Accepted in 24th IEEE/ACM International Conference on Automated Software Engineering, 2009 (ASE '09), 2009

A Tool Suite for the Generation and Validation of Configurations for Software Availability

A. Gherbi[1], A. Kanso[1], F. Khendek[1], M. Toeroe[2] and A. Hamou-Lhadj[1]
[1]*Concordia University, Montréal, Canada*
{gherbi, al_kan, khendek, abdelw}@ece.concordia.ca
[2]*Ericsson Inc., Montréal, Canada*
maria.toeroe@ericsson.com

Abstract

The Availability Management Framework (AMF) is a service responsible for managing the availability of services provided by applications that run under its control. Standardized by the Service Availability Forum (SAF), AMF requires for its operations a complete and compliant AMF configuration of the applications to be managed. In this paper, we describe two complementary and integrated tools for AMF configurations generation and validation. Indeed, writing manually an AMF configuration is a tedious and error prone task as a large number of requirements defined in the standard have to be taken into consideration during the process. One solution for ensuring compliance with the standard is the validation of the configurations against all the AMF requirements. For this, we have designed and implemented a domain model for AMF configurations and use it as a basis for an AMF configuration validator. To further ease the task of a configuration designer, we have devised and implemented a method for generating automatically AMF configurations.

Keywords: High-Availability, Availability Management Framework, Automated Configuration Generation, Validation, Domain Model.

1. Introduction

The Service Availability Forum (SAF) [1] is a consortium of several telecommunications and computing companies that work towards standardized solutions for enabling the development of highly available software systems. One of the main outcomes of the SAF standardization effort is the Availability Management Framework (AMF) [2], which is a service responsible for managing redundant resources to ensure high availability of services provided by software applications.

The AMF service, implemented as part of a SAF middleware, requires a configuration for any application that operates under its control. An AMF configuration describes the organization of the resources and applications services. More precisely, it describes a set of entities to be managed by AMF in a running system, their types and relationships and their deployment on the cluster nodes. The basic entity of an AMF configuration is the component, which represents a software and/or hardware resource that provides the service that needs to be made available. The workload assigned to a component is referred to as a component service instance. Components are logically grouped into service units in order to combine their functionality and provide higher level services referred to as service instances. To protect these services, service units are grouped into service groups. A service group protects a set of service instances (i.e., the workloads) assigned to its service units according to a redundancy model. When a particular service instance is assigned to a service unit, its composing component service instances are assigned to the components in this service unit. The grouping of service groups forms an AMF application. From a deployment perspective, each service unit is deployed on an AMF node (i.e., a node on which AMF implementation is running). The set of AMF nodes forms the AMF cluster.

In the AMF specification, the notion of type is used to capture common characteristics of entities that belong to the same type. The types define also relations among entities. For example, a service unit type specifies the set of component types, which defines the types of the components that must compose a service unit of this service unit type.

Creating manually an AMF configuration can be a tedious and error-prone task [8, 9]. The problem is that there are many entities that a configuration designer needs to handle. The grouping of these entities is constrained by the information described in their types, which requires several consistency checks to be performed at various levels of the configuration design process. There are also many dependencies that need to be taken into account. For example, it is important to understand how components depend on each other in order to build a valid configuration.

Keeping all these constraints in mind while writing a configuration is a very demanding task for configuration designers. Ensuring compliance with the standard specification for configurations developed manually is also complex if not impossible, especially when creating an AMF configuration describing several applications with a large number of components to be deployed on a large cluster of nodes. In order to alleviate this task we propose to use the domain model of the UML profile for AMF [6], we are developing, for the validation of configurations. The AMF model and requirements have been captured in this domain model using a class diagram and the Object Constraint Language [7]. We introduce and discuss in this paper the configuration validator.

To further alleviate the task of configuration designers, we have developed a second tool for generating automatically AMF configurations based on the work presented in [8, 9]. This tool takes as input the software characteristics as provided by the vendor in a so-called Entity Types File(s) (ETF) [5], the services to be provided and the deployment cluster. When the provided software can be configured in the given deployment cluster to provide and protect the services as requested, a configuration is generated automatically.

2. Configuration Validation

An AMF configuration, created manually or generated automatically, is saved in the IMM (Information Model Management) XML format [4], and made available to SAF services through the IMM service [3]. In order to help validate third-party or manually created configurations, we have captured most of the concepts defined in the AMF specification and their relationships in the domain model of the UML profile for AMF that is currently under development. The class diagram of this domain model describes the different entity types, entities and their relationships. In contrast to the AMF UML model provided in the standard specification, our model organizes the entity types and the entities, for instance the component types and components, differently and uses extensively object oriented paradigms such as multiple inheritance. Other AMF constraints and requirements are formalized in OCL. The corresponding Ecore model has been generated using the Eclipse Rose importer and implemented using the Eclipse Modeling Framework (EMF) code generation feature [10].

The validation process, as shown in Figure 1, includes a mapping of an instance of the AMF standard model to an instance of the AMF domain model of the profile and a validation of the later against the OCL constraints. The AMF standard model instance is created from the input provided by the user as an IMM XML file, which is the standard carrier for AMF configurations. The checking of the OCL constraints is not completely separated from the mapping but crosscuts its different steps.

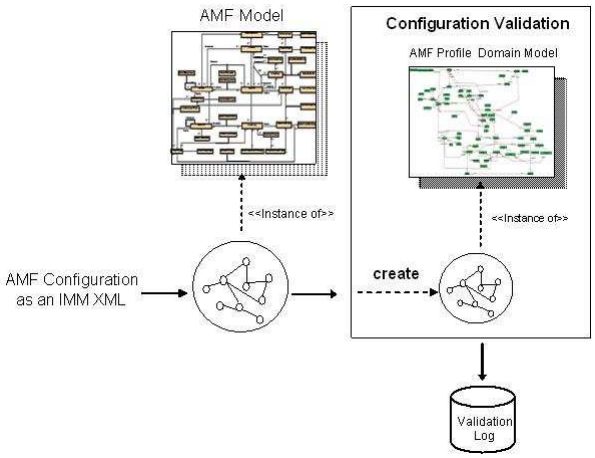

Figure 1. Configuration validation tool

One of the main tasks in this validation is checking the created AMF domain model instance of the profile against the OCL constraints. For this purpose, we have experimented with some UML modeling tools. As mentioned earlier we have used Rational Rose to build the class diagram of the domain model and the Eclipse EMF Rose importer to build the Ecore model [10]. We have used Rational Software Architect (RSA) [11] to specify the OCL constraints and checked them against the domain model for static consistency. However, these tools do not support the validation of instances of the domain model (object diagrams) against the OCL constraints, which actually capture constraints on the domain model class diagram and other constraints from the standard defining a valid AMF configuration. We have considered the Dresden OCL toolkit [12] for this instance validation; however several of our constraints are not supported and cannot be checked with the current release of the tool. We have therefore implemented our OCL constraints in Java.

3. Configuration Generation

To further ease the task of a configuration designer we have devised a method for the automatic generation of AMF configurations [8, 9]. This work has been done in parallel with the design of the profile and is therefore based on the AMF model as provided in the standard instead of the AMF profile. The method and the corresponding tool for configuration generation take as input the description of the software as provided by the vendor in the ETF(s) and the configuration designer requirements in terms of services to be provided,

protected with a certain redundancy model and the description of the deployment cluster and nodes. In the ETF file(s) the vendor describes the software in terms of AMF meta-types reflecting the complete range of AMF features supported by the software. The main steps in the generation method are captured in Figure 2, where the Type Finder selects the appropriate ETF types for creating the AMF types to provide the required services. The AMF entities are then generated from these AMF types. The ETF type selection is based on the services to be provided. For instance the component types are selected from the ETF in order to match the component service instances provided by the configuration designer as a requirement. Other criteria have to be checked, for instance whether the component capability model satisfies the requested redundancy models. Moreover, a service unit type is defined in terms of possible component types and the maximum number of components of each type allowed in a service unit. Therefore when selecting component types, we need to keep in mind the maximum number of components of this type in the service unit and the necessary capacity in terms of component service instances for a given component service type. Several calculations are necessary as described in [8, 9]. An example of those calculations is to carefully configure the order in which the components need to be instantiated by AMF based on the component dependency relationships. A misconfiguration of this aspect could lead either to the failure of the component instantiation or malfuncting of the component causing AMF to either resort to another service unit to provide the services or to a recovery procedure.

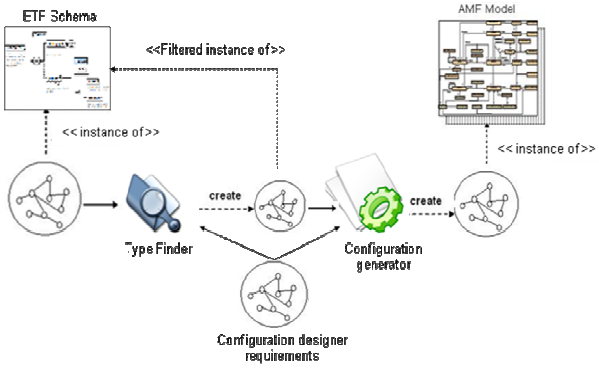

Figure 2. Configuration generation tool

The configuration generation tool is built as an Eclipse plug-in. It reads ETF(s) and takes constraints and requirements from the designer. It builds an instance of the AMF model as defined in the standard [2], which is then saved in the IMM XML format.

4. Conclusion

In this paper, we presented a tool suite that allows for the automatic generation and validation of AMF configurations. The next step will be the completion of the profile by developing a concrete syntax for the AMF profile. The configuration generation approach will be fully integrated with the profile for the generation and analysis of multiple configurations.

5. Acknowledgment

This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson Software Research.

6. References

[1] Service Availability Forum™, URL: http://www.saforum.org

[2] Service Availability Forum, Application Interface Specification. Availability Management Framework SAI-AIS-AMF-B.03.01.

[3] Service Availability Forum, Application Interface Specification. Information Management Service SAI-AIS-IMM-A.02.01.

[4] Service Availability Forum. IMM XML Schema, SAI-AIS-IMM-XSD-A.01.01.xsd.

[5] Service Availability Forum, Application Interface Specification. Software Management Framework SAI-AIS-SMF-A.01.01.

[6] A. Gherbi, P. Salehi, M. Toeroe, F. Khendek, A. Hamou-Lhadj, "Towards a UML Profile for AMF Configurations Modeling and Analysis", *Technical Report, Electrical and Computer Engineering, Concordia University*, April 2009.

[7] OMG, Object Constraint Language (OCL), Version 2.0, OMG Available Spec.: formal/06-05-01, 2006.

[8] A. Kanso, M. Toeroe, F. Khendek, A. Hamou-Lhadj, "Automatic Generation of AMF Compliant Configurations", *In Proc. of ISAS'2008, LNCS, Vol. 5017*, pp.155-170, 2008.

[9] A. Kanso, M. Toeroe, A. Hamou-Lhadj, F. Khendek, "Generating AMF Configurations from Software Vendor Constraints and User Requirements", *In Proc. of ARES'2009,* Fukuoka, Japan, 2009.

[10] D. Steinberg, F. Budinsky, M. Paternostro, E. Merks. *EMF: Eclipse Modeling Framework, Second Edition.* Addison-Wesley Professional, 2009.

[11] IBM, Rational Software Architecture, http://www-01.ibm.com/software/awdtools/swarchitect/

[12] Dresden OCL Toolkit, http://dresden-ocl.sourceforge.net/

Appendix A: The Tool Suite Screen Snapshots

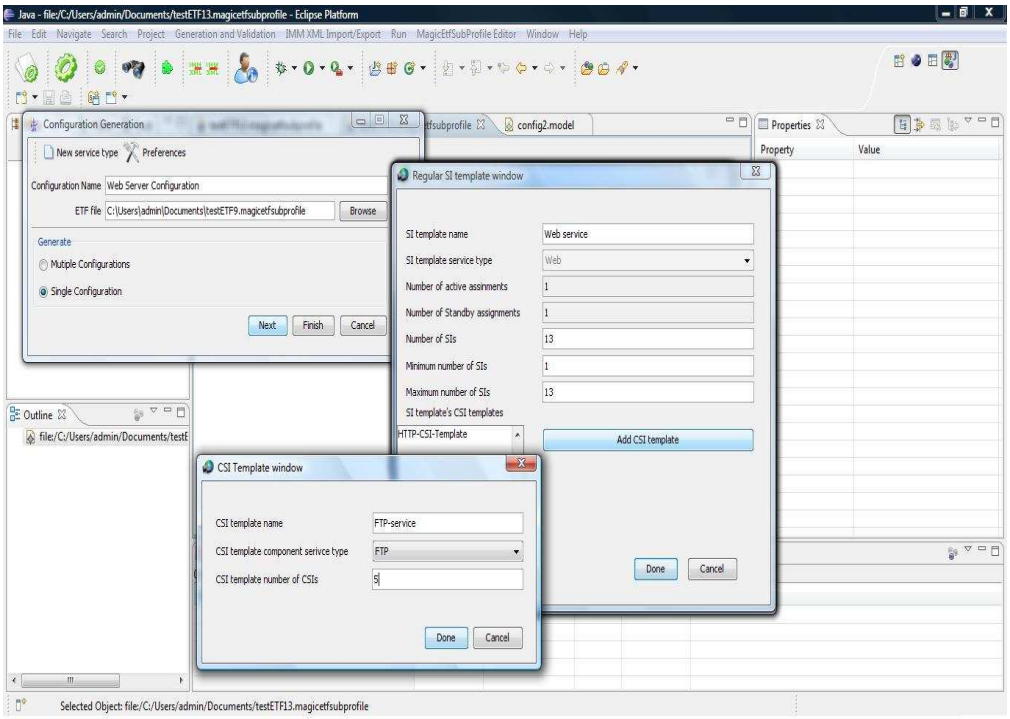

Figure 3. Snapshot of the GUI through which the configuration designer inputs the services to be provided

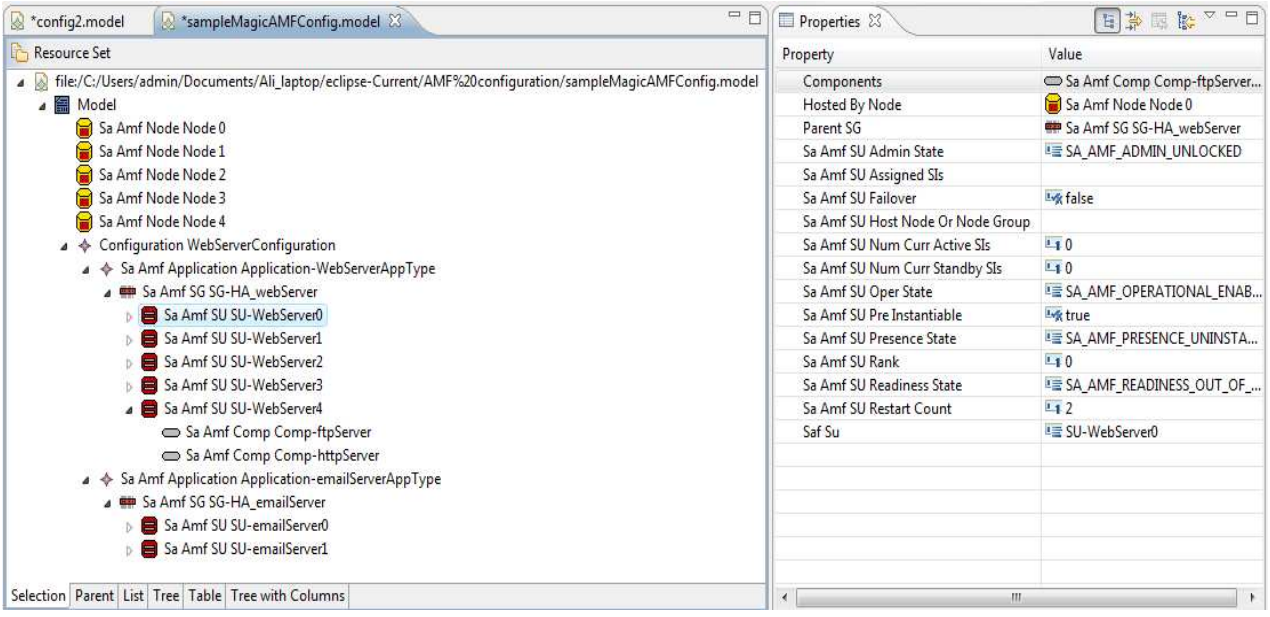

Figure 4. An Example of an AMF configuration, the left hand side of the figure shows the configuration entities, the right hand side shows their attributes

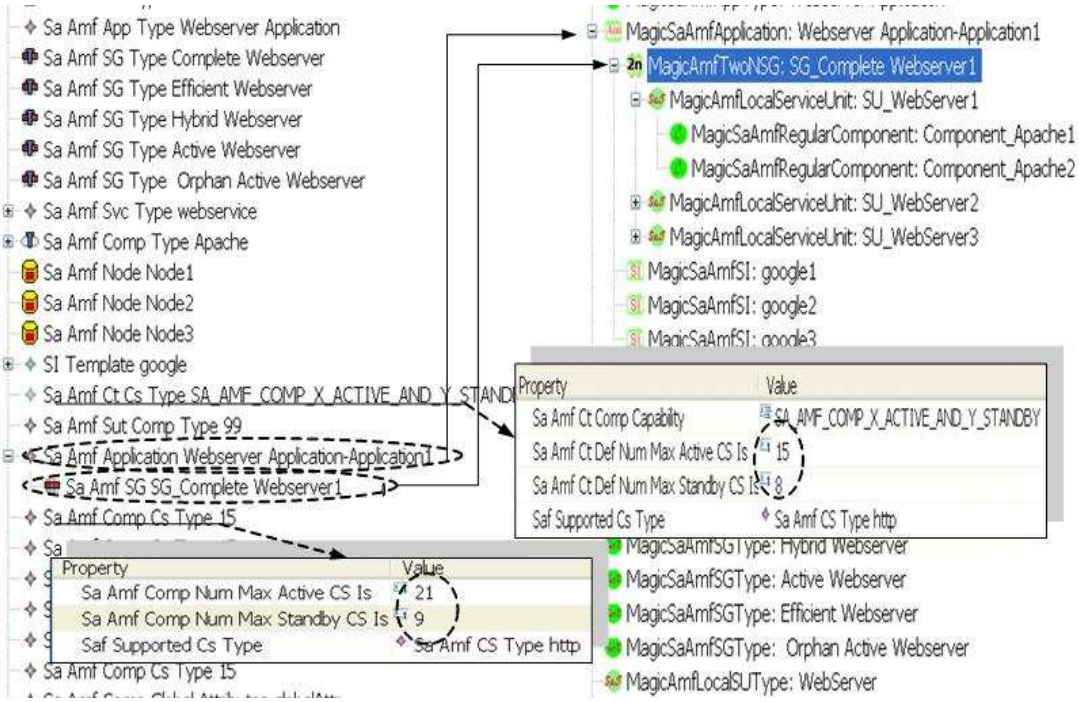

Figure 5. Mapping from the AMF model to the MAGIC model. This snapshot shows how a part of the validation process is performed through the mapping of AMF standard model to the newly created model.

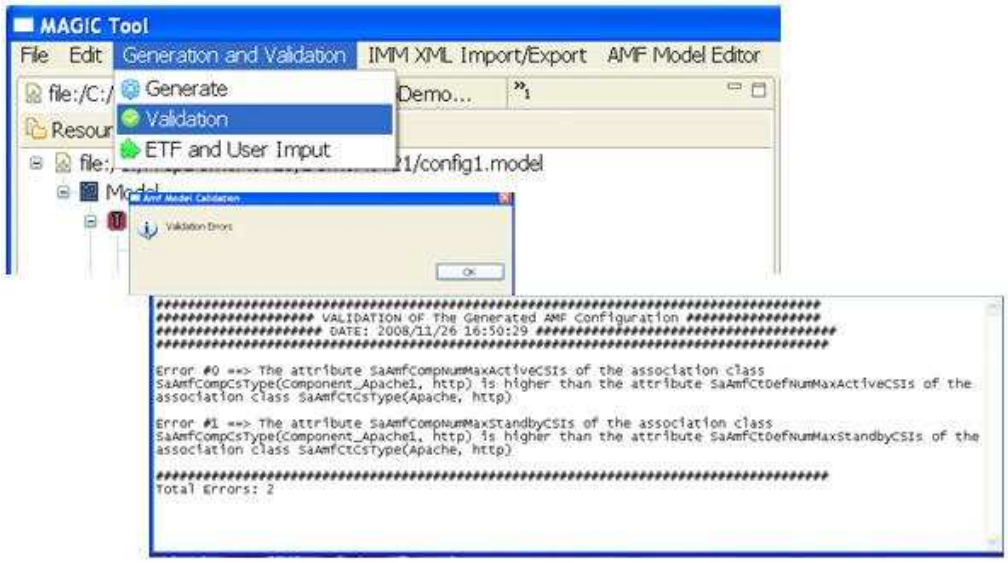

Figure 6. An Example of OCL constraint violation during the validation of an AMF configuration