A Survey On Verification And Analysis Of Non-Functional Properties Of AADL Model Based On Model Transformation

Xu Biao[1, a *], Lu Minyan[1, b]

[1]BeiHang University School of Reliability and Systems Engineering

Science and Technology on Reliability and Environmental Engineering Laboratory

Beijing, China

[a]xxbbzxc@126.com, [b]lmy@buaa.edu.cn

*Corresponding author, Email: xxbbzxc@126.com

Keywords: Model Transformation; AADL Model; Non-Functional Properties; Verification; Analysis

Abstract. Architecture Analysis and Design Language (AADL) is an architecture description language that has been adopted as an industry standard for embedded real-time systems by the International Society for Automotive Engineers (SAE) in 2004.To meet the need of verification and analysis of the AADL models, model transformation technologies are generally used to automatically extract a formal specification suitable for verification and analysis.

This paper surveys the research and practice of verification and analysis of non-functional properties of AADL Model based on model transformation, presents a discussion from different perspectives of the state of the art, identifies open issue and presents some future work directions in the field.

Introduction

Widely used in domains such as avionics and aerospace platforms, embedded real-time systems with features like limited-resource, real-time response, fault-tolerance and hardware-special have to face higher demands of timeliness and reliability. Those systems were named performance-critical systems[1]. Moreover, they are becoming more and more complex, so reducing both the cost and time of development becomes a significant problem that academia and industry face[2].

The Architecture Analysis and Design Language (AADL)[3] is an architecture description language that has been adopted as an industry standard for embedded real-time systems by the International Society for Automotive Engineers (SAE) in 2004. AADL employs formal modeling concepts for the description of software/hardware architecture and run-time environment in terms of distinct components and their interactions, and it is especially effective for model-driven design of complex embedded real-time systems[4].

All systems shall pass qualification and certification processes before their deployment especially those real-time systems (for example DO-178C). When such a system specification is described using an AADL model,it’s often transformed into another formal language/model for verification and analysis. Examples of such transformations are numerous: translations to Behavior Interaction Priority (BIP) [5], to TLA+ [6], to IF [7],to real-time process algebra ACSR [8], to Fiacre [9], to Real-Time Maude [10],to Polychrony [11], to Lustre [12],to TASM [13] etc. The goal of such a transformation is to reuse existing verification and analysis tools and to validate the AADL models.

This paper surveys the research and practice of verification and analysis of non-functional properties of AADL Model based on model transformation, presents a discussion from different perspectives of the state of the art, identifies open issue and presents some future work directions in the field.

The rest of the paper is organized as follows. Section 2 surveys 36 proposals from 51 papers, ordering according to their presentation. Section 3 presents a discussion from different perspectives of the state of the art. Section 4 identifies open issue and presents some future work directions in the field.

Proposals for verification and analysis

During the last decade, embedded real-time system modeling for verification and analysis has received a growing interest. In particular, significant efforts have been focused on defining architecture models (generally based on AADL) and transformations of architecture models into different types of formal modeling languages or analysis models used to verify or to analysis non-functional properties.

In this section, we consider 36 representative proposals that have been developed in a total of 51 papers: Table 1 lists the papers that gather the proposals, ordered according to their presentation. These proposals are all based on AADL.

Table 1 List of surveyed proposals

Properties	Approach name	Papers
Behavior Verification	Bjornander	[14]
Behavior Verification	Rolland	[6]
Behavior Verification	Bomel	[15]
Behavior Verification	Berthomieu	[9]
Behavior Verification	Bodeveix	[16]
Behavior Verification	Yang	[17] [18] [13]
Behavior Verification	Belala	[19]
Behavior Verification	Cherfia	[20]
Behavior Verification	Jahier	[12] [21, 22]
Behavior Verification	Niz	[23]
Behavior Verification	Monteverde	[24]
Behavior Verification	Zheng	[25]
Behavior Verification	Liu Bo	[26]
Schedulability	Chkouri	[5] [27]
Schedulability	Sokolsky-a	[8] [28]
Schedulability	Zhou	[29]
Schedulability	Hugues	[30]
Schedulability	Liu Qian	[31]
Schedulability	Li Zhensong	[32]
Timing	Yu	[33-35] [36]
Timing	Ma	[33-35] [37]
Timing	Renault	[38] [39]
Timing	Sokolsky-b	[40]
Timing	Abdoul	[7]
Timing	Varona-Gomez	[41]
Dependability	Rugina	[42-44] [45]
Dependability	Dong	[46]
Dependability	Gao	[47]
Dependability	Yang Zhiyi	[48]
Dependability	Baudali	[49]
Dependability	Sun	[50]
Dependability	Bozzano	[51] [52]
Dependability	Johnsen	[53]

Dependability	Olveczky	[10]
Dependability	Bae	[54]
Dependability	Liu Wei	[55]

Behavior Verification. All the 13 proposals surveyed in this section, aim at providing support to a behavior verification of embedded real-time systems. Some proposals also support other kinds of verification or analysis.

Bjornander's proposal [14] presents a verifier of systems behaviors called ABV (AADL and the Behaviors Annex Verifier), using which a subset of the AADL models and their Behavior Annex are converted into CTL (Computation Tree Logic) and then verified.

Rolland's proposal [6] transforms AADL models into TLA+. TLA+ can only express discrete time, and the only tool support, TLC, is not very efficient.

Bomel's proposal [15] firstly converts AADL models into ATL, and then the ATL can automatically generate SystemC models which support simulation.

Berthomieu's proposal [9] translates a synchronous subset of AADL and the behavior annex into Fiacre, and then the Fiacre model is compiled into Timed Petri Nets (TPN) for verification. The paper did not explain the semantics, but just presented the transformation principle. Based on Berthomieu's proposal, Bodeveix's proposal [16] considers the semantics, and transforms an AADL synchronous subset into Fiacre. This proposal also provides transformation rules. The two methods are both integrated into TOPCASED (AADL integrated development tool).Yang has some cooperation with Berthomieu and Bodeveix in the former proposals, and his proposal firstly formalizes some subsets of the AADL (such as Behaviors Annex [17] and mode conversion [18]) using TASM (Timed Abstract State Machines), and then presents a translation of AADL into TASM and a methodology to prove the semantics preservation under the assumption that the reference semantics expressed in TTS are correct [13].

Belala's proposal [19] translates an AADL subset into Real-Time Maude, which could be used to verify timing requirements and behavioral attributes.

Cherfia's proposal [20] uses BRS (Bigraphical Reaction System) model to formalize AADL system dynamic architecture reconstruction and analysis embedded real-time systems.

Jahier's proposal [12] includes transformation of a subset of AADL except behavior annex, and the transformation is in the form of synchronous program language Lustre. In this work, AADL is translated into Polychrony which focuses on the multiple partition structure of the embedded architecture to simulate and verify GALS (globally asynchronous locally synchronous) model got from AADL specifications. In two other papers [21, 22], they transforms AADL model into BIP and PCP at the same time to analysis the complex scheduling policy behavior of shared resources.

Niz's proposal [23] transforms AADL concurrent execution semantics into Alloy, then defines an AADL attribute set which contains the sorting, reliability and fault tolerance, to analyze the system behavior by comparing the AADL and Alloy model.

Monteverde's proposal [24] proposes Visual Timed Scenarios (VTS) as a graphical language used in AADL behavior property specification, which is then transformed into Timed Petri Net (TPN). So model checking of attributes expressed in VTS could be done based on TPN tools.

Zheng's proposal [25] uses MDE heterogeneous model transformation framework to transform AADL model into Interface Automaton (IA),then verifies the compatibility of IA using formal method, constructs components compatible operation environment using IA tools, maps the environment to AADL components to solve the problem of compatibility of behaviors of AADL component combination.

Liu Bo's proposal [26] verifies AADL behavior model using a formal verification tools calls NuSMV, but the deficiency of MuSMV itself limits its application.

Schedulability. All the 6 proposals surveyed in this section, aim at providing support to schedulability analysis of embedded real-time systems. Some proposals also support other kinds of verification or analysis.

Chkouri's proposal [5] transforms most AADL concepts into BIP language which is a diversified real time modeling framework, including three levels, low-level describe the behavior, the middle-level describes interactions and the top-level describes scheduling policy. BIP provides two kinds of verification, using Aldebaran to detect deadlock and observers to verify some simple time performance. In another paper [27] this idea is extended, new approach generates one executable model from AADL and another from BIP based on network communication protocol, so we can simulate the distributed system described by AADL and verify it formally based on verification technology of BIP at the same time.

Sokolsky's proposal [8] mainly focuses on the schedulability analysis of AADL models, a smaller subset (modes and the behavior annex are excluded) is translated into the real-time process algebra ACSR, and use the ACSR-based tool VERSA to explore the state space of the model, looking for violations of timing requirements. ACSR can express the notation of resource explicit in system models, but focuses on single processor environment. So Sokolsky extends the application of ACSR [28], presents two tools which support AADL simulation respectively to track the resource utilization and analyze the schedulability to determine whether the resources meet the time constraints or not. A subset of the asynchronous

Zhou's proposal [29] maps an asynchronous subset of AADL into TASM to support the resource consumption and schedulability analysis.

Hugues's proposal [30] presents a tool named Ocarina, which can transform AADL model into Petri Nets to do formal verification, do schedulability analysis based on Cheddar and generate Ada and c code automatically.

Liu Qian's proposal [31] uses model checking tool UPPAAL to formally verify and analyze the schedulability of thread components of AADL model with non-preemptive scheduling police, and presents the model transformation tool which could transform AADL model into UPPAAL model.

Li Zhensong's proposal [32] carries out the verification and analysis of AADL behavior model. First, it presents model transformation rules from AADL to UPPAAL TASM based on Behavior Annex and its behavior description, designs and implies the prototype tool for model transformation. Then simulate the automatic model get from automatic transformation rules in UPPAAL to validate the automatic model's behavior and prove the availability of the model transformation.

Timing. All the 6 proposals surveyed in this section, aim at providing support to timing analysis of embedded real-time systems. Some proposals also support other kinds of verification or analysis.

Ma and Yu's proposal [33-35] expresses functional behavior with synchronous data flow model and uses Simulink/Gene-Auto for modeling ;distribute hardware architecture is modeled with AADL ;bridges the two different types of models with SME/Polychrony based on computing polychromous. They present the transformation rules from Simulink and AADL to SME model, and then simulate and analyze the timing properties in the framework of Polychrony. This approach can simulate time constraints in early phases. Yu [36] extends this idea based on previous works. The new method changes AADL semantics based on computing polychronous model and then maps Polychrony to SynDEx to ensure the consistency of timing semantics between different formal models. Finally, real-time schedulability analysis and the optimum configuration of multi-processor are done based on SynDEx. In another paper, Ma [37] transforms AADL model into synchronous language SIGNAL, to analyze and verify the timing with the Polychrony tools.

Renault's proposal [38] extends the approach transforms AADL into Symetric Nets (SN) [39] to transforms AADL into Timed Petri Net (TPN) and verify timing properties. Verification based on SN could be reference to validate these properties.

Sokolsky's proposal [40] extracts analysis model based on Real-Time Calculus, which supports performance analysis and end-to-end timing analysis.

Abdoul's proposal [7] translates a behavioral subset, minus mode changes, to IF, to analyze some safety properties of the system. The behaviors are not expressed using AADL Behavior Annex, but their own behavioral language.

Varona-Gomez's proposal [41] uses AADS tool to transform AADL model into SystemC model, and then uses SCoPE tool for performance, and other NFPs (such as CPU occupation, timing, energy consumption) analysis.

Dependability. All the 11 proposals surveyed in this section, aim at providing support to dependability analysis of embedded real-time systems. Some proposals also support other kinds of verification or analysis.

The original definition of dependability is the ability to deliver service that can justifiably be trusted. As pointed out by [56], this definition stresses the need for justification of trust. An alternative definition states the dependability of a system as the ability to avoid failures that are more frequent and severe than acceptable. In this case the definition provides the criterion for deciding if the service is dependable. According to [paper], dependability encompasses five attributes :reliability, the continuity of correct service ;availability, the readiness for correct service ;maintainability, the ability to undergo modifications and repairs ;integrity, the absence of improper system alterations ;safety, the absence of catastrophic consequences on the users and environment.

Rugina's proposal [42-44] annotates AADL architecture model with dependability related information through standard Error Model Annex, such as failure, failure mode, maintenance strategy, error propagation, etc. It also provides a set of transformation rules, which can automatically transform AADL model into Generalized Stochastic Petri Net (GSPN) used in the dependability analysis. These rules are applied in ADAPT tool [45] which connect to the open-source tool OSATE which provides AADL tool support; SURF - 2 at the same time provides GSPN tool support. There are domestic similar researches like [46-48] also use GSPN on dependability analysis.

Baudali's proposal [49] presents an extendable formal reliability evaluation framework: Arcade (ARChitecturAl Dependability Evaluation), which supports a wide range of modeling languages including AADL. This proposal transforms the input model into I/O-LMC (Input/Output Interactive Markov Chains) model, and further transforms the I/O-IMC model into CTMC (Continuous Time Markov Chains).Finally, reliability analysis shall be made upon CTMC based on the CADP tools, at the same time this approach also supports compositional analysis, so it could support complex systems.

Sun's proposal [50] combines AADL model with Fault Tree Analysis (FTA) to make reliability and safety analysis.

Bozzano's proposal [51] uses event-data automata and probabilistic finite-state machine to define the formal semantics of AADL and the Error Model Annex. Then, it integrates the normal behavior and error behavior by constructing two formal models, the formal semantics of the formal models support multiple types of qualitative and quantitative system analysis, including the verification of the dependability properties. It also develops integrated toolset COMPASS [52].COMPASS gets the information of models specified by formal semantics and the set of attributes needed to be verified and generates FMEA table and stochastic fault tree. COMPASS verifies system properties based on symbolic model checking technique.

Johnsen's proposal [53] supports automatic fault avoidance of AADL model and verifing the integrity and consistency of model. It formalizes a subset of AADL through semantic anchoring, and presents corresponding transformation rules to transform the subset into TASM, finally verifies TASM using UPPAAL.

For safety-critical systems, Bae and Olveczky's proposal [10] transforms an AADL subset into Real-Time Maude, and presents AADL simulators and LTL model checking tool AADL2Maude for testing system safety-critical behaviors. Real-Time Maude is a formal real-time rewriting logic. In another paper [54] they presents Synchronous AADL Language, modeling synchronous real-time systems, and provides the synchronous AADL formal semantics in Real-Time Maude, integrates the semantics with the OSATE modeling environment. These works simplifies the verification process.

Liu Wei's proposal [55] the transformation from AADL model to the real-time system modeling framework BIP based on component, verifies the safety of AADL model indirectly through verifying

BIP model. And it's similar to other model transformation methods, which are simple, feasible and beneficial to check model's consistency or model reuse.

As complex real-time embedded systems become networked, modularized, systems may also be influenced by external attacks and malicious code. So security is getting concern. The ability of security modeling of AADL is still rudimentary, only supports simple tests for security level. Eby's proposal [57] makes it clear that almost no current modeling tools support the security modeling, and in most cases embedded system engineers are not aware of security problems. Generally, system vulnerability can be found only after being attacked. What's more, even with firewall and intrusion detection system, systems are hard to avoid the internal attack. So it is necessary to integrate security into the modeling environment at an early phase.

Discussion

In this section we discuss, from different perspectives, the state of the art in the area of NFPs' verification and analysis based on model transformation of AADL models, based on the contributions presented in last section. More specifically, we surveyed 36 approaches based on 51 papers, which are summarized in Table 1 in the order of their presentation. As mentioned before, all the approaches are using AADL as a system modeling language.

Software Engineering Criteria.

Life Cycle Phase. The support provided by the approaches within the software life cycle spans from the requirements to the deployment phase (Table 2); however, the most important contributions are given in the early phases, in particular during the system architecture, and design specification. This result is not surprising, as it occurred also for performance model-based approaches [58]. It is due to the fact that a major modeling effort is placed early in the life cycle, where the detection of both functional and non-functional (e.g., schedulability, dependability) problems can be fixed more effectively with less effort and lower costs that late fixes.

Table 2 Contributions by life cycle phase

Approach	Requirements	Architecture	Design	Impl./Deployment
Rolland		√		
Baudali		√		
Rugina		√	√	
Bozzano		√	√	
Dong		√	√	
Gao		√	√	
Yang Zhiyi		√	√	
Li		√	√	
Liu Qian		√		
Bodeveix			√	
Liu Wei		√		
Liu Bo	√	√	√	
Renault			√	
Johnsen		√		
Bjornander		√		
Chkouri			√	
Sokolsky			√	
Zhou		√		
Yu		√	√	

Bomel			√	
Abdoul	√	√	√	√
Varona-Gomez			√	
Berthomieu			√	
Yang		√		
Zheng			√	
Bae			√	
Olveczky			√	
Belala		√		
Ma			√	
Cherfia		√		
Niz		√		
Jahier	√	√	√	
Monteverde			√	

It's observed that a significant number of approaches aim at providing solutions for verification or analysis of embedded real-time systems, while quite fewer contributions aim at requirement elicitation or NFP specification (i.e., how to express dependability characteristics in the AADL model).

The implementation and deployment phase are addressed by only one approach (Abdoul); while none of the works we considered focus on testing phase. We suggest that research efforts should be devoted to combine model-driven approaches with experimental ones in the testing phase, e.g., by exploiting use case to drive the testing activities through test cases and to trace back the latter to NFP requirements.

Application Domain. Most of the works either focus on a specific software domain or provide specific support for embedded real-time systems. The majority of the surveyed works support verification or analysis of general software systems. We observe that the kind of NFP to be evaluated is influenced by the software and the application domains considered by a given work. In particular, considering in detail the application domain (Table 3), we notice that most of the works that address avionics, automate, and critical systems are interested in providing support for dependability analysis. On the other hand, in the case of real-time system applications, it is often desirable to guarantee the timing of service delivery, when requested by the end-user. Therefore, schedulability and timing are the main issues addressed by the works dealing with this type of applications.

Table 3 Application domain

Approach	general	avionics	automate	Real-time system	Critical system	communicate	aerospace	electronic
Rolland	√							
Baudali	√							
Rugina		√	√					
Bozzano		√	√					
Dong		√						
Gao		√						
Yang Zhiyi		√						
Li							√	
Liu Qian	√							
Bodeveix	√							

Liu Wei		√						
Liu Bo	√							
Renault				√				
Johnsen				√	√			
Bjornander					√			
Chkouri	√	√						
Sokolsky	√					√		
Zhou							√	
Yu	√	√						
Bomel								√
Abdoul	√							
Varona-Gomez	√							
Berthomieu		√					√	
Yang	√							
Zheng							√	
Bae		√						
Olveczky	√	√			√			
Belala	√							
Ma		√						
Cherfia	√							
Niz		√						
Jahier	√							
Monteverde	√							

Tool Support. All of the surveyed contributions provide tool support for the approaches they propose. However most of the tools are research prototypes that do not cover all the aspects or processes, the potential for building more powerful tool support exists. Many approaches could be automated since they propose either rigorous transformation techniques of AADL models into formal modeling languages or analysis models. Only a few proposals are difficult to implement or do not provide any indication of an existing implementation.

NFP Characteristics Criteria.

Attributes. We observe a significant number of works intent to verify system behaviors, and then they can further analyzing schedulability, timing, or to do reliability analysis. But fewer works pay attention to security. Verification of behaviors is tightly interrelated with schedulability or timely analysis, because the system schedulability and timing is generally associated with real-time states of system behaviors.

Analysis Types. One of the criteria for discussion is the type of NFPs analysis proposed, i.e., qualitative or quantitative. Qualitative analysis aims to verify systems, while quantitative analysis aims to compute NFPs measures. We notice that behavior verification, schedulability and timing contributions fall basically in the first category (i.e., qualitative) while the works that focus on dependability (except safety which is mainly analyzed qualitatively) belong to the second category (i.e., quantitative). There are also some exceptions that support both types of analysis.

Considering the approaches aimed at quantitative dependability analysis, the majority of them rely on stochastic (or probabilistic) assumptions.

Formal Modeling Language and Analysis Model. Table 4 summarizes the techniques adopted by the surveyed works to support NFPs verification or analysis of AADL-based specifications.

Table 4 Formal languages/models

Approach	TASM	Petri Net	CTL	Fiacre	GSPN	Maude	BIP	ACSR	FTA	FMEA	others
Rolland											√
Baudali											√
Rugina					√						
Bozzano									√	√	
Dong					√						
Gao					√						
Yang Zhiyi					√						
Li	√										
Liu Qian	√										
Bodeveix				√							
Liu Wei							√				
Liu Bo			√								
Renault		√									
Johnsen	√										
Bjornander			√								
Chkouri							√				
Sokolsky								√			√
Zhou	√										
Yu											√
Bomel											√
Abdoul											√
Varona-Gomez											√
Berthomieu		√		√							
Yang	√										
Zheng											√
Bae						√					
Olveczky						√					
Belala						√					
Ma											√
Cherfia											√
Niz											√
Jahier							√				√
Monteverde		√									

Quality Criteria.

Validation. The validation of the proposed methods is not a primary issue in the surveyed approaches; few of them even do not consider validation at all. However, when validation is considered, it is carried out mainly to show the applicability and/or the scalability of the method to realistic examples, i.e., through case studies. Only a few works (Li and Yang) conduct empirical analysis in an academic environment, to assess not only the applicability of the proposed approaches but also their correctness.

We suggest that those approaches providing support for quantitative NFPs analysis and missing the validation of the correctness of the proposed methods could be verified, e.g., by comparing the analysis results with results obtained from testing activities.

Presentation of Result. The majority of the approaches provide a basic support to present the results of the verification or analysis. The most common way is textual, followed by graphical and tabular presentation. We observed that the most promising approaches are those that feedback the results to the original AADL specification (Zheng); this makes the analysis process transparent to the software analyst.

However, more research is needed to address this issue. In particular, the problem of how to identify the critical elements of the system specification which are the cause of unsatisfactory NFPs is still open. Good solutions would provide useful information to the software engineers for changing the design accordingly.

Limitations. Almost all the surveyed approaches present limitations, as summarized in detail in Table 5.One of the more common deficiency is the lacking of direct feedback to the AADL models.

Table 5 Limitations

Approach	Limitations
Rolland	TLA+ can only express discrete time, and the only tool support TLC is not very efficient.
Baudali	The expression of system failure is simple, does not support stochastic model checking.
Rugina	
Bozzano	The levels of Fault Tree that COMPASS generates are few.
Dong	
Gao	
Yang Zhiyi	
Li	UPPAAL does not support preemptive scheduling.
Liu Qian	UPPAAL does not support preemptive scheduling, UCaS, the modified tool, is inefficient.
Bodeveix	The ability of model constructing is inefficient, does not support some attributes, such as resource.
Liu Wei	The platform special module (PSM) of AADL is hard to be described in BIP, which could describe the behavior of system platform, but cannot define the parameters of platform hardware of system in detail.
Liu Bo	
Renault	
Johnsen	The supported types of time and data are few, more subsets of AADL need to be formalized
Bjornander	Don’t support enough time description semantics.
Chkouri	The verification types supported by verification tools of BIP are few.
Sokolsky	Don’t support complex system, and the assumption of instant communicating is unreasonable.
Zhou	The supported types of time and data are few, more subsets of AADL need to be formalized
Yu	Don’t support round-trip transformation of SIGNAL and SynDEx
Bomel	

Abdoul	The behavior description language developed isn’t supported by any other tools
Varona-Gomez	
Berthomieu	The definition of logistic attributes is tedious; fault reports and verification process need to be improved.
Yang	The supported types of time and data are few, more subsets of AADL need to be formalized
Zheng	Don’t support compatibly operation of components whose behaviors are incompatible.
Bae	Maude does not support real-time description; the performance of tool is inefficient.
Olveczky	Maude does not support real-time description
Belala	Maude does not support real-time description
Ma	
Cherfia	
Niz	The verification speed is slow for Alloy’s limitation
Jahier	Can’t verify complex system (for states explosion);Supports simulation and testing but doesn’t support analysis and verification when AADL model including dispersal threads that Lustre doesn’t support
Monteverde	Timed Petri Net lack attributes of time divergence.

Discussion Summary. This section summarizes the conclusion of the discussion from the previous section and identifies open issues that emerged from the study.

Firstly, most of the works focus on behavior verification schedulability timing and dependability, and fewer efforts have been devoted to security modeling and analysis. Moreover, we have not found any work addressing specifically how to extend AADL with integrity NFP.

Secondly, the surveyed works provide support mainly in the early phases of the life cycle (i.e., architecture and design), while there is a lack of support for former or later phases, as, for example, for testing NFPs guided by selected use cases.

Thirdly, the contributions using model transformations mainly focus on obtaining formal modeling languages or analysis models which can be used for NFPs verification and analysis. Only a few go one step further to provide feedback from the analysis results to the original AADL model specification, in order to pinpoint the causes for requirement inconsistencies or design flaws.

Fourthly, it is also worth noticing that tool support and method validation are crucial factors to make an approach effective. Although the majority of the surveyed approaches are characterized by a high automation degree, most of them are not fully supported by software tools. Moreover, in many cases method validation consists only in applying the proposed method to a case study. Considering the approaches that provide support for quantitative NFPs analysis, the validation of the correctness of the proposed methods is in fact missing. More efforts should be devoted to the validation of the methods themselves.

Last but not least, more research work should be invested in providing a standard common AADL framework for the modeling and verification/analysis of several NFPs, in order to support the consistent specification of different NFPs and their relationships, as well as the trade-off analysis between different NFPs (such as performance and timing, security and dependability).

Advanced Open Issues

Experience in conducting model NFP verification and analysis based on model transformation shows that the domain is still facing a number of challenges.

Human qualifications. Software developers are generally trained in only a few of formal languages/models used for the analysis of different non-functional properties (NFPs),when some kind of formal languages/models are not known by developers, it leads to the idea that we need to hide some of the analysis details from developers. However, the AADL models have to be annotated with extra information for each NFP and the analysis results have to be interpreted in order to improve the designs. A better trade-off needs to be made between what needs to be hidden and what to be exposed to the software developer.

Round-trip NFP analysis. The concept of round-trip engineering means consistently refining a high-level model of a system into a lower-level model (forward engineering) and abstracting a low-level model into a higher-level one (reverse engineering). The round-trip concept can be applied also to the AADL-based NFP analysis, where the forward trip transforms an AADL model into a formal language/model, and the reverse trip produces feedback from the analysis results to the original AADL model. This raises the need for automated diagnosis tools specific to different formal languages/models. It also raises the need for better traceability links between corresponding system and analysis model elements. It also brings us to the next step: the need allow the designer to ask specific NFP questions in AADL terms (for instance, what is the total down time for a given component due to a certain use case) which should be translated automatically into a specific query in terms of the formal language/model used for analysis. Assuming that the formal model is Petri Nets, the query translation needs to map the dependability values related to the given AADL model elements to Petri Nets measures related to the corresponding places and transitions. Such queries should be supported by special model transformations.

Abstraction level. The analysis of different NFPs may require different views of the AADL models at different levels of abstraction/detail. The challenge is to keep all these views consistent not only at the beginning, but also throughout the development process, when changes are applied to the model in order to add new features or to improve different properties.

Semantic extension. Dependability analysis of AADL model is mainly dependent on the Error Model Annex (EMA), but analysis merely relying on EMA is not adaptive enough for a wide range of demand of dependability verification/analysis. So AADL needs to be expanded and studied for more abundant modeling and analysis methods. In the aspect of schedulability analysis, AADL supports some classical scheduling scenarios, but the influencing factors of schedulability also includes resource sharing, conversion, and some other scenarios in multiprocessor environment. Adding this content to Behaviors Annex also helps to improve the existing analysis methods.

Test activities based on AADL. Integrating software testing technique with model-driven development process and generating test cases based on AADL model still remain to be study. Moreover, test cases need corresponding test processes, methods and tools, which should be also studied.

Software verification and analysis based on AADL in deployment phase. With the application of distributed embedded real-time system, the system deployment and configuration are getting more attention. Deployment means integrating a distributed application component with the corresponding hardware and making it ready to run, and using middleware is beneficial to system deployment. Configuration means the selection of components and parameters in deployment phase. Deployment based on architecture is beneficial to the following several aspects: (1) provides high-level architecture view describing the hardware and software model in deployment phase;(2)analyzes quality attributes of the deployment scheme based on the architecture model, and choose reasonable deployment scheme;(3)records system deployment experience through architecture, in order to reuse for the next time. AADL describes the configuration and deployment of information and literature in detail through property. [Combining model processing and middleware configuration for building distributed high - integrity systems] uses Ocarina tool to automatically generate configurable middleware according to the properties of AADL definition, and deploys distribute applications on middleware. According to the configuration and deployment information in AADL architecture model, analyst can analyze and evaluate the deployment plan.

Research of architecture evolution Based on AADL model. Studies show that the internal and external environment will cause runtime changes of the system architecture. So the problem we face is how to get the dynamic information of architecture in early phase of the software life cycle, and select the most optimal solution to guide the change of the runtime system, so as to realize the evolution of the system. AADL supports mode conversion, supports component changing description such as component replacement or adding, but the existing supports will not be enough to meet demands with the expansion of the system. Therefore we need to extend AADL semantic to make it support dynamic architecture, and formalize the related semantic.

Software process. Integrating the analysis of multiple NFPs raises software process issues. For each NFP, it is necessary to explore the state space for different design alternatives, configurations, and workload parameters in order to diagnose problems and decide on improvement solutions. The challenge is how to compare and rank different solution alternatives that may improve some NFPs and deteriorate others, and how to decide on the trade-offs.

Analysis of multiple NFPs. Another advanced open issue is the fact that by integrating the analysis of multiple NFPs, we need to handle (i.e., generate, analyze, and diagnose) different formal modeling languages/analysis models. How do we iterate between different languages/models? Are the languages/ models obtained from exactly the same AADL model, or from different views and/or abstractions of the AADL model? If this is the case, how do we keep these views consistent? When we change one to improve one NFP, how do the other change?

Design space exploration. Another issue is using optimization techniques over multiple NFPs to obtain the best system design and configuration. How do we explore the state space? Some kind of parameterizations would be necessary, which would allow an automatic experiment manager to modify all the formal languages/models included in the optimization procedure.

Incremental propagation of changes through the model chain. Currently, every time the software design is changed to improve some properties, a new analysis model is derived from scratch in order to redo the analysis. The challenge is to develop incremental transformation methods for propagating a certain change from the software model to the analysis model, thus keeping different model consistent, instead of starting from scratch after every model improvement.

Multi-paradigm modeling. A research direction that gets attention these days in the field of model-driven development, called multi-paradigm modeling, is looking at how to use multiple modeling languages and formalisms to solve a larger problem. The questions are as follows: how to interface and bridge different modeling languages that are used in a large problem, how to query not only one model but different related models, how to trace-link related model element in different models (e.g., a software and a corresponding analysis model), how to carry results from one model to another, how to control complex experiments, how to integrate the tool support, etc. The idea is to accept that we have to handle different related models expressed in different modeling languages that cannot be joined together in a single super modeling language. Instead, we need to learn how to navigate between different models and to do overall reasoning. In other words, we could approach the idea of analyzing multiple NFPs as a multi-paradigm modeling problem.

Tool inter-operability. There are many kinds of tools that need to work together: editors for the software model, model transformations, solver for the analysis models, optimization, experiment controllers, etc. Some of the extended abilities we propose (e.g., incremental propagation of change, translation of queries from a domain to another, synchronization of different software model versions) require new tools. Software engineering will need more engineering methods and better tool support to take advantage of the verification of NFPs as proposed in the book. Experience shows that it is difficult to interface and integrate seamlessly different tools, which may have been created at different times with different purposes and maybe running on different platforms.

References

[1] Feiler P H, Lewis B A, Vestal S. The SAE Architecture Analysis & Design Language (AADL) a standard for engineering performance critical systems [J]. Computer Aided Control System Design, IEEE International Conference on Control Applications, IEEE International Symposium on Intelligent Control, IEEE. 2006: 1206-1211.

[2] Vliet H V. Software Engineering: Principles and Practice [Z]. Wiley Publishing, 2008.

[3] Aerospace S, As S. Architecture analysis and design language (AADL) [J]. AS-5506, SAE International. 2004.

[4] Walker M, Reiser M, Tucci-Piergiovanni S, et al. Automatic optimisation of system architectures using EAST-ADL[J]. Journal of Systems and Software. 2013, 86(10): 2467-2487.

[5] Chkouri M Y, Robert A, Bozga M, et al. Translating AADL into BIP - Application to the verification of real-time systems[Z]. 20085-19.

[6] Rolland J O, Bodeveix J, Filali M, et al. Modes in Asynchronous Systems[Z]. 2008282-287.

[7] Abdoul T, Champeau J, Dhaussy P, et al. AADL Execution Semantics Transformation for Formal Verification[C]. 2008.

[8] Sokolsky O, Lee I, Clarke D. Schedulability analysis of AADL models[J]. Parallel and Distributed Processing Symposium, International. 2006, 0: 164.

[9] Berthomieu B, Bodeveix J, Chaudet C, et al. Formal Verification of AADL Specifications in the Topcased Environment[M]. Reliable Software Technologies – Ada-Europe 2009, Kordon F, Kermarrec Y, Springer Berlin Heidelberg, 2009: 5570, 207-221.

[10] Olveczky P C, Boronat A, Meseguer J, et al. Formal Semantics and Analysis of Behavioral AADL Models in Real-Time Maude[J]. Lecture Notes in Computer Science. 2010: 47-62.

[11] Ma Y, Talpin J, Shukla S K, et al. Distributed Simulation of AADL Specifications in a Polychronous Model of Computation[C]. 2009.

[12] Jahier E, Halbwachs N, Raymond P, et al. Virtual execution of AADL models via a translation into synchronous programs[Z]. 2007134-143.

[13] Yang Z, Bodeveix J, Hu K, et al. From AADL to Timed Abstract State Machines: A Verified Model Transformation[J]. Journal of Systems and Software. 2014, 93(2): 42-68.

[14] Xf S B, Rnander, Seceleanu C, et al. ABV - A Verifier for the Architecture Analysis and Design Language (AADL)[C]. 2011.

[15] Bomel P, Blouin D, Lanoe M, et al. Functional validation of AADL models via model transformation to SystemC with ATL[J]. ACM/IEEE 15th International Conference on Model Driven Engineering Languages &amp; Systems. 2012.

[16] Bodeveix J, Filali M, Garnacho M, et al. On the Mechanization of an AADL Subset[J]. Science of Computer Programming: special issue on Architecture Design Language (submitted, 2013). 2013.

[17] Yang Z, Hu K, Ma D, et al. Towards a formal semantics for the AADL behavior annex[C]. 2009.

[18] Yang Z, Hu K, Ma D, et al. Formal Semantics And Verification Of Aadl Modes In Timed Abstract State Machine[J]. Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on. 2010: 1098-1103.

[19] Belala F, Benammar M, Barkaoui K, et al. Formal Modeling and Analysis of AADL Threads in Real Time Maude[J]. Journal of Software Engineering and Applications. 2013, 5(12): 187.

[20] Cherfia T A, Faiza B, Benlahrache N. Modeling of Architectural Reconfiguration Case Study: Automated Teller Machine[C]. 2012.

[21] Jahier E, Halbwachs N, Raymond P. Synchronous modeling and validation of schedulers dealing with shared resources[Z]. 2008.

[22] Jahier E, Halbwachs N, Raymond P. Synchronous modeling and validation of priority inheritance schedulers[J]. Springer Berlin Heidelberg. 2009: 140-154.

[23] de Niz D. Architectural Concurrency Equivalence with Chaotic Models[J]. Model-based Methodologies for Pervasive and Embedded Software, 2008. MOMPES 2008. 5th International Workshop on. 2008: 57-67.

[24] Monteverde D, Olivero A, Yovine S, et al. VTS-based Specification and Verification of Behavioral Properties of AADL Models[M]. Francia: Toulouse, 2008.

[25] Xiaomei Z, Chenjun H, Gang L, et al. MDE-based compatible combination method of AADL components[J]. Computer Engineering and Design. 2014, 35: 1862-1867.

[26] Bo L, Shuyu L. AADL Behavior Based on NuSMV Model Validation of Inquiry[J]. Computer Technology and Development. 2012: 110-113.

[27] Chkouri M Y, Bozga M. Prototyping of Distributed Embedded Systems Using AADL ⋆[Z]. 2009.

[28] Sokolsky O, Lee I, Clarke D. Process-Algebraic Interpretation of AADL Models[J]. Reliable Software Technologies – Ada. 2009.

[29] Zhou J, Johnsen A, Lundqvist K. Formal Execution Semantics for Asynchronous Constructs of AADL[Z]. New York, NY, USA: 201243-48.

[30] Hugues J, Zalila B, Pautet L, et al. From the prototype to the final embedded system using the Ocarina AADL tool suite[J]. ACM Transactions in Embedded Computing Systems. 2008, 7(4): 161-170.

[31] Qian L, Shenglin G, Yun L, et al. Schedulability verification of AADL model based on UPPAAL[J]. Journal of Computer Applications. 2009, 29: 1820-1824.

[32] Zhensong L, Bin G. Research on Verification Method of AADL Behavior Model Based on UPPAAL[J]. Computer Science. 2012, 39: 159-161.

[33] Ma Y, Yu H, Gautier T, et al. System synthesis from AADL using Polychrony[J]. 2011 Electronic System Level Synthesis Conference (ESLsyn). 2011: 1-6.

[34] Yu H, Ma Y, Glouche Y, et al. System-level co-simulation of integrated avionics using polychrony[J]. Proceedings of the 26th ACM Symposium On Applied Computing. 2011.

[35] Yu H, Ma Y, Gautier T, et al. Polychronous modeling, analysis, verification and simulation for timed software architectures[J]. Journal of Systems Architecture. 2013, 59(10): 1157-1170.

[36] Yu H, Ma Y, Gautier T, et al. Exploring system architectures in AADL via Polychrony and SynDEx[J]. Frontiers of Computer Science. 2013, 7(5): 627-649.

[37] Ma Y, Yu H, Gautier T, et al. Toward Polychronous Analysis and Validation for Timed Software Architectures in AADL[Z]. San Jose, CA, USA: 20131173-1178.

[38] Renault X, Kordon F, Amp J, et al. Adapting Models to Model Checkers, A Case Study : Analysing AADL Using Time or Colored Petri Nets[C]. 2009.

[39] Renault X, Kordon F, Hugues J. From AADL Architectural Models to Petri Nets: Checking Model Viability[Z]. 2009313-320.

[40] Oleg Sokolsky A C. Analysis of AADL Models Using Real-Time Calculus with Applications to Wireless Architectures[Z]. 2008.

[41] Varona-Gomez R, Villar E, Rodr I Guez A I, et al. Architectural Optimization & Design of Embedded Systems based on AADL Performance Analysis[J]. American Journal of Computer Architecture. 2012, 1(2): 21-36.

[42] Feiler P, Rugina A. Dependability Modeling with the Architecture Analysis & Design Language (AADL)[J]. Dependability Modeling with the Architecture Analysis & Design Language (AADL). 2007.

[43] Rugina A E, Kanoun K, Niche M K A. An architecture-based dependability modeling framework using AADL[J]. CoRR. 2007, abs/0704.0865.

[44] Rugina A, Kanoun K, Ka âniche M. A System Dependability Modeling Framework Using AADL and GSPNs[M]. Architecting Dependable Systems IV, de Lemos R, Gacek C, Romanovsky A, Springer Berlin Heidelberg, 2007: 4615, 14-38.

[45] Rugina A, Kanoun K, Ka A Niche M. The ADAPT tool: From AADL architectural models to stochastic petri nets through model transformation[C]. 2008.

[46] Yunwei D, Guangren W, Fan Z, et al. Reliability Analysis and Assessment Tool for AADL Model[J]. Journal of Software. 2011, 22: 1252-1266.

[47] Jinliang G, Gang Z, Xiaochuan J, et al. Software System Reliability Modeling and Evaluation Using AADL[J]. Journal of Frontiers of Computer Science and Technology. 2011: 942-952.

[48] Zhiyi Y, Chenyu Z, Yunwei D. Modeling and Assessment of Reliability of a Software Fault Tolerant System Using AADL[J]. Computer Measurement & Control. 2009, 17: 779-782.

[49] Boudali H, Crouzen P, Haverkort B R, et al. Arcade - A Formal, Extensible, Model-Based Dependability Evaluation Framework[C]. 2008.

[50] Lutz R, Hauptman M, Sun H. Integrating Product-Line Fault Tree Analysis into AADL Models[J]. Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05). 2007: 15-22.

[51] Bozzano M, Cimatti A, Katoen J, et al. Safety, Dependability and Performance Analysis of Extended AADL Models[J]. COMPUTER JOURNAL. 2011, 54(5): 754-775.

[52] Bozzano M, Cimatti A, Katoen J, et al. The COMPASS Approach: Correctness, Modelling and Performability of Aerospace Systems[M]. 2009: 5775, 173-186.

[53] Johnsen A, Lundqvist K, Pettersson P, et al. Automated Verification of AADL-Specifications Using UPPAAL[J]. Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05). 2012: 130-138.

[54] Bae K, O Lveczky P C, Al-Nayeem A, et al. Synchronous AADL and Its Formal Analysis in Real-time Maude[Z]. Berlin, Heidelberg: 2011651-667.

[55] Wei L. Research on AADL Model Transformation and Model Verification[D]. Shaanxi Normal University, 2013.

[56] Avizienis A, Laprie J C, Randell B, et al. Basic concepts and taxonomy of dependable and secure computing[J]. Dependable and Secure Computing, IEEE Transactions on. 2004, 1(1): 11-33.

[57] Eby M, Werner J, Karsai G, et al. Integrating security modeling into embedded system design[Z]. IEEE, 2007221-228.

[58] Balsamo S, di Marco A, Inverardi P, et al. Model-based performance prediction in software development: a survey[J]. Software Engineering, IEEE Transactions on. 2004, 30(5): 295-310.