Most Influential Paper Award presented at MODELS-2021
10 years after publication in SoSyM 2011

A dependability profile within MARTE

Simona Bernardi¹, José Merseguer¹, Dorina C. Petriu²
¹ University of Zaragoza, Spain
² Carleton University, Ottawa, Canada
Talk Outline

- **Part I: Paper contributions**
  - Context
  - Goals
  - Approach
  - Application to a case study

- **Part II: Influence of the paper**
  - Citations
  - Influence on our own work
    - Theses supervised
    - Publications (book, journal and conference papers)
    - Tool support
  - Future challenges
    - Integrating the analysis of multiple NFPs in the MDE process
    - Standardization
Dependability evaluation

Fault Forecasting:  
*Means to estimate the present number, the future incidence, and the likely consequences of faults [Avizienis et al. 2004]*

- Conducted by carrying out an evaluation of the system behavior w.r.t. fault occurrence
  - **Qualitative**: identify, classify and rank failure modes, causes-effect analysis
  - **Quantitative**: probabilistic/stochastic evaluation of dependability measures via modeling and testing

Integration of dependability modeling and analysis in Model-Driven Software Engineering
Use of UML profiling to define DSML for the assessment of Non-Functional Properties

- Construction of formal models for the assessment of NFP of software systems [from ‘99 to today]
  - Specified in UML
  - From the early phases of the life-cycle
- Such contributions lead to the definition of standard OMG UML profiles for modelling and analysis
  - Performance & schedulability (UML SPT [2005], UML MARTE[2011])
  - QoS characteristics (UML QoS&FT [2008])
  - Dependability for safety consumer devices (UML DAF [2016])
Aim of the Dependability Analysis and Modeling (DAM) profile

- Define a UML Profile to support the dependability evaluation of software systems with focus on RAMS properties
  - Reliability: the continuity of correct service delivery
  - Availability: the promptness of correct service delivery
  - Maintainability: the capability to undergo modifications and repairs
  - Safety: the absence of catastrophic consequences on the users and environment

- Unify the terminology and concepts for different dependability aspects under a common dependability domain model
Methodological approach for the definition of the DAM profile

1. Literature review
2. Requirement checklist
3. Definition of DAM domain model
4. Completeness assessment of the DAM model
5. Design of the DAM profile
6. DAM profile assessment

[no] [yes] complete?
[no] [yes] all reqs satisfied?
DAM domain model: Package overview
DAM domain model: Redundancy

```
Redundant structure
  FTlevel

self.FTlevel < self.comp->size()
```

```
(DAM::DomainModel::System::Core)
  Component

+ substitutesFor

1..*

+ substitutedBy

+ *

Spare  Controller  Variant  Adjudicator

self.substitutesFor->excludes(self.oclAsType(Component))
```
DAM domain model: Threats
DAM domain model: Maintenance
Mapping the domain concepts into a UML profile

- General guideline to extend UML meta-model
- Best practice of UML MARTE for traceability
- Reuse of UML MARTE (import/application of sub-profiles)
Mapping process: classes

- Eventually only a subset of classes have been mapped to stereotypes
  - Abstract classes: not considered
  - Threat/Maintenance classes: complex dependability types (DAM library)
  - Specialization of MARTE stereotypes vs/ Extension

```
«Stereotype»
GaScenario
(MARTE::MARTE_AnalysisModel::GQAM)

«Stereotype»
DaService

«Metaclass»
UML::Classes::Kernel::Classifier

«extend»

«Stereotype»
DaServiceRequest
```
Mapping process: dependability types

- **Basic dependability types**
  - Enumeration
  - Datatypes that inherit from MARTE basic NFP types

- **Complex dependability types**
  - Datatypes with attributes of basic NFP types, basic dependability types or complex dependability types
Mapping process: attributes

- Domain attributes have been mapped to attributes of
  - stereotypes
  - complex dependability types
- For each attribute
  - a type is associated
  - a multiplicity is defined

<table>
<thead>
<tr>
<th>«Stereotype»</th>
<th>DaService</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ execProb: NFP_Real [*]</td>
<td></td>
</tr>
<tr>
<td>+ ssAvail: NFP_Percentage [*]</td>
<td></td>
</tr>
<tr>
<td>+ instAvail: NFP_CommonType [*]</td>
<td></td>
</tr>
<tr>
<td>+ unreliability: NFP_CommonType [*]</td>
<td></td>
</tr>
<tr>
<td>+ reliability: NFP_CommonType [*]</td>
<td></td>
</tr>
<tr>
<td>+ missionTime: NFP_CommonType [*]</td>
<td></td>
</tr>
<tr>
<td>+ availLevel: DaLevel [*]</td>
<td></td>
</tr>
<tr>
<td>+ reliabLevel: DaLevel [*]</td>
<td></td>
</tr>
<tr>
<td>+ safetyLevel: DaLevel [*]</td>
<td></td>
</tr>
<tr>
<td>+ complexity: NFP_Real [*]</td>
<td></td>
</tr>
</tbody>
</table>

```
+ condition: String [0..1]
+ consequence: DaCriticalLevel [*]
+ consistency: Consistency [0..1]
+ cost: DaCurrency [*]
+ description: String [0..1]
+ detectability: Detectability [0..1]
+ domain: Domain [0..1]
+ MTBF: NFP_Duration [*]
+ MTTF: NFP_Duration [*]
+ occurrenceDist: NFP_CommonType [*]
+ occurrenceProb: NFP_CommonType [*]
+ occurrenceRate: DaFrequency [*]
+ risk: NFP_Real [*]
```
Mapping process: associations

- Application of the “reference association” pattern
Application to a case study

- Message redundancy service (MRS)
  - Aimed at delivering of only uncorrupted messages to the target system

- Modelling of fault (intrusion) tolerance mechanisms
  - Hw/sw redundancy
  - Voting & time-out

- Analysis of different NFPs
  - Availability
  - Response time
  - Robustness (application specific metric)
    - Rate of filtered messages

- Sensitivity analysis
  - Availability and response time metrics vs/ fault occurrence rate and time-out duration
  - Rate of filtered messages vs/ fault occurrence rate and probability of incorrect messages
# DAM annotation

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Stereotype</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Diagram</td>
<td>DaService</td>
<td>availability metric</td>
</tr>
<tr>
<td>Deployment Diagram</td>
<td>DaController, DaVariant, DaComponent</td>
<td>redundancy structure (hw redundancy level, fault occurrence rate)</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td>DaController, DaVariant</td>
<td>use of redundant sw component (redundancy level)</td>
</tr>
<tr>
<td>State Machine</td>
<td>DaStep</td>
<td>failure occurrence (caused by fault annotated in DD)</td>
</tr>
</tbody>
</table>
## MARTE annotation

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Stereotype</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Diagram</td>
<td>GaAnalysisContext</td>
<td>parameters declaration</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td>GaWorkloadEvent</td>
<td>closed arrival pattern</td>
</tr>
<tr>
<td></td>
<td>GaService</td>
<td>response time metric</td>
</tr>
<tr>
<td>State Machine</td>
<td>GaStep</td>
<td>throughput metric, probability and duration</td>
</tr>
</tbody>
</table>
Analysis with MARTE-DAM

- Deterministic and Stochastic Petri Net (DSPN)
- Customization of a model transformation approach based on model composition*

* Merseguer, Bernardi, Campos, Donatelli, “A compositional semantics for UML State Machines aimed at performance evaluation”. WODES02
Part II

Our paper’s influence

Citations
Our paper’s citations

- **Google Scholar**: 195 citations
- **Microsoft Academic**: 202 citations
- **Scopus**: 99 citations
- **Semantic Scholar**: 151 citations
  - Highly influential citations: 8
  - Background citations: 41
  - Method citations: 58
- **Springer Link** metrics:
  - **Web of Science**: 67 citations
  - **CrossRef**: 96 citations
  - Accesses to the paper in Springer Link: 483
## Google Scholar citations: how cited are they in turn?

<table>
<thead>
<tr>
<th>Citing work</th>
<th>Cited by</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Selic &amp; Gerard, 2013)</td>
<td>118</td>
</tr>
<tr>
<td>(Bernardi et al., 2012)</td>
<td>111</td>
</tr>
<tr>
<td>(Biggs et al., 2016)</td>
<td>72</td>
</tr>
<tr>
<td>(Bernardi et al., 2010)</td>
<td>61</td>
</tr>
<tr>
<td>(Bernardi et al., 2013)</td>
<td>56</td>
</tr>
<tr>
<td>(Yang et al., 2012)</td>
<td>51</td>
</tr>
<tr>
<td>(de la Vara et al., 2013)</td>
<td>48</td>
</tr>
<tr>
<td>(Bernardi et al., 2013)</td>
<td>47</td>
</tr>
<tr>
<td>(Bernardi et al., 2011)</td>
<td>43</td>
</tr>
<tr>
<td>(Saadatmand et al., 2011)</td>
<td>42</td>
</tr>
<tr>
<td>(Brosch et al., 2011)</td>
<td>41</td>
</tr>
<tr>
<td>(Yang et al., 2010)</td>
<td>34</td>
</tr>
<tr>
<td>(Giese &amp; Schäfer, 2010)</td>
<td>31</td>
</tr>
<tr>
<td>(Marrone et al., 2015)</td>
<td>31</td>
</tr>
<tr>
<td>(Leitner &amp; Leue, 2011)</td>
<td>31</td>
</tr>
<tr>
<td>(Pham et al., 2011)</td>
<td>30</td>
</tr>
<tr>
<td>(Montecchi et al., 2011)</td>
<td>27</td>
</tr>
<tr>
<td>(Rodriguez et al., 2010)</td>
<td>26</td>
</tr>
<tr>
<td>(Merseguer &amp; Bernardi, 2013)</td>
<td>26</td>
</tr>
<tr>
<td>(Marrone et al., 2014)</td>
<td>26</td>
</tr>
<tr>
<td>(Aziz &amp; Rashid, 2016)</td>
<td>25</td>
</tr>
<tr>
<td>... 174 more</td>
<td></td>
</tr>
<tr>
<td></td>
<td>under 25</td>
</tr>
</tbody>
</table>

... 174 more citations with under 25 citations.
# Influence on our own work: Supervised theses

<table>
<thead>
<tr>
<th>Student</th>
<th>Thesis Title</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhao Zhao, M.A.Sc., 2014</td>
<td>Automatic derivation of Fault Trees from UML models</td>
<td>Develops an ATL transformation from UML (Use case, Composite Structure Dgr and Sequence Dgr) extended with MARTE+ DAM) into Fault Tree Models.</td>
</tr>
<tr>
<td>Paul Devi Deji, M.A.Sc., 2016</td>
<td>Derivation of Failure Mode and Effects Analysis (FMEA) from UML Software Model</td>
<td>Develops a transformation in Epsilon that takes as input an UML model annotated with failure mode info and produces an FMEA model.</td>
</tr>
<tr>
<td>Bashar AlShboul, Ph.D., 2019</td>
<td>Safety Analysis of SysML Models in the Context of Model-Driven Engineering</td>
<td>Develops a pattern-based transformation in Epsilon which transforms SysML+ MARTE+DAM into Fault Tree models and feeds back the results.</td>
</tr>
</tbody>
</table>
## Influence on our own work: Publications

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Book: 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Journal papers: 10</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Influence on our own work: Journal publications

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Contributions</th>
</tr>
</thead>
</table>
Influence on our own work: Journal publications

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Contributions</th>
</tr>
</thead>
</table>

Conference papers: 19 (not listed)
Tools supporting DAM

- **MASDES**: [https://bitbucket.org/masdesgroup/masdes/wiki/Home](https://bitbucket.org/masdesgroup/masdes/wiki/Home)
  - First version of DAM implemented as Eclipse plugin

- **DICE Simulation**: [https://github.com/dice-project/DICE-Simulation](https://github.com/dice-project/DICE-Simulation)
  - Developed within the EU project "Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements" (DICE).
  - Tool Functionalities:
    - Annotation of UML diagrams with system quantitative properties for performance and reliability, based on UML.MARTE and DAM profiles.
    - Determines *performance metrics* (response time, throughput and resource utilization) and *reliability metrics* (MTTF, availability, reliability, prob. of failure) on a UML scenario that represents a system execution.
    - Specializes DAM for platform-dependent data-intensive applications: Apache Hadoop, Spark, Storm and Tez.
    - Performs *what-if* or *sensitivity* analysis for performance and reliability metrics. The user can see multiple output results (e.g., as plot charts).
    - Transforms a UML scenario into stochastic Petri nets.
Use of DAM in influenced work

- Use/extend DAM to annotate different dependability NFPs:
  - reliability
  - availability
  - safety
  - fault tolerance
  - maintainability
  - survivability
  - security
  - privacy (incipient work)

- Different software and dependability models in influenced work:
  - Software/system models:
    - UML (class, composite structure, use case, deployment, state machine, seq., activity)
    - SysML (block definition diagram BDD, internal BD, state machine)
  - Dependability analysis models:
    - Different kind of Stochastic Petri Nets: DSPN, GSPN.
    - Stochastic Reward Networks (SRN)
    - Fault trees, Component Fault trees, Stochastic Fault trees
    - FMEA models
Future Challenges
Integrating the analysis of multiple NFPs in the MDE process

Software modeling domain

- $M$ Software model with $NFP_1$ to $NFP_n$ annotations
- Presenting feedback in model context
- Prepare feedback to developers

Transformation to analysis model

Mapping results to software domain

NFP$_n$ domain

NFP$_1$ domain

- $A_1$ $NFP_1$ analysis model
- Solving $NFP_1$ analysis model
- $R_1$ $NFP_1$ analysis results
Analysis models can be integrated

Example:
A: performance
B: reliability

A+B: performability

Work using this approach:
Analysis models cannot be integrated

- Using different formalisms for the NFP analysis models
- Example of NFPs to be analyzed: \(\text{(performance, reliability, safety, cost)}\)
  \[\text{QN, DSPN, FaultTree}\]

- The NFP models cannot be integrated in a single analysis model

- Modeling artifacts:
  - \(\mathcal{M}\) : Software system model with \(NFP_1\) to \(NFP_n\) annotations
  - \(\mathcal{D}_{i} \in \mathcal{M}\) : subset of diagrams in \(\mathcal{M}\) with annotations for \(NFP_i\)
  - \(\mathcal{A}_i\) : Analysis model for \(NFP_i\) (obtained from the subset \(\mathcal{D}_{i}\))
  - \(\mathcal{T}_i\) : Transformation from the source model \(\mathcal{D}_{i}\) to the target model \(\mathcal{A}_i\)
  - \(\mathcal{S}_i\) : Solver for \(\mathcal{A}_i\)
  - \(\mathcal{R}_i(\mathbf{X}_{ij})\) : Results produced by \(\mathcal{S}_i(\mathbf{X}_{ij})\) for the set of input parameters \(\mathbf{X}_{ij}\), which corresponds to the \(j^{th}\) data point for which model \(\mathcal{A}_i\) is solved.

  - Challenge: select the data points for which to solve each model
  - Visualize the results in a user-friendly way (e.g., as charts).
Design space exploration

Research challenge: how to use multiple NFP analysis models to find good design solutions (preferably optimal)?

1. The problem of balancing multiple NFPs lends itself to multi-criteria optimization
   - Complexity and huge design state space -> intractable solution
   - Traditional optimization methods used mostly when a single NFP analysis model is required
   - If multiple NFP are required, use metaheuristics search techniques
     ♦ genetic algorithms, simulated annealing

2. Another approach for balancing different NFPs is using decision support systems for reasoning under uncertainty
   - decision support systems based on Bayesian Belief Network models

3. Diagnostic and design-change rule-based techniques are used to find good design solutions when a single NFP is considered at a time.
Standardization

- DAM is an extension of the MARTE profile standardized by OMG
  - Purpose: to add dependability annotations to UML and/or SysML software/system models
  - Compliance with OMG standards allows for using standard UML tools.
- MARTE versions:
  - Version 1.0: November 2009
  - Version 1.1: June 2011
  - Version 1.2: April 2019
  - Major revision in progress:
    - MARTE 2.0 RFI: issued September 2008
    - RFI Response: August 2019
- Future work: to align DAM with the new version MARTE 2 when it will be made public.
Experience in conducting model-driven NFP analysis in the context of MDE shows that there still are several challenges:

- **Human qualifications**
  - Software developers are not trained in all formalisms used for NFP analysis -> this leads to idea of hiding analysis details from developers.
  - However, the software models must be annotated with extra info for each NFP, and the analysis results must be interpreted in order to improve the designs.
  - Challenge: find a better balance between what to hide and to expose.

- **Consistent model evolution**
  - The analysis of different NFPs may require source models at different levels of abstraction/detail.
  - Changing diagrams $D_i \in M$ for $NFP_i$ may affect other NFP models.
  - Challenge: to keep all the models/views consistent while making design changes to improve the NFPs.
Conclusions (2)

- **Tool interoperability**
  - Difficult to interface and to integrate seamlessly different MDE tools and NFP analysis tools, which were created at different times with different purposes and maybe running on different platforms.

- **Software process**
  - Integrating the analysis of different NFP raises process issues.
  - For each NFP should explore the state space for different design alternatives, configurations, workload parameters.
  - How to compare different solution alternatives that may improve some NFPs and deteriorate others, and how to decide on trade-offs.

- **Propagation of changes through the model chain**
  - Currently, every time the software design changes, a new analysis model is derived from scratch to redo the analysis.
  - Challenge: to develop incremental transformation methods and to keep different models synchronized.