Architecting Emergent Configurations in the Internet of Things

Fahed Alkhabbas, Romina Spalazzese and Paul Davidsson
Department of Computer Science and Media Technology
Internet of Things and People Research Center, Malmö University
Malmö, Sweden
{*fahed.alkhabbas, romina.spalazzese, paul.davidsson*}*@mah.se*

***Abstract*—The Internet of Things (IoT) has a great potential to change our lives. Billions of heterogeneous, distributed, intelligent, and sometimes mobile devices, will be connected and offer new types of applications and ways to interact. The dynamic environment of the IoT, the involvement of the human in the loop, and the runtime interactions among devices and applications, put additional requirements on the systems' architecture. In this paper, we use the Emergent Configurations (ECs) concept as a way to engineer IoT systems and propose an architecture for ECs. More specifically, we discuss (i) how connected devices and applications form ECs to achieve users goals and (ii) how applications are run and adapted in response to runtime context changes including, e.g., the sudden unavailability of devices, by exploiting the Smart Meeting Room case.**

***Keywords*-Internet of Things, Emergent Configurations, Architecture, Self-adaptation.**

I. Introduction

In the next years, billions of heterogeneous, distributed, and intelligent things will be connected and enable new types of applications and ways to interact. The Internet of Things (IoT) will have a social and societal impact both on our way of working and living. Also, the IoT will open a full range of new possibilities, e.g., new business models and ecosystems.

To be able to benefit of the advantages that the IoT will bring, a number of engineering challenges have to be tackled relating to heterogeneity, adaptability and interoperability to mention few [1], [11]. A key for addressing these engineering challenges is the system architecture, both at design and at runtime. Requirements arise due to the dynamic nature of the IoT and of its constituents. Further, the involvement of the human in the loop imposes additional requirements on the architecture since it is hard, if not impossible, to predict and model all user activities in such dynamic environment.

The above mentioned challenges have been addressed to various degrees in previous work in several different research areas. For instance, the IoT-A project (http://www.iot-a.eu), according to [2], investigated an architectural reference model for the IoT [8]. The IoT-A is based on a service-oriented approach where devices, data, and interfaces, are abstracted and implemented as services. These services are composed into complex processes to support business needs. The composition is based on processes modelled by experts and enacted by execution engines.

Kramer and Magee [9] proposed a three-layer architecture to support automatic (re)configuration of components to achieve system goals. The architecture assumes predefining a set of plans which satisfy constraints for several potential system states. When new goals are introduced, new plans could be generated in a time-consuming process [5]. In the context of the IoT, the number of connected objects is supposed to be very large and dynamic and involving humans make the situation more complex. Then, it is unfeasible to foresee/define plans for a wide range of potential situations. The Aura architecture [12], provides a solution for the achieving user tasks in mobile contexts. In addition, Aura provides self-healing and self-optimization capabilities [13]. However, the approach does not seem to support the notion of an overall plan which represents the execution of coordinated tasks to achieve a more abstract user goals. We envision that the user activities are goal-directed where a goal can also be a simple task. For instance, the user goal "deliver presentation" might include several concrete tasks such as "find and select display" and "connect display media".

The Service-Oriented Architecture (SOA) is considered a promising approach to achieve interoperability among IoT heterogeneous objects and enable collaboration to achieve goals [1], [11]. However, service discovery, composition, and access require high communication and computation cost. Furthermore, the SOA approach is designed to connect complex but static enterprise services [6]. From this perspective, and due to the large scale of the IoT, the SOA approach needs further development to fulfil the needs. Several SOA-based IoT solutions have been suggested [6], [4]. MobIoT [7] is an approach which provides efficient service discovery, composition, and access in heterogeneous, dynamic, and mobile IoT contexts. The MobIoT approach revisits the standard SOA approach by providing probabilistic registration, look-up and thing-based composition based on comprehensive ontologies. However, the approach does not support runtime interaction with users to specify their goals. In addition, it still needs proper evaluation from the scalability perspectives when the number of unique registered services is very large as expected to be in the IoT. SIA [14] is a SOA which supports abstracting objects with embedded software as services. The architecture supports

heterogeneous hardware, software and communication protocols among embedded systems. However, the architecture seems not to support self-adaptation due to runtime changes of the user's requirements and does not support runtime interaction with users to understand their goals and preferences. To overcome the shortcomings of existing approaches, while answering the needs of IoT systems, in this paper we use the concept of Emergent Configurations (ECs) as the basis for a novel approach for engineering IoT systems (and compliant with IoT-A). An EC is defined as a set of things with their functionalities and services that connect and cooperate temporarily to achieve a goal [3]. A thing is any (smart) connected object or device with its functionalities and services or applications. In this paper, we focus on complex cases that involve (human and/or non-human) users, goals and require adaptations either due to dynamic changes in the environment or due to evolving user goals. However, the EC concept also addresses cases which do not involve (non-)human users and/or evolving goals at runtime, e.g., [3]. In this work, we consider the user goal as a driver of ECs. User activities are inherently situated, stochastic and hard to predict. This makes both ECs and their context dynamic and very difficult to capture through traditional procedural process modelling. In this paper, we introduce a goal-oriented EC approach and present an architecture of IoT systems based on the IoT-A reference model (Sections III) by using the case of the Smart Meeting Room (Section II).

II. The Smart Meeting Room Case

Lorraine books a smart room for a meeting where she will deliver a presentation about climate changes. In this room, almost every object is connected for e.g., lights, curtains and display media, as well as a set of sensors for detecting e.g., temperature, light, presence. During the meeting, Lorraine uses her smart phone to express her goal to "deliver presentation". The goal is interpreted and an EC is formed to achieve it. Potential display media, like projectors and screens, are highlighted. Using the smart phone, Lorraine chooses the projector and specifies the source of the slides to be her laptop. The available sensors detect that there is too much sunlight to properly see the presentation. Therefore, the curtains are automatically closed. In addition, Lorraine defines a service that simulates the effect of the climate changes in the coming ten years by increasing the heat in the room. To point to the slides, Lorraine brings a pointer configured as a public resource from another room. Thus, it is automatically detected and configured. During the presentation, the projector turns off suddenly. The failure is automatically detected and Lorraine is proposed to continue her presentation on the screens installed in the room. Since different display media require different light conditions, Lorraine is proposed to open the curtains, when the presentation is resumed on the screens.

III. Engineering Emergent Configurations

In this section, we illustrate the concept of ECs through the smart meeting room case. We introduce the logical view of ECs and the processes of forming and managing them.

A. Emergent Configurations logical view

Figure 1 illustrates the *Logical layer* of ECs (note that this is a logical representation, i.e., it might be realized both in a centralized or a decentralized way) and the *Physical layer*. An EC is formed, operated and managed through the collaboration, interaction and coordination of the following components: (i) *User Agent* (UA): is used by the user to express her/his goals and it interacts with the ECM at runtime. (ii) *Emergent Configuration Manager* (ECM): is responsible to form, operate and manage ECs to satisfy user goals. (iii) *Device Manager* (DM): registers the IoT things, i.e., (smart) connected devices and objects with their functionalities, that are available in a certain environment. In addition, it continuously monitors the availability of these things and informs the ECM when things are discovered or become unreachable. (iv) *Set of IoT things*: following indications of the ECM, they connect and cooperate to realize the EC.

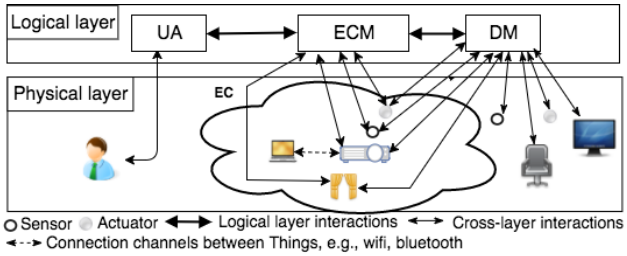

Figure 1. Logical and physical layers of ECs

The process of achieving a user goal through enacting ECs consists of two subprocesses: (1) Form the EC as illustrated in Figure 2 and explained in the following; (2) Manage the EC by continuously executing the MAPE-K loop as illustrated in Figure 3 and explained in Section III-B. We envision that the subprocess (1) is achieved through a semantic-enabled approach. This phase consists of several subtasks. First, the user expresses her/his goal through one (or more) of the available IoT things that we refer as the user agent, e.g., the user's smartphone in the smart meeting room scenario. It is also possible to switch the user agent at the runtime, e.g. from smartphone to iPad, based on the user selection. In the next step, the ECM interprets the expressed goal leveraging semantic technologies, e.g., ontologies. To better define the goal, the goal location boundaries needs to be specified, e.g. a certain meeting room. For specifying the location boundaries the ECM analyses the user location in correlation with the expressed goal. For instance, the ECM might propose to the user to deliver the presentation in the meeting room where he/she is. Then, the ECM interacts with

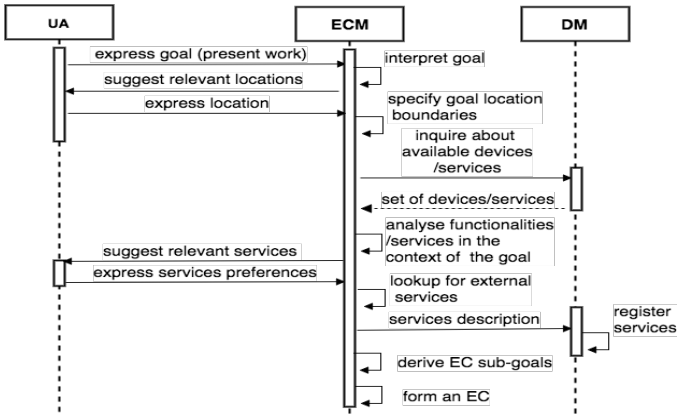

Figure 2. Sequence diagram of forming an EC

and inquires the DM about the things located within the meeting room such as projectors, screens, curtains, etc. After that, the ECM analyses their functionalities to understand how they can contribute to achieving the user goal. e.g., the ECM understands that a projector can display media. By exploiting semantic properties and reasoning, the relations between these abstract functionalities and the user goal are inferred.

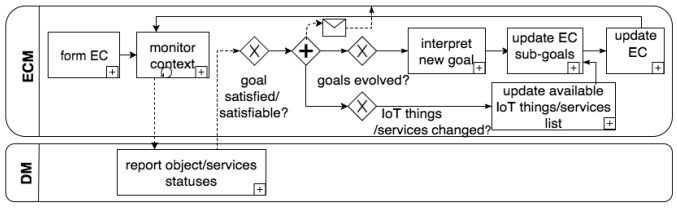

Figure 3. Process to manage ECs

The user may specify some external services to be used during his/her activity. These services are either required to achieve the goal or they contribute to achieving the goal with higher quality. For instance, in the smart meeting room scenario, simulating the climate changes by increasing the room temperature indirectly supports the user in delivering her presentation. An example of a required service to achieve the goal is the service responsible to download the presentation from the cloud to the user's laptop. After that, the ECM derives the EC subgoals. This task aims to derive atomic goals by decomposing the user goal into subgoals. Atomic goals, as for instance set light level, are achievable by things. At this stage, the ECM relates the analyses of the functionalities of the available things to the derived atomic goals. This step also includes defining the order to perform the subgoals and the dependencies among them. For instance, the light level is detected before the curtains are closed/opened. These are examples of subgoals that contribute to the quality of the goal "deliver presentation". Alternative ways to achieve the goal are also generated at this step and are continuously updated at runtime. If the goal is not decomposable or there are no things to achieve all required atomic goals, the user agent is informed that the goal is unachievable in the current context. If the user goal is achievable, the ECM forms an EC based on the derived subgoal model and instructs things to operate. For instance, the ECM automatically wakes up the projector and instructs it to receive slides from the laptop of the user. As ECs operate in dynamic and unpredictable environment, the ECs design has to be resilient and enable runtime adaptations in response context changes [10]. In the second subprocess, to support adaptations in ECs, we envision a self-adaptive architecture implementing the MAPE-K loop. For instance, in the smart meeting room, the screens are automatically proposed as an alternative to continue the presentation.

B. ECs Architecture

The process management layer in the IoT-A functional view is composed of two components: a process modelling and a process execution [2]. We refine it by providing a concrete architecture for the IoT-A process management layer, illustrated in Figure 4 and described below, and a (user) goal-directed approach. The components, illustrated in Figure 4, represent the core components of the ECM.

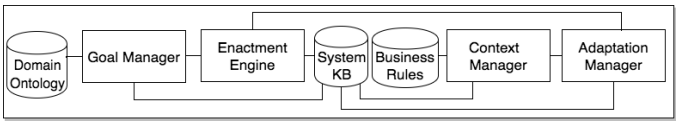

Figure 4. EC architecture refining the IoT-A process management layer

The *Goal Manager* is responsible to: 1) Interpret the user goal, i.e., reason about and transform the user goal to a concrete representation. 2) Specify the goal location boundaries, i.e., transform the expressed location to spatial knowledge and stores it in the System KB (see later). 3) Analyse available IoT objects/services in the context of the goal and map the goal to the set of functionalities provided by the set of available objects/services. This requires vendors to expose objects functionalities through APIs. This phase involves the knowledge defined in the Domain Ontology (see later) and the System KB. 4) Suggest relevant services to the user. 5) Lookup for external services, based on the expressed service preferences, and send their description to the DM. 6) Derive the EC subgoals model, i.e., atomic goals by decomposing the user goals. The Goal Manager then connects IoT devices to relevant atomic goals by semantically mapping them to proper functionalities. This step involves specifying the plan needed to achieve the main goal -for instance, subgoals achievement order and dependencies. Plans are generated at runtime and actions triggered by events. To properly enable the Goal Manager, we are investigating supporting solutions in areas such as goal-oriented engineering, semantic-enabled interactions and ontological relationships between concepts and functional-

ities. The *Adaptation Manager* is responsible to execute the MAPE-K loop. Mainly, it a) Monitors changes in the context. These changes include evolvement of the users goal and the effect of dynamically discovering new devices or losing them towards achieving the goal. b) Analyses the effect of the detected changes towards satisfying the user's goal. c) Plans changes to the EC subgoal model and d) Executes the plan by communicating the instructions to the Enactment Engine. To properly enable the Adaptation Manager, we are investigating supporting solutions in areas such as Self-managed Systems, Ubiquitous Computing and other relevant research areas. The *Context Manager*: is responsible for maintaining the ECs context. This involves detecting dynamic events in the environment and updating the system KB. It maintains the primary context (raw data) through the available sensors/services. The secondary context is derived by processing the primary context. In addition, it is responsible to infer new business rules based on the context. To properly enable this component, we are investigating supporting solutions in areas such as in Context-aware Systems, Rule-based Systems and inference engines. The *Enactment Engine* is responsible to enact ECs. In other words, it instructs IoT things to coordinate and collaborate their activities to execute the plan. To properly enable this component, we are investigating supporting solutions in areas such as Multi-agent Systems, Service-Oriented Architectures and Business Processes enactment engines. The *Business Rules* component is a repository for domain related rules and inferred rules based on context changes. The *Domain Ontology* contains a taxonomy of semantic knowledge about domains like "Smart office". This knowledge-base is used to derive ECs subgoals and the relations between goals and the available functionalities and services. The *System KB* is the container of the context of the ECs.

IV. Conclusion and Future Work

ECs have the potential to be a powerful means for realizing complex IoT-based systems. In this paper, we presented an initial architecture for ECs and a goal-oriented approach. As future work, we will continue investigating how to realise the proposed architecture also taking into consideration non-functional aspects. Moreover, we plan investigations about approaches and mechanisms to support evolving user goals and handle situations which involve multiple users with conflicting goals. Further investigations will be also devoted to the correlation between ECs and areas such as: goal-oriented analyses, semantics technologies, team formation and management, dynamic service discovery and invocation, context-awareness and self-adaptation.

Acknowledgment

This work is partially financed by the Knowledge Foundation through the Internet of Things and People research profile (Malmö University, Sweden).

References

[1] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] M. Bauer, M. Boussard, N. Bui, J. De Loof, C. Magerkurth, S. Meissner, A. Nettsträter, J. Stefa, M. Thoma, and J.W. Walewski. *IoT Reference Architecture*, pages 163–211. Springer, Berlin, Heidelberg, 2013.

[3] F. Ciccozzi and R. Spalazzese. MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering. In *Intelligent Distributed Computing X: Proc. of the 10th Intern. Symp. on Intelligent Distrib. Computing*, pages 67–76. Springer, 2017.

[4] M. Eisenhauer, P. Rosengren, and P. Antolin. HYDRA: A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems. In *The Internet of Things*, pages 367–373. Springer, 2010.

[5] E. Gat. On Three-Layer Architectures. *Artificial intelligence and mobile robots*, pages 195–210, 1998.

[6] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *IEEE Transactions on Services Computing*, 3(3):223–235, 2010.

[7] S. Hachem, A. Pathak, and V. Issarny. Service-Oriented Middleware for the Mobile Internet of Things: A Scalable Solution. In *IEEE GLOBECOM: Global Communications Conference (Accepted)*, 2014.

[8] IEC ISO. IEEE: 42010: 2011 Systems and Software Engineering Architecture Description. *International Standard*, 2011.

[9] J. Kramer and J. Magee. Self-Managed Systems: an Architectural Challenge. In *Future of Software Engineering*, pages 259–268. IEEE, 2007.

[10] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, et al. Software Engineering for Self-Adaptive Systems: A Second Research Roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.

[11] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[12] J. P. Sousa and D. Garlan. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Software Architecture*, pages 29–43. Springer, 2002.

[13] J. P. Sousa, V. Poladian, D. Garlan, B. Schmerl, and M. Shaw. Task-based adaptation for ubiquitous computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(3):328–340, 2006.

[14] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker L. M., De Souza, and V. Trifa. SOA-Based Integration of the Internet of Things in Enterprise Services. In *Web Services, IEEE International Conference on*, pages 968–975. IEEE, 2009.