Autonomic Management Approach for Dynamic Service Based IoT Systems

Guillaume Garzone, Nawal Guermouche, Thierry Monteil

Autonomic Management Approach for Dynamic Service Based IoT Systems

Guillaume Garzone
LAAS-CNRS, Université de Toulouse, INSA
Toulouse, France
Email: guillaume.garzone@laas.fr

Nawal Guermouche
LAAS-CNRS, Université de Toulouse, INSA
Toulouse, France
Email: nawal.guermouche@laas.fr

Thierry Monteil
LAAS-CNRS, Université de Toulouse, INSA
Toulouse, France
Email: thierry.monteil@laas.fr

***Abstract*—The Internet of Things (IoT) continues to expand undeniably fast to reach billions of connected heterogeneous devices. This is changing the way systems are built: new applications integrating software and physical devices are emerging in different domains, such as health, smart building, and smart cities. This brings opportunities to enable new added value services. Nevertheless, building and managing such highly dynamic and heterogeneous infrastructures built upon a multitude of mobile and resource-limited devices is challenging. In this paper, we propose a semantic based autonomic management approach for service oriented IoT systems. The aim is to support building and managing highly dynamic new value added IoT services. The proposed approach relies on a semantic based model to characterize the system properties and then enables semantic reasoning, and graph grammars to enable its management and evolution. A use case is proposed to show the related features of the proposed approach and an evaluation study is presented.**

I. Introduction

During the past few years, the IoT expanded in an impressive way and is still growing in terms of number of connected devices. More than 20 billions of connected things are expected in the next years[1]. This growth goes along with the development of new solutions and the enhancement of smarter and more complex systems.

IoT presents a suitable opportunity to enable the development of innovative applications as it is accepted and adopted in several domains such as smart cities, health domain, industry and transportation. Indeed, the combination of IoT with existing technologies and paradigms such as Cloud Computing, Service Oriented Computing, and powerful software analytic capabilities, is changing the way systems are developed.

In parallel to advances, IoT brings complex challenges which require investigations. In this paper, we are particularly interested in the problem of managing on the fly dynamic IoT based systems. Usually, such systems rely on the integration of heterogeneous connected *things*, such as sensors (e.g., temperature, presence, pollution, etc.), actuators (e.g., remotely controllable devices that acts on the environment), and other entities such as software services which add value to data provided by the connected objects (e.g., meteorological information, open services and open data, remote control of entities) and enrich the possibilities offered by these systems.

[1]http://www.gartner.com/newsroom/id/3598917

The integration of those complex and high level services along with elementary services provided by the connected devices to build complex smarter systems is still challenging. The cooperation and interoperability between things and services become a key point to increase and to improve the individual value they provide [1], [2]. Many application domains may be impacted by the development of those smart systems [3]: in the context of smart city, it enables the automatic management of the system with several considerations, such as energy consumption optimization or diverse metrics monitoring.

In this paper, we focus on the problem of autonomic management of high dynamic IoT based systems. Our goal consists in autonomously managing such complex systems to cope with their dynamic nature. Indeed, IoT based systems (e.g., in a smart city scenario) evolve in dynamic environments where the devices can be mobile, appear and disappear, the requirements can evolve, etc. Thus, defining autonomic system able to self-manage while tackling evolutions and changes is necessary. This problem has been studied in the field of service oriented information systems [4] [5] [6]. However, these works assume that exhaustive description of the whole behaviour of the system is given by users. Then, the goal is to instantiate the given specification dynamically. In this paper, we do not assume that the specification of the targeted system is given. The proposed work relies on the description of requirements based on awaited data and the properties of the available objects to build and manage dynamically the system. In this context, we propose a semantic and graph based autonomic management approach for service oriented IoT based systems.

The paper is structured as follows: a motivating example is presented in Section II and the global structure of the proposed work is presented in Section III. The autonomic management framework models are presented in details in Section IV and V. Then, it is followed by the evaluation study in Section VI. Before concluding, background and related works are studied and discussed in Section VII.

II. Motivating example

In this paper, a smart city scenario is considered where different entities exchange contents. A content is equivalent of any information or data, that is made available (i.e., accessible through desktop or mobile applications, software

services, cloud platforms, etc.). This content can have different provenances and topics, for instance city related data which gives information about the city itself (e.g., general information, alerts), sensors deployed in the city (e.g., air quality monitoring, parking slots availability, etc.), data related to third party entities such as shops, cultural organisms, events in the city or any other interesting events.

In the smart city two kind of entities interacting with this content are to be distinguished:

- **content producers**: entities that create and provide content to broadcast in the city or sent to the interested users (e.g., connected sensors, etc.)
- **content consumer**: entities that are interested in receiving specific information about topic(s) they are interested in, but also alerts, content updates, etc. For instance, a smart phone of a citizen registered as a user in the smart city is a content consumer.

So, different kinds of contents are produced and provided by *content producers* and *content consumers* express their interest for a certain kind of content more customized or not so they could receive updates about this kind of content. Contents can be broadcast in the city (in public transportation, display or advertising boards, etc.). Display devices may be available with several broadcast modes (audio, text, image, video...) and will diffuse collective content on public devices, and more customized content on personal devices (a smart-phone for instance).

An interesting kind of entity interacting within the smart city are the connected vehicles and particularly the connected buses. These connected buses are a key element to bring content to the citizens. Citizens interact with the system through their own devices (e.g. smart-phones) and mobile applications where they can express interests and communicate their position. In addition, it is assumed that the buses are equipped with connected display devices and are accessible through services available in the Cloud. These devices will display dynamic content regarding to the location of the bus in the city, where it is going ahead and the services available. Moreover, more general information may be displayed too, as well as urgency or priority content.

Users can express different kind of interests that can change, a multitude of contents can be provided dynamically by different providers (connected devices, software components deployed on the Cloud, etc.), IoT devices (e.g., sensors) can be mobile around the city and can be connected, disconnected and reconfigured. In this context, being able to build autonomic systems to self-adapt to changes and evolutions while guaranteeing the satisfaction of citizens needs is of paramount importance.

The problem

To summarize, the problem we focus on can be described as follows: given a dynamic system which is built upon a multitude of entities abstracted as services and characterized by their inputs and outputs, evolving targets that aim to provide data to services, our goal is to enable autonomic management of this kind of systems to cope with changes and evolutions so that the specified targets are fulfilled throughout the execution.

III. High level architecture

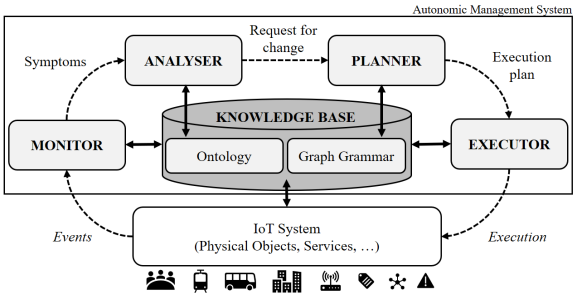

Fig. 1. Global structure of the system

To deal with the stated problem, we propose the management framework depicted in Figure 1. It consists of three main elements: the **autonomic manager**, the **Knowledge base (KB)**, and the underlying **IoT system**.

The KB enables to store and share knowledge about the monitored system, its behaviour which can be used, for instance, to adapt the system regarding what happens. The autonomic manager is the high-level entity in charge of managing the underlying part. It relies on the MAPE-K model introduced in [7]. This paradigm structures an autonomous system in four main components: the **Monitor** monitoring an environment, the **Analyser** making decisions to act on the system with a **Planner** planning actions to be taken, and an **Executor** executing the actions. The monitor generates *symptoms* based on the relevant events coming from the monitored system. The analyser considers the different symptoms sent by the monitor component and generates *requests for change* (RFC) based on the symptoms and given policies. The planner receives an RFC from the analyser and generates *execution plans* to be executed on the system by the executor.

In this work, the manager interacts with both the KB and IoT system. The IoT system regroups the connected devices deployed in the city, the connected vehicles, and other connected entities, but also the services associated to the objects or software services (e.g., deployed on the Cloud). The monitor maintains the KB up to date with the current state of the system thanks to the connection with specific sensors and metrics, and the executor receives plans, generated by the planner, to execute on the system: basically a set of services to invoke according to branching structures. In this paper, we particularly focus on the *analysis* and *planning* phases where the decision making takes place and the actions to perform are generated. The **monitor** and **executor** components are not studied in this paper however existing advanced monitoring techniques can be used [8] [9] [10].

The KB is a key component of the system. In our approach, it is compound of two models: an **ontology** [11] and a **graph grammar** (as introduced in [12]) based models. It is important to notice that those models can evolve or change, regarding to the requirements and context or the expected behaviour of the

managed system. The ontology is used to store semantic meta-data about the IoT system and to enable semantic reasoning which helps the analyser to make decisions (Section IV). Regarding the planner, it relies on a graph grammar based model to generate the execution plan (Section V-B) according to an RFC generated by the analyser.

In the following sections, we present the different models used by the proposed framework and their instantiation.

IV. The IoT System Ontology Based Model

The ontology based model used in the framework is presented in Figure 2. The core of this ontology has been developed for the semantic characterization of the IoT system of our use case. It extends IOT-O ontology[2] [11] concepts to define objects (devices) and services. Indeed, the IOT-O ontology gathers several recognised ontologies and enrichment regarding to relevant meta data about services, such as operations and parameters for REST services and IoT devices (e.g., HRESTS[3] ontology) or properties for actionable devices (e.g., SAN[4]). For more details on the IOT-O ontology and the gathered ontologies, we refer reader to [11].

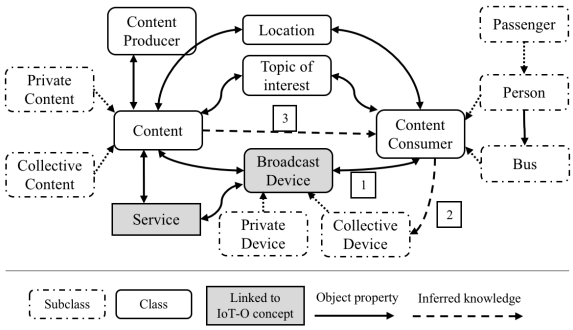

Fig. 2. The ontology based model of the content spreading to citizens

A. Entities at stake in the model

In the following, the proposed extensions of IOT-O are presented.

Content: This concept can represent a raw data or an information in several formats such as html document, video teaser, audio file, temperature, etc. provided by content producers entities.

Topics of interest This concept represents any kind of topic that categorizes the content and may be interesting for entities connected to the system. For instance, a topic of interest can be "*air quality*", so the interested third parties can express an interest in "*air quality*", related content i.e. any information, update of data, and so on.

Content consumer (*resp. producer*) represents any entity in the system that is interested in getting access to content (*resp. provides a content*). In our use case, content consumers (*resp. providers*) consist in persons and connected vehicles (e.g. buses) (*resp., sensors, city communication pole, etc.*).

[2]https://www.irit.fr/recherches/MELODI/ontologies/IoT-O.html
[3]http://www.wsmo.org/ns/hrests/
[4]https://www.irit.fr/recherches/MELODI/ontologies/SAN

Location The location concept is a representation to distinguish different places in the city or even the whole city itself.

The broadcast devices (also called display devices) They represent the physical objects that broadcast content to the considered entities. For instance, if a person possesses a smart-phone it is considered as a broadcast device as it is accessible to send content to it (e.g., through a mobile application).

B. Relations

The object properties defined in the ontology (relations between concepts) are represented in Figure 2. Regarding broadcast devices and entities, the object property number 1 in Figure 2 represents entities that own a broadcast device and property 2 represents entities that have access to broadcast devices. For instance, citizens can own a smart-phone with a mobile application (considered as a private broadcast device) and if they are passenger of a bus with embedded displays, they have access to the broadcast device(s) of the bus too. Moreover, besides the locations and topics of interest, a key object property is the object property number 3. This property characterizes the fact that a content should be sent to a set of entities based on their location and interests. This knowledge is a key element in the decision making for the analyser, and is presented in more depth in Section IV-C.

Example 1:

In our use case, the monitor instantiates the ontology and keeps it up to date with the current state of the system. A simple example of instantiation of the ontology is presented in Figure 3.

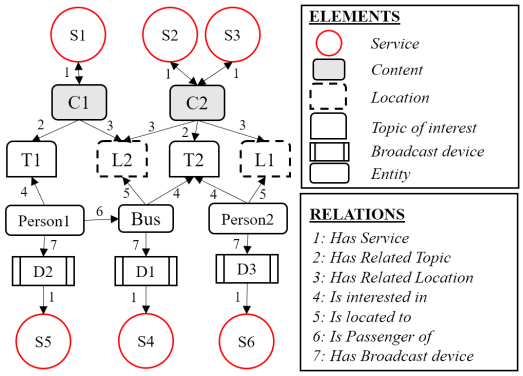

Fig. 3. A simple instantiation of the ontology

In this example two contents are identified (C_1 and C_2) and there are one bus and two persons considered in the system. The person 1 is passenger of the bus and the person 2 is not a passenger yet. Two different topics are instantiated (T_1 and T_2) with two different locations (L_1, L_2). The first content is related to the second location and the second content is related to both locations. The associated services are presented on top and bottom of the figure. For instance, the second content has two possible services to retrieve the content (e.g., two different protocols and data types can be used). The relations are described on the right side of the figure.

C. SWRL Rules and Analysis

The analyser determines which entities are interested in the targeted contents thanks to the semantic KB. This feature is characterized by Semantic Web Rule Language (SWRL) [13] rules embedded in the semantic KB. The set of rules enable the system to infer some knowledge using a reasoner, e.g., the different contents that should be sent to a set of entities. However, it does not include how to do so. The role of the analyser is to extract this knowledge from the KB and to generate an RFC (Request For Change) for the planner to plan how to send each content to the concerned entities, based on which devices they have access to.

Several rules are exploited in the inference of knowledge. An example of an SWRL rule is shown the Listing 1: it determines if either a content should be sent to an entity (a bus for instance) or not, based on the current knowledge. Other rules enable the system to identify if persons are passenger of a bus and so infer they have access to its broadcasting device and inherit its position.

```
ContentConsumer(?e) ^
isInterestedIn(?e, ?topic) ^
CollectiveContent(?c) ^
hasRelatedTopic(?c, ?topic) ^
hasRelatedLocation(?c, ?loc) ^
isLocatedTo(?e, ?loc) -> shouldBeSentTo(?c, ?e)
```

Listing 1. Example of SWRL rule of the ontology

Other similar rules are at stake to consider different possibilities, including the differentiation between private contents and collective ones.

Using the proposed SWRL rules and the ontology, when new contents are available (symptom) the analyser may trigger the reasoner to infer knowledge based on what is currently in the semantic KB. Then, the analyser generates an RFC based on the inferred knowledge in the ontology: the inferred object properties indicating to which entity the content should be sent will represent the link between the data producing services and the data consuming services the manager has to consider. An example of inference based on the ontology instance of Example 1 and then RFC generation is depicted in Figure 4.

Now the RFC, the execution plan and the graph transformation models will be detailed.

V. The graph oriented model

In this section the graph based models used by the framework are detailed: the RFC, the transformation and execution plan models.

A. Request for change to perform (RFC)

When an action needs to be performed on the managed system, a requirement for the action to perform is generated by the analyser. It is represented by an RFC graph and an associated graph grammar that enables the system to generate the execution plan accordingly.

The RFC is a graph containing the relevant information regarding the different services to use and some other elements linked to the services. In our use case, it contains content to broadcast and the interested parties extracted from the KB. This RFC graph is transmitted to the planner to perform graph transformations in order to generate the execution plan linking the different producing and consuming services. An important point to highlight here is the generic aspect and modularity of the analyser: new features can be easily integrated in the system through new sets of symptoms associated with RFC and a graph grammar to generate execution plans.

The RFC is the initial graph necessary for the planner to enable transformations and thus provide a plan to be executed.

Definition 1: (**Requirement model: RFC**) A requirement is defined by two elements: an RFC graph and an associated graph grammar.

Let Γ be a set of types of nodes. An RFC graph is a tuple (N,E,τ) in which :

- N is the set of nodes that characterizes the set of services and entities at stake;
- $E \subseteq N * N$ is the set of edges that characterizes the data flows
- $\tau : N \rightarrow \Gamma$ is a function that maps nodes to their types

The RFC can be instantiated as follows with the following types of nodes:

- **services**, noted S: represent the different services at stake (data producing or data consuming services)
- **contents**, C: represent relevant data (provided by services) that can be consumed by a set of entities.
- **entities**, E: represent persons, buses, or whatever entity in the system interested in getting access to data produced by services.
- **broadcast devices**, D : represent any kind of devices that enable the system to broadcast data to entities. It can consist in a smart-phone through an application, or connected screens in buses. Those devices are associated with a set of services that consume any data produced by other services and will perform the display action. Also, the entities have access to one or more broadcast devices and this knowledge is represented by an edge between an entity and a device node.

To summarize, the analyser identifies which elements to include in the RFC based the received symptoms and the current knowledge in the KB. In our use case, it considers which contents should be sent to which entities, and the necessary elements to enable by the planner to generate the execution plan. To do so, it gathers informations about entities: which devices they have access to, and what are the producing and consuming services associated to the content and the broadcast devices.

Example 2: Keep going with our previous example. As shown in Figure 4, the analyser triggers the reasoner of the semantic KB (instantiated in Figure 4.(a)) and observes changes. This behaviour is based on received symptoms in consequence of the availability of new contents (here C_1 and C_2). Some knowledge is inferred as shown in Figure 4.(b): the targeted new contents are interesting for a set of entities

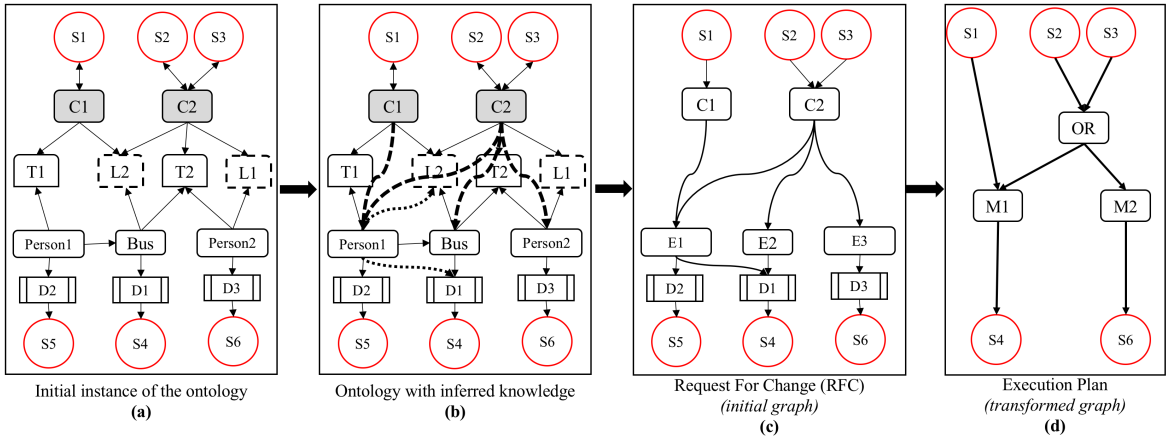

Fig. 4. An instance of the system and the different steps of the manager

and should be sent to them. The inferred knowledge is shown in discontinuous lines (content to send to entities) and dot lines (inferred properties for passengers). Then, the analyser generates the RFC accordingly (as shown in Figure 4.(c)).

In the next section, the *execution plan model* and the *graph grammar* model are defined and an instance is provided in the context of the considered use case.

B. Graph transformation and execution plan

As mentioned previously, the planner receives the RFC from the analyser component and the goal is to generate an execution plan. This implies transformation actions to be performed on the RFC graph in order to generate the plan graph. Indeed, the execution plan directly produces a workflow of services to call (e.g., retrieving data from a set of services, performing actions or transformations before sending contents to the identified entities by adding additional services to fit awaited formats, merging contents, etc.).

Such transformations are formalized using graph grammar based formalism. A set of graph-rewriting rules (transformation rules) is defined. The result of the transformation is a *plan graph model* that characterizes the set of concrete linked producing and consuming services according to the interests of the considered entities and the broadcast devices they have access to (associated to content consuming services). The produced contents can either be merged via the merge gateway or chosen via a choice gateway between two producers that provide the same content.

Definition 2: (**Execution plan graph model**)
The execution plan model is similar to the RFC model. Instantiated in our use case, the following types of nodes are distinguished:

- **services** S: the set of services node as defined in RFC model.
- **gateway nodes** G: gateways are used to define the structure branching to use when handling service's outputs. It can consist in merge nodes (noted M), that indicate some *outputs* or *inputs* of services need to be merged before the next step, or gateway OR nodes that indicate that several choices of services are available for the same action.

Several ways of specifying graph rewriting rules have been proposed [12]. In this work, the *Single Push-Out* (SPO) [14] approach is used.

Definition 3: (**Graph grammar and rewriting rules**)
A graph rewriting rule $r = L_r \xrightarrow{m_r} R_r$ is characterised by a couple of graphs ($L = (N, E, \tau)$, $R = (N', E', \tau')$) and a morphism $m_r = (f_r, \emptyset)$ of a sub-graph $L^I = (N_L^I = Dom(f_r)E_L^I)$ from L to R.

The *left graph* (L_r) of the rule represents the pattern that is matched in the graph to transform and the *right part* (R_r) represents the transformed pattern. When the rule r is applied to a graph G, if an image of L_r exists in G, the rule is applied and replaces the matched L_r graph by the associated R_r. Sometimes a *Not Applicable Condition* (NAC) is used to avoid appliance of the rule in case the indicated pattern is found.

For lack of space, the instance of the whole graph grammar used in our use case can not be shown in this paper. However the core rules of the grammar are presented in Figure 5. The L and R graphs of each rule are represented and the f_r function associated to the morphism m_r of the rule is materialized by the arrow. This provided set of rules enables the planner to merge services before sending their contents to a broadcast device based on who should receives this content and which devices these entities have access to. The rules are separated in three layers: the first one is dedicated to introduce the merge nodes (M) in the graph to merge the contents, the second layer works on the results of the first layer to make the connection between the created M nodes and the services, and the third layer is dedicated to clean up the graph to remove the remaining isolated nodes $\{E, D, C\}$ (i.e., persons, broadcast devices, and contents nodes) and keep only the nodes $\{S, M, OR\}$ (i.e., services, Merge, Or nodes) to obtain the execution plan.

These rules enable the creation of intermediary nodes in the graph (merge nodes, M) in order to aggregate the output of several services before sending it to the consuming services.

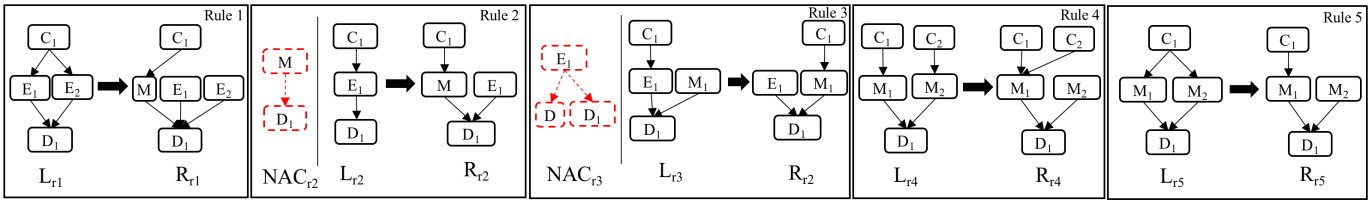

Fig. 5. Set of transformation rules from the Graph Grammar instantiated in our use case

In our use case, the idea is to regroup the contents interesting several entities who have access to a common display device. These contents, retrieved from the content producing services, will be merged during the execution phase and used as input for the consuming service associated to the broadcast device.

The rule 1 and 2 aim to create M nodes. The first rule creates M nodes based on the fact that two entities should receive the same content and share a common display device (for instance they are passenger of the same bus). The strategy of the rule 2 is similar but creates M nodes for each remaining content-entity link. The rules 3, 4 and 5, use existing M nodes to connect contents to devices, or aggregates M nodes to avoid multiple M nodes linked to the same device (rule 4 and 5).

Now one example of the behaviour of the whole grammar on a given RFC is introduced.

Example 3: Let us consider the example presented in Figure 4. Figure 4.(b) shows a simple instance of a RFC graph generated by the analyser. Figure 4.(c) presents the plan graph model resulting from the RFC graph transformation according to the defined rewriting rules. The obtained plan graph states that contents obtained thanks to service 1 and either service 2 or service 3 are merged before being sent to service 4, and the content obtained with service 2 or service 3 is sent to service 6.

In the next section the experimental set-up of the proposed approach is presented.

VI. Evaluation study

To validate the approach presented in this paper, two evaluations were conducted: a prototype of the proposed system demonstrating its functionalities, and a performance evaluation with respect to the response time of the system according to different parameters.

A. Proof of concept prototype

A prototype of the autonomic manager has been implemented and deployed to demonstrate the functionalities of the proposed approach and include it in a mock-up. The idea is to represent a connected bus in a smart city with different possible passengers and areas in the city with different kind of data available regarding the area. The prototype is compound of several elements: a computer with the Eclipse OM2M middleware[5] to connect the devices and expose their data thanks to the standard oneM2M[6] REST Application Programming Interface (API), an Intel Edison, to connect several sensors to the IoT middleware (OM2M) and a RaspberryPi to simulate the connected bus with a mini display broadcasting content dynamically. Some sensors are used for monitoring only as an RFID sensors to detect which passenger in the bus or the changes of location of the bus based on GPS position (the bus moves among the areas).

[5]http://www.eclipe.org/om2m

[6]http://www.onem2m.org

TABLE I
Setting of the demonstration system

Area	1	2	3	4
Name	Init	Industrial	City Centre	Museum
Content	Temperature	Temp. Particles Gas	Temp. Particles	Temp. State of museum

TABLE II
Displayed content regarding the passengers (pass.) and the position of the bus

Area	**1**	**1**	**2**	**3**	**3**	**4**
Pass.	$\emptyset$	P1	P1, P2	P1, P2	P2	P2
Display	$\emptyset$	Temp.	Temp., Gas, Particles	Temp., Particles	Temp.	Temp., Museum info

The simulated system is configured as shown in Table I with several areas among the smart city and different services providing data (sensors or else). The different data provided is linked to several topics: *general information* for Temperature, *air quality* for particles and polluting gas measurements, and *museums* for museums information. Also, two persons are considered in the system. *P1* is interested in *general information* and *air quality* and the *P2* is interested in *general information* and *museums*.

The different contents displayed in the bus are shown in Table II. The displayed content is obtained when the manager executes the generated service plan retrieving data from the data producing services and sending the merged result to the service of the bus display.

Thanks to this deployment, the approach has been validated and its functionality enables the autonomic manager to automatically use different services to retrieve data and provide it to the right service(s) dynamically and on the fly.

B. Response time dimensioning

Other experiments were conducted to study the execution time of the approach. Indeed, in our use case the system should respond in a reasonable time to avoid the case where entities miss information. To do so, a set of initial Ressource

Description Framework (RDF) files were generated regarding different parameters in order to simulate different situations and contexts: number of available contents in the system at a given time, number of persons, number of passengers among the persons, number of possible topics and locations. For the semantic part of the analyser, (RDF and OWL files, SWRL rules, inference) several tools have been used: OWLAPI (v. 4.2.7), SWRLAPI (v. 2.0.0), Drools engine (v. 6.5.0) and JFact reasoner (v. 4.0.4). For the graph grammar based part, the AGG engine (v. 2.1) has been used for the transformation of the graphs (RFC) generated by the analyser. The measurements were made using a Ubuntu server (Intel Xeon CPU (3 Ghz), 32 GB of RAM, OpenJDK version 1.8).

The considered deployment handles a bus at once and its passengers. In this context, various number of passengers are considered and the system has to decide which content to display in the bus or to send it individually to the targeted entities. For each input, the interests of each person are randomly established. The related topics of interest and locations to each content are attributed randomly too.

The experiments were conducted with various number of passengers (10 to 60), various amount of contents to broadcast (10 to 100) and possible topics of interest are variable (10 to 100). The number of considered interesting locations is fixed to 25. The different results of the experiments are shown in Figure 6. The average response time is presented with respect to the number of passengers in the system (with a fixed number of topics).

The response time for the *semantic inference* is shown in *hatched grey*. The response time for the *graph* part (generation of the RFC graph to transform and transformation time cumulated) is shown in *black*.

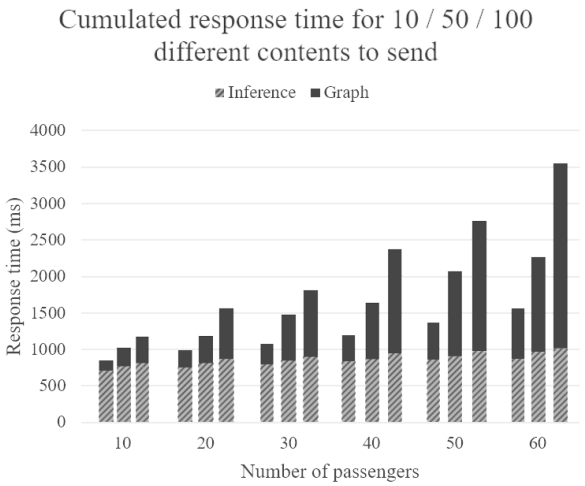

Fig. 6. Average response time in function of number of passengers for several number of contents (10 / 50 / 100)

Regarding the *response time of the semantic part*, several points can be highlighted. First, it takes globally more time when more contents are considered. This is explained by the complexity to handle and to determine which content should be sent to which entity.

On the other hand, regarding the *response time of the graph part* shown in Figure 6, it increases relatively to the number of entities to handle (here passengers). Indeed, for a given set of contents and linked topics of interest, when the number of passengers of the bus increases, the response time of the graph transformation increases up to 2 seconds with 100 contents to diffuse. This is due to the necessity to aggregate more contents together as there may be more persons interested in the same kind of contents which is done by repeating one rule.

The worst case in Figure 6 takes up to 3,5 seconds to be executed with 60 passengers and 100 contents to broadcast. In this case, around 5340 axioms are considered in the ontology and the RFC graph to transform contains 500 elements. However, in a real use case, such numbers should not be reached: broadcasting more than 50 contents at once on a display device would completely saturate the users of information and could not fit easily in one screen.

It is worth noting that a decentralised deployment may suit our use case in terms of response time. Indeed, an important point to highlight here is that the number of content and topics used for the measurements are considered at once, in one and only one loop of the autonomic manager. The more contents and persons the system handles at once, even regarding to one bus, the longer the response time will be.

VII. Related works & Discussion

The IoT brings opportunities, but also challenges and concerns. This paper focuses on the autonomic management of IoT based systems. In this context, a semantic and formal grammar graph based approach is proposed. The planner of the autonomic system manager presented in this paper provides an execution plan which enables building new value added complex services through the composition of elementary IoT services. The composition of services has been widely studied in the literature.

The first important aspect many works focus on is providing composition methods and algorithms to automatically generate the composition plan based on a pre-established abstract workflow of services and given set of known services. [15], [16] and [17] studied several contributions in this field. They gather several works based on AI algorithms or particular calculation and planning techniques to compute an orchestration plan based on the given set of services and the target abstract workflow. Also, they present some works that aim at producing a composition based on semantic enrichment for the description of the services (e.g., WSDL).

Considering a predefined workflow, some works perform classification or clustering of services [18] [19], or guarantee a single Quality of Service (QoS) parameter with inputs and outputs characterization [20].

Finally, another important aspect of service composition remains in service selection: it is the process of selecting the best services to perform a given composition plan (i.e., given abstract workflow) according to different parameters. In [4]

Guidara *et al.* propose a dynamic selection approach for the composition of services: the goal is to fulfil dynamic selection actions to avoid changes and faults during execution of a pre-established plan. In [21], Raj *et al.* propose another approach to perform service selection based on QoS parameters.

Most of the aforementioned works assume that the abstract definition of the composition is known in advance through a definition of abstract workflow. This assumption can be restrictive in several domains such as in the smart cities where user requirements cannot be specified in advance as precise abstract workflow that match existing services. However, in this paper we do not assume that the description of the plan is already given. In this work, the aim is for the manager to automatically generate the plan according to expressed requirements and then execute it on the fly. Moreover, those approaches, particularly service composition selection approaches, could be integrated in the execution phase to enrich our approach to find the "best" services composition. Indeed, the selection algorithms would be an enhancement for the execution phase including dynamic selection mechanisms based on the set of services provided in the execution plan. Indeed, for now the service selection remains arbitrary in the execution phase.

VIII. Conclusion

In this paper, we presented a service oriented autonomic management approach for IoT systems. This framework relies on a semantic and graph transformation based approach. It is composed of three main elements: the underlying IoT system to manage, the autonomic manager which is based on the MAPE-K paradigm, and the knowledge base which is defined on top of semantic and graph grammar based models.

The proposed approach enables IoT systems to cope with the dynamic and versatile nature of these systems and their environments by enforcing autonomic management actions while guaranteeing provisioning complex new value added services. The analysis and planning steps which present the core of the proposed work have been implemented and evaluated using a decentralized deployment. Also, a first full prototype in the context of smart city with connected vehicles has been deployed and validated.

As stated in the evaluation section, the proposed approach is suitable to handle particular problem classes such as the presented use case. We plan in our future work to study in depth the scalability of the proposed approach to handle larger scale systems and the possibilities of various deployments to cope with it. Another important direction of our future work is to integrate non-functional properties (Quality of Service, energy, etc.) in the decision making but also in the service management phase.

Acknowledgements

This work has been co-funded by a FEDER-FSE 2014-2020 fund of the Région Midi-Pyrénées & the European Union and French Government (program: investment for future) in the project: Smart Services for Connected vehiCles - S2C2.

References

[1] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi, "Toward better horizontal integration among IoT services," *Communications Magazine, IEEE*, vol. 53, no. 9, pp. 72–79, 2015.

[2] F. Aïssaoui, G. Garzone, and N. Seydoux, "Providing Interoperability for Autonomic Control of Connected Devices," in *Interoperability, Safety and Security in IoT. InterIoT 2016, SaSeIoT 2016. LNICST* (N. Mitton, H. Chaouchi, T. Noel, T. Watteyne, A. Gabillon, and P. Capolsini, eds.), vol. 190, ch. InterIoT, pp. 33–40, Paris: Springer, Cham, 2017.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[4] I. Guidara, I. Al Jaouhari, and N. Guermouche, "Dynamic Selection for Service Composition Based on Temporal and QoS Constraints," in *2016 IEEE International Conference on Services Computing (SCC)*, pp. 267–274, IEEE, 6 2016.

[5] R. Aschoff and A. Zisman, "QoS-Driven Proactive Adaptation of Service Composition," in *9th International Conference, ICSOC 2011* (G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad, eds.), vol. 7084, (Paphos, Cyprus), pp. 421–435, Springer, Berlin, Heidelberg, 2011.

[6] R. Ramacher and L. Monch, "Reliable Service Reconfiguration for Time-Critical Service Compositions," in *2013 IEEE International Conference on Services Computing*, pp. 184–191, IEEE, 6 2013.

[7] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, pp. 41–50, 1 2003.

[8] N. Delgado, A. Gates, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *IEEE Transactions on Software Engineering*, vol. 30, pp. 859–872, 12 2004.

[9] M. Palacios, J. García-Fanjul, and J. Tuya, "Testing in Service Oriented Architectures with dynamic binding: A mapping study," *Information and Software Technology*, vol. 53, pp. 171–189, 3 2011.

[10] A. Mosincat and W. Binder, "Automated maintenance of service compositions with SLA violation detection and dynamic binding," *International Journal on Software Tools for Technology Transfer*, vol. 13, pp. 167–179, 4 2011.

[11] N. Seydoux, K. Drira, N. Hernandez, and T. Monteil, "Iot-O, a core-domain IoT ontology to represent connected devices networks," in *Knowledge Engineering and Knowledge Management. EKAW 2016. LNCS* (E. Blomqvist, P. Ciancarini, F. Poggi, and F. Vitali, eds.), vol. 10024 LNAI, pp. 561–576, Springer, Cham, 11 2016.

[12] G. Rozenberg and H. Ehrig, *Handbook of graph grammars and computing by graph transformation*, vol. 1. World Scientific, 1997.

[13] I. Horrocks, P. F. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL : A Semantic Web Rule Language Combining OWL and RuleML," *W3C Member submission 21*, no. May 2004, pp. 1–20, 2004.

[14] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, and A. Wagner, "Algebraic Approaches to Graph Transformation: Part II: Single Pushout Approach and Comparison with Double Pushout Approach," *Handbook of graph grammars and computing by graph transformation*, pp. 247 – 312, 1997.

[15] K. S. M. Chan, J. Bishop, and L. Baresi, "Survey and comparison of planning techniques for web services composition," *Africa*, no. October, pp. 43–54, 2007.

[16] M. Aljawarneh, L. D. Dhomeja, and Y. A. Malkani, "Context-aware Service Composition of Heterogeneous Services in Pervasive Computing Environments : A Review," no. 1, pp. 0–5, 2016.

[17] D. Hutchison and J. C. Mitchell, *Semantic Web Services and Web Process Composition*, vol. 3387 of *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[18] Z.-z. Liu, D.-h. Chu, Z.-p. Jia, J.-q. Shen, and L. Wang, "Two-stage approach for reliable dynamic Web service composition," *Knowledge-Based Systems*, vol. 97, pp. 123–143, 4 2016.

[19] R. A. H. M. Rupasingha, I. Paik, and B. T. G. S. Kumara, "Domain-aware Web Service Clustering based on Ontology Generation by Text Mining," 2016.

[20] S. Chattopadhyay, A. Banerjee, and N. Banerjee, "A Scalable and Approximate Mechanism for Web Service Composition," *Proceedings - 2015 IEEE International Conference on Web Services, ICWS 2015*, vol. 11, no. 4, pp. 9–16, 2015.

[21] R. J. R. Raj and T. Sasipraba, "Web service selection based on QoS Constraints," *Trendz in Information Sciences & Computing (TISC), 2010*, vol. 6, pp. 156–162, 2010.