Towards Control as a Service models and architecture for the Industry 4.0

Minhu Lyu

▶ **To cite this version:**

Minhu Lyu. Towards Control as a Service models and architecture for the Industry 4.0. Artificial Intelligence [cs.AI]. Université de Lyon, 2020. English. NNT : 2020LYSEI048 . tel-03127288

HAL Id: tel-03127288
https://theses.hal.science/tel-03127288

Submitted on 1 Feb 2021

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.

N°d'ordre NNT : 2020LYSEI048

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
INSA-Lyon

Ecole Doctorale N° 512
INFOMATHS

Spécialité/ discipline de doctorat :
Informatique

Soutenue publiquement le 03/07/2020, par :
Minhu LYU

Towards Control as a Service Model and Architecture for the Industry 4.0

Devant le jury composé de :

Mme Christine VERDIER,	Professeur des Universités, Université de Grenoble, Présidente
M. Bernard GRABOT,	Professeur des Universités, ENIT - INP Toulouse, Rapporteur
M. Ernesto EXPOSITO,	Professeur des Universités, Université de Pau et des Pays de l'Adour, Rapporteur
Mme Frédérique BIENNIER,	Professeur des Universités, INSA-Lyon, Co-Directrice de thèse
Mme Parisa GHODOUS,	Professeur des Universités, Université Claude Bernard Lyon 1, Co-directrice de thèse

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** http://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr INSA : R. GOURDON	**M. Stéphane DANIELE** Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX directeur@edchimie-lyon.fr
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** http://edeea.ec-lyon.fr Sec. : M.C. HAVGOUDOUKIAN ecole-doctorale.eea@ec-lyon.fr	**M. Gérard SCORLETTI** École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 gerard.scorletti@ec-lyon.fr
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES secretariat.e2m2@univ-lyon1.fr	**M. Philippe NORMAND** UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX philippe.normand@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://www.ediss-lyon.fr Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Curien - 3ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tel : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tel : 04.72.44.83.69 hamamache.kheddouci@univ-lyon1.fr
Matériaux	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction ed.materiaux@insa-lyon.fr	**M. Jean-Yves BUFFIÈRE** INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX jocelyn.bonjour@insa-lyon.fr
ScSo	**ScSo*** http://ed483.univ-lyon2.fr Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 veronique.cervantes@univ-lyon2.fr	**M. Christian MONTES** Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Abstract

To fit the renewed globalized economic environment, enterprises, and mostly SMEs, have to develop new networked and collaborative strategies, focusing on networked value creation (instead of the classical value chain vision), fitting the blue ocean context for innovative products and service development. Even if collaborative organizations have been studied for decades, the closer connection of information systems involved by the so-called "Industry 4.0" developed by leading industries in Europe, US and Asia requires to set new IT models to support agile and evolving collaborative Business Process (BP) enactment, integrating both traditional Information Systems (IS) and production control processes. By now, these product/service ecosystems are mostly supported by software services, which span multiple organizations and providers, and on multiple cloud-based execution environments, increasing the call for openness, agility, interoperability and trust for both production and Information System organization. These requirements are well supported by SOA, Web 2.0 and XaaS technologies for Information Systems. Taking advantage of IoT, services and Cloud technologies, the development of Cloud of Things (CoT) changes the way control application are engineered and developed moving from a dedicated design and development of control applications to a Control as a Service vision. This vision requires developing a new architecture to connect physical and logical objects as well as integrating basic control patterns to organize a consistent control service orchestration. To fit this challenge, we propose a multi-layer Control as a Service architecture to describe control systems in a holistic way. Our Control service model is built according to an event-driven orchestration strategy. Thanks to the integration of a context manager, analyzing continuously the system environment as well as the control system behavior, these context-aware control services can be deployed.

Keywords: Control Service, Control as a Service, control ontology, Service Oriented Control Architecture, Industry 4.0

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Résumé

Pour s'adapter au contexte de l'économie globalisée, les entreprises, et principalement les PMEs doivent développer de nouvelles stratégies de collaboration. Ces stratégies sont axées sur la création de valeur en réseau en remplacement de l'organisation classique de chaîne de valeur, s'adaptant ainsi au modèle dit de « Blue Ocean » qui conduit au développement de produits et services innovants. Bien que les organisations collaboratives aient été étudiées depuis des décennies, l'Industrie 4.0, actuellement largement développée par les principales industries en Europe, Amérique ou Asie, impose une intégration plus poussée des Systèmes d'Information pour y inclure des processus opérationnels collaboratifs, tant pour les activités administratives que pour la production. En outre, ces processus doivent être adaptatifs pour s'adapter au contexte. Actuellement, ces écosystèmes de produits/services sont principalement implémentés par des services logiciels, déployés sur le Cloud et utilisables par différentes organisations. Pour répondre aux besoins d'agilité, d'ouverture, d'interopérabilité et de confiance, ces services utilisent largement les architectures orientées services, les technologies Web2.0 et XaaS. Tirant parti de l'IoT, des technologies de services et du Cloud, le Cloud des objets (Cloud of Things ou CoT) change la manière de concevoir des applications de contrôle, passant d'une ingénierie traditionnelle à une vision de composition de services. Cette vision suppose de définir une nouvelle architecture pour connecter les objets physiques, leur double virtuel et intégrer des patrons de contrôle pour composer et orchestrer ces services pour répondre aux besoins. Pour répondre à ce défi, nous proposons uns architecture multi-niveaux de Control as a Service permettant de décrire les systèmes de contrôle selon une vision holistique. Notre modèle de service de contrôle est construit pour permettre une exécution pilotée par les événements. L'intégration d'un gestionnaire de contexte, analysant continuellement l'environnement d'exécution et le comportement du système, permet d'assurer le déploiement de services de contrôle contextualisables.

Mots cléfs: Control as a Service, modèle de service, Industrie 4.0, ontologie pour le contrôle, orchestration pilotée par les événements

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Acknowledgment

Without much support and encouragement from professors, friends, SOC team, family and even our country, I would not have achieved my Ph.D degree. Therefore, I would like to thank them.

Firstly, I would like to thank my doctoral advisors Professor Frédérique Biennier and Professor Parisa Ghodous for their guidance of my work. They provides a lot help on how to make a progress during my Ph.D career. A special thank will be given to Professor Frédérique Biennier as she has always been patient, responsible, efficient and kind.

Then, I am kindly grateful to Professor Bernard Grabot and Professor Ernesto Exposito for their review and evaluation, Professor Christine Verdier for her examination. Meanwhile, my thanks go to my colleges and friends, Dr Xiaoyang Zhu, Dr Faiza Loukil, Dr Hind Benfenatki, Ph.D candidate Jingya, Yuan, for their collaboration and help.

Furthermore, I do appreciate my country, more specifically, Chinese Scholarship Council (CSC), for their financial support So that I could study in INSA de Lyon, France, in the past 44 months.

Finally, I owe my deepest gratitude to my family. My parents provides a stably warm and sweet environment throughout my life. I feel much obliged to my wife (Yanpei Zhou) as separation is not easy for any couple.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Contents

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

List of Figures

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

List of Tables

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Chapter 1

Introduction

Contents

1.1 Context

To fit the renewed globalized economic environment, enterprises, and mostly SMEs, have to develop new networked and collaborative strategies, focusing on networked value creation (instead of the classical value chain vision), fitting the blue ocean context for innovative products and service development [63]. Moreover, the fast development of Internet of Things, FabLabs and new standalone manufacturing means (such as 3D printers) allow to bring production means closer to the consumer. This context challenges for developing new value chain organization and new digitized production model leading to the so-called “Industry 4.0" developed by leading industries in Europe, US and Asia. This 4^{th} industrial revolution takes advantage of the Internet of Services, Internet of Things, Cyber-Physic system models to organize smart factories. These new production organization challenges IT development to provide agile and on-demand collaborative process support as well as renewing production operation control on one hand by integrating production data in the information system and on the other hand by setting distributed control process, interacting with their environment thanks to smart devices.

The large adoption of web-service technologies extended to IoT devices as well as the Cloud-based Everything as a Service (XaaS) model, provide a strong basis to set dis-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

tributed systems, increasing their flexibility and agility to achieve the Industry 4.0 vision. Nevertheless, these models and technologies are mostly focused on the way IoT devices can be interconnected over the Web to exchange data, without paying attention to their physical characteristics nor paying attention to the way the “smart control process" is organized. To overcome this limit, dedicated researches have been conducted to set Sensing as a Service (SenaaS) or Object as a Service models, extending the traditional web service vision integrating only “logical" functional and non functional properties to enriched IoT service model integrating both logical and physical properties. Paying attention to the core control process itself, traditional automation provides a strong background, including several control patterns, to design and develop complex control systems. Nevertheless, these systems are set using “hard connections" between components. Consequently, they lack of agility and flexibility and do not support context-aware control applications. functional properties to enriched IoT service model integrating both logical and physical properties.

To sum-up, while these models are more and more used to manage and deploy smart objects and provide corresponding services, the way of implementing a traditional control system or Networked Control System (NCS) has to move from embedded control to cloud-based control. Integrating both service oriented architecture and traditional control organisation could provide an efficient and agile support to develop on demand control applications integrating smart devices. By this way, online available control services could be mixed with physical smart objects to compose agile control systems.

1.2 Motivation and Challenges

Although in recent years, control systems have assumed an increasingly important role in the digital transformation [36], traditional control system are less agile such that it can not meet user’s evolving and multi-functional requirements:

1) since physical devices (i.e., sensor, controller, actuator) can only be leveraged by owner itself, control devices are lacking of re-usability, which will lead to costs increasing and even resources wasting;

2) these devices are inter-dependent physically.

This will boost maintenance cost when control system breaks. The improvement of control level can benefit to individual (better and safer living conditions), environment (improved traffic regulation, smarter city, reduced energy consumption...) and industry (more efficient production, better product quality and traceability control, reduced

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

wastes...). To achieve this goal new smarter and distributed control models, integrating various sensors and actuators must be defined. Fortunately, in today's environment where Internet of Things (IoT) and cloud computing provide strong basis for the current technological infrastructures, the number of smart objects (sensors, actuators, and even controllers) available and accessible to users grows in an exponential way. This can shape the way control system is constructed to be promoted to meet the corresponding objectives from the customers. Subsequently, Cloud Control System [103],[104] (CCS, or Cloud based Control System [57]) deemed as a natural paradigm for the next generation of control system is proposed from initially traditional wired control system, to Networked Control System (NCS). From wired control system to Networked Control System, the connection way has been shifted to be wireless (Internet, Blue tooth, WIFI), taking advantage of IoT technique. Cloud control systems extend networked control system via making use of cloud computing technology. In the cloud control system, the plant and controllers are both observed as network nodes that are able to perform certain tasks [71]. Mahmoud Magdi [57] emphasizes the concept of control as a service (CaaS) is enforced behind the cloud control system. In summary, our cloud control system is a system where there are four types of elements, sensor, controller, actuator and controlled object (plant). Moreover, sensor, controller and actuator could deemed as individual network nodes able to communicate with each other and carry out their own tasks. Due to the development of cloud computing and the proposition of fog computing, cloud-fog control system[112] is proposed as well, but essentially it is same with cloud control system.

Focusing on the IoT and smart device side, IoT system or service should have abilities of automatic configuration and context awareness. As IoT components have similar characteristics as services e.g. a unique identification, being discoverable, composable, modular, and providing network-based connectivity, they provide flexibility and agility at the device level, supporting efficiently networked control systems.

In addition to ones above inherited from the networked control system, cloud control systems have the following two advantages.

- **Elements in cloud control system can be more fully utilized.**
  Cloud computing especially public cloud bring a comprehensive services available categorized into IaaS, PaaS and SaaS. This will boosts the possibility of use of smart sensors, controllers and actuators, and thus save the fundamental cost and add the competitive competence for the enterprises.

- **Cloud control system can attain the more optimized performance globally.**

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Thousands types of massive data are collected from sensors and controllers and then sent to cloud. Big data provides a functionality of data analysis and control in the cloud. Controllers in the cloud control system might implement an algorithm of machine learning, deep learning, or reinforcement learning. This makes the predictability more precise. For instance, car license recognition controller applies the deep learning method to recognize the car; fault detection of a machine in advance can also be achieved after feeding masses of historical data, leveraging the advanced data-driven control policy.

Regarding the cloud control system, the most fundamental but also most significant question is that how we can implement one? The professor Xia [104] summarizes that there are four challenges which cloud control systems are faced with at the current stage:

1) How to recognize, process and transfer quantities of data?

2) How to make sure the real-time control could be realized?

3) How to guarantee the control quality and stability of cloud control system?

4) How much is required to develop it as it demands a huge cost?

Cooperative cloud control is pointed out to solve the problems above. It also divides the implementation of cloud control system into two phases: Initial phase (NCS phase) and cloud control phase. The first stage is aimed at designing controller taking into account the time delay and data dropout induced by Internet transfer, in order to meet the requirements on stability and other control qualities. The second phase is targeted to allocate the precise controller that can satisfy the requirements of communication, computation, to perform the given task.

In our research, we only focus on cloud control characteristics that essential to any IoT system [118] : heterogeneity, autonomy and adaptivity.

- **Heterogeneity.** In a cloud control system, there are three types of services: sensing, actuating and controlling services. These services have their own characteristics. The format, connected way, communication protocol of each of them may be heterogeneous. This challenges the way we describe the cloud control system.

- **Autonomy.** Autonomy is the most fundamental feature of any control system, including cloud control system. It requires the integration of sensing, controlling and actuating services automatically. Higher the level of integration is, more efficient the cloud control system is. This can make sure that cloud control system can work in an automatic way.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

- **Adaptivity.** In IoT environment, IoT devices change frequently. For example, a device might often leave and join the network. The computation and storage capabilities could be no more available as constrained resources are being occupied. Sometimes, user requirements varies as well. For example, the expected temperature change may cause the replacement of the bound sensing service. So cloud control system should be context-aware so that it can be adaptive.

1.3 Research Issues and Contributions

Basic assumptions are defined as a rudiment of a classical cloud control systems by YuanQing Xia [104]:

1) A broadcast domain is involved in this rudiment, in which all nodes can reach each other by broadcast at the data link layer.

2) All nodes in the broadcast domain mentioned above are intelligent enough to undertake the cloud control task, but their computation abilities are assumed to be equal to each other, and the available computation resources change unpredictably.

3) The network delivery is not ideal, bounded time delays and data delivery dropout could occur during any transmission.

4) The network delivery time delays and dropout statistics between any two particular nodes could be obtained by the same nodes in some ways.

Also, in our work there is a prerequisite that quantities of sensing, actuating, and controlling services (devices) are available on the Internet. It is noted that our study does not pay attention on the controller design for the cloud control system. Instead, it is focused on the cloud control after the complement of controller design.

In chapter 2, we investigate the state of the art regarding both traditional information systems, IoT and Cloud control. In fact, Cloud control aims at providing more autonomy, adaptivity and interoperability for the manufacturing processes and even the whole industry 4.0. According to this challenge, we develop a research strategy according to two basic questions: how can we define a Control service and how can we organize a Service oriented control architecture gathering both physical devices and control services in Control as a Service vision.

- Firstly, we investigate both traditional enterprise architecture frameworks and service oriented architecture to identify whether they can be adapted to fit the Cloud

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

control requirements. This state of the art research shows that the service field provides a strong background to support loosely coupled complex systems but it lacks integrating physical devices and exact control system requirements.

- Secondly, we investigate both IoT and Cloud Control field to identify the way smart devices and associated control service are defined and modeled. This state of the art review shows that Cloud Control is a dynamic research field and that IoT takes an important part. Nevertheless, it lacks of integrating control patterns to develop efficiently cloud control system.

To overcome these limits, we identify two research questions:

- **Q1: What is a Cloud control service:** whereas basic sensing services, controlling services and actuating services are supported by the related physical sensors, controllers and actuators respectively, defining a cloud control system requires a more generic and precise definition of the control service.
- **Q2: Based on the control service definition, how can we define a service oriented control architecture to support our Control as a Service model:** similar to the IT Service Oriented Architecture organisation, such an architecture is necessary to define how to select, compose and orchestrate these distributed control services and their related physical smart devices involved sensing services, control services and actuating services.

To answer these questions, our contribution is organized into two chapters: chapter 3 presents our Control as a Service model, paying attention to functional and non functional requirements; and chapter 4 is focused on our Service Oriented Control Architecture (SOCA), provisioning the development steps for a Cloud Control System (CCS). Each of these contributions have been analyzed and have lead to the following sub-contributions related to our key research questions (see table 1.1):

Table 1.1 – Relationships between contributions and research questions

No	Sub contribution	Research question
1	control service functional model	Q1
2	control service non functional model	
3	control ontology	
4	multi-layer architecture (SOCA)	Q2
5	SOCA operation deployment at design time	
6	SOCA operation deployment at runtime	

- Contribution 1: building a Control as a Service model

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

a) Sub-contribution 1: control service functional model
Setting a control service model starts with identifying the way it can be functionally defined. We propose a control service model built according to controlled object and controlled variable, allowing the interconnection of sensors, actuators and controllers according to control requirements. Leveraging the knowledge of control field, a detailed control system model is achieved by adding other concepts of controlled object, controlled variable, control pattern and environment variable, and corresponding connections of all entities. Definition of controlling service is defined in a recursive way using the mathematical function representation.

b) Sub-contribution 2: control service non functional characteristics
Whereas the functional description allows identifying precisely what the control service will do, non functional requirements must be integrated as well to define the context and devices characteristics. Although several ontologies have been defined for IoT devices, they are devoted to some control characteristics such as precision, device computing capabilities and only few address environmental characteristics. To allow a more precise definition of the control environment, non functional control requirements as well as device computing capabilities, we propose a global NFP classification to support a unified NFP ontology

c) Sub-contribution 3: identifying a control ontology
To guide Cloud Control System requirements, we gather the main concepts related to our control service model, including both functional and non functional characteristics in a single ontology. Paying attention to the "loosely coupled" requirement while defining control services, we propose to interconnect these services thanks to an event-based organisation. By this way, control services and their interface can be fully defined, allowing an efficient selection and composition process according to the control system requirements.

- Contribution 2: building a Service Oriented Control architecture

a) Sub-contribution 4: designing a multi-layer Service Oriented Control Architecture
Based on our control service model, we propose a multilayer Service oriented Control architecture:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

- The physical layer is used to manage physical devices (sensor, controller, actuator, physical object, gateway, and even the whole control system) and constraints including physical environment (e.g., location) and physical capabilities (e.g., memory, CPU, connectivity...).
- The micro-service layer is defined as an interface tier. It allows embedding the physical device in a micro-service interface to allow a smoothed connection to the Cloud Control System.
- The Logical layer is populated logical descriptions and constraints. It consists in controlling service, actuating service, sensing services and other concepts defined in the service oriented architecture (SOA) and quality of service (QoS, e.g., precision, availability, security, cost, interoperability.).

b) Sub-contribution 5: operations related to the SOCA deployment at design time

Based on the traditional Service Oriented Architecture and on our Control service model, we define service registries and associated service selection operation. By this way control services (including sensing controlling and actuating services) can be retrieved and selected thanks to their functional and non functional characteristics. Then, in order to build an adaptive control system, we redefine the pre-composition operation to manage candidate services

c) Sub-contribution 6: operations related to the SOCA deployment at runtime

To support the loosely-coupled control service invocation at runtime, we adapt the traditional Enterprise Service Bus to our Control service model. The orchestration step is based on an event-driven organisation. Instead of routing messages among services, our Control Service Bus is in charge or coordinating control services thanks to events. Then a context manager is in charge of a late binding process allowing to select the best candidate from the pre-composition graph according to the precise context constraints. This last feature increases the Cloud Control System flexibility and agility.

The dissertation is organized as follows (see Fig 1.1.) Totally, there are five chapters, general introduction, state of the art, Control as a Service model, Service Oriented Control Architecture (SOCA for short) and conclusion. Chapter 2 presents the requirements for new model and architecture for cloud control system in the distributed and collaborative environment. In the chapter 3, Control as a Service model is designed, ending with a full-fledged control ontology that describe cloud control system from functional,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

non functional and interactive views. Service Oriented Control Architecture connecting physical devices and logical services is achieved in chapter 4, allowing to implement control services selection, composition and orchestration for a cloud control system in a contextual and event based way. The last chapter is devoted to summarizing the current work and showing the potential directions of future work.

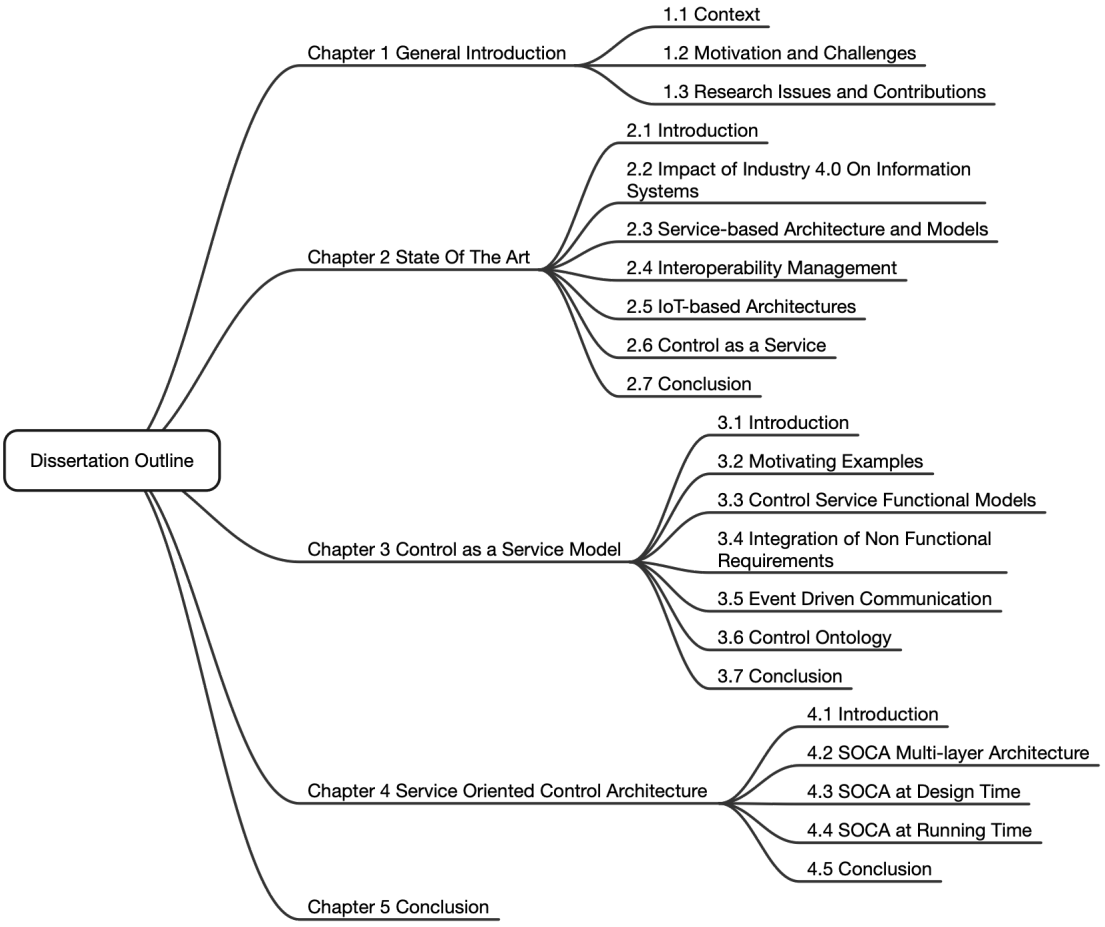

Figure 1.1 – Dissertation outline

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Chapter 2

State Of The Art

Contents

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

2.1 Introduction

The fourth industrial revolution named “Industry 4.0" has been firstly and formally introduced during the Hannover Fair in 2011 [49], [105]. It challenges latest information and communication technologies (such as cyber-physical systems (CPSs), Internet of Things (IoT), industrial internet of things (IIOT), cloud computing, cognitive computing and even artificial intelligence (AI).), to improve automation and data exchanges in manufacturing processes. Industry 4.0 and the related digital transformation of the industrial systems are “hot topics", integrating smart factory as well as intelligent production system design. It represents the factory of the future shifting centralized production to decentralized production, changing popular items to customized items, and isolated end-users to interconnected end-users. Factory who is able to embed elements of production or realize the intelligent production needs to attain comprehensive advantages in productivity, revenue growth, employment, investment [117]. Furthermore, Entire value chain of a producer will be influenced from design to after-sales service. Manufacturing processes will be enhanced more flexibility, quickness, productivity and quality as a result of introducing intelligent machines, automatic robots, and other smart apparatus.

Paying attention to the smart factory research field, most of works are focused on networked organisation of CPS (Cyber-Phsyical System) defined by defined by Dr. James Truchard [91]. It states that virtual space and physical world can be merged together via allying pervasive sensors, terminal systems, intelligent control system [73]. It can be seen as a way to map physical objects related to smart devices to their logical function. Therefore, it is an integration of computation, communication and control. In addition, computation devices, embedded sensors and actuators in CPS are able to let the operations monitored, cooperated and controlled in real time. Consequently, cyber physical system (CPS) and more specific cyber physical production system (CPPS) are the foundations of industry 4.0 [66], [65].

As stated in a survey [33] on the current researches in the field or in [117] for a research roadmap, a network of cyber physical system (CPS) should be built for researching two major themes, smart factory and intelligent production. This requires to implement different integrations, namely, horizontal integration, vertical integration and end-to-end integration. Meanwhile, eight planning objectives: 1) standardization of

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

systems and building a reference architecture, 2) efficient management, 3) establishment of a comprehensive and reliable industrial broadband infrastructure, 4) safety and security, 5) organization and design of work, 6) staff training and continuing professional development, 7) establishment a regulatory framework, 8) improvement the efficiency of resource use, ought to be achieved.

However, we need to state that it's not easy to get to the scene of industry 4.0 and this will take years due to many challenges experts from the industry and academia have been facing with and working hard to solve. Industry 4.0 involves defining easy to design control systems based on the interconnection of reusable smart devices. Smart Buildings, smart city or any other distributed control system using the Internet of thing have also the same requirement. This leads to different challenges:

1) Providing a user friendly control system design environment integrating smart devices. Although smart devices offer a remote access interface be accessed, implement self-awareness functions and may accommodate friendly GUI, setting a consistent distributed control system remains complex. The first problem is due to the large variety of these smart devices and second on the way control patterns can be identified and reused to propose consistent control system. This leads to two sub-challenges:

   a) identifying a pattern-based control system design process can be inspired by traditional enterprise architecture frameworks, gathering best practices to design and implement information systems. Adapting these frameworks to the distributed control needs integrating the control process itself.

   b) supporting smart device and control blocks reusability and adaptation: this challenges for the development of loosely coupled control elements. This feature has already developed in Service Oriented Architecture. It needs to be adapted to suit the control field

2) Cyber Physical Systems (CPS for short) couple a real objects from the physical world with digital artifacts in a virtual space. It takes advantage of communication technology, computation technology and control technology. Hence, a Cyber Physical System is a complex system resulting from virtualization and cross-techniques. An efficient CPS needs to solve the problems that how to connect and control sensors and actuators seamlessly. Moreover, despite the works achieved by different consortiums, by now, no generic CPS model nor CPS reference architecture have been developed yet Also, more attention should be paid to the integration, ver-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

ification, testing of CPS. In a word, a long journey is waiting for us to go and finish.

3) How to interoperate with existing heterogeneous systems and platforms is still a big problem. Many researchers have applied semantic web technologies to IoT domains. Though some semantic technologies, such as resource description framework (RDF), RDF scheme (RDFs), web ontology language (OWL), are proved effectively in some projects mostly supported by EU and some standards are achieved but not recognized and applied globally and pervasively. Within industry 4.0, manufacturing domain is influenced most and thus will upgrade faster than other fields. Enterprise architecture as an important part needs to be improved and adjusted to new business collaborative environment. This is a mean of how business organizations challenge latest information technologies to keep competitive and leading roles. Hence, enterprise organization is bound to be reorganized and embrace new services.

4) Industry 4.0 as well as other smart-devices related distributed control systems such as smart buildings or smart city involve the interconnection of smart devices providing large quantities of heterogeneous data related to devices, machines, products, processes that can be integrated with other data produced by applications and services, etc. How to collect, analyze, process big data may be dependent on some artificial intelligent methods. However, the application of artificial intelligent in many aspects are at their initials. At the same time, fast response abilities required for industry 4.0 and other smart-control applications is hard to realize.

Our work in this dissertation is mainly focused on building a new control model and an service oriented control architecture for the cloud control system adapted to the industry 4.0 environment. This will add the control intelligence to Cyber Physical System (CPS) and contribute to the development of smart devices. It also involves in the interoperability of devices and real time data processing. In what follows, information system that calls for the distributed control service architecture and model, and reference architectures and models that provide the strong basis for them, are reviewed firstly. Then, control as a service encompassing its predecessor(networked control system) , its procedure (servilization of control) and cloud control steps (selection, composition and orchestration), is surveyed.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

2.2 Impact of Industry 4.0 on Information Systems

Industry 4.0 presents us a scene where a product will be manufactured immediately after the order is placed in the business department from a user. This efficient and highly customized characteristics requires deep integration of business process and manufacturing units. Enterprise architecture guides the design, implementation and deployment of information system. It allows the different system or applications from heterogeneous partners communicate with each other. Meanwhile, it needs to deal with the complex business processes. Down to the production side, various manufacturing resources ought to be combined together to guarantee the lowest cost of workforces, materials, transportations. Advanced production strategies and means would be applied as much as possible. For instance, cloud based solution helps to fast construct and deploy an system or application, leveraging the cloud based services. This cloud based strategy could also provide adequate adaptiveness due to the transparent logging information. As a consequence, cloud manufacturing integrating the business process and cloud computing needs to be studied.

2.2.1 Traditional Enterprise Architecture Frameworks

Enterprise architecture (EA for short) has been introduced to face the complexity of information systems. EA frameworks are used to provide a clear vision on information system organisation and manage enterprise resources. It allows decreasing time and efforts while developing information systems and delivering maximum business value. Enterprise architecture is layered hierarchically by five architectural layers: bottom to up technology (or infrastructure) architecture, software architecture, integration architecture, process architecture and business architecture [98]. Years ago, four top enterprise architectures, Zachman Framework, The Open Group Architecture Framework (TOGAF), Federal Enterprise Architecture (FEA) and Gartner, have been introduced and detailedly compared and analyzed [93],[83] [53].

First, in order to mitigate the complexity of information systems and facilitate the management of software components, John Zachman has proposed an enterprise framework [108], [109]. This enterprise architecture is a taxonomy of corporate artifacts (i.e., specific document, report, analysis, model or other tangible files). It also provide a guidelines to set an information system, known as Zachman grid (see Fig. 2.1). This grid gathers a stakeholder view (planner, owner, designer, builder, subcontractor and user) and six fundamental questions frequently used in journalism, namely, what, how, where, who, when, and why. By this way Zachman's framework provides a comprehen-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

sive methodology for managing a complex corporate information system, achieving a great impact on both research and industrial practitioners. A precise use case showing how the enterprise architecture is defined by using the Zachman framework is discussed in [70]. However, Zachman's framework can be seen as a complete Enterprise Architecture framework if and only if all cells in the Zachman's grid are fully filled. What's more, each artifact can only be linked to one cell. These conditions put much difficulty on creating a new domain-specific enterprise architecture.

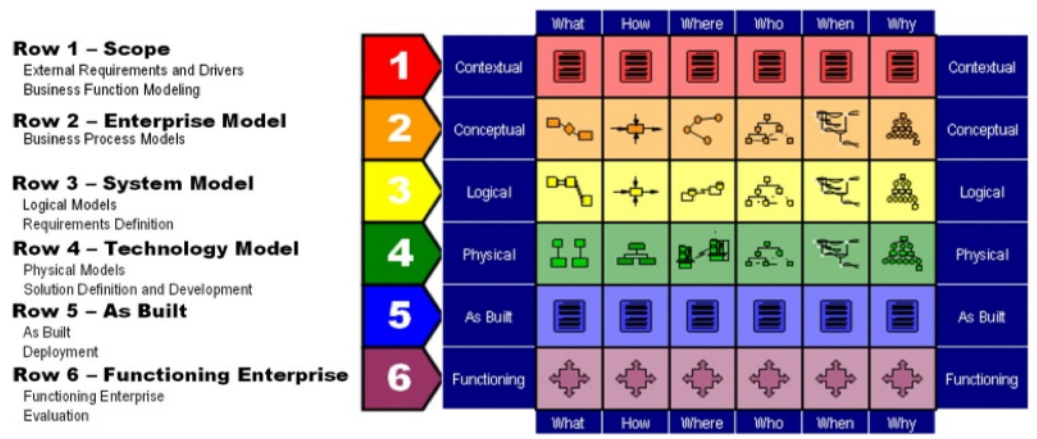

Figure 2.1 – Zachman framwork of enterprise architecture [2]

Then a consortium of the main IT companies, the Open Group, has proposed its own framework : The Open Group Architecture Framework (TOGAF [1]) . TOGAF also guides the Information System engineering and implementation process by providing four different architectures organized in a top-down way: Business Architecture, Data Architecture, Application Architecture and Technology Architecture. The Architecture Development Method (ADM, described in TOGAF part II) (see Fig. 2.2) defines how to derive an organization-specific enterprise architecture to address business requirements. This method is supported by ADM guidelines and techniques (part III). Deliverables produced by the ADM are stored in the repository (part IV) according to the different deliverables types (process description, data models...). Of course, an enterprise architecture project cannot be set in a big bang strategy. This involves organizing it according to enterprise continuum (part V) and also identifying how different enterprise architecture projects can be connected to each other. To support this last feature, the repository is incorporated within the TOGAF reference model (part VI). Lastly, the architecture capability framework (part VII) describes the way an enterprise architecture project can be organized and governed from the first design steps to its deployment.

[1] https://pubs.opengroup.org/architecture/togaf9-doc/arch/

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

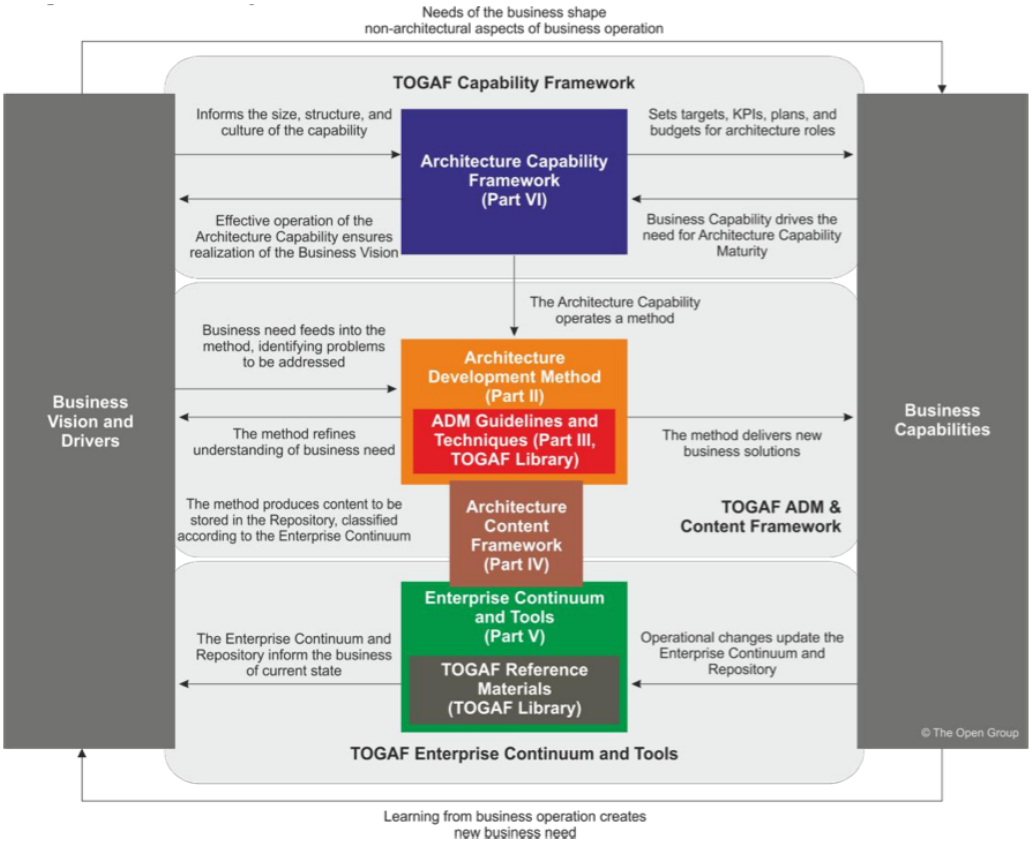

Figure 2.2 – Excerpt of TOGAF Content Overview (TOGAF Standard 9.2 p8)

The Federal Enterprise Architecture (FEA) framework [2] is a latest attempt to form a persuasive architecture more complete methodology due to its integration of both taxonomy features described in the Zachman's framework and the process specified in the TOGAF framework. FEA framework encompasses six reference models, namely, performance reference model, business reference model, data reference model, application reference model, infrastructure reference model and security reference model (see Fig. 2.3). FEA framework provides a toolset to support a global description of the way distributed Information Systems interact with each other and with their environment according to "business goals". The associated enterprise engineering methodology embedded in FEA framework includes different parts: 1) a perspective on how enterprise architectures should be viewed, 2) a set of reference models for describing different perspectives of the enterprise architecture, 3) a process for creating an enterprise architecture, 4) a transitional process for migrating from a pre-EA to a post-EA paradigm, 5) a taxonomy for cataloging assets that fall within the purview of the enterprise architecture, and 6) an approach to measuring the success of using the enterprise architecture

[2] https://obamawhitehouse.archives.gov/sites/default/files/omb/assets/egov_docs/fea_v2.pdf

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

to drive business value[83].

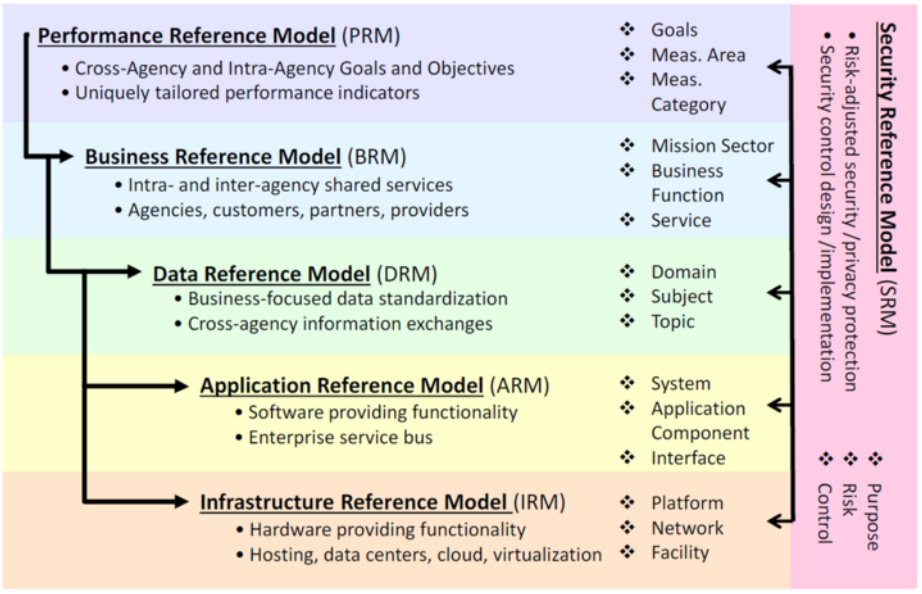

Figure 2.3 – Federal Enterprise Architecture [3]

These different enterprise architecture frameworks provide a strong background for engineering Information Systems thanks to both design and implementation patterns. Nevertheless, they need to be extended as they do not provide any pattern regarding the manufacturing processes.

2.2.2 Integration of Manufacturing Requirements and Patterns

As said previously, these frameworks are focused on “Business Information Systems" and they need to be extended to integrate products and smart production means as industry 4.0 aims at integrating closely production process. To fit this goal, RAMI 4.0 model (Reference Architecture Model Industry 4.0, see Fig. 2.4) is a service oriented architecture, which is specified from the aspects of functional layers, value stream and “hierarchy levels" from “basic" products to the connected world [111], [5]. Vertical layers consists of from the bottom to top layers of asset, integration, communication, information, functional and business. Horizontal layer is Value stream describing the stages of products from design, verification, production, to maintenance. Hierarchy levels introduces all participants who are able to interact. These participants are product, field device, control device, station, work unit, enterprise and the connected world. As an IT model, cyber security of RAMI 4.0 model is also addressed [29]. An I4.0 (Industry 4.0) component model is also provided to produce I4.0 style components by encapsulating them with an administrative shell.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

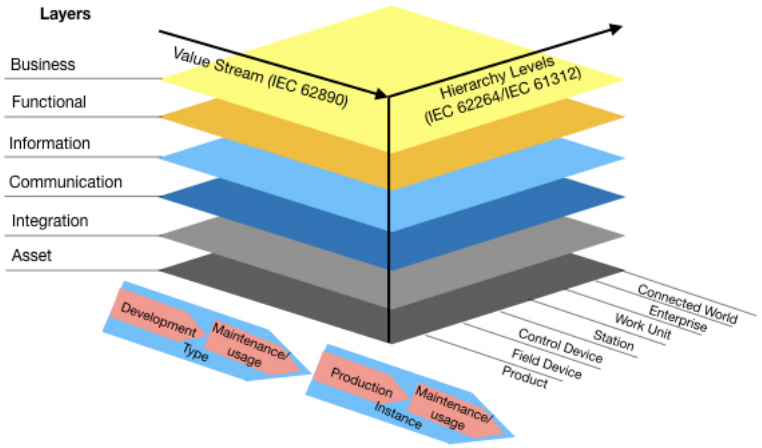

Figure 2.4 – RAMI 4.0 model [7]

RAMI 4.0 model and other existing enterprise architectures allowing distributed components from different partners interact in the collaborative context, Nevertheless these information systems are generic without integrating a specific field. Manufacturing as a key domain in the industry 4.0 has been researched for decades.

Collaborative organizations are set to support distributed manufacturing / supply chain. This involves defining new organizational models (such as VM (virtual manufacturing [119])) interconnecting different information systems. At the same time, new production organisation such as lean manufacturing [100] are developed to set more efficient organisation. When a large-scale distributed and efficient production organisation like other manufacturing models is required, it is also intended to realize the goals of TQCSEFK (i.e., fastest Time-to-market, highest Quality, lowest Cost, best Service, cleanest Environment, greatest Flexibility, and high Knowledge) [87]. In [101], key characteristics of cloud manufacturing are summarized: 1) customer centricity, 2) temporary, reconfigurable, dynamic, 3) turn no job away, 4) demand driven, demand intelligent and 5) shared burden, shared benefit. The architecture of cloud manufacturing is composed of from bottom to up resource layer, perception layer, service layer, middleware layer, application layer, as well as cloud manufacturing standards and criteria/ communication/ knowledge/ safety and security across the former five layers (see Fig. 2.5). A simplified cloud manufacturing architecture is organized in four layers: manufacturing resource layer, virtual service layer, global service layer and application layer [106]. Attention is paid to the lowest resource and perception layer in paper [88] where the classification of production resources are completed. Multi-agent system (MAS) because of its distributed nature and recursive structure has been applied into manufacturing system planning, scheduling, exaction control and material handing by researchers

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

and practitioners [19]. An agent is perceived as intelligent manufacturing service able to communicate with other agents. Data from autonomous agents are acquired to build control system based on its proposed semi-hierarchical control architecture [80]).

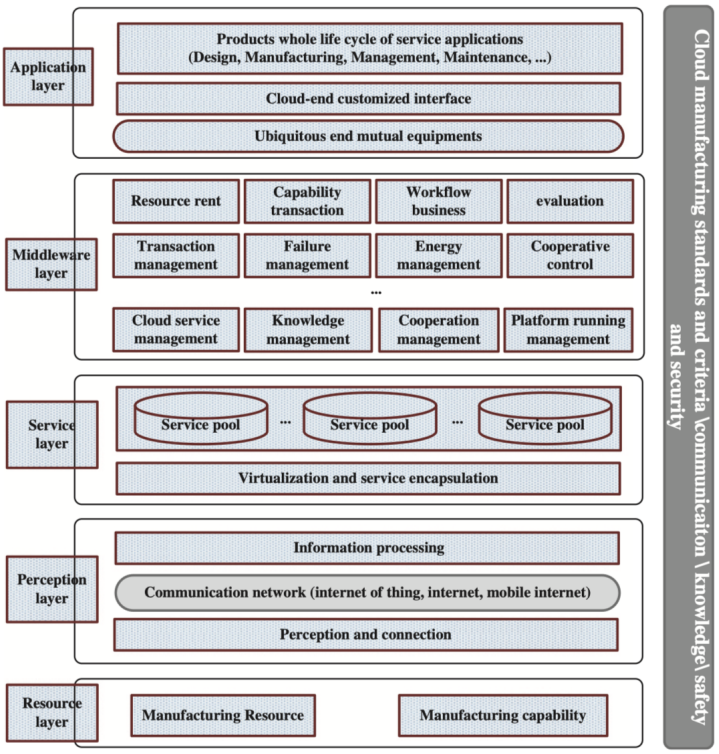

Figure 2.5 – Cloud manufacturing architecture[113]

2.2.3 Conclusion

In this section, we have presented enterprise architecture from Zackman framework, TOGAF, FEA framework to current RAMI 4.0 model. RAMI 4.0 model provides a conceptual descriptions of how to implement industry 4.0 visions. All these frameworks and models provide a strong basis to set design and implementation patterns. Then, a precise cloud manufacturing architecture is dedicated to working out the difficulties production information system are faced with, leveraging the cloud computing technique. So in this highly connected but geographically distributed context, architectures or models of each layer in both enterprise architecture and cloud manufacturing architecture should not be centralized, and system or application originated from of them ought to be possessed of features interoperability, together with other qualities, e.g., flexibility, agility, reusability, reconfigurability, maintainability and adaptiveness. To fit this goal, most of the frameworks advocate for Service Oriented Architecture regarding the implementation.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

2.3 Service-based Architecture and Models

In this globally collaborative and quickly changed environment where there are more and more interactions and connections among enterprises, agility and flexibility are more emphasized by them. SOA brings systems or applications much benefits on the aspects of flexibility. Clear definition of service interface provides users with loosely coupled feature, and enterprise service bus makes it possible that distributed entities interact with each other. Business process and enterprise ontologies are dedicated to achieving the interoperability between collaborative partners. Implementation of logical layer functionalities depends on the physical tier devices. The description and usage of physical devices are attained, taking advantage of IoT ontology and reference architecture respectively. Sensor and actuators would not be connected without mediate controllers. As a result, control system or control architecture consisting of controllers is also necessary as a part of vertical information systems.

2.3.1 OASIS SOA Reference Architecture

In this section, OASIS SOA Reference Architecture[3] is used to introduce the relevant paints of models of service. It it followed by the the subsection which are dedicated to delineate how to compose services or objects in a dynamic way. Focusing on the service implementation, different descriptions protocols such as WSDL, RSDL... can be used to describe the service interface whereas an Enterprise Service Bus can be use to support the SOA implementation.

Service oriented architecture (SOA) is a paradigm for a set of capabilities distributed across the network and possibly under the control of different ownership domains. Thus, the core of SOA is flexibility. It aims at defining the essence of service oriented architecture, and delineating a unified vocabulary and a common understanding of SOA. It provides a normative reference that remains relevant for SOA as an abstract and powerful model, irrespective of the various and inevitable technology evolutions that will influence SOA deployment. SOA consists of dozens of parts or sub-models and they are connected and interacted. For example, resource model, service participants model, general description model, etc.

2.3.1.1 Service Description Model

Service in SOA is specified by service description including service reachability, service interface, service functionality, metrics, policies and contracts (see Fig. 2.6). Service

[3]http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

reachability comprises protocols, service presence, and endpoint. Service interface is mainly represented by behavior model and information model. The former is primarily embodied by action model and process model while the latter is dominantly related to contents of semantics and structure. Among service functionality part, there are functions, technical assumptions and service level real world effect. Real world effect is produced by one or more functions and must be consistent with technical assumptions. Policies and contracts have two parts, contracts and service level interaction policies. Metrics provide evolutions for quality of service.

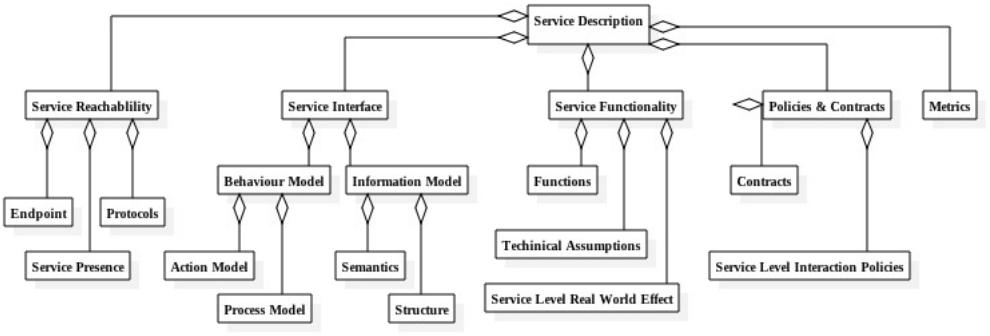

Figure 2.6 – Excerpt of service description model (OASIS 2008 p42)

Before introducing the service interaction, service visibility should be illustrated as it is a necessary condition of service interaction. Attaining service visibility means achieving service awareness, determining service willingness and establishing service reachability. Service reachability requires finding endpoint, then verifying interface and eventually establishing presence of service itself. Interaction is the usage of a service to get a particular goal. Service Participants can receive message from and send messages to service. Often, one atom service can not meet the user's intentions, so composition of service is persuasive. In business environment, a business process (for example, a bank transaction) is defined as a set of consistent activities. Service composition defines the service chain and orchestration techniques provide a solution for the execution of this chain composition via applying a scripting language (such as WS-BPEL, Web Services Business Process Execution Language).

2.3.1.2 Service Interface Specification

Specifications of interface include Internet protocol, message model (message structure and semantics), and message exchange pattern (event notification mode in control service bus) (see Fig. 2.7). Event notification can be associated to message transfer. This

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

event-based mode allows building more reactive systems. Internet protocol and message model will be described further.

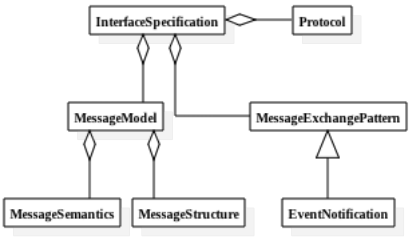

Figure 2.7 – Interface description

Internet exchange protocol depicts how a message is transported between service interfaces. Logically, application and transport layer protocols (e.g., HTTP, TCP) provide a formalism of service while Tower Network oriented protocols is physically related to data linker and network layers(e.g., Ethernet, IP). Protocols applied to physical interfaces can affect the characteristics of communication (transmission speed). Message is exchanged between services. Thus it is an input from the angle of service consumer but an output form the view of service provider. In order to understand a message, message structure and semantics must be both known. The former emphasizes the format of data while the latter stresses the meaning of data. Regarding the message structure, different levels of data structure can be applied to the conveyed message, for instance, the format of encoding character data (e.g., UTF-8), the form of the data (e.g., integer), representation structure (e.g., XML), and so on. Knowing the representation and format of message is not enough to interpret it. For example, only numbers as message are exchanged between service participants. In this condition, other information (such as unit and domain) must be given to provide unambiguous data for users. System engineering interface has been delineated from several points of views, namely, part interface, layered part interface, interface specification, interface connection, interface object flow, interface function, interface function occurrence, and performance and limitations on interfaces [31]. These descriptions are essentially similar with that of WSDL. From the angle of Interface Specification, it is intended to describe the operation, which belongs to PortType. Layered part interface is aimed at differentiating the input and output message. Then binding interface specification into port with one or more instances makes the service accessible by several endpoints in an interface view.

- **Service interface description using WSDL**

  The WSDL[4] (Web Service Description Language) is the XML-based service repre-

[4]https://en.wikipedia.org/wiki/Web_Services_Description_Language

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

sentation language used to describe the details of the complete interfaces exposed by web service. It is platform and language independent. A WSDL document should include not only abstract interface definitions but also concrete implementation definitions, of a web service. Among interface specifications, there are totally four elements defined in XML, namely, ⟨types⟩, ⟨message⟩, ⟨operation⟩ and ⟨portType⟩. The ⟨types⟩ element works as a container collecting all abstract data types required in the web service. These data types can be basic built-in data types (e.g., int, string, boolean), or customized data types. The ⟨message⟩ element describes the payload of a message used by a web service. It consists of ⟨part⟩ elements. The ⟨operation⟩ element represents the method in the web service. It usually requires messages, including input message, output message, or both of them. This depends on interaction patterns (see table 2.1) between service provider and consumer. Note that the sequence of input and output is different so that the operation interaction patterns are not the same. The ⟨portType⟩ element groups all the operations needed. Regarding web service implementation specifications, there are three XML elements, ⟨binding⟩, ⟨port⟩, and ⟨service⟩. The ⟨binding⟩ element is the most important because it specify how the elements in an abstract service interface are converted into concrete representation. The binding procedure will determine and instantiate protocols, messaging styles, formatting (encoding) style. The ⟨port⟩ element defines a web service's location, which is expressed by a URL, in general. Moreover, ⟨binding⟩ element has to be linked to port itself. The ⟨service⟩ element groups ⟨port⟩ elements. It means that a service can be exposed and located by one or more ports.

Table 2.1 – Relation between message and operation interaction pattern

Operation interaction pattern	**Message**
One-way operation	input message
Request/response operation	input message, output message
Notification operation	output message
Solicit/response operation	output message, input message

- **REST service interface description using RSDL**

  The RSDL (RESTful Service Description Language) is a machine- and human-readable XML description of HTTP-based web applications, intended to simplify the reuse of web service. In RSDL, there are three key components, ⟨link⟩, ⟨request⟩ and ⟨response⟩. Link is usually expressed in a way of URI (Unified Resource Identifier). Behind this style REST web service is a ROA (Resource Oriented Architecture) where almost every thing (figure, computing function, etc.) is perceived

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

as a resource. The resource is located by its URI and is invoked by its packaged methods (HTTP methods, e.g., Get, Post, Put, Delete). Parameters required is in the ⟨request⟩ part. Result, if required, will be returned and its format is illustrated in the ⟨response⟩ part. In this paper, we applies this style of service interface description.

2.3.2 Enterprise Service Bus (ESB)

An Enterprise Service Bus (ESB) is a middleware connecting distributed services in a heterogeneous environment and an implementation of service oriented architecture (SOA). ESB is needed when three or more applications or services need to be integrated, especially ones from the external third party and with requirements of high quality of service (QoS) integration [51]. Thus, ESB is dedicated to managing service invocation between service provider and service consumer, making these interactions simple and reliable. Among ESB, core parts are service registry and mediation service. Service registries are used to store service description so that services can be discovered and selected [81]. Based on a request, a link connecting service requester and service provider is built. Mediation service are generally needed to adapt different interface types (data format, data exchange pattern). In addition, mediation service needs to realize other functions, such as message transformation, protocol translation. To support the functionalities accommodated by ESB, technologies (i.e., web service, SOA, cloud computing, Internet of Things (IoT), etc.) are leveraged. Thus, ESB plays a significant role in composing distributed services or integrating enterprise applications [42]. For example, in a service composition architecture (called SynchroESB), ESB layer as lowest one of four tiers provides the mediated message exchange infrastructure, allowing distributed services to interoperate with standard-formatted messages [69].

Essentially, ESB implements a communication system between mutually interacting software applications in a SOA. However, functionalities of traditional Enterprise Service Bus (ESB) have to be adapted to integrate specific control requirements.

In summary, this reference architecture provides us a theory of how a service is described, of how services interact with one another, and of how services are composed (process model). However, this architecture has been defined for software components. Paying attention to devices and control process, device interaction and event-based organizations should be added. In order to ensure IoT service interaction, a general realizable interface description must be given. To manage the event notification message of IoT services and launch them, Enterprise Service Bus (ESB) is required as well.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

2.3.3 Conclusion

OASIS reference architecture specifies the services functionally and involves in the descriptions of their qualities and security. Services could be developed, deployed and operated congenitally. Service Oriented Architecture and models increase interoperability between systems as service interface are clearly detailed. Moreover, they also provide loosely coupled features. Thank to the ESB message routing more agile systems can be built. Nevertheless, a particular attention must be paid on the way data is semantically defined to support interoperability. To this end, ontologies must be defined.

2.4 Interoperability Management

As stated previously, services and their associated standards only support a technical interoperability between the different components of the information information system. To manage efficiently heterogeneity, ontologies must be defined not only for both business and manufacturing processes but also for smart devices, i.e. Internet of Things.

2.4.1 Business Process and Enterprise Ontologies

To fit the renewed globalized economic environment, enterprises, and mostly SMEs (Small and Medium-sized Enterprises), have to develop new networked and collaborative strategies, focusing on networked value creation (instead of the classical value chain vision), fitting the blue ocean context for innovative products and service development. Even if collaborative organizations such as virtual enterprises, collaborative networks... have been studied for decades [35], [95], [15], the closer connection of information systems involved by the so-called “Industry 4.0" developed by leading industries in Europe, US and Asia requires to set new IT models to support agile and evolving collaborative Business Process (BP) enactment, integrating both traditional Information Systems (IS) and production. By now, these product/service ecosystems are mostly supported by software services, which span multiple organizations and providers, and on multiple cloud-based execution environments, increasing the call for openness, agility, interoperability and trust for both production and Information System organization. Control information traditionally managed by Manufacturing Execution Systems must be extended to integrate various data provided by IoT devices and to provide accurate data exchange to coordinate collaborative production processes.

Traditional enterprise information systems include various IT components (Enterprise Resource Planning, Customer Relationship Management, Supply Chain Manage-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

ment, Product Life-cycle Management, Manufacturing Execution Software...), each of them devoted to a particular business area and managing its own database. To allow consistent and smoothed interactions between these components requires managing syntactic and semantic interoperability. SOA and its related technologies provides a strong background to support technological interoperability between IT components thanks to a "syntactic mediation" achieved in Enterprise Service Bus, routing, adapting messages through the middleware according to some conditional logic [1]. Nevertheless, such a syntactic transformation is not enough to allow processing consistently information in a collaborative Business Process. Semantic mediation and organizational interoperability must be taken into account as well. To this end, ARPA I3 architecture [90] identifies different services to support this semantic mediation: coordination services (discovery, invocation), semantic transformation services (ontology translation, process integration) and adaptation services (event managing, form conversion). Nevertheless, it does not take into account business interoperability, i.e. integrating business organisation constraints.

To overcome this limit, more precise descriptions of the enterprise and production system are needed, requiring adapted ontologies. Paying attention to the enterprise business organisation, the Enterprise Ontology (EO) provides most of the terms necessary to describe business areas, strategy and process organisation (see Fig. 2.8) [94]. Thanks to dedicated sub-ontologies related to activities and processes, organization, strategy and marketing, the Enterprise Ontology provides a consistent support to describe the enterprise organisation which is necessary to set consistent Business Process, integrating different enterprises views. Last but not least, it also provides a meta-ontology, allowing a more flexible description and efficient integration. Nevertheless, it is mostly focused on the business and management organisation and is rather poor concerning product and production processes descriptions. To this end, another ontology can be used, ISA S-95 standard [20] which allows describing the product life-cycle, taking also production models and performance indicators (see Fig. 2.9). It allows integrating production management systems with other IT related components related to technical data management or to the production control process itself, mostly interfacing ERP and MES systems. This fits mostly interfaces between enterprise and production control systems by specifying which information should be exchanged between ERP and MES systems. To this end, ISA S-95 standard gathers different models such as production definition, production scheduling, production performance, production capacity and maintenance models. Nevertheless, the production organisation part does not support a precise description of the control system internal organisation, as each

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

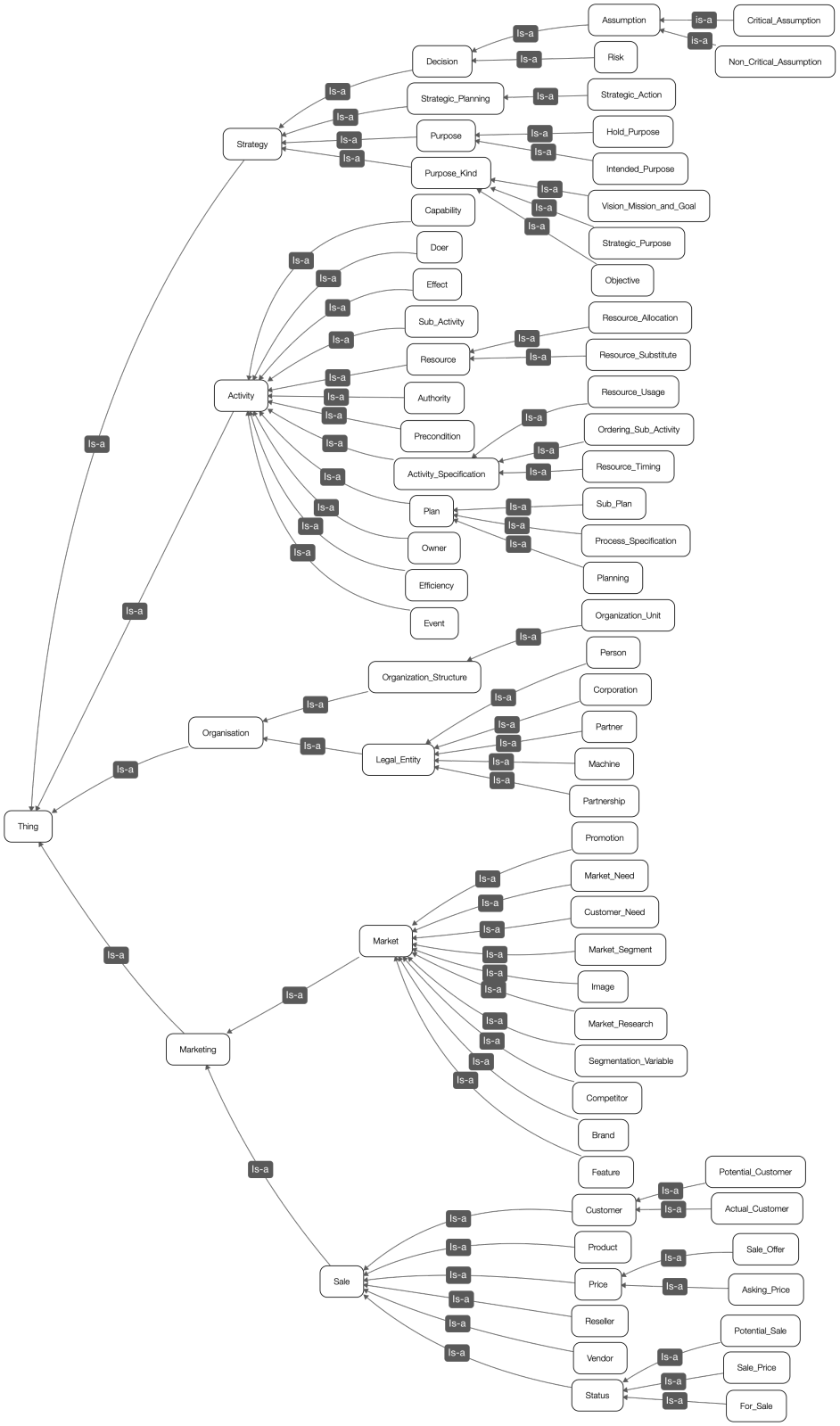

Figure 2.8 – Main concepts taken from the Enterprise Ontology [23]

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Figure 2.9 – Main concepts taken from the ISA S-95 ontology

part is associated to a resource or an equipment and considered as a black-box. Turning this black-box description to a white-box one, providing more knowledge on the production process and a fine-grained control of the equipment requires integrating a Cyber Physical System model or a dedicated architecture to manage IoT equipments.

2.4.2 IoT Ontology

The concept of IoT (Internet of Things), whose target is to connect things together through the Internet, is paid attention by more and more enterprises and researchers. IoT Systems are composed of physical objects and virtual objects where both objects mean “things" in “Internet of Things". It is with broad use in the industry, for example, smart city, manufacturing, intelligent transportation and so on. In order to design a

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

complete IoT system, some key technologies: system technology, communication technology and information technology, must be taken into account. Basic IoT characteristics IoT should include System Characteristics, IoT Service Characteristics, IoT Component Characteristics, Compatibility, Usability, Reliability, Security & Privacy, etc. IoT services are connected to physical IoT devices. Thus, composing IoT services means integrating IoT objects as things in IoT are interconnected via network. All key properties of these devices must be considered.

In order to implement the composition of IoT devices or their services, main characteristics of them must be taken into consideration. Since ontologies can describe the main concepts of targeted objects, ontologies for IoT devices, such as sensors, actuators, are necessary to promote the communication smoothly among heterogeneous IoT devices, and facilitate the integration into platforms and/or applications.

There are already numerous ontologies mostly attained from IoT projects supported and funded by individual or federated countries or consortium (i.e. US, Europe, W3C (World Wide Web Consortium), OGC (Open Geospatial Consortium)).

We have identified ontologies classification criteria from the perspectives of target domain and device view to research and analyze typical IoT ontologies. The details of category are shown (see Fig. 2.10). Based on the criterion of target domain, IoT ontology is either generic or domain-specific one. Precisely, generic IoT ontologies are grouped into generic IoT device ontology, generic IoT middleware ontology and generic IoT application ontology according to the research objects of them. IoT device is either actuator or sensor which has two connection types, wireless connection and wired connection. Considering IoT application domain, domain-specific IoT ontologies are partitioned by fields, such as marine, transportation and logistics, health, agriculture, and so on. From the angles of device view, IoT ontology is either physical, or logical, or compound (physical and logical). Physical IoT ontologies mainly delineate IoT device physical capabilities (i.e., memory, CPU (processor), connection, energy) or physical environment (i.e., location, physical measurand). Note that Logical IoT device ontology is primarily dedicated to non-physical descriptions, i.e., functionality specification, semantic data interoperability, taxonomy, discovery, composition, etc. Compound IoT device ontology is principally characterized of key features owned by both physical and logical IoT device ontologies.

We have identified 12 IoT ontologies listed in 1^{st} column of table 2.2. In the following, a brief introduction of them is given.

- Generic IoT ontologies:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

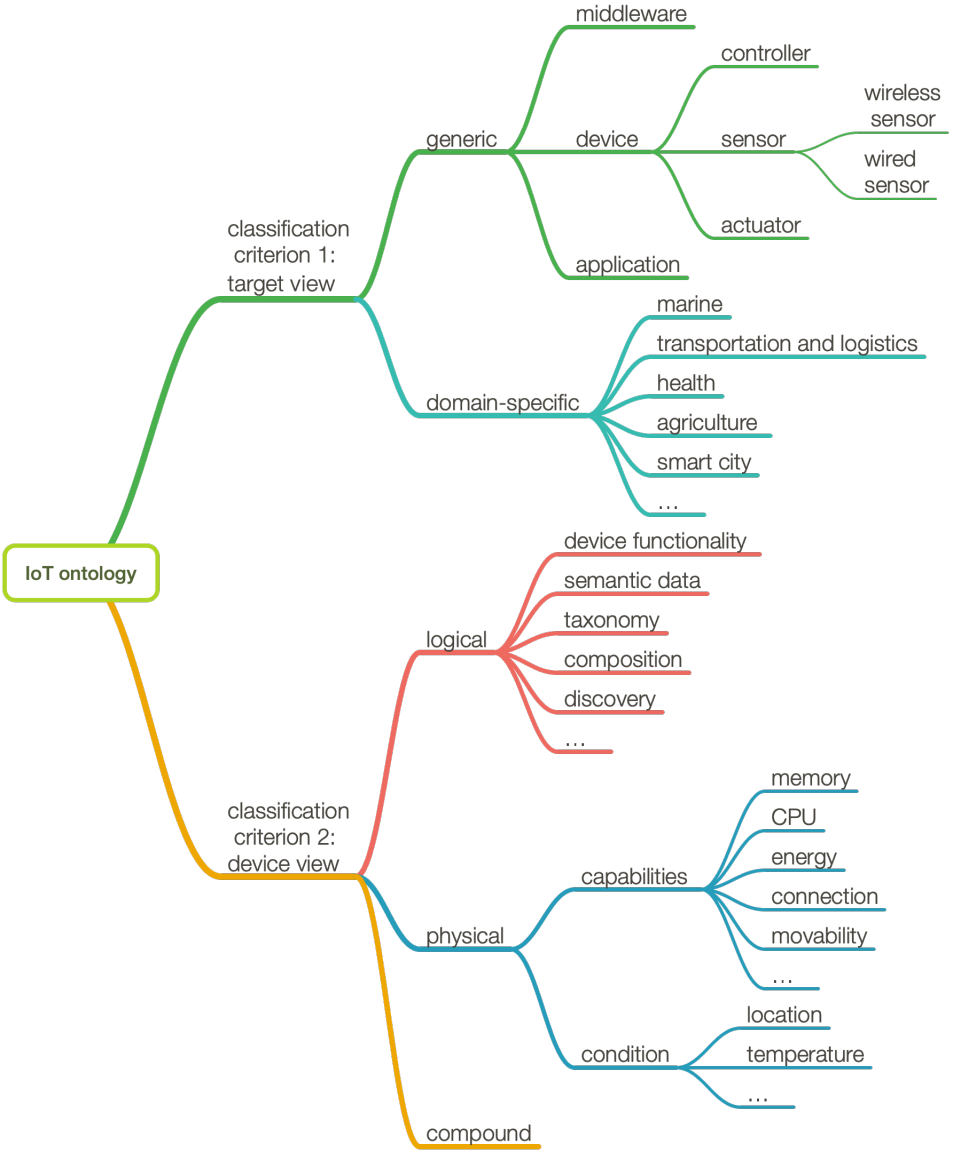

Figure 2.10 – IoT ontology classification criteria [56]

- SensorML Processes is a standard logical and generic ontology designed by OGC SWE (Sensor Web Enablement). This generic ontology is reused by other ones such as CISRO (The Commonwealth Scientific and Industrial Research Organization Sensor ontology) [68], OntoSensor [79], MMI (Marine Metadata Interoperability) device ontology [38]. Its main orignality relies on the introduction of the process concept [9], [77].

- CSIRO sensor ontology is a generic and compound IoT ontology focused on sensors. It introduces different key concepts such as sensor, sensing ground (location), domain of sensing (feature), operational model and process. However, sensing ground just refers to physical quality, it lacks of details of phys-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Table 2.2 – IoT ontology

Ontology Name	Sensor actuator controller	Physical	Logical
FIPA device ontology	sensor actuator	capability	
CSIRO SENSOR ontology	sensor	environment	functionality
ONTOSENSOR	sensor	capability	functionality
SWAMO ontology	sensor		composition
SDO	sensor		semantic data
MMI device ontology	sensor actuator		interoperability
SENSORML processes	sensor		functionality
CESN ontology	sensor	environment	semantic data
WISNO	sensor		semantic data
A3ME ontology	sensor actuator		taxonomy discovery
ONTONYM SENSOR	application(sensor)	environment	functionality
SSN ontology	sensor actuator	environment	functionality

ical capabilities.

- OntoSensor is also a generic and compound IoT ontology focused on sensors. It is extended to standard ontologies (i.e. IEEE Suggested Upper Merged Ontology (SUMO), International Organization for Standardization (ISO) 19115 standards) and thus compatible with other semantic web ontologies. Meanwhile, physical characteristics such as mass, radio frequencies, dimensions, and supply information for the platform are taken into account [37].

- Agent-based ontologies where device is seen as an agent:

  - SWAMO (Sensor Web for Autonomous Mission Operations) ontology is designed to solve interoperability challenges and allow large scale system-wide resources sharing as well as making distributed decisions and achieving autonomic operations [92], [99]. Although it integrates real time information, such as position, orientation, it is mostly devoted to a logical organisation of the system, physical information such as the physical location remaining rather limited.

  - A3ME (Agent-based Middleware Approach for Mixed Mode Environments) ontology is a basic classification for self-description and discovery of devices [45], [44]. Devices are classified as tag, mote, mobile, workstation, Server, vehicle and multimedia. General capabilities classes are namely sensor, actuator, communication, storage, computing, energy. In addition, it also refers to other two other IoT device ontologies, FIPA Device Ontology and OntoSensor.

  - FIPA (Foundation for Intelligent Physical Agents) ontology is a physical and generic ontology fully focused on devices. It characterizes devices according

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

to both hardware and software dimensions. Hardware description contains specifications related to connection, memory, User Interface (screen) and their sub-specifications. Software information incorporates descriptions of OS (operating system) and platform.

- Ontologies describing semantically sensors data to manage interoperability:

  - MMI (Marine Metadata Interoperability) device ontology is designed for the marine domain, describing both sensors and actuators. It pays a particular attention to their interoperability for the collaborations of involved devices.
  - SDO (sensor data ontology) is devoted to gaining the interoperability of sensor data from distributed and heterogeneous sensor networks [26]. In this ontology, data can be calibration, format, or parameter. Some property names, such as “Calibartion_Of_Type", “parameter_CanBe",“Format_Has", and their ranges are defined. For instance, “Calibartion_Of_Type" can be a “Curve", "Frequency_Response", or "Table". A “Format_Has" is either a “Physical_Unit" or “prototye". It is a part of Suggested Upper Merged Ontology (SUMO) where a sensor hierarchy ontology and extension plug-ins ontologies are included as well[27].
  - CESN ontology (Coastal Environment Sensor Network) is created from the “CESN Semantic Data Reasoner" project and aimed at describing the relationships between sensors and their measurements[10]. The main concepts found in this ontology are similar to the terminology described in SensorML.
  - WISNO (Wireless Sensor Network Ontology) is dedicated for processing wireless sensor networks data for communicating adaptively among anomalous and heterogeneous applications [46]. For example, when the density of smoke is abnormal, the adaptive action will be triggered to deal with it as quickly as possible.

- Full-fledged ontologies:

  - Ontonym is a set of ontologies representing core concepts in pervasive computing (time, location, people, sensing, provenance, events, device, resource) [24], [84].
  - SSN (Semantic Sensor Network) ontology takes into account sensors and their observations, the involved procedures, the studied features of interest, the samples, and the observed properties, as well as actuators. Its core ontology

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

is called SOSA (Sensor, Observation, Sample, and Actuator). It is a comprehensively generic device-cantered and compound ontology supporting a wide range of applications and use cases as it gathers advantages and discards disadvantages from all previously presented ontologies.

2.4.3 Conclusion

Business process and enterprise ontologies describe the collaborations between different partners horizontally and integrations of business layer and system layer vertically. These ontologies together with Service Oriented Architecture model show that artifacts of top layers in the enterprise architecture or cloud manufacturing architecture are rather complete. Entering into bottom layers, IoT layer is so critical that it not only support the upper layers with related IoT services but also links the physical devices. Although, IoT ontologies provide full descriptions of sensors and actuators mainly, including their NFPs related to physical environment or devices themselves, there is no unified one integrating these key NFPs. Furthermore, none of these ontology allows describing globally the control system associated to a production process, nor any application using smart devices.

2.5 IoT-based Architectures

IoT takes advantage of service-based and Internet technologies to extend traditional embedded and control devices with strong interconnection abilities. Efficient integration and management of Cyber Physical Systems in distributed control application require a dedicated architecture to couple these CPS description to higher level service interface, focusing on the way these CPS exchange information with other components. To reach this goal the NIST has proposed an IoT reference architecture. By this way, CPS can be integrated in more complex service oriented control architectures.

2.5.1 NIST IoT Reference Architecture

Typical designs and implementations of architecture of IoT is 3–tier architecture, each layer: IoT devices, gateway, and cloud [76], [82]. IoT devices layer, consisting of sensors and actuators, is responsible for collecting data, monitoring environment, operating instructions and so on. The tier of Gateway perceived as a node in the internet is possessed of abilities of communicating with other nodes or platforms, exchanging or sharing information, accessing to physical devices, controlling the data flow, etc. Cloud, including

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

front-end servers and back-end applications provides large storage abilities requested by the large amount of data collected from the different nodes. It also provides the necessary computing capabilities to analyze data, make decisions, predict trends... A survey of specific IoT architecture can be analyzed from the aspects of RFID (Radio Frequency Identification), SOA (Service Oriented Architecture), WSN (Wireless Sensor Network), supply chain management and industry, health care, smart society, cloud service and management, social computing, security and observation [75]. The author also summarizes the existing IoT cloud platforms[74].

IoT system can be described by IoT Conceptual Model (CM), IoT Reference Model (RM), and IoT Reference architecture (RA). Hence, IoT CM, RM and RA for IoT should be discussed necessarily.

In NIST IoT reference architecture[5], IoT CM (Conceptual Model) provides common structure and definitions for describing the concepts of and relationships among the entities within IoT systems. Key entities of this IoT conceptual model are IoT-user, digital entity, service, component, virtual entity, digital user, human user, endpoint, application, IoT device, physical entity, network, IoT gateway, actuator and sensor.

IoT CM provides common structure and definitions for describing the concepts of and relationships among the entities within IoT systems. It must be generic, abstract and simple. All key entities: IoT-User, Digital Entity, Service, Component, Virtual Entity, Digital User, Human User, Endpoint, Application, IoT Device, Physical Entity, Network, IoT Gateway, Actuator and Sensor, are provided in IoT CM. The relationships among them are delineated in Fig. 2.11. We can categorize these entities into different groups. IoT device and physical entity belong to physical part. IoT user (either digital user or human user) is in the user part. Endpoint, network and IoT gateway exist in the communication part. Service, application and component are proposed from usage control view (usage control part). Meanwhile, service, component and digital entity are classified into digital twin part.

IoT RM is divided into two aspects: entity-based RM and domain-based RM. In entity-based RM, entities are Application Service System, Operation & Management System, Resource & Interchange System, IoT User, IoT Gateway, IoT Device, Network, Physical Entity. One domain includes one or more entities. In domain-based RM, there are User Domain (EUD), Application Service Domain (ASD), Operation & Management Domain (OMD), Resource & Interchange Domain (RID), Sensing & Actuating Domain (SAD) and Physical Entity Domain (PED). The corresponding relationship between the entity-based RM and the domains in the domain-based RM is specified in table 2.3.

[5]ISO/IEC 30141: Internet of Things (IoT) – Reference Architecture

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

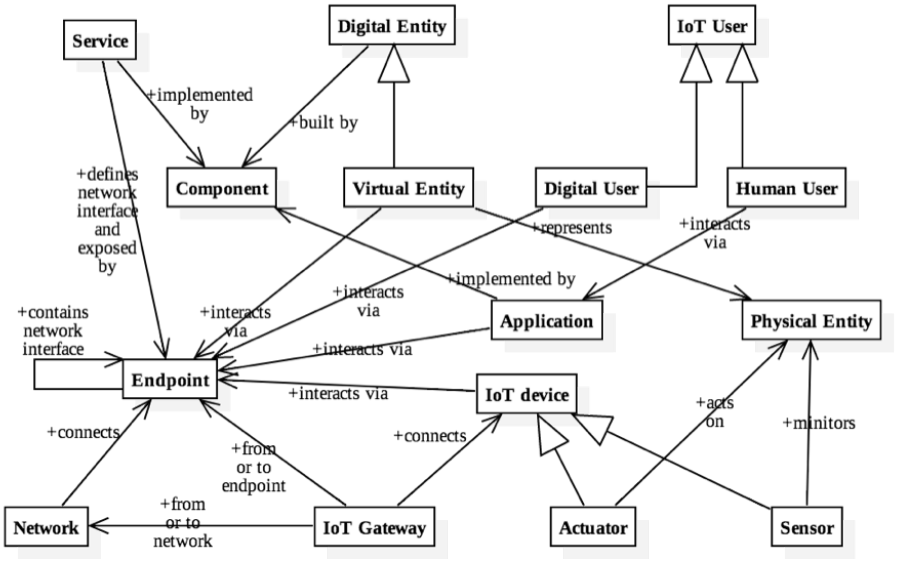

Figure 2.11 – Excerpt of NIST IoT RA big picture for IoT concepts of CM

Table 2.3 – Excerpt of relationship between entity and domain (NIST IoT RA p49)

Entity in Entity-based RM	**Domains in Domain-based RM**
IoT User	User Domain (EUD)
Application Service System	Application Service Domain (ASD)
Operation & Management System	Operation & Management Domain (OMD)
Resource & Interchange System	Resource & Interchange Domain (RID)
IoT Device	Sensing & Actuating Domain (SAD)
IoT Gateway	
Physical Entity	Physical Entity Domain (PED)
Network	Communication and interactions among domains

The IoT RA is with the goals of describing the characteristics and general requirements of IoT systems, defining the IoT system domains, representing the Concept Model (CM) and Reference Model (RM) of IoT systems and showing the interoperability of IoT system's entities. It is directly originated from the domain based IoT RM based on entity based IoT RM, which is evolved from IoT CM. For the time being, there are five main IoT RA views which are IoT RA Functional View, IoT RA System View, IoT RA Communications View, IoT RA Information View and IoT RA Usage View, illustrating the IoT RA. The function view is a technology-neutral view of the functions necessary to form a system (see Fig. 2.12). The system view specifies the generic functional devices and systems in each domain to form an IoT ecosystem and support of functions components in the functional view. The communication view describes concepts for handling the complexity of communication in heterogeneous IoT context. The information view analyses the entities from the views of data (data source, data carrier, data destination, data classification). The usage view emphasizes the concepts of roles, activities, parties,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

etc.

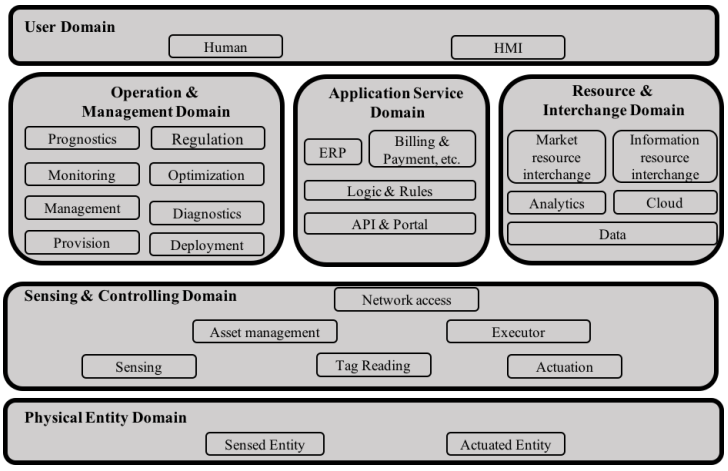

Figure 2.12 – Excerpt of IoT RA Functional View (NIST IoT RA p49)

In NIST IoT RA, sensing and controlling domain or system (see Fig. 2.12) serves as a whole. So the detail description of IoT services themselves (sensing service, controlling service and actuating service), of their relationships, of how they are composed to work as a middleware layer are missing. Instead, Fig. 2.12 just shows that sensing, actuation, executor and network are inevitable parts of sensing and controlling domain.

2.5.2 “Service" Oriented Control Architecture

Industrial automation system persuasively involves physical devices (such as sensors and actuators.). As a matter of fact, a technological standard of SOA is web service owning three cores, Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration(UDDI). The problem of how to virtualize the physical devices into corresponding web service was to be solved years ago via cyber physical system technology (CPS). Device Profile for Web Service (DPWS) language is dedicated to model physical objects to provide abstract logical web service [52][48]. [21] proposes a service oriented device architecture (SODA) integrating device and SOA. Real service oriented control architecture are implemented in two European projects SIRENA[6] and SOCRADES[7]. In the paper[62], a survey on service oriented engineering for automation systems is mainly focused on how to connect “loosely-coupled" entities together from aspects of 1) semantic web service and ontology, 2) modeling, orchestration and choreograph, 3) service composition, 4) analysis and simulation, and 5) collaboration.

[6]http://www.itea3.org
[7]http://www.socrades.net

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Semantic web markup and in particular the Web Ontology Language (OWL-S) is leveraged to assist in automating service selection, invocation and composition in manufacturing [18]. To build a service oriented control architecture [61] for reconfigurable production system is, the architecture applies process model for service composition, Petri net model for logical controller, event notification pattern for communication. [43] integrates a Multi Agent System (MAS) with web service to implement a decision making entity in manufacturing system.

Although the idea of SOA is integrated into control system by some control architectures mentioned above, their concentration is still on the level of separated device. In the roadmap of service oriented control architecture, some experts use service description to hide the physical device behind for automation service selection. However, web service description via DPWS does not give enough focus on the NFPs of physical constraints. These control architectures only implement the specific controllers that can serve in a remote place.

2.5.3 Conclusion

NIST IoT reference architecture provides the big picture for IoT and different viewpoints for it. Nevertheless, the control descriptions from the automation system view are missing. Existing service oriented control architecture aimed at solving the problems such as reconfiguration essentially implements a networked control system where device is considered as a service. What we required is a cloud control system or control service, instead of networked control system. So a new thoroughly distributed control model and/or architecture supporting control service is needed.

2.6 Control as a Service (CaaS)

Evolvable and collaborative industrial systems enhancing flexibility and automatic configuration to meet mass customized and rapidly changing user requirements are becoming an emergent paradigm. Service oriented architecture (SOA) is the abstract concept of a software architecture, where the focus is the offer, search and sue of services over the network, providing a communication platform, based on open protocols, addressing the interoperability in heterogeneous system. Consequently, the introduction of SOA into industrial system can bring service characteristics, e.g., distributed, reusable, loosely coupled, etc, to automation systems. Cloud control system has evolved from networked control system (NCS). NCS is developed from the originally physical one as the communication way is changed to network from wired connections. Due to the prominent

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

influence of cloud computing, cloud control system (CCS) with the support of control service or Control as a Service (CaaS) will be the next generation of control system. In this section, the overviews of network control system, servitization of control and cloud control steps including service selection and composition, are conducted.

2.6.1 Networked Control System (NCS)

Networked control system(NCS) can be defined as a system whose control loops are closed through communication networks such that both control signals and feedback signals can be exchanged among system components (sensors, controllers, actuators and so on)[116]. Generally, there are three configurations of NCSs, i.e., centralized configuration, decentralized configuration, and distributed configuration[32]. In the centralized NCS is only a single feedback loop where all sensors and actuators are connected through one controller (see Fig. 2.13). A decentralized Network Control System integrates several cooperating controllers, each of them linking senors and actuators. As stability is the most fundamental performance of a control system, analysis of NCS is accomplished by Zhang et al [114] focused on the solutions for NCS fundamental issues, e.g., network-induced delay, dropping network packet, and by Walsh et al [97] who introduce a novel control network protocol (try-once-discard (TOD)) for multiple inputs and multiple outputs (MIMO) NCSs. The scheduling research on NCS is conducted in the paper [96], given sufficient network speed. Tipsuwan et al [89] reviews NCS control methodologies, i.e., augmented deterministic discrete-time model methodology, queuing methodology, optimal stochastic control methodology, perturbation methodology, sampling time scheduling methodology, robust control methodology, fuzzy logic modulation methodology, event-based methodology and end-user control adaptation methodology. Event-triggered method applied to solve the problems of NCS one-step packet dropout [102] and nonlinear NCS [115] is able to reduce the communication traffic as it will take action when needed. Yuan et al [107] proposes a delta operator approach to get the optimal control in NCS considering the disturbances.

2.6.2 Servitization of Control

Similar to sensing service and actuating service, control service is supported by the whole control system for one or more given control objectives. Nowadays, moving controller to cloud become a trend in many domains (e.g., automotive [86]) since more and more sensing services and actuating services are available in the IoT context and online control commands are optimal calculated based on global data in cloud. Virtualized

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

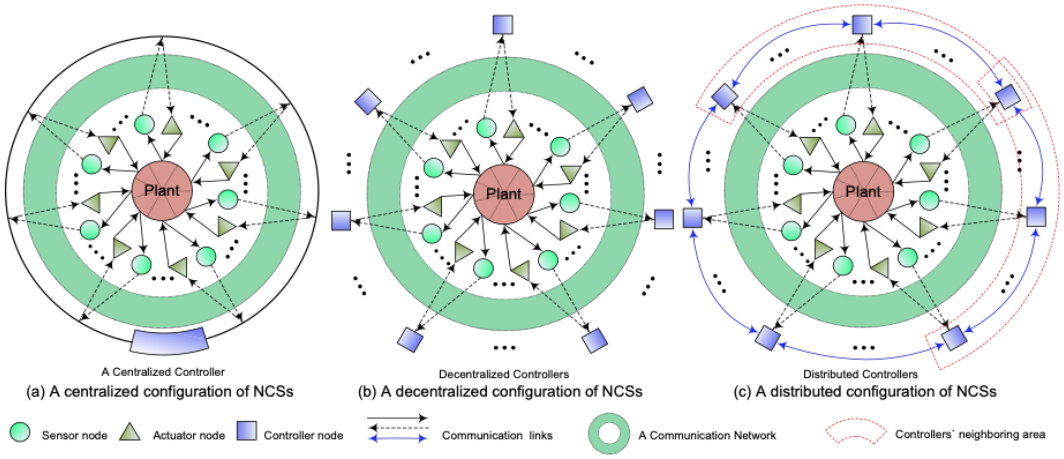

Figure 2.13 – Three general configurations of NCSs [32]

PLC controlling service is gained and abstracted from physical PLC controller [34], [6]. A typical use case for monitoring and controlling the soil environment for the growth of grains is analyzed according to acquired kinds of sensor data [47]. Another global control on IoT service redundancy is implemented in the context of IoT Wireless Sensor Networks (WSNs) [40]. Some control services are characterized thanks to classical control patterns, such as process control (including continuous control and distributed control system, [58], [120], [85], adaptive control, optimization control, and even heuristic way [16].

Based on our state of the art review, main characteristics used to compare control services in table 2.4 are controlled object, controlled variable, control objective, control pattern, sensing service, actuating service, interface type, implementation. Thus, control service could be described using these features. Control service in the first two literatures represent the whole control system while it in the other only means the controller. To distinguish it, we define control service is virtualized from Cloud Control System (CCS) while controlling service is rooted from controller. From the table, we notice the difference control service are involved in sensing and actuating services. In addition, Whether a physical controller does exist or not behind a controlling service is dependent on the service itself.

2.6.3 Building a Cloud Control System Based on Control Services

In the networked control system, we conduct our research by creating new or improving existing algorithms to make sure the control performance especially for stability under various situations. Using controlling service changes the way control elements are connected from networked devices to services. However when there are quantities of IoT

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Table 2.4 – Control Service

Control service	Controlled object	Controlled variable	Control objective	Sensor	Actuator	Interface type	implementation
1[34]	general object	general property	general objective	Yes	Yes	physical/ logical	virtualized from PLC controller/ cloud based
2[47]	Soil	soil environment (humidity)	keep optimal	Yes	Yes (pump)	physical/ logical	software component/ cloud based architecture
3[6]	general object	general property	general objective	No	No	physical/ logical	virtualized from PLC controller/ cloud based
4[40]	IoT services	redundancy	decrease	No	No	logical	software component
5[58]	process	sequence	keep optimal	No	No	logical	software component
6[120]	manufacturing process and product	sequence and quality	operate well	No	No	logical	software component
7[85]	multi stage service operations		optimize globally	No	No	logical	software component
8[16]	control station	action	run in best way	No	No	logical	software component

services available, we need to pay attention to the second stage of control service: cloud control. This step involves in the selection, composition and orchestration of control elements, i.e., sensing, controlling and actuating services. Service orchestration implementation relies on the enterprise service bus discussed previously. In this subsection, service selection, service composition and late binding providing the dynamic feature for them, are reviewed.

2.6.3.1 Service Selection

Accurate service description of non functional properties (NFPs, i.e., availability, channels, charging style, settlement, quality of service, security and trust, ownership and rights) can be used to improve service discovery, service substitution, service composition and management as specified in [25]. Nevertheless, several NFPs are defined, providing more or less details on IoT devices or services. Several ontologies have been defined for sensors and actuators , such as FIPA device ontology, CSIRO SENSOR ontology [68], ONTOSENSOR [79], SWAMO ontology [92], [99] SDO [26], [27], MMI device ontology Ontology [38], SENSORML processes [9], [77], CESN ontology [10], WISNO [46], A3ME ontology [45], [44], ONTONYM – SENSOR [24], SSN ontology. As synthetized in table 2.5 , some of them propose Non Functional Property (NFP) descriptions.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Table 2.5 – Non Functional Properties (NFPs).

Name	Ontology	Sensor/actuator /controller	Description
MeasurementRange	SSN	sensor	The set of values that the Sensor can return as the Result of an Observation under the defined Conditions with the defined system properties.
ActuationRange	SSN	actuator	The range of Property values that can be the Result of an Actuation
Accuracy	SSN	sensor actuator	The closeness of agreement between the Result of an Observation (resp. the command of an Actuation) and the true value of the observed Observable Property (resp. of the acted on Actuatable Property).
Frequency	SSN	sensor actuator	The smallest possible time between one Observation, Actuation
Latency	SSN	sensor actuator	The time between a command for an Observation (resp. Actuation) and the Sensor providing a Result (resp. the Actuator operating the Actuation)
Precision	SSN	sensor actuator	As a Sensor Property: the closeness of agreement between replicate Observations on an unchanged or similar Property value: i.e., a measure of a Sensor's ability to consistently reproduce an Observation. As an Actuator Property: the closeness of agreement between replicate Actuations for an unchanged or similar command: i.e., a measure of an Actuator's ability to consistently reproduce an Actuation
Resolution	SSN	sensor actuator	As a Sensor Property: the smallest difference in the value of an Observable Property being observed that would result in perceptably different values of Observation Results, under the defined Conditions. As an Actuator Property: the smallest difference in the value of an Actuation command that would result in a value change of the Actuatable Property being acted on.
ResponseTime	SSN	sensor actuator	As a Sensor Property: the time between a (step) change in the value of an observed Observable Property and a Sensor (possibly with specified error) 'settling' on an observed value, under the defined Conditions. As an Actuator property: the time between a (step) change in the command of an Actuator and the 'settling' of the value of the acted on Actuatable Property, under the defined Conditions.
SystemLifetime	SSN	sensor actuator	Total useful life of a System (expressed as total life since manufacture, time in use, number of operations, etc.)
BatteryLifetime	SSN	sensor actuator	Total useful life of a System's battery in the specified Conditions.
location	CISRO	sensor	
Memory	FIPA	sensor actuator controller	The type of the central processing unit that the device has.
CPU	FIPA	sensor actuator controller	The type of the central processing unit that the device has.
connection	FIPA	sensor actuator controller	The type of the connection the device uses.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Gathering these different NFP definition may allow capturing contextual information.

With the fast development and adoption of IoT devices, the Cloud of Things (CoT) emerge, extending the resource sharing model used for infrastructures (IaaS), platforms (PaaS) and software (SaaS) [22] to objects (OaaS) [4]. Meanwhile, some new conceptions, for instance, object as a service (OaaS), sensor as a service (SenaaS), are proposed. CoT provides us an idea of or a model for facilitating the procedures of using physical things or devices from the cloud end when physical things are becoming smart and thus possessed of abilities of sensing, actuating, communicating. Service composition where CoT are involved was and is still a hot research area and the performance of dynamicity of is dependent on the late binding technology.

2.6.3.2 Service composition

As far as traditional IT process are concerned, Business Process models are analyzed to select and/or compose services to support a particular business function. While traditional service composition relies on a dedicated engine analyzing control and data flows to discover and select concrete services exhibiting the requested properties [30], several abstract service based composition models are still developed to enhance the adaptation to user-intentions [28]. In the cloud or web environment, service composition will be unavoidable when one expects to reuse others' services or integrate her or his several sub ones. It is obvious that web service composition is dependent on service requirements, both functional requirements and non-functional requirements. Based on requirements, some concrete services are selected taking into account their functionalities, policies and contracts. Meanwhile, this requires properties of each service involved, such as service reachability, service interaction (in both information and behavior way) when selecting them at the design and running time. Service composition in IoT environment or for Cloud of Things (CoT) is still a concern of great importance and need to be paid more attention because of physical constraints being taken into consideration. General requirements for composing service from smart objects are summarized, namely, resource constraint, low-power and loss communication link, power efficiency, data/event-driven services, asynchrony, discovery, management requirements, and Quality of service (QoS) awareness [41]. The difference of classical web service and IoT service is that the latter pays more attention to the physical constraint. In IoT control context, non functional properties (NFPs) should be taken into consideration due to physical constraints from device behind the service. To achieve features of automation and flexibility, composition rules based on abstract services for a better adaptation to user intentions are proposed

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[28]. Service composition process could be simply divided into two steps, 1) an engine (or a schema, or a model, or an architecture, etc.) responsible for specifying the control and data flows between services, 2) concrete services with logical and discoverable property [30]. Some web composition solutions are discussed in [64], and part of them can be applied in industrial automation field [60]. In order to adapt to quick change of ambient environment, dynamicity ought to be attached into service composition.

2.6.3.3 Late Binding

Dynamicity requires late binding, i.e. being able to select the “best" service depending on the current context at runtime, implemented as a part of service composition/orchestration. For a given process composition chain, each process can be instantiated by a concrete service at design time or running time. This binding step can be achieved statically (static binding) or dynamically (late binding strategy integrating execution information to select the most convenient service). Compared to early binding, late binding increases adaptability, agility and robustness whereas it requires (1) extra-monitoring services to capture quality of service parameters (e.g., availability, response time, precision) to detect failed services (compared to the current SLA), identify surrounding environment changes and, (2) providing an efficient service selection based on a multi-criteria evaluation of Quality of Service preferences. Several research works have already addressed these requirements:

- Different works have paid attention to the QoS management and SLA violation. [78] proposes to formalize QoS aggregation as algebraic expressions. Composite QoS analysis used to integrate late binding in service composition is illustrated in [14], [50], [11], [12], [67]. identifies SLA violation (e.g., web service failure, web service performance degradation, etc.) to update automatically processes deployment whereas [39] integrates non-functional properties related to communication in service composition achieved in dynamic ad hoc environments.

- As far as selection is concerned, a LCP-net formalism capable of expressing qualitatively user preference of different QoS can be used [17]. Some other methods are also applied for this NP hard problem such as Integer Programming [110], or Genetic Algorithm [12].

- Considering the way late binding is integrated in the composition process, the service planning problem is solved by using Hierarchical Task Network and Partial Order Casual Link planning techniques [8]. Artificial Intelligence (AI) planning

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

and scheduling for workflow is analysed in [72]. Re-planning problem is mentioned and triggered by giving an example of response time in [13].

2.6.4 Conclusion

In this Control as a Service section, we firstly review the cloud control system (CCS) predecessor: networked control system (NCS). Researches on NCS mostly pay attention on controller design to make sure NCS stability in many cases, such as network-induced delay, data packet dropout, to mention a few. This provides a strong basis as the first phase of CCS and a prerequisite for our dissertation. Then, we summarize part of existing control and controlling services. These services are not generic but accurate. A generic description of CCS including controlling services should be provided. The researches on the second phase: cloud control depends on the selection, composition and orchestration. Selecting and composing IoT services can not neglect the NFPs related to them.

2.7 Conclusion

Industry 4.0 shows a distributed and collaborative environment, and requires an advance in all aspects: i.e., standards, smart devices, integration, algorithms, models, architectures, frameworks, platforms ... Enterprise architecture plays a critical role in assuring the operations while increasing the efficiency and lowering the cost. Enterprise information system makes possible the success of interactions among various components to add the value for the enterprises, especially for the small medium enterprises. Cloud manufacturing helps to achieve the intelligent production and smart factory, taking advantage of quantities of and numerous types of smart devices and intelligent Cyber-physical system.

Focusing on the Information System side, Business Processes are defined logically and enterprise ontology or ISA S95 provides consistent description of the main information. Moreover, Service Oriented Architecture has lead to more flexible and agile Information systems as services can be invoked in a loosely coupled way. Despite their interest these works do not integrate lower-level devices such as basic CPS systems. Entering into IoT realm, control processes are not paid enough attention in any IoT ontology or architecture. From the view of automation processes, networked control system is targeted at solving the controller design in networked context. However, this model needs to be adapted to fit the Cloud context and services defined in this Service oriented Control Architecture must be extended as it only relies on sensors and actuators

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

“digital service" twin. Sensors and actuators can be substituted by sensing and actuating services separately. Controller or the whole control system can be also replaced with a controlling service and control service thanks to the servilization of control. These controlling or control services are so specific that they can not used in a generic way.

Hence, a new model for control system should be created. This control service model should avoid existing IoT ontology limits besides the neglect of control element. There is no IoT ontology taking into account all physical constraints. Control service is an composite IoT service. As a result it needs to integrate these key NFPs. Moreover, it ought to take into consideration the interaction way while traditional ontologies pays attention to the separated devices. After the describing the control service model, we need to build a control service. Consequently, a real service oriented control architecture should be made to develop a context aware and event driven cloud control system. This architecture is involved in selecting, composing and orchestrating sensing, actuating and controlling services. Previous selection and composition of web service do not taking into account key NFPs related to devices. Enterprise service bus can not provide adaptiveness that control service elements orchestration intends to have.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Chapter 3

Control as a Service Model

Contents

3.1 Introduction

Industry 4.0 requires control system to be more agility and flexibility to meet users' customized requirements. Consequently, traditional control system should be innovated via shifting the connection way from wired to wireless. This leads to the networked control system. However, the current collaborative manufacturing environment requires agile and reusable systems, taking advantage of the loosely coupled feature provided

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

by Service Oriented Architecture. This involves extending SOA to low level devices. Cloud control system that inherits the advantages of networked control system in IoT environment and owns the inherent features of cloud computing, is deemed as a new generation of control system. In a cloud control system, services are introduced to replace the original devices as they are possessed of agility, flexibility and reusability. To support the cloud control for the cloud control system or control service abstracted from it, new control service model that is able to ensure these characteristics above for a control system under the context of IoT and cloud computing, is required.

Control service model should mix both automation and service vision. Focusing on the functional description, automation view should be used whereas the SOA vision will contribute to the way services will be selected, composed and orchestrated. This is why, we use the control vision to set the control service functional description. Before the discussion of control service functional model, controller and control system models are introduced firstly according to the experiences from the automatic control system. In the control system model, entities of sensor, actuator, controller, controlled object, controlled variable, environment control pattern, control requirements, and their relationships of them are stated. In order to elaborate the core of control system, a controller model is developed from the perspectives of input and output. Moreover, definition of control pattern is specified in a recursive way using the mathematical function representation. Entering into functionality of control service model, physical devices are replaced by their correlated services. specifically, sensor, actuator and controller are substituted by sensing, actuating and controlling services. At the end, a basic service based control model is built from the automation view.

From the service side, traditional web service approach pays attention on its functional descriptions, QoSs (Quality of Services) and securities. Service based control model is also attained from the functional view. Nevertheless, control service is a composite IoT service so that it connects the physical devices closely. NFPs related with these physical devices (i.e., sensor, actuator and even controller) must be integrated in the control service description. The different ontologies we reviewed in the state of the art section (see table 2.5) allow describing most of these non functional requirements and properties. However, we find that some of them (e.g., precision and accuracy) are similar. These original NFPs are processed to keep some key NFPs. These nine vital NFPs, i.e., frequency, range, precision, system life time, memory, CPU, location, connection media, transmission latency, are listed in the table of NFPs processed. They are grouped into four categories, i.e., physical environment, embedded device performance, communication and system capability, to ease the management of them. These

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

NFPs are used to describe control service, specifically, sensing, controlling and actuating services, from a non functional view. They are considered as selection criteria of control service elements. Meanwhile, they defines a fundamental control service execution environment.

Existing ontologies provide us available NFPs, however, most of them just study the sensor and or actuator. Although some of them are concentrated on the generic device, few ontologies pay attention on control engineering from the IoT view. As a result, a full-fledged control ontology should be developed. Thanks to the sections of control service functional description and NFPs processed, our control ontology is achieved. In summary, in our control ontology there are three parts: service based control model, NFPs and event driven communication.

Conclusively, this chapter is mainly focused on the first research question that what is a control service. It is organized initially from the three motivating examples in the aspects of smart home, smart factory and smart transportation. Then, models from the various views, i.e., control service functional model, non functional requirements, event driven communication and control ontology integrating the former three parts, are built.

3.2 Motivating Examples

In this section, three motivating examples from the domains of smart home, smart factory and smart transportation are given to help to model the proposed control service. Smart home use case focuses on the distributed environment of control system. Attention is paid to the execution context awareness in the smart factory example. Smart transportation instance shows us the event driven requirement for the cloud control system.

3.2.1 Smart Home Motivating Example

Smart building or connected building as an application field of Internet of Things (IoT) is devoted to adding comfort, increasing security and saving energy. A building can be separated into rooms that belong to different types, such as office, data center, storage, production or shopping hall for business and commercial buildings or living room, bathroom, kitchen, garage, and so forth for houses and residential buildings. There are different types of sensors (e.g., an ultrasonic sensor detecting the soap level in the laundry, temperature sensor, humidity sensor, etc.) and actuators (such as motor for window or door, fan device able to dry the air) deployed in the building. Consequently,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

many control systems are required to connect and manage these sensors and actuators to provide better living or work experience as well as optimizing energy consumption.

Here a temperature control system for a room as a use case is analyzed to support the event manager and data manager evaluation. As the temperature of the room can be changed by many factors (e.g., weather condition, central boiler, room heating device, etc.), an ideal temperature control system may be too complicated to be controlled when taking into account all these factors. Trying to save energy, a user wants to design a new heating control strategy, taking advantage of the available IoT devices. The goal consists in stopping the heating device while a window is opened and adjusting the required temperature depending on the forecast external temperature (see Fig.3.1). In sensing service part, indoor sensing service acquiring the current room temperature may be supported by a physical temperature sensor. Outdoors sensing service (S3) collecting the outside real-time temperature may be a web service on the Internet. The sensing service (S2) detecting the state of window can help to make decisions to save energy. In the section of actuating service, heating or cooling actuating service receives a command variable limited to an input power scale. Here a control service is required to manage the behavior of involved actuating service based on sensing services concerned. Modeling such a control system involves: 1) coordinating different control services, each of them associated to a dedicated requirement, 2) setting a consistent data manager to integrate both sensing and Internet data as valuable inputs of control services and 3) managing events to decide if the control strategy need to ba adapted or not.

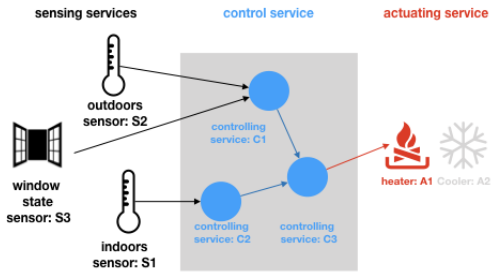

Figure 3.1 – Temperature control use case

3.2.2 Smart Factory Motivating Example

This motivating example is picked from a FabLab, where car-seat moulds prototype are produced. While using new material, moulds characteristics must be adapted, and injection and press processes must be tuned. Using sensors related to the environment (temperature, humidity), material quality are used as inputs to control injection and

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

press control processes. Actuating system A1 consists in opening/closing material injection, and deburring the parts. The process is shown Fig. 3.2. Different IoT devices and numeric machines are proposed to implement the different operations. Paying attention to the place the mould is manufactured and tested, appropriate sensing information should be sent to the control system. In this example C2 provides information on the material quality, C1 evaluates the environment context (temperature and humidity) and C3 provides the injection control. The cost for communication between controllers have to be considered. The locations, precision of sensors and actuators also have to be taken into account. To sum up, this use case calls for a context-aware control application.

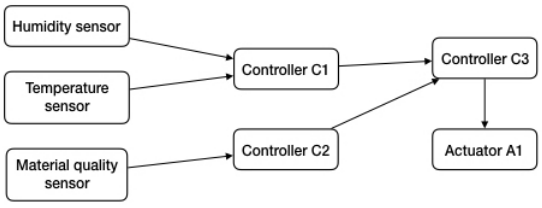

Figure 3.2 – Car-seat manufacturing example

3.2.3 Smart Transportation Motivating Example

Taking advantage of various interconnected objects allowing to capture various data such as traffic information, pollution, air quality, humidity, temperature..., smart cities aim at improving the well-being, providing more efficient transportation, allowing smarter services for people. By now, more and more information can be collected, stored in clouds, and used to develop new services or classical automation control. Focusing on smart mobility management, several sensors are deployed in streets to count cars crossing a given line; cameras can also provide pictures that can be used to identify potential traffic jams; traffic lights can be managed to select green light duration; dangerous car drivers behaviors such as red line crossing can also be detected. Moreover, extra information can be provided by cars themselves including their current position, forecasted trip... All these traffic management information can be worthy used to improve traffic regulation thanks to smarter traffic light control, increasing air global quality, emergency services efficiency and more globally the well-being. Such smart traffic control application can be designed locally, taking advantage of the huge amount of available data, coordinating these different subsystems to support more reactive and context-aware control. In what follows, we use a smart traffic control for emergency cars (see Fig. 3.3) with the intention of letting the emergency car get to the accident point as quickly as possible. A route planning controller runs to produce the route information based on current

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

position and destination while considering the traffic information detected by cameras. Emergency accident traffic light controller optimizes switch time of all involved traffic lights according to the real-time car position and dynamic route information and constraints of lights themselves. Such a system relies on a distributed control architecture, taking advantage of cloud services to integrate traffic mining components, image analysis with basic embedded traffic light control systems. It challenges for a new distributed and reactive control architecture allowing to design loosely coupled control applications, embedding various sensing and actuating devices in a unified interface and allowing a more reactive and event-based orchestration of the different control components.

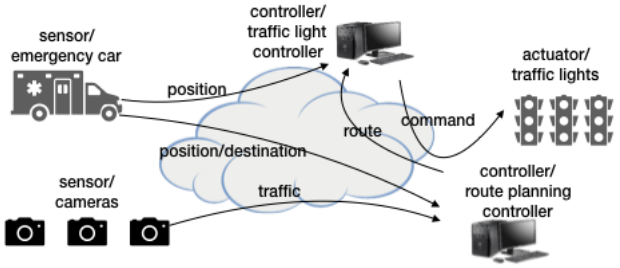

Figure 3.3 – Smart transportation motivating example

3.3 Control Service Functional Model

As shown in the three motivating examples, a new control model that is distributed, adaptive and reactive is required to adapt to the industry 4.0 requirement. This control model is aimed at refactoring the control projects leveraging the SOA, IoT, cloud computing and other technologies. This new model is inspired by business process ontology as automatic process is a specific one. Organizing a collaborative BP (Business Process) consists in identifying the different tasks and actors involved and then selecting the convenient IT services used to support each task and to exchange information between activities. As mentioned in the state of the art, several ontologies and standards are proposed to support semantic interoperability in a given business area: enterprise organisation can be used to describe the main actors and process organisation involved in business process specification. Connecting the enterprise ontology process specification to the ISA S95 work process segment capability, allows refining the BP specification by integrating the different production resources, integrating the physical assets capabilities. Finally, these physical assets and work process segment can be related to a control ontology, allowing to integrate the main characteristics issued from the different IoT ontologies (see Fig. 3.4).

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

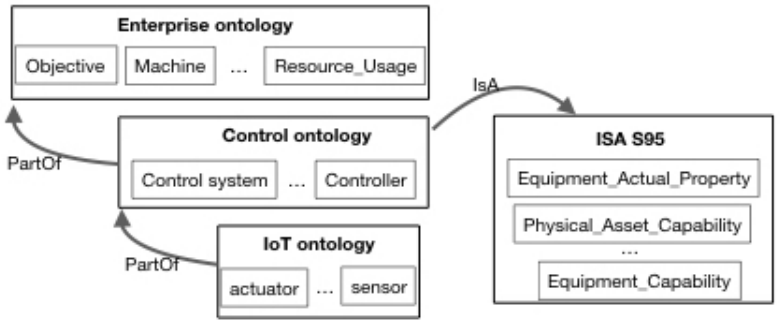

Figure 3.4 – Relationships between control ontology and other ontologies

Dedicated to building a control ontology that is able to provides a comprehensive understanding of control field in the era of IoT and cloud computing, the core of this ontology relies on the automation knowledge on automatic control system and its controller. Meanwhile, the characteristics that are introduced by services must be incorporated. Due to the physical features of related devices, i.e., sensors, actuators, controlled objects and controllers, non functional requirements are worthy of being considered when selecting and composing their upper layer services. Eventually, the behavior of control system at running time is specified via the event driven mechanism. In the next three sections, control service functional model, non functional requirements, and event driven mechanism will be discussed orderly.

3.3.1 Control System Model

Entering into a control system, we extract some key entities, such as controller, sensor, actuator, controlled object, controlled variable, controller pattern (or controller function) and control requirement(or control objective). Based on them, we build a control system model (see Fig. 3.5) from the point of view of computer science.

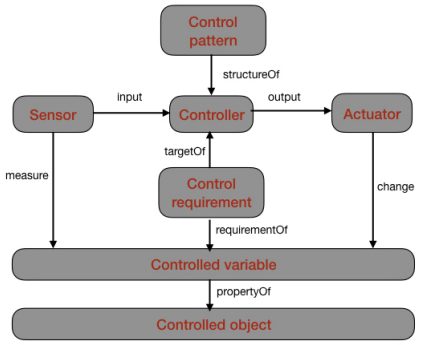

Figure 3.5 – Control system model

Focusing on the smart building use case, the temperature regulation example re-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

quires us to build a distributed control application as the formats of sensing services and/or the locations of all involved services can be heterogeneous. The main definitions of these critical concepts are following:

- **Controller**: is a device, historically using mechanical, hydraulic, pneumatic or electronic techniques often in combination, with a microprocessor or computer, which monitors and physically alters the operating conditions of a given dynamical system. A controller can be also a program implemented by a software component. A controller requires inputs from sensors and/or human beings. For instance, a user is necessary to set desired states as inputs expressed by user intention(s). Meanwhile, it will produce the control signal(s) as output(s) in order to manage the actuator(s). Its functionality is characterized of a control pattern.

- **Sensor**: is an electronic component, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other components, frequently a computer-based process. It is always used with other electronic devices, whether as simple as a light or as complex as a computer. It usually transforms the physical property into electrical signal. Sensing service provided by a sensor requires a controlled variable as input and output the measurement value of this controlled variable. From our smart building example, we use a temperature sensor.

- **Actuator**: is a physical component that can be integrated in a machine. The actuator is responsible for moving or controlling a mechanism, or a system, or a simple physical object. It provides the mechanism by which a control system acts upon an environment. The actuating service supported by an actuator requires a control signal from controller and source of energy. The control signal is relatively low energy. The source of power may be electric voltage or current, pneumatic or hydraulic pressure, or even human power. From our smart building use case, we integrate one heating system actuator

- **Controlled object**: any object or system to be controlled, e.g., a room in the smart home motivating example, a manufacturing process in the use case of smart factory, a car route in the smart transportation.

- **Controlled variable**: is related to a control object. It is defined via selecting one of the physical properties of a controlled object. Such physical property could be light, motion, temperature, magnetic fields, gravity, humidity, moisture, vibration, pressure, electrical fields, sound, and other physical aspects of the external

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

environment. The value of controlled variable is detected and changed by using sensing service and actuating service respectively. From our smart home use case, the temperature of the room is the controlled variable.

- **Control requirement**: is the goal of control service, for example, the lowest energy cost for the temperature control in the first example, an optimal or adaptive control operation in the second case, and the least of time in the third use case.

- **Control pattern**: is a model applied into control service for attaining the control objective. There is a plethora of control patterns, proportional–integral–derivative (PID) control, fuzzy control, to mention a few. Here, it is associated to control function for control system (control service) or controller function for controller (controlling service). Regarding the smart building use case and its temperature regulating control application, this control service itself is organized by composing different available controlling services C1, C2, and C3. Target temperature value can be an input of these sub controlling services. The expression of C1 (see equation.(3.3)) shows that its result will be 1 if window is closed and outside temperature is not approximately equal to target value. C2 can be a normal PD(Proportional and Derivative) control and its result is a command required by the heating device. C3 (see equation (3.4)) indicates whether the heating device should work or not, depending on the states of window and outside of temperature.

$$s2 = \begin{cases} 0\text{,if window state is open.} \\ 1\text{,if window state is closed.} \end{cases} \tag{3.1}$$

$$f1 = \begin{cases} 0\text{,if outside temperature} \approx \text{target value.} \\ 1\text{,otherwise.} \end{cases} \tag{3.2}$$

$$C1 = s2 \cap f1 \tag{3.3}$$

$$C3 = C1 \times C2 \tag{3.4}$$

Control systems are designed to fit control requirements. A control requirement is a target of controller or the whole control system, describing a property associated to a controlled object, formalized as a feature of a controlled variable. Focusing on the controller behavior, a sensor sends measurand to the controller which produces a control signal to operate the actuator. Measuring result (current value) attained by sensor ob-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

servation and changed by actuator action is another characteristic of controlled variable. Meanwhile, control requirement sends desired value of controlled variable to controller as well. Semantic of sensor input and output can help controller to connect exact sensors and actuators. Structure of controller is expressed by a function, named as controller function. Inspired from Matlab Simulink commonly used blocks represented by a function, some typical and generic control services can be developed directly. When control parameters are hard to tune, these smart control algorithms can reduce the task of tuning parameters. When the control logic is simple, some control service can be shared by different control applications, requiring all the same semantic controller inputs.

3.3.2 Controller Model

The most important part of a control system is its controller. In the smart factory use case, a global controller consists in three sub controllers, C1, C2 and C3. These three sub controllers coordinate so that the final precise switch control on the operation of actuator A1. As a consequence, the research on control system is aimed at the design a controller to meet the various requirements on stability, rapidity and accuracy. A abstract controller may have multiple inputs and multiple outputs (MIMO, see Fig. 3.6). Herein, we build an abstract controller model, including control requirements (control objectives) and three key devices: sensor, controller and actuator.

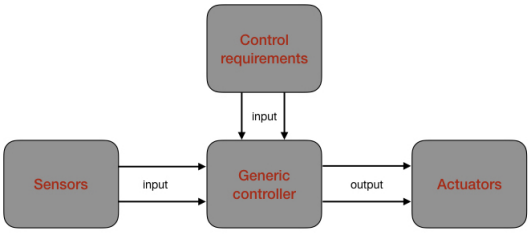

Figure 3.6 – Controller model

The structure of controller can be represented by a following mathematical function:

$$Y = F(U), C$$

U, Y, C and F respectively represent controller inputs, controller outputs, control system constraints, and controller function. It is obvious that this kind of description can hardly be reused as a whole due to its complexity. From the view of each output, a controller can be described by another function defined previously and perceived as a sub controller with inputs from sensors and/or control requirements. So the generic controller is composed of one or more sub-controllers. Output of each sub-controller

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

is connected to one or more actuators. According to the information of the actuator, we can identify what is the controlled variable and which sensor is selected to measure this variable. Sometimes, besides controlled variables, there are other input variables detected by sensors as well called environment variables required for making decisions. If two controlled variables are dependent in a complex system, one is controlled variable while the other is used as assisted variable in a sub control system. When the controlled variable is not observable, assisted controlled variables are needed to infer the value of it.

The controller can be specified by a control pattern. A control pattern definition is illustrated in Fig. 3.7, leveraging the method of Polish Notations. According to this recursive definition, a controlling service can produce a result, implement an operator and use one or more operands. An operand can be associated to a controlled variable (related to a sensor and actuator), a parameter (related to the controller configuration) or to another function expression result. The difference between variable and parameter is that a parameter is fixed value for a given period of time. Moreover, the generation of a new value associated to a variable (note that this value can be equal to the previous one) will trigger an event. An operator is associated to a computing function and defines the way how operands are composed. In addition, a parameter can be tuned to affect the performance of the control system and its unit is often a prototype (usually the value is 1).

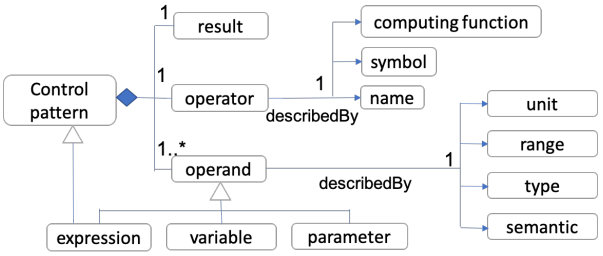

Figure 3.7 – Definition of control pattern

For short, a generic controller composed of sub ones can be applied in any control project. In a project, there may be one or more controlled objects, of which each can have one or more controlled variables. For each controlled variable, there is a sensor and an actuator to be selected to measure and act on it respectively. For each actuator, there is an output of generic controller. Whereas, an output of generic controller can be used for one or more actuators. As a consequence, different actuators may have a same controller (sub controller of generic one). So in a project, provided is each sub controller where semantics of operands, operators and function result should be attached.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

3.3.3 Service Based Control Model

The idea of service brings into system many benefits, e.g., easing the configuration and maintenance, improving the flexibility and agility, and lowering the cost. But the most important reason is that it can match the distributed environment. This results in integration of control and service. Thanks to the CPS or virtualization technology , more and more sensing, actuating and controlling services are available. This leads to the service based control model being developed [54] (see Fig. 3.8).

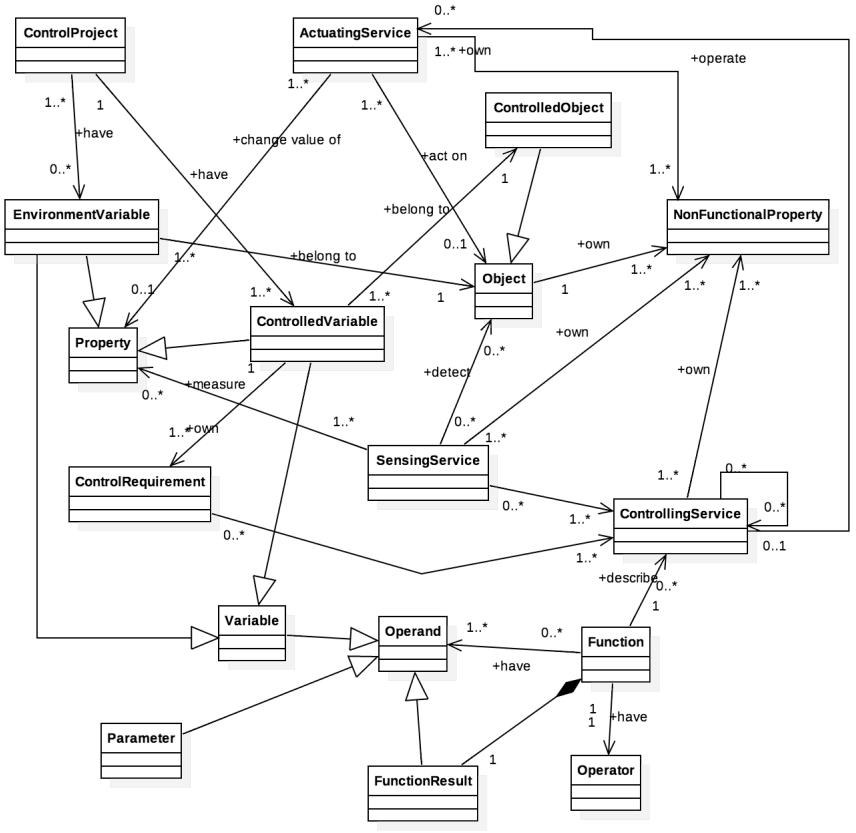

Figure 3.8 – Service based control diagram

As said previously, a control system includes different elements: sensor, controller, actuator and controlled object. A controlled object reflects what is controlled. The controlled variable is a property of the control object. The controlled variable owns information of control requirements. A controlled object is described using several properties, such as temperature, humidity, intensity, location (longitude, latitude and altitude), speed, and so on. Some of properties can be selected as controlled variables. Hence a controlled object possesses of at least one controlled variable. A controlled variable can belong to different controlled objects. Some of attributes of controlled variable class

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

are intended to specify the control objective. Desired value is an expected value of controlled variable. As the real steady state value may do not equal to the ideal one, steady state error reflects the deviation of them. Rise time is used to show the time in transient process. Peak and valley values set a condition that the changing value of controlled variable can not be greater than the peak value and less than the valley value. Based on type of controlled variable and the specific controlled object, sensor and actuator type will be determined. The type of sensing service can be temperature, speed, position, and so forth. Nevertheless, there are actuator types of motor, air conditioner, dryer, and so on. In a control system, its controller is the most significant role that is responsible for meeting the user control requirements. The structure of controller is specified by a function of which control pattern can be selected referring to knowledge from the books or practice. Function is expressed by function tree composed of function node. The function element of each node is either an operator or operand. A control service is a composite object composed of at least one sensing service, one controlling service, and one actuating service. Features of them are distinct, but they are connected and can not exist individually in a control service. As a consequence, sensing services, controlling services and actuating services will be selected and composed from service registry. Controlling service is characterized of a function explaining how to calculate the data from sensing service, web service and/or users. Sensing service send measuring results to one or more controlling service. A controlling service can send a control signal to an actuator or other controlling services. A service in IoT context owns couple of non function properties (NFPs, e.g., location, connection, CPU, precision, etc.). One or more NFPs can be managed by one NFP policy.

3.4 Integration of Non Functional Requirements

In the previous section, attention is paid to the description of cloud control system from the view of functionality. Generally speaking, control requirements and functional descriptions of sensing, controlling and actuating services as well as their relationships are discussed. However, these services are IoT services which are associated to the physical devices closely. In our different use-cases, location of sensors and actuators may be considered while defining the input / output of controlling services: this location is important to get the temperature of the room (in the smart building use case) or of the manufacturing environment (in the FabLab use-case), to identify the traffic light actuating stem (in our smart transportation use case). Other non functional requirements such as sensing precision (in the Fablab use case) may also be considered as well when

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

selecting the proper controller / actuator. These non functional requirements and context related information may also affect the controlling service itself (for example, in the FabLab use case, the temperature and humidity may affect the injection control element).

Consequently, NFPs related to the physical constraints should be emphasized and thus studied in a further step. In the state of the art chapter, some key NFPs defined in the existing ontologies are picked up. Nevertheless, these original NFPs have not been processed as some of them (e.g., accuracy and precision) are quite identical

Table 3.1 – Device related Non Functional Properties (NFPs)

NFP name	Ontology	Sensor/actuator/controller	NFP group name
Frequency	SSN	sensor + actuator + controller	System capability
Range	SSN	sensor + actuator + controller	System capability
Precision	SSN	sensor + actuator + controller	System capability
SystemLifetime	SSN	sensor + actuator + controller	Embedded device performance
Memory	FIPA	controller (sensor + actuator)	Embedded device performance
CPU	FIPA	controller (sensor + actuator)	Embedded device performance
Location	CISRO	sensor + actuator (controller)	Physical environment
ConnectionMedia	FIPA	sensor + actuator + controller	Communication
TransmissionLatency	FIPA	sensor + actuator + controller	Communication

Nine NFPs are selected and organized into four main groups (see table 3.1): system capability describing the device capability (frequency, range and precision) associated to its interaction with its environment, embedded device owning constraints (CPU, memory and LifeTime), environmental constraints (location and operation conditions), communication system performance (connection media and communication latency). From the table, we notice that in general these NFPs can be applied all devices, namely, sensor, actuator and controller. Nevertheless, when a NFP does not plays a key role in a type of devices, the device type for this NFP is represented in a pair of parenthesis. For instance, Memory is not so critical when describing a sensor and actuator.

These NFPs can be used in two places. First, when selecting sensing, controlling and actuating services, they are considered as selection criteria. In our smart building use case, the sensor and actuator locations are necessary to select the convenient device to control the temperature in a dedicated room. Second, control application context is defined thanks to them. Based on this NFP classification, we define four policies, each of them associated to a dedicated group, to describe the control system execution context so that the context-aware composition process can be managed.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

3.5 Event Driven Communication

Section 3.2 and 3.3 have focused on the static descriptions of control service. In traditional service based organisation, the message routing feature proposed by ESB increases the loosely coupled abilities. Paying attention to the distributed control field, we need to adapt it in order to maintain this “loosely coupled” feature, allowing a same control service element to be reused in multiple systems and providing on-demand service interconnection. Moreover, this control service model needs “reaction requirements”, leading to complex distributed systems. For example, in our smart transportation use case, the emergency car trip is used to define the way controllers should exchange messages and “react" according to the traffic information messages. Similar requirements have been set for decades for complex telecommunication systems. This has contributed to event-based protocols definition. Hence we apply the same event-based organisation to support our distributed control service model [55]. The problem of orchestrating these involved services is solved via this event driven way. Event driven vision can also provide control system with features of adaptivity and reactiveness (autonomy). These two characteristics are essential to a cloud control system. Based on this event driven communication mechanism instead of a fixed invocation or remote procedure calls (RPC), a loosely coupled control system can be implemented as there are no more fixed service invocation but only event-based coordination between services.

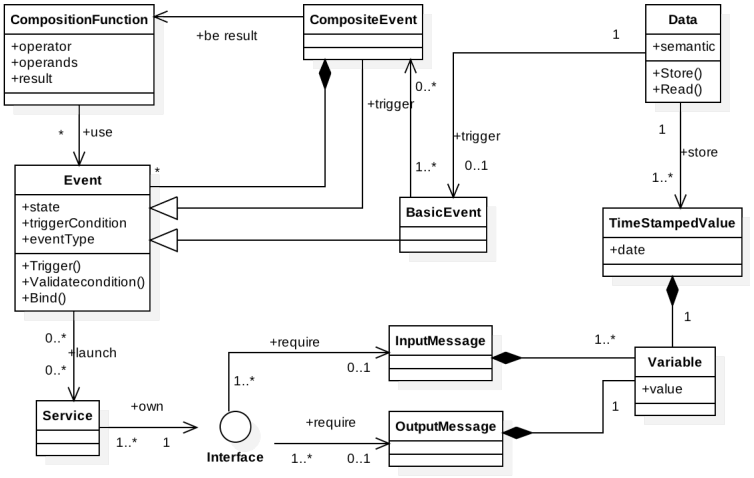

Figure 3.9 – Event class diagram

An event is either a basic event, or a composite one. A composite event is constructed of at least two events (basic event or another composite event). Generally, these events are based on data, the outputs of services involved in the control application. Sensing service outputs are input variables of controlling services. Controlling service outputs

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

are input variables of controlling services themselves or actuating services.The class diagram specifying the relationship of data and event is illustrated (see Fig. 3.9). An event can bind services from the distinct nodes. Functionalities of services saved in a node are so similar that only one of them will invoked at running time. This sorting task is completed in a dynamic way. When an event is triggered, bound services from separated node service groups with better performance than the other services in the same group will be invoked immediately. Activating one sub basic event or all sub basic events is determined by composition function operator related to this composite event. A service owns an interface composed of two messages (input and output messages). The output message is populated of one value while input message is constituted of one or more values. An output value together with time when it is received will construct a time-stamped data. According to the data flow described in the block/node diagram, the matching data object will store it. As a result, “Trigger" function which aims at triggering related basic events will be invoked after “Store" method is conducted. In the smart transportation use case, when the emergency car position sensing services produce a new value, a related event will be triggered, leading to invocation of the route planning controlling service.

3.6 Control Ontology

As stated in the state of the art, none of the IoT ontologies integrate the different control elements, nor IoT device interface specification. To overcome this limit, we have designed a control ontology (see Fig. 3.10), gathering concepts issued from both IoT ontologies (for sensor, actuator and some of the non functional properties description), IoT reference model (to capture the IoT interface description), service description from the OASIS reference architecture to capture the service non functional properties and automation knowledge [56]. By this way, our ontology integrates a logical specification of the control system and a description of the physical resources involved while implementing it. To this end, a control system is abstracted as a control service. This service is described functionally, thanks to a control function, similar to traditional automation transfer function specification. The controlling function is described recursively as an operator transforming several operands in an output vector gathering the output variables. An operand can be defined either as a basic input variable associated to a sensing system or as the result of another control function (called later data variable). The controlling function can also include computation parameters that are tuned to fit the current control context and is associated to a computing service. Both sensors, ac-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

tuators and data variables are associated to units and range to precise their value. The control function is implemented thanks to a controller which can be implemented either due to a physical device or a dedicated control service called later “logical device". To address the physical deployment of this “logical device", we introduced three key device models, namely sensor, controller and actuator, each of them providing sensing service, controlling service and actuating service respectively. Device constraints limiting the functionality service or device can offer are physical environment (such as location) and physical capability (e.g., memory, CPU, connectivity, energy, resolution) are taken from classical IoT ontologies. Interactions between physical and logical control components are achieved by events associated to inputs used by controllers or actuators. As a consequence, these events are also associated to data produced as outputs by sensors or controllers. By this way, control system functional description (what the system does) and non functional characteristics (related to the way the job will be processed) can be described. Intended for a comprehensive understanding of cloud control system, control ontology is achieved.

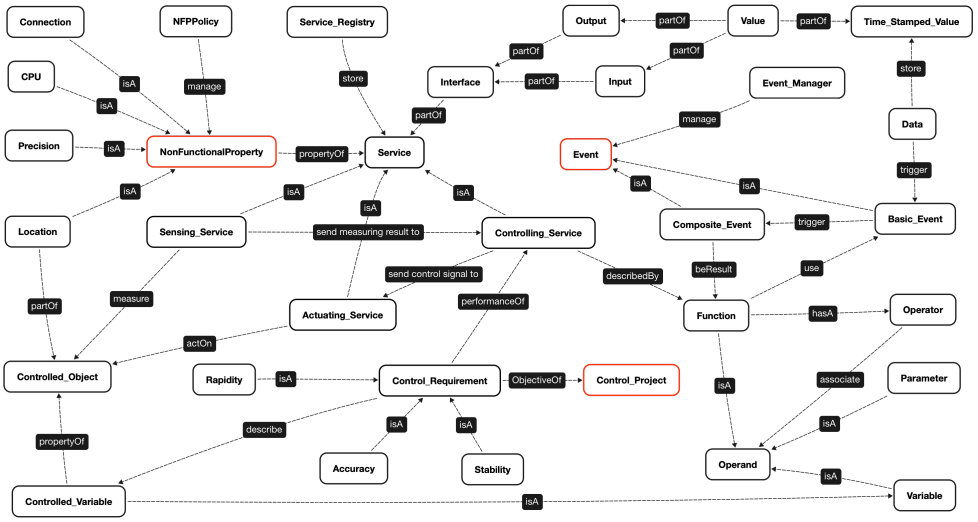

Figure 3.10 – Control ontology

In brief, our control ontology is composed of three parts: control service functional model dedicated to describe control service elements (sensing service, controlling service, actuating service and controlled object) integrating their functional characteristics, non functional property part endeavoring to specify the device capabilities and their surrounding environment, and event part intended to delineate the communication mechanism.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

3.7 Conclusion

In this chapter, we have defined how to model a control service through building a control ontology composed of three modules: control service functional model, non functional requirements, and event driven communication mechanism. This ontology describes the cloud control system fully from aspects of control service elements (sensing, controlling and actuating services), their relationships, service selection, composition and orchestration. Service based control model is evolved from control system model by integrating the service concept and controller model from the automation view. NFPs play a important role in selecting sensing, controlling and actuating services and describing the cloud control system execution context. Event driven communication as an efficient way control service elements are orchestrated, exchanging events between them, is chosen.

This ontology shows us what is a control service and provides a comprehensive vocabulary for cloud control system. These basic involved concepts promote the building of Service Oriented Control Architecture which is targeted to facilitate development of cloud control application and to support our Control Service registries specification in chapter 4.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Chapter 4

Service Oriented Control Architecture

Contents

4.1 Introduction

In chapter 3, we focused on the first research question, i.e. defining and modeling a control service and a cloud control system. Based on the Control as a Service model we presented in chapter 3, we have proposed a control ontology, integrating both a static vision based on functional and non functional properties used to describe requirements and control services' behaviors, and a dynamic vision of the control system organisation based on events, interconnecting control services. This first contribution leads to a

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

second research question, i.e. designing a service oriented control architecture to implement a cloud control system, integrating both the logical control vision and the physical devices in a loosely coupled way.

A Cloud Control System (CCS) is a distributed control organisation that adapts the traditional XaaS Cloud model to integrate Control services. As stated in chapter 3, a control service is composed of sensing, controlling and actuating services, gathering both software and hardware resources. To support this Cloud Control System vision, control service elements (sensing services, actuating services and controlling services) are pre-selected at design time, based on functional and non functional requirements so that a pre-composition graph associated to the control process "workflow" can be set. Then, contextual information is used at runtime to select the "best" control service to be orchestrated dynamically. Thanks to the event-based organisation, control services are interconnected in a loosely coupled way.

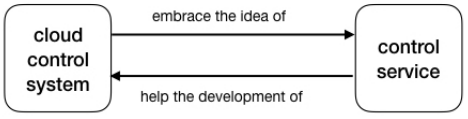

Figure 4.1 – Relationship between control service and cloud control system

To support this Cloud Control System implementation, a service oriented control architecture for control service is built, gathering both SOA and NIST IoT Reference Architecture characteristics. On one hand, SOA describes services in an abstract way, allowing to define logically a control process similarly to business processes, taking advantage of services reusability to set loosely coupled systems. On the other hand, NIST IoT reference architecture pays attention to physical characteristics of IoT components, i.e. sensors and actuators and integrates controlling and sensing modules dedicated to control description according to its usability. Nevertheless, this reference architecture lacks of detailing control elements and relationships between them. This may lead to organize "hard" wired connections between devices, inspired from traditional control system, or "hard coded" remote procedure calls to implement networked control systems.

To fill the gap between the NIST "physical" IoT architecture and the SOA logical world, a Service Oriented Control Architecture will take advantage of the loosely coupled feature provided by services and of the precise physical device model provided by the NIST RA. Our multi-layer architecture consists in a three main layers:

- The logical layer enriches the NIST IoT reference architecture by organizing devices logically and providing service artifacts associated to the different components.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

- The Micro service layer implements an interface tier to connect physical devices to logical services.

- The Physical layer is based on the NIST reference architecture to describe precisely physical devices. It enlarges the scale of SOA to physical IoT device-based implementation

To allow context awareness and management, fine-grained device management as well as contextual adaptation of Cloud Control Systems (CCSs for short) implementation, a NFP monitoring and governance module is added so that the physical execution context can be taken into account while orchestrating control services.

Based on this architecture, service selection, composition and orchestration are adapted to integrate Non Functional Properties, due to the physical control device management. As discussed in chapter 3, NFPs are introduced to support more precisely control requirements related to the process "hard" characteristics (location, temperature...) that must be considered while selecting the candidate services to set the pre-composition graph at design time. At runtime, NFPs are also taken into account to select the "best candidate" to be orchestrated in a late binding strategy. This requires managing loosely coupled interactions between control services.

Thanks to our micro-service interface tier, a device artifact is built. This artifact either produces time-stamped data (for sensing and controlling devices) or consumes these time stamped data (for controlling and actuating devices). This allows turning the device hard connection problem to a "soft" and loosely-coupled event-based interconnection of control services instead of exact service calls. We organize a centralized event management for each Cloud Control System, gathering time-stamped data from the distributed devices, as the micro-service tier provide an interface to the devices.

To support this event-based organisation we adapt the traditional Enterprise Service Bus to this event-driven control service organisation, including a event manager, used to generate events from incoming data, a context manager related to our NFP monitoring and governance module and an event manager in charge of the late binding process, propagation the different events to the "best" service.

Implementing this Service Oriented Control Architecture, our prototype developed using Java, MySQL[1], MongoDB[2], Jena[3] and Vertx[4] has been used on the different use cases presented chapter 3. These experiments will be presented throughout this chapter.

[1]https://www.mysql.com/
[2]https://www.mongodb.com/
[3]https://jena.apache.org/
[4]https://vertx.io/

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Our prototype runs on a MacBook Pro with 2.2 GHz processor, 16 GB Memory, java version 1.8.0_171, jena API version 2.6.4, vertx version 3.8.4, mongoDB server version 3.6.5 as well as IntelliJ IDEA 2018.2.5 (Ultimate Edition). Jena is a free and open source Java framework for building Semantic Web and Linked Data applications. Vertx framework is selected because it is JVM-based, light-height, reactive and high-performance, and because it supports reactive programming provides an asynchronous and non-block communication way. In addition, it owns a basic event bus mechanism and supports back pressure which is able to manage traffic flow and event congestion. MongoDB as a NoSQL database that is free and open source cross platform documented oriented is chosen and its JDBC interface helps to save the data. This chapter is organized in three parts:

- first we present our multilayer Service Oriented Control Architecture (SOCA for short) consisting in a logical layer, a micro-service layer and a physical layer,
- second we define the way this SOCA is deployed at design time to achieve the pre-selection and pre-composition of the sensing, controlling and actuating services
- third, we present the way SOCA is deployed at runtime, using the data manager to generate events, manage the context and finally orchestrate the services thanks to a late binding process.

As a result of this architecture and the event-driven orchestration strategy, our SOCA allows designing and orchestrating Cloud Control Systems taking advantage of the loosely coupled feature.

4.2 SOCA Multi-layer Architecture

We use our ontology to identify the key concepts while designing our Service Oriented Control Architecture (SOCA for short). To this end, we embed control components (controllers, sensors and actuators) into micro-services. Micro-service based artifacts standardizes description of the service interface, i.e. the input data it requires and the output data it produces. This SOCA (see Fig. 4.2) integrates three layers [55]:

- The **physical layer** is used to store the control system description based on the classical automation vision. Besides physical object (controlled object), there are generic devices associated to infrastructure (gateway) and to traditional control devices: sensor, controller, and actuator. A physical variable is a property reflecting the status of the related physical object. From the control point of view, sensor,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

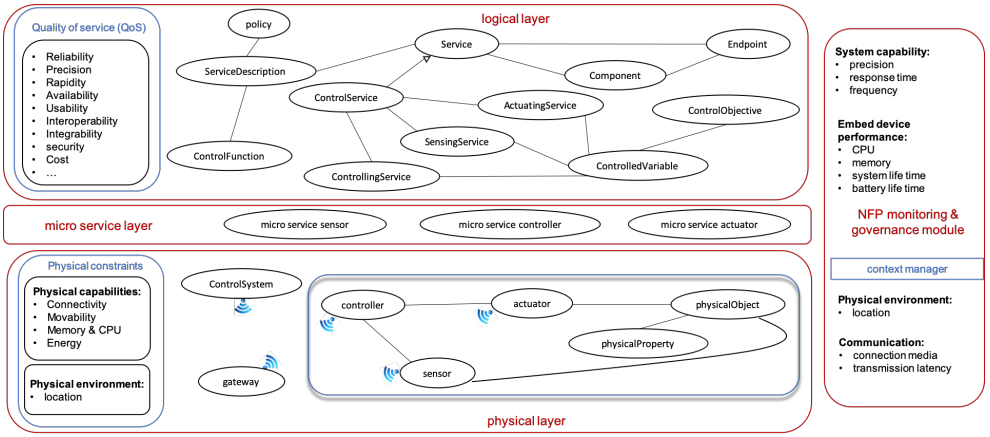

Figure 4.2 – Control as a Service architecture

controller, actuator, physical object are connected using gateways (especially in the wireless environment) to implement the traditional control loops. Physical constraints are related to both environmental initial conditions and limitations of devices' capabilities.

- The **logical layer** is populated of services associated to the physical elements such as control service, sensing service and actuating service. More precisely, a controlling service is defined by the mathematical function (transfer function or control function) specification, its inputs and outputs allowing to connect it to other sensing, controlling and actuating services. The control objective (requirements for control system and controlled variable) is also connected to the control service. Service descriptions include both functional information (describing what the service implements) and endpoints as well as policies including non functional properties description. Three IoT services are connected via one controlled variable belonging to a controlled object. Here, IoT service represents the virtual twin of a physical CPS device. For instance, one sensing service provides the functionality of a specific sensor. So do controlling service and actuating service.

- The **micro-service** layer is used to support the loosely coupled control feature. It carries out a single point of interaction from the device view. Each micro-service is associated to a logical service and is generated depending on a project control flow. Basically, a micro-service behavior consists in 3 steps: waiting for the initial condition (its input event) associated to the data it has to process, then selecting the best device depending on the context (late binding feature), to implement the logical service process and finally checking the device interface to invoke it properly.

Control context consists in both physical and logical contexts. To support Context-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

aware application development, different Control Objectives may be defined to fit different contexts. At runtime, a context manager is in charge of capturing context information, consolidating this information to decide if the current control objective must be tuned or changed. The execution context defined in our architecture is populated with all context information related to both physical and logical layer entities. Environment dependent context is captured in a traditional control engineering approach, identifying several transfer functions associated to different contexts. In this paper we focus on NFP-based context description. To support an efficient context management, we propose to organize these non functional properties (extending the different works on control ontologies presented in the state of the art section) into five types, namely, physical environment (e.g., location), embedded device performance (e.g., CPU, memory, battery life time), communication (e.g., connection media, transmission latency), data (e.g., realtime controlled variable), system capability (e.g., precision, response time, frequency). These NFPs are related to both physical constraints and QoS and measured by NFP monitoring and governance block. A context manager is also included in this module. When control context is changed, sometimes node instance will be substituted; sometimes composition graph need to be recomputed.

Persuasively, control does exist in almost every domain and IoT application can be applied in many fields, such as household industry, manufacturing, transportation (city), agriculture, medical equipment, and so forth. To apply our service oriented control architecture in different control applications, three motivating examples are reused and analyzed in different stages. In the following sections, we will introduce the classical temperature control as an individual control application in smart home at the design time. Next, context-awareness control application in smart manufacturing and event driven interactions in cloud control system in smart transportation are discussed at the running time.

4.3 SOCA at Design Time

Engineering a control architecture is usually achieved through the following steps: identifying the requirements, defining the transfer function and organizing the control system and the embedded equipment. Compared to this traditional approach, control service will take advantage of the selection/composition and orchestration of standardized components. Coupled to the Cloud of Thing abilities, this will increase agility, flexibility, satisfaction. Meanwhile, applying automation method into service leads to improvement of service composition as composition rule is represented in the form of

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

control function. Therefore, control application development process involves in generating the pre-composition graph and implementing the control service orchestration.

In this section, attention is paid to the pre-composition of the control elements at the static time. This requires the pre-selection of them and it should be accomplished before or at the same time. Pre-selection task depends the service registry where services are stored, control requirements that illustrates the control objectives and block diagram. In the control objectives, the global control requirements such as response time, static error, are recorded. In the block diagram, a primitive pre-composition graph is made and each block in it has its own requirements. To sum up, this section presents service registry, control requirements and finally the operations of pre-selection and pre-composition.

4.3.1 Service Registry

This subsection is focused on the physical and micro service layers of our service oriented control architecture. These two layers are digital twins. IoT micro services are encapsulated from the IoT devices. Service registry stores in a centralized way the microservices encapsulating the IoT devices. Service registry connects service provider and service consumer. Service provider publishes services into service registry and service consumer subscribes for services from it. This service registry is picked from traditional SOA but has been enriched by integrating control characteristics. It means that service registry needs to encompass both functional and some of the NFPs. The mechanism of service registry is implemented thanks to the involved tables, such as “Services", “SeningServices", “ControllingServices", “ActuatingServices", etc. So we need to discuss the design of related tables in data base.

Besides the four tables mentioned above (“Services", “SeningServices", “ControllingServices", “ActuatingServices"), tables of “ControlPatterns", “NFPs", “Properties", “R_Services_NFPs" are defined to provide a complete service registry for users to select services for their control projects. The overview of these eight tables is illustrated in Fig. 4.3.

“Services" (see table 4.1) provides a global identity management for all services. Service type can be either a sensor, an actuator or a controller. Service endpoint shows how a service can be accessed. In the table of “R_Services_NFPs" where the identities of service and NFP are a pair of composite identities. NFPs for each service are described.

Precisely, there are three types of tables, i.e., “SensingServices" (see table 4.2), “ActuatingServices" (see table 4.3) and “ControllingSerivces" (see table 4.5). The identity of service (“ServiceID") is the main key in the “Services" and the foreign key in its three

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

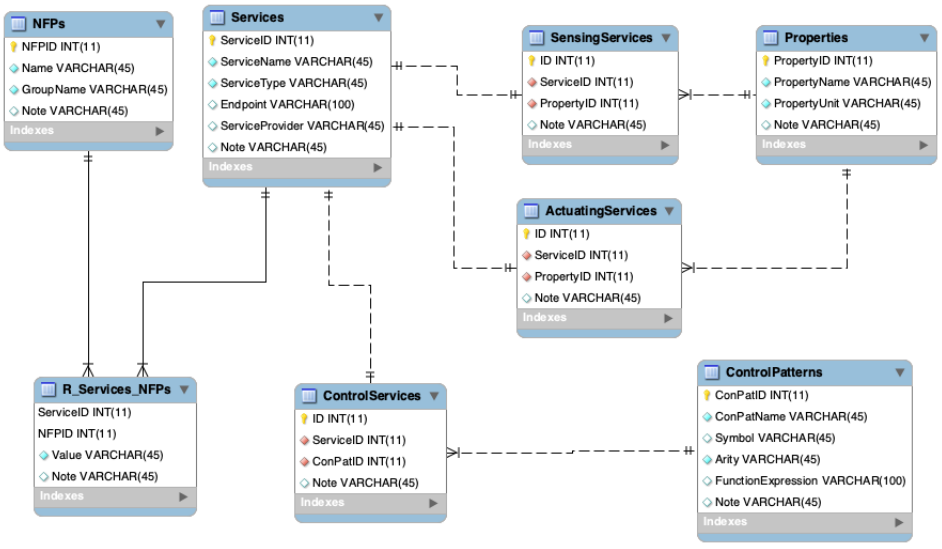

Figure 4.3 – Basic tables ER diagram

subclasses. The identity of property named after"PropertyID" is defined in the table of "Properties", a property, e.g., temperature, humidity, must have a unit. The unit of a property can be a prototype "1". In the "SensingServices" and "ActuatingServices" tables, "PropertyID" is deemed as a foreign key that shows the property measured and affected by the sensor and actuator respectively.

Table 4.1 – Services

ServiceID	ServiceName	ServiceType	Endpoint	Service Provider	Note
1	temperature sensor	sensor	Root/SensingService1/		
2	temperature sensor	sensor	Root/SensingService2/		
3	add controller	controller	Root/ControllingService1/		
4	multiply controller	controller	Root/ControllingService2/		
5	humidity actuator	actuator	Root/ActuatingService1/		
6	temperature actuator	actuator	Root/ActuatingService3/		
7	humidity sensor	sensor	Root/SensingService3/		
8	minus controller	controller	Root/ControllingService3/		

Root: http://Localhost:8081/Services

Table 4.2 – Sensing Services

ID	ServiceID	PropertyID	Note
1	1	1	temperature
2	2	1	temperature
3	7	4	humidity

A controlling service can be a single one or the combinations of its sub controlling services. Its most important feature is control pattern depending on the operators that are able to connect different operands or other operators' outputs (see Fig. 3.7 defined

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Table 4.3 – Actuating Services

ID	ServiceID	PropertyID	Note
1	5	4	humidity
2	6	1	temperature

Table 4.4 – Properties

PropertyID	PropertyName	PropertyUnit	Note
1	temperature	Celsius	
2	displacement	meter	
3	speed	m/s	
4	humidity	1	

in the chapter 3). In the “ControlPatterns" table (see table 4.6), a control pattern must own its identity, name and possibly may have a symbol, a fixed number of arities and a representable function expression. Some basic control patterns (for example, add, subtract, multiply, divide, and, or, and so on.) and complex ones (such as sigma, pid.) are defined in it. A service associated to a complex control pattern can be perceived as a one controlling service and invoked directly. This will simply the pre-selection task for given complex controllers.

Table 4.5 – Controlling Services

ID	ServiceID	ConPatID	Note
1	3	1	add
2	4	4	multiply
3	8	3	minus

Table 4.6 – Control Patterns

ConPatID	ConPatName	Symbol	Arity	Functional Expression	Note
1	add	+	2	a1+a2	plus
3	subtract	-	2	a1-a2	minus
4	multiply	*	2	a1*a2	
6	divide	/	2	a1/a2	
7	and	&&	2	a1&& a2	
8	or	||	2	a1||a2	
15	greater	>	2	a1>a2	
16	greater or equal	>=	2	q1>=a2	
19	sigma	$\sum$	n	a1+a2+...+a(n)	
20	pid	pid	4	PID equation	operand *

Conpat : control pattern
PID equation : $kp * e(k) + ki * \sum_{i=0}^{k} e(i) + kd * []e(k) - e(k-1)]$
operand * : operand1=e(k), operand2=kp, operand3=ki, operand4=kd, m=1,2...n.

In the physical tier of our multi-layer architecture, physical capabilities and environment are becoming more and more important to provide adaptive feature for IoT systems. Based on the Non functional property table (see table 3.1) mentioned in the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

chapter 3, it is easy to design and fill in the “NFPs" (see table 4.7). When a service is supplied, its NFP descriptions will be included and accessible. Each NFP for each service, if exist, is specified in the “R_Services_NFPs" (see table 4.8).

Table 4.7 – NFPs (Non Functional Properties)

NFPID	Name	GroupName	Note
1	Frequency	system capability	
2	Range	system capability	
3	Precision	system capability	
4	SystemLifeTime	embedded device performance	
5	Memory	embedded device performance	
6	CPU	embedded device performance	
7	Location	physical environment	
8	Temperature	physical environment	
9	ConnectionMedia	communication	
10	TransmissionLatency	communication	

Table 4.8 – R_Services_NFPs

ServiceID	NFPID	NFPValue	Note
1	1	1	
1	2	-20, 100	
1	3	0.1	
1	4	2 year	
1	5	500	
1	6	2 GHz	
1	7	30 degree, 40 degree, 5 meter	
1	8	WIFI	
1	9	0.5 ms	
2	1	1	
2	2	-20, 100	
2	3	0.1	
...	...	...	

Our prototype uses MySQL to support the Database storing the registries. In order to select the data in the data base, data base helper classes are required. Herein we design a “DBHelper" class (see Fig. 4.4). “GetServices" is designed to select all the services. “GetServiceIDFromEndpoint" is used to query the service based on the end point address. “GetControlPatternIDByConPatName" and “GetControlPatternIDBySymbol" are used to select the identity of control pattern according to to the control pattern name and symbol respectively, assisting the pre-selection of controlling services. “GetPropertyID" can help to match the feature of interest of sensing or actuating service. “GetNFPIDFromName" is aimed at managing the NFPs via transforming the NFP name to NFP identity. An example of implementing “GetNFPIDFromName" method is proposed in Fig. 4.5.

Service registry provides an interface for users to select services they need. It makes a prerequisite for pre-selecting sensing, actuating and controlling services. Due to it, the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

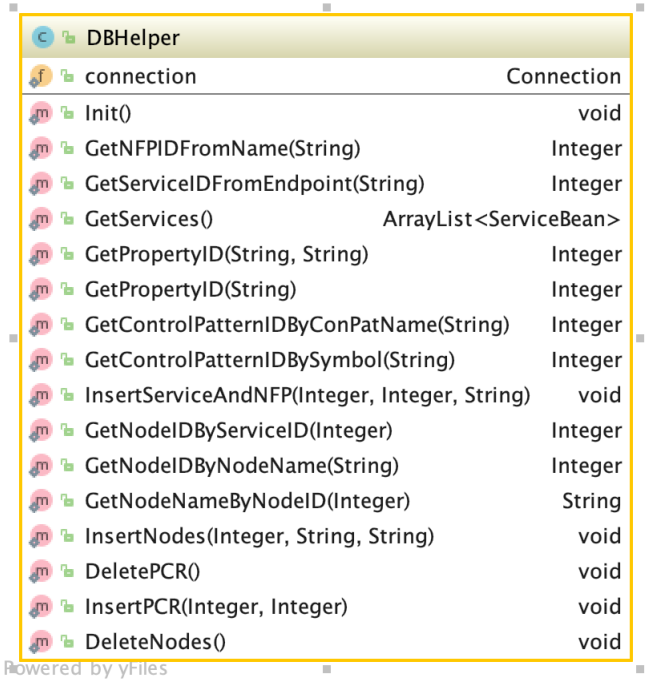

Figure 4.4 – DBHelper class

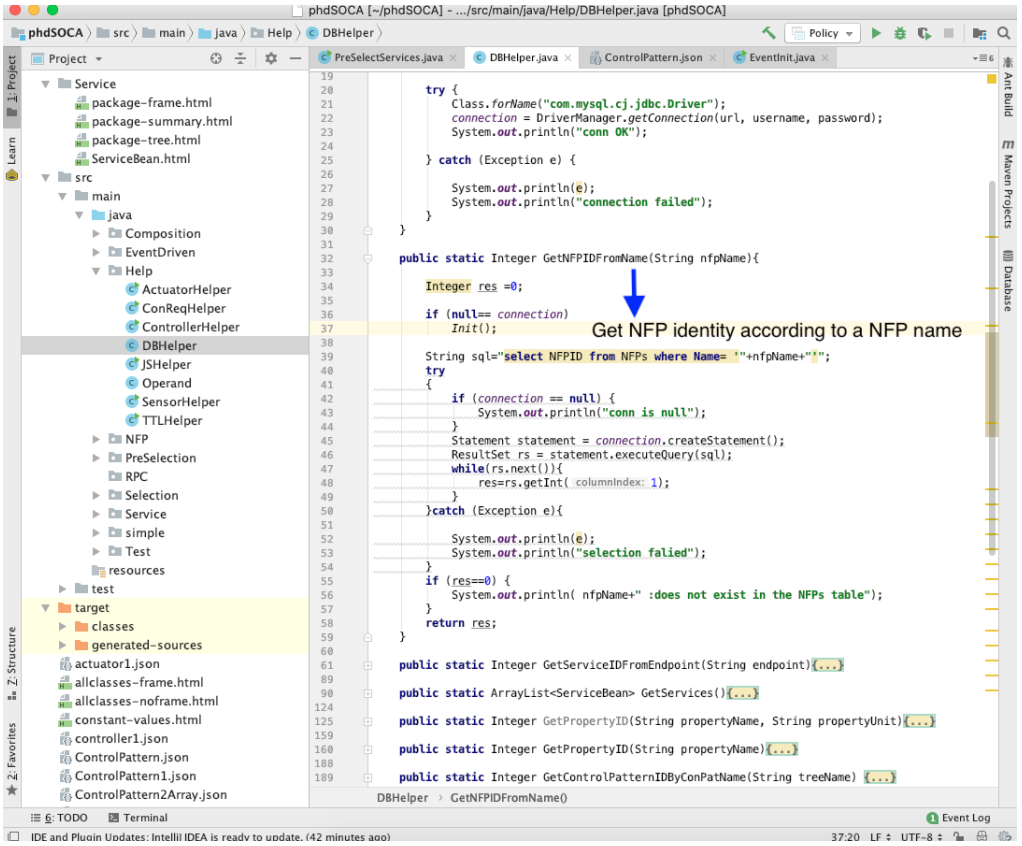

Figure 4.5 – Screen shot of “GetNFPIDFromName" method

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

pre-selection task of these involved sensing, controlling and actuating services can be accomplished.

4.3.2 From Control Block Diagram Description to Service Selection Criteria

Service registry shows us where we can find the services. We also need to define the selection criteria first. These selection criteria are extracted from the control requirements. As a result, this subsection is focused on service requirements, associated to the logical layer of our SOCA. A block diagram describes the control process initially. For each block in the block diagram, functional and non functional requirements are set before carrying out the selection task. In the following paragraphs, we sill use our smart home use case (see Fig. 3.1 in the chapter 3) to present the control requirements clearly for temperature control application. As the global controller composed of sub controllers C1, C2 and C3 is complex, only C2 is focused and considered as the global controller which applies the proportional control pattern.

In short, requirements for developing a cloud control system consist in a control requirement elaborating the user intentions, and a block diagram describing the primitive structure of the control application.

4.3.2.1 Control Requirement

Control requirement should explain clearly the following two questions:

1) What is controlled?
2) What is the control objective?

The answer for first question is produced by the controlled object and its properties. The descriptions of a controlled object are populated its name, location, temperature, humidity, etc. Among them, some named of controlled variables (sometimes environment variables are necessary) are so predominant that many control objectives are concentrated about it. The specifications of a controlled variable, including its initial value, steady-state value, precision, frequency, rising time, peak values including the maximum and minimum ones, stability extent, are the answer of second question. The relationship between controlled variable and control objective is that one control objective may involves more than one controlled variables while controlled variables may have more than one control objectives.

From our smart home use case, we focus on the temperature control requirement. Controlled object is room1 and controlled variable is temperature whose unit is Celsius.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

The current temperature is 5 Celsius and steady state value is 25 Celsius. The precision is 1 Celsius and the steady state error is less than 2 Celsius. The rise time is no more than 5 minutes. The maximum and minimum temperature in the room is 30 and 5 Celsius respectively. The temperature control system must be stable. In our prototype we use JSON format to support these different objects description. Fig 4.6 presents the JSON description of this temperature requirement.

```
{ "controlRequirement":
      { "controlledObject":
            {"name": "room1",
             "location":
               {"longitude":"", "latitude": "", "height": ""}
            },
        "controlledVariable":
           {"type": "temperature",  "unit": "Celsius"},
         "controlObjective":
            {"domain": "time domain", "timeUnit":"minute",
             "stability": "stable", "Valuetype":"Integer",
             "initialValue": 5,  "precision":1,
             "accuracy":
                  {"desiredValue": 25,
                  "stableVauleRange":{"min":23,"max": 27}
                  },
               "rapidity": {"riseTime":5},
               "peakValue": {"max":30, "min":0},
               "frequency":1
             }
      }
}
```

Figure 4.6 – Control requirement

4.3.2.2 Block diagram

A block diagram works as a business process diagram indicating how data flows in the control application. A typical block diagram of a single variable (controlled variable) closed loop control system is presented in Fig. 4.7. Designing a control system consists in combining sensors, controllers, actuators and a control plant (controlled object) according to the user's requirements. This primitive control block diagram is built based on the physical characteristics of the control system.

Fig. 4.8 presents the simplified smart home use case, only involving two sensing services (S1 and S4), one controlling service (C2) and one actuating service (A1). Definitions of S1, C2 and A1 have been presented more precisely (see Fig. 3.1) in the chapter

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

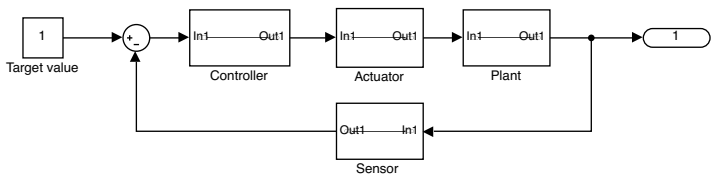

Figure 4.7 – Block diagram of representation of a typical feedback control system

3. S4 shows the desired value of controlled variable from the user. Possibly, it can be a web service, such as a web form. So there are only three blocks, a sensor, a controller and an actuator block (see Fig. 4.9).

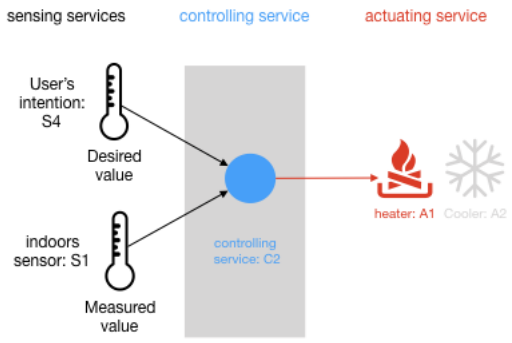

Figure 4.8 – Simplified smart home use case

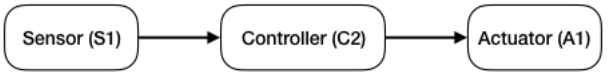

Figure 4.9 – Block diagram for simplified smart home use case

For each block in the block diagram, there are functional and non functional requirements. These requirements defines the selection criteria of candidate services for the block. This leads to the discussion of requirements of sensor, controller and actuator blocks.

Requirements of Sensor Block. Sensor block is the entrance of block diagram. Requirements of sensor block for our experiment are shown in Fig. 4.10. From the functional view, its feature of interest is temperature. This means that the sensing service is able to measure the real time temperature. Non functional properties of IoT services should be taken into account while selecting and composing them. Sensing service S1 should be possessed of at least 500 MHz CPU, 2 MB memory. The Life of it ought to be one year. It can be connected by WIFI and its transmission latency is around 10 ms.

Requirements of Controller Block. The data of sensor block flows into one or more controller blocks.Functional description of controller block in the block diagram can be

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

```
{ "sensor":
  { "CPU":"500",
    "memory":"2",
    "systemLifeTime":"365",
    "connectionMedia":"WIFI",
    "transmissionLatency":"10",

    "featureOfInterest":"temperature",
    "valueSource": "realTime"
  }
}
```

Figure 4.10 – Sensor block requirement from our smart home experiment

leveraged to specify the control component interface where variables and parameters are controller inputs and function result is its output. From the view of control component, it may be slightly complex so that a basic operator (function) can not completely describe it. As a result, one control service interface description may be dependent on more than one function. Each function has a control component (which can result of the composition of several control components). Slightly complex control component serves as embedded control system populated with sub elements. To reuse the control component especially for a commonly used block, the definition of operator is extended. An operator can be not only a basic operator (e.g., arithmetic operators or logical operators), but also typical control pattern (e.g., PID (Proportion, Integration, and Differentiation)). Control pattern of the controlling service can be selected according to the system characteristics.

An example of controller C2 from our smart home use-case is presented in JSON format Fig. 4.11. It shows that C2 applies the proportional control pattern represented in a polish notation way. There are three operands for it. Operand 1 is the output of sensor block (or sensing service S1). Operand 2 is the desired value of controlled temperature. Operand 3 is the transmission gain to amplify the difference between operand 1 and operand 2. Controlling service C2 also has NFPs. To facilitate the selection process, they are set to be with same sensor block. Note that each operator, operand or the expression is put in a pair of brackets.

Requirements of Actuator Block. The output of controller block is an input of another controller block or an actuator block. In the simplified smart home use case block diagram, there is only one controller. So the controller block (controlling service C2) produce the input of actuator block. The requirements of actuating service A1 is specified (see Fig. 4.12). The feature of interest of the actuator block is temperature as

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

```
{
"controller":
  {
    "domain": "time domain",
    "style": "polish notation",
    "CPU":"500",
    "memory":"2",
    "systemLifeTime":"365",
    "connectionMedia":"WIFI",
    "transmissionLatency":"10",
    "controlPatternName":"p",
    "mathExp": "((*)((-)()())())",
    "operandNO":3,
    "operands":[
        {"name":"operand1","type":"variable", "semantic":"
            output of S1 block"},
        {"name":"operand2","type":"parameter", "semantic":"
            desired value"},
        {"name":"operand3","type":"parameter", "semantic":"
            transimission gain"}
    ]
  }
}
```

Figure 4.11 – Controller block requirement from our smart home experiment

well. And non functional requirements of actuator block is the same with sensor and controller blocks.

```
{ "actuator":
  { "CPU":"500",
    "memory":"2",
    "systemLifeTime":"365",
    "connectionMedia":"WIFI",
    "transmissionLatency":"10",
    "featureOfInterest":"temperature"
  }
}
```

Figure 4.12 – Actuator block requirement from our smart home experiment

After introducing the block diagram globally and describing the each sensor, controller or actuator block in it detailedly, we starts to focus on how to represent a block diagram. The implementation of a block diagram relies on a directed graph showing the way control data flows In the Fig. 4.13. A graph stores a list of vertexes and a list of edges. This graph also saves the global control requirement where controlled object and

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

control objectives are provided. A vertex is associated to a block in the block diagram. It is instantiated according to the linked file of requirement, e.g., sensor, controller or actuator block requirements. The service candidates will store the results of pre-selection for this block. Edge representing the data flow links two nodes. Head and tail name are short for starting and ending nodes separately. Based on this use-case, for instance, an edge shows the data flows from sensing service S1 to controlling service C2. Its semantic are the output of sensing service S1. An edge will have an bro age if there is another edge owing the same head name. This linked list can helps to traverse the all edges with the same starting node. The implementation of this block diagram is shown in Fig. 4.14.

4.3.3 Pre-selection and Pre-composition of Sensing, Actuating and Controlling Services

When service registries and requirements of each block in the block diagram are set, the pre-selection and pre-composition work can be processed. In this subsection, we need to first select service candidates for each block in the block diagram. One block, especially for controller block, may be divided into one or more sub blocks. Then, the pre-composition graph can be produced by completing the edges of newly generated or originally split blocks.

4.3.3.1 Pre-selection of Sensing, Actuating and Controlling Services

Pre-selection of sensing, actuating and controlling services will associate convenient control services to vertexes in the block diagram according to the corresponding requirements (i.e. requirements associated to the block and globally to the Cloud Control System). Sensor and actuator blocks are elementary blocks and cannot be split. This makes the pre-selection of sensing and actuating services simple. The pre-selection of controlling service is a recursive procedure as controlling services are defined recursively. In conclusion, from complexity to simplicity, the pre-selection of controlling services is discussed first and then that of sensing and actuating services will be analyzed.

Pre-selection of controlling services. According to the definition of control pattern (see Fig. 3.7 in chapter 3), a controller block requirement (see Fig. 4.11) can be represented by a tree structure. In this tree organisation, presented in Fig. 4.15, a node is either an operand or an operator. If it is an operator node, “treeName" may be a composite control pattern (e.g., pid.). So it can be associated to the whole sub tree. Leaf nodes in the tree are all operands. We use a list to store the different nodes of the tree.

As said previously, we use a recursive process to define the “control pattern tree"

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

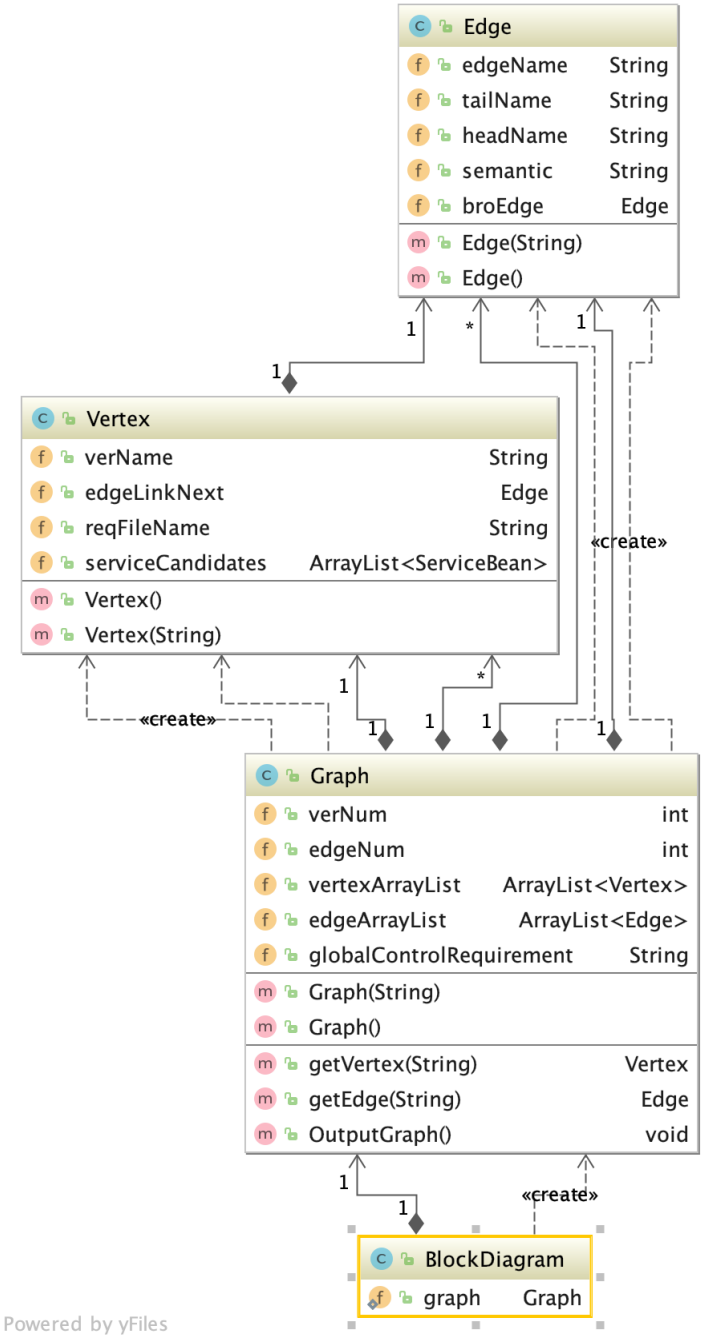

Figure 4.13 – Block diagram related classes

associated to a controller block. To this end, we have defined a “CreateExprTreeRecursive" method in charge of transforming a controller block requirement into a tree. Based on the polish notation, separating operators and operands through brackets, it analyses the control function to identify operators and operand. Each time an operator node is built, child nodes associated to its operands (either leaf of sub-controller) are set. A final leaf operand is set when an empty bracket is found. Algorithm 1 defines more precisely

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

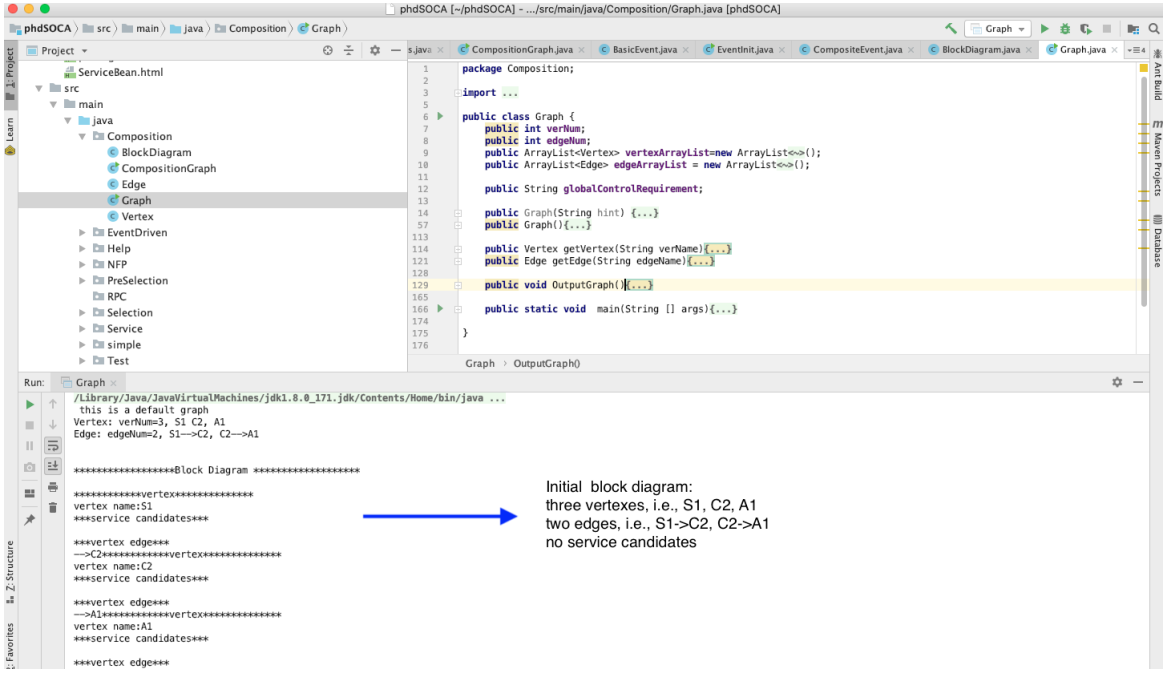

Figure 4.14 – Screen shot of presenting the smart home use case block diagram

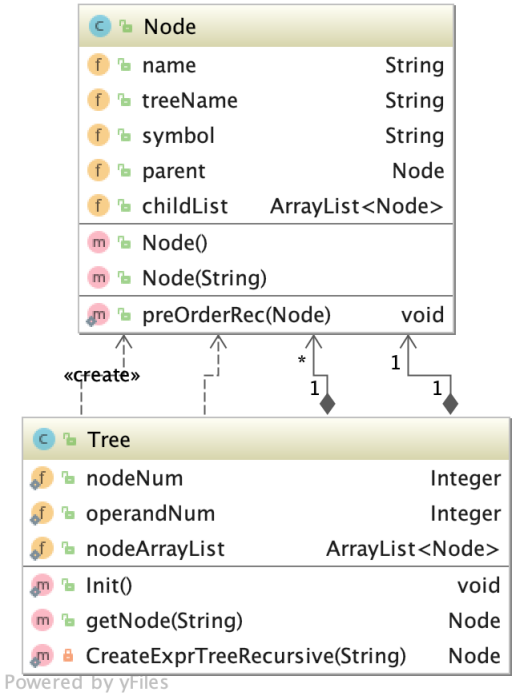

Figure 4.15 – Control Pattern Tree

this process. The final result of constructing this control pattern tree is illustrated in Fig. 4.16.

Based on this expression tree of control pattern, selecting controlling services is also

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Algorithm 1 Create control pattern tree

```
Input: String mathExp                ▷ controller function expression in a Polish notations way.
Output: Node root
function CreateExprTreeRecursive(String mathExp)
    Node root;
    ArrayList<Node> rootChild;
    if mathExp.equals("()") then                                  ▷ this is a operand leaf node
        root.symbol = "operand"; return root;
    else
        ArrayList<Integer> splitLoc;
        Integer bracket=0;
        operatorFlag=true;
        StringmathExp1 = mathExp.substring(1, mathExp.length() − 1);
                                                                  ▷ remove an outside pair of brackets.
        Integerlength = mathExp1.length();
        for i = 0; i < length; i + + do
            if mathExp1.charAt(i) ==′ (′ then
                bracket + +;
            else if mathExp1.charAt(i) ==′ )′ then
                bracket − −;
                if bracket == 0 then                              ▷ a node or another tree (string) appears.
                    splitLoc.add(i);
                    if operatorFlag then ▷ the first one is an operator due to the usage of polish notation.
                        operatorFlag = false;
                        root.symbol = (mathExp1.substring(1, i));    ▷ set the symbol of the operator node.
                        nodeArrayList.add(root)
                    else
                        if mathExp1.length() > i then
                            Integer lowBound = splitLoc.get(splitLoc.size() − 2);
                            String str = mathExp1.substring(lowBound + 1, i + 1);
                            Node operand = CreateExprTreeRecursive(str);
                            rootChild.add(operand);
                            operand.parent = root;
                        end if
                    end if
                end if
            end if
        end for
        root.childList = rootChild;
    end if
return root;
end function
```

a recursive procedure. The details of this procedure is illustrated in algorithm 2. When a new vertex is generated and added into the vertex list in the graph, the edge two controlling vertexes in the graph should be added. For operand node in the control pattern tree, no vertex will be generated. This may change the edges relationships when one controlling vertex are divided into several ones. Pre-selection of controlling service depends the related tables, e.g., “ControlPatterns", “ControllingServices", etc. The result shows that both multiply controller (identity number is 4) and minus controller (identity number is 4) are selected. The pre-selection result of controlling service is shown in Fig. 4.20. Controller block has been split into two vertexes C2 and C2-1. Nevertheless, new

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

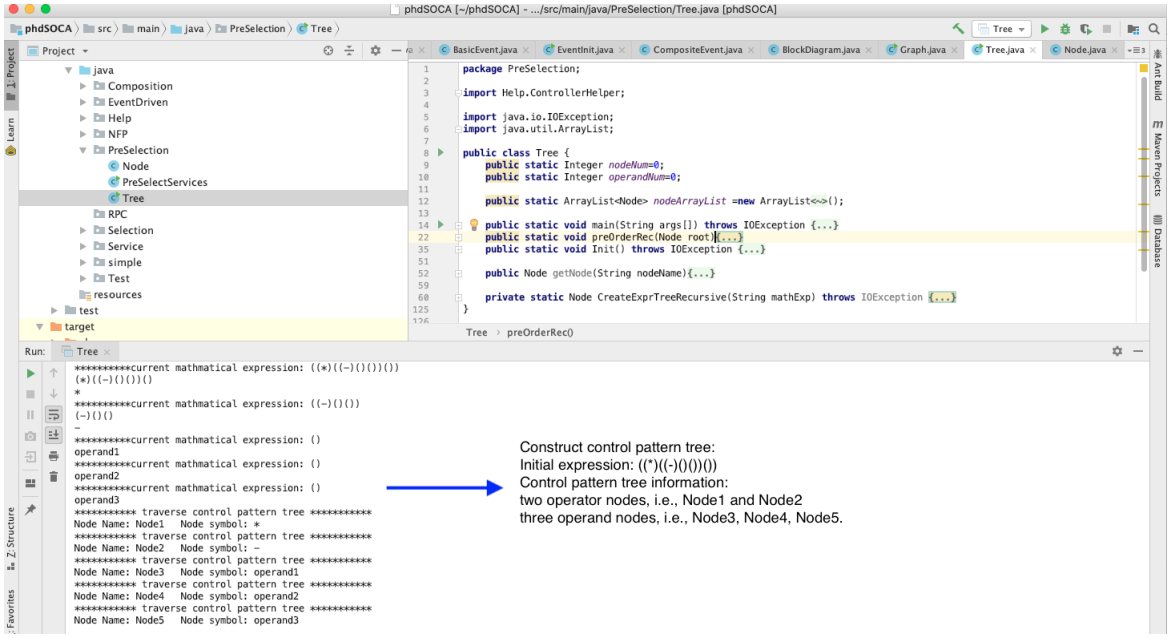

Figure 4.16 – Screen shot of showing the control pattern tree under construction from the controller requirement in our smart home use case

C2 is only a multiply controller instead of the previous proportional control pattern. C2-1 is a minus controller. Once the pre-selection task is accomplished, each node in it should point to one or more services/objects. Sensor S1 has tow service candidates whose identities are 1 and 2 while actuator A1 has only one service candidate with an identity of 6. Each of controller vertexes C2-1 and C2 owns one service candidate. C2 vertex stores a multiply controller recognized by service identity 4. An minus controller whose identity is 8 is saved in the C2-1 vertex. C2-1 is a sub node of C2. This means that C2-1 points to C2 node. Based on different experiments on our smart home use-case, the average cost of pre-selecting controlling services is 3.7 ms (see table 4.9).

Pre-selection of sensing services. The selection of sensing services relies on non functional properties and the controlled variable as the functional descriptions. Controlled variable can be measured by sensing services. A typical selection query of sensing services is shown in Fig. 4.17. The selection pattern explains the standard of pre-selecting sensing services. The query code in a Java language considers the relationships of related data tables. An example of pre-selecting sensing service ("GetSensingServices") involving with two NFPs is illustrated in Fig. 4.18. The result of this operation used in our smart home experiment is that there are two sensors: one is sensing service 1 while the other is sensing service 2 (see Fig. 4.20). The cost of this procedure is 2.4 ms (see table 4.9) according to our experiments using the simplified smart home use case.

Pre-selection of actuating services. Selecting actuating services also depends on the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Algorithm 2 Pre-selection of controlling service

```
Input: Graph graph, Vertex vertex, Node root        ▷ block digram, controller vertex and controller tree
function SelectControllingServices(Graph graph, Vertex vertex, Node root)
    if root == null then return ;
    end if
    if root.childList() == null then return ;                                   ▷ an operand leaf node.
    else
        String treeName=root.treeName;
        if treeName.trim().toString()! = "" then
            Integer conPatID= DBHelper.GetControlPatternIDByConPatName(treeName);
            result=GetControlServicesByControlPatternID(conPatID);
            if o!=result.size() then
                vertex.serviceCandidates = result;
                return ;
            end if
        end if                                          ▷ find the current root node and its child nodes
        StringoperatorName = root.symbol;
        IntegerconPatID = DBHelper.GetControlPatternIDBySymbol(operatorName);
        result = GetControlServicesByControlPatternID(conPatID);
        vertex.serviceCandidates = result;
        ArrayList < Node > rootChild = root.childList;
        for k = 0, m = 1; k < rootChild.size(); k++, m++ do
            Edge edge=new Edge();
            edge.tailName=vertex.verName;
            if rootChild.get(k).childList.size()>0 then                  ▷ ensure this is a operator node
                Vertex vertex1=new Vertex();
                vertex1.verName=vertex.verName+"-"+m;
                edge.edgeName =vertex1.verName+"->"+vertex.verName;
                edge.headName=vertex1.verName;
                edge.broEdge=vertex1.edgeLinkNext;
                vertex1.edgeLinkNext =edge;
                graph.vertexArrayList.add(vertex1);          ▷ insert newly generated vertex1 into graph
                SelectControllingServices(graph, vertex1, rootChild.get(k));
            else
                edge.edgeName=rootChild.get(k).symbol;                   ▷ symbol = "operand"+"k"
                edge.semantic=ControllerHelper.GetSemantic(edge.edgeName);
            end if
            graph.edgeArrayList.add(edge);
        end for
    end if
end function
```

both functional and non functional requirements. Controlled variable can be affected by actuating services. A classical query of selecting actuating services is exhibited (see Fig. 4.19). The implementations are similar to that of sensor (see Fig . 4.18). Several “GetActuatingServices" methods are defined to deal with the different number of non functional property parameters, e.g., 0, 1, 2. There is one temperature actuator available whose identity number is 6 (see Fig. 4.20). The average cost for the query of actuating service is 1.6 ms (see table 4.9) according to our experiments using the simplified smart home use case.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

```
Select services where service type = sensor
                and  property = feature of interest
                and NFP ...
```

(a) selection pattern.

```
String str="select ServiceID from R_Services_NFPs where NFPID = '"+nfpID+"' and
            NFPValue <= '"+nfpValue+"' and ServiceID in"+
            "(select ServiceID from R_Services_NFPs where NFPID = '"+nfpID1+"'
            and NFPValue <= '"+nfpValue1+"' and ServiceID in" +
            "(select ServiceID from SensingServices where
            PropertyID = '"+propertyID+"'))" ;
```

(b) query code

Figure 4.17 – Pre selection of sensing services

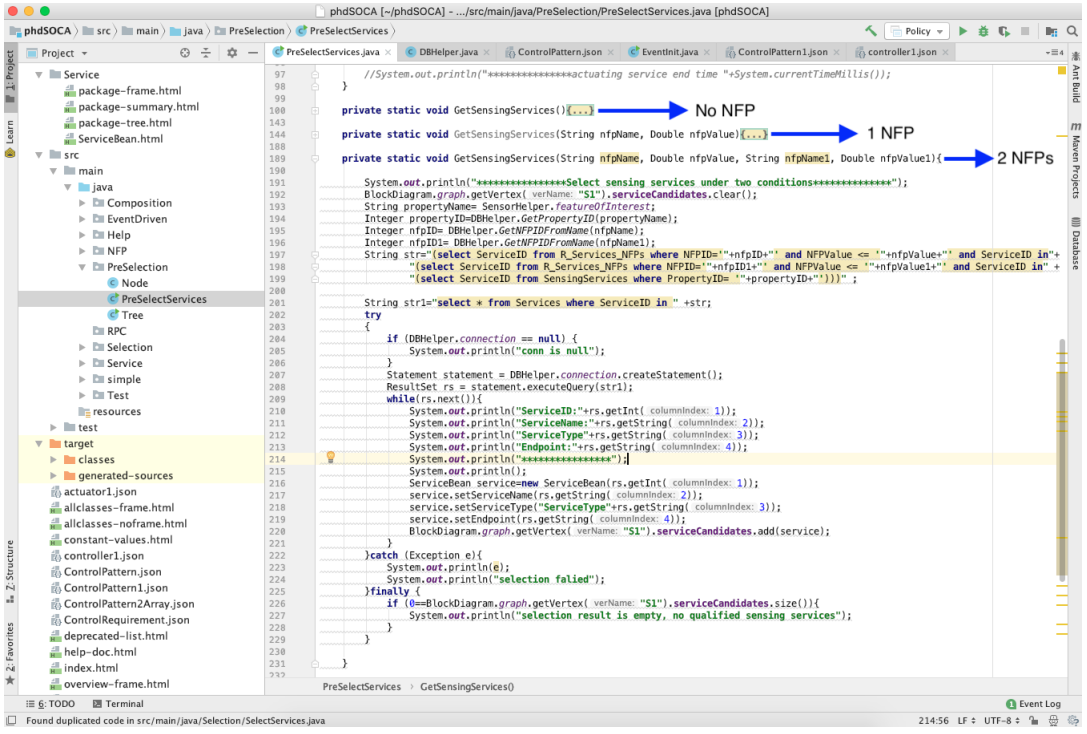

Figure 4.18 – Screen shot of “GetSensingServices" method

4.3.3.2 Pre-composition of Sensing, Actuating and Controlling Services

The final result of this stage is the pre-composition graph, which is used by the composition engine. A pre-composition graph is originated from a block diagram which is described by a “Graph” data structure. It is generated by the service pre-selection according to the control function analysis. After pre-selection of services for each node in the block diagram, the pre-composition graph is almost built. This means some newly generated controlling vertexes are added into the block diagram. The edges between them will be added at the same time. However some edges connecting sensor and con-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

```
Select services where service type = actuator
                and  property = feature of interest
                and NFP ...
```

(a) selection pattern.

```
String str="select ServiceID from R_Services_NFPs where NFPID = '"+nfpID+"' and
            NFPValue <= '"+nfpValue+"' and ServiceID in"+
            "(select ServiceID from R_Services_NFPs where NFPID = '"+nfpID1+"'
            and NFPValue <= '"+nfpValue1+"' and ServiceID in" +
            "(select ServiceID from ActuatingServices where
            PropertyID = '"+propertyID+"'))" ;
```

(b) query code

Figure 4.19 – Pre selection of actuating services

Table 4.9 – Costs of Pre-selecting services

Service type	Sensing service	Actuating service	Controlling service
Cost (ms)	2.4	1.6	3.7

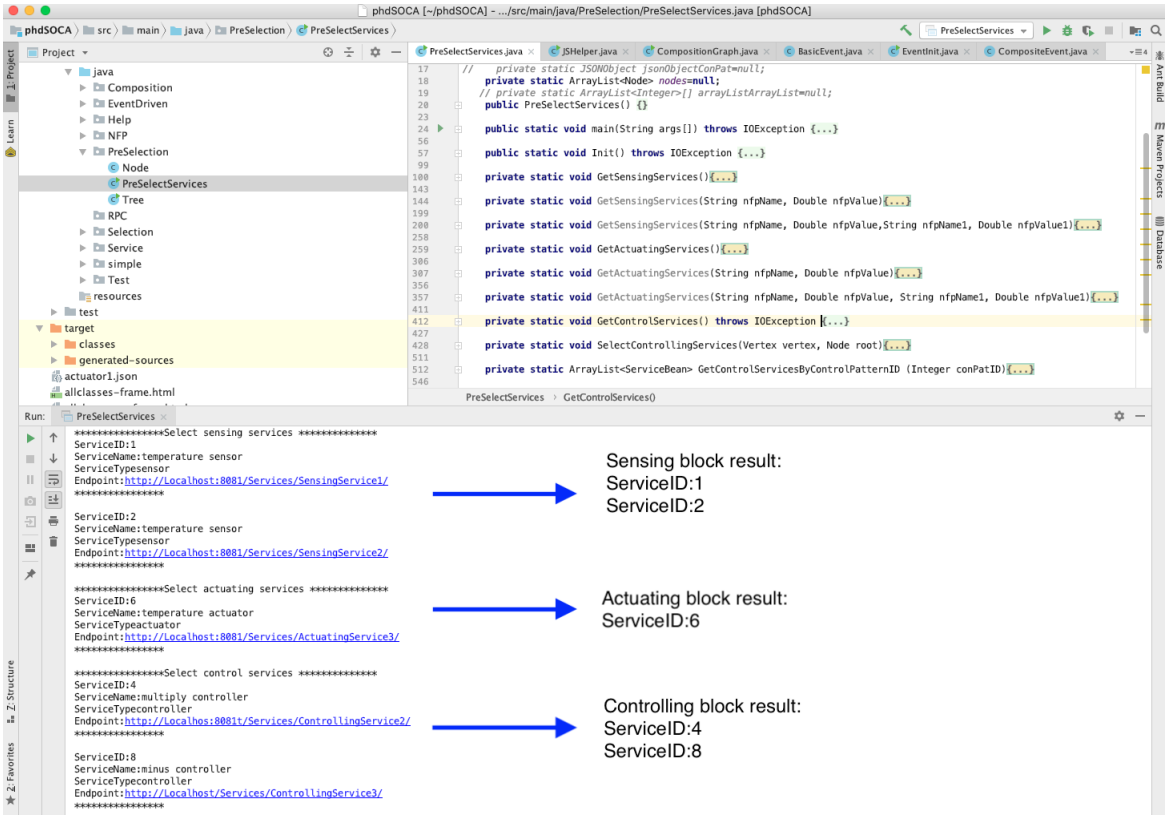

Figure 4.20 – Screen shot of pre-selection result

troller vertex may be changed. Hence in the pre-composition stage, these relationships should modified. A "ModifyEdge" method is defined in the class of pre composition graph. The algorithm of modifying edges to complete a pre-composition graph is illustrated in algorithm 3.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Algorithm 3 Modify edges for nodes in the pre-composition graph

```
1: function MODIFY(Block diagram: graph)
2:     for each controller block in the block diagram do
3:         for each edge pointing to this controller block do
4:             if the operand semantic of a new sub controller equals to that of egde then
5:                 modify the edge tail node to the new controller node.
6:             end if
7:         end for
8:     end for
9: end function
```

Eventually, a block diagram becomes a pre-composition graph although by chance they are the same. A pre-composition graph for the simplified smart home use case (see Fig. 4.21) is produced from its block diagram (see Fig. 4.9) after the pre-selecting sensing service S1, actuating service A1 and controlling service C2. S1 vertex redirects to the C2-1 instead of previous C2. C2 receives the inputs from the C2-1 and removes the connection with S1. Once the composition graph is built, a full pre-composition graph will be attained (see Fig. 4.21). Information of Vertexes are stored in the “Nodes" data table. For each node, the service candidates are saved in the “PCR" data table. These two operations are achieved in the “UpdateDB" function and the implementation details of pre-composition is exhibited in Fig. 4.22.

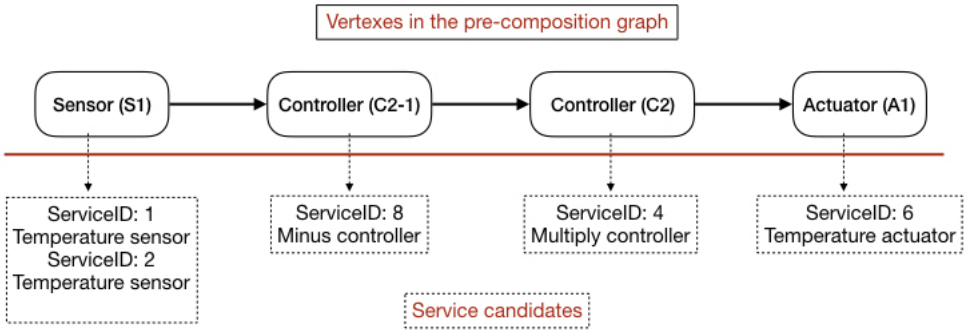

Figure 4.21 – Pre-composition graph for simplified smart home use case

4.3.4 Event Management

The event-driven Cloud control system relies on a data manager in charge of generating events and on an event manager in charge of activating the control services waiting for events as soon as the corresponding event occurs. An event is either a basic event or a composite one. The classes defined for them in the format of UML is illustrated in Fig. 4.23. A basic event can be related to more than composite events. An composite event is involved in at least two basic events. We can see that an event can be subscribed by one or more groups of service candidates. At running time, only one is chosen to work

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

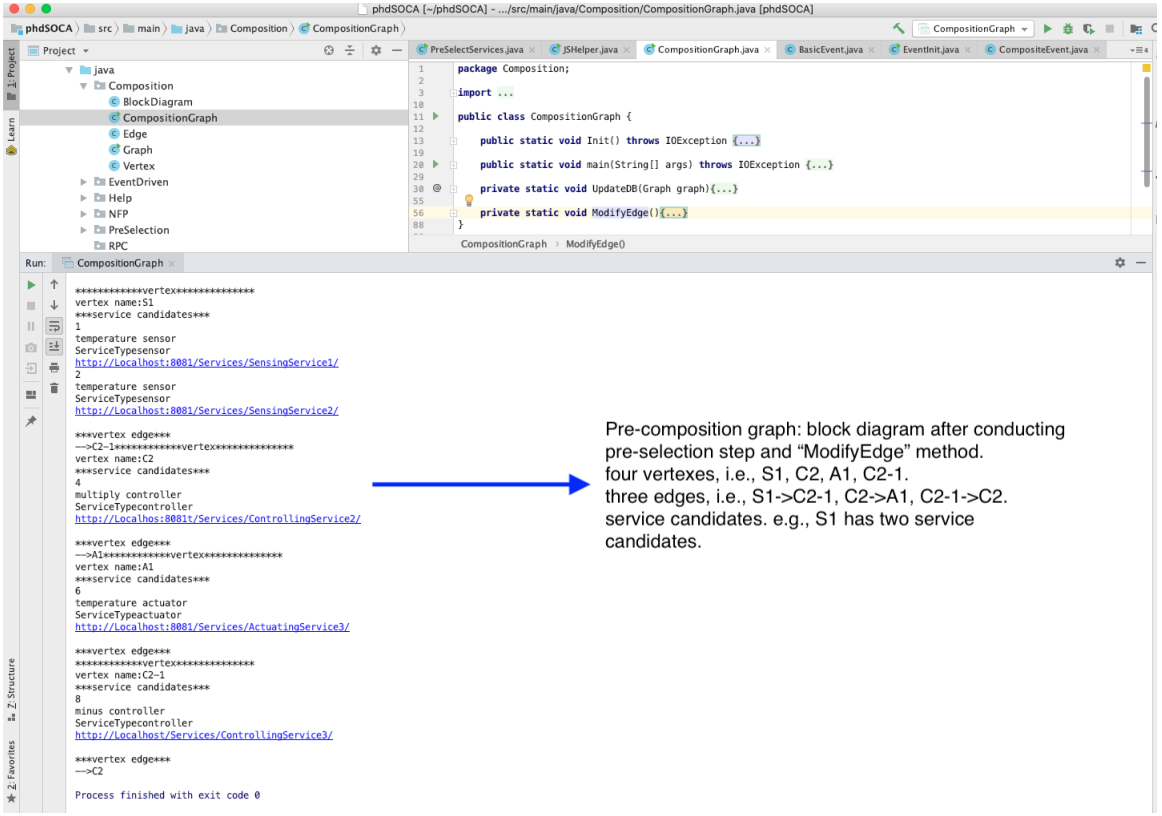

Figure 4.22 – Screen shot of pre-composition result

in each group.

“EventInit” class (see Fig. 4.24) is defined to define basic events and composite events and bind services candidates to events, based on the pre-composition graph. “inputNums” and “outputNums” are two assisted properties aimed at calculating the number of inputs and outputs for each node in the pre-composition graph. The definition of basic events are dependent on the number of inputs of a vertex in the pre-composition graph. If a vertex has an output or if there is an edge starting from the it, a basic event will be defined for it. The definition of composite events relied on the number of inputs of a vertex. When there are more than one vertex points to a specific one, a composite event should be this vertex. Hence the activation of a composite event relies on its involved basic events.

The binding process connecting services to events is based on the pre-composition graph, e.g., a pre-composition graph for simplified smart home use case (see Fig. 4.21). From the input angle, if there is only one vertex (C2) pointing to this vertex (A1), services stored in this vertex (A1) will be bound to the basic event (ebC2) defined by previous one (C2). If there are more than one vertexes flowing to one vertex(e.g., named after N), a composite event (ecN) must be defined and the services bound to this vertex

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

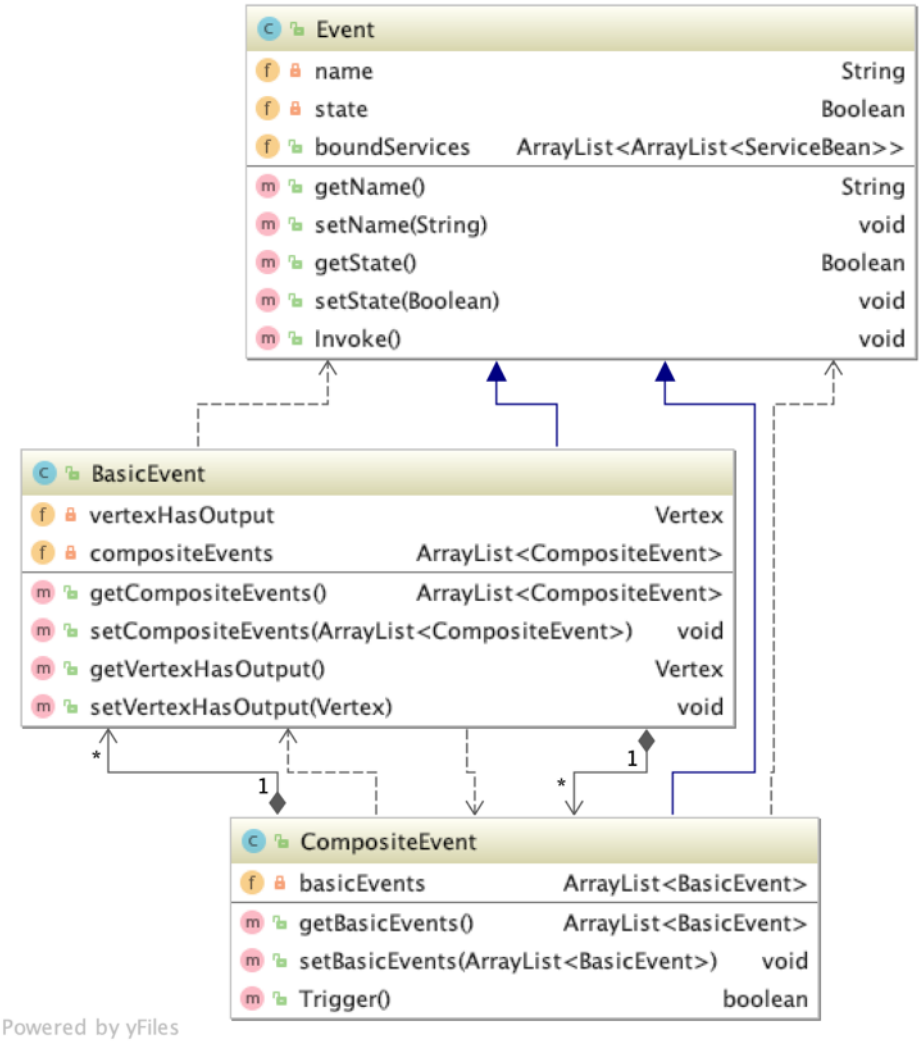

Figure 4.23 – Event structure

will be bound to this composite event. Meanwhile, basic events related to them will be perceived as input events of this composite one. It is noted that sensing service is launched by control application itself, and there is no defined event for actuator vertexes as they have no output. The algorithm of binding services to events is shown in the below (see algorithm 4). The implementations of defining basic and composite events and binding services to them is presented in Fig. 4.25 where our prototype is used on our smart home experiment.

4.4 SOCA at Running Time

To support our SOCA at running time, we need to describe how the sensing, controlling and actuating services interact, and are orchestrated. Control service bus will specify

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

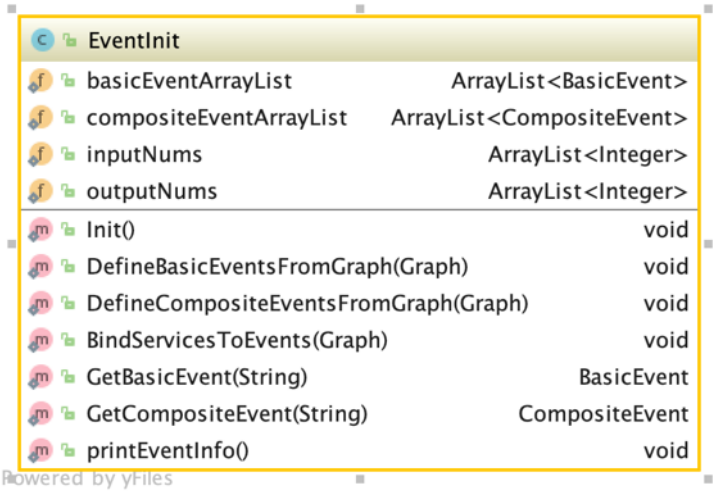

Figure 4.24 – EventInit Class

Algorithm 4 Bind services to events

Input: Graph graph ▷ a pre-composition graph
calculate the number of inputs (inputNums) and outputs (outputNums) of each vertex based on graph.edgeArrayList.
for each vertex in the graph **do** ▷ define basic events
  if $vertex.outputNums >= 1$ **then**
    define a basic event eb (eb+vertex.verName) and eb.vertexHasOutput=vertex.
  end if
end for
for each vertex in the graph **do** ▷ define composite events
  if $vertex.inputNums > 1$ **then**
    define a composite event ec (ec+vertex.verName).
    for each vertex1 points to the vertex **do**
      find the basic event eb1 related to the vertex1.
      ec.basicEvents.add(eb1).
    end for
  end if
end for
for each vertex in the graph **do**
  if $vertex.inputNums == 1$ **then** ▷ bind services to the basic event
    find the basic event eb related to the vertex pointing to the current one.
    eb.boundServices.add(vertex.serviceCandidates);
  else if $vertex.inputNums > 1$ **then** ▷ bind services to the composite event
    find the composite event ec related to the current vertex.
    ec.boundServices.add(vertex.serviceCandidates);
  end if
end for

this running time process conceptually. Thanks to our micro-service artifact tier, logical services do not need to take into account specific interfaces of the CPS devices they will use. Moreover, to support the loosely coupled feature, we organize the service orchestration in an event driven strategy. Due to this event-driven strategy and to the micro-service artifact created for each IoT device that may interact in a Cloud Control System, services do not invoke each other directly: the data produced by a control

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

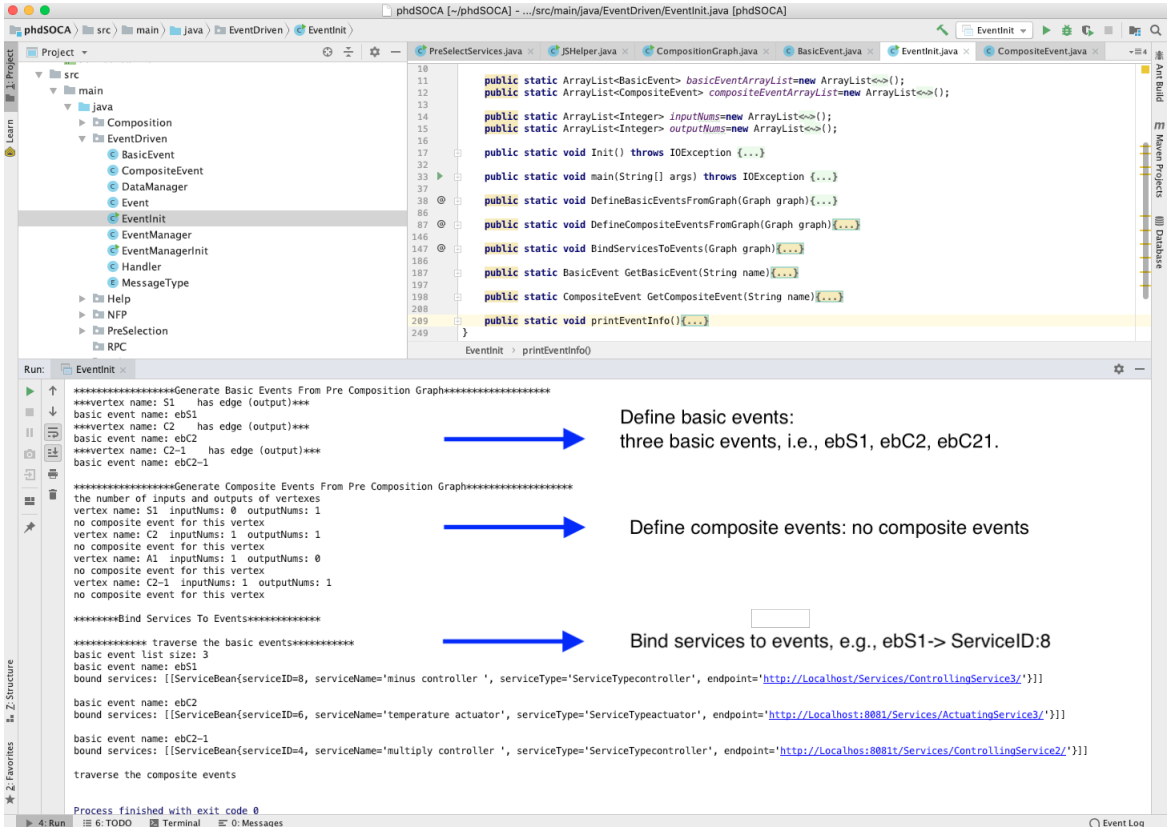

Figure 4.25 – Screen shot of event initialization

service (sensing service or controlling service) is turned into an event. The context manager hosts a late-binding function to select the “best" service candidate waiting to this event and launches it. By this way, our Cloud Control System can fit the context-aware requirement.

4.4.1 Control Service Bus

From control system design space to runtime space, Control Service Bus (CSB) provides an implementation for our SOCA. Given the increasing of transmission speed of communication and computing capabilities of processor in the distributed environment, event-triggered control instead of traditional periodic control attracts attention from academia and industry, and it can be applied in IoT context [59]. This event-driven orchestration “firing" services depending on input events is integrated in our control service bus to support the distributed control application execution (see Fig. 4.26). Event manager provides control system a reactive feature. Orchestrator will select the optimal service for each node in the pre-composition graph, thanks to the decision or constraint from context manager monitoring services. Sometimes, the control application needs to be re-composed once execution context change is detected by the context manager, providing

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

a self-autonomous control feature.

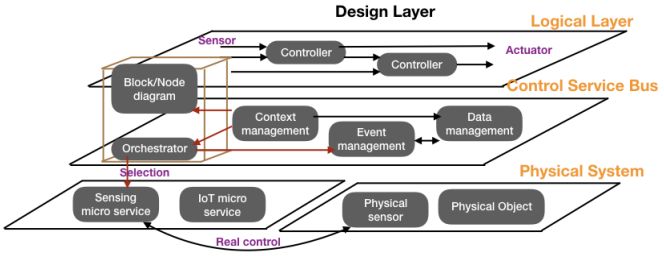

Figure 4.26 – Control service bus

4.4.2 Event-based Orchestration

When an event is triggered, the late selected service(s) via late binding will be called. The “Invoke" method defined in the “Event” class will be called by the event manager to invoke its bound services. In Fig. 4.27, we see that this method depends on the function “GetHelp” specified in the “ClientHelper" class to implement the RPC (via http).

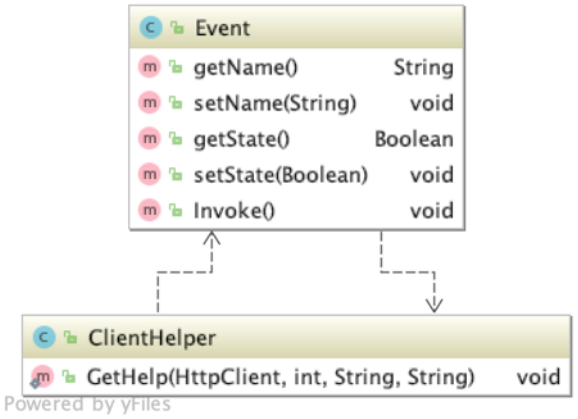

Figure 4.27 – Event invocation-assisted diagram

Each event manager manages some events shared on the control service bus (see Fig. 4.28). Generally speaking, an event manager is responsible for an individual control system. Through naming mechanism of events and event managers, we can easily find the precise event manager based on a event message. Regarding a control system, the event manager begins with “StartSensingServices” defined in the “EventManagerInit” class. When these sensing services are invoked, their outputs will trigger all of basic events related to sensing nodes in the pre-composition graph for the current control system, leveraging the method “FromDataToBasicEvent”.

The method “FromDataToBasicEvent”is described as follows (see Fig. 4.29).

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

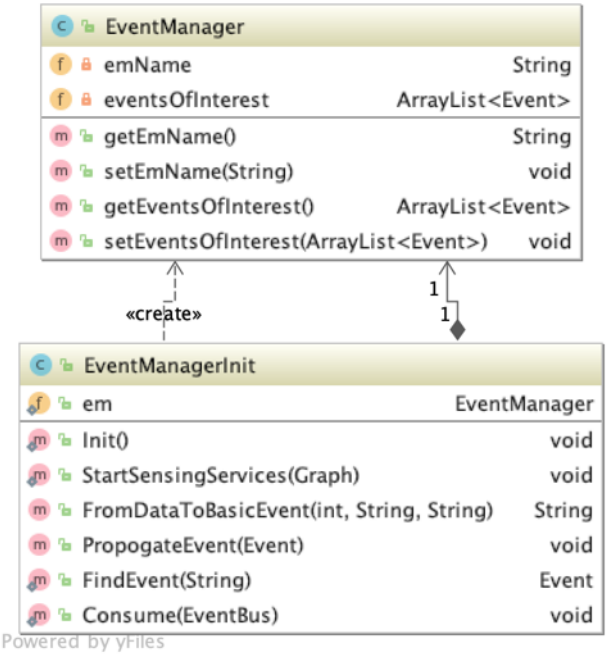

Figure 4.28 – EventManager and EventManagerInit classes

Step 1: Event manager receives the data message from a service whose URL is reachable.

Step 2: Through it, the identity of the service could be achieved, making use of the “GetServiceIDFromEndpoint” method in the “Services" data table.

Step 3: The identity of node (vertex) can be retrieved based on the result of step 2, depending on the “PCR" table where there are a combinational primary key (ServiceID and NodeID).

Step 4: The node name can be easily to get leveraging the node identity number in the “Nodes" table.

Step 5: A basic event name constructed with a prefix “eb” and the node name can be acquired.

Step 6: Through event registry, the basic event can be selected and triggered.

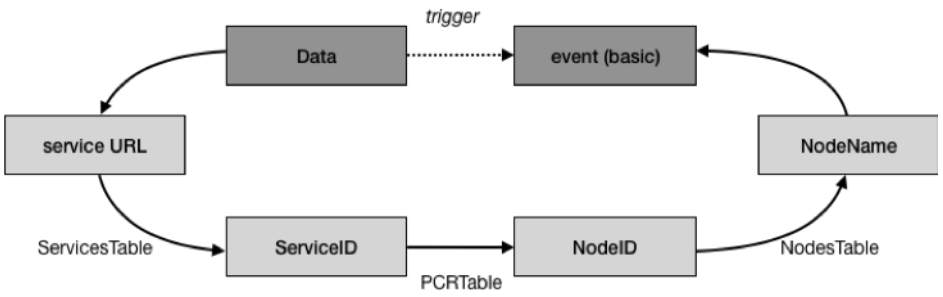

Figure 4.29 – Workflow of a data triggering the basic event

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

To fulfill the loosely coupled event driven principle, we also introduce a data manager in our control service bus. Data manager can store the data sent by sensors or exchanged by controllers. Then, it will trigger basic events associated to a given data availability whereas these basic events are used to trigger composite events. For instance in Fig. 4.30, data manager, run in our smart transportation use case experiment, interacts with "SensingService1", publishes the basic event related to it and save its data. Basic events are triggered and published together with data message on the control service bus leveraging the Event bus provided by the Vertx. The method "Consume" monitors all events and dispatch them to the corresponding services. If the received event is a basic one, "PropagateEvent" method will be called to check the triggering condition of its involved composite events. When the condition is satisfied, the composite event will be triggered and polished on the control service bus. Another important job is to invoke the service subscribing to this event. Then according to these outputs, "FromDataToBasicEvent" will be invoked to trigger basic events related to them and publish them on the service bus. The event interaction diagram is illustrated in Fig. 4.31. Services bound to events will be launched immediately when the corresponding events are triggered [55].

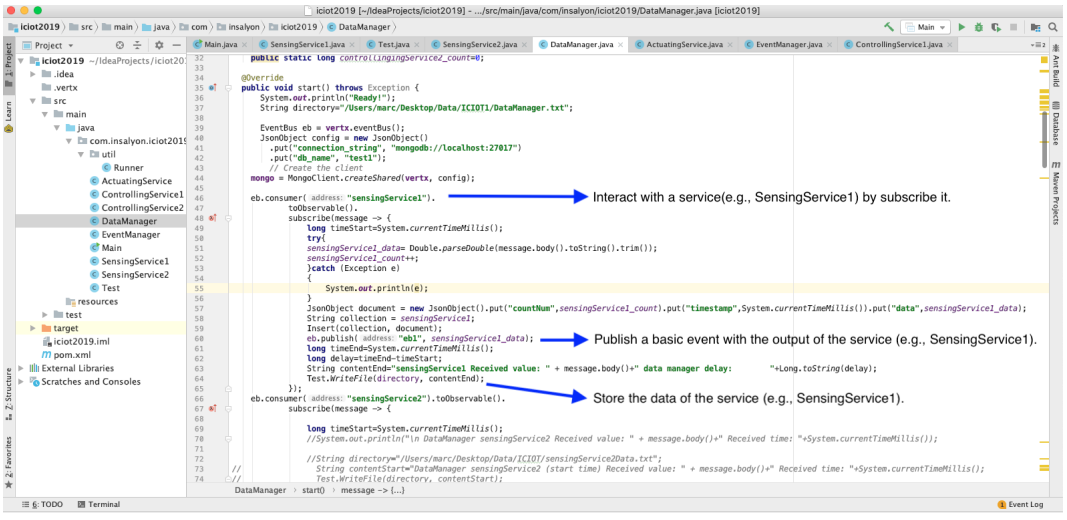

Figure 4.30 – Screen shot of data manager

To evaluate the distributed control application applying event driven orchestration, we also experiment our prototype for the emergency car use case (see Fig. 4.32). Controlling services consisting of route planning controlling service and traffic light controlling service, and control service bus comprised of data manager and event manager are deployed on a cloud server. Virtual Machines (VMs), each of which represents a sensing or actuating service, communicate with the cloud server through different transmission

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

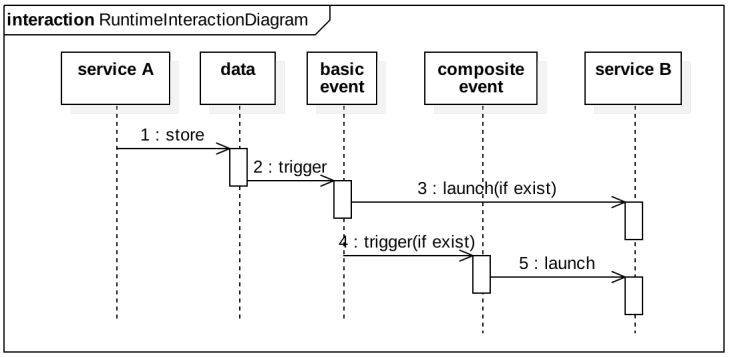

Figure 4.31 – Event interaction diagram

media (e.g., ADSL(Asymmetric Digital Subscriber Line), 4G and wired fiber). Logically distributed VM 1, 2, 3 are visualized from camera capturing sensing service, traffic light actuating service, and GPS sensing service respectively. The prototype is implemented under the environment where Idea IntelliJ IDE, Vertx framework and MongoDB database are selected and installed.

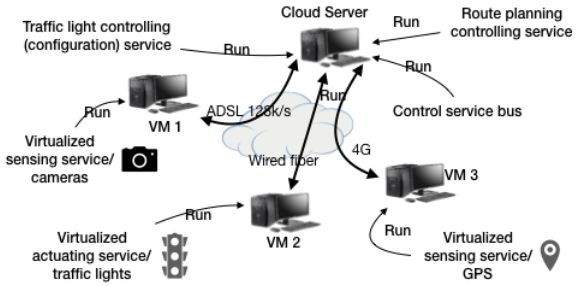

Figure 4.32 – Emergency car leading control prototype

Compared to physically connected control application, our cloud based and event driven one has to pay attention to delays since control architecture must take into account the different delays to assess the control system performance and to check if physical constraints can be met. From our loosely coupled distributed control architecture point of view, three different delays may affect the global control performance: data communication, data management (i.e. the time necessary to store the data and launch the first basic event) and event management (i.e. the time consumed by the process in charge of launching all events that can be activated once a new basic event is produced) as these three components impacts the control process.

Here, we pay attention to communication delay, which takes a major role in the control application performance especially when images need to be transferred. Communication time of different transmission media (wired fiber, ADSL 128 k/s and 4G) are tested. Images manufactured by cameras are transferred through ADSL with the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

speed of 128 k/b and three images are bound as one data package. GPS sensing service is mapped from embedded system in the emergence car and is accessed through 4G. Smart lights are configured after receiving control signals through wired fiber. We take 64 characters as a short message size between cloud server and VM 1, and VM 3. 35 samples are extracted, processed and illustrated in table 4.10. All these measures show that our distributed event-driven control architecture can fit most of the controlled systems delays.

Table 4.10 – Delay of communication

test type			mean	unit	variance
Communication delay	ADSL 128k/s	image	6.2	s	0.6
	4G	short message	33.5	ms	15.7
	Wired fiber	short message	9.4	ms	1.9

Delay time of Data Manger and Event Manager must be measured since they will affect the system performance (e.g., stability). The communication between Data Manager and services is event driven as well and called communication event. In our prototype for event driven orchestration, Event Manger class is implemented to support the composite event firing. Architecture of prototype (see Fig.4.33) shows that data time is the delay which data manager uses to store the received data and trigger corresponding basic event; event time is spent by event manager to trigger one or more composite events based on the former basic event.

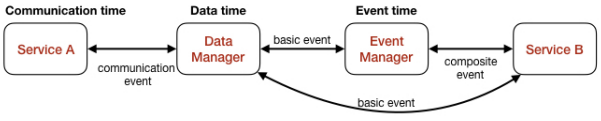

Figure 4.33 – Prototype architecture

As is described in this smart transportation use case, a simplified pre-composition graph for emergency car use case is shown in Fig. 4.34. S1 represents the position sensing service while S2 stands for camera sensing service. C1 and C2 are route planning controller and traffic light controller respectively. A1 indicates the traffic light. So the events defined for this use case and relationships among services, data manager and event manager is shown in table 4.11. Data Manager subscribe the all sensing and actuating services. There are four basic events, i.e., eb1, eb2, eb3 and eb4 defined by the data manager according to the output of “SensingService1", “SensingService2", “ControllingService1",“ControllingService1" respectively. Moreover, two composite events, ec1 and ec2 generated to invoke “ControllingService1" and “ControllingService2" separately, are activated depending on the three basic events, i.e., eb1, eb2, and eb3.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

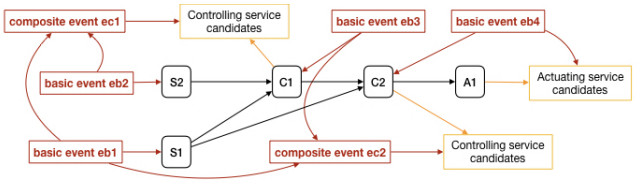

Figure 4.34 – A pre-composition graph for smart transportation use case

Table 4.11 – Event Relationships among entities in smart transportation use case

Service name	Subscribed event name	Produced event name
SensingService1		sensingService1
SensingService2		sensingService2
ControllingService1	ec1	controllingService1
ControllingService2	ec2	controllingService2
ActuatingService	eb4	
DataManager	sensingService1	eb1
	sensingService2	eb2
	controllingService1	eb3
	controllingService2	eb4
EventManager	eb1	ec1
		ec2
	eb2	ec1
	eb3	ec2

Fig. 4.35 reports one of the experiments from our emergency car use case. To measure the cost of data manager and event manager, we deploy these services locally and link them in a fixed way, without the loosely coupled RPC invocation. The experiment result (see table 4.12) shows that on average the delay of Data Manager and Event Manager is 1.0 ms and 0.2 ms. If the control service execution time reaches to 500 ms, the delay of event manager receiving this controlling service output requires 2 s. In conclusion, our solution for designing cloud based control application can be feasible for most controlling service. For some long lasting controlling service, a delay compensation block or optimized event management module is recommended and added into the global control application.

Table 4.12 – Delay of data manager and event manager

delay type	mean	unit	variance
data manager delay	1.0	ms	0.4
event manager delay	0.2	ms	0.2

4.4.3 Context Aware Cloud Control System Context Management

As said previously, the control process is governed according to the context. Thus, we propose a complete policy file integrating five sub policy files, leveraging all NFPs listed in the table 3.1. NFPs violation rule is set to evaluate whether there is a conflict between

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

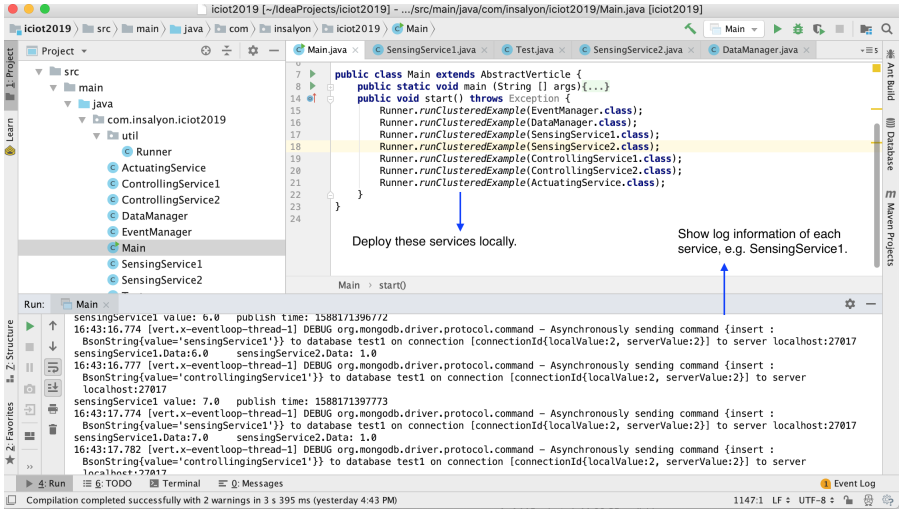

Figure 4.35 – Screen shot of reporting one of our smart transportation experiments

different policies and/or between one unified policy and QoS in SLA (Service Level Agreement) in order to integrate these Non Functional Requirements (NFRs). Including a context monitoring part and a context-aware control manager, the context manager takes a major part in our architecture [54]. The context-aware context manager integrates three functions:

1) NFPs aggregation rule merges NFPs, consolidating NFRs for each node issued from the whole control system.

2) NFPs violation detection relies on a policy matching process to detect if a service “promising" a Non Functional property matches the NFRs or if these requirements are conflicting

3) NFPs priority rule is dedicated to attaching weighed value to each NFP and being able to sort the service candidates in the pre and final selection steps.

In order to integrate these NFRs in the orchestration processes, our context-aware orchestration process consists in several steps, deployed either at design time or at run-time:

1. Attaining a composition graph following control process specification of control block diagram.
2. Capturing context requirements by aggregating policies to set a consistent requirements policy.
3. Achieving policy matching between requirements and QoS policies associated to the different candidate services for violation detection.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

4. Refining selection according to a matching process between the aggregated requirements policy and the different candidate services non functional policy description.
5. Ranking the remaining candidates according to the context multi-criteria evaluation function and performing the late binding operation to launch the “best" services, paying attention to the integration of the necessary mediation services and communication services (if necessary) in the composition graph.
6. Selecting and parameterizing the corresponding NFP monitoring services.

Policy aggregation usually happens on “middle nodes" as one policy is required from the previous node while the other is demanded from the next node, requiring invoking the NFP aggregation function. At run time, comparing the “promised" NFP with their current achievement level is achieved thanks to monitoring services, selected and launched while selecting the service candidate. These monitoring services are used to capture information on surrounding environment, communication, embedded device performance, data, system capability, of control system. The aggregation function defined in table 4.13 will be invoked to achieve one merged policy. An example of policy file (original) is shown in rdf/turtle format (see List 4.1).

```
<http://somewhere/Service2/>
a                          usdl:service ;
csiro:hasLocation   "30 degree,40 degree,5 meter";
fipa:hasCPU                "2 GHz" ;
fipa:hasConnectionMedia    "WIFI" ;
fipa:hasMemory             "50 MB" ;
usdl:RealtimeControlledVariable
"temperature 2 celsius" ;
usdl:hasTransmissionLatency  "0.5 ms"
ssn-system:hasFrequency    "2 ms" ;
ssn-system:hasPrecision    "0.1" ;
ssn-system:hasRemaningSystemLIfeTime "2 year";
ssn-system:hasResponseTime   "0.5 ms";
```

Listing 4.1 – Example of policy file picked from the smart factory use-case experiment

Table 4.13 – Aggregation functions

NFP name	Req Aggregation (Req1, Req2)
frequency	return max(Req1.frequency, Req2.frequency)
responseTime	return min(Req1.responseTime, Req2.responseTime)
precision	Return min(Req1.Precision, Req2.Precision)
remaingSystemLifetime	return max(Req1.remainingSystemLifeTime, Req2.remainingSystemLifeTime)
memory	return max(Req1.memory, Req2.memory)
CPU	return max(Req1.CPU, Req2.CPU)
location	return Req1.location
connectionMedia	if (Req1.connectionMedia = Req2.connectionMedia) return Req1.connection else return communicationServiceMark = 1;
transmissionLatency	return min(Req1.transmissionLatency, Req2.transmissionLatency)

Policy matching is similar to policy aggregation. After NFPs aggregation task is fin-

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Table 4.14 – Comparison functions

NFP name	bool Comparison (Req, QoS)
frequency	return Req.frequency < QoS.frequency
responseTime	return Req.responseTime > QoS.responseTime
precision	return Req.precision > QoS.precision
remaingSystemLifetime	return Req.remainingSystemLifeTime < QoS.remainingSystemLifetime
memory	return Req.memory < QoS.memory
CPU	return return Req.CPU < QoS.CPU
location	return Req.location ≈ QoS.location
connectionMedia	if (communicationServiceMark = 1) ‖ (Req.connection ! = QoS.connection) call communication service; return 0; else return 1;
transmissionLatency	return Req.transmissionLatency > QoS.transmissionLatency

ished, violation detection for every NFP should be accomplished. Violation detection task relies on comparison function defined in the table 4.14. The comparison function is called to check whether there is any violation on NFPs. If there is no contradictory NFP for all NFPs between merged policies from the policy aggregation step and policy defined in the SLA of the given node. Otherwise, the precise service should be substituted.

Late binding is intended to bind the node from the composition graph with most appropriate service/object selected from the refined group correspondingly. The optimal candidate is achieved depending on the ranking function provided by context manager. Orchestrator will mark the “best" service for each node(see the sequence diagram Fig. 4.36)

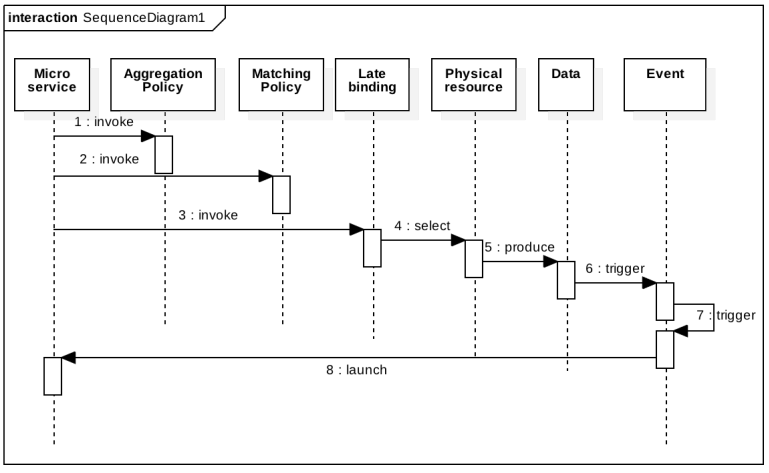

Figure 4.36 – Orchestrator working sequence diagram

Based on our prototype, we also experiment context-aware orchestration on the smart factory use case. We present hereafter the context-aware orchestrator performance analysis. Here, it is used to specify the NFPs from views of requirements and service qualities

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

(see List 4.1). Fig. 4.37 presents the screen shot associated to the policies agregation and matching process.

Table 4.15 shows that the time cost of policy aggregation and matching does not make much difference when the number of NFPs changes from 1 to 10. The mean time cost of policy aggregation functions is less than 6 μs while that of policy comparison is less than 5 μs, invoking them 10 times respectively. In fact, the cost is mostly due to the database access and NFP instance construction. If just running each policy once, the result is easy to disturb. Once the policy is built, the policy matching and/or aggregation costs are very low, allowing the late binding and context-aware orchestrator integration as a cloud-based component.

Table 4.15 – Costs of aggregation and matching policy

Mean time(μs) / NFP number / Policy name	1	5	10
Aggregation policy	5.6	5.9	4.6
Matching policy	4.0	4.0	4.2

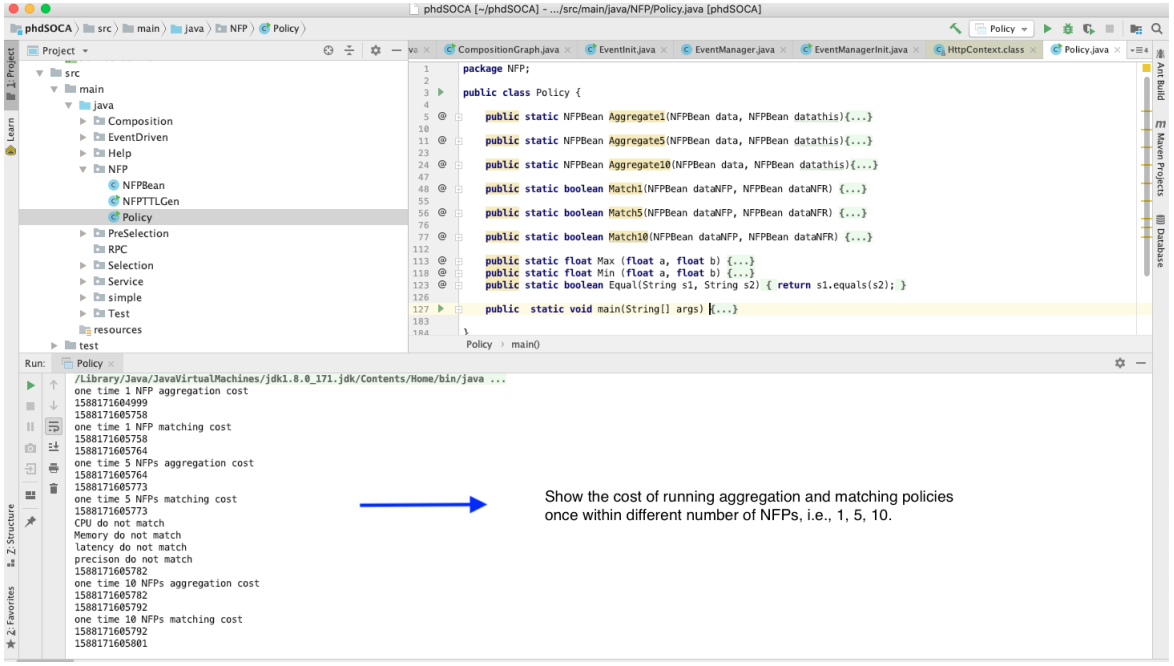

Figure 4.37 – Screen shot of reporting the policy aggregation and matching process execution

4.5 Conclusion

In this chapter, we have presented our multi layer service oriented control architecture. One one hand, it connects logical and physical world closely, integrating different

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

architectures, i.e., SOA, enterprise ontology, Business Process, NIST IoT Reference architecture. On the other hand, it facilitates the development the context aware and event driven cloud control system.

At design time, we focus the pre-selection and pre-composition of sensing, controlling and actuating services. Before selection, service registry and requirements for each module of them are given. After pre-selecting of each block (sensor, controller and actuator) in the block diagram, a pre-composition graph where each node stores no less than one service candidate is attained. , a simplified smart home use case is analyzed To evaluate pre-selection and pre-composition of invovled sensing, controlling and actuating services.

At running time, control service bus integrating event manager and context manager is proposed. An event driven cloud control system can be implemented thanks to the event manger. An adaptive and optimal cloud control system are targeted as the context manger can help to make the late binding feasible and reorganize the control application.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Chapter 5

Conclusion

Industry 4.0 premises the research environment where distributed models, architectures and systems that can cooperate with each other are required in this collaborative world thanks to the support of IoT and cloud computing techniques. Entering into control field, cloud control system as a new generation of control system based on networked control system is proposed to embrace the idea of control service because of the following two reasons: 1) usage rate of devices can reach to its maximum as much as possible due to the shareability on the Internet; 2) cost of energy / economy may fall to its minimum as possible as enterprise requires thanks to global optimization and scheduling. Cloud control system also brings some challenges: 1) how to model involved sensing, controlling and actuating services in a unified way; 2) how these services interact to exchange the information; and 3) how it can be adaptive to meet the customized user' requirements and fast changing environment. This may be similar to the way traditional Business Processes are engineered, as shown in chapter 2. On one hand, enterprise architecture frameworks and Service Oriented Architecture provide a strong background to set adaptive and loosely coupled systems. Nevertheless, although products and information on the production process are introduced in RAMI 4.0, these models lacks of taking into account the physical devices supporting the control process. On the other hand, IoT reference architectures and current Cloud Control systems are mostly focused on CPS description and pay a poor attention on the way they can be composed and orchestrated to support real reusable Control as a Service feature. To overcome these limits, we have first defined a Control as a Service model, gathering functional and non functional requirements. To allow users to identify clearly these CaaS elements, we have proposed a complete ontology gathering requirements, devices and process information. Then our Service Oriented Control Architecture integrates both traditional SOA and IoT

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

reference models to support a consistent organisation of the Cloud Control System. By adapting the service selection, composition and orchestration of these control services to support contextual and event-driven orchestration, our architecture provides a strong support to design and orchestrate Cloud Control systems.

Based on our current work, there are still aspects that needs to be further investigated:

- First, we have focused on Cloud control without detailing the way controllers are designed. To overcome this first limit, design of discrete controller in a distributed environment and complex situations, e.g., data package loss, communication delay, system frequency, non linear system ought to be researched. This is the prerequisite of Cloud control in this thesis.

- Second, the definition of control pattern is limited to one output although an abstract controller may have multi outputs. To solve this limit, the controller model should be enriched. Currently, state function is persuasively applied to analyze and design control system. As a consequence, the controller model represented by state function can be more reasonably.

- Third, our event organization is currently centralized. In order to increase CaaS reusability and shareability (as for other Cloud components), decentralized event organisation is requested.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Bibliography

[1] IBM WebSphere Developer Technical Journal: A practical intro-duction to message mediation – Part 1 available on (http://www.ibm.com/developerworks/websphere/). 27

[2] Zachman Framework, November 2019. Page Version ID: 924621079. https://en.wikipedia.org/wiki/Zachman_Framework. ix, 16

[3] Federal enterprise architecture, January 2020. Page Version ID: 938260502. https://en.wikipedia.org/wiki/Federal_enterprise_architecture. ix, 18

[4] Mohammad Aazam, Imran Khan, Aymen Abdullah Alsaffar, and Eui-Nam Huh. Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved. In *Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on*, pages 414–419. IEEE, 2014. 43

[5] Reiner Anderl. Industrie 4.0–Digital transformation in product engineering and production. In *21st International Seminar on High Technology-Smart Products and Smart Production. At Piracicaba (SP), Brazil*, 2016. 18

[6] Mahyar Azarmipour, Haitham Elfaham, Caspar Gries, and Ulrich Epple. PLC 4.0: A Control System for Industry 4.0. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 5513–5518. IEEE, 2019. 40, 41

[7] T. Bangemann, C. Bauer, H. Bedenbender, M. Diesner, U. Epple, F. Elmas, J. Friedrich, T. Goldschmidt, F. Göbe, and S. Grüner. Industrie 4.0-Technical Assets: Basic terminology concepts life cycles and administration models. *VDI/VDE and ZVEI*, 2016. ix, 19

[8] Julien Bidot, Christos Goumopoulos, and Ioannis Calemis. Using ai planning and late binding for managing service workflows in intelligent environments. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, pages 156–163. IEEE, 2011. 44

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[9] Mike Botts, George Percivall, Carl Reed, and John Davidson. OGC® sensor web enablement: Overview and high level architecture. *GeoSensor networks*, pages 175–190, 2008. 31, 41

[10] Matt Calder, Robert A. Morris, and Francesco Peri. Machine reasoning about anomalous sensor data. *Ecological Informatics*, 5(1):9–18, 2010. 33, 41

[11] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, Francesco Perfetto, and Maria Luisa Villani. Service composition (re) binding driven by application–specific qos. In *International Conference on Service-Oriented Computing*, pages 141–152. Springer, 2006. 44

[12] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1069–1075. ACM, 2005. 44

[13] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. Qos-aware replanning of composite web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 121–129. IEEE, 2005. 45

[14] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software*, 81(10):1754–1769, 2008. 44

[15] S. Chaari, F. Biennier, J. Favrel, and C. Benamar. Towards a service-oriented enterprise based on business components identification. In Ricardo J. Gonçalves, Jörg P. Müller, Kai Mertins, and Martin Zelm, editors, *Enterprise Interoperability II*, pages 495–506. Springer London, London, 2007. 26

[16] Edgar Chacón and Orestes Llanes Santiago. Definition of a control service in the application layer of the ISO/OSI reference model for control stations. *Computers in industry*, 20(2):187–192, 1992. 40, 41

[17] Pierre Châtel, Jacques Malenfant, and Isis Truck. Qos-based late-binding of service invocations in adaptive business processes. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 227–234. IEEE, 2010. 44

[18] A. W. Colombo, F. Jammes, H. Smit, R. Harrison, JL Martinez Lastra, and I. M. Delamer. Service-oriented architectures for collaborative automation. In *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*, pages 6–pp. IEEE, 2005. 38

[19] Armando W. Colombo, Ronald Schoop, and Ralf Neubert. An agent-based intelligent control platform for industrial holonic manufacturing systems. *IEEE Transactions on Industrial Electronics*, 53(1):322–337, 2006. 20

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[20] I. E. Commission. Enterprise-Control System Integration Part 1: Models and Terminology. *Comitê: TC 184/SC 5*, 2013. 27

[21] Scott de Deugd, Randy Carroll, Kevin Kelly, Bill Millett, and Jeffrey Ricker. SODA: Service oriented device architecture. *IEEE Pervasive Computing*, 5(3):94–96, 2006. 37

[22] Manuel Díaz, Cristian Martín, and Bartolomé Rubio. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, 67:99–117, May 2016. 43

[23] Jan LG Dietz. Enterprise ontology-understanding the essence of organizational operation. In *Enterprise Information Systems VII*, pages 19–30. Springer, 2007. ix, 28

[24] Simon Dobson, Graeme Stevenson, Stephen Knox, and Paddy Nixon. Ontonym: a collection of upper ontologies for developing pervasive systems. 2009. 33, 41

[25] David Edmond, J. OSullivan, and A. ter Hofstede. What's in a service? Towards accurate description of non-functional service properties. *Distributed and Parallel Databases Journal*, 12:117–133, 2002. 41

[26] Mohamad Eid, Ramiro Liscano, and Abdulmotaleb El Saddik. A novel ontology for sensor networks data. In *Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on*, pages 75–79. IEEE, 2006. 33, 41

[27] Mohamad Eid, Ramiro Liscano, and Abdulmotaleb El Saddik. A universal ontology for sensor networks data. In *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*, pages 59–62. IEEE, 2007. 33, 41

[28] Emna Fki, Said Tazi, and Khalil Drira. Automated and flexible composition based on abstract services for a better adaptation to user intentions. *Future Generation Computer Systems*, 68:376–390, 2017. 43, 44

[29] Holger Flatt, Sebastian Schriegel, Jurgen Jasperneite, Henning Trsek, and Heiko Adamczyk. Analysis of the Cyber-Security of industry 4.0 technologies based on RAMI 4.0 and identification of requirements. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, Berlin, Germany, September 2016. IEEE. 18

[30] Matthias Fluegge, Ivo JG Santos, Neil Paiva Tizzo, and Edmundo RM Madeira. Challenges and techniques on the road to dynamically compose web services. In *Proceedings of the 6th international conference on Web engineering*, pages 40–47. ACM, 2006. 43, 44

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[31] E. Fosse and C. L. Delp. Systems engineering interfaces: A model based approach. pages 1–8. IEEE, March 2013. 23

[32] Xiaohua Ge, Fuwen Yang, and Qing-Long Han. Distributed networked control systems: A brief overview. *Information Sciences*, 380:117–131, 2017. ix, 39, 40

[33] Morteza Ghobakhloo. The future of manufacturing industry: a strategic roadmap toward Industry 4.0. 2018. cites: ghobakhloo_future_2018. 12

[34] Omid Givehchi, Jahanzaib Imtiaz, Henning Trsek, and Juergen Jasperneite. Control-as-a-service from the cloud: A case study for using virtualized PLCs. In *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, pages 1–4. IEEE, 2014. 40, 41

[35] Steven L Goldman. Agile competitors and virtual organizations. *Strategies for enriching the customer*, 1995. 26

[36] Farid Golnaraghi and Benjamin C. Kuo. *Automatic control systems*. Wiley, Hoboken, NJ, 9. ed edition, 2010. OCLC: 637226063. 2

[37] Caleb Goodwin and David J. Russomanno. An ontology-based sensor network prototype environment. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, pages 1–2, 2006. 32

[38] John Graybeal, Luis Bermudez, Philip Bogden, Steven Miller, and Stephanie Watson. 'Marine metadata interoperability project: leading to collaboration. In *2005 IEEE International Symposium on Mass Storage Systems and Technology*, pages 14–18. IEEE, 2005. 31, 41

[39] Christin Groba. Synchronising service compositions in dynamic ad hoc environments. In *Mobile Services (MS), 2012 IEEE First International Conference on*, pages 56–63. IEEE, 2012. 44

[40] Hedi Haddad, Zied Bouyahia, and Nafaâ Jabeur. Towards a Three-Level Framework for IoT Redundancy Control through an Explicit Spatio-Temporal Data Model. *Procedia Computer Science*, 109:664–671, 2017. 40, 41

[41] Son N. Han, Imran Khan, Gyu Myoung Lee, Noel Crespi, and Roch H. Glitho. Service composition for IP smart object using realtime Web protocols: Concept and research challenges. *Computer Standards & Interfaces*, 43:79–90, 2016. 43

[42] Wu He and Li Da Xu. Integration of distributed enterprise applications: A survey. *IEEE Transactions on Industrial Informatics*, 10(1):35–42, 2014. 25

[43] V. Villaseñor Herrera, A. Bepperling, A. Lobov, H. Smit, A. W. Colombo, and JL Martinez Lastra. Integration of multi-agent systems and service-oriented architecture for industrial automation. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*, pages 768–773. IEEE, 2008. 38

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[44] Arthur Herzog and Alejandro Buchmann. Predefined classification for mixed mode environments. *Behavioral Ecology*, pages 134–141, 2009. 32, 41

[45] Arthur Herzog, Daniel Jacobi, and Alejandro Buchmann. A3me-an Agent-Based middleware approach for mixed mode environments. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*, pages 191–196. IEEE, 2008. 32, 41

[46] Yuheng Hu, Zhendong Wu, and Ming Guo. Ontology driven adaptive data processing in wireless sensor networks. In *Proceedings of the 2nd international conference on Scalable information systems*, page 46. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007. 33, 41

[47] Shruti A. Jaishetty and Rekha Patil. IoT Sensor Network Based Approach for Agricultural Field Monitoring and Control. *IJRET, eISSN*, pages 2319–1163, 2016. 40, 41

[48] François Jammes and Harm Smit. Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics*, 1(1):62–70, 2005. 37

[49] Henning Kagermann, Wolf-Dieter Lukas, and Wolfgang Wahlster. Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. *VDI nachrichten*, 13:11, 2011. 12

[50] Ajay Kattepur, Nikolaos Georgantas, and Valerie Issarny. QoS composition and analysis in reconfigurable web services choreographies. In *Web Services (ICWS), 2013 IEEE 20th International Conference on*, pages 235–242. IEEE, 2013. 44

[51] Jürgen Kress, Berthold Maier, Hajo Normann, Danilo Schmeidel, Guido Schmutz, Bernd Trops, and Torsten Winterberg Utschig. Enterprise service bus. *an Industrial SOA*, pages 1–10, 2013. 25

[52] Rumen Kyusakov, Jens Eliasson, Jerker Delsing, Jan van Deventer, and Jonas Gustafsson. Integration of wireless sensor and actuator nodes with IT infrastructure using service-oriented architecture. *IEEE Transactions on industrial informatics*, 9(1):43–51, 2013. 37

[53] Juan Li. *Business as a service multi-layer governance architecture*. PhD Thesis, Lyon, INSA, 2014. 15

[54] Minhu Lyu, Hind Benfenatki, Frédérique Biennier, and Parisa Ghodous. Control as a service architecture to support context-aware control application development. *IFAC-PapersOnLine*, 52(13):1085–1090, 2019. 58, 100

[55] Minhu Lyu, Frederique Biennier, and Parisa Ghodous. Control as a service architecture to support cloud-based and event-driven control application development.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

In *2019 IEEE International Congress on Internet of Things (ICIOT)*, pages 41–49. IEEE, 2019. 61, 68, 96

[56] Minhu Lyu, Frédérique Biennier, and Parisa Ghodous. Integration of Ontologies to Support Control as a Service in an Industry 4.0 Context. IEEE, 2019. ix, 31, 62

[57] Magdi S. Mahmoud. Cloud-Based Control Systems: Basics and Beyond. *Journal of Physics: Conference Series*, 1334:012006, October 2019. 3

[58] Rita Di Mascio. Service process control: conceptualising a service as a feedback control system. *Journal of Process Control*, 12(2):221–232, 2002. 40, 41

[59] Manuel Mazo and Paulo Tabuada. Decentralized Event-Triggered Control Over Wireless Sensor/Actuator Networks. *IEEE Transactions on Automatic Control*, 56(10):2456–2461, October 2011. 93

[60] J. Marco Mendes, Paulo Leitão, Francisco Restivo, and Armando W. Colombo. Composition of Petri nets models in service-oriented industrial automation. In *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pages 578–583. IEEE, 2010. 44

[61] João M. Mendes, Paulo Leitão, Armando W. Colombo, and Francisco Restivo. Service-oriented control architecture for reconfigurable production systems. 2008. 38

[62] João M. Mendes, Paulo Leitão, Francisco Restivo, Armando W. Colombo, and Axel Bepperling. Engineering of service-oriented automation systems: a survey. In *Innovative Production Machines and Systems: Fourth I* PROMS Virtual International Conference, 2008*, 2008. 37

[63] Ji Mi. Blue ocean strategy. *Wiley Encyclopedia of Management*, pages 1–1, 2015. Publisher: Wiley Online Library. 1

[64] Nikola Milanovic and Miroslaw Malek. Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51–59, 2004. 44

[65] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641, 2016. 12

[66] László Monostori. Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. *Procedia CIRP*, 17:9–13, 2014. 12

[67] Adina Mosincat and Walter Binder. Automated maintenance of service compositions with SLA violation detection and dynamic binding. *International Journal on Software Tools for Technology Transfer*, 13(2):167–179, April 2011. 44

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[68] Holger Neuhaus and Michael Compton. The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor assets. 2009. 31, 41

[69] Fu Ning, Duan Junhua, and Guo Yan. A Service Composition Environment Based on Enterprise Service Bus. In *Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on andand Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom)*, pages 738–743. IEEE, 2014. 25

[70] Carla Marques Pereira and Pedro Sousa. A method to define an Enterprise Architecture using the Zachman Framework. In *Proceedings of the 2004 ACM symposium on Applied computing - SAC '04*, page 1366, Nicosia, Cyprus, 2004. ACM Press. 16

[71] Nataša Popović and Milica Naumović. Networked and Cloud Control Systems-Modern Challenges in Control Engineering. *International Journal of Electrical Engineering and Computing*, 2(2):91–100, 2018. 3

[72] María Dolores R-moreno, Daniel Borrajo, Amedeo Cesta, and Angelo Oddi. Integrating planning and scheduling in workflow domains. *Expert Systems with Applications*, 33(2):389–406, 2007. 45

[73] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010. 12

[74] Partha Pratim Ray. A survey of IoT cloud platforms. *Future Computing and Informatics Journal*, 1(1-2):35–46, December 2016. 35

[75] P.P. Ray. A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*, October 2016. 35

[76] Laurynas Riliskis, James Hong, and Philip Levis. Ravel: Programming IoT Applications as Distributed Models, Views, and Controllers. In *Proceedings of the 2015 International Workshop on Internet of Things towards Applications - IoT-App '15*, pages 1–6, Seoul, South Korea, 2015. ACM Press. 34

[77] Alexandre Robin and Michael E. Botts. Creation of Specific SensorML Process Models. *Earth System Science Center-NSSTC, University of Alabama in Huntsville (UAH), HUNTSVILLE, AL*, 35899, 2006. 31, 41

[78] Sidney Rosario, Albert Benveniste, and Claude Jard. Flexible probabilistic QoS management of transaction based web services orchestrations. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 107–114. IEEE, 2009. 44

[79] David J. Russomanno, Cartik R. Kothari, and Omoju A. Thomas. Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In *IC-AI*, pages 637–643, 2005. 31, 41

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[80] Silviu Răileanu, Florin Anton, Theodor Borangiu, Silvia Anton, and Maximilian Nicolae. A cloud-based manufacturing control system with data integration from multiple autonomous agents. *Computers in Industry*, 102:50–61, November 2018. 20

[81] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The Enterprise Service Bus: Making service-oriented architecture real. *IBM Systems Journal*, 44(4):781–797, 2005. 25

[82] Dimitrios Serpanos and Marilyn Wolf. *Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies*. Springer, 2017. 34

[83] Roger Sessions. Comparison of the top four enterprise architecture methodologies. 2007. 15, 18

[84] Graeme Stevenson, Stephen Knox, Simon Dobson, and Paddy Nixon. Ontonym: a collection of upper ontologies for developing pervasive systems. In *Proceedings of the 1st Workshop on Context, Information and Ontologies*, page 9. ACM, 2009. 33

[85] Joanne M. Sulek, Ann Marucheck, and Mary R. Lind. Measuring performance in multi-stage service operations: An application of cause selecting control charts. *Journal of Operations Management*, 24(5):711–727, September 2006. 40, 41

[86] Manji SUZUKI. Managing Complexity Using Model-Based Design in the Context of Application Life-Cycle Management. In *Mathworks Automotive Conference, Stuttgart*, 2012. 39

[87] Fei Tao, Ying Cheng, Li Da Xu, Lin Zhang, and Bo Hu Li. CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. *IEEE Transactions on Industrial Informatics*, 10(2):1435–1442, 2014. 19

[88] Fei Tao, Ying Zuo, Li Da Xu, and Lin Zhang. IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. *IEEE Trans. Industrial Informatics*, 10(2):1547–1557, 2014. 19

[89] Yodyium Tipsuwan and Mo-Yuen Chow. Control methodologies in networked control systems. *Control engineering practice*, 11(10):1099–1111, 2003. 39

[90] Jihed Touzi, Frédérick Bénaben, and Hervé Pingaud. Model transformation of collaborative business process into mediation information system. *IFAC Proceedings Volumes*, 41(2):13857–13862, 2008. 27

[91] James Truchard. Introduction and Welcome to NIWeek 2013 - National Instruments. 12

[92] A. Underbrink, K. Witt, J. Stanley, and D. Mandl. Autonomous mission operations for sensor webs. In *AGU Fall Meeting Abstracts*, 2008. 32, 41

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[93] Lise Urbaczewski and Stevan Mrdalj. A comparison of enterprise architecture frameworks. *Issues in Information Systems*, 7(2):18–23, 2006. 15

[94] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The Enterprise Ontology The Knowledge Engineering Review, 13 (Special Issue on Putting Ontologies to Use). 1998. 27

[95] Helen Walker. The virtual organisation: a new organisational form? *International Journal of Networking and Virtual Organisations*, 3(1):25–41, 2006. 26

[96] Gregory C. Walsh and Hong Ye. Scheduling of networked control systems. *IEEE Control Systems*, 21(1):57–65, 2001. 39

[97] Gregory C. Walsh, Hong Ye, and Linda G. Bushnell. Stability analysis of networked control systems. *IEEE transactions on control systems technology*, 10(3):438–446, 2002. 39

[98] Robert Winter and Ronny Fischer. Essential layers, artifacts, and dependencies of enterprise architecture. In *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*, pages 30–30. IEEE, 2006. 15

[99] Kenneth J. Witt, Jason Stanley, David Smithbauer, Dan Mandl, Vuong Ly, Al Underbrink, and Mike Metheny. Enabling Sensor Webs by utilizing SWAMO for autonomous operations. In *8th NASA Earth Science Technology Conference*, pages 263–270, 2008. 32, 41

[100] James P. Womack, Daniel T. Jones, Daniel Roos, and Massachusetts Institute of Technology. *The machine that changed the world: The story of lean production*. Harper Collins, 1991. 19

[101] Dazhong Wu, Matthew John Greer, David W. Rosen, and Dirk Schaefer. Cloud manufacturing: Strategic vision and state-of-the-art. *Journal of Manufacturing Systems*, 32(4):564–579, 2013. 19

[102] Di Wu, Xi-Ming Sun, Ying Tan, and Wei Wang. On designing event-triggered schemes for networked control systems subject to one-step packet dropout. *IEEE transactions on industrial informatics*, 12(3):902–910, 2016. 39

[103] Yuanqing Xia. From networked control systems to cloud control systems. In *Proceedings of the 31st Chinese control conference*, pages 5878–5883. IEEE, 2012. 3

[104] Yuanqing Xia. Cloud control systems. *IEEE/CAA Journal of Automatica Sinica*, 2(2):134–142, 2015. 3, 4, 5

[105] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018. 12

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[106] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and computer-integrated manufacturing*, 28(1):75–86, 2012. 19

[107] Yuan Yuan, Huanhuan Yuan, Zidong Wang, Lei Guo, and Hongjiu Yang. Optimal control for networked control systems with disturbances: a delta operator approach. *IET Control Theory & Applications*, 11(9):1325–1332, 2017. 39

[108] John A. Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292, 1987. 15

[109] John A. Zachman. A framework for information systems architecture. *IBM systems journal*, 38(2/3):454, 1999. 15

[110] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5):311–327, 2004. 44

[111] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl. Industry 4.0 – An Introduction in the phenomenon. *IFAC-PapersOnLine*, 49(25):8–12, 2016. 18

[112] Yufeng Zhan, Yuanqing Xia, and Athanasios V. Vasilakos. Future directions of networked control systems: A combination of cloud control and fog control approach. *Computer Networks*, 161:235–248, 2019. 3

[113] Lin Zhang, Yongliang Luo, Fei Tao, Bo Hu Li, Lei Ren, Xuesong Zhang, Hua Guo, Ying Cheng, Anrui Hu, and Yongkui Liu. Cloud manufacturing: a new manufacturing paradigm. *Enterprise Information Systems*, 8(2):167–187, 2014. ix, 20

[114] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips. Stability of networked control systems. *IEEE Control Systems*, 21(1):84–99, 2001. 39

[115] Xian-Ming Zhang and Qing-Long Han. Event-triggered H∞ control for a class of nonlinear networked control systems using novel integral inequalities. *International journal of robust and nonlinear control*, 27(4):679–700, 2017. 39

[116] Xian-Ming Zhang, Qing-Long Han, and Xinghuo Yu. Survey on recent advances in networked control systems. *IEEE Transactions on Industrial Informatics*, 12(5):1740–1752, 2016. 39

[117] Keliang Zhou, Taigang Liu, and Lifeng Zhou. Industry 4.0: Towards future industrial opportunities and challenges. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pages 2147–2152. IEEE, 2015. 12

[118] Xiaoyang Zhu. *Building a secure infrastructure for IoT systems in distributed environments*. PhD Thesis, Université de Lyon, 2019. 4

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

[119] Ing. Andrej Ŝtefánik, Ing. Milan Gregor, Ing. Radovan Furmann, and Ing. Peter Ŝkorík. Virtual Manufacturing in Research & Industry. *IFAC Proceedings Volumes*, 41(3):81–85, January 2008. 19

[120] Gašper Škulj, Rok Vrabič, Peter Butala, and Alojzij Sluga. Statistical Process Control as a Service: An Industrial Case Study. *Procedia CIRP*, 7:401–406, 2013. 40, 41

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés

UNIVERSITÉ DE LYON

INSA

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : LYU
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 03 July 2020

Prénoms : Minhu

TITRE :
Towards Control as a Service models and architecture for the Industry 4.0

NATURE : Doctorat

Numéro d'ordre : 2020LYSEI048

Ecole doctorale : ED 512 - INFOMATHS

Spécialité : Informatique

RESUME :

To fit the renewed globalized economic environment, enterprises, and mostly SMEs, have to develop new networked and collaborative strategies, focusing on networked value creation (instead of the classical value chain vision), fitting the blue ocean context for innovative products and service development. Even if collaborative organizations have been studied for decades, the closer connection of information systems involved by the so-called "Industry 4.0" developed by leading industries in Europe, US and Asia requires to set new IT models to support agile and evolving collaborative Business Process (BP) enactment, integrating both traditional Information Systems (IS) and production control processes. By now, these product/service ecosystems are mostly supported by software services, which span multiple organizations and providers, and on multiple cloud-based execution environments, increasing the call for openness, agility, interoperability and trust for both production and Information System organization. These requirements are well supported by SOA, Web 2.0 and XaaS technologies for Information Systems. Taking advantage of IoT, services and Cloud technologies, the development of Cloud of Things (CoT) changes the way control application are engineered and developed moving from a dedicated design and development of control applications to a Control as a Service vision. This vision requires developing a new architecture to connect physical and logical objects as well as integrating basic control patterns to organize a consistent control service orchestration. To fit this challenge, we propose a multi-layer Control as a Service architecture to describe control systems in a holistic way. Our Control service model is built according to an event-driven orchestration strategy. Thanks to the integration of a context manager, analyzing continuously the system environment as well as the control system behavior, these context-aware control services can be deployed.

MOTS-CLÉS : Control as a Service, Service model, Industry 4.0, control ontology, event driven orchestration

Laboratoire (s) de recherche : LIRIS – UMR

Directeur de thèse: Pr. Frédérique Biennier et Pr. Parisa Ghodous-Shariat-Torbaghan

Président de jury : Pr Christine Verdier

Composition du jury : Pr. Chirstine Verdier, Pr. Ernesto Exposito, Pr. Bernard Grabot, Pr. Frédérique Biennier, Pr. Parisa Ghodous-Shariat-Torbaghan

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2020LYSEI048/these.pdf
© [M. LYU], [2020], INSA Lyon, tous droits réservés