SysML-based and Prolog-supported FMEA

Fabio Scippacercola, Roberto Pietrantuono, Stefano Russo
Università degli Studi di Napoli Federico II
{fabio.scippacercola, roberto.pietrantuono, stefano.russo}@unina.it

Nuno Pedro Silva
Critical Software, SA
nsilva@criticalsoftware.com
Failure mode and effects analysis (FMEA) is widely used to identify the potential failures of components and for evaluating their effects on the system, which could adversely affect its overall reliability or safety.

FMEA is a time-consuming technique that includes repetitive task:

- The support provided by tools is still limited for specific tasks, e.g. the analysis of faults propagation among components.

- More complex tasks, such as the analysis of the effects of multiple failures, are often neglected by the analyst.
A SysML-based and Prolog-supported FMEA approach

- We propose **an approach to support FMEA** by enabling formal knowledge representation **within a SysML-based Model-Driven context**:
  - Failure modes and propagation conditions are specified complementing SysML models, by means of annotations and stereotypes.
  - This enables automatic transformation into a Prolog knowledge base, which can be queried (e.g., to identify the flows’ and blocks’ states that lead to system failures).

- We support **the reasoning in the same conceptual framework of a model-driven design methodology**, favoring communication among the designer and the analyst, and automating inductive reasoning steps about fault propagation.
Illustrative Example: alarm system

The battery (connected to the power supply) can mask short outages of the energy.

When the push-button is pressed the energy flows into the solenoid.

The solenoid generates the electromagnetic field that makes ring the bell.
FMEA-oriented modeling (1/3)

- We complement model-driven processes and benefit from SysML architectural model (that can be provided by design engineers).

  **The system architecture is modeled using structural diagrams;** information flow among components is modeled using ports and flows.

  SysML fosters communication between the designer and the FMEA analysts, and allows to model components failure behaviors in an incremental way, as low-level system design proceeds.
A FMEA-oriented SysML profile defines the aspects of interest for FMEA and allows to model the failure modes and of the propagations of failures.

By means of annotations, the analyst enriches the SysML model with:

1. the description of the logical states that hold for all input flows;
2. the description of the logical internal states of the blocks that are related with the failure modes of the components, and can be dependent on the logical conditions on the input flows;
3. the constraints on the block’s and flows’ logical states that can depend on local or global conditions.
4. additional information (e.g., for quantitative analyses).

This activity is mostly local and the modeler focuses on local conditions and effects of the failure modes.
The failures of components and the conditions for their propagation are expressed in logical terms, by a set of Horn clauses.

\[
\begin{align*}
\text{powerSupply} &= \{\text{on, outage}\} \\
\text{battery} &= \{\text{lowBattery, chargedBattery, cell\_Malfunction}\} \\
\text{energy} &= \{\text{on, outage}\} \\
((\text{powerSupply} = \text{outage}) \; \text{AND} \; (\text{battery} = \text{lowBattery})) \; \text{OR} \; (\text{battery} = \text{cell\_malfunction}) \\
&\Rightarrow \text{energy} = \text{outage}
\end{align*}
\]
The extended (SysML) model is transformed into a knowledge base (KB) for subsequent analysis through a Model-to-Text transformation (M2T). To this end, we adopt the logic programming language **Prolog**.

Any language with a Prolog-like syntax (e.g. ProbLog) supports the M2T transformation and queries, that be formulated by the FMEA analyst in terms of logic expressions (with details to can be made transparent to the user).
The **knowledge base** contains:

1. the definition of **rules and facts** that support the operations in a FMEA analysis.
2. the **knowledge specific** for the instance of the system under analysis. The KB derived for the alarm system is listed in the following.
3. the **knowledge** that can be reused in multiple projects, part of the domain under analysis.

Since the KB is extern to the model, we can reuse the knowledge in multiple projects! E.g. a KB can include the behavior of the domain components.

```prolog
state(powerSupply, on, _, 0.9999).
state(powerSupply, outage, _, 0.0001).
state(battery, lowBattery, _, 0.009).
state(battery, chargedBattery, _).
state(battery, cell_malfunction, 0.99).

state(energy, on, M, 1) :- (member([powerSupply, on], M), not(member([battery, cell_malfunction], M))), !.
state(energy, on, M, 1) :- member([battery, chargedBattery], M).
state(energy, outage, M, 1) :- (member([powerSupply, outage], M), member([battery, lowBattery], M)), !.
state(energy, outage, M, 1) :-
```
Model Analysis (1/2)

- Prolog supports the analysis of the modeled system, extracting knowledge that is hard to derive by a manual or a pure model-based analysis. For instance, it allows:
  - to follow the propagation of failures inside the system;
  - to identify root causes of a component’s failure;
  - to compute failures derived from multiple errors;
  - to study the effectiveness of fault tolerance mechanisms.

- Complex queries can be expressed in Prolog, while the most common features can be offered through an easy-to-use front-end that makes transparent the details of the computation.
Model Analysis (2/2)

- FMEA queries expressible in the form of Horn clauses are performed on the KB: this supports typical inductive reasoning tasks of the FMEA analyst.
- It is also possible to associate a probability to the satisfaction of Horn clauses, enabling a first form of quantitative FMEA.

Can the System fail in the alarm active state?

YES, there are three occurrences!

\[
M = \ \{ \text{powerSupply, on}, \text{battery, lowBattery}, \text{energy, on}, \text{mechanicalForce, inactive}, \text{switchButton, stuckClosedCircuit}, \text{solenoidEnergy, on}, \text{solenoid, working}, \text{electromagneticField, active}, \text{bell, working}, \text{acousticSignal, active} \} 
\]

\[
M = \ \{ \text{powerSupply, on}, \text{battery, chargedBattery}, \text{energy, on}, \text{mechanicalForce, inactive}, \text{switchButton, stuckClosedCircuit}, \text{solenoidEnergy, on}, \text{solenoid, working}, \text{electromagneticField, active}, \text{bell, working}, \text{acousticSignal, active} \} 
\]

\[
M = \ \{ \text{powerSupply, outage}, \text{battery, chargedBattery}, \text{energy, on}, \text{mechanicalForce, inactive}, \text{switchButton, stuckClosedCircuit}, \text{solenoidEnergy, on}, \text{solenoid, working}, \text{electromagneticField, active}, \text{bell, working}, \text{acousticSignal, active} \} 
\]
Additional Model-To-Text transformation can generate FMEA worksheets and reports.
The SysML and Prolog-based approach

(In a model-driven process) Design Engineer defines the **SysML model of the system**

1. **FMEA-oriented model** can exploit an incremental knowledge base (in Prolog) of components’ failure modes.

   - **Design Engineer** defines the **SysML Design Model**
   - **FMEA Engineer** defines the **FMEA-Oriented SysML Model**
   - **Past Project Knowledge**
   - **FMEA Profile**

2. A **M2T transformation** generates the Prolog KB that contains the knowledge about the model and the system domain.

3. **Prolog** enables to **perform queries** on the model to get results and to produce FMEA worksheets.

The FMEA Engineer refines the model, using the **FMEA-oriented Profile** (SysML diagrams with stereotypes/annotations).

**Prolog** enables to **perform queries** on the model to get results and to produce FMEA worksheets.
An Eclipse implementation

- An implementation of our approach is being developed as a **plug-in in the Eclipse environment**, adopting open source software.

**SWI-Prolog** provides an efficient implementation of Prolog, and offers a rich set of libraries to support the execution, and the interfaces needed to interact with Java applications.

**Papyrus** is an open source Eclipse **SysML/UML** modeling tool, providing support for custom profiles.

**Prolog Knowledge Base**
Related Work

- Several approaches have been proposed in the literature, starting from a system model (including in UML and SysML) or from fault trees.
  - We similarly augment the system model with FMEA-oriented profile, but we exploit a Model-Driven process.

- Past studies introduced the idea of defining an (external) knowledge base in order to promote the reuse.
  - We define the KB in Prolog and relate the model to the KB using custom annotations.

- Very few works envisage Prolog for FMEA, and with no link to design models.
  - Prolog is more familiar to the engineering that more complex formalisms (e.g. ontologies) since it mimics the deductive reasoning that an analyst performs during FMEA.
Conclusions

- We have proposed a **model-driven approach** to support the FMEA, **based on FMEA-oriented SysML models and Knowledge Base in Prolog**.
  - The **approach is meant to work with modern model-driven methodologies**.
  - **Prolog is suited for representing the knowledge base**, as this can be algorithmically **generated by a model-to-text transformation** of structural SysML diagrams.

- A **prototypal tool has been set up and to show the approach in operation**:
  - **Typical FMEA queries** expressible in the form of Horn clauses **can be performed on the knowledge base** by means of an inference engine.
  - **Reuse of knowledge** of components failure modes over different FMEA projects in an organization **is also supported** in the envisaged approach.
Future Work

- The research is however still on-going. We foresee the following next steps:
  - We plan to define a **SysML profile** and to provide guidelines for the analyst to integrate the information needed for FMEA.
  - This will enable the formal definition of the **model-to-text transformation** into the knowledge base.

- The effectiveness of the approach requires also that the queries in Prolog be feasible when dealing with complex systems. This calls for the investigation of optimization techniques when dealing with the problem of state space explosion.
  - The application to real-world FMEA case studies will assess the applicability and scalability of the approach, as well as the performance of the envisaged support tools.
Acknowledgement

- This work has been partly supported by the SVEVIA (Innovative methods and techniques for Software VERification and ValldAtion of near-realtime complex systems) Research Project funded by the Italian Ministry of Education and Research (Grant no.: PON02 00485 3487758), and by EU project under Marie Curie Actions of the FP7 People Programme, REA grant n.324334 CECRIS (CErtification of CRItical Systems, www.cecris-project.eu).

- The work of F. Scippacercola has been partially done during secondment periods at Critical Software S.A., Coimbra (PT).