A Framework for Autonomic, Ontology-based IT Management

Fabian Meyer
Hochschule RheinMain
University of Applied Sciences
Distributed Systems Lab
Wiesbaden, Germany
Email: fabian.meyer@hs-rm.de

Reinhold Kroeger
Hochschule RheinMain
University of Applied Sciences
Distributed Systems Lab
Wiesbaden, Germany
Email: reinhold.kroeger@hs-rm.de

***Abstract*—The growing complexity and heterogeneity of modern IT systems demand for intelligent management tools, capable of horizontally integrating technologies of different vendors and domains and vertically relating them with business processes and high level requirements. Due to their often hard-coded and unextendable management models, existing tools are not able to meet those requirements well. Recent advances in semantic web technologies let ontologies experience a revival for the modeling of domain knowledge and new standards as the Web Ontology Language (OWL) have been established by the W3C. The abilities to semantically model, map and link independent domains let them appear as very suitable for the application in IT management. Nevertheless, approaches for the implementation of an ontology-based IT management have shown that the scalability for large domain models is insufficient, important aspects are not expressible using the standards and the integration of continuous monitoring data is difficult. In this paper, dedicated solutions for each of those problems will be presented and combined into a comprehensive, ontology-based management framework. The proposed approach has been experimentally validated in a case study of a medium-sized management problem for an air traffic management system.**

I. Introduction

Modern IT systems are heterogenous conglomerates of diverse components from different vendors, in many cases patched-up poorly. Due to the increasing digitalization of companies, the implementation of business processes and hence the realization of corporate objectives entirely depends on a smooth operation of the underlying IT systems. As a consequence, IT management has become an indispensable ingredient for business success.

Today, monitoring and assurance of certain Quality of Service (QoS) measures as contracted in Service Level Agreements (SLAs) and compliance to IT governance specifications play a significant role in IT management. The origin of the agreed terms is manifold. Internal guidelines between the IT and departments within a company, outsourcing contracts or governmental regulations demand the strict observance of rules and thresholds for Service Level Objective (SLOs) in order to avoid contractual infringements and associated penalties. Considering the complexity of modern IT environments, manual management has become impossible.

Automated management tools already support administrators in acquisition, consolidation and analysis of various data sources. Yet, the variety of management standards and technologies requires the deployment of multiple isolated solutions without a shared context. An overarching view originates only in the heads of human operators, resulting in a high manual effort and error-proneness. Tools for a joint management exist, but in general are limited to the product range of one specific vendor and typically rely on hard-coded information models and management logic, thus making the integration of new domains more or less impossible.

Recent advances in semantic web technologies let ontologies experience a revival for domain modeling. Supported by theoretical research in the field of description logics, W3C released the Web Ontology Language (OWL). Compared to existing modeling languages such as UML, domains are modeled semantically instead of just syntactically. Hence, the taxonomy (TBox) and instance data (ABox) can be validated and new knowledge can be derived automatically, using automated inference engines (reasoners). Since ontologies can be extended, combined and linked to each other, they can be regarded as promising candidates for horizontal and vertical integration of different management domains. First approaches on applying ontologies in IT management have confirmed that impression (see section II), yet some essential problems have to be solved.

For a holistic modeling of management logic, OWL lacks some important features such as functional dependencies and aggregations [1] as well as a designated concept for temporal knowledge representation, limiting to a mere snapshot consideration, unaware of past states. Hence, external components have to be deployed for data storage, analysis and decision making. Furthermore, the inclusion of continuous monitoring data proves to be difficult. The flood of information requires a context-aware preprocessing and smart semantic lifting algorithms, not viable by hard-coded components, working purely syntactical. The resulting fragmentation of management logic leads to a hard to maintain and error-prone management process. Additionally, the large amount of instance data and high update frequency paired with the complexity of OWL reasoning algorithms result in poor performance and scalability.

Our goal is to build a general framework for uniform, ontology-based management of modern IT systems. We will present dedicated solutions for the identified problems and combine them into a comprehensive management framework.

978-3-901882-77-7

Section II presents related work on ontology-based IT management. Section III addresses the enhancement of reasoning performance, the representation of temporal knowledge and the modeling of management semantics. Sections IV introduces our novel management model, offering a uniform way to express the different aspects of ontology-based management. Section V introduces our management architecture. A case study and corresponding evaluation results are presented in section VI, before a conclusion is given in section VII.

II. Related Work

Jorge E. López de Vergara et al. firstly expose the benefits applying ontologies for IT management [2]. They investigate the expressive power of well-established management models [3], using an ontology classification schema presented in [4] and reveal the Web Ontology Language to be an appropriate format for mapping and merging them [5]. In [6], they outline a policy-based management architecture, using OWL as information model and the Semantic Web Rule Language (SWRL) for determining SLA compliance and execute management operations. Use cases for the application of the architecture and a vertical integration using SWRL rules are presented in [7], [8], [9] and [10].

In [11] and [12], Debao Xiao et al. present an approach similar to [7], using OWL as information and SWRL rules as behavior and control model. A use case is presented, using Management Information Bases (MIBs) to derive the information model and the Simple Network Management Protocol (SNMP) to obtain management information.

The Reaction after Detection (ReD) architecture presented in [13] consists of a preceding event processing engine, detecting attacks based on intrusion detection messages, which are further mapped to an ontology where SWRL rules are applied to extract a threat context. The context is combined with policies, modeled in the Organization Based Access Control (Or-BAC) [14], [15] ontology and SWRL rules are applied to identify attacker and target. A reaction component generates and establishes network policies, blocking the attack.

In [16], Liam Fallon et al. present the Aesop management framework, using the End-User Service Management Ontology (EUSAO) [17] as information model. A Monitor Analyze Plan Execute (MAPE) loop is implemented for autonomic end-user service experience optimization in home area networks. Monitoring data is mapped to a knowledge base, partitioned by dynamics and temporal properties of the contained data. An analysis process periodically creates a comprehensive view and uses SPARQL queries to identify non-compliant sessions. In order to restore SLA compliance, a hard-coded algorithm plans and executes throttling and unthrottling actions.

In [18] and [19], Annie Ibrahim Rana et al. present an approach where network policies are translates into SWRL rules, which are applied to semantically lifted network packets combined with domain knowledge. Based on the rule matches and the original policies, IP tables are generated and distributed onto deployed network devices.

A semantic monitoring and management framework is presented in [20], using raw service messages, hardware monitoring data, symptom-based pattern matching and anomaly detection as input for the Semantic Attribute Reconciliation Architecture (SARA) [21] in order to create semantically enriched data, using domain expert rules. The semantic data is used as input for a visualization component and the Semantic-based Service Control Engine (2SCE), implementing user-defined policies.

III. Dedicated Solutions

In this section we will analyze the problems outlined in section I in detail and present dedicated solutions, which will later be integrated into our management framework.

A. Reasoning Performance

Existing ontology-based management approaches as presented in section II generally use OWL-DL ontologies for the representation of domain knowledge, exploiting $\mathcal{SHOIN}^{(\mathcal{D})}$ description logics as theoretical foundation and hence using standard description logic reasoners based on the analytic tableaux for the standard reasoning task. The tableau calculus is perfectly adequate for consistency checking and querying facts of a static or slowly changing knowledge base, but considering the high information update frequency of complex systems, the algorithm is not suitable for IT management.

The recently introduced recommendation for OWL 2 defines three profiles [22], each designated for a specific field of application. OWL-EL is designed for the efficient analysis of complex TBoxes, OWL-QL for processing large amounts of instance data, utilizing a strongly constrained language subset and OWL-RL for scalable reasoning in a rule engine, without losing to much expressiveness.

We compared the expressiveness of the profiles with the results of the ontological classification of existing IT management models from [3] and the ontological conversion of the established and comprehensive Common Information Model (CIM) from [23]. The analysis revealed the RL profile as most fitting, covering the expressiveness required for the representation of existing IT management models (except for cardinality restrictions greater than one, as only used in CIM), furthermore being capable of processing large amounts of instance data in a rule engine. Hence, the application of OWL-RL can increase reasoning performance.

B. Representation of Temporal Knowledge

Time plays an important role for many aspects of IT management, enabling the consideration of courses, trends and previous system configurations to ensure SLA compliance. OWL has no designated concept for representing temporal knowledge, yet several approaches for an extension exist [24].

We consider the 4D fluents approach [25] as most promising. It is compliant to the OWL standard (making standard modeling tools and reasoners applicable) and can be used to augment existing domain models without changing them. Temporal relations are attached to temporal parts of the original individual, each valid during a time interval. The state of an individual at a certain point in time is the union of the states of all its currently valid parts.

The 4D fluents approach was taken up and extended by different concepts and roles to clarify relations between

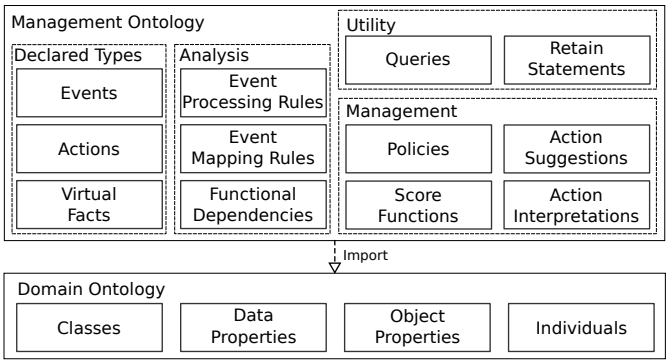

Fig. 1. Structure of an OntML module.

temporal parts in [26] and [27]. It was also applied to different domains for the spatio-temporal representation of objects (e.g. [28] and [29]).

C. Knowledge Fragmentation

The combination of semantic ontology models, event processing and automated management leads to a severe knowledge fragmentation, resulting in hard to maintain and error prone management processes. Hence, a uniform management model is required, capable of expressing all these aspects. Since no such model is known to the authors, we decided to develop the Ontology-based Management Language (OntML), presented in section IV.

IV. Ontology-based Management Language

The Ontology-based Management Language (OntML) aims to provide a management model for ontology-based management, encompassing a uniform way to express context-aware event processing and mapping, functional dependencies within the management domain, management policies for a fast, automated resolution of isolated problems, autonomic management procedures for the resolution of complex problems and housekeeping mechanisms for knowledge base maintenance. The goal of OntML is not the extension of the OWL semantics and the standard reasoning task, it rather offers a possibility to model the management logic in a declarative, uniform and centralized way.

An ontology vocabulary provides types and properties to augment domain ontologies for management semantics in order to achieve a tight integration of the information model and the management logic. OWL's modularization concept allows the management logic to be either part of the domain ontology itself or to be separated into a dedicated management ontology, importing the domain model.

The root concept of OntML is the management module (see figure 1), containing a set of type declarations, analysis logic, management logic and utility functionality. Type declarations define the structure of events, actions and virtual facts, used by the management module. Events represent the different manifestations of monitoring events from the managed system as well as complex events, correlated during the management process. Actions describe commands, executable on the managed system in order to control its behavior (reconfiguration). Virtual facts are auxiliary constructs to store coherent, deduced data. Each declared type has a set of named attributes, either typed with a declared or a primitive (boolean, float, integer, long, string and individual) type. Declared types can be shared between OntML modules.

The analysis logic is used to infer the state of the managed system, based on the current configuration and the observed monitoring data. Event processing rules are production rules, processing raw monitoring or complex events in order to derive new complex events or filter existing ones. The event processing is context-aware, meaning the rules can incorporate with the ontology. Event mapping rules map events to the ontological representation of the managed system by adding, removing or overwriting assertions. Functional dependencies are logical implications, deriving new class or property assertion based on the ontology's current state.

The management logic is subdivided into automated and autonomic management. Automated management is implemented through policies. A policy is a production rule, observing the state of the knowledge base in order to control the system by executing appropriate actions. Autonomic management is composed of action suggestions, action interpretations and score functions. Action suggestion rules are logical implications of possible actions, currently executable on the managed system. Action interpretations are action equivalents of event mappings, describing the expected impact of an executed action. Score functions are logical implications of score objects, expressing the systems healthiness.

Utility functionality is provided through queries and housekeeping rules. Queries are logical implications of virtual facts, allowing rule modularization without affecting OWL reasoning (as functional dependencies do). Retain statements are used for housekeeping, by defining lifespans for temporal property or classes, stating how long terminated assertions should be retained.

The representation of the different rule types in OntML is based on the rule structure of the W3C Rule interchange format [30]. Yet some restrictions have been applied and support for temporal operators has been added.

Rules are subdivided into rule implications and production rules. Implications consist of a single term (the implication) and a condition formula, production rules have a condition formula and a consequence. A term binds an event, action, virtual fact or property/class assertion, based on its attribute values. The attributes can either be restricting using relations and expression or unified with variables.

A condition formula is a term, an equality term or a conjunction, disjunction, universal quantifications or negative existential quantification of a condition formula. Equality terms unify expressions, terms or aggregations with a variable. Aggregations apply functions (sum, average, min., max., count) on a set of values, bound in a nested condition formula. terms of temporal types (events or classes/properties flagged as temporal) can be restricted by sliding windows (e.g. within 10 seconds), points in time (e.g. 5 minutes ago) or Allen relations (e.g. before x) [31].

Consequences of production rules can insert/retract events into/from the event stream, create/delete/overwrite class and property assertions or execute actions on the managed system.

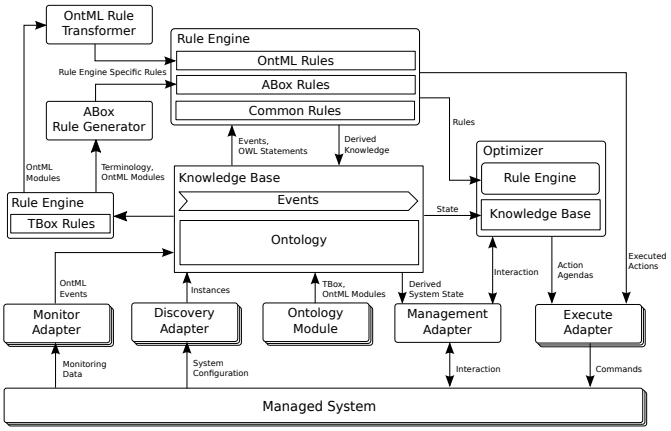

Fig. 2. Function blocks and dataflow of the management framework.

V. Management Framework

Our proposed ontology-based management framework implements the Monitor Analyze Plan Execute Knowledge (MAPE-K) paradigm [32] by combining the dedicated solutions introduced in section III and the management language introduced in section IV. An overview of the resulting architecture is depicted in figure 2.

A. Function Blocks

The cornerstone of the architecture is the knowledge base, containing an ontological representation of the managed system, the monitored events from the runtime system and the aggregated complex events. So-called ontology modules encapsulate the domain and management ontologies, offering dependency management and resource loading mechanisms. In many cases, domain models can be derived from existing information models automatically. Such a syntactical transformation results in a basic ontology, which has to be enriched by semantic information manually. By importing the 4D fluents ontology, entities of the domain model can be flagged as temporal, allowing the appropriate processing during runtime. Management ontologies import and extend domain ontologies for management information by using the OntML vocabulary from section IV.

The central logic component of the framework is a rule engine, implementing three different rule sets. The common rule set defines the domain independent logic. It contains types and rules for processing temporal knowledge, collecting outdated facts and offering complex ontology operations to the other rule sets. The ABox rule set implements the OWL-RL ABox reasoning. Its rules are generated by the ABox rule generator dynamically, based on the loaded domain ontologies and the OWL-RL rules from [22]. The management logic is implemented by the OntML rule set, which is generated from the OntML modules, contained in the loaded management ontologies.

Autonomic management functionality is implemented by the optimizer component. It deploys a copy of the rule engine and the knowledge base, trying to optimize the configuration of the managed system by simulating and evaluating different action chains.

Besides the development of domain and management ontologies, a mediation layer consisting of different adapters has to be implemented for each managed system.

Initial discovery and import of the system's configuration is implemented by discovery adapters, typically using system specific discovery mechanisms (reading a CMDB or configuration file, scanning the infrastructure, etc.) to reveal the structure of the managed system, before translating it into ontology statements, according to the domain ontology.

Monitoring events of the managed system are interfaced by monitoring adapters. They collect monitoring information from the underlying system (reading log files, using management interfaces, etc.) and translate them into OntML events, before writing them into the event stream of the knowledge base.

The execution of reconfiguration commands on the managed system is implemented by execute adapters. They receive the OntML actions from the rule engine and optimization component and translate them into system specific commands, which are subsequently executed on the managed system (e.g. using management interfaces, logging onto a host and executing shell commands or creating tickets for a system administrator).

A management adapter serves as northbound interface, constantly accessing the derived system state, passing it to external components (visualization, reporting, etc.). Furthermore, it can interfere with the management process by triggering the optimization process manually. Reconfiguration agendas acquired by the optimizer have to be approved by the management adapter before they are executed on the managed system. In general, the agenda is passed to a human operator in order to make the decision and bear the responsibility for carrying it out.

B. Runtime

During runtime, the first step is the population of the knowledge base with domain and management ontologies, by using the appropriate mechanisms of the ontology modules. A TBox reasoning is performed on the unified ontology, checking its consistency and deriving facts, using the rules for the semantics of the schema vocabulary from [22].

The ABox rule generator uses the derived knowledge to generate domain specific OWL-RL ABox rules. In order to enhance reasoning performance during runtime, only rules are generated deriving management relevant facts (classes/properties referred to by the loaded OntML modules), including the respective inference chain. A drawback of generating fixed ABox rules during startup is a static TBox, which cannot be changed dynamically during runtime easily, but requires a re-generation of the rules.

The OntML transformer extracts the OntML modules from the knowledge base and performs a Model To Text (M2T) [33] transformation. Each declared type is transformed into a Java class, reflecting the specified structure. Each rule implication and production rule is transformed into a rule engine specific rule. Thereby, terms referencing temporal domain entities have to be translated into pattern for corresponding 4D fluents structures in condition formulas and into commands creating

the corresponding 4D fluents structures in implications and consequences.

After knowledge base and rule engine are initialized, the adapters of the managed systems are created. The discovery adapters are executed, filling the knowledge base with instance data according to the taxonomy of the domain ontologies. The rule engine is put into a continuous processing mode, automatically considering knowledge base changes in order to derive the system's state and healthiness, create complex events and execute actions. Finally, the monitor, management and execute adapters are started, mediating between the managed system and the framework by forwarding events and executing commands.

When an optimization is started, the management framework creates a temporary copy of the knowledge base and the rule engine, in order to work in parallel. Both are passed to the optimizer, which uses a hill climbing algorithm to generate a reconfiguration agenda. The optimizer extracts all currently suggested actions from the temporary knowledge base and executes them consecutively, using the defined intepretation rules. The resulting score is mapped with each action and the state of the knowledge base is rolled back, before the next action is evaluated. After all suggested actions have been considered, the optimizer selects the action with the best score, executes it on the temporary knowledge base and starts evaluating the next step. The termination criterion of the optimization is reaching a local optimum. If the acquired agenda is approved by the management adapter, the selected actions are passed to the execute adapters.

C. Implementation

A prototype of the presented management framework was implemented using Java as programming language, OSGi as service framework, Apache Jena for ontology handling and Drools as rule engine.

Given that the manual modeling of the OntML modules including types, rules, etc. using standard ontology editors such as Protégé is very inconvenient and error prone, a domain specific language with IDE integration was developed, using the Eclipse Xtext framework. Ontologies can be imported in order to write management modules referencing the domain entities. Management ontologies are generated as output, containing instances and assertions according to the OntML vocabulary. The framework consists of 12 OSGi bundles with 10,850 lines of Java code.

VI. Case Study

The prototype was applied to a real world scenario of our project partner DFS Deutsche Flugsicherung GmbH, the German air traffic control agency. The data assessment, transfer, consolidation, processing and visualization of different sources such as radars, flight plans and weather information is implemented by so-called Air Traffic Management (ATM) systems. Such critical environments require the whole infrastructure to be fault-tolerant, resulting in full hard and software redundancy. Consequently, every tower (controlling of starts, landings and surface movements) and center (controlling of overflights) has deployed two different ATM systems in parallel, resulting in a highly complex infrastructure. Governmental

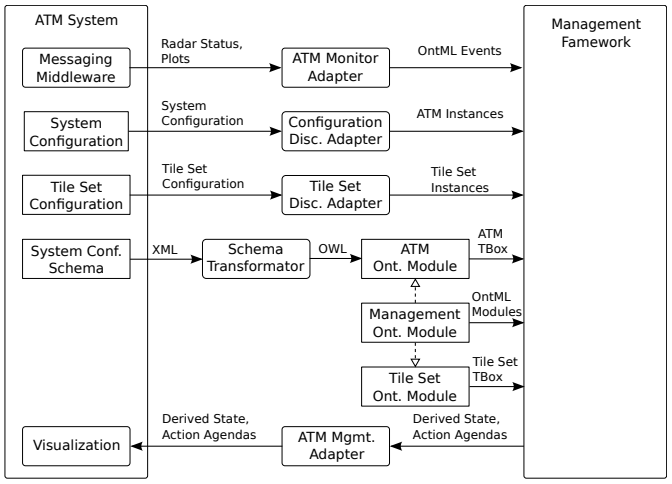

Fig. 3. Adaptation of the framework to the use case of DFS Deutsche Flugsicherung GmbH.

regulations and internal SLAs make the manual management of such complex systems impossible. In the following, we will present a management use case from the ATM context, which was automated by applying the management framework.

The primary ATM system implements a tile-set-based radar data processing, subdividing the controlled area into so-called tiles of fixed size. Each tile has up to six assigned radars, producing surveillance data used for the tracking of aircrafts in the area. Radars can support up to three different features: primary surveillance, second surveillance and mode-s. An SLA defines a minimum feature coverage per tile. Domain experts, familiar with the characteristic of each radar (features, field of vision, etc.), create tile configurations compliant to the SLA.

Despite redundancy, hardware and software failures (sensors, network infrastructure, etc.) can result in SLA violations during operation. Currently, a process checks the SLA compliance on a daily basis using the traversed states of each radar and generates a report. In case of violations, a coverage detection is executed manually, revealing possible surrogate radars using the recent plots (radar observations). Subsequently, a new tile set configuration is created. The currently deployed procedure includes offline analysis and a lot of manual interaction, resulting in a possible operation in a non-compliant state for hours.

Our goal was to use our prototype to automate this process, implementing online monitoring of SLA compliance and generation of traceable reconfiguration proposals. Figure 3 depicts the resulting architecture of the adapted management framework.

For the tile set configuration, a tile set ontology module and a tile set discovery adapter was developed. For the radar configuration a schema transformator was implemented, transforming the existing configuration schema into an ontology. A configuration discovery adapter maps the file-based system configuration to the knowledge base. The ATM monitor adapter hooks on the messaging infrastructure of the ATM system, extracting radar status messages and plots.

A management adapter continuously extracts the state of each tile from the knowledge base, forwarding it to a manage-

ment console. Furthermore, the component obtains proposed reconfiguration agendas from the optimization component and suggests them to a system administrator.

The adaptation consists of 5 OSGi bundles with 2,626 lines of Java code, implementing schema transformation, config parsing and communication with the managed system. Since automated reconfiguration is not allowed in such a critical environment, reconfigurations are carried out by administrators manually and no execution adapter exists.

The OntML editor was used to define the events, actions and rules for the scenario, using the two domain models. A radar status and a plot event represent the monitoring information from the system. Assert radar and replace radar actions represent the possible operations on the tile set.

The analysis logic consists of rules, relating the two different radar concepts from the domain models (one functional dependency), mapping the state of radar status messages on temporal property assertions of the corresponding radar individuals in the ontology (one event mapping), determining the SLA compliance for each feature of each tile, based on the radar status (four functional dependencies), relating each plot to a corresponding radar and tile individual, using the sensor identifier and the location (five queries), consolidating plots to enhance performance, by removing plots of tiles already having the radar assigned and duplicated plots (two event processing rules).

The autonomic management logic consists of rules, suggesting the assertion of new radars or the replacement of existing radars for non-compliant tiles, based on the coverage of the radars within the last 24 hours (six action suggestions), defining the interpretation of the actions on the knowledge base (two interpretation rules) and calculating the healthiness of the system based on the SLA infringement (three score functions).

Overall, the scenario consists of three event definitions, two action definitions, six virtual facts, nine functional dependencies, five queries, one event mapping rule, two event processing rules, six action suggestion rules, two interpretation rules and three score functions. During runtime, 28 OntML and 66 OWL-RL rule engine specific rules are generated.

An experimental setup was deployed on a desktop system (Intel Core i7-3770 CPU @ 3.40GHz, 16GB RAM) and the management system was tested under real conditions, processing recorded plot data from the live system in real-time (approx. 1000 plots per second). Every ten minutes, the failure of three radars was simulated, leading to 29 violations. After failure detection, an optimization was triggered by the management adapter, evaluating 121 actions, resulting in an eleven step optimization agenda, reducing the violations from 29 to 8. The management cycle from radar failure to reconfiguration proposition took 10.8 seconds in average. The CPU utilization settled for 1.5% and the memory consumption for 636 MB in average, peaking temporary up to 76% and 1050 MB during optimization. The summarized performance statistics are shown in table I.

The use case has shown that the developed framework is capable of handling real management scenarios with large information models and high event rates.

TABLE I. STATISTICS OF THE SCENARIO.

	Avg.	**Std. Dev.**
Violation detection (ms)	1788	420
Optimization setup (ms)	6364	707
Optimization simulation (ms)	2639	681
Fact count	81744	991
Rule calls (s^{-1})	2301	312
CPU usage (%)	1.44	4.09
Memory consumption (MB)	636	129

VII. CONCLUSION

In this paper, a novel approach for an automated, knowledge-based management framework was presented.

Existing approaches on ontology-based IT management were examined, unsolved problems exposed and dedicated solutions introduced for each of them. For reasoning scalability, the application profiles of the recent OWL 2 recommendation were compared with the expressiveness of existing IT management ontologies and the RL profile was selected as best fitting and performing for the management of IT systems. Due to its processability by standard tools and reasoners, and its ability for augmenting existing domain models without a need for adjustment, the 4D fluents approach was applied for the representation of temporal knowledge. The uniform expression of management logic, incorporating event processing and automated reconfiguration, was achieved by defining the Ontology-based Management Language (OntML), facilitating the modeling of management types and rules as part of the domain ontology. The dedicated solutions were combined into a comprehensive management framework, implementing the MAPE-K autonomic loop. The prototypical implementation of the framework and its application to the management of an air traffic management system has shown its practicability and efficiency in handling medium complex management problems. Especially, the following advantages have been experienced:

The selection of the OWL-RL profile for domain ontologies allowed us to deploy a rule engine for reasoning in our framework, enhancing the performance significantly when processing large ontologies. The uniform representation of the different management aspects and its tight, in-ontology integration with the domain knowlege resulted in a minimization of the knowledge fragmentation and hence reduction of configuration effort and error-proneness. Supporting the representation of temporal knowledge for domain models enabled comparison with previous system configurations, observation of trends, etc. as an integral part throughout the whole management process. The mechanisms for suggesting, interpreting and scoring reconfiguration alternatives faciliated the application of local optimization algorithms for autonomic management. The framework's adaptable design, providing different interfaces for the integration of custom domain models, management models and technology adapters, makes it very accessible and versatile.

Future work will apply the framework in the storage management domain, extend the management language for pluggable analysis modules (e.g. for machine learning) and support the interchangeability of the optimization algorithm, making the framework more adaptable to different use cases.

REFERENCES

[1] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "Owl 2: The next step for owl," *Web Semant.*, vol. 6, no. 4, pp. 309–322, Nov. 2008.

[2] J. L. De Vergara, V. A. Villagrá, and J. Berrocal, "Semantic management: advantages of using an ontology-based management information meta-model," in *Proceedings of the HP Openview University Association Ninth Plenary Workshop (HPOVUA'2002), Böblingen, Germany*, 2002, pp. 11–13.

[3] J. L. De Vergara, V. A. Villagrá, J. I. Asensio, and J. Berrocal, "Ontologies: Giving semantics to network management models," *Network, IEEE*, vol. 17, no. 3, pp. 15–21, 2003.

[4] A. Gómez-Pérez and O. Corcho, "Ontology languages for the semantic web," *Intelligent Systems, IEEE*, vol. 17, no. 1, pp. 54–60, 2002.

[5] J. E. L. D. Vergara, V. A. Villagrá, and J. Berrocal, "Applying the web ontology language to management information definitions," *IEEE Communications Magazine*, vol. 42, pp. 68–74, 2004.

[6] A. Guerrero, V. Villagr, J. de Vergara, and J. Berrocal, "Ontology-based integration of management behaviour and information definitions using swrl and owl," in *Ambient Networks*, ser. Lecture Notes in Computer Science, J. Schnwlder and J. Serrat, Eds. Springer Berlin Heidelberg, 2005, vol. 3775, pp. 12–23.

[7] V. A. Villagrá and J. E. L. D. Vergara, "Ontology-ba sed policy refinement using swrl rules for management information definitions," in *in OWL. In: Proc. 17th IFIP/IEEE International Workshop on Distributed Systems, Operations and Management (DSOM*, 2006, pp. 227–232.

[8] A. Guerrero, V. Villagr, J. de Vergara, A. Snchez-Macin, and J. Berrocal, "Ontology-based policy refinement using swrl rules for management information definitions in owl," in *Large Scale Management of Distributed Systems*, ser. Lecture Notes in Computer Science, R. State, S. van der Meer, D. OSullivan, and T. Pfeifer, Eds. Springer Berlin Heidelberg, 2006, vol. 4269, pp. 227–232.

[9] J. E. L. de Vergara, V. A. Villagrá, C. Fadón, J. M. González, J. A. Lozano, and M. Álvarez-Campana, "An autonomic approach to offer services in osgi-based home gateways," *Computer Communications*, vol. 31, no. 13, pp. 3049–3058, 2008.

[10] J. E. L. De Vergara, A. Guerrero, V. A. Villagrá, and J. Berrocal, "Ontology-based network management: study cases and lessons learned," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 234–254, 2009.

[11] H. Xu and D. Xiao, "A common ontology-based intelligent configuration management model for ip network devices," in *Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on*, vol. 1. IEEE, 2006, pp. 385–388.

[12] D. Xiao and H. Xu, "An integration of ontology-based and policy-based network management for automation," in *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*. IEEE, 2006, pp. 27–27.

[13] N. Cuppens-Boulahia, F. Cuppens, F. Autrel, and H. Debar, "An ontology-based approach to react to network attacks," *International Journal of Information and Computer Security*, vol. 3, no. 3, pp. 280–305, 2009.

[14] F. Cuppens and A. Miège, "Modelling contexts in the or-bac model," in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*. IEEE, 2003, pp. 416–425.

[15] F. Cuppens, N. Cuppens-Boulahia, and A. Miège, "Inheritance hierarchies in the or-bac model and application in a network environment," *Proc. Foundations of Computer Security (FCS04)*, pp. 41–60, 2004.

[16] L. Fallon and D. O'Sullivan, "The aesop approach for semantic-based end-user service optimization," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 2, pp. 220–234, 2014.

[17] ——, "Using a semantic knowledge base for communication service quality management in home area networks," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 43–51.

[18] A. I. Rana, B. Jennings, M. Foghlu, and S. van der Meer, "Autonomic policy-based han traffic classification using augmented meta model for policy translation," in *Wireless and Optical Communications Networks (WOCN), 2011 Eighth International Conference on*. IEEE, 2011, pp. 1–8.

[19] A. I. Rana and B. Jennings, "Semantic uplift of monitoring data to select policies to manage home area networks," in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. IEEE, 2012, pp. 368–375.

[20] J. Keeney, O. Conlan, V. Holub, M. Wang, L. Chapel, M. Serrano, and S. Van Der Meer, "A semantic monitoring and management framework for end-to-end services," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 658–661.

[21] C. Hampson and O. Conlan, "Supporting personalized information exploration through subjective expert-created semantic attributes," in *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*. IEEE, 2009, pp. 384–389.

[22] B. Motik, B. C. Grau, I. Horrocks, Z. W. A. Fokoue, and C. Lutz, "Owl 2 web ontology language profiles (second edition)," http://www.w3.org/TR/owl2-profiles/, 2012.

[23] A. Textor, "A CIM-based Ontology for Semantic IT-Management," in *5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud*, Paris, France, October 2011.

[24] H.-U. Krieger, "A detailed comparison of seven approaches for the annotation of time-dependent factual knowledge in rdf and owl," in *Proceedings of the 10th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (held in conjunction with LREC 2014). ACL-ISO Workshop on Interoperable Semantic Annotation*. European Language Resources Association, 2014.

[25] C. Welty and R. Fikes, "A reusable ontology for fluents in owl," in *Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2006, pp. 226–236.

[26] F. Frasincar, V. Milea, and U. Kaymak, "towl: Integrating time in owl," in *Semantic Web Information Management*. Springer, 2010, pp. 225–246.

[27] V. Milea, F. Frasincar, and U. Kaymak, "towl: A temporal web ontology language," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 1, pp. 268–281, 2012.

[28] S. Batsakis and E. G. M. Petrakis, "Sowl: Spatio-temporal representation, reasoning and querying over the semantic web," in *Proceedings of the 6th International Conference on Semantic Systems*, ser. I-SEMANTICS '10. New York, NY, USA: ACM, 2010, pp. 15:1–15:9.

[29] B. Harbelot, H. Arenas, and C. Cruz, "Continuum: A spatiotemporal data model to represent and qualify filiation relationships," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming*, ser. IWGS '13. New York, NY, USA: ACM, 2013, pp. 76–85. [Online]. Available: http://doi.acm.org/10.1145/2534303.2534312

[30] C. de Sainte Marie, G. Hallmark, and A. Paschke, "Rif production rule dialect," http://www.w3.org/TR/2013/REC-rif-prd-20130205/, February 2013.

[31] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[32] IBM Corporation. (2006, June) An Architectural Blueprint for Autonomic Computing, Technical Whitepaper (Fourth Edition).

[33] O. M. Group, "Mof model to text transformation language (mofm2t)," http://www.omg.org/spec/MOFM2T/1.0/, January 2008.