A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare

Emna Mezghani, Ernesto Expósito, Khalil Drira

▶ **To cite this version:**

Emna Mezghani, Ernesto Expósito, Khalil Drira. A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare. IEEE Transactions on Emerging Topics in Computational Intelligence, 2017, 1 (3), pp.224-234. 10.1109/TETCI.2017.2699218 . hal-01535140

HAL Id: hal-01535140
https://hal.science/hal-01535140

Submitted on 8 Jun 2017

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.

A Model-driven Methodology for the Design of Autonomic and Cognitive IoT-based Systems: Application to Healthcare

Emna Mezghani[a,b], Ernesto Exposito[c] and Khalil Drira[a]

[a] LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
[b] Luxembourg Institute of Science and Technology, 5, Avenue des Hauts-Fourneaux, L-4362, Esch/Alzette, Luxembourg
[c]Laboratoire LIUPPA, University of Pau and Adour Countries, Anglet, France
{emna.mezghani|khalil}@laas.fr, ernesto.exposito@univ-pau.fr

Abstract. Due to its abilities to capture real-time data concerning the physical world, the Internet of Things (IoT) phenomenon is fast gaining momentum in different applicative domains. Its benefits are not limited to connecting things, but lean on how the collected data is transformed into insights and interact with domain experts for better decisions. Nonetheless, a set of challenges including the complexity of IoT-based systems and the management of the ensuing big and heterogeneous data and as well as the system scalability; need to be addressed for the development of flexible smart IoT-based systems that drive the business decision-making. Consequently, inspired from the human nervous system and cognitive abilities, we have proposed a set of autonomic cognitive design patterns that alleviate the design complexity of smart IoT-based systems, while taking into consideration big data and scalability management. The ultimate goal of these patterns is providing generic and reusable solutions for elaborating flexible smart IoT-based systems able to perceive the collected data and provide decisions. These patterns are articulated within a model-driven methodology that we have proposed to incrementally refine the system functional and nonfunctional requirements. Following the proposed methodology, we have combined and instantiated a set of patterns for developing a flexible cognitive monitoring system to manage patients' health based on heterogeneous wearable devices. We have highlighted the gained flexibility and demonstrated the ability of our system to integrate and process heterogeneous

large scale data streams. Finally, we have evaluated the system performance in terms of response time and scalability management.

Keywords. IoT-based system, Big Data, Autonomic Computing, Design Patterns, Cognitive Computing

1. Introduction

Information and communication industry has made great strides, and is gaining further recognition with the integration of the Internet of Things (IoT). Based on a study done by Cisco [1], the effective use of IoT is much more than just connecting things, it encompasses as a primary concern the management and the transformation of the IoT generated data into insights and business benefits. Nowadays, the IoT market witnesses an important increase. According to Gartner, around 25 Billion Connected "Things" will be in use in 2020 .The wide adoption of IoT leads to the production of complex system of systems (SoS) [2], which churn out myriad of heterogeneous data, that should interact with each other to meet the system requirements. As these technologies sense the physical world, it is easy to collect data, but hard to manage.

Automating the management of IoT-based systems may address the system complexity, accelerate and facilitate the interaction with the domain experts for better decision making. The autonomic computing initiative [3] has a strong focus on managing complex systems through automating tasks based on the MAPE-K loop pattern (abbreviation of Monitoring, Analysis, Plan, Execution, and Knowledge). It has been widely used [5-8] for designing self-managed systems that automatically adapt their structure and behavior based on the context changes. It seems that adopting the autonomic computing for managing IoT-based system complexity is promising [4].

Nevertheless, the integration of IoT poses new challenges for the development of flexible smart IoT-based systems, based on the autonomic computing, to continuously perceive the data and generate insights at the right time. These challenges include the dynamic evolution of the system

requirements, as well as the big data and scalability management. For instance, integrating new sensors at run-time to monitor and collect more data about the system may hinder the well-run of components implementing computational intelligence to learn from this data and generate hidden information, due to its heterogeneity and the non-understanding of the meaning of the received data. Consequently, the system re-engineering is required to integrate these new data sources, which leads to a high design and implementation cost.

Furthermore, IoT-based systems implementing real-world applications continuously evolve and generate unforeseen requirements [9], which requires the coordination of the management processes as well as interacting with humans to learn from his intelligence and provide more accurate analytics. Recently, a new era of IoT called "Cognitive IoT" [10] has been enunciated. It aims at integrating cognitive technologies into IoT-based systems to ensure the smart management through enabling the cooperation and interaction between IoT and human. Coupling autonomic computing with cognitive computing sheds light on unprecedented opportunities for the development of smart IoT-based systems.

Consequently, we propose a model-driven methodology for the design and development of smart IoT-based systems. Within this methodology, we defined a set of design patterns, which combine autonomic computing and cognitive computing principals, to effectively assist software architects providing flexible solutions based on the system's requirements. To this end, we harvested the list of software design patterns, and we surveyed knowledge engineering and management techniques in order to propose autonomic and cognitive design patterns that elucidate how to manage IoT-based system evolution and detail their interaction with human.

This paper is organized as follows. Section 2 discusses existing works dealing with the design of autonomic systems and highlights their limitations to support the integration of IoT. Section 3 harvests existing patterns that can be reused for designing smart IoT-based systems. Then, Section

4 portrays our proposed methodology for the design of autonomic and cognitive IoT-based systems, and deepens some of the design patterns that we have proposed to manage problems related to the system context evolution, as well as the big data and scalability management. Section 5 illustrates the efficiency of the proposed patterns through the implementation of a cognitive monitoring system for the management of the patient health evolution, based on semantic and big data technologies. Section 6 evaluates the system performance on the cloud and highlights the impact of IT resources configuration. Finally, conclusion and future work are presented in Section 7.

2. Related Work

In this section, we surveyed research works dealing with designing autonomic systems, and investigated their ability to support IoT challenges. Many research activities within the autonomic and self-adaptive research community proposed software patterns for the design of autonomic self-adaptive systems [9, 11]. However, there is a lack of dynamic coordination of the management processes (Monitoring, Analysis, Plan and Execution), as well as concrete implementation of these patterns. Al-Shishtawy et al. [12] proposed four patterns for the coordination of multiple autonomic managers for the network management: the stigmergy pattern, the hierarchical management pattern, the direct interaction pattern and the sharing of the managed elements pattern. The stigmergy pattern represents indirect interaction where the autonomic managers make changes on the managed resources, and these changes are sensed by other autonomic managers that will do more actions. Within the stigmergy, undesired behaviour may occur at runtime if many autonomic managers are involved, especially in system of systems. The hierarchical management pattern adopts a reflexive approach in which the autonomic managers (children) monitors and manages the systems. In turn, these managers are controlled and orchestrated by another autonomic manager (parent). This pattern may cause scalability problems in complex systems, when the parent should manage a big number of autonomic managers. The direct interaction relies on binding to the

appropriate managers, while the sharing of the managed elements pattern refers to sharing information about their states (knowledge) and synchronizing their actions. The proposed patterns are abstract since the interactions among the MAPE-K functions are not clearly presented.

Weyns et al. [13] presented a seminal work identifying the interactions of MAPE loops through considering both the inter/intra-interactions. The authors proposed five decentralized design patterns for self-adaptive systems: the coordinated control pattern, the information sharing pattern, the master/slave pattern, regional planning pattern and the hierarchical control pattern. Within the coordinated control pattern, management processes belonging to the same autonomic manager can interact with external management processes having the same type and belonging to another autonomic manager. The information sharing pattern illustrates the M-M interaction while the other processes operate independently. From the scalability perspective, the second pattern provides a more scalable solution than the first one, since it limits the interactions to M-M. The Master/Slave pattern considers the (M, E) as slaves specific to the managed element, while the (A, P) as masters shared among the slaves. For large-scale systems, this pattern may cause problems related to the master overhead, if many Monitoring processes send data to the Analysis process. The regional planning pattern proposes for each region one Plan process which is shared by (M, A, E) of each sub-system belonging to this region. This Plan may interact with another Plan belonging to another region. Finally, the hierarchical control pattern, which is similar to the hierarchical management pattern, is presented in [12]. Choosing the right pattern or combination of patterns depends on the system requirements and its complexity. Both Al-Shishtawy et al. [12] and Weyns et al. [13] proposed patterns for the coordination of autonomic managers at design time. However, they did not consider the communication and data integration challenges as well as the system requirements evolution at run-time (e.g. the deployment of new monitoring and analysis processes due to the integration of new sensors), which are among the characteristics of IoT-based systems. In this

context, Oliveira et al. [14] provided an autonomic coordination at runtime based on the knowledge component and exchanging events among the monitoring and execution components. They identified two types of knowledge: the private knowledge which is shared with the components of the same loop, and the public knowledge which is shared among the existing MAPE loops. To synchronize the execution of the communicating control loops, the authors proposed to share and exchange a token among all the loops. Thus, each loop needs to request for the token, and when this latter is released the first loop requesting the token starts its execution, and so on. This solution is very efficient to answer confidential and concurrency issues and enable the system stability. It doesn't impact the system scalability but highly impact on the processing time. However, locking the execution of the loop excludes the parallel management and increases the waiting time of the adaptation, which is not adapted for real-time system management.

We noted that none of these patterns integrate the IoT-based systems, except the work of Vidal et al. [15] which presents a model-driven methodology, named MindCPS, for the design and development of autonomic Cyber Physics System (CPS). The methodology implements a meta-model that drives the system design through defining the interaction of the autonomic components and the main functionalities. The proposed model describes the sensors measurements and protocols, identifies simple and complex filters to process the received data and detects symptoms to enable the adaptation. A set of model-to-code transformation, which automatically generates the JAVA code implementing the autonomic control loop as well as EPL queries for the real-time management of the events and SQL queries for the management of the non-real time data, are identified. However, this work did consider data heterogeneity and its volume, as well as the scalability of such complex system.

To conclude, almost all works detail the interactions among the management processes (M, A, P, E) for the design of autonomic systems. Nonetheless, with the proliferation of IoT, most of the

data are unstructured generated by sensors. Thus, new challenges occur such as the data heterogeneity, the data volume and velocity. Consequently, new patterns that extend the basic management processes to support big data and scalability are required. Moreover, despite the importance of the knowledge component for the development of smart IoT-based systems, its structure and its interaction with the management processes remain cloudy. Indeed, autonomic computing technologies do not rely on instructive and procedural information [16] which are the basis for smart decision-making. The knowledge should not be overlooked, it is important to enrich the knowledge with cognitive capabilities that allows interacting with human to learn from his intelligence and to generate hidden business insights.

In the next section, we identify the main design problems that we may encounter when designing flexible and smart IoT-based systems, as well as existing reusable software patterns.

3. Identification of Design Problems and Reusable Patterns for Smart IoT-based Systems

Generally, the coordination of the management processes remains an open issue in autonomic management; most of the research works focus on the interaction among these processes. We found that in Artificial Intelligence domain, the blackboard pattern [17] has been widely used for the dynamic control and coordination of the knowledge sources (KS) based on a control component (C). This later supervises the shared blackboard among the KS. If the blackboard is modified, the control component activates the appropriate KS. It seems that this pattern is useful to model the management processes coordination, since we consider them as knowledge sources that generate new knowledge concerning the managed element. In complex systems, the distribution of the management processes allows gaining performance management, but their interactions become more difficult. Within a distributed system, management processes may not have enough information concerning other separate processes such as their definition and endpoint in order to

interact with. In this context, we found that the mediator pattern [18] provides a loosely coupled solution where the distributed management processes are communicating through the mediator. Nevertheless, the heterogeneity of the processes description is challenging to ensure the interoperability. Thus, we propose to extend the mediator with a common semantic description of the processes.

From the data management level, managing the IoT-based system scalability is crucial to provide timely information. The publish/subscribe pattern [19] offers a scalable management of the monitoring process as well as an easy extensibility of the system to support new devices. However, the publish/subscribe pattern presents inflexible semantic coupling. Indeed, in this pattern, the semantic and data structure are well defined. Thus, if a new device is deployed to send different data structure (e.g. different data ordering and delimiter), the consumer fails processing the data. With this in mind, we propose to provide a flexible solution by enriching the publish/subscribe pattern with a semantic description of the data structure and ordering. Thus, adding new devices will not affect the data consumption process, since the consumer knows all information related to the received data: the meaning of the generated data and its structure. Consequently, we propose to use ontology as a semantic description method to deal with the heterogeneity from two perspectives: first to describe the management processes, their functions and endpoints; second to describe the meaning of the managed data and its structure to provide more accurate analysis. Another important aspect that characterizes the IoT-based system is the near-real time data processing. With the variety of the received data, it is important to filter and curate the data for better interpretation and visualization. Coupling the interceptor pattern with the chain of responsibility pattern [18] provides a solution for near-real time data curation based on filters implemented in the subscriber.

The semantic description including ontologies plays an important role in providing cognitive systems able to perceive and understand the data. However, in large-scale systems, the frequent access to this layer may cause bottleneck and performance limitation, especially in distributed systems. In this sense, we found that the cache pattern is useful. Nonetheless, the update of the cache is challenging, especially in highly dynamic systems where new devices are added and the context is changing. At this point, we found that the observer pattern [18] can be used to automatically update the distributed caches at the right time.

From the infrastructure management level, deploying a high number of management processes and knowledge components to manage complex system is constrained by the availability of IT resources. Cloud computing comes with solutions to deal with such problems. Virtualizing the resources will gain the system the ability to automatically allocate the resources based on the process workload. Moreover, conceiving a multi-tenant architecture, where the management processes and the knowledge components share the IT resources with respect to the confidentiality, fosters sharing the cost among the consumers, thus, reducing the cost.

Based on this study, we propose a set of Autonomic Cognitive Design Patterns that will be enunciated within a model-driven methodology for the design of smart IoT-based systems.

4. A Model-driven Methodology for the Design of Smart IoT-based Systems

We propose a model-driven methodology that combines software modeling and knowledge engineering principals to facilitate the design and development of autonomic and cognitive IoT-based systems. Figure 1 depicts an overview of the proposed methodology. Two main phases are identified: (1) Requirements Identification and (2) Requirements Formalization.

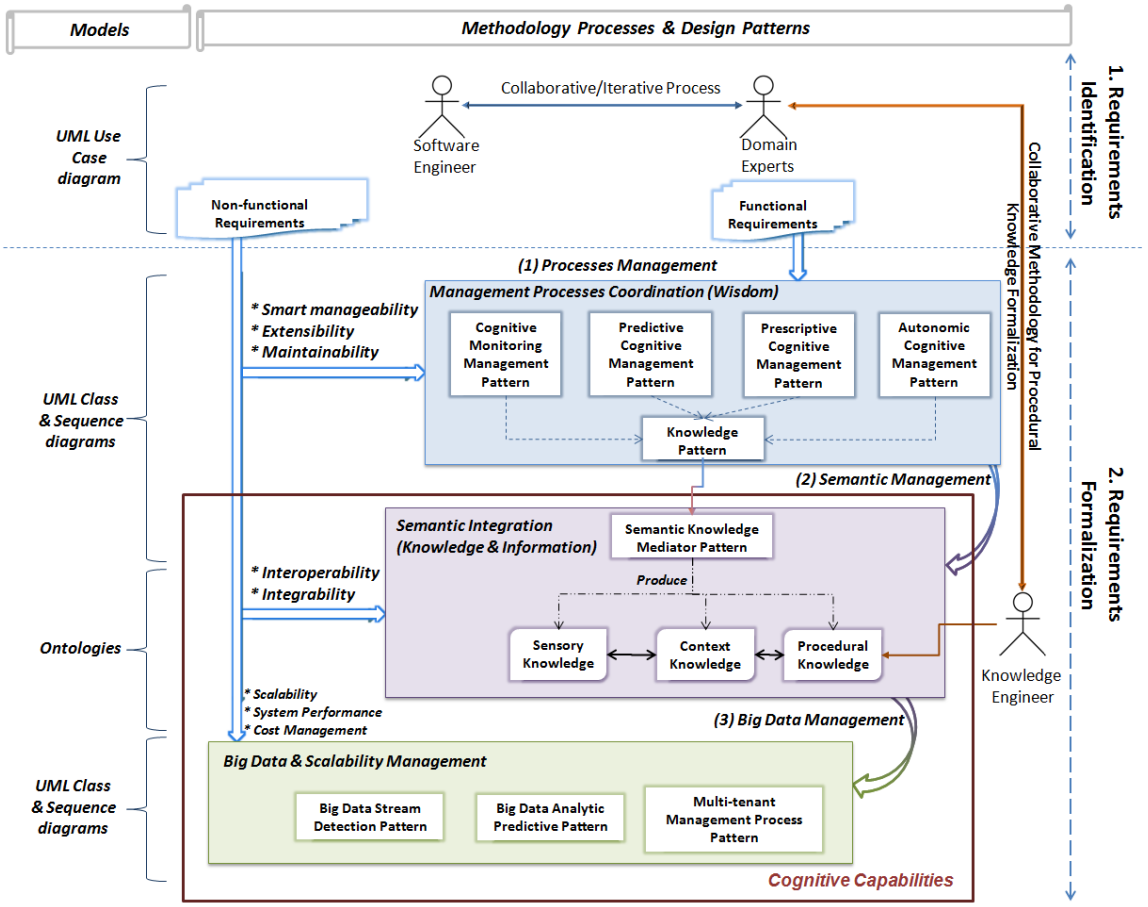

Figure 1. A model-driven methodology for the design of Smart IoT-based systems

Requirements identification is based on discussions with the domain experts in order to retrieve the system functions and identify the non-functional requirements. It is an iterative process, where the functional requirements are incrementally refined and represented using the UML use cases diagrams describing the functions of the system without specifying any implementation details. The next phase, which is the requirement formalization, focuses on formalizing and structuring the identified requirements into concrete models describing the system processes' interactions. The aim of our methodology is providing smart IoT-based systems, thus, we propose to map the system functions into management processes which can be the monitoring, analysis, plan and execution process; or their combination. For the development of smart IoT-based systems, we denote the existence of a new actor, in the second phase, who is the knowledge engineer. This later

collaborates with the domain experts in order to formalize the business problem-solving knowledge in order to allow the system generating business decisions.

Within the Requirement Formalization phase, we introduce three sub-levels in order to incrementally deal with challenges related to the design of smart IoT-based systems such as the coordination of management processes, the semantic integration and the big data management. The first level is the Management Processes' Coordination. Within this level, we identified five patterns that consider the smart manageability, extensibility and maintainability of IoT-based systems. These patterns are represented based on UML class diagram to design the structure and based on UML sequence diagrams to draw the behaviour, and delineate how the management processes should be coordinated and interact to meet the system's functional requirements based on the knowledge pattern. The proposed patterns can be combined to solve complex requirements. Once the management processes are identified and modeled, the next level is the Semantic Integration where mainly the information about the system and its environment as well as the procedural knowledge (know-how) for decision-making are formalized in order to be automatically reused by the management processes. Within this level, we identified the Semantic Knowledge Mediator pattern to guarantee the interoperability and the integration for the smooth functioning of the system. The application of this pattern leads to the production of three types of semantic models (the sensory, the context and the procedural knowledge) which are designed based on ontologies. It is worth mentioning that the knowledge engineer intervenes at this level and collaborate with the domain experts when formalizing their tacit knowledge (know-how). Finally, the last level is the Big Data & Scalability Management where we defined three patterns that deal with the big data challenges (volume, variety, velocity), scalability, system performance as well as the system cost management. In the following, we provide an overview of the proposed patterns for the design of autonomic and cognitive IoT-based systems. We adopted the template proposed by Buschmann et

al. [20] to describe the proposed patterns. We mainly present the pattern name, the context of its use, the problem and the proposed solution.

4.1.Management Processes' Coordination

We referred to the blackboard pattern as the basis for the coordination of the management processes. We proceed with a separation of concerns approach where the functional requirements are decomposed into atomic requirements ascribed to the appropriate atomic management process that will be coordinated to manage complex requirements. We propose four patterns that model the management processes coordination, and a knowledge pattern.

- ***Pattern Name: Knowledge pattern***

Context: Many works implementing the autonomic computing centralize the knowledge component and provide a common repository describing the knowledge.

Problem: Centralizing the knowledge components may cause a bottleneck and scalability problems. Moreover, mixing everything together causes problems when maintaining the knowledge.

Solution: Thus, we propose to decompose the knowledge for the smart management of IoT-based systems into three sub-components: Sensory Knowledge, Context Knowledge and Procedural Knowledge. This decomposition is inspired from the human memory model [21] (sensory memory, short memory and long-term memory), more precisely we referred to the Tulving classification [22] which identifies within the long-term memory the episodic memory describing events, semantic memory describing real-world facts and procedural memory describing human skills. So, in our pattern, the Sensory Knowledge semantically describes the monitored data and the main characteristics of the sensors used to capture the data. The Context Knowledge semantically describes the managed element including the conditions, the context evolution and events. Finally,

the Procedural Knowledge semantically describes the know-how that includes the procedures and solutions that can be performed to solve problems.

Consequences: The decomposition of the knowledge structure fosters the collaboration when acquiring its content and facilitates the maintenance process of each sub-component. Moreover, by distributing these components, the system offers better scalability. However, the distribution may raise problems when maintaining the classes that interconnect these knowledge sources. As solution, we propose to provide a generic schema for each knowledge sub-component, then, it will be specialized to the domain.

- ***Pattern Name: Cognitive Monitoring Management pattern***

Context: Conventional monitoring systems are developed for specific devices generating data with the same unit, using the same syntax and representation.

Problem: If new IoT devices are integrated, new applications need to be developed to explore the acquired data which is costly. Moreover, there is a lack of bi-directional interaction between IoT systems and human, since the primary objective was getting the captured data.

Solution: Based on the blackboard pattern, the Cognitive Monitoring Management pattern enables the interaction of IoT with human. It identifies a bidirectional interaction: IoT-Human interaction to visualize the data, extract new insights and receive notifications in case of context changes; and the Human-IoT interaction to manage the system through modifying its context and allow the IoT-based system learning from experts and acquiring knowledge. Within this pattern, only the

monitoring process is automated. Figure 2 illustrates the proposed pattern and the main interactions among the knowledge sources (human, monitoring process) and the knowledge component.

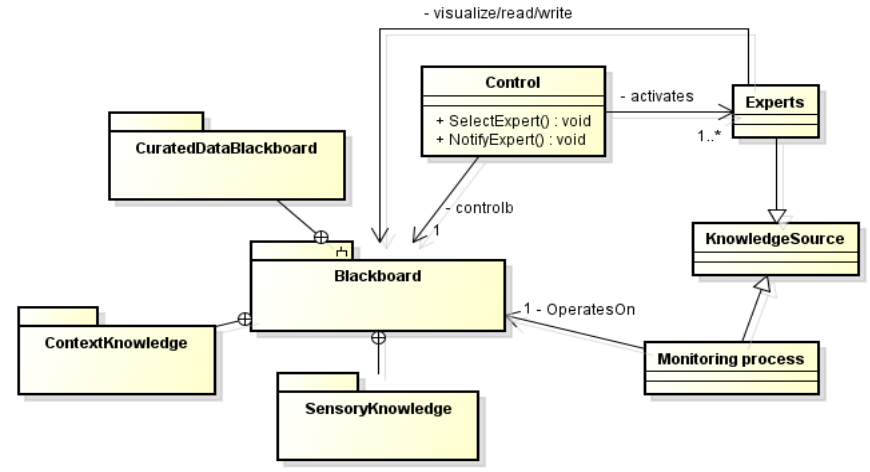

Figure 2. Cognitive Monitoring Management pattern

As shown, the blackboard component is composed of the sensory knowledge and the context knowledge that respectively allows the monitoring system understanding the meaning of the data and detects personalized detections. Moreover, it includes the CuratedDataBlackboard that contain the harmonized data sent by the monitoring, after processing, for visualization.

Consequence: Based on the sensory knowledge, the proposed pattern is able to manage heterogeneous IoT devices and offers a flexible system that can be easily extended with new devices that collect individualized data. The blackboard maintainability is guaranteed and can be done collaboratively

- ***Pattern Name: Predictive Cognitive Management pattern.***

It extends the Cognitive Monitoring Management pattern through modeling the coordination between the monitoring, the analysis processes and the expert. It also delineates the interaction of the management processes with the sensory and context knowledge in order to generate new knowledge about the managed element.

- ***Pattern Name: Prescriptive Cognitive Management pattern.***

This pattern coordinates the monitoring, the analysis and the plan processes, while interacting with the experts. It incorporates new mechanisms that allow the IoT-based system generating recommendations to assist the experts in taking business decisions.

- ***Pattern Name: Autonomic Cognitive Management pattern.***

This pattern proposes the dynamic discovery of management processes based on the system context in order to manage at runtime the system requirement evolution. The dynamic discovery is achieved based on a semantic characterization of the management processes and the conditions for their activation. The expert is always kept in the loop to validate the execution of the discovered processes and the generated plans.

4.2.Semantic Integration

- ***Pattern Name: Semantic Knowledge Mediator***

Context: Enabling a smart management of the IoT-based systems requires the collaboration of various sources (human, machines and processes) to get a deep understanding of the business knowledge (know-how) and detailed information about the system and its environment (know-what) as well as their integration.

Problem: The heterogeneity of the knowledge representation and its distribution are identified as impediments of the integration, system maintainability and extensibility.

Solution: We propose in Figure 3 the Semantic Knowledge Mediator pattern that extends the basic Mediator pattern with semantic capabilities for data/knowledge integration. The proposed pattern enables the collaboration of different kind of providers when populating the knowledge. Based on the knowledge pattern, three types of knowledge are distinguished which are produced by the providers based on a semantic model describing a common vocabulary of the domain. The annotations are stored in large scale triple stores for flexible reuse.

Consequence: The proposed pattern enables the collaboration and offers the semantic integration of various knowledge sources based on a common description of the domain. It guarantees also the flexibility of the system and its extensibility to support new knowledge sources. Many research activities proposed to store the monitored data in ontologies. However, this approach raises scalability problems in such data-intensive systems. Thus, this pattern characterizes only the data structure and meaning and stores these annotations in the ontology.

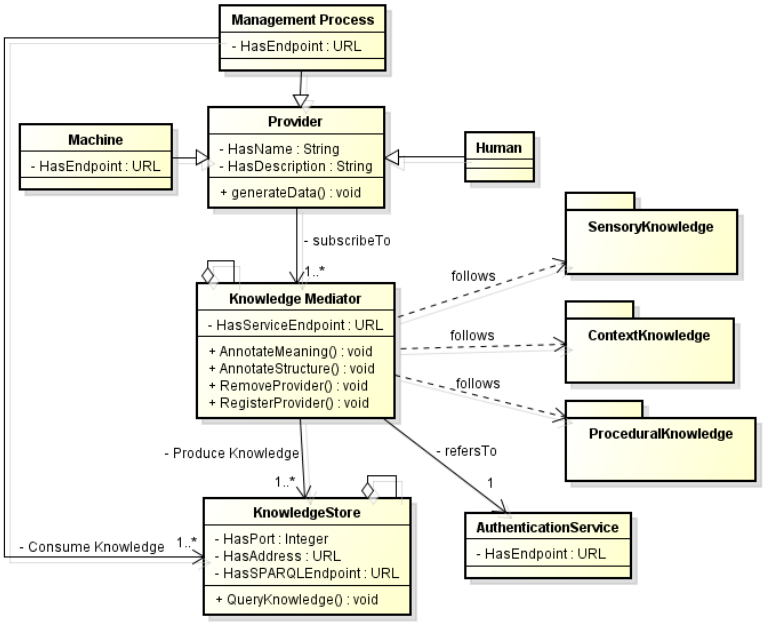

Figure 3. Semantic Knowledge Mediator pattern

4.3.Big Data & Scalability

The monitoring and the analysis are the main processes encountering big data challenges. Existing systems implementing autonomic computing do not consider such challenges. In this context, we propose the Big Data Stream Detection and the Big Data Analytic Predictive patterns to manage respectively the near real-time processing and the batch processing to predict new information. Besides these patterns, we propose the Management Process Multi-tenant pattern which delineates

the deployment of the management processes and the knowledge sub-components to provide scalable solution.

- ***Pattern Name: Big Data Stream Detection pattern.***

Context: Many vendors are developing their devices implementing their own applications which results in a diversity of the generated data. Offering a service to a wide range of patients to monitor the evolution of their health status as well as providing personalized detection requires guaranteeing the scalability to support huge data streams and the ability to correctly interpret it.

Problem: Besides the heterogeneity, the data velocity and their huge volume hinder data integration for near-real time anomalies detection as well as data stream visualization. Manually harmonizing and curating IoT-generated data requires IT skills, which is costly and time consuming.

Solution: Thus, we propose the Big Data Stream Detection pattern which extends the Cognitive Monitoring Management pattern with sub-processes that understand and automatically curate the received data, and retain attention. These sub-processes which belong to the Monitoring process refer to the SensoryKnowledge to get the meaning, store the curated data in the CuratedDataBlackboard for visualization and analytics, and detect changes based on the ContextKnowledge. Figure 4 portrays the proposed pattern. We reused existing software design patterns such as the publish/subscribe pattern (represented with the orange color) to guarantee the scalability between the data publishers (IoT devices) and the consumers responsible for harmonizing and/or detecting the context changes. We added a coordinator in order to enable the distributed and parallel execution of tasks within consumers and ensure fault-tolerance. Furthermore, we reused the interceptor pattern (represented with the pink color) when intercepting the received data from topics to pre-process data and harmonize it based on filters. We extended this pattern with a cache that retrieves semantic information concerning the IoT devices in order to provide flexible filters that operate independent from the tuples structure. The cache pattern has

been adopted twice to guarantee better system performance when frequently accessing to SensoryKnowledge and ContextKnowledge repositories. We adopted also the observer pattern (represented with the green color) to keep the caches up-to-date, for example when adding new IoT devices; or when the experts modify the context.

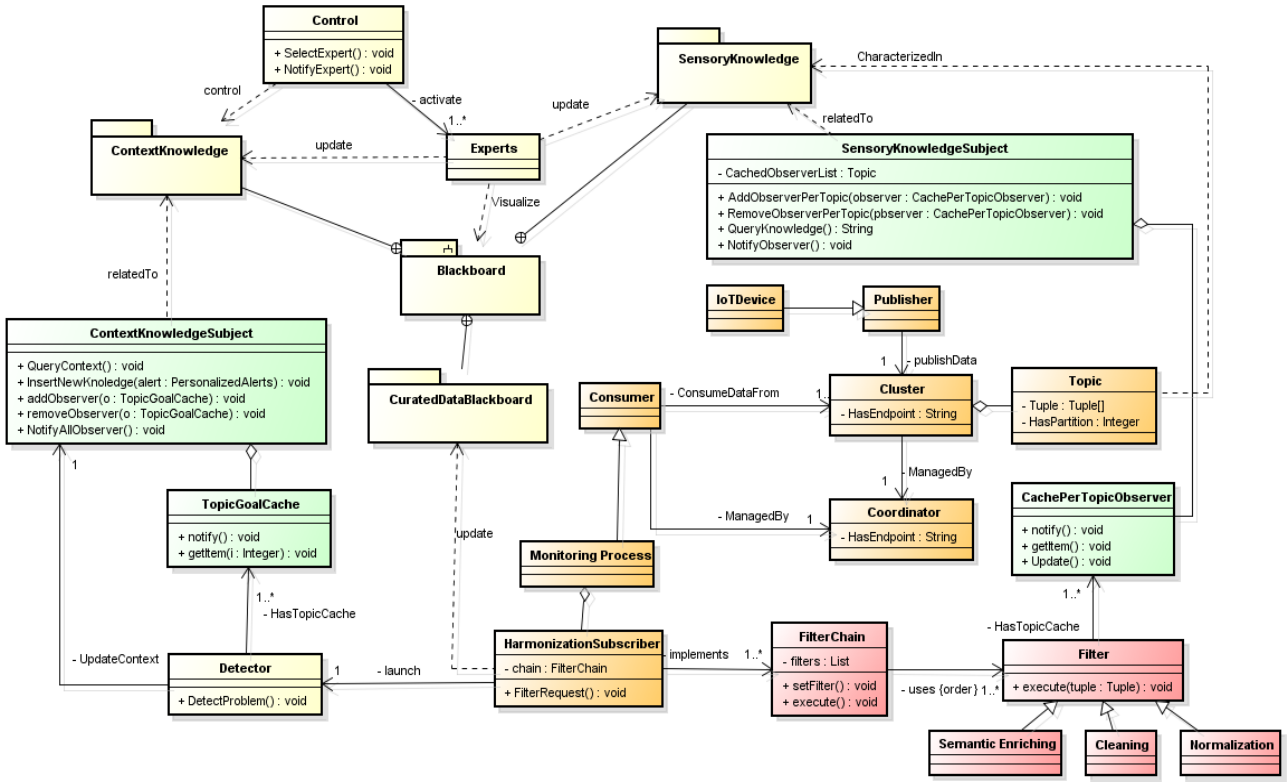

Figure 4. Big Data Stream Detection Pattern

Consequence: Based on the semantic description of the generated data, the proposed pattern fosters the stream data integration, and proposes a flexible harmonization process that can be shared and reused by various data providers. Likewise, it guarantees the system maintainability and flexibility through separating the data processing algorithms from the data structure and type. It offers also a scalable solution based on the publish/subscribe pattern and provides better performance through caching the sensory and context knowledge.

- ***Pattern Name: Big Data Analytic Predictive pattern.***

This pattern extends the *Predictive Cognitive Management* pattern to support the big data management for batch processing. This pattern is also inspired from the human brain information processing: it imports data from external database into a temporary database (short-term memory), applies parallel batch-processing to harmonize the data and store it in clusters (long term memory). Then, it introduces parallel data analytic services that may implement machine learning operating on the harmonized data clusters to generate new knowledge.

- ***Pattern Name: Management Process Multi-tenant pattern.***

In complex IoT-based systems, different sub-systems need to be managed. The deployment of the management processes, knowledge and databases for each managed element will increase the system complexity as well as the cost in terms of memory, network traffic and CPU consumption. In this sense, based on the multi-tenancy pattern[1] defined by the cloud computing, the *Management Process Multi-tenant* pattern is introduced to manage the system scalability and cost through delineating the deployment of the management processes, the knowledge components and databases, and highlighting their interactions. This pattern combines the different modes offered by the multi-tenancy pattern (isolated and shared) in order to allow multiple users sharing the same management processes' instances and both the sensory and procedural knowledge, while isolating the context knowledge and the database of each tenant to guarantee confidentiality. It implements also the dedicated mode to allow the tenants sharing the IT resources to reduce the cost based on virtualization.

5. Use Case: Patient Health Management

We propose to follow our model-driven methodology to design and develop a smart monitoring system able to manage patients' health evolution based on wearable devices. In the context of

[1] http://www.cloudcomputingpatterns.org/cloud_application_architectures/#multi-tenancy

managing diabetes, we assume that we have wearable devices that measure the "Blood Sugar". A group of them express the measured data in mg/dL and are using ';' as delimiter when sending data, while the rest measured data in mmol/L and are using ',' as delimiter.

The functional requirements of this system encompass the visualization and detection of the patient health anomalies while interacting with the appropriate physician. Thus, we mainly instantiated and integrated three patterns: the *Cognitive Monitoring Management* pattern in order to identify the interactions among the wearable devices and the physicians for visualization and anomaly detection; the *Semantic Knowledge Mediator* pattern in order to manage data heterogeneity and enable the collaboration among the wearable devices and the physicians; and the *Big Data Stream Detection* pattern in order to manage data heterogeneity and velocity. Figure portrays the interaction among the different system components. We developed a collaborative semantic web platform, based on Semantic MediWiki[2] (SMW) and Apache Fuseki[3],for the semantic annotation of wearable devices based on the Wearable Healthcare Ontology *(WH_O)* [23]. This platform represents an implementation of the *Semantic Knowledge Mediator pattern*.

Once the wearable devices are semantically annotated, they will be configured to send asynchronous data to Apache Kafka[4], which is a scalable and high-throughput distributed *publish/subscribe* messaging system. We propose to use Apache Storm[5] to consume the data from the Kafka topics based on Kafka-storm connector. Thus, we developed a topology that encodes the data workflow processing using three bolts that normalize the data, store it in distributed cluster to be analyzed, and detect problems at near real-time (instantiation within the *Big Data Stream Detection pattern*).

[2] SMW: https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki
[3] https://jena.apache.org/documentation/serving_data/
[4] http://kafka.apache.org/
[5] http://storm.apache.org/

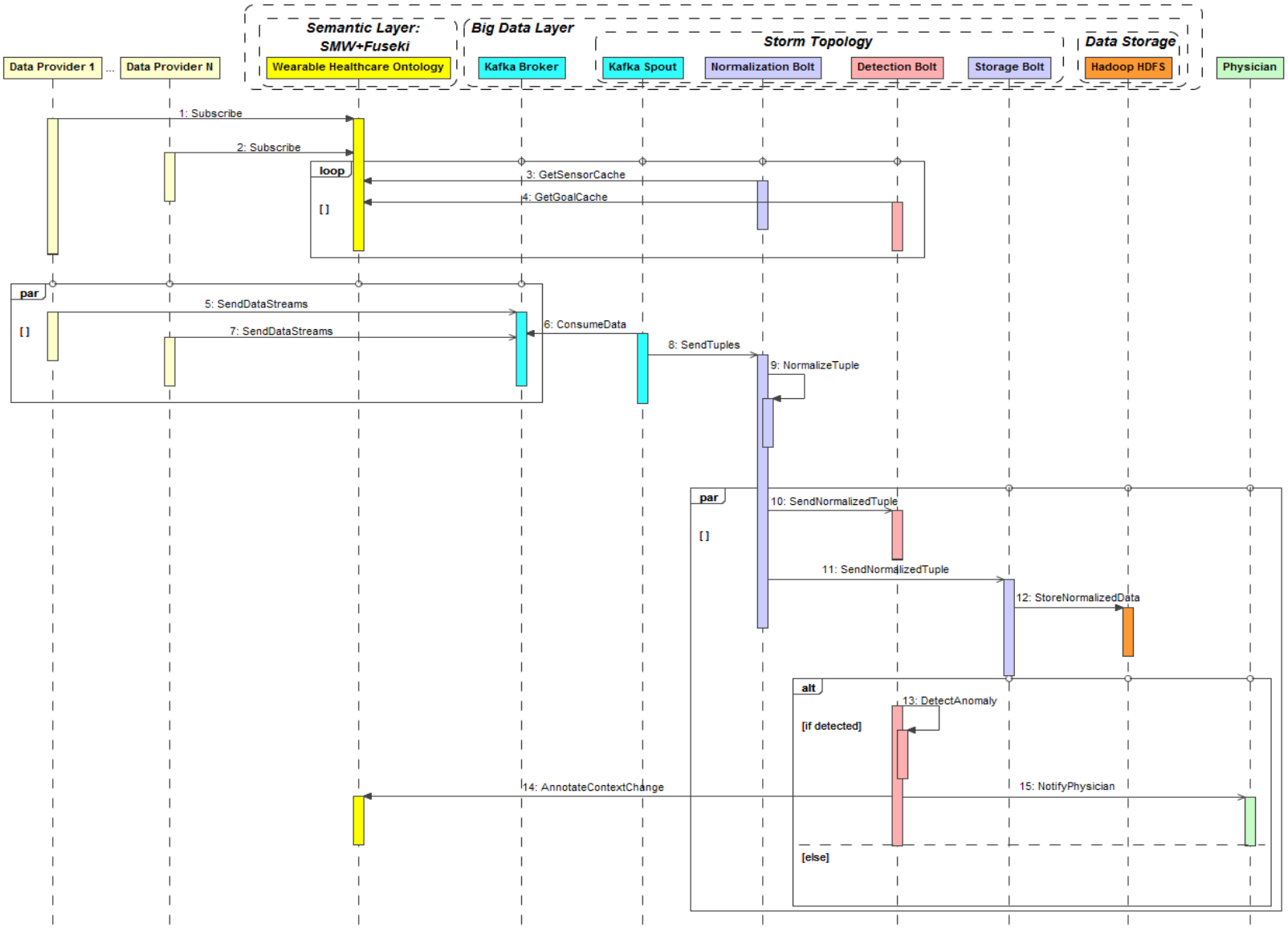

Figure 5. Cognitive Monitoring System sequence diagram (functional requirements management)

The **Normalization bolt** refers to the *Sensor Cache* to extract the unit, and decide if the measured value should be converted or not. The *Sensor Cache* contains information about the active wearable' unit and delimiter retrieved from Fuseki through enacting the query presented in Figure (Query1) which is based on WH_O [23]. Once data is normalized, it will be sent in parallel to the Hadoop's HDFS system[6], and to the Detection bolt. The harmonized data stored in HDFS can be visualized, and also used by advanced analytic services implemented using Apache Spark and MapReduce framework. It is worth to mention that wearable data are not stored in the WH_O. Only the wearable characteristics and the semantic representation of data, as well as its meaning are stored in WH_O in order to cover scalability problems. The **Detection bolt** refers to the patient *Goal Cache* in order to detect personalized anomalies according to the fixed patient target goal. The Goal Cache is dynamically up-to-date through enacting the SPARQL query presented in Figure 6 (Query2) based on WH_O. If a problem is detected, it will be populated in the SMW platform, and a notification is sent to the physician. In the next section, we evaluate the performance and scalability of the proposed cognitive monitoring system.

Query1. Retrieve the units and delimiter of active wearables

```
Select distinct ?wid ?unit ?delim where
{?wc prop:IsSpecificTo ?w.
?w prop:HasID ?wid.
?w prop:Implements ?ms.
?ms prop:HasUnit ?unit.
?ms prop:HasDelimiter ?delim.
?ms prop:MonitorParameter ?param.
?param prop:HasName ?paramName.
?wc prop:HasStartDate ?sd.
?wc prop:HasEndDate ?ed.
Filter (?paramName ="topic").#such as Blood Sugar
Filter (?sd<= "date"). # date when updating the cache
Filter (?ed>="date").}
```

Query2. Retrieve the list of patients with their target goal

```
Select ?ID ?m where
{?p prop:HasIndividualizedTargetGoal ?goal.
?goal prop:IsRelatedTo ?param.
?goal prop:LessThan ?m.
?param prop:HasName ?lab.
Filter (?lab="ParamName").# for example, Blood Sugar
?p prop:HasPatientID ?ID.}
```

Figure 6. SPARQL queries used within the bolts

6. Evaluation

[6] http://hadoop.apache.org/

The objective of this evaluation is to highlight the ability of the system to manage the data variety (heterogeneity), velocity and volume, as well as its scalability and performance based on the instantiation of the proposed patterns. Thus, for this evaluation, we consider different patients with type 2 diabetes equipped with heterogeneous wearable devices measuring the blood sugar parameter. These wearable devices are connected and characterized following the WH_O. In turns, the patient medical conditions and target goals are also updated in the system. We simulated the data stemming from wearable devices. Some of data is expressed in mg/dL, and the rest in mmol/L. We created different data structure and add some fields such as the wearable ID, the delimiter (":" or ";") based on the wearable devices, etc. Thus, data is heterogeneous. Then, we duplicated the data to create different scenarios presented in Table 1 in order to highlight the velocity and volume. We used the following cloud server having these IT resources: 8 Intel(R) Xeon(R) CPU D-1521@2.40GHz, 32GB of memory, and 1.77TiB of disk storage (Config1).

To fully leverage the distribution and parallelism in the cloud, Figure proposes an instantiation of the *Management Process Multi-tenant pattern* when deploying the cognitive monitoring system. We used the ArchiMate standard to represent the business, application and technology layers. Following this deployment, we run the storm topology (data processing) in production cluster with two different configurations: (i) *1 worker* where only one supervisor is selected to run the topology without parallelism; and (ii) *2 workers* where the parallelism is supported in both supervisors. We evaluated both configurations' performance through running the different testing scenarios. The measurement of the processing time of each scenario is illustrated in Table 1, while a visual representation of the performance is represented in Figure .

We noted that despite the heterogeneity of the received data stream and their huge amount, our system is able to successfully process all the incoming data. The parallelism contributes to reducing the processing time, especially when dealing with a big number of data streams. For instance, when

processing 1,200,000 data streams, with the parallelism we gained 53 seconds, which is an important value for near-real time applications.

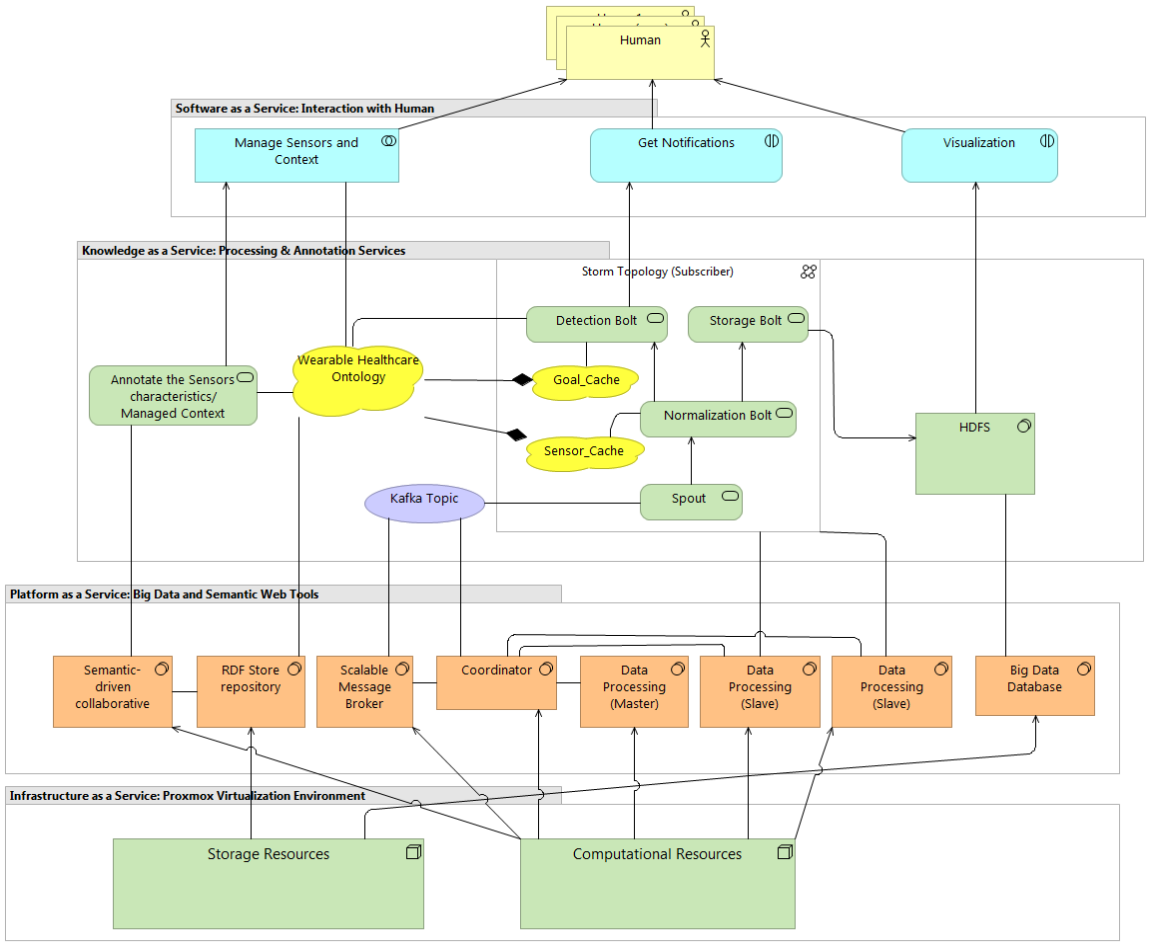

Figure 7. Cognitive Monitoring System Deployment on the Cloud

After that, we evaluated the impact of IT resources on the system performance. Thus, we increased the allocated CPU units of the data processing and of the broker in a new more powerful cloud server (Config2). Then, we evaluated again the same testing scenarios and compared the system performance when implementing 2 workers in two different cloud infrastructures. Figure and Table 1 portray the obtained results. We noted that increasing the CPU contributes to reducing the processing time, especially when processing a big number of incoming data streams. For instance, when receiving 1,200,000 data streams, with the second cloud infrastructure and IT

resources configuration, we gained up to 30 seconds, which is an important gain in real-time systems.

Table 1. Response Time Measurements on the Cloud

Number of Concurrent Streams	***1 Worker Cloud Config1***	***2 Workers Cloud Config1***	***2 Workers Cloud Config2***
1000	1.497	1.251	1.285
4000	4.174	4.081	4.08
8000	6.485	6.18	5.247
40000	17.281	15.668	14.623
80000	28.682	27.41	27.247
120000	43.74	39.775	34.861
400000	132.685	118.348	111.443
600000	194.636	170.841	163.003
1200000	390.2	337.745	301.639

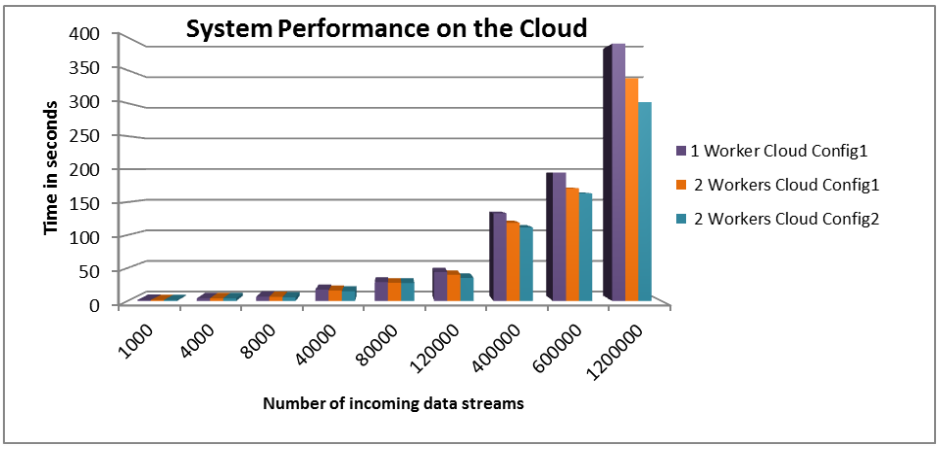

Figure 8. System Performance on the Cloud

Consequently, choosing the appropriate cloud infrastructure and system architecture deployment depends on the system requirement and the availability of IT resources. For instance, if the system is used to manage patients with chronic disease at an early stage to prevent the patient health degradation, 30 seconds that we gained when deploying the system in the second cloud infrastructure is not considered as a delay in the first infrastructure. In this case, the function cost will be used to select the right configuration. However, if the system is conceived for managing patients with severe clinical conditions, near-real time detection is mandatory. In this case, the distributed architecture deployed on the second cloud server is more adequate.

7. Conclusion

The sheer scale and design complexity of IoT-based systems lead us to propose a model driven-methodology that adopts different refinement levels that allows incrementally incorporating the system requirements through modeling the interactions among management processes and knowledge sources. Within this methodology, a set of autonomic cognitive design patterns have been proposed to drive the architect providing flexible smart IoT-based systems. These patterns have been proposed to deal with the system functional requirements through delineating the coordination of the management processes and their interactions with human; and the system nonfunctional requirements through ensuring the semantic integration, and managing big data and scalability issues. We demonstrated the efficiency of the proposed patterns through developing a cognitive monitoring system for managing the patient health based on heterogeneous wearables. Finally, we evaluated the ability of the system to manage the IoT data variety, velocity and volume as well as the system performance in terms of response time and scalability management.

Our future work consists in elaborating a set of transformation rules that automatically refine the IoT-based system design based on a specification of the system requirements, and also working on providing optimal configuration of IT resources for better response time.

Acknowledgment

This research is entirely funded by the National Research Fund (FNR) of Luxembourg under the AFR project.

References

[1] A. Noronha, R. Moriarty, K. O'Connell, N Villa, "Attaining IoT Value: How To Move from Connecting Things to Capturing Insights," Cisco, 2014.

[2] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton and T. Razafindralambo, "A survey on facilities for experimental internet of things research," in IEEE Communications Magazine, vol. 49, no. 11, pp. 58-67, November 2011.

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

[4] M. Ben Alaya, Y. Banouar, T. Monteil, C. Chassot, K. Drira "OM2M: Extensible ETSI-compliant M2M service platform with self-configuration capability," Procedia Computer Science, vol. 32, pp. 1079-1086, 2014.
[5] E. Mezghani, R. Ben Halima, K.Drira, "DRAAS: Dynamically Reconfigurable Architecture for Autonomic Services," in Web Services Foundations, A. Bouguettaya, et al., Eds., New York, NY: Springer New York, 2014, pp. 483-505.
[6] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," ACM Computing Surveys (CSUR), vol. 40, p. 7, 2008.
[7] C. Klein, R. Schmid, C. Leuxner, W. Sitou and B. Spanfelner, "A Survey of Context Adaptation in Autonomic Computing," Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08), Gosier, 2008, pp. 106-111.
[8] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," IBM systems Journal, vol. 42, pp. 5-18, 2003.
[9] R. De Lemos, et al., Software engineering for self-adaptive systems: A second research roadmap: Springer, 2013.
[10] V. Foteinos, D. Kelaidonis, G. Poulios, P. Vlacheas, V. Stavroulaki and P. Demestichas, "Cognitive Management for the Internet of Things: A Framework for Enabling Autonomous Applications," in IEEE Vehicular Technology Magazine, vol. 8, no. 4, pp. 90-99, Dec. 2013.
[11] P. Dazzi, F. Nidito and M. Pasquali "New perspectives in autonomic design patterns for stream-classification-systems," in Proceedings of the 2007 workshop on Automating service quality: Held at the International Conference on Automated Software Engineering (ASE), 2007, pp. 34-37.
[12] A. Al-Shishtawy, V. Vlassov, P. Brand and S. Haridi, "A Design Methodology for Self-Management in Distributed Environments," 2009 International Conference on Computational Science and Engineering, Vancouver, BC, 2009, pp. 430-436.
[13] D. Weyns, et al., "On patterns for decentralized control in self-adaptive systems," in Software Engineering for Self-Adaptive Systems II: Springer, 2013, pp. 76-107.
[14] F. A. de Oliveira Jr., R. Sharrock and T. Ledoux., "Synchronization of multiple autonomic control loops: Application to cloud computing," in Coordination Models and Languages, 2012, pp. 29-43.
[15] C. Vidal, C. Fernández-Sánchez, J. Díaz and J. Pérez, "A model-driven engineering process for autonomic sensor-actuator networks," International Journal of Distributed Sensor Networks, vol. 2015, p. 18, 2015.
[16] Y. Wang, "Special Issue on Cognitive Computing, On Abstract Intelligence," International Journal of Software Science and Computational Intelligence, vol. 1, 2009.
[17] P. Lalanda, "Two complementary patterns to build multi-expert systems," in Pattern Languages of Programs, 1997.
[18] E. Gamma, Design patterns: elements of reusable object-oriented software: Pearson Education India, 1995.
[19] P.Th. Eugster, P.A. Felber, R. Guerraoui, and A-M. Kermarrec., "The many faces of publish/subscribe," ACM Computing Surveys (CSUR), vol. 35, pp. 114-131, 2003.
[20] F. Buschmann, M. Stal, H. Rohnert, P. Sommerlad and R. Meunier "A system of patterns: Pattern-oriented software architecture." (1996).
[21] R. C. Atkinson and R. M. Shiffrin, "Human memory: A proposed system and its control processes," The psychology of learning and motivation, vol. 2, pp. 89-195, 1968.
[22] E. Tulving, "How many memory systems are there?," American psychologist, vol. 40, p. 385, 1985.
[23] E. Mezghani, E. Exposito, K. Drira, M. Da Silveira, C. Pruski, "A Semantic Big Data Platform for Integrating Heterogeneous Wearable Data in Healthcare," Journal of medical systems, vol. 39, pp. 1-8, 2015;