2014 Tenth European Dependable Computing Conference

Towards a Resilience Benchmarking Description Language for the Context of Satellite Simulators

Denise Rotondi Azevedo, Ana Maria Ambrosio
Space Engineering and Technology
National Institute for Space Research, INPE
São José dos Campos, Brazil
denise.rotondi@inpe.br, ana.ambrosio@inpe.br

Marco Vieira
CISUC, Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
mvieira@dei.uc.pt

***Abstract*—Specifying a resilience benchmark is a difficult task due to the complexity of the benchmark components and the need for standardization. Existing approaches for benchmark specification, including document-based and program-based approaches, are limited in terms of their scope and in the support they provide to the benchmark users. In this short paper we present the work we are conducting towards the definition of a description language for resilience benchmarks for the domain of satellite simulators.**

Keywords- resilience benchmarking; description language; XML

I. INTRODUCTION

Simulation tools have been used in different areas of knowledge and in many industrial sectors, including manufacture, automobile, etc. Simulators are also widely used in the context of space systems and are applied in different phases of a space mission, supporting system analysis, system verification and validation, operators training, among others.

In the last decade different standards for simulators development have been defined to promote portability, reusability, and interoperability. For example, the Institute of Electrical and Electronics Engineers (IEEE) specified the High Level Architecture (HLA), which defines a software architecture for creating simulators from computational simulation components [1]. The use of a standard infrastructure to build simulators enables the integration and reuse of models developed by different teams and even different organizations. At first sight, this possibility of reuse has a positive impact on the dependability attributes since it promotes quality and stability. However, this evolutionary and modular characteristic may also have a negative impact on those same attributes when considering the constant integration of new and different models.

Evaluating a simulation infrastructure before using it is of utmost importance to verify if it is able to satisfy the relevant dependability attributes, e.g. by isolating faults and preventing error propagation, ensuring that new models do not affect the simulation behavior or results, etc. This creates the need for benchmarking approaches that allow evaluating, measuring and comparing, in a systematic and standardized way, the dependability and resilience attributes of satellite simulators built using infrastructure standards, such as the HLA.

Benchmarks can be provided in the form of standardized tools ready to be run or in the form of documents that define how they should be implemented and executed. In general, a benchmark provided in the form of a standardized tool is more limited (in terms of scope) than a benchmark provided as a document, since a computer program has to be written assuming a given structure regarding the system under benchmark, while a document just needs to define the services provided by the system (in an abstract manner) and the benchmark implementation rules [2]. A key issue is that a benchmark provided through a document still lacks of standardization and formalization what may hinder its interpretation and its direct use as input to a software tool. This short paper puts forward a Resilience Benchmarking Description Language (RBDL) for the domain of satellite simulators built using HLA standards. This language is based on eXtensive Markup Language (XML) and XML Schema and can be used as a framework in the specification of a benchmark.

There are several advantages on using a descriptive language compared to a standardized tool or a specification document. In fact, a descriptive language structure may be used as a framework for the benchmark implementation, guiding its execution and promoting standardization. In addition, the use of the RBDL in the benchmark specification provides the flexibility of a document combined with the possibility of using available tools and resources to support the generation, validation and presentation of the benchmark components. Finally, as XML documents are *processable* by computer programs, benchmarks defined using RDBL may be directly processed and used by benchmark tools in order to automatic implement and execute the benchmark experiments.

Although the work on the RDBL is being conducted in the context of a resilience benchmark for satellite simulators, it is structured in a way that allows also the extension to other types of benchmarks (e.g. performance and dependability) and to other domains (e.g. databases) by adding or removing specific elements.

The outline of the paper is as follows. Section 2 introduces background. Section 3 overviews the description language and Section 4 presents an example. Section 5 concludes the paper.

II. BACKGROUND

A. XML as a Base for the RBDL

The XML (eXtensible Markup Language) is a hierarchical and extensible language useful to describe and

978-1-4799-3804-9/14 $31.00 © 2014 IEEE
DOI 10.1109/EDCC.2014.19

represent different types of data and for information exchange. The XML extensibility allied with the possibility of using XML Schemas for defining and validating XML files, make XML a language frequently used for describing domains and as basis for other derived languages, such as the eXtensible Business Reporting Language (XBRL) and the Election Markup Language (EML), among others.

There are generally accepted best practices and design patterns for defining XML Schemas, including: *Russian Doll*, *Salami Slice, Venetian Blind, Garden of Eden* and *Chameleon*. These differ mainly in respect to the elements location (global or local) and in the way the elements and types are used. Different patterns must be adopted according to the Schemas needed and used; e.g. some patterns better encapsulate the elements, while others emphasize reuse and extensibility [3].

In addition to design patterns suitable for reuse, an XML Schema has features that facilitate the extension of documents. Among these features there are the *wildcards*, which are well-defined extension points within XML schemas, and the *substitution groups* that can be used in association with abstract types and elements. A *substitution group* contains elements that can appear interchangeably in an XML instance document in a way that resembles subtype polymorphism in Object Oriented Programming (OOP) languages [4][5].

For the definition of the Resilience Benchmark Description Language, we have used the *Chameleon* design pattern to define the generic types and the *Garden of Eden* design pattern to define the elements within the Schemas, trying to make the definition more generic and to promote the reuse of elements and types. The parts of the schema that are dependent on the problem domain have been defined in domain-specific Schemas with specific namespaces associated, using *substitution groups.*

B. Resilience Benchmarking

A computer system benchmark is a standardized tool for assessing and comparing different systems within a given application domain according to specific characteristics, e.g. performance, dependability, resilience, etc. [6]. Performance benchmarks are based on two components: the workload, which is a representation of the load that the system should execute, and a set of performance measures that characterize the system performance during the execution of the workload.

Dependability benchmarking takes forward the concept of performance benchmarking to characterize computer systems in the presence of faults, aiming at evaluating and comparing their behaviour in terms of dependability attributes. A dependability benchmark includes two more elements: the *faultload*, which is a representation of the possible faults to which the system may potentially be exposed, and measures that portray dependability attributes during the execution of the workload and the injection of the *faultload* [6][7].

Resilience benchmarking takes into consideration evolutionary characteristics providing generic forms for characterizing and comparing computer systems when subjected to changes [8]. A resilience benchmark brings two new elements: *changeloads*, representing what changes are expected in the system in terms of workload faults, configuration, etc., and resilience metrics that can be used to characterize different aspects of the system resilience in the presence of changes.

The *changeload,* which incorporates and extends the concept of dependability benchmark *faultload*, is the most complex aspect of a resilience benchmark, defining a set of changes, each of them characterized by a type and having their behavior expressed by specific parameters [9]. For example, one can indicate that a change would be "*a new version of a dll is used*" and the change parameter may be the dll version. The procedure is then based on a set of baseline scenarios that are mutated by applying specific changes. In an adaptation of the solution proposed in [9][10], we define a resilience benchmark as shown in Fig. 1.

```
Resilience Benchmark = {base scenario, changeload, goals}
Base Scenario = {workload, operational conditions}
Changeload = {changes}
    Change = {change type, source, parameters type, change rules}
    Change Rules = {application, [recover]}
Goals = {attributes, metrics}
```

Figure 1. Benchmark Definition

The execution of a resilience benchmark can be expressed by the instantiation of change scenarios where a certain base scenario is subjected to a series of change types that have a trigger factor, duration, and values for their parameters. During the benchmark execution all elements defined are instantiated for a particular system under benchmark architecture. Fig. 2 shows the elements of a resilience benchmark instantiation.

```
Change Scenario = {base scenario, {changes instances}}
Change Instance = {type, parameters instances, trigger, duration, [recover duration]}
Parameter Instance = {value | function}
```

Figure 2. Benchmark Instantiation

A standardized description of a benchmark is of key importance from the formalization, communication and verification points of view. However, a benchmark is always strongly associated with the application domain to which it applies, and usually those domains are complex and have a number of specificities that hinder its representation through a formal language. Thus, in this paper, we propose a semi-formal language as an alternative for specifying benchmarks. This standardized descriptive language provides a framework to guide the definition of the benchmark elements, the generation rules and the execution architecture, etc., allowing the same description document to be used for the benchmark definition and implementation, facilitating standardization and communication.

Furthermore, in order to have credibility, a benchmark should be relevant, representative, portable, and should also be repeatable, providing similar results when reapplied in the same environment; non-intrusive, not changing the evaluated system and simple to use. In terms of benchmark properties, the use of a description language through XML files benefits the benchmark simplicity, since these *processable* files may be directly used in the benchmark implementation and execution, and also improves the repeatability property by the use of a unique interchangeable file in the setup of experiments environment and rules.

III. RESILIENCE BENCHMARKING DESCRIPTION LANGUAGE FOR SATELLITE SIMULATORS

In the proposed description language we detail the benchmark elements based on what is presented in figures 1 and 2. As the workload tends to be inextricably linked to the problem domain, we have considered the workload for the HLA simulation infrastructure domain, comprising three different requirements dimensions: (i) *design requirements* (number of models, volume of data exchanged between models, typical frequency of models execution, repeatability constraints etc.); (ii) c*onfiguration requirements* (update rate of data exchanged, transport method used by the communication channel, etc.); (iii) t*echnological requirements* (programming languages used to develop the model, version of the HLA library, etc).

A resilience benchmark may be divided into definition and instantiation/execution. The benchmark definition is the fixed part that specifies the main elements of the benchmark that do not change among several executions of the benchmark experiments. The *BENchmarkDefinition* tag (marked with (a) in Fig. 3) describes the benchmark and its major elements: the benchmark description ((b) in Fig. 3), the goals expressed by the attributes to be measured and their metrics ((c) in Fig. 3), the workload to which the benchmark target should be subjected ((d) in Fig. 3), the *changeload* ((e) in Fig. 3) and the elements of the system under benchmark ((f) in Fig. 3)).

The *BENchmarkInstantiation* tag ((a) in Fig. 4) describes the benchmark instantiation/execution. Using this XML document, different executions with different focuses may be defined. The specified language provides a description of one or many base scenarios. The *BaseScenario* ((b) in Fig. 4) may change from one execution to another, i.e. one may vary the workloads used and/or the operating conditions which are directly related to the definition of the system under benchmarking and execution architecture. The benchmark instantiation also defines one or more scenarios instantiation, which represent the effective execution of a benchmark experiment. Through the *InstantiationScenario* tag ((c) in Fig. 4) one can define: the attribute being measured in a specific experiment, the base scenario to be used, a description of the execution parameters and the group of changes to be applied during execution.

Many elements of the benchmark can be specified in a generic way within the description language, for example, attributes may be described with the same properties for different application domains or benchmark types, as well as changes in a *changeload* since they are generically characterized by their parameters and source. However, as mentioned before, the requirements for the workload generation in a benchmark are entirely dependent on the domain. For example, while an Operational Simulator using a HLA infrastructure has federates (simulation models) as workload, a dependability benchmark for OLTP Systems [2] uses a workload based on transactions issued by different terminals (as specified in TPC-C [10]).

In order to generalize the language to meet different domains, we analyzed the following possibilities for Schema extension: (i) *XML schemas inclusion*, where generic XML elements are referenced in specific points of the language; (ii) *wildcards*, where wildcards (*any elements*) are used in extension points; (iii) *substitution groups* that use abstract elements and types at the points identified as extension points and each domain-specific XML Schema indicates the elements to be used to replace the abstract ones.

We have adopted the third approach, as it has the advantage of using intrinsic XML Schema features and concepts similar to polymorphism in OOP. However, for each new domain, the domain-specific XML Schema must be imported, i.e. a change in the generic Schemas is needed. Other advantages of this approach are the definition of abstract types and elements, and the possibility of using both *substitution groups* feature and

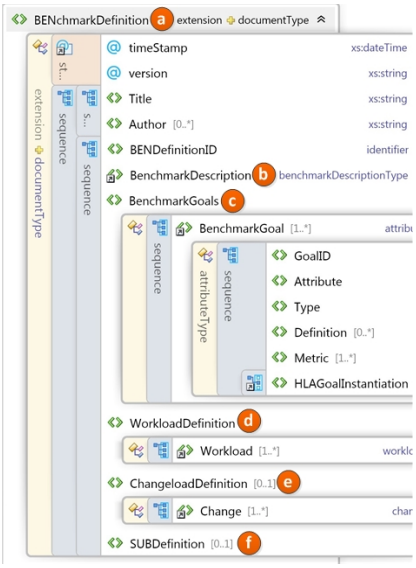

Figure 3. Benchmark Definition

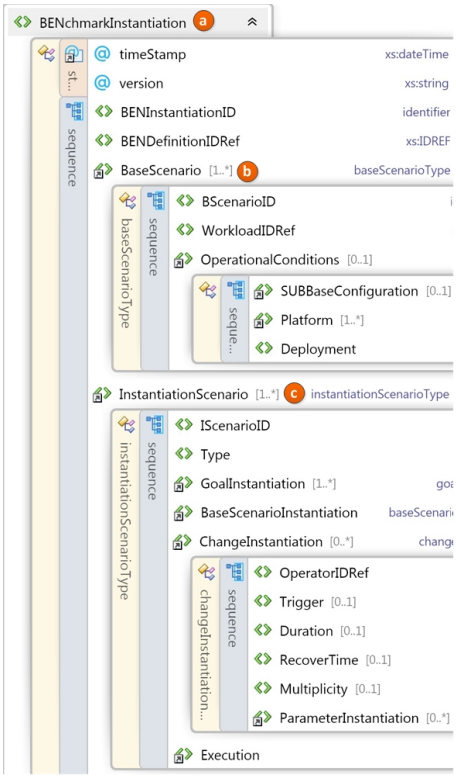

Figure 4. Benchmark Definition

different concrete types at XML creation time.

As an example of extensibility and adaptability, we have defined XML Schemas for both our domain (HLA based simulators) and the TPC-C specification ((a) in Fig. 5). The language defined for TPC-C workload is a simple example, aiming only at validating the chosen approach. Using the extensions proposed, we have defined the language for two different domains just changing the name of the imported domain-specific Schema.

It is worth noting that the proposed solution specifies a language that provides extension points to be adapted for different domains. However, the definition of domain-specific Schemas may be a complex work and represents a considerable part of the description language definition.

IV. Resilience Benchmarking Description Language Example

As a case study, we are currently specifying a resilience benchmark for the context of satellite simulator using the proposed language. A simple workload and just one change in the *changeload* are considered to keep the example simple and in a reasonable size.

The example presents an excerpt of the Benchmark Resilience definition file for an Operational Satellite Simulator. The part of the XML file presented in Fig. 6 defines two elements: the workload and the *changeload*. The Workload tag ((a) in Fig. 6) defines the number of simulation models, the simulation time step, the volume of data exchanged between models, repeatability characteristics of the simulation, among others. The Change chosen (FAttrib, (b) in Fig. 6) represents a modification on the volume of data exchanged among the models. This tag presents the source that will be changed, the change class and the parameters that characterize the change.

V. Final Consideration

The proposed Resilience Benchmark Description Language aims at being an alternative to the specification of resilience benchmarks serving as a framework and helping in standardization and communication. The language can also be used for dependability and performance benchmarks and provides extension points to be adapted for different domains.

RBDL can either be used directly for the benchmark specification or existing specifications can be translated into this language. The advantage of using a derivative language specified using markup languages and XML Schemas is that we automatically inherit many tools and APIs for XML files validation, generation and presentation. Furthermore, the *processability* of XML files enables them to be directly used by the benchmark tools to generate workloads, *changeloads* and to implement and conduct the benchmark experiments.

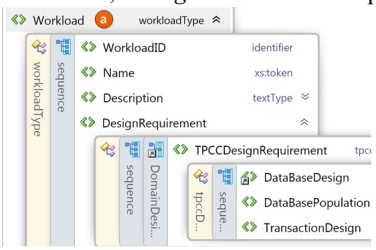

Figure 5. TPC-C Workload

Figure 6. RBDL-XML file for Operational Satellite Simulator

As future work we intend to extend the language for defining other benchmark elements, such as the benchmark validation and the format of disclosure reports. In addition, we intend to employ the specified language as part of a broader work on the definition of a methodology for specifying resilience benchmarks for Satellite Simulators using HLA infrastructures.

References

[1] Möller, M. Karlsson, and B. Löfstrand, "Developing fault tolerant federations using HLA evolved," In: Spring Simulation Interoperability Workshop, 2005.

[2] M. Vieira, Dependability Benchmarking for Transactional Systems, Ph.D thesis, Faculty of Sciences and Technology of the University of Coimbra, July 2005.

[3] E. Hewitt, Java SOA Cookbook, O'Reilly, 2009.

[4] WC3 XML Schema Part 1: Structures Second Edition, October, 2004. available at: http://www.w3.org/TR/xmlschema-1/.

[5] D. Obasanjo, "Designing extensible, versionable XML formats," 2004, available at: http://msdn.microsoft.com/en-us/library/ms950793.aspx.

[6] A. Bondavalli, et al., "Research Roadmap Deliverable D3.2," AMBER – Assessing Measuring and Benchmarking Resilience, Funded by European Union, 2009.

[7] D. Costa, R. Barbosa, R. Maia, and F. Moreira, "DeBERT: Dependability benchmarking of embedded real-time off-the-shelf components for space applications," In: K. Kanoun, L. Spainhower, Dependability Benchmarking for Computer Systems, New Jersey: John Wiley & Sons, Inc., 2008. cap. 13, p.255-283.

[8] R. Almeida and M. Vieira, "Changeloads for Resilience Benchmarking of Self-Adaptive systems: a risk-based approach,", Dependable Computing Conference (EDCC), 2012 Ninth European, 2012, 173-184.

[9] J. Cámara, R. de Lemos, M. Vieira, R. Almeida, and R. Ventura, "Architecture-based resilience evaluation for self-adaptive systems".Computing, 2013, pp. 1-34.

[10] Transaction Processing Performance Council, "TPC Benchmark C, Standard Specification, Revision 5.11", 2010, available at: http://www.tpc.org/tpcc.