System Dependability Evaluation using AADL (Architecture Analysis and Design Language)*

Ana – Elena Rugina
LAAS-CNRS
7 avenue Colonel Roche
31077 Toulouse Cedex 4, France
aerugina@laas.fr

Abstract

In the context of an increasing complexity of new-generation embedded real-time systems, the work presented in this paper aims at facilitating the evaluation of dependability measures of prime importance, such as reliability or availability. To fulfil this objective, our work focuses on defining a modelling framework allowing the automatic generation of dependability-oriented analytical models from high-level AADL models that are easier to handle for users. This paper presents a stepwise approach for system dependability modelling and evaluation, using AADL and GSPNs (Generalised Stochastic Petri Nets). The AADL dependability models are built on the architecture skeleton by using features of the AADL Error Model Annex, a draft annex to the AADL standard. The modelling and evaluation approach is illustrated on a simple example.

1. Introduction

In order to remain competitive with regards to costs and delays, the European real-time embedded systems industry must solve crucial problems related to the increasing complexity of new-generation systems. These problems are addressed in the FP6 European Integrated Project ASSERT (***A**utomated proof based **S**ystem and **S**oftware **E**ngineering for **R**eal-**T**ime applications*) coordinated by the European Space Agency [4]. This project aims mainly at i) identifying reference architectures for different system families, ii) replacing the classical system engineering approach by a proof-based method and iii) demonstrating the validity of the newly introduced concepts on real industrial case studies. In this context, high guarantees on the dependability properties are required at lower costs. Mature dependability-oriented analytical modelling techniques do exist ([1], [3], [6]). They are mainly based on the use of Petri nets and Markov chains. Existing tools support the analysis of such analytical models. However, analytical modelling techniques require substantial amount of training to be used effectively. On the other hand, description languages such as UML (Unified Modelling Language) and AADL (Architecture Analysis and Design Language) have emerged. They are more and more extensively used by industry. In the context of the ASSERT project, we aim at developing a modelling framework allowing the automatic generation of dependability-oriented analytical models from high-level AADL architecture models. This approach is meant to hide the complexity of analytical models to the end-user and, in this way, to facilitate the evaluation of dependability measures, such as reliability, availability and maintainability.

The remainder of the paper is organised as follows. Section 2 presents possible links between AADL and dependability-oriented analytical modelling techniques. Section 3 is an overview of our stepwise approach for system dependability modelling and evaluation, using AADL. Section 4 illustrates our approach on a simple example and section 5 concludes the paper.

2. AADL and analytical modelling

System analysis using AADL [8] can reveal the impact of different architecture choices such as scheduling policy or redundancy scheme on the system's

* This work is partially supported by 1) ASSERT (**A**utomated proof based **S**ystem and **S**oftware **E**ngineering for **R**eal-**T**ime applications) - European Integrated Project No. IST 004033. www.mayeticvillage.com/assert and 2) the European Social Fund.

architecture [5]. An architecture specification in AADL describes how components are combined in sub-systems and how they interact. Architectures are described hierarchically.

AADL is a core language that can be extended. Extensions can be analysis-specific notations that are associated to components. This is the case of the *AADL error models*. AADL error models are described in the *AADL Error Model Annex*, which was created by the AADL Working Group. This document is still a "work in progress"[1]. It is to be published together with the next version of the AADL standard and it is intended to support qualitative and quantitative analysis of dependability attributes. The AADL Error Model Annex defines a sub-language that can be used to declare error models within an error annex library. The AADL architecture model serves as a skeleton for the error models as they can be associated to AADL components. They describe the behaviour of the components to which they are associated in presence of internal faults and repair events, as well as in presence of external propagations from the component's environment. An architecture specification containing error models provides a dependability-centered view of the system and may be subject to a variety of analysis methods. Classical dependability models such as fault trees or Markov chains can be generated as specified in the AADL Error Model Annex itself. Unlike Markov chains, fault trees are not appropriate for modelling real-life systems exhibiting stochastic dependencies that result for example from error propagations between components. The AADL Error Model Annex does not mention possible generation of (Stochastic / Time) Petri nets. A dependability model under the form of Generalised Stochastic Petri Nets (GSPNs) has the advantage to allow structural verification before deriving the Markov chain from which the dependability measures are evaluated. Also, it is widely recognised that GSPNs facilitate the generation of complex Markov chains characterising the behaviour of real-life systems. Our research objective is to develop a modelling approach allowing GSPN models to be automatically derived from AADL models.

As stated in the introduction, we propose a stepwise approach for system dependability modelling and analysis using AADL. The ultimate aim is to evaluate quantitative dependability measures. In the next section we summarise this approach, which is then applied to a simple example.

3. Overview of the modelling approach

This approach supposes that a description of the system to be analysed is available. The system description must contain i) its structure, ii) its functional behaviour and iii) its behaviour in presence of faults. Interactions between architectural components of the system must be analysed at this stage, as such interactions induce dependencies between components and consequently between their models.

An overview of our modelling approach, which is composed of four main steps, is illustrated in Figure 1 and it is more detailed hereafter.

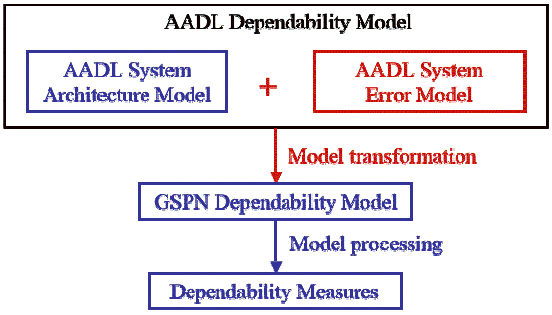

Figure 1: General approach

The ***first step*** is devoted to the modelling of the system architecture in AADL (i.e., its structure in terms of components and operational modes of these components). Sometimes the AADL system architecture is already available, as it may have been already built for other analyses.

The ***second step*** concerns the modelling of the behaviour of the system in presence of faults through AADL error models associated to components of the AADL architecture model. The set of error models associated to components of the architecture forms the AADL system error model. In order to master the complexity and the evolution of the system error model, this second step is incremental and consequently, multi-phased. More concretely, in a first phase we model the behaviour of each component, as if it were isolated from its environment, in presence of its own faults and repair events. Then, dependencies are modelled in an incremental manner. In this way, the final model represents the behaviour of each component not only in presence of its own faults and repair events, but also in its environment, i.e., faults and repair events in components with which it interacts.

The ***third step*** aims at constructing a global analytical dependability model that can be processed by existing tools. The information that is necessary to the generation of an analytical dependability model is extracted from the AADL dependability model. The global analytical dependability model is generated in the form of a Generalised Stochastic Petri Net (GSPN) by applying model transformation rules. Already existing dependability analysis tools can then process the GSPN. Note that this third step can also be incremental; as it is

[1] Copies of the draft AADL Error Model Annex can be asked by e-mail to info@aadl.info.

possible to enrich the global analytical model each time the second step is iterated. In this way, the GSPN model can be validated progressively using classical methods and tools. So, if validation problems arise at GSPN level during phase ***i***, only the part of the current AADL error model corresponding to phase ***i*** is questioned. It is worth stressing that in the case of an isolated system or in the case of a set of systems considered to be independent, the AADL to GSPN transformation is rather straightforward. However, the transformation becomes complex in the case of realistic systems formed of dependent components as shown in [7]. Also, some of the problems linked to the relationship between *abstract* and *concrete* stochastic automata models obtained from AADL error models have been mentioned in [2].

The ***fourth step*** is devoted to the GSPN model processing that aims at obtaining dependability measures. We stress that this fourth step is entirely based on classical GSPN processing algorithms and existing tools. This step includes both i) syntactic and semantic validation of the model and ii) evaluation of quantitative dependability measures.

4. Example

This section illustrates our approach on a simple example. A more realistic one is presented in [7]. The system considered here is formed of two communicating software components. One of them is considered to be completely dependent on the other one. The system is described as follows.

- *structure*: two software components linked in order to allow transfer of data from one to another;
- *functional behaviour*: every component has only one operational mode;
- *behaviour in presence of faults*: every component can be either error free, or failed. The dependent component fails if the other component fails. The components are restarted independently.

4.1. First step - AADL architecture model

Figure 2 shows the AADL architecture model of the system described above. Two AADL components (S1 and S2) of type *system* are linked through a unidirectional port connection, as the data transfer is considered unidirectional. The behaviour in presence of faults will be described in the second step by error models associated to each component.

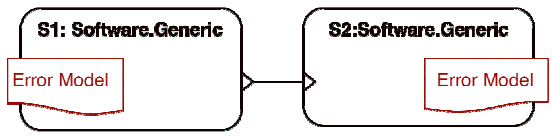

Figure 2: AADL architecture

4.2. Second step - AADL error models

An error model is specified under the form of one *error model type* and one or more *error model implementations,* declared to be suitable for different dependability analyses. The error model type declares error states, events and propagations. Error model implementations declare transitions between error states, as well as stochastic characteristics of error events and out propagations. A simple error model that can be associated to both AADL components is given in Error Model 1.

```
error model forSoftware
features
-- Phase 1
Error_Free:initial error state;
Failed: error state;
Fail, Restart: error event;
-- Phase 2 (inter component dependency)
Software_KO: in out error propagation;
end forSoftware;
```

```
error model implementation
forSoftware.Basic
transitions
-- Phase 1
Error_Free-[Fail] -> Failed;
Failed-[Restart] -> Error_Free;
-- Phase 2 (inter component dependency)
Error_Free-[in Software_KO] -> Failed;
Failed-[out Software_KO] -> Failed;
properties
-- Phase 1
occurrence => poisson 10e-4
           applies to Fail;
occurrence => poisson 5
           applies to Restart;
-- Phase 2 (inter component dependency)
occurrence => fixed 1 applies to Software_KO;
end forSoftware.Basic;
```

Error Model 1: Simple error model

The error model type *forSoftware,* from Error Model 1, specifies two error states: *Error_Free* (the initial state) and *Failed*, two error events: *Fail* and *Restart*, and one in out error propagation *Software_KO*. The error model implementation *forSoftware.Basic,* from the same Error Model 1, declares transitions between the states declared in the error model type *forSoftware*. Transitions are triggered by error events and propagations (named between right brackets between the source and the destination state). The error model implementation *forSoftware.Basic* associates occurrence properties to error events (*Fail* and *Restart* follow Poisson

distributions) and propagations (*Software_KO* occurs with a probability of 1).

This step is two-phased: error states and error events (with associated stochastic properties) are declared together with transitions triggered by these events in a first phase. The propagation *Software_KO* together with its stochastic property and with the transitions that it triggers is introduced in a second phase to explicit the unidirectional dependency from one software component to the other one, as highlighted in Error Model 1.

4.3. Third step - AADL model transformation

As the previous step, this third step is two-phased. Error states and transitions triggered by error events are transformed respectively into places and transitions of the Petri net in a first phase. Transitions triggered by error propagations are transformed in a second phase. Also, sub models obtained from the error models associated to the two AADL components are merged. In a general case, the sub model composition is a rather complicated task. However, in this simple example, the composition is done by matching the *Software_KO* out propagation from the error model associated to component S1 to the in propagation *Software_KO* from the error model associated to component S2. The resulting GSPN is shown in Figure 3. *Blocks S1* and *S2* correspond to the AADL sub models for the two software components. The *interface block* describes the interaction between these two components.

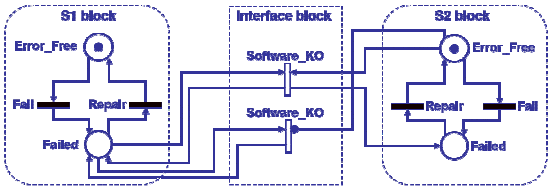

Figure 3: GSPN model

4.4. Fourth step – model processing

This step is not detailed here as it is supposed to be completely automated by using existing analytical model processing tools proven to be efficient (i.e., SURF2 - www.laas.fr/surf/surf.html).

5. Conclusion

This paper presented a stepwise approach for system dependability modelling and evaluation using AADL. The aim of this approach, which was illustrated on a simple example, is to ease the task of evaluating dependability measures, by hiding the complexity of classical analytical models to the end-user. Our approach has two main characteristics: i) it is incremental, as it needs to support and trace model evolution and ii) it is based on model transformation, from AADL dependability models (architecture + dependability-related information) to GSPNs that can be processed by existing tools.

After having defined the approach, the main purpose of the work carried out until now was to assess its feasibility. So, we applied it to a complex enough case study, presented in [7]. The next step of the work concerns the formalisation of transformation rules in order to automate model transformation.

Acknowledgements

I would like to thank my research advisors, Karama Kanoun and Mohamed Kaâniche, for their support and assistance.

References

[1] C. Betous-Almeida and K. Kanoun, "Construction and stepwise refinement of dependability models", *Performance Evaluation*, 56 (1-4), pp.277-306, 2004.

[2] P. Binns and S. Vestal, "Hierarchical composition and abstraction in architecture models", in *18th IFIP World Computer Congress, ADL Workshop,* (Toulouse, France), pp.43-52, 2004.

[3] A. Bondavalli, I. Mura and K. S. Trivedi, "Dependability Modelling and Sensitivity Analysis of Scheduled Maintenance Systems", in *3rd European Dependable Computing Conference (EDCC-3)*, (Prague, Czech Republic), pp.7-23, Springer, 1999.

[4] E. Conquet and P. David, "Preparing the System and Software engineering of the 21st century for critical systems with the ASSERT project", in *Fifth European Dependable Computing Conference, Supplementary Volume,* (Budapest, Hungary), pp.27-32, 2005.

[5] P. H. Feiler, D. P. Gluch, J. J. Hudak and B. A. Lewis, "Pattern-Based Analysis of an Embedded Real-time System Architecture", in *18th IFIP World Computer Congress, ADL Workshop,* (Toulouse, France), pp.83-91, 2004.

[6] K. Kanoun and M. Borrel, "Fault-tolerant systems dependability. Explicit modeling of hardware and software component-interactions", *IEEE Transactions on Reliability*, 49 (4), pp.363-376, 2000.

[7] A. E. Rugina, K. Kanoun, M. Kaâniche and J. Guiochet, *Dependability modelling of a fault tolerant duplex system using AADL and GSPNs*, LAAS-CNRS, N°05315, 2005.

[8] SAE-AS5506, *Architecture Analysis and Design Language*, Society of Automotive Engineers, 2004.