Integrating an Online Configuration Checker with Existing Management Systems: Application to CIM/WBEM Environments

Ludi Akue, Emmanuel Lavinal, Thierry Desprats, Michelle Sibilla
IRIT, Université de Toulouse
118 route de Narbonne
31062 Toulouse, France
Email: {akue, lavinal, desprats, sibilla}@irit.fr

***Abstract*—Runtime configuration validation is a critical requirement if we are to build reliable self-adaptive management systems. This paper presents a generic framework that includes a runtime configuration checker built upon a high-level language dedicated to the specification of configurations and validity constraints. In addition, we describe a methodology for using this framework and integrating the configuration checker with existing management systems. In particular, we show how we use the framework to enrich a CIM/WBEM management environment with automatic runtime configuration validation against a defined set of constraints guarding structural correctness and service behavior conformance. Our experiments with management models conforming to the CIM Virtual System profile show viable results demonstrating the feasibility of our approach.**

Keywords*—Self-management, configuration validation, online configuration validation, CIM/WBEM environments.

I. Introduction

The self-management vision – in which systems are able to self-adjust in response to their environment – has gained a lot of momentum in networked systems management where it is viewed as a promising solution for the management of today's large scale and complex systems. Therefore, it becomes fundamental to have automatic online verification and validation techniques to provide grounds for confidence that the autonomous systems are operating correctly. This issue is particularly significant as today's systems are increasingly supporting critical missions (health care, avionics, clouds) where service failures can be extremely expensive and affect people's lives.

Notably, runtime configuration changes being one of the principal means to achieve self-management, we believe that enhancing existing management systems with runtime validation capabilities is a major step towards ensuring the correctness and safety of reconfiguration activities. Furthermore it would ease the adoption of ongoing self-management solutions. We are therefore working on a generic framework to enrich management systems – especially those that have not built-in validation capabilities – with online configuration checking.

In this paper, we present a validation framework that includes an online configuration checker for enabling runtime configuration validation within existing management systems, along with its usage methodology. The framework proposes an integrated approach based on MeCSV, a high-level language that allows operators to specify at design time, a platform-neutral configuration schema of their managed system with the constraints guarding desired service architecture and behavior. At runtime, an online configuration checker relying on this specification, can verify dynamically candidate configuration instances.

In addition, we describe how we use the framework to enrich a CIM/WBEM management environment with online configuration checking capabilities. Following the same process, the validation framework could be integrated with other existing management systems.

The rest of the paper is organized as follows: Section II presents related work and Section III describes the framework and its usage methodology. We describe the integration of the framework with the CIM/WBEM standards in Section IV. A prototype experiment, Section V, of this integration on *CIM Virtual System Profile* models shows viable results proving the feasibility of the approach. Finally, Section VI concludes the paper and identifies future work.

II. Related Work

The need for configuration representation standards and configuration automation are growing concerns regarding the complexity of the configuration management of today's large-scale and heterogeneous systems [1], [2], [3], [4]. Our work is at the junction of these topics as the validation system we provide enables a platform-independent runtime configuration validation which is a prerequisite for configuration automation and self-configuration.

In the context of self-adaptation, configurations are highly dependent on the operational conditions where ongoing operational states may invalidate their suitability. For example, when a reconfiguration operation erroneously sets the maximum number of requests parameter smaller than the current number of requests sent to a server process, it can introduce some inconsistencies thus can compromise the system's operation. Yet, most current works [5], [6] still provide structural checks that rely purely on configuration parameters, typically checking syntactical and type correctness, checking that configuration values remain within authorized bounds and respect some cross-elements dependencies.

Standards like the DMTF Common Information Model (CIM) [7] and the YANG data modeling language [4] include constraints constructs for configuration validation, nevertheless, their enforcement is left to implementors and solutions developers. Moreover, YANG's constraints concern configuration data only and do not admit a checking against current runtime states.

Regarding the verification approaches related to CIM/WBEM environments, our framework is very close to the work on CIM Apache Verifier [8] and SANChk [9]. They both stress the critical requirements of the verification of system's configurations with the use of formal constraint-checking techniques.

The approaches in [10], [11] also use CIM as a base information model for correct configuration synthesis at design time. In our case we do not directly rely on CIM for our validation framework, we use it as a case study to provide an integrated online checking solution for every WBEM management system.

The common limitations of these works consist of providing structural checks that rely purely on configuration parameters or confining configuration validation to design time. Our work targets online configuration checking and addresses particularly the issue of the influence of ongoing execution conditions on the validity of configuration instances. This is useful for complementing existing configuration validation approaches.

III. Validation Framework

This section presents the framework we propose along with a usage methodology.

A. Vision and Design Principles

The validation framework aims to offer an online configuration checking service that can be used by management systems without changing existing tools. It specifically targets current autonomous and self-management approaches. The principal characteristics of these autonomous approaches that we address, are the runtime management of the systems' dynamics and the rapidly changing service and architecture requirements. The framework supports these new requirements through three main design principles:

- Enabling an operational validation of configurations that takes into account their dependency on running execution states. Indeed, in the context of self-adaptation, configurations are highly dependent on the operational conditions, and ongoing operational states can invalidate the suitability of a candidate configuration.
- Allowing modification of validity properties at runtime: management systems are likely to have their requirements evolve at runtime, and these evolutions are to be translated at runtime into the creation or modification of properties on configurations.
- Supporting existing management systems in order to enhance their reliability with a validation functionality. This can notably be achieved by integrating existing management standards such as CIM.

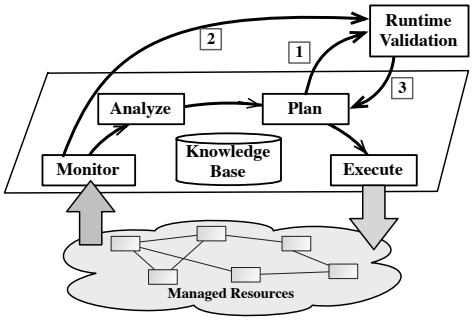

Fig. 1. The Vision of Online Configuration Checking

Fig. 1 illustrates our vision of online configuration checking. A runtime validation framework with platform-neutral interfaces, interacts with a management system (represented through the common management control MAPE loop) through the Plan block (1), center of runtime configuration decisions, while relying on the Monitor block (2) for the retrieval of the required ongoing execution states, and thus processes an online validation of configurations (3).

B. Overview of the Framework's Salient Features

The framework provides an integrated approach relying on a high-level specification language MeCSV that includes an online configuration checker. At design time, a human operator can use the MeCSV metamodel to formally specify a reference model, that is, a configuration schema of a given application domain with the desired architecture and service behavior constraints (Fig. 2 (a)). The reference model will be used at runtime for the evaluation of proposed configuration instances ("Ref. Model" on Fig. 2 (b)).

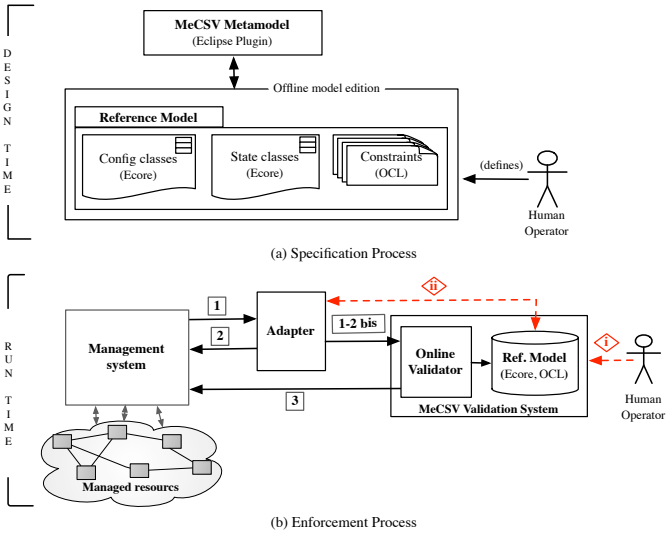

Fig. 2. Framework's Approach and Runtime Architecture

1) MeCSV Language: MeCSV is a high-level language dedicated to a formal representation of configuration information for runtime validation. It supports platform-neutral configuration specification including innovative features that enable validation against the runtime execution conditions.

a) Configuration Description: MeCSV includes the conventional constructs for configuration description such as `configuration parameter` (name and type) grouped into `configuration` classes, their composition and cross-references.

b) Operational Data Representation: Essential for the validation of the runtime suitability of configuration instances, MeCSV offers the constructs of `managed element` and `state parameter` for the representation of the monitoring view of a managed element. This allows connection to an existing monitoring framework.

c) Dedicated Constraint Model: MeCSV enables constraint definition over configuration data through two types of constraints, `offline constraint`, equivalent of usual structural rules and `online constraint` for the expression of operational validity rules whose evaluation depends on the availability of current runtime data (`state parameters`).

Constraints in MeCSV feature additional attributes for flexibility and lifecycle management, such as "constraint level" that allows modulation of their strictness and priority (e.g.; warning, error), in the case of soft constraints for example, and "active" that permits to activate or deactivate constraints depending on usage scenarios (e.g.; critical, non-critical).

We do not detail the MeCSV metamodel as well as instance examples in this paper. An extensive description of the language can be found in [12]. MeCSV is currently available as a UML profile and Ecore model [13]. Each has been tested with two popular modelers `Eclipse MDT` and `TopCased` but, any UML or Ecore modelers can be potentially used.

2) Target Domain Reference Model: The central objective of MeCSV is to allow the definition of a *Reference Model* that every possible configuration of the target system should conform to (Fig. 2(a)). The reference model can be viewed as a desired high-level system's architecture and service behavior in terms of configuration structure and constraints that should be respected.
The reference model is organized in a structural part made of configuration and state information that can be represented as UML or Ecore classes, and an assertion part containing offline and online constraints expressed in OCL [14], a formal specification language extension to UML. Operators or vendors can thus use the MeCSV metamodel at design time to define the reference model of a given managed application domain (e.g.; an application server, a messaging middleware or storage area networks) according to their management requirements.

This reference model is to be defined only once, however it can be modified at any time during the management system's life cycle to cope with the evolution of management and system requirements (cf. <i> on Fig. 2(b)). Once defined, it will be processed at each reconfiguration decision, to dynamically evaluate configuration instances.

3) Online Configuration Checker: The framework includes a runtime architecture, Fig. 2(b), with an online configuration checker and a model repository.

The model repository stores designed reference model classes and constraints. This model repository also supports the online edition of the reference model data and constraints to adapt the model to evolving management requirements.

The configuration checker verifies any configuration instance against the reference model. The checker is based on the open source Dresden OCL interpreter library [15] that we have enriched with MeCSV specific features such as offline and online constraints differentiation, constraint checking with filtering conditions depending on constraint metadata, and the extra functionality of online reference model modification.

The configuration checker is designed to process automatically MeCSV-compliant information (information modeled with the MeCSV language). Thus, to operate effectively with existing management platforms, the online checker must receive candidate configuration instances in a format it can understand, notably the checker requires serialized model instances conforming to the defined MeCSV reference model (XMI files). Therefore, a runtime adapter that operates as a sort of wrapper for the existing management system needs to be defined. The validation framework provides an adapter interface that can be implemented to tailor the online checker to a specific management system.

Once an adapter is provided, the system is ready to process validation. The legacy platform will send configuration checking requests containing platform-specific configuration instances to the appropriate adapter (1 on Fig. 2(b)). These validation requests will trigger retrieval of state parameters of interest (2 on Fig. 2(b)). Once retrieved, state parameters values and configuration values are packaged into a convenient MeCSV-compliant instance format and sent to the configuration checker (1-2 bis on Fig. 2(b)). The validator can then process the evaluation of those instances against the stored reference model and reports validation results (3 on Fig. 2(b)).

C. Usage Methodology

1) Specification Process: The specification process is a two-step process that starts at design time. First the representation of the reference model structural classes, then the expression of offline and online constraints.

The first step allows the representation, in a MeCSV-compliant way, of the structural part of the system reference model that will serve for online validation. Operators use the MeCSV metamodel to model configuration parameters and state parameters that can influence the suitability of a given configuration, organized into classes with convenient composition and dependency relationships.

The second step is the definition of constraints that configuration data should respect. Operators thus define the offline constraints that restrict the structure of configuration information and the online constraints that help evaluate the compliance of a given configuration information with the execution context at hand.

The completion of these two steps provides the reference model of the system that is to be uploaded into the validation system repository and serves as an input for the checking process.

This general process slightly differs when a management information model already exists. Indeed, the first step could be automated, mapping rules can be directly defined between the specific management model and MeCSV translating the legacy constructs into the related MeCSV ones. For example, one could use model-driven techniques such as model to model transformation or reflection for the implementation of such mapping rules. The second step of constraints expression remains identical.

2) Enforcement Process: The essential requirement for the runtime validation enforcement consists of the design of the runtime adapter that will allow existing management platforms and the online validation system to work together. This adapter will enable online translation of platform-specific configuration data and monitored data into a compatible format that can be processed by the configuration checker.

The validation framework offers a default adapter interface that can be extended to tailor the adapter to a given platform. The adapter interface provides a list of methods including a method for the acquisition of operational state data, required for the evaluation of online constraints, and a method for translating existing management information into a MeCSV-compliant format.

Note that this adapter is only needed when a legacy platform exists. A platform that uses natively the MeCSV format does not require an adapter.

IV. Application to the CIM/WBEM Standards

CIM (Common Information Model) and WBEM (Web-based Enterprise Management) are management standards defined by the Distributed Management Task Force (DMTF). CIM is a platform-neutral conceptual information model commonly used for describing and interchanging management data in systems, networks, applications and services and WBEM is a set of management and Internet standard technologies developed to unify the management of distributed computing environments built-upon CIM.

A. Design Time: Model Transformation from CIM to MeCSV

We started with the UML version of all the various CIM schemas classes, available on the DMTF website ("all_classes.xmi"). Relevant classes for a usage with our framework are those inheriting from `CIM_ManagedElement` and `CIM_SettingData` classes and their associations. `CIM_ManagedElement` provides information about the current state of an element and `CIM_SettingData` represents the configuration view of an element.

We used model-driven transformation techniques and defined mapping rules between the UML version of the CIM Core Model and the ECORE version of the MeCSV metamodel. More precisely, we used the Query/View/Transformation (QVT) transformation language with its implementation engine Eclipse QVT Operational (QVTo) [16].

QVT Operational is an implementation of the standard set of model transformation languages defined by the Object Management Group (OMG) that allows to define mapping rules between a source and a destination model. The mapping rules are automatically executed by a rule engine transforming any given instance of the source model to a new instance conforming to the destination model. For example, `CIM_ManagedElement` classes and their attributes are respectively mapped to MeCSV `Managed Element` classes and MeCSV `State Parameter` elements. `CIM_SettingData` classes and their contained attributes are transformed into MeCSV `Configuration` and subsequent `Configuration Parameter` elements. These rules are compiled by the QVTo transformation engine and can be executed each time a model conforming to the relevant CIM Core Model pattern is defined, automatically transforming it into a MeCSV reference model (structural part) to be completed by constraints.

B. Runtime: CIM to MeCSV Adapter

For the enforcement of the validation process, we also provided an adapter (CIM2MeCSV Adapter on Fig. 3) capable of mapping at runtime, CIM instances to MeCSV-compliant instances. In the context of the WBEM environment, this adapter was designed as a WBEM client (based on the SBLIM CIM client API), indeed the adapter needs to gather state parameters from the WBEM server (in the experiment, we used the `Openpegasus` WBEM server).

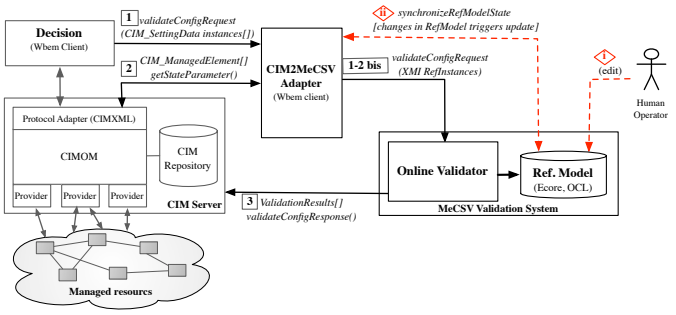

Fig. 3. Integration with CIM/WBEM: runtime architecture

Fig. 3 depicts the subsequent online checking process: the CIM2MeCSV runtime adapter is capable of communicating with a "configuration decision" that requests configuration validation of some `CIM_SettingData` instances (1), and querying the WBEM server at runtime for state parameters of interest (2). The adapter is then capable of dynamically translating the CIM instances into to MeCSV-compliant instances (1-2 bis) and sends them for validation.The online validator is then capable of checking the instances against the available reference model stored in the model repository (3).

The reference model can be edited at runtime if needed (<ii> on Fig. 3). In this particular case, modifications of the structural part of the reference model are notified to the adapter for the update of its wrapper capabilities, especially when new state parameters to query are added.

C. CIM Virtual System Profile Case Study

This prototype integration with the CIM/WBEM standards has been applied to several *CIM Virtual System Profile* models taken from the CIM Virtual System Profile document [17] (Fig. 4).

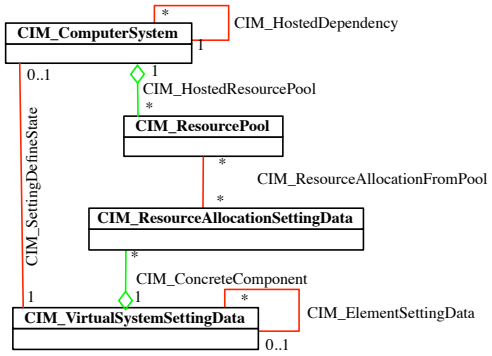

Fig. 4. Subset of CIM Virtual System Profile

At design time, the related MeCSV reference model has been successfully generated and exported into the validation system.

The system's architecture features a machine hosting some virtual machines (`CIM_ComputerSystem`) and providing a pool of memory, processor and disk resources (`CIM_ResourcePool`) . By multiple reconfigurations (of `CIM_VirtualSystemSettingData`), the architecture grows to several virtual machines sharing different resource pools according to resource allocation schemes (`CIM_ResourceAllocationSettingData`).

Possible operational data of interest are the current resource usage that can influence the suitability of resource allocation (its reservation) during a reconfiguration. A reconfiguration that fails to allocate appropriately available resource can cause a degradation of a virtual machine's operation. Examples of offline and online constraints that have been tested:

1) A virtual machine setting data should include a memory and a CPU resources allocation.
2) A resource allocation setting data reservation should be provided.
3) The current resource allocation (reservation setting) should not be greater than the available resource capacity.
4) The sum of the total resource reservations for all the virtual machines on the host should not attain 90% of the host's resource capacity.

At runtime, validation of candidate `CIM_VirtualSystemSettingData` instances with related `CIM_ResourceAllocationSettingData` instances are requested. The validation request triggers the CIM2MeCSV adapter connection to the WBEM Server for the retrieval of additional operational parameter information then translates the configuration data and the operational data into a MeCSV-compliant format and sends the result to the validation system.

V. Evaluation

This section presents the evaluation of the prototype integration of the CIM and WBEM standards and discusses observations and results. We respectively test the ability of the MeCSV metamodel to serve as a formal specification notation and evaluate the time taken by the `CIM2MeCSV` adapter to translate CIM instances into MeCSV-compliant ones. We also measure the validation time of the online configuration checker.

A. Evaluation Scenarios

We performed our experiments on three different virtual system settings varying in size and complexity.

The first case (Case 1) features a host with a unique virtual machine for a total of 180 configuration parameters (from `CIM_SettingData instances`) to manage. The second case (Case 2) consists of a host with two virtual machines and three different types of resource pools to be allocated on each virtual machine (about 360 configuration parameters). The third (Case 3) consists of three virtual machines with three different types of resources to allocate on each machine (800 configuration parameters). Besides, 140 state parameters (from 17 `CIM_ComputerSystem` and `CIM_ResourcePool instances`) values are to be retrieved at each request for the evaluation of online constraints. For each proposed configuration, the validator gradually ran validations with 10, 50 and 100 OCL constraints.

We took 100 measurements of the execution time in milliseconds for each validation request, and computed the arithmetic mean. The tests were run on a Intel® Core™ 2 Duo with 2.66 GHz and 4 Gigabytes of main memory. The validation framework hosting machine and the `Openpegasus` WBEM server are in the same network domain and wired by a 100 Megabits/second connection.

B. Results and Discussions

Table I presents the execution time for each configuration scenario.

1) Framework's Application to CIM/WBEM: The overall experiment shows that we successfully applied the framework's methodology in a CIM/WBEM environment. At design time, defined mapping rules allowed to translate a CIM management model to a MeCSV reference structural model that we enriched with constraints. At runtime, a CIM2MeCSV adapter translated CIM instances into a MeCSV compatible format and allowed the configuration checker to handle validation requests and report validation results.

TABLE I. Performance results

	Case 1	Case 2	Case 3
Adapter: total time (ms)	644.46	1,039.99	1,651.57
Adapter: instance translation only (ms)	170.82	175.37	188.52
Validation time with 10 constraints (ms)	183.33	237.50	323.75
Validation time with 50 constraints (ms)	331.00	403.75	565.00
Validation time with 100 constraints (ms)	482.25	663.00	868.50

2) Adapter's Overhead: The overall adapter time includes both the querying of the WBEM server for the retrieval of 140 state parameters and the transformation of CIM instances to MeCSV-compliant ones. The high values observed are due

to the WBEM server querying. Indeed, it does not suffice to retrieve just the CIM classes instances that contains the 140 state parameters; in order to translate them, the CIM2MeCSV adapter further acquires all the CIM Association instances in which they are involved.

While the transformation time itself is globally under 200 ms, the rest of the value is due to the navigation through CIM associations and the connection with the WBEM server. These measurements offer very encouraging results for a small-scale system's configuration. The adapter total time can be reduced with some optimizations techniques (for e.g.; connection bandwith to the WBEM server, caching).

3) Validation time: The total checking time for the three deployed scenarios is under 1 second which is also very encouraging. A very interesting result lies in the effect of the number of configuration parameters and constraints on the validation time. The validation time is not proportional either to the size of configuration instances or to the size of constraints. For example, while the number of CIM_SettingData instances quadruples from Case 1 (180 configurations parameters) to Case 3 (800 configurations parameters), the ratio of their average validation time hardly doubles (ratio is 1.8). Similarly, although the number of constraints increased by ten, from Case 1 to Case 3 the average validation cost barely tripled (2.8). These results confirm previous experiments on a message oriented middleware [18].

Altogether, the complete process time is very encouraging yet several points remain to clarify. We need to experience the methodology further in clarifying the costs of integration and usage. Indeed, while the framework does not require changing existing tools, it however needs to be adapted to the existing management system thus requires implementation of model mapping rules and a runtime adapter interface. Furthermore, even if the design time mapping is automated, expressing constraints on the resulting MeCSV reference model can required to learn MeCSV and how the new model works.

Another challenge is the dual evolution of both models (existing management and MeCSV reference models) over the management lifecycle. We stress the importance of the full operation of this evolution to avoid errors and ease the adoption. These points are to be explored in future work.

VI. Conclusion and Future Work

Designing integrated and lightweight online validation approaches is a critical requirement if we are to build reliable self-adaptive management systems. This is fundamental as misconfigurations can be prejudicial to the proper operation of the system. This paper presented a validation framework including an online configuration checker relying on a high-level language named MeCSV. The main objective of this framework is to provide a configuration validation service for existing management systems, that includes verification capabilities based not only on structural checks but also on running operational conditions.

We then proposed a methodology for the usage of the validation framework from design time to runtime, especially when dealing with legacy management information models. We applied this methodology to integrate CIM/WBEM environments thanks to both the definition of mapping rules from CIM modeling patterns to MeCSV constructs and the implementation of a runtime adapter capable of retrieving state parameters from a WBEM server. Experiments with a subset of the CIM Virtual System Profile model allowed us to discuss results and observations demonstrating the feasibility of the approach.

In future work, we intend to further experience the methodology by integrating more legacy systems and case studies so that we can ease the integration process and lower subsequent costs of integration and implementation.

References

[1] P. Anderson and E. Smith, "Configuration tools: working together," in *Proceedings of the 19th conference on Large Installation System Administration Conference*, 2005.

[2] M. Burgess and A. L. Couch, "Modeling Next Generation Configuration Management Tools," in *LISA*, 2006, pp. 13–147.

[3] R. Enns, "NETCONF Configuration Protocol," Internet Engineering Task Force (IETF), RFC 6241, june 2011.

[4] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," Internet Engineering Task Force (IETF), RFC 6020, October 2010.

[5] I. Warren, J. Sun, S. Krishnamohan, and T. Weerasinghe, "An Automated Formal Approach to Managing Dynamic Reconfiguration," in *ASE'06: Inter. Conference on Automated Software Engineering*, 2006, pp. 37–46.

[6] T. Delaet and W. Joosen, "PoDIM: A Language for High-Level Configuration Management," in *LISA*, 2007, pp. 261–273.

[7] "CIM Schema version 2.29.1 - CIM Core," Distributed Management Task Force (DMTF), June 2011.

[8] C. Sinz, A. Khosravizadeh, W. Kuchlin, and V. Mihajlovski, "Verifying CIM models of Apache Web-server configurations," *Proceedings of the 3rd International Conference on Quality Software, 2003*, pp. 290–297, 2003.

[9] E. Gençay, C. Sinz, W. Küchlin, and T. Schäfer, "SANchk: SQL-based SAN configuration checking," *IEEE Transactions on Network and Service Management*, vol. 5, no. 2, pp. 91–104, 2008.

[10] T. Hinrichs, N. Love, C. J. Petrie, L. Ramshaw, A. Sahai, and S. Singhal, "Using object-oriented constraint satisfaction for automated configuration generation," in *DSOM*, 2004, pp. 159–170.

[11] L. Ramshaw, A. Sahai, J. Saxe, and S. Singhal, "Cauldron: a policy-based design tool," in *7th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2006, pp. 113–122.

[12] L. Akue, E. Lavinal, and M. Sibilla, "A Model-Based Approach to Validate Configurations at Runtime," in *4th International Conference on Advances in System Testing and Validation Lifecycle (VALID)*, November 2012, pp. 133–138.

[13] "Eclipse Modeling Framework Project (EMF)," The Eclipse Foundation, january 2013. [Online]. Available: http://www.eclipse.org/modeling/emf/

[14] "Object Constraint Language (OCL), Version 2.0," Object Management Group (OMG), May 2006.

[15] "Dresden OCL," TU Dresden, Software Technology Group, january 2013. [Online]. Available: http://www.dresden-ocl.org/

[16] "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.0," Object Management Group (OMG, april 2008. [Online]. Available: http://www.omg.org/spec/QVT/1.0/PDF/

[17] "Virtual System Profile," Distributed Management Task Force (DMTF), april 2010.

[18] L. Akue, E. Lavinal, T. Desprats, and M. Sibilla, "Runtime Configuration Validation for Self-configurable Systems," in *IFIP/IEEE International Symposium onIntegrated Network Management (IM)*, may 2013, pp. 712–715.