A Framework to Create Multi-domains Autonomic Middleware

Mahdi Ben Alaya, Thierry Monteil, Khalil Drira, Tom Guérout

A Framework to Create Multi-domains Autonomic Middleware

Mahdi Ben Alaya and Thierry Monteil and Khalil Drira and Tom Guérout
CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse Cedex 4, France
Université de Toulouse; UPS, INSA, INP, ISAE; UT1, UTM, LAAS; F-31077 Toulouse Cedex 4, France
Email: ben.alaya@laas.fr, monteil@laas.fr, khalil@laas.fr, tguerout@laas.fr

***Abstract*—This paper proposes an enumeration and a classification of the services or functionality needed in the autonomic middleware. This allows to propose a second time the foundation for a framework that will be able to generate different middleware implementing autonomic loop and adapted to areas with different constraints and different needs. An illustration in the field of "Machine to Machine" and more particularly of smart metering is given.**

Keywords- autonomic computing; middleware; architecture; components.

I. INTRODUCTION

The increasing complexity for the management of current distributed software and system needs new solutions. The computer system "selfware" was created in the year 1995 for this purpose. By applying the properties of "self-*" to the computer systems, Kephart and Chess [2] and Brantz [1] define in 2003 the four paradigms to be implemented at least in such systems to become self-managed: self-configuring, self-optimizing, self-healing and self-protecting.

In the last year, we are witnessing a widespread use of autonomic loop in many areas: high performance computing, service management, M2M (Machine to Machine) system, network, etc. The expression of autonomic behavior in each area often results in the construction of middlware completely different. Yet, the basic principle remains the same even if the elementary actions constituting the various phases of the autonomic loop vary.

We propose in this work in progress paper to enumerate the main "components", "services", "features" needed to create a generic framework for building autonomous middleware specific to each area. We then describe a generic architecture between the proposed components. Finally, we give an example of future utilization of this Framework in the case of M2M.

II. RELATED WORK

There are many middlewares to implement autonomic principles. They can be intrusive in the managed system or not.

DeployWare [3], manages the deployment of autonomic distributed applications. This approach defines three roles in the management of the software. The "expert software" is the specialist in software technology to deploy. "The system administrator" gives the network configurations (description of the physical infrastructure deployment). "The end user" is using the application deployed. The peculiarity of DeployWare is that it proposes specific language for deployment (DSL Domain Specific Language) and a virtual machine for this language. DeployWare language is defined by a meta-model and provides a graphical notation in the form of a UML profile. Two concepts are important for us: the importance of defining roles and use of specific language nearest for users.

OceanStore [4] is used for the field of distributed and persistent storage of data. Its main goal is the implementation of the four properties of "self-management" applied to the high data availability. Its features are for "self-healing" fault tolerance through data redundancy and automatic repair. Here the focus is on the ability of middleware to provide services. This requires an application of the autonomic loop in the middleware itself.

Oceano [5] is applied to the field of cluster management for computing intensive applications or applications with a processor load varies with time (web servers). The peculiarity of this approach is the way it manages the property of "self-Optimizing" policies dictated by contracts SLAs (Service Level Agreement) to specify a level of service by type of cluster or client (using one or more clusters). The use of SLA seems an important step by the possibility of applying it to many areas.

Gryphon [6] brings to the monitoring the notion of prioritization of events, as well as processing and agglomeration of events. This is an event management system self-adaptive, in fact, this approach also uses the events described as meta-events that trigger a reconfiguration of its internal functioning. Intelligent processing of events is a prerequisite for scalability. This will also be addressed in the framework that we propose.

Astrolabe [7] is an approaches providing an API to develop applications with properties of "self-management". It is used to collect the states of a very large scale (several thousand to several million nodes) according to zones. The area is also cutting into a solution that we wish to implement through the use of the concept of group and adapted communication patterns.

TUNe [8] is based on a component model. Its particularity is to add autonomous behavior to different types of existing legacy software. It provides a uniform vision of controlled

softwares using the method of encapsulation with components. The administration then uses the standardized interface provided by the component model and a set of generic sensors or probes reusable skeletons. we will implement a model of components and services based on SCA (Service Component Architecture)[10].

None of those middlewares can address different domains. Each one has some specific characteristics. The goal of our framework is to build different autonomic middlewares with specific properties covering the needs of the domain of utilization.

III. Framework principle

A. Functionalities

In this part, we presented the functionalities that should be provided by our framework. We are inspired by the list of the M2M (Machine to Machine) functionalities detailed by ETSI (European Telecommunications Standards Institute) [9] to specify our classification. We decided to structure our framework features into six classes which are: communication, security, data toolkit, autonomic, management and entity classes. **Complex/structured Communication class** involves machine-to-machine, machine-to-man, and man-to-machine communications based on multiple communication means, e.g. SMS, GPRS and IP Access:

- Event processing: integrate different kind of event processing style: simple, stream and complex flow.
- Service oriented interactions: support service invocation between requester and provider.
- Transmission scheduling: Manage the scheduling of network access and of messaging.
- Delivery modes: support any-cast, uni-cast, multi-cast and broadcast communication.
- Flow management: handle asymmetric flows and support flow priority.
- Multi path: support physical paths diversity.
- Addressing: abstraction of the underlying network structure including any network addressing mechanism.

Security class involves structures and processes needed to protect the system and the connected users and devices against danger, damage, loss, and crime:

- Authentication: support two-way authentication and strength level selection.
- Encryption: support appropriate confidentiality of the data exchange.
- Anonymous: Possibility to hide the identity and the location of the requestor.
- Data integrity: support verification of the integrity of the data exchanged.
- Privacy: System shall be capable of protecting privacy.
- Security credential and software upgrade: secure updates of application security software and context (keys and algorithm).

Data toolkit class contains modules used for collection, representation and reporting of data:

- Data Base: gives a tool to store all necessary data
- Data collection: it includes pre collection activities (target data, definitions, method, etc.), collection and present findings.
- Reporting: Supports many type of reporting: periodic, on-demand, scheduled and event-based reporting.
- Graph modeling: provide mechanism to represent data in advanced structures like tree or graph to have a mapping describing in details the physical system.

Autonomic class contains modules making a system able to manage itself [2] (self-configuring, self-healing, self-optimization and self-protecting) and dynamically adapt to change in accordance with business policies:

- Monitoring: Provides the mechanisms that collect, aggregate, filter and report details collected from managed entities.
- Analyzing: provides the mechanisms that correlate and model complex situations. Help to learn about the environment and predict future situations.
- Planning: provides mechanisms that construct the actions needed to achieve objectives using policy information to guide its work.
- Executing: provides the mechanisms that control the execution of a plan with considerations for dynamic updates.
- Policy: supports different type of behavior for the planning component.

Management class contains modules that allow remotely configuring and controlling connected devices.

- Configuration: supports maintaining consistency of a system performance and its functional and physical attributes with its requirements, design, and operational information.
- Deployment: manages components life-cycle and activities of release, install, uninstall, activate, deactivate, update, adapt, built-in and version tracking.
- Remote administration: Supports advanced control request and receive acknowledgments to administrate the middleware
- HMI (Human-Machine Interface) system: helps to manage the system graphically .

Entity class it handles resource that exist in the run-time environment of an IT system and that can be managed:

- Group: support a mechanism to create and remove groups, to introduce an entity into a group, modify the invariants of the members, remove an entity, list members, search entities in a group, identify entity groups where the entity is a member, etc.
- Session: start and stop session supporting cooperation between two or more communicating entities [11].

- Profile: support computer representation of a user and device model [11].
- Role: possibility to assign role to a connected entity to manage their behaviors, rights and obligations.
- Discovery: a connected entity to a network should be able to advertise itself and to discover other entities.
- Description: each entity should be able to describe itself and to detail its hosted services in a standard format.
- Registration: allowing an entity to subscribe to asynchronous event messages produced by a given service.
- Meta-data exchange: provide dynamic access to a device's hosted services and to their meta-data.

B. Architecture

The different classes and services defined above are included in an architecture based on the SCA standard. This will have great flexibility in the use and construction of a middleware. Indeed, the interactions between components can be defined by different means: rmi, web-service, java, etc. Similarly, it is very easy to replace the instantiation of a component by another, or to take only part of the available components.

It is planned to establish the composition of middleware from eclipse by drawing in architecture and components available and so generate the autonomic middleware corresponding to his need. In Figure 1, a UML components diagram shows a part of the relation between components of the framework.

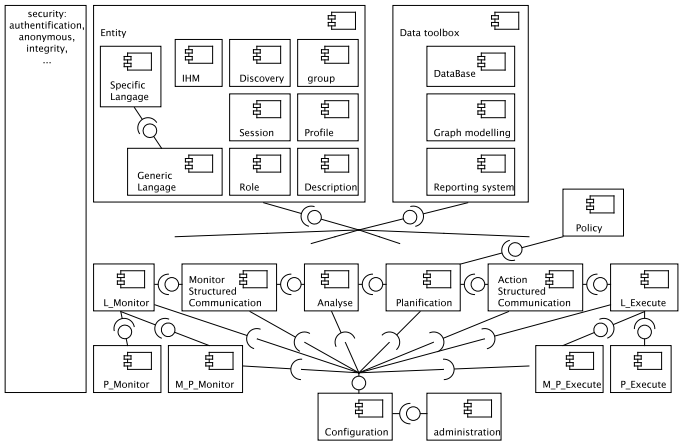

Figure 1. Framework architecture

We find the different elements of the autonomic loop with the monitoring of the system to manage (*P_monitor*) but also the monitoring of the autonomic middleware (*M_P_monitor*). The observed data are transformed into a generic vision through the *L_monitor* to be transmitted via a component of effective communication (*Structured Communication Monitor*, one or more component of the class communication) to be used by the component analysis (*Analysis*). The latter built a diagnosis that will generate a reaction (*Planning*) based on various policies (*Policy*). The set of elementary actions are effectively transmitted via the communication component (*Action Structured Communication*) to component (*L_Execute*) responsible for transforming the logical actions in specific actions to be executed by the actuators of the managed system (*P_Execute*) or middleware (*M_P_Execute*).

Depending on the area treated, there is a set of component toolbox class entity that can be used to provide specific services needed. There should be also the possibility in the toolbox to define specific languages readily available in various trades. There is also the use by the major components part of a toolbox to manage data from different patterns. Safety aspects are transversal to all this by using the notion of politics in SCA. All these components are configurable via the component *configuration* that orchestrates the system. Components of administration (*Administration*) and visualization (*HMI*) also allow to control the use of middleware in its execution.

IV. Example in M2M domain

The smart metering is a domain of M2M where autonomic loop could be used. Information such as energy consumption, temperature, light etc are collected with sensors. They are networked into a communication network that allows the sensed information to be fed to a central system where data can be analyzed then a list of actions can be planned. Actuators and appliances can next be automatically configured such as remotely reducing the level of the lamps or turning off the heating. ETSI specified six functionalities [12] related to smart metering expressed in broad terms, so that they can be related to electricity, gas, heating/cooling and water. Identifying functionalities at high level will permit flexibility, innovation and competition:

- Remote reading of metro-logical registers and provision to designated market organization.
- Provide two-way communication between the metering system and designated market organization.
- Support advanced tariffing and payment systems.
- Remote activation and deactivation of supply.
- Communicating with (and where appropriate directly controlling) individual devices within the building
- Providing information via gateway to an in-home display or auxiliary equipment.

The Figure 2 describes our smart metering architecture which involves the smart building ADREAM [13] that will serve as a real experimental platform to test our solution capabilities.

MAPE-K Loop modules will be used to self-manage smart metering operations: (*P_Monitor*) collects data from smart meters (electric, gaz, water and photovoltaic meters) and also from sensors (temperature, light, presence, etc). After analysing and planning, (*P_Execute*) executes required

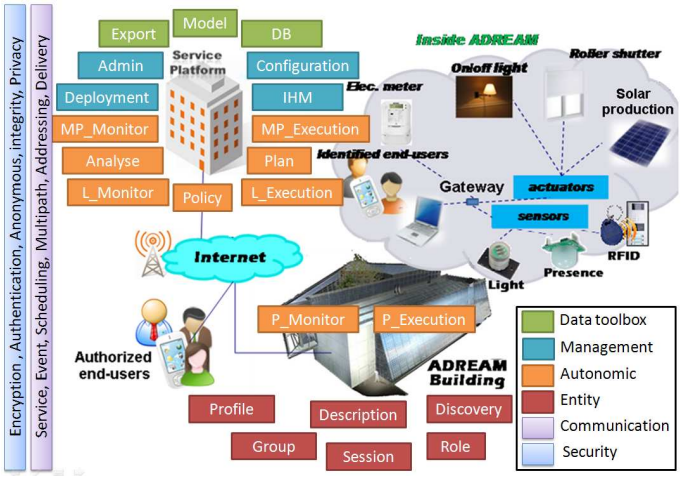

Figure 2. Smart metering architecture

actions to control different kind of actuators (roller shutter, on/off light, heating level, etc.).

(*M_P_Monitor*) supervises the middleware components and devices. It collects information about the middleware distributed machines context (Server memory, CPU, Rooting, etc.). If a problem is detected (Server down, big number of users, etc.) so, after analysing and planning, (*M_P_Execution*) executes actions such as deploying additional servers in new distributed machines or re-configuring gateways parameters to optimise the communication flows to add more scalability to the middleware.

For example, consider a scenario where a Customer decides to add a new sensor to regulate his consumption as a function of luminosity, so he looks his consumption and changes the way he wants that energy is consumed in his house. After being authenticated, the system characterizes his rights (*role*). It connects the new sensor that will be inserted dynamically into the system after authentication (*discovery*, *Registry*, *description* and inclusion in the *database*). It then displays (*HMI*) the consumer consumption and decides to change the behavior of the system of energy regulation (*profile*) because the system has automatically update the new possibilities offered in terms of regulation of energy thanks to this new sensor. The new autonomous policy (*policy*) is connected to the planning module (*Planning*).

V. Conclusion

In this paper, we present the basis for a framework that aims to create autonomic middleware specific to different application areas. The goal is to build a single tool that will be enriched and developed with many new components. The use of SCA should facilitate this. This framework will be used in various research projects and industry. A first prototype showing the feasibility of concepts is underway with the first application as the area of M2M. Future work will develop the framework and use it as part of large scale distributed computing. We can even think about interpretability scenarios between multiple autonomic middleware allowing to link several M2M domains.

References

[1] D. F Bantz, C. Bisdikian, D. Challener, J. P Karidis, S. Mastrianni, A. Mohindra, D. G Shea, and M. Vanover, *Autonomic personal computing*, IBM Systems Journal, pp. 165-176, 2003

[2] J. O. Kephart, and D. M. Chess, *The vision of autonomic computing*, Computer, pp. 41-50, 2003

[3] Areski Flissi, J Dubus, Nicolas Dolet, and Philippe Merle, *Deploying on the Grid with DeployWare*, 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID), Lyon France, pp. 177-184, 2008

[4] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and others, *Oceanstore: An architecture for global-scale persistent storage*, ACM SIGARCH Computer Architecture News, V. 28 N. 5, pp. 190-201, 2000

[5] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing and B. Rochwerger, *Oceano-SLA based management of a computing utility*, Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management, Seattle USA, 2001

[6] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward, *Gryphon: An information flow based approach to message brokering*, IBM TJ Watson Research Center Reports, 1998

[7] R. Van Renesse, K. P Birman, and W. Vogels, *Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining*, ACM Transactions on Computer Systems (TOCS), V. 21 N. 2, pp. 164-206, 2003

[8] Remi Sharrock, Thierry Monteil, Patricia Stolf, Daniel Hagimont, and Laurent Broto, *Non-intrusive autonomic approach with self-management policies applied to legacy infrastructures for performance improvements*, International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS), V. 2 N. 2, pp. 1-20, 2010

[9] ETSI TS 102 689 Machine-to-Machine communications (M2M); M2M service requirements.

[10] Simon Laws, Mark Combellack, Raymond Feng, Haleh Mahbod, and Simon Nash, *Tuscany SCA in Action*, February, 2011 — 472 pages ISBN 9781933988894, manning

[11] M.Ben Alaya, V.Baudin, and K.Drira, *Dynamic deployment of collaborative components in service-oriented architectures* 11th International Conference of New Technologies in Distributed Systems (IEEE NOTERE2011), Paris, France, 2011.

[12] ETSI TR 102 691 Machine-to-Machine communications (M2M); Smart Metering Use Cases.

[13] http://www.laas.fr/ADREAM/