Autonomic framework based on decision models for self-management of ubiquitous systems

Mahdi Ben Alaya, Thierry Monteil, Khalil Drira

▶ **To cite this version:**

Mahdi Ben Alaya, Thierry Monteil, Khalil Drira. Autonomic framework based on decision models for self-management of ubiquitous systems. Smart Gadgets Meet Ubiquitous and Social Robots on the Web (Ubirobots'12), Sep 2012, Pittsburg, United States. hal-01228325

HAL Id: hal-01228325
https://hal.science/hal-01228325v1

Submitted on 12 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Autonomic framework based on decision models for self-management of ubiquitous systems

Mahdi BEN ALAYA[1,2] , Thierry MONTEIL[1,2], Khalil DRIRA[1,2]
[1]CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2]Université de Toulouse; INSA, LAAS; F-31077 Toulouse, France
{ben.alaya, monteil, drira }@laas.fr

Abstract— **Internet of things consists of a high amount of heterogeneous objects that are widely distributed and evolve frequently according to their context changes. Management of such complex environment is costly in terms of time and money. Designing context aware autonomic framework with capability of self-management is a challenge. This paper proposes FRAMESELF, a generic and extensible autonomic framework based on ontology and graph models, and reasoning rules to self-manage ubiquitous environment. A smart metering use case is experimented to illustrate the proposed solution.**

Keywords- Internet of things, autonomic computing, self-management, context-awareness, expert system, model, ontology, graph, smart metering.

I. INTRODUCTION

The emergence of communicating objects, the decrease of the communication cost and the improvement of networks performance emphasizes the interest to build widely distributed ubiquitous environment. Such systems consist of a high amount of heterogeneous entities having mutual interactions and delivering high level services that could be discovered, composed, executed and monitored. Managing ubiquitous systems requires autonomic frameworks[1] with self-management functionalities, and multi-view knowledge representation and reasoning capabilities. These framework should be generic and capable of providing a logic and semantic description of the interacting objects with respect to their context characteristics and their behaviors or situations[2]. Management system should be able to dynamically detect these entities and capture their contexts in order to identify what are the changes that are happening in the environment, infer the current situation and react accordingly. To solve the challenges above, we propose FRAMESELF, a generic autonomic framework based on decision knowledge models.

II. FRAMESELF AUTONOMIC ARCHITECTURE PRINCIPLE

The FRAMESELF architecture is based on the IBM autonomic architecture reference [3]. It is organized into four autonomic modules which are Monitor, Analyzer, Planer and Executer. These modules share a same knowledge and constitute together the control loop to dynamically manage entities using sensors and effectors. In our approach, each autonomic module operates as an expert system[4] to emulates the decision-making ability of a human expert. It is designed to solve complex problems by reasoning about knowledge, like an expert, and not by following the procedure of a developer as is the case in conventional programming. Each module is divided into two parts, one fixed, independent of the system: the inference engine, and one variable: the knowledge base model.

In our vision, it is not enough to have a framework performing the autonomic control loop. It is necessary to provide extensible and generic framework. Genericity can be considered in two aspects: Firstly, the framework must be generic with available external managed resources, in that way the autonomic manager could be used in different complex environment and may respond to different self-* problems with low cost. Secondly, the framework must be internally generic formed with reusable modules exchanging standardized data structure using generic interfaces. In that way each module could be substituted by another one more efficient according to necessity and environment changes. FRAMESELF global architecture is described in Figure 1.

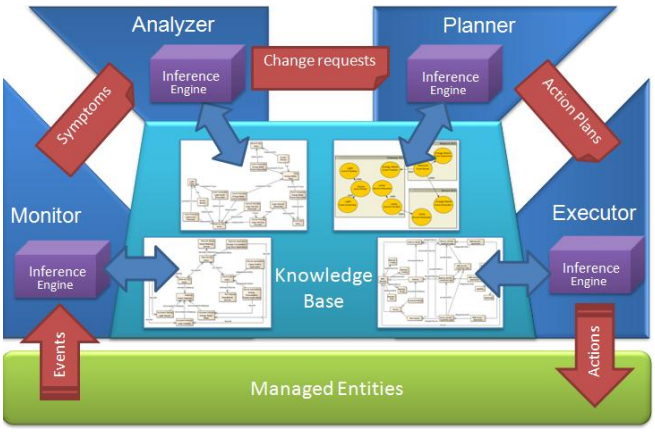

Figure 1. FRAMESELF architecture overview

The monitor answer to the "what is happening ?" question. It collects events from sensors of managed resources, applies on them function of filtering, masking, aggregation, and normalizes them according to the Web Event Format (WEF)[5]. It updates an ontology model instance describing the sensors environment and topology located on the knowledge base with relevant information of collected events, and applies on it semantic rules to infer new symptom occurrences. It extracts relevant information, normalizes them according to the "Symptom 2.0"[6] format, and send them to the analyzer.

The analyzer focus on the "what to do ?" question. It receives symptoms as input, uses them to update an instance model describing the complex situation. With logic, the inference engine is able to generate new information from the knowledge about required requests for change. It extracts relevant information, normalizes them, and send them to the Planner.

The planner focus on the "how to do ?" question. It saves received requests for change as goal states, situations which it is trying to reach, verifies the policies model to guide its work, reads model of possible actions and facts, and checks the initial state which is given as the set of conditions that are initially true. It constructs a planning problem representation based on STRIPS or PDDL language [7], then applies a planning algorithm to determine a sequence of actions that can be executed from the initial state and that leads to a goal state. The planner normalize action plan, and send them to the executer.

Since the executor receives as input only logical description of actions to be executed which are independent of the managed resources, it should answer to the "who to invoke ?" question. There is a need to consults a knowledge model describing effectors environment and topologies to match logical actions with their correspondents physical ones. The executer performs the plans using effectors of managed entities, and controls the sequence action execution with consideration for dynamic updates.

III. SMART METERING USE CASE

A smart metering use case is experimented on the ADREAM smart building[8] as a proof of concept of our solution. In this example we aimed to self-manage a room environment based on various sensors and effectors to optimize energy consumption . e.g. controlling lamps and window blinds levels according to luminosity, and presence sensors.

The monitor subscribes to event channels, and receives all published events from available sensors in a specific format. It filter useless events and normalize relevant one for example: luminosity, presence, lamp state, and blind state events in the WEF format. It updates the knowledge model and applies rules to infers symptoms for example: high/low luminosity, true/false presence, opened/closed lamp state, and opened/closed blind in the Symptoms 2.0 format. The analyzer updates its model and applies rules to infer new request for change such as: decrease or increase luminosity. The planner then checks knowledge base to get the room initial state and availables facts and actions, and constructs a planning problem according to STRIPS language. Then, it determines the best sequence of actions to achieve the goal for example open/close lamp, or open/close window blind. Finally, the executor checks knowledge model to determine the web service details corresponding to each actions for example: setLamp(String boolean) and setBlind(String boolean) methods, performs the plan. If the execution process fails, the executor is able to display an alert message with the occurred errors to the administrator, or to return back to the previous system state, or to plan once more new actions taking in consideration occurred errors. If the execution process succeeds, the autonomic manager starts the next control loop operation.

Our framework is independent of managed resources. In facts, it is possible for the monitor to read events details from log file instead of subscribing to event channel. It is also possible for the executer to deploy and run script file on specific devices instead of invoking web service to perform actions. The considered exchanging structures between autonomic module are normalized. This make the framework more extensible and able to substitute or to integrate new modules to adapt to eventual environment changes.

IV. CONCLUSION

In this paper we explained the basis of generic autonomic framework called FRAMESELF aiming to self-manage complex distributed system. We presented an overview of the architecture principle of the proposed solution, and explained the role of its main modules and how should they interact with each other and with knowledge base to achieve their tasks. We also showed our interest to provide a generic and extensible solution based on decision model and rules. Then we experimented our approach with a basic smart metering use case to illustrate the control loop steps.

As future work, we propose to experiment our solution in more complex scenario to self-manage a high number of heterogeneous objects to face scalability problems and calculate the overload that the framework generate. Actual solution proposes a centralized autonomic framework, we think that is essential to experiment a distributed architecture. e.g. distributed monitors, analyzers, planners and executers sharing one or more knowledge bases and interacting using pear to pear or hierarchical communication mode.

ACKNOWLEDGEMENT

This work is part of the European project ITEA2 A2NETS. It will be integrated on the smart metering business case demonstrator.

REFERENCES

[1] Kephart, J.O.; Chess, D.M.; , "The vision of autonomic computing," Computer , vol.36, no.1, pp. 41- 50, Jan 2003

[2] Stojanovic, L.; Schneider, J.; Maedche, A.; Libischer, S.; Studer, R.; Lumpp, Th.; Abecker, A.; Breiter, G.; Dinger, J.; , "The role of ontologies in autonomic computing systems," IBM Systems Journal , vol.43, no.3, pp.598-616, 2004

[3] IBM: An Architectural Blueprint for Autonomic Computing,. IBM White Paper 4th Ed., June 2006

[4] Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2

[5] Organization for the Advancement of Structured Information Standards, Web Services Distributed Management Technical Committee, Management Using Web Services (MUWS).

[6] IBM Research Center, 2006. Symptoms Reference Specification, Version 2.0. IBM Autonomic Computing Symptom Specification.

[7] Ghallab, Malik; Nau, Dana S.; Traverso, Paolo (2004), Automated Planning: Theory and Practice, Morgan Kaufmann, ISBN 1-55860-856-7.

[8] http://www.laas.fr/ADREA