean encoding

Embedding of the relation $<_C$:

Vars $\{betterThan_C(P, P') | C \in \mathrm{Comps}, (P, P') \in \mathrm{CompPat}(C)^2\}$, such that $betterThan_C(P, P') = \top$ if and only if $P\langle C\rangle >_C P'\langle C\rangle$

Ctrs Chosen pattern is better than previously chosen pattern:

$$\mathrm{BetterThanPrev}(C, P) \equiv$$
$$\neg\mathrm{SelectPat}(C, P) + \mathrm{GT}_{\mathrm{CompPat}(C)}(P, \mathrm{PrevPattern}(C)) \leq 1$$

Pseudo-Boolean encoding

- Optimization criterion: select a minimum number of components:

$$\textit{Minimize} \sum_{C \in \mathrm{Comps}} \mathrm{SelectComp}(\mathrm{Evt2Comp}(e))$$

- Optionally, minimize more than one criteria using leximin criterion aggregation:
  - memory consumption
  - cpu consumption
  - etc.

Example on case study

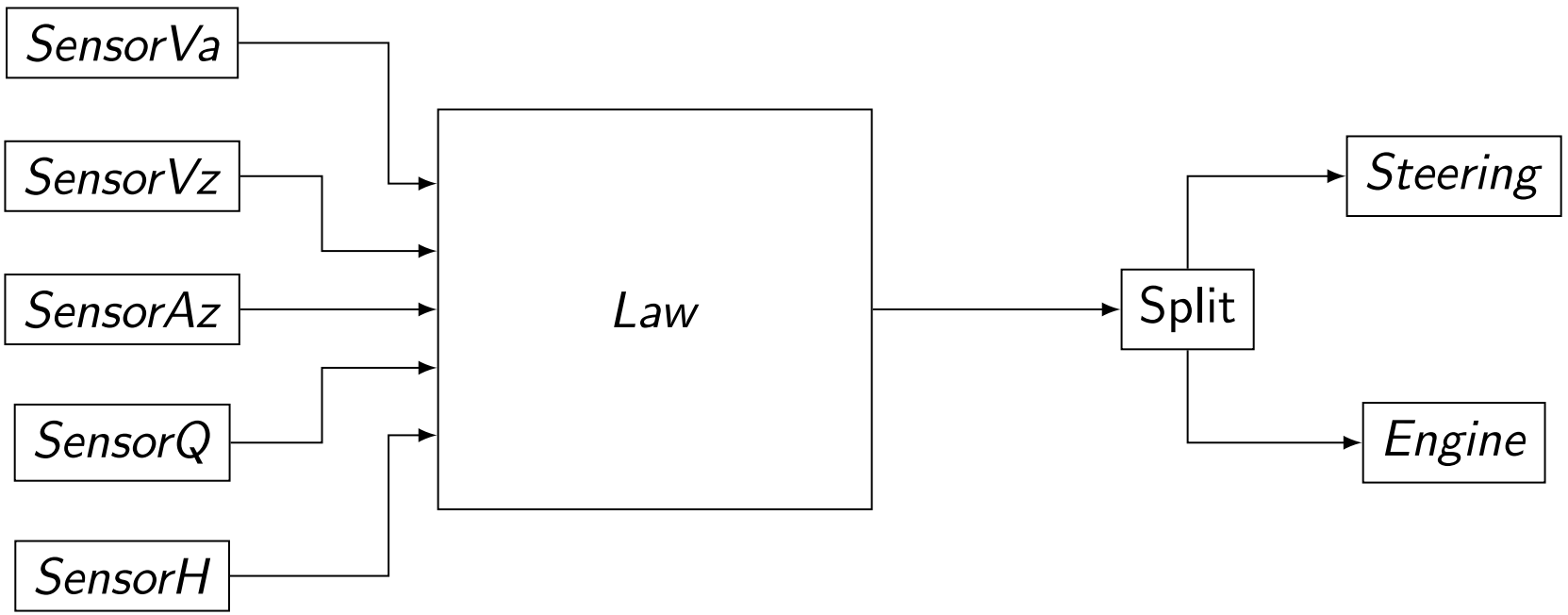


Figure: Initial Architecture of ROSACE

Example on case study

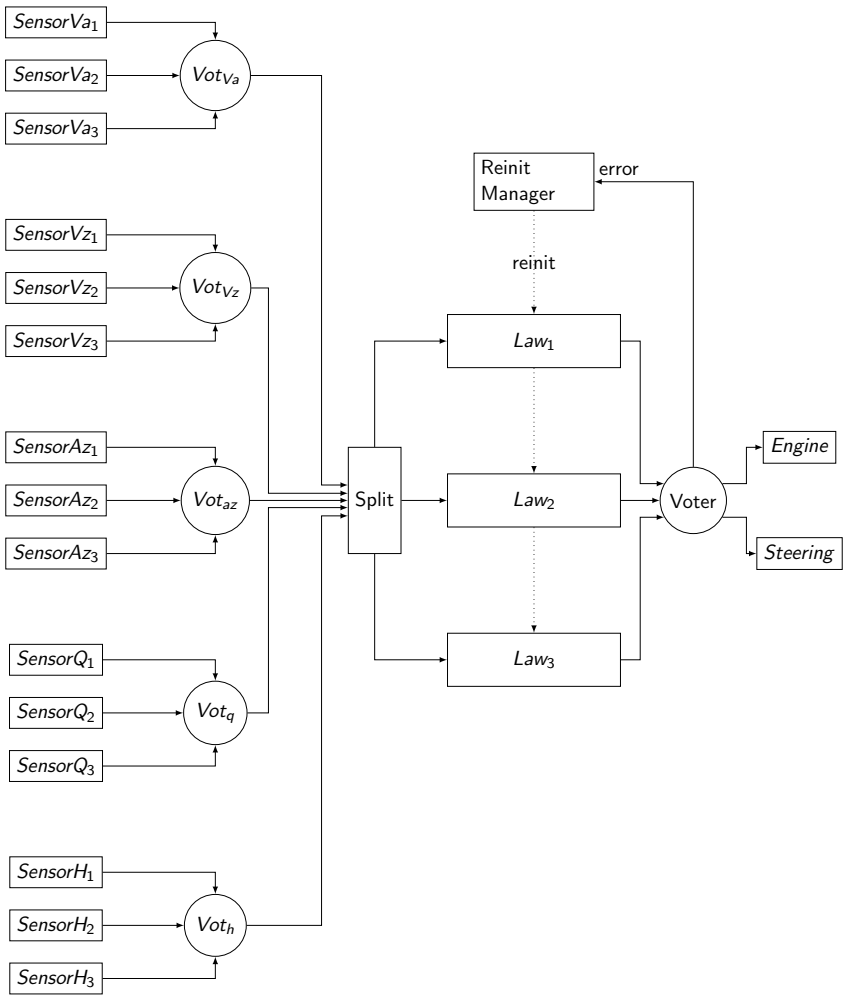


Figure: Final Architecture

Related Work

- Most other approaches are based on *genetic algorithms* [APS+11],[GO11],[WRP+13]: breed best previous solutions together, generate candidates by random mutations, evaluate (multi-criteria), select best, iterate.
- Applied mutations can be unnecessary,
- Many variants to evaluate, seems to be the main bottleneck.

Conclusion

Conclusion

- A constraint-based method for model hardening.
- Guaranteed increase of minimal cut sets size.

ıture work

- Provide a language to describe patterns (today, instances are built by hand for each component and pattern),
- Automate pattern ordering generation,
- Automate model transformation,
- Refine pattern ordering relation to take into account *erroneous* and *lost* failure modes separately,
- Benchmark on large system (hundreds of components).

Bibliography


Masakazu Adachi, Yiannis Papadopoulos, Septavera Sharvia, David Parker, and Tetsuya Tohdo.
An approach to optimization of fault tolerant architectures using hip-hops.
Softw., Pract. Exper., 41(11):1303–1327, 2011.


Matthias Güdemann and Frank Ortmeier.
Model-based multi-objective safety optimization.
In *SAFECOMP*, pages 423–436, 2011.


Sabrine HOUSSANY.
Méthodologie d'évaluation de la sensibilité des microprocesseurs vis-à-vis des rayonnements cosmiques.
PhD thesis, UNIVERSITÉ DE GRENOBLE, 2006.


Christophe KEHREN.
Motifs formels d'architectures de systèmes pour la sureté de fonctionnement.