Software Framework for the Development of Context-Aware Reconfigurable Systems

Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der Naturwissenschaftlich-Technischen Fakultät der Universität des Saarlandes
&
Tunisia Polytechnic School
University of Carthage

von
Soumoud FKAIER

Saarbrücken
2021.

Abstract:

English: In this project we propose a new software framework for the development of context-aware and secure controlling software of distributed reconfigurable systems. Context-awareness is a key feature allowing the adaptation of systems behaviour according to the changing environment. We introduce a new definition of the term “context” for reconfigurable systems then we define a new context modelling and reasoning approach. Afterwards, we define a meta-model of context-aware reconfigurable applications that paves the way to the proposed framework. The proposed framework has a three-layer architecture: reconfiguration, context control, and services layer, where each layer has its well-defined role. We define also a new secure conversation protocol between distributed trustless parts based on the blockchain technology as well as the elliptic curve cryptography. To get better correctness and deployment guarantees of applications models in early development stages, we propose a new UML profile called GR-UML to add new semantics allowing the modelling of probabilistic scenarios running under memory and energy constraints, then we propose a methodology using transformations between the GR-UML, the GR-TNCES Petri nets formalism, and the IEC 61499 function blocks. A software tool implementing the methodology concepts is developed. To show the suitability of the mentioned contributions two case studies (baggage handling system and microgrids) are considered.

Deutsch: In diesem Projekt schlagen wir ein Framework für die Entwicklung von kontextbewussten, sicheren Anwendungen von verteilten rekonfigurierbaren Systemen vor. Kontextbewusstheit ist eine Schlüsseleigenschaft, die die Anpassung des Systemverhaltens an die sich ändernde Umgebung ermöglicht. Wir führen eine Definition des Begriffs “Kontext” für rekonfigurierbare Systeme ein und definieren dann einen Kontextmodellierungs- und Reasoning-Ansatz. Danach definieren wir ein Metamodell für kontextbewusste rekonfigurierbare Anwendungen, das den Weg zum vorgeschlagenen Framework ebnet. Das Framework hat eine dreischichtige Architektur: Rekonfigurations-, Kontextkontroll- und Dienste-Schicht, wobei jede Schicht ihre wohldefinierte Rolle hat. Wir definieren auch ein sicheres Konversationsprotokoll zwischen verteilten Teilen, das auf der Blockchain-Technologie sowie der elliptischen Kurven-Kryptographie basiert. Um bessere Korrektheits- und Einsatzgarantien für Anwendungsmodelle zu erhalten, schlagen wir ein UML-Profil namens GR-UML vor, um Semantik umzufassen, die die Modellierung probabilistischer Szenarien unter Speicher- und Energiebeschränkungen ermöglicht. Dann schlagen wir eine Methodik vor, die Transformationen zwischen GR-UML, dem GR-TNCES-Petrinetz-Formalismus und den IEC 61499-Funktionsblöcken verwendet. Es wird ein Software entwickelt, das die Konzepte der Methodik implementiert. Um die Eignung der genannten Beiträge zu zeigen, werden zwei Fallstudien betrachtet.

Français: Dans ce projet, nous proposons un framework pour le développement d'applications sécurisés et contextuelles des systèmes reconfigurables distribuées. La conscience au contexte est un élément clé permettant l'adaptation du comportement des systèmes par rapport à leurs environnement. Une définition du terme "contexte" pour ce type de systèmes est introduite ainsi qu'une approche de modélisation et de raisonnement du contexte. Après, un méta-modèle d'applications reconfigurables et conscientes du contexte est défini. Le cadre proposé a une architecture à trois niveaux: reconfiguration, contrôle du contexte et services, où chaque couche a son rôle bien défini. Nous définissons également un protocole de conversation sécurisée entre les parties distribuées basées sur la technologie de blockchain et sur la cryptographie à courbe elliptique. Pour obtenir de meilleures garanties en termes de correction et de déploiement des modèles d'applications, un profil UML appelé GR-UML est introduit pour ajouter les sémantiques permettant la modélisation de scénarios probabilistes fonctionnant sous contraintes de mémoire et d'énergie, puis une méthodologie utilisant des transformations entre le GR-UML, le formalisme des réseaux de Petri GR-TNCES et les blocs fonctionnels de IEC 61499 est introduite. Un outil logiciel mettant en œuvre les concepts de la méthodologie est développé. Pour montrer la pertinence des contributions mentionnées, deux études de cas sont considérés.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 General Context and Motivation

Reconfigurable automation control systems are core elements of the contemporary world [Liu *et al.* 2017]. Smart transportation systems, smart factories, autonomous driving, smart health-care systems, smart electricity grids are some examples of the present-day systems where reconfiguration is a fundamental enabler for their efficient operation. Particularly, the controlling software applications are playing a key role in developing their functionalities. Continuous attempts are carried out in order to apply advances of the software concepts in the field of reconfigurable systems, however many challenges are still ahead related to the development and engineering of the software. Figure 1.1 depicts a generic schematic view of a reconfigurable automated control system of three examples of systems (smart microgrids, smart factories, and smart baggage system).

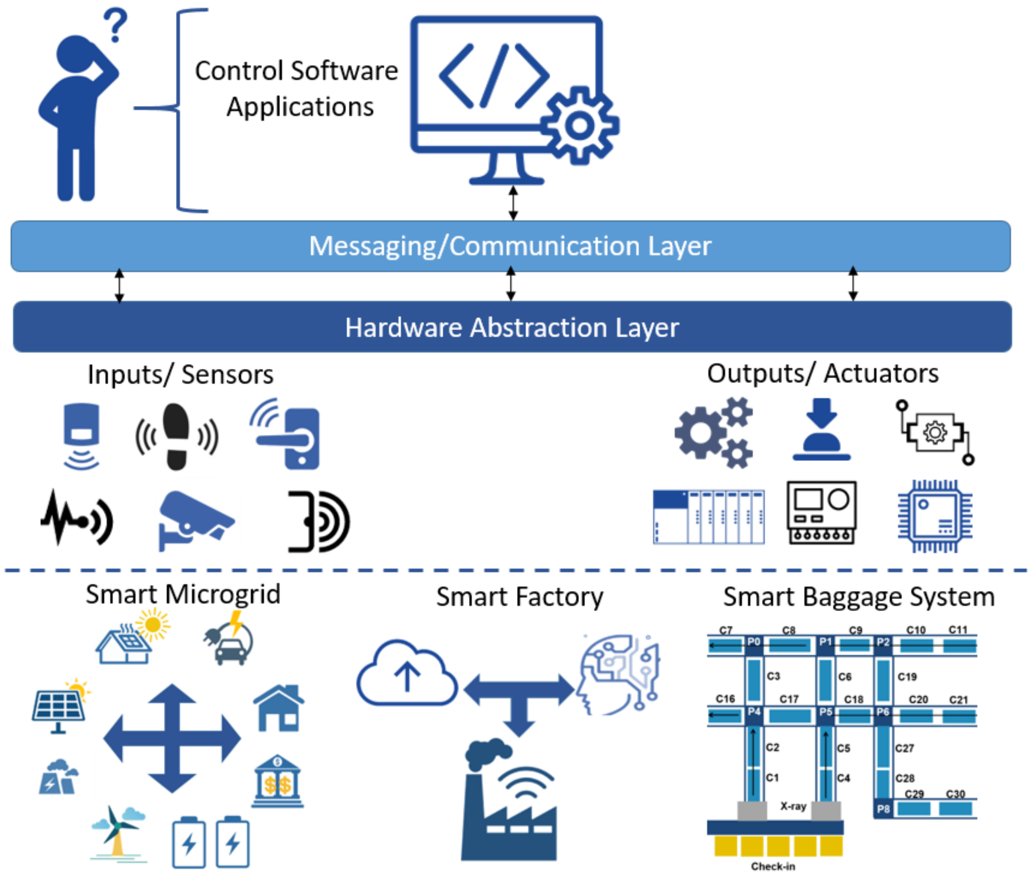


Figure 1.1: Reconfigurable automation and control systems presentation.

One important research focus is how to provide self-adaptation ability to applications/systems. In previous works adaptation has been frequently associated to reconfiguration and context-awareness [Mahalle & Dhotre 2020], [Krupitzer *et al.* 2018], [Weyns 2019]. In fact, the integration of ICTs made a huge amount of data available in a real-time and accurate way, which triggers the idea of exploiting it in the software logic.

The possibility to boost systems with awareness about the context in which they are working may considerably improve their behavior. Hence, there is a crucial need to a software infrastructure which enables the development of the complex increasing requirements all with promoting the handling of context knowledge [Iqbal *et al.* 2018], [Bucchiarone *et al.* 2017], [Thramboulidis *et al.* 2017].

The growing complexity of modern systems requirements has also made the modeling task of the main logic harder, especially when it comes to guaranteeing some constraints under which the system should operate. It is necessary to find a methodology enabling an efficient modeling through a practical process and reliable results. UML has been widely considered for the modeling of software applications due to the expressivity it provides through the various types of diagrams. However, it does not provide a way to get sureness about models correctness. For this, many works consider the formal verification for additional guarantees [Smida *et al.* 2019]. Formal verification allows to apply mathematical concepts for models analysis in a way to recognize model ambiguities and errors. These approaches (UML and formal verification) are very fruitful for all software types, but despite this importance, other modeling languages/tools are often used for the automation and control systems such as the IEC 61499 standard [Yang *et al.* 2019a]. Using IEC 61499 function blocks to model distributed applications provides the way to clarify the structure and the elementary functions of a system/application. Also it has a great asset enabling the mapping of function block models into implementations on some hardware platforms. However, it is hard to start the modeling of big and complicated applications from the function block level despite the compositional and hierarchical encapsulation ability. Therefore a modeling research challenge can be formulated in the following question: is it possible to define a method that takes profit from the advantages of all the mentioned modeling techniques (UML, formal verification, and IEC 61499 function blocks) but through a lightweight process?

Information security is being always an inherent challenge to all computerized systems, especially when it comes to distributed parts of a system, in fact, protecting exchanged data against falsification is an important condition to guarantee systems availability and reliability. Many of the existing security mechanisms rely on the concept of a trust third party which plays the role of a legitimate authority that verifies transactions security. However, the legitimacy and/or trustworthiness of the authority part is itself questionable since this part can manipulate private data of managed users. Blockchain technology [Nakamoto 2019] was introduced to cope with this limitation through making the security build upon a consensus paradigm. It is considered as a distributed ledger containing all transactions verified by all (or the majority) the network nodes [Salman *et al.* 2018]. It provides transparency and immutability, which makes it very suitable to distributed trustless systems. However, its adoption it not free from flaws. In fact, blockchain has a scalability challenge since all data are registered and never deleted, therefore its size is continuously growing. In addition, it resolves the trust and transparency issues but violates the privacy since all participants should be publicly identified. Hence, to make its use suitable we need to answer the following research question: how can we use the blockchain technology with preserving peers privacy? Is there a way to improve ledgers scalability? In the following section, the recognized problems and challenges are presented in detail.

1.2 Problems and Challenges

Researchers have spent an outstanding effort in proposing system modeling approaches, software architectures, and software infrastructures. However, developing a controlling software that takes into account the emerging technologies and the newly arising requirements is still challenging. Complexity of the requirements is the main problem that underlie these challenges. Hence, abstract and generic software support is required to facilitate the development of the applications needs, especially that a considerable number of functionalities are repetitive and can be reusable for specific applications. It is efficient for developers to avoid re-performing the specification for every case (there is a need to avoid re-inventing the wheel) [Chen 2017].

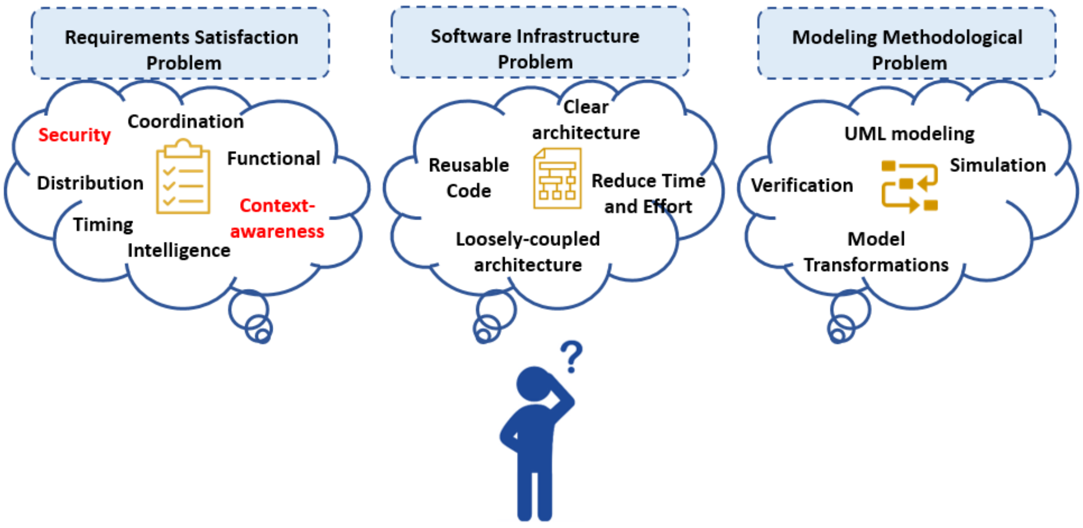


Figure 1.2: Reconfigurable applications development challenges.

Three types of challenges are recognized in relation with the software development process (see Figure 1.2):

1. Challenges related to the satisfaction of applications requirements,
2. Challenges related to the software infrastructure,
3. Challenges related to the methodological process of efficient applications modeling.

1.2.1 Applications Requirements Challenges

Satisfying requirements and needs of applications must be the core objective of any application development. This is why an ultimate attention should be paid on satisfying requirements, especially those promoting smartness. For this, we identified a set of common requirements that are particularly suited to ensure intelligence features.

- **Need to Efficiently Get Contextual Knowledge.** One of the key features that promote system's smartness is context-awareness. In fact, from the context-awareness computing paradigm, a context-aware system is the one specified by the ability to adjust its behavior in accordance with the changing situations related to the system goals and/or to the environment. Many concepts are defined to deal with leveraging context information in proper way. However, given the intricacy, size, and nature of the considered systems, open questions regarding the extent of efficiency are still open: given the huge information sensed from input devices, and given that the same parameters could be present in different contexts, how to make decision about which context is the more proper? In case of faults occurrences, is it possible to lead the proper adaptation/reconfiguration without human intervention? And how much the sensed data can be trustworthy? Hence, defining proper context modeling and reasoning mechanisms are needed.

- **Need to Ensure Coordination (Distributed Systems).** Distribution is the latest engineering paradigm promoting the flexibility of reconfigurable systems. In order to lead an adaptation process, peers need to coordinate with each other to achieve a global system coherence. After any coordination the system should maintain a stable state. Consistent interactivity between the mentioned peers is crucial in fulfilling such a need, but how to provide an efficient mechanism facilitating the coordination of the distributed components?

- **Need to Provide Security.** As it is the case for any computerized system, software applications of reconfigurable systems can be vulnerable to cyber-attacks. The distribution of the system components/peers implies the use of networking and communication technologies. These technologies have a set of inherent vulnerabilities that might be exploited in malicious way. Vulnerabilities could threaten authentication (verifying the identity of a user), privacy (the ability to exchange data secretly), data tampering (modifying some credentials, measurements, etc.), or many other attacks that depends on the system. Therefore, having the ability to include security mechanisms in some functionalities is required to build trustworthy applications. The software should enable the use of standard or personalized security technologies especially the emerging ones.

- **Need to Leverage Artificial Intelligence Mechanisms.** Artificial intelligence (AI) refers to the computer science branch dealing with making machines learn from human expertise in order to adapt itself to the changing inputs [Nassar *et al.* 2020]. Applications of AI prove its efficiency in many domains such as autonomous car driving, military, health-care, etc. A variety of AI tools and approaches are introduced to facilitate the use of the mentioned concepts. Hence, taking profit from these techniques in the development of better adaptation processes of future reconfigurable systems would help in minimizing the difficulty of setting up software processes and provide better results, especially it would help in building prediction tasks and problems anticipation.

- **Need to Handle Unpredictable Reconfigurations (functional).** Reconfiguration is a key feature of adaptive systems [Khalgui *et al.* 2019]. It is defined as the ability to switch from one system state to another without human intervention and without putting the system off. These reconfigurations could be originated from the system owners that change operational goals or from system electrical/mechanical failures. The latter are critical since they often happen unpredictably. Therefore, there is a need to create a mechanism guaranteeing safe reconfiguration executions, i.e., reconfiguring the application without functional conflicts that might relate to sharing resources, dependencies, or priorities.

- **Need to Satisfy Real-Time Constraints.** Future digital smart systems are not only required to be smart and intelligent, but also accurate. Applications need to process data and return results within a determined time. Response time should depend on the processed task, i.e., whether it is event driven or it is a regular task, as well as on its priority and capacity. Some of the events or regular tasks need to be processed before specific deadlines, for some cases if tasks deadlines are exceeded, the system could fail. These failures induce damages that could end with economic losses. Therefore, introducing real-time analysis techniques [Ghribi *et al.* 2018] is also required.

Given the above-mentioned analysis of requirements, we rise the following research questions: What is context and context-awareness for such systems? And how to allow efficient context reasoning? How to enable coordination between distributed parts in efficient secure way? What security technologies to adopt for such distributed systems? How to ensure intelligence, functional, and timing efficiency?

1.2.2 Software Infrastructure Challenges

Developing applications of context-aware reconfigurable systems that have many different requirements -as detailed in the previous paragraphs- is challenging. The most important challenges are presented in the following.

- **Need to Provide an Infrastructure Enabling Reusability.** A software infrastructure that provides reusable generic, tested, structured modules may considerably promote the productivity of developers, by making them focus on the unique requirements of the considered case in place of spending time and effort on repetitive concerns related to the infrastructure.

- **Need to Provide Multidisciplinary Aspects (Holistic).** Each single discipline, used in the software logic, presents many attractive concepts and original philosophy behind it. However, in real world the application of these concepts must complete each other and/or work together in harmony in order to provide expected results. For example, using sensed/collected information from outside is beneficial as far as secure and trusted communication channels are set up. Otherwise, data might be falsified therefore very harmful to the application logic. It is required to build a software that takes into account the different disciplines as mentioned in Figure 1.3.

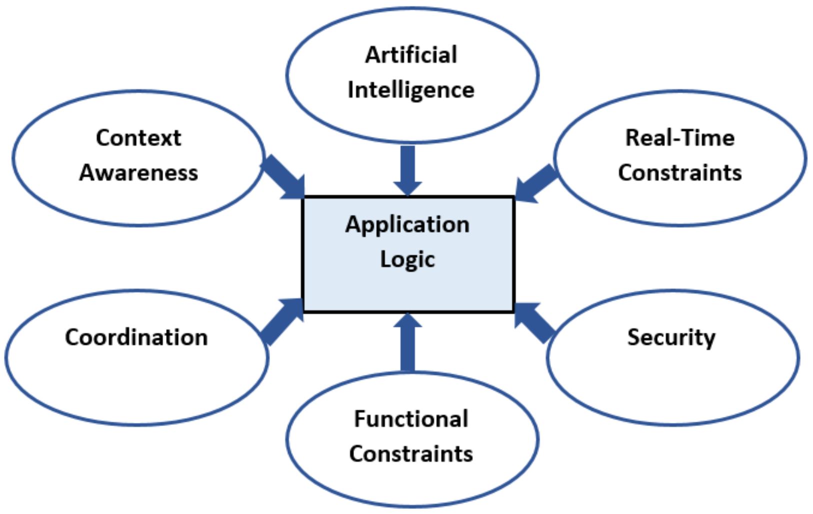


Figure 1.3: Multidisciplinary application's logic.

- **Need to Provide Clear and Understandable Software Architecture (Abstract and Generic).** Given the numerous requirements and the involvement of different disciplines into the software logic, defining a clear and understandable architecture becomes a necessity. The application process should be tailored into elements responsible for well-defined tasks. In addition, abstraction layers should be introduced to distinguish the separate roles. So the question that arise here is how to provide functionalities in a loosely couple way?

Given the aforementioned analysis of software infrastructure, the hereafter research questions arise: How to define a software infrastructure that guarantees multidisciplinarity but at the same time clarity and simplicity?

1.2.3 Modeling Methodological Challenges

Over the years, many software modeling methodologies have been proposed to satisfy development needs [Grichi *et al.* 2017], [Meskina *et al.* 2018], [Oueslati *et al.* 2018], [Ghribi *et al.* 2018]. However, for complicated systems, more suitable methodologies are still required. In fact, for such systems it is needed to emphasize some features over others, i.e., some constraints are more important over others. Thus, developers should pay more attention to these constraints. Complexity of applications urges the necessity to test and validate models during early stages of developments, otherwise ambiguities and/or error may induce difficult/expensive corrections if discovered lately. Also, different application views need to be provided. The definition of an efficient methodology is constrained with a set of challenges classified as follows:

- **Need to Provide the Ability to Model Applications from Structural and Behavioral Views.** Application design is a very important phase of any application development, therefore accomplishing it in a proper way is primarily needed. A good modeling phase is the one ending with clear and understandable structural and behavioral views of the application. Modeling the dynamics of applications helps in assessing the application logic. The conventional (but not formal) modeling language UML represents a good solution for developers to perform modeling. It provides the possibility to design different system perspectives such as the user, implementation, environmental, structural, and behavioral perspectives. The structural view offers the class diagram as a powerful tool frequently used in most of the projects, and statechart as well as activity diagrams to model the behavioral view. However, UML does not provide the semantics to specify probabilistic and resources-constrained features. Further, for the behavioral view modeling of complicated systems, using UML diagrams is questionable as it is not formal. More guarantees about the system behavior correctness are required.

- **Need to Provide the Ability to Perform Formal Verification.** Having guarantees about applications correct behavior during early stages of development is necessary to reduce faulty behaviors and to improve applications reliability. Therefore, there is a need to a process that allows the exhaustive testing of all applications behaviors and the exploration of all states.

- **Need to Provide the Ability to Test Deployment in Specific Hardware Environment.** Before real-world implementation, models need to be tested. Simulations are traditionally used to emulate the real-world systems. However, many problems could be faced. In fact, simulations may not give complete information about the developed applications since tests will depend on the prototype hardware specifications and the chosen scenario. Hence, there is a need to a method enabling the deployment testing in a target hardware environment and in efficient way.

- **Need to Provide Modification-Friendly Process.** Whenever multiple modeling techniques are adopted, designers may need to refine and modify models if pitfalls are catched or new features are added. This modification should not imply the restart of the modeling process, rather there should be a way allowing the seamless modification. Therefore, providing a process flow allowing iterative and cumulative modifications is also required. More importantly, there is a need to a software tool that supports the move from one modeling scope to another.

Based on the analysis above-mentioned, the following research questions arise: Is it possible to define a method that takes into consideration all the mentioned techniques? If yes, how to tackle the different model testing/analysis needs in one systematic process? What makes a considered process more efficient and easy?

1.3 Thesis Contributions

In response to the research questions raised by the analysis conducted in the previous section, the current thesis contributes with a set of solutions to improve the software engineering approaches of context-aware reconfigurable systems. The contributions of this research project are summarized in Figure 1.4.

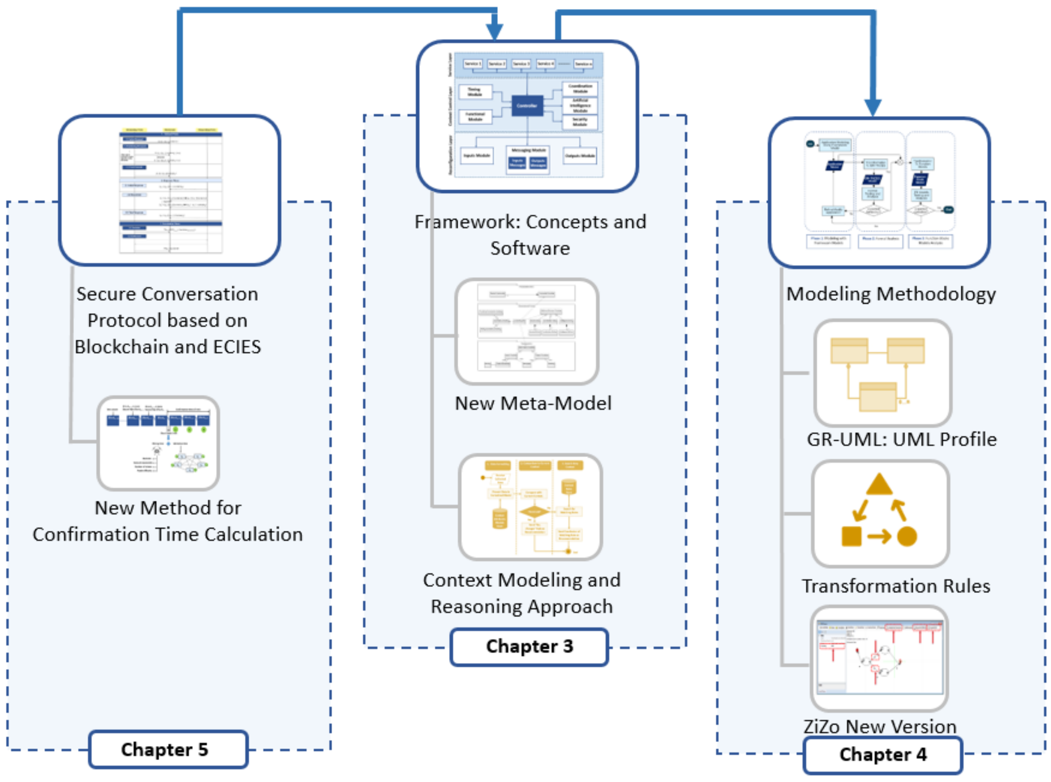

Secure Conversation Protocol based on Blockchain and ECIES
New Method for Confirmation Time Calculation
Chapter 5
Framework: Concepts and Software
New Meta-Model
Context Modeling and Reasoning Approach
Chapter 3
Modeling Methodology
GR-UML: UML Profile
Transformation Rules
ZiZo New Version
Chapter 4

Figure 1.4: Overview of the thesis contributions.

Advanced concepts are used together to introduce: a new software framework, a new context reasoning approach, a new secure conversation protocol, a new UML profile, a new modeling methodology, and new software supports. These contributions aim to make software applications of reconfigurable systems smarter. The following sections resume the contributions.

1.3.1 Software Framework for Smart Applications.

In order to overcome the challenges related to the applications requirements as well as the software infrastructure problems showed in Section 1.2.1 and Section 1.2.2, a software framework is introduced. The framework has a three-layer architecture: (i) Reconfiguration Layer: is the first layer playing the role of an interface between the framework core and the environment that the framework behaves within, (ii) Context Control Layer: is the central layer containing the main logic, and (iii) Service Layer: is the third layer and it contains the services provided by the system. Each layer is composed of a set of modules, where each module provides a well-determined functionality.

The proposed framework introduces a set of original contributions presented in the following.

- **Provide Multidisciplinary Development Model.** The proposed framework allows to implement a plethora of requirements that could be needed to develop future smart systems:

  Context Awareness

  The proposed framework pays a great care to context-awareness paradigm as it is considered a key enabler of dynamic interaction with the surrounding. The framework, through its Reconfiguration Layer, allows to pick-up context information, performs reasoning on it, finds recommendations, and feeds it to the second layer.

  Include Security Techniques

  In order to ensure systems reliability, information security mechanisms are included as one of the features provided by the framework. In particular, a blockchain based protocol is proposed as one example of the security tools.

  Coordination in Distributed Systems

  The proposed framework introduces a coordination mechanism allowing the distributed peers to collaborate together to establish system global operation coherence.

  Include Artificial Intelligence Mechanisms

  The proposed framework introduces a way of leveraging artificial intelligence concepts to allow some optimization operations. In particular, it provides the way to develop prediction tasks.

  Functional Constraints Satisfaction

In order to guarantee coherent operation of system services, the proposed framework defines a mechanism to check service's functional safety such as priorities and dependencies.

Real-time Constraints Respect

Providing the expected results at the right time and before exceeding their deadlines is an important task of the software application. This is why, the proposed framework includes a timing analysis mechanism to allow analyzing tasks schedulability and time-efficiency.

- **Complexity Taming through Clear Architecture.** The layered-architecture of the framework allows to build a clear understanding of applications logic. The separation of concerns principle is adopted in defining the different functional layers of the architecture. Also, it allows to define the necessary modules composing each layer. Through separating the interrelated tasks into different layers and modules, it becomes possible to tame the complexity of the applications development. The framework presents its different elements (i.e., layers and modules) in a loosely coupled way in order to make applications more testable.
- **Introduce a Novel Context Modeling and Reasoning Approach.** The proposed framework introduces a new efficient context reasoning mechanism allowing better intelligence and better resources usage. The reasoning process builds upon an ontology-based context modeling and it consists in a hybrid reasoning using different checking perspectives to conclude context information.
- **Compliance with Emerging Technologies: Blockchain.** As blockchain becomes more and more integrated in modern systems, and given its multiple assets, the proposed framework includes the possibility to use its concepts and techniques as security enablers. The blockchain is used as a support for secure transactions between trustless peers composing a distributed system.

1.3.2 Modeling Methodology using the Framework.

A new modeling methodology of context-aware reconfigurable applications of distributed systems is introduced. A new UML profile called the Generalized Reconfigurable UML (GR-UML) is introduced to enrich existing semantics with probabilistic and resources control semantics. The methodology consists of three phases: (1) modeling of applications using the models of the proposed framework as well as the new GR-UML concepts, (2) performing formal verification on the models obtained from the first phase, and (3) performing deployment testing on a target hardware environment according to the distributed systems standard IEC 61499. A set of transformation rules is defined to allow an automatic transformation from application's models to GR-TNCES and to IEC 61499 function blocks. A new software tool that supports the said transformations is implemented.

1.3.3 Secure Conversation Protocol of a Network of Frameworks

A secure conversation protocol among distributed peers having software applications developed using the proposed framework is introduced. The security of the protocol builds upon the blockchain technology as well as the elliptic curve cryptography. Transparency, reliability, and cost-efficiency are guaranteed. The protocol resolves the problem of identities revealing implicated by the blockchain technology through identifying transactions rather than identifying its source. A key pair of the Elliptic Curve Integrated Encryption Scheme (ECIES) [Gayoso Martínez *et al.* 2010] is newly created for every new transaction where the public key is used as a transaction identifier. Scalability has been an inherent problem to blockchain since its creation since all registered transactions are never deleted, so the size of the ledger keeps growing. To cope with the scalability problems, the phases and steps of the protocol are defined in a sort that reduces the number and size of the information to be stored in the blocks, hence to improve the scalability.

1.3.4 Software Tool Implementing the Framework Concepts.

The proposed framework concepts are implemented in a new software tool which allows to develop applications of context-aware reconfigurable systems.

1.4 Publications

The results of the thesis are published in the following publications.

1. Fkaier, Soumoud, Mohamed Khalgui, and Georg Frey. "Modeling Methodology for Reconfigurable Distributed Systems using Transformations from GR-UML to GR-TNCES and IEC 61499." In ENASE, 2021. DOI: 10.5220/0010422102210230. In Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021), pages 221-230. ISBN: 978-989-758-508-1. **Published, Class B.**

2. Fkaier, Soumoud, Mohamed Khalgui, and Georg Frey. "A Software Framework for Context-aware Secure Intelligent Applications of Distributed Systems." In Proceedings of the 16th International Conference on Software Technologies - ICSOFT 2021, ISBN 978-989-758-523-4; ISSN 2184-2833, pages 111-121. DOI: 10.5220/0010604701110121." **Published, Class B.**

3. Fkaier, Soumoud, Mohamed Khalgui, and Georg Frey. "Hybrid Context-awareness Modelling and Reasoning Approach for Microgrid's Intelligent Control." In Proceedings of the 15th International Conference on Software Technologies - ICSOFT 2020, ISBN 978-989-758-443-5; ISSN 2184-2833, pages 116-127. DOI: 10.5220/0009780901160127 **Published, Class B.**

4. Fkaier, Soumoud, Mohamed Romdhani, Mohamed Khalgui, and Georg Frey. "Context-awareness Meta-model for Reconfigurable Control Systems." In Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2017, ISBN 978-989-758-250-9; ISSN 2184-4895, pages 226-234. DOI: 10.5220/0006328502260234. **Published, Class B.**

5. Fkaier, Soumoud, Mohamed Romdhani, Mohamed Khalgui, and Georg Frey. "R2TCA: New tool for developing reconfigurable real-time context-aware framework—Application to baggage handling systems." In Proc. Int. Conf. Mobile Ubiquitous Comput., Syst., Services Technol.(UBICOMM), pp. 113-119. Venice, Italy, 2016. **Published, Selected Paper, Class C.** (https://www.iaria.org/conferences2016/AwardsUBICOMM16.html)

6. Fkaier, Soumoud, Mohamed Romdhani, Mohamed Khalgui, and Georg Frey. "Enabling reconfiguration of adaptive control systems using real-time context-aware framework." In 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pp. 1-8. IEEE, Agadir, Morocco,2016. **Published, Class C.**

7. Fkaier, Soumoud, Mohamed Khalgui, and Georg Frey. "Meta-Model for Control Applications of Microgrids." In ENERGYCON, 2020. doi:10.1109/energycon48941.2020.9236582. **Published. Indexed by IEEE Xplore, Scopus, and SJR platform.**

1.5 Thesis Plan

This research project aims to provide novel solutions to enhance the development of smart software applications for distributed reconfigurable systems. The structure of the dissertation aligns with the main contributions of the thesis through the third, fourth, and fifth chapters. The outline of the manuscript is given as follows:

In Chapter 1, which is the current chapter, the thesis is placed into its axes, where the need to novel concepts, tools and methods in the development of smart software of distributed reconfigurable systems is analyzed. The analysis has led to identify a set of related problems and challenges classified into three types: problems related to the satisfaction of the multitude of requirements especially those related to secure coordination and context-awareness, problems related to the software infrastructure that may facilitate the development of applications, and problems related to modeling methodologies that allows to explore structural and behavioral views as well as to test its correctness and deployability.

Chapter 2 presents a survey of the context-awareness concepts and tools as well as the related context-aware computing concepts. This chapter also presents the areas that the work deals with such as the Petri nets classes (more precisely the GR-TNCES formalism), the standard of distributed systems IEC 61499, and researches in the field of model transformation. In addition, this chapter provides an overview of the blockchain technology and points out the challenges associated to its use.

In response to the problem related to the software infrastructure, Chapter 3 introduces the proposed framework. Chapter 3 presents also the design principals, the meta-model, and the architecture of the framework. A formal definition of its components is provided as well as UML models of each component are presented. Chapter 3 introduces also a novel context reasoning mechanism that builds upon a new definition of the term “context” for reconfigurable systems. The contributions introduced in this chapter are applied to a running example of a software of an airport baggage handling system.

Chapter 4 introduces at first a new UML profile, the GR-UML, then a three-phased modeling methodology which draws upon the framework models and GR-UML. The definition of the phases is presented as well as the definition of the transformation rules of the models issued from the first phase to GR-TNCES Petri nets extension and to IEC 61499 function blocks. An example of use of the defined methodology is shown using the same example of a baggage handling conveyors application.

Chapter 5 defines a secure conversation protocol among distributed and trustless peers having a controlling software developed with the proposed framework. The security of the protocol draws upon the blockchain technology as well as the difficulty of the elliptic curve cryptography. The phases and steps of the protocol are introduced in a way to optimize the data size of exchanged transactions with the aim of improving the scalability in terms of storage burden. The protocol is applied to a case of electricity trading among distributed microgrids.

Chapter 6 concludes the dissertation and shows its contributions and outputs. Perspectives of the improvement of the framework as well as future software development approaches are proposed.

Chapter 2

State of the Art

Contents

2.1 Introduction

This chapter starts with an analysis of the existing software development approaches of smart/adaptive reconfigurable systems then discusses its limitations. After that it presents overviews of the topics to be tackled to propose new solutions in the current thesis.

In Section 2.3, context-awareness computing paradigm is surveyed through presenting the existing context definitions, context life-cycle, and context aware software tools, where a thorough focus is made on surveying software frameworks dedicated to the development of context-aware systems.

Based on the identified problem concerning the methodological process of efficient modeling phase, Section 2.4, Section 2.5, as well as Section 2.6 present in nutshells the topics to be involved in the proposed modeling methodology, which are the formal verification, the distributed industrial automation systems standard IEC 61499, and the model transformation of UML models to the two latter.

Section 2.7 briefly presents the foundations of the blockchain technology and its great potential to ensure security to distributed systems. An analysis of its limitations is then carried out.

Finally, Section 2.8 recapitulates the outcomes of this chapter and points out the limitations of the reviewed topics.

2.2 Motivation

Any control automation system strives to ensure more intelligent behavior to improve users satisfaction. However, given the increasing complexity originated from distribution matters, data sources heterogeneity, and requirements variety, the development of the software for such systems becomes really a difficult task. This is why, over the years many concepts and paradigms have been introduced to promote the integration of knowledge-based and intelligence techniques to the software of reconfigurable systems. For instance, the researches reported in [Schneider *et al.* 2017], [Schneider *et al.* 2019], and [Ekaputra *et al.* 2017] have proposed ontology and knowledge-based solutions for the operation of control logic of automation systems. The works reported in [Krupitzer *et al.* 2018] and [Weyns 2019] have surveyed existing solutions where intelligence and adaptation are driven by approaches such as architectures, run-time models, goals, and guarantees under uncertainties. Among possible solutions, context-awareness computing represents an important alternative enabling smart behaviors. In fact, thanks to the advances of embedded devices as well as the highly powerful networking and communication technologies, smarter operations and better services are expected to be provided. Ubiquitous input devices such as sensors and measurement units, have helped to afford huge amounts of data in a continuous way about the changing environment and conditions of their operation. **But how to take profit of this paradigm and apply its concepts for the reconfigurable systems? What is a context in such type of systems? How can we efficiently use contex-**

tual information in the software logic? Which software infrastructure to use in order to be able to implement systems requirements and integrate context awareness? From another side, cyber security is considered a major concern for computerized and distributed systems. Due to the complexity and scale of such type of systems, they are generally deployed in reconfigurable distributed modules, which makes communication at the heart of most of the activities. **But which security techniques to use for distributed reconfigurable systems? How can we set secure coordination among distributed peers all with guaranteeing exchanged data integrity and peers anonymity?** To provide robust software applications, the underlying models should be appropriate and clear enough to ensure successful code implementation, especially that reconfigurable systems are becoming day-by-day more complex. UML modeling provides the way to figure out different views, but despite this advantage additional analysis concerning the mapping to hardware devices and the correct behaviors are required to pick up errors and ambiguities in early stages. **But what modeling techniques could be used to get more guarantees about models efficiency? How to proceed to get reliable models?**

In order to answer the raised questions, we need first to look at existing works.

2.3 Context-Awareness Computing

This section provides an overview of the context-awareness computing and highlights the related topics to the current thesis.

2.3.1 Context-Awareness Definition

A huge amount of data related to the changing environment is made available thanks to the growing development of ubiquitous computing technologies such as advanced sensing and networking tools. This availability has created the motivation to exploit data in applications logic in order to provide the users with more accurate and proper functionalities that fit the current context. Systems that have the ability to integrate such ambient information in the operational behavior of the software applications are known as context-aware systems [Mahalle & Dhotre 2020]. To understand how can adaptive systems include the context-awareness paradigm, two paramount basics need to be defined: context and categories of context-awareness [Li *et al.* 2015] [Alegre *et al.* 2016].

2.3.1.1 Context Definition

In 1994, Bill N. Schilit has introduced the term "context-aware computing" to describe distributed mobile software applications that can react to the changing context of a person [Schilit *et al.* 1994]. Context was determined through answering three questions: *"where are you? who you are with?, and what resources are nearby?"*[Schilit *et al.* 1994]. Hence, the context covers different information such as the location, the time of the day, the communication bandwidth, the position of the surrounding persons, etc.

Over time, the context-aware paradigm was integrated to various systems, which made the context definition no longer fitting to all systems, environments, and targets. This is why, researchers have introduced different definitions of "context" given their current context, i.e., considered project characteristics and environments. Brown *et. al.* in the work reported in [Brown *et al.* 1997], define the context as *"location, identities of the people around the user, the time of day, season, temperature and so forth."*. However, Franklin *et. al.*, [Franklin & Flaschbart 1998] simply define it as *"what is happening at this moment"*.

It was difficult to provide an accurate, generic, and fit-to-all definition to the context concept. For this reason, in 1999, Dey and Abowd reviewed the existing attempts to define context in their research presented in [Abowd *et al.* 1999] and introduce the following definition *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."* This definition has been well received by many researchers and has been adopted in many works. But despite this acknowledgment, it remains imperfect. Therefore, researchers have continued to suggest other definitions in an effort to either improve this definition or to provide definitions appropriate for system's particularities [Perera *et al.* 2013].

Most of the definitions consider the user as a central element of context-aware applications. But does this apply to reconfigurable automated control systems that are usually based on interaction between devices and where the user's role is minimal?

2.3.1.2 Context Categories

To facilitate the handling of context information, researchers resorted to classifying these information according to the nature of systems. Based on some specific perspectives, different categories are defined. For instance in some systems, context is categorized based on surveillance during run-time into static and dynamic. For other systems it is classified into physical and virtual, where physical refers to physical objects such as nearby objects and persons, and virtual refers to the inferred and learned information. It is also categorized into direct and indirect, into primary and secondary, into internal and external, etc. One other different categorization approach is the one adopting the distinction of context through answering a set of questions such as who?, when?, where?, what?, and why?.

Authors of [Perera *et al.* 2013] surveyed the categorization schemes and summarize it into two main schemes, conceptual and operational. Xin *et. al.* also surveyed context taxonomies in the work reported in [Li *et al.* 2015]. Both researches end with a conclusion that there is no perfect approach in the absolute, but depending on the context one can be preferred over others.

2.3.1.3 Context-Awareness Applications

At the beginning of its appearance, the context-aware notions were applied to mobile applications that aim to provide users with more intelligent applications [Luo *et al.* 2020]. For instance, In [Ortiz *et al.* 2019] mobile context-aware applications architecture are presented and an analysis of its resources consumption is conducted. In [Luo & Feng 2015] context-awareness is used along with Near Field Communication (NFC) technologies to provide better mobile services, while in [Khan *et al.* 2020] is used to create a recommender system for intelligent markets. Since then, its concepts and tools are refined, expanded, and consolidated and made its adoption invade more fields such as the employment [Belkadi *et al.* 2020], education [Psyche *et al.* 2020], cyber-security [Verginadis *et al.* 2017], [Kayes *et al.* 2019], [Arfaoui *et al.* 2020], Internet-of-Things [Gochhayat *et al.* 2019], and computer science tools [Lee *et al.* 2018], [Wang & Varghese 2020]. Context-awareness has also revolutionized the field of health-care systems through enabling more sophisticated functionalities that save people lives and improve the medical surveillance of some critical cases. Important works are reported in [Abdellatif *et al.* 2019], [Forkan *et al.* 2015].

In general, reconfigurable systems services are also improved through context-awareness concepts. In [Singh 2020], a context-awareness solution is provided for cars smart parking. In [Horcas *et al.* 2019] context-awareness is used as a key solution of software adaptation enabling energy efficient applications of cyber-physical systems. By the advent of Industrie 4.0, many works have tried to provide solutions to the automatic adaptations. The authors of [Flatt *et al.* 2015] propose a context-aware assistance systems for smart factories. In [Choi *et al.* 2018], authors introduce a power equipment management solution for smart cities, while authors of [Donohoe *et al.* 2013] use context-awareness for intelligent microgird storage. In [Fkaier *et al.* 2016b], [Fkaier *et al.* 2016a] context-awareness is introduced to the software of an airport baggage handling system. In [Fkaier *et al.* 2017] it was integrated to the software processes of reconfigurable systems, while in [Fkaier. *et al.* 2020a] it was used for better awareness of microgrids.

2.3.2 Context Life-cycle

In order to use efficiently its contributions, context-awareness requires a definition of a clear and consistent information managing process. Many information processing life-cycles are already defined in the field of software engineering, such as the Intelligence Cycle [Shulsky & Schmitt 2002]. However, in the context-awareness computing realm, specific life-cycles are introduced such as the one reported in [Hynes *et al.* 2009]. In the current research, four phases are considered to be parts of the context life-cycle as mentioned in Figure 2.1; context acquisition, modeling, reasoning, and distribution.

Context Acquisition: In the context acquisition phase, data is received, measured, sensed, and/or inferred from data sources. Sensors play a fundamental role

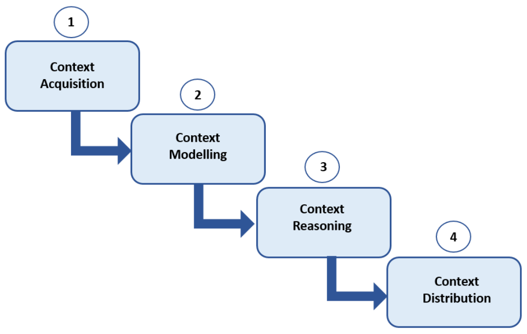


Figure 2.1: Context life-cycle.

in providing context information, different types of sensors can be utilized such as light, humidity, temperature sensors, etc.

Context Modeling: After obtaining the context information, a common presentation format is needed to be defined in order to make the information ready for processing and storage by applications.

Context Reasoning: Modeled context information may not imply clear meanings. Due to this fact, a context reasoning is required to analyze data and produce conclusions based on context data.

Context Distribution: In this phase, context information is disseminated to applications.

In this thesis, the context modeling and reasoning phases are considered the most important phases in the context life-cycle as they contain conception and logic inside, and their performance impacts greatly the quality of the applications. Accuracy as well as resources consumption are to be taken into account when selecting or proposing modeling and reasoning techniques.

2.3.2.1 Context Modeling

To represent acquired context information, various modeling techniques are defined. In the following, the most used techniques are presented in a nutshell.

Key-Value Modeling: consists in pairs of properties and values that are used to model context data in many formats like text files. It is generally used for simple cases as it is a very simple modeling technique.

Markup Modeling: consists in modeling data using tags. Famous markup techniques such as XML can be used. It is better than the key-value modeling through enabling efficient data retrieve.

Object-Oriented Modeling: leverages the Object Oriented Programming paradigm since it provides re-usability and encapsulation. Concepts of class and its relationships (inheritance, polymorphism, etc.) help in creating sophisticated context rows.

Logic-based Modeling: consists in using the logic theories such as predicate, first-order, or fuzzy logic, depending on the considered system. Facts, rules, relations could be defined to express context data.

User-centric Modeling: consists in modeling context data through answering the five questions: when, where, who, what, why, and how. This technique makes the data easily readable and understandable by users.

Ontology-based Modeling: consists in using semantic representation of data. It allows to present a set of properties of context data, the relationships between them, and their categories. The Web Ontology Language (OWL) is considered as an important tool providing rich and complex knowledge about subjects.

After having over-viewed the most important techniques, **which modeling technique to use for context modeling of reconfigurable systems?**

2.3.2.2 Context Reasoning

Similarly as the modeling, in order to process collected contextual data, different reasoning techniques are defined according to the considered case.

Probabilistic Reasoning: the logic of this method assigns probabilities to facts related to the considered system. Then, mathematical probability theories are applied to infer knowledge, such as the stochastic process models, Markov models, Naive Bayes.

Ontology-based Reasoning: it is a semantic reasoning based on descriptive logic. It uses the ontology models modeled with the Semantic Web Rule Language (SWRL), Web Ontology Language (OWL), etc.

Rule-based Reasoning: it consists in a simple and traditional approach of knowledge inferring. IF-THEN rules are triggered to conclude knowledge. Rules are generally composed of two parts: condition and conclusion. The condition can be a premise or a set of premises valid simultaneously.

Supervised Reasoning: consists in training labeled data in order to predict outputs of unforeseen data. It helps to classify data by means of finding relationships of input data. Many techniques can be used such as Naive Bayes, random forests, support vector machines, linear and logistic regression, artificial neural networks,etc.

Unsupervised Reasoning: consists in finding unknown data patterns. This approach helps in exploratory analysis through automatically identifying data structures. Different techniques can be used such as K-means, cluster algorithms, hierarchical clustering, etc.

Fuzzy Logic Reasoning: consists in finding the degree of truth (real number between 0 and 1) of a variable instead of assigning it a boolean value (1 or 0). This technique helps to identify the partial truth of data.

After scanning the existing techniques, **which one to use for context reasoning of automated control systems?**

2.3.2.3 Discussion

Based on the literature overview of the context modeling and reasoning presented previously, it is clear that context-awareness computing represents a huge computing field that has manifold techniques in each step of the context life-cycle.

Concerning the modeling and reasoning (the two steps considered in the current work), a survey on the big variety of techniques ends with a conclusion that the ontology-based modeling method has privileges in terms of re-usability, interoperability, and extensibility. It allows also to ensure independence from use case applications through abstracting entities. More importantly, it offers better expressive context representations [Alegre-Ibarra *et al.* 2018].

Despite its efficiency in context modeling, the ontology-based approach has some limitations when used for reasoning. Researchers have shown that ontology-based reasoning is complex and ambiguous for most cases. Also, its suitability remains questionable for systems having timing constraints [Fkaier. *et al.* 2020a]. However, rule-based approach has the potential to provide promising results.

Machine/deep learning is an outstanding reasoning technique that is gaining great interest in the last few years and it is very powerful for various use cases. However, its efficiency and suitability for applications of reconfigurable systems are still open to questions. In fact, to fulfill its intended operation, machine learning techniques require substantial amount of information for training and obtaining new knowledge. For this, preparing **enough and good** data for each particular instance of a system is necessary, especially that even one additional condition could considerably impact the reasoning. This task becomes more and more costly by the increasing intricacy of the considered system. Moreover, machine learning draws upon the so-called "black box" models. Models are called black boxes because it is very difficult to understand them and they are of complex and unpredictable nature. However, it is required to provide easy and clear explanations of the conclusions obtained from a reasoning agent, especially for systems having the obligation to make financial and security reports (e.g. microgirds). As it can be seen, the adoption of such a technique may add complexity to the main tasks rather than making it simpler. Hence, it is wiser to use clear controllable approach such as the rule-based approach [Fkaier. *et al.* 2020a].

Given the presented analysis, a hybrid context information obtainment mechanism is proposed in this work. The mechanism builds upon ontological context models and rule-based reasoning. An enhanced reasoning process is proposed to minimize resource usage.

2.3.3 Context-Awareness Software

Despite of its fascinating and great contributions from a consumption perspective, context-awareness computing gives rise to difficult challenges for software engineers/developers. It is difficult to implement applications that have to perform multidisciplinary functionalities, i.e., application's functional requirements as well as context-awareness services. This is why, researchers have spent a great effort in proposing approaches at the aim of providing development supports for context-aware applications. Several approaches are introduced, among them software frameworks represent a promising solution.

2.3.3.1 Context-Awareness Programming Solutions

In order to produce a viable context-aware application, different approaches (including methods, tools, and programming paradigms) are defined in the state of the art and could be adopted during the development phase.

Concerning the programming paradigms, which are methods with which the programming activity is performed, some of the already existing paradigms in the field of software engineering are also applicable to the development of context-aware applications. Object-oriented, feature-oriented, aspect-oriented, agent-oriented, and service-oriented are examples of paradigms used to develop context-aware systems. The research reported in [Alegre *et al.* 2016] provides a survey on the usage of these paradigms in the development of context-aware systems such as the works reported in [Murukannaiah & Singh 2014].

Concerning the methods, which are systematic development processes composed mainly of phases and their related techniques, a set of examples are used in literature. Different terming is used to describe the proposed approaches such as development environments, toolkit, platforms, services, etc. Also, example use cases are introduced to show how these methods are to be used. The work reported in [Perera *et al.* 2013] surveys some of the most remarkable approaches. For instance, in [Chen *et al.* 2004] a context broker architecture called CoBrA is proposed, a context-aware service platform called CaSP is proposed in [Devaraju *et al.* 2007], an autonomic context management system is proposed in [Hu *et al.* 2008].

Concerning the tools, two main solutions have been largely adopted: middlewares and frameworks. Given that sensors are fundamental actors in context-aware systems, middlewares were a very interesting solution for many developers. A big variety of middlewares exists in literature. For example, the survey presented in [Li *et al.* 2015] summarizes the most important middlewares proposed between 2009 and 2015. Middelware design is generally influenced by directives specific to the use case such as interoperability, scalability, and storage needs. For instance, the authors of [Chaqfeh & Mohamed 2012] introduce a middleware of context-awareness in the Internet-of-Things (IoT), in [El Khaddar *et al.* 2015] and [Forkan *et al.* 2014] middlewares are introduced for healthcare systems, etc. Software frameworks have been widely researched due to their importance in terms of providing ready to expand applications. This is why, an overview of the most important works is presented in a separate paragraph in the following subsection.

2.3.3.2 Context-Awareness Software Frameworks

There is a plethora of frameworks defined with the aim of supporting context-aware applications development. Due to space limitation, this section presents only some of the recent ones that are defined in journal papers since 2014.

In 2015, the authors of [Forkan *et al.* 2015] propose a framework for personalized knowledge discovery in the assisted healthcare field. The framework uses huge data provided by the Ambient Assisted Living systems which is stored in cloud.

Later, in 2016, the authors of [Kim *et al.* 2016] propose a context-aware risk management framework for cold chain logistics called i-RM framework. The framework consists of two parts: a risk management system and ontology base. In [Alhamid *et al.* 2016] also authors propose a multimedia recommendation framework for user experience improvement. Usage history as well as social network are used to extract context data.

Then, in 2017, authors of [Aid & Rassoul 2017] propose a framework for natural disasters management. The framework is based on a SoA and relies on web services to allow the integration of devices. Additionally, in [Bucchiarone *et al.* 2017] authors propose a planning-based composition framework for featuring abstract requirements and context awareness in the field of Internet-of-Services (IoS). An automated IoS-based car logistics case is used to show the proposed approach.

After, in 2018, authors of [Iqbal *et al.* 2018] propose a framework for fog infrastructures in the Internet-of-Vehicles (IoV). The framework provides real-time context-aware services and it builds upon a fog layer architecture. Further in [Alhanahnah *et al.* 2018], authors propose a multifaceted framework to support the trust evaluation of cloud service providers. The framework relies on mathematical methods to accomplish its tasks. Moreover, in [Cheng *et al.* 2018] authors propose a service-oriented framework for mobile applications. The said framework relies on ontology context-reasoning.

Last, in 2019, [Sikder *et al.* 2019], authors propose a framework for detecting threats originated from sensors of smart devices. The framework offers an intrusion detection observation system to recognize benign and malicious behaviors.

2.3.4 Discussion

Frameworks represent a promising solution for the development of software applications. In fact, frameworks are generally created to improve the efficiency of software creation. This is achieved through promoting: (1) reliability: thanks to providing tested model of applications infrastructure, (2) developers productivity: via making them focus on the specific/additional requirements of applications rather than on the infrastructure, and (3) rapidity: through extending the generic framework building blocks to specific application blocks. Frameworks promote reusability which facilitates the applications development. However, a study of the above-mentioned context-awareness frameworks has led to identify some limitations (a recapitulation of the comparative analysis of the mentioned frameworks is presented in Table 2.2, the taxonomy used in its presentation is presented in Table 2.1):

- First of all, most of the existing frameworks are dedicated to the development of mobile applications and few are those that can be applicable to complicated reconfigurable systems. More importantly, existing frameworks either focus on the context-data life-cycle and do not take into consideration additional constraints, or they provide it in restricted ambiguous way.

- Providing UML models of a software infrastructure is of great importance and may help developers to efficiently extend its concepts especially when it comes to complicated systems. Whereas, most of the existing works do not provide the frameworks UML models and settle for textual description or non-unified graphical representation. Thus, it becomes hard to build complete UML models, and consequently to get difficulties in tasks related to the modeling.

- Despite its importance, verification of concepts by simulation still need additional theoretical proofs since simulations are generally dependent on the chosen test cases.

- Few are the frameworks that take into consideration the distribution of peers in a context-aware system. However, distribution is a very important feature of today's systems.

- Security concerns are inherent to any computerized system. Particularly, systems relying on geographically distributed peers inevitably need to include proper security measures in order to ensure systems reliability. Nevertheless, none of the mentioned frameworks has provided a solution to the security.

- Taking into consideration timing constraints when leading a reconfiguration process in the software logic is a crucial need to satisfy if accurate and reliable behavior is desired.

Table 2.1: Taxonomy used in Table 2.2.

Taxonomy	**Meaning**
UML Modeling	UML model available (✓), not clear how to use the framework to model applications (-).
Verification Technique	Formal (F), Simulation (S), Not mentioned (-).
CContext Modeling	Ontology-based (O), Key-Value (KV), Graphical modeling (G), Object-oriented (OO), Logic-based (L), What/When/Where/Who/What-about (5W), Property-based (P).
Context Reasoning	Ontology-based (O), Supervised learning (SL), Unserpervised learning (USL), Rule-based (R), Probabilistic (P), Artificial-Intelligence-based (AI).
Distributed Control	(✓) support distribution, (-) does not support distribution.
Knowledge Management	(✓) available, (-) not available.
Secure Coordination	(✓) available, (-) not available.
Real-time reconfiguration	(✓) available, (-) not available.
Application	Example provided by authors.

Table 2.2: Comparison of existing software frameworks for the development of context-aware systems.

Reference	Year	UML Modeling	Verification Technique	Context Modeling	Context Reasoning	Distributed Control	Knowledge management	Secure coordination	Real-time reconfiguration	Application
[Tang *et al.* 2014]	2014	-	S	O	-	✓	✓	-	-	health-care
[Forkan *et al.* 2015]	2015	-	S	L	SL	✓	✓	-	✓	health-care
[Kim *et al.* 2016]	2016	-	S	O	R	-	✓	-	✓	cold chain logistics
[Alhamid *et al.* 2016]	2016	-	-	L	P	-	-	-	-	multimedia
[Aid & Rassoul 2017]	2017	-	S	5W	-	-	✓	-	-	disaster management
[Bucchiarone *et al.* 2017]	2017	✓	S	P	AI	-	✓	-	-	Internet of Services
[Cheng *et al.* 2018]	2018	-	S	O	O,R	-	✓	-	-	smart mobile phones
[Sikder *et al.* 2019]	2019	-	S	P	P	-	✓	-	-	smart mobile phones

Given the shortcomings identified by the analysis and comparison of the existing frameworks, it can be concluded that there is a crucial need for a new framework that can be able to overcome all the mentioned challenges in a single software infrastructure using novel engineering techniques.

2.4 Formal Verification

As the development of reconfigurable systems is increasing in complexity, formal verification of the logic models on early stages becomes necessary. Formal verification consists in automatically checking design correctness through tracking errors and revealing ambiguities using mathematical techniques. Comparing to other verification techniques, formal verification offers a set of advantages such as allowing the detection of pitfalls in early design phases as well as the fast remedy (i.e., the sooner a bug is catched, the easier it can be fixed). Comparing to simulations, formal verification provides better efficiency since simulations require hardware or test beds, consume large time, and results are guaranteed unless the same tested scenarios are reproduced. On top of all this, simulations become more and more difficult to set with the increasing design complexity.

In addition to all the aforementioned advantages, formal verification allows also to obtain better written software, to generate test cases, and even to get refined documentation and maintenance support.

Formal verification helps verifying design correctness using mathematical techniques. These techniques can be classified into three classes: model checking, theorem proving, and equivalence checking [Grimm *et al.* 2018]: (1) Model checking is considered as property checking that builds upon a state-based approach. (2) Theorem proving is considered as a mathematical reasoning technique used to prove that an implemented system meets its design requirements. It builds upon a proof-based approach. (3) Equivalence checking consists in verifying that two designs are functionally similar. Two approaches can be used: the Logical Equivalence Checking (LEC) and the Sequential Equivalence Checking (SEC).

2.4.1 Petri Nets

Petri nets are a modeling tool that builds upon graphical and mathematical bases. It helps generally to design asynchronous, concurrent, parallel, distributed, and/or stochastic systems. Thanks to its simple elements, which are places and transitions, Petri nets play the role of visual aid as block diagrams and flow charts do.

Petri nets help to analyze behavioral properties of modeled systems through the defined initial marking. Properties are in general the reachability, boundedness, liveness, reversibility and home state, faireness, synchronic distance, coverability, and persistence. These properties are analyzable through different methods such as the incidence matrix and state equation, the reduction rules, the coverability tree, reachability graphs, linear invariant analysis, and model checking, etc.

To allow formal verification of distributed systems, specifying the system using distributed state formalism is generally performed. Different formalisms exist such as the Net Condition/Event Systems (NCES) which is a formalism for the modeling of discrete-event systems. It is considered as a Petri-net subclass extension allowing the modular representation of the system components. This formalism is improved through the ability to specify temporal properties in an extension called Timed Net Condition/Event Systems (TNCES), with the ability to specify reconfigurations in an extension called Reconfigurable Timed Net Condition/Event Systems (R-TNCES), and with the ability to specify probabilistic properties in an extension called Generalized Reconfigurable Timed Net Condition/Event Systems (GR-TNCES).

2.4.2 GR-TNCES Formalism

Comparing to previous formalism extensions, GR-TNCES provides the possibility to create rich specification , i.e., in addition to the modular discrete event-based modeling, timing, reconfiguration, and mainly it allows to specify probabilistic properties as well as resources constraints. In addition, it allows to control the verification time and complexity.

As presented in [Khlifi *et al.* 2019], the GR-TNCES formalism is defined as a network of R-TNCES given by: G= $\{\sum$ R-TNCES$\}$, where R-TNCES=(B, R) with B a behavior module and R a control module. Its behavioral module contains an additional set allowing the specification of probabilities on arcs. An example of the behavioral module, is depicted in Figure 2.2.

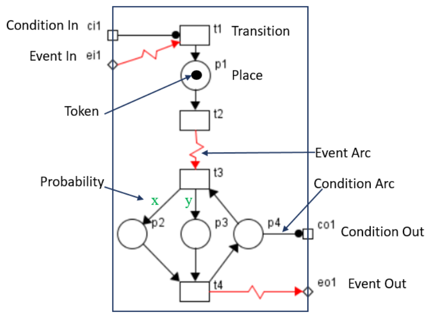


Figure 2.2: Example of behavioral module of GR-TNCES.

GR-TNCES Petri nets extension, as it is a generalized formalism, it is adopted in the specification and verification of many reconfigurable distributed systems such as medical robots, smart grids, microgrids, automotive, etc. The work reported in [Guellouz *et al.* 2016] uses the formalism to perform the formal verification of reconfigurable function blocks applied to a medical robot called BROS. The research reported in [Guellouz *et al.* 2018] uses GR-TNCES to verify a fault recovery strategy of smart grids. In [Khlifi *et al.* 2017] the formalism is used to verify the unpredictable reconfigurations running under resources constraints of a skid conveyor system of automotive industry. The work reported in [Smida *et al.* 2019] selects the extension to analyze the reliability of an autonomous and self-sufficient control strategy of microgrids.

2.4.3 Discussion

Given the numerous assets that are provided by the formal verification techniques, precisely by the GR-TNCES formalism, considering it to verify the models of designed applications may provide promising results. A solution that combines UML and formal verification modeling techniques can considerably reduce the burden on developers/designers to create robust and reliable applications behaviors.

2.5 IEC 61499: Standard of Distributed Systems

The IEC 61499 is a standard for the development of distributed control systems that involve intelligent devices and sensors. IEC 61499 comes to overcome the limitations of the IEC 61131-3 [Thramboulidis 2015] for the centralized control through allowing applications reconfigurability, portability, and interoperability. Applications are built by a network of function blocks. The function block is the elementary unit introduced by IEC 61499.

2.5.1 IEC 61499 Function Block Presentation

A function block is the basic unit of IEC 61499 standard and it is encapsulating data and algorithms. Every function block has an interface composed of a set of input/output events, input/output data, and the association between these events and data as depicted in Figure 2.3. The flow of events and data is from left (i.e., inputs) to right (i.e., outputs).

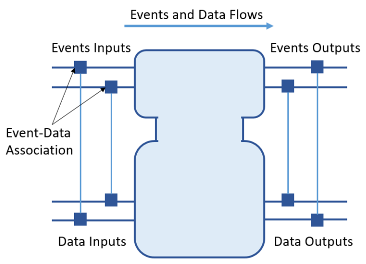
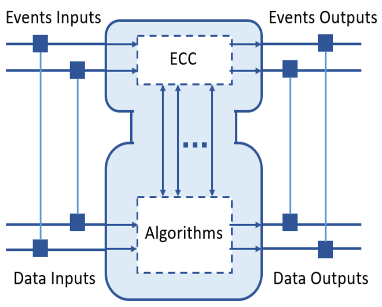


Figure 2.3: Function block type.

Two main types of function blocks exists: basic function block and composite function block. The basic function blocks are composed of two parts: the execution control chart (ECC) and a set of algorithms as depicted in Figure 2.4. These algorithms are written in object oriented languages such as C++ and Java or also in IEC 61131-3 programming languages such as Structured Text (ST). The ECC is an event-driven state machine connected to the algorithms. The execution of a particular algorithm is triggered by ECC transitions.

Basic function blocks could also be interfaces to services provided by other parts and it is called in this case Service Interface Function Block (SIFB).



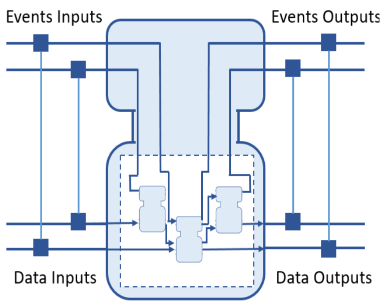


Figure 2.4: Basic function block. Figure 2.5: Composite function block..

A network of basic function blocks can be encapsulated in one higher-level function block known as composite function block as depicted in Figure 2.5. The work flow is determined through the interface of the composite type. Also the execution flow of internal algorithms of component function blocks is determined through the composed network.

2.5.2 IEC 61499 Applications

IEC 61499 concepts are used in the modeling of different distributed systems such as smart grids, material handling systems, product lines, manufacturing systems, telecommunication, and others [Thramboulidis 2007], [Andren *et al.* 2017].

The works reported in [Patil *et al.* 2013], [Yang *et al.* 2019a], and [Veichtlbauer *et al.* 2016] use the IEC 61499 function block as a basic modeling element of smart energy systems. For example, in [Patil *et al.* 2013], function blocks are used to develop a distributed load balancing approach. The standard is also used to develop material handling systems such as airports Baggage Handling Systems (BHS). The researches presented in [Yan & Vyatkin 2013], [Dai *et al.* 2015],and [Dai *et al.* 2016] provide important approaches to the automation and control of BHS using IEC 61499 concepts. The standard helps the development of product lines and manufacturing systems as mentioned in [Panjaitan & Frey 2007], [Garcia *et al.* 2017]. It is also used in the field of telecommunication, the work reported in [Lindgren *et al.* 2016] uses function blocks in the Ethernet switches development.

2.5.3 IEC 61499 and Intelligence

In order to improve systems responsiveness to changes without or with minimal human intervention, embedding intelligence in the system's control logic is highly recommended. However, few are the works that provide solutions that exploit intelligence paradigms (such as artificial intelligence, context-awareness, etc.) in favor of such systems. In [Nikolakis *et al.* 2018] the authors proposed an approach based on context-awareness to design shop floor intelligence and automation. IEC 61499 was used to characterize the system behavior. Authors of [Alsafi & Vyatkin 2010] have proposed an approach for fast reconfiguration based on an ontology knowledge of the manufacturing environment. In [Mousavi & Vyatkin 2015] authors have introduced an approach of agent function block in order to integrate intelligence in the controlling software. The intelligence is exhibited via communication with the environment and via knowledge reasoning.

2.5.4 Discussion

Although UML provides a multitude of useful diagrams allowing to properly design applications, using automation related standards to model applications helps definitely to improve its quality. Combining UML and IEC 61499 may clarify more issues related to the field. More importantly, IEC 61499 provides the opportunity to simulate models in a defined hardware environment thus to explore topics that help in the implementation. Hence, it is of great importance to align with IEC 61499 to get more powerful modelings.

2.6 Model Transformation

Model transformation approaches have been widely adopted in the software engineering field thanks to its benefits. It consists in an automated method of models creation and modification based on source models. Its main advantages are reducing modeling errors, avoiding deviations of the views of a model, and saving efforts.

In this thesis, we propose a modeling methodology of applications of context-aware distributed reconfigurable systems that builds upon model transformations. A new UML profile, called GR-UML, as well as a new software tool implementing the contributions of the methodology are introduced.

In the following, we present an overview of the related transformations methods that involve UML, the Petri nets formalism, and the IEC 61499 function blocks.

2.6.1 UML, Probability, Resource Constraints

UML is a semi-formal modeling language for the specification and modeling of software and systems. It offers a set of advantages that allow better development processes, such as using different types of diagrams, detecting errors, the organized view to the system/application, etc. However, UML still do not provide the semantics to model probabilistic scenarios running under memory and energy constraints, which are very important features to the software of reconfigurable distributed systems.

In [Addouche *et al.* 2006] have added new semantics to the class and statechart diagrams in order to verify probabilistic properties. Later, the authors of [Nokovic & Sekerinski 2013] have extended state diagrams in order to allow probabilistic verification. However, both works do not allow the verification of probabilistic resource-constrained scenarios neither they allow to express reconfigurations. The research reported in [Salem *et al.* 2015a] has extended the semantics of UML to become suitable for reconfigurable systems, however, despite its importance, this work does not enable the modeling and verification of reconfigurable probabilistic logic that must run under resources constraints.

2.6.2 Transformation Between UML and Petri Nets

Despite of its various advantages, UML still not ensuring models reliability and correctness. Hence, developers have introduced many transformation approaches that map UML and its profiles to formal models. Authors in [Noulamo *et al.* 2018] have proposed a transformation of UML statechart diagrams to time colored Petri nets. In [Grobelna *et al.* 2010] a transformation of UML activity diagram into Petri nets is introduced. The study provided in [Salem *et al.* 2015a] proposes a transformation of R-UML profile to R-TNCES formalism. The research reported in [Cardoso & Sibertin-Blanc 2001] presents a transformation of UML sequence diagram to Petri nets. To the best of our knowledge, there are no transformations that between UML and GR-TNCES.

2.6.3 Transformation Between UML and Function Blocks

IEC 61499 is the standard concerned with the distributed control system and having as main concept the building of applications using the function block as primary element. This fact (i.e., using the function block as the only modeling means) imposes some limitations especially when it comes to big and complex systems. Thus, many works have been proposed to integrate the UML diagrams and to introduce transformations between UML diagrams and function blocks.

These existing transformations can be classified into two types: (1) Those adding UML diagrams to improve the whole development process of applications based on function blocks. In fact, UML supports the different phases of applications development life-cycle with a set of diagrams (for example during the requirements elicitation), while the IEC 61499 proposes only one tool which is the function block one [Tranoris & Thramboulidis 2003], [Thramboulidis 2004], and [Panjaitan & Frey 2006]. (2) Those adding UML to have more fine-grained application architectural modeling through exploiting the manifold types of diagrams offered by UML [Dubinin *et al.* 2005].

2.6.4 Transformation Between Petri Nets and Function Blocks

Looking from another point of view -this is generally adopted by control engineers not software ones- applications can be modeled directly using function blocks then transformed into Petri nets for formal verification. The work reported in [Pang & Vyatkin 2008] presents an automatic function block model generation using the Petri nets formalism Net condition/Event Systems (NCES). Authors in [Ivanova-Vasileva *et al.* 2008] have also proposed a transformation between function block interfaces and NCES formalism in order to prove the correct behavior of distributed control systems. Recently, the study reported in [Guellouz *et al.* 2018] presents a transformation between an extended version of function blocks and the GR-TNCES formalism.

2.6.5 Discussion

Using UML for modeling software applications is adopted in most of the projects, hence it is wiser to rely on "almost" conventional language to create applications models. Therefore, considering the UML model as the basic modeling technique and using transformations for additional modeling and analysis techniques may provide good results. However, current UML semantics and existing profiles are not suitable to the modeling of probabilistic scenarios that must run under memory and energy constraints. Hence, there is a need for more adequate semantics for the efficient modeling of such features.

Coming now to transformations, for the best of our knowledge, there are no transformations from UML to GR-TNCES models. Also, existing transformations from UML to function blocks are simplistic and not sufficiently adequate. Consequently, there is a pressing necessity for more thorough transformations that would transform UML models to function blocks and to GR-TNCES. A systematic methodology involving model transformations as well as appropriate software tools are required for better modeling efficiency.

2.7 Blockchain for Distributed Systems Security

Blockchain [Yuan & Wang 2018] is a digital ledger of transactions that is distributed and duplicated in every peer in the system. The ledger is presented by a list of blocks chained using cryptography. A block has a cryptographic hash of the precedent block, transaction data, and a timestamp. Each time a transaction is made, a record containing its data is added in every peer's copy. The blockchain is designed to be resilient to data change or hacks on the system. In fact, the deployment in a peer-to-peer network helps in validating the blocks. The validation is generally ensured using a consensus protocol that must be run by all or the majority of the network. Blockchain is introduced in 2008 by Satoshi Nakamoto to be the public transaction ledger of the cryptocurrency Bitcoin [Nakamoto 2019]. It was introduced to overcome the need to a trust authority/party.

2.7.1 Blockchain Applications and Types

Applications: Blockchain technology was first used in the financial field, namely the digital currency Bitcoin. Thanks to its advantages, especially the decentralization, the transparency, and the immutability, blockchain was evolved and introduced in many other fields. Ethereum is an important blockchain solution that allows the development of all distributed applications types through the DApps. DApps provide the execution of application's code through smart contracts in which the user's rights and obligations are formulated. In December 2015, the Linux Foundation has introduced Hyperledger to be an open source blockchain platform dedicated to the development of global business transactions. The use of blockchain is still expanding to cover many other fields such as the IoT, the health insurance, the smart grids, etc., [Salman *et al.* 2018].

Types: Blockchain technology is of three types: (1) Public blockchain: where the access to the network is open to anyone. The validation of transactions is also open. Bitcoin and Ethereum are two famous example of public blockchains. (2) Private blockchain: are blockchains where there are rules indicating who can access and see the chain. Private blockchains are generally better suited to enterprises. (3) Consortium blockchain: this type is considered as semi-private since the validation of the blocks is made by some authorized nodes in the network [Salman *et al.* 2018].

2.7.2 Consensus Protocols

To reach an agreement about the state of the blockchain, consensus protocols are introduced to ensure a Byzantine Fault Tolerance. Different protocols are designed to fit different use cases, among them are the proof-of-work, proof-of-stake, delegated-proof-of-stake, leased-proof-of-stake, proof-of-activity, proof-of-burn, proof-of-capacity, proof-of-authority, proof-of-elapsed-time, directed-acyclic-graphs, practical-Byzantine-fault-tolerance, proof-of-importance, etc. [Yang *et al.* 2019b]. Most of the listed protocols are not developed rather are still concepts needing to be analyzed and proved. However, proof-of-work and proof-of stake are the most used and proved ones in the context of blockchains. This is why we build conduct a comparison between these two.

Proof-of-Work (PoW) is a consensus method used in public and private blockchains and its reliability and security are proved. It provides a solution for the transactions taking place between trustless parties. However, it has one main drawback which is greediness in terms of energy consumption.

Proof-of-Stake (PoS) is introduced to overcome this limitation of the proof-of-work therefore to consuming less electricity. However, proof-of-stake entails some drawbacks making its efficiency questionable: **The nothing at stake problem:** whenever a fork is created, whether it is accidental or a malicious attempt, this method states that validators can rewrite the history and reverse a transaction, then collect rewards without verification about simultaneous voting. **Vulnerability:** rich malicious nodes have the ability to destruct the system since money is the only used tool, however in the proof-of-work money, electricity, expertise, and time are used for verification. **The rich-get-richer problem:** the richest stakeholders have the highest chance to validate blocks thus to collect more rewards, and consequently to get richer and control the whole system. **The long range problem:** this is a prospective attack where validators try to start a fork from the old past (hundreds of thousands of blocks back). It can happen that attackers mine thousands of blocks into the future since no proof of work is required. Therefore, new users cannot know which chain is the correct one.

As it can be seen, both techniques have inherent limitations (for PoW is the energy consumption and for PoS is the security). In the current thesis, the focus is made on security and it is assumed that the energy consumption is manageable. For this, the proof-of-work is considered as better solution for enabling consensus protocol between the untrusted parties.

2.7.3 Blockchain Challenges

As any other technology, despite the benefits that it provides, blockchain has some challenges mainly the privacy and scalability ones.

Privacy: Generally, participating in a blockchain network implies revealing the identity of the participants. This fact can impose a serious problem for participants especially when it comes to competitor traders (no one like to show its balance to competitor neither to clients). In addition, knowing all financial balance details can be exploited by malicious parties to conduct attacks based on political or criminal purposes, etc. Hence, privacy concerns should be taken into account for blockchain-based systems.

Scalability: The concept of blockchain dictates that new blocks are continuously added to the chain without ever deleting any information. Hence, the size of the chain is always growing consequently there is a challenge of exponential storage. Therefore, it is necessary to deeply analyze the way of use of the chain (exchanged data intensity, size, etc.) and to provide convenient and “affordable” solution.

2.7.4 Discussion

The adoption of the blockchain technology for ensuring security especially in distributed systems is as stated very beneficial. However, since it is still relatively a new technology, some of its issues are still open to research. To begin, it is substantial to provide privacy solutions to the participants otherwise malicious tracking and tracing could happen. Privacy is required to protect not only identities but also exchanged data. Further, in order to guarantee timing efficiency, defining time frames of transactions exchange over the blockchain should be defined, consequently it is first required to know how to calculate the confirmation time of blocks added to the blockchain. However, and to the best of our knowledge, the calculation of confirmation time is not yet defined. Moreover, scalability has been an inherent challenge to the blockchain, hence any proposed solution should take into account the storage costs optimization.

2.8 Conclusions

As it can be seen in the literature overview (see discussion subsections), many limitations are faced when trying to find suitable solutions and tools that may help in obtaining smart applications and in efficient way.

The need to integrate advanced intelligence techniques, namely context-awareness, in the controlling software becomes day-by-day a necessity to keep up with the increasing requirement's complexity. Most of the existing frameworks are dedicated to the development of context-aware mobile applications and few are those that can be applicable to complicated reconfigurable systems. More importantly, existing frameworks either focus on the context-data life-cycle and do not take into consideration additional constraints, or they provide it in restricted ambiguous way. From another side, security techniques that fit to both the distribution paradigm (as it is a prominent paradigm of modern systems) and safety requirements are needed. Integrity, transparency, and privacy are important features to ensure for reliable communications. Having an efficient systematic methodology to model applications is also of great importance. Guarantees about obtained models correctness and efficiency are required.

In this context, we propose a software framework for the development of context-aware applications of distributed reconfigurable systems. Briefly, the contributions of the thesis lie in introducing: a definition of the term "context" for reconfigurable automation and control systems, an effective context reasoning approach, a framework meta-model, a framework (architecture, composition, mechanisms, and software infrastructure), a modeling methodology for better correctness and deployment guarantees in early stages, a new UML profile, a software tool implementing the concepts of the methodology, and a secure coordination protocol based on the blockchain technology (a method to calculate confirmation time for blockchain in general is also introduced).

Chapter 3

A Software Framework for Context-Aware Reconfigurable Applications

Contents

3.1 Introduction

This chapter introduces the basics of the proposed software framework. First, the theoretical abstraction is presented through a meta-model upon which the framework concepts are built. Thereafter, the framework architecture design and mechanisms are defined including the different layers and modules. The contributions of the framework are applied to an airport baggage handling system as an example application implementing the architecture.

The contributions of this chapter are summarized as follows:

- Introducing a new definition of the term “context” for reconfigurable automation and control systems.
- Definition of a meta-model for context-aware reconfigurable automation and control systems.
- Introducing an efficient context modeling and reasoning approach.
- Introducing the framework (the architecture including the layers, modules, and its mechanisms are defined).

3.2 Motivation

As the smart behavior becomes more and more important for future reconfigurable systems, the reusability of its software model and code gains more concern. Reusability offers the opportunity to reduce development costs via decreasing time and risks and increasing efficiency. In fact, due to the increasing size and complexity of systems, providing reusable software infrastructures may help to achieve better effectiveness, by making developers spend more time and effort on the system particularities rather than on repetitive and generic parts.

As analyzed in the state-of-the-art chapter, a set of requirements needs to be taken into account when introducing a software framework for reconfigurable context-aware systems. These needs are summarized in the following list:

- The need to define the term context for reconfigurable automation and control systems.
- The need to define an efficient context-awareness reasoning process.
- The need to ensure coordination in distributed reconfigurable systems.
- The need to introduce artificial intelligence mechanisms to provide systems with more sophisticated operations.
- The need to include security techniques.
- The need to preserve functional constraints.

- The need to preserve real-time constraints.

In order to take profit of the assets of software reusability, and in order to satisfy the list of redundant requirements above-mentioned, we propose in this chapter a software framework dedicated to the development of context-aware reconfigurable systems. We present the architecture and mechanisms of the framework side by side with the component's UML models (both behavioral and structural views).

3.3 Context Definition for Context-Aware Reconfigurable Automated Control Systems

Before diving in the details of the mechanisms defined here, it is needed to define what is a context for reconfigurable automated control systems. As presented in Chapter 2, most of the context definitions existing in the literature are user-centric and they generally draw upon the fact that the application is mobile. However, these definitions are not that relevant for reconfigurable automated control systems. Even the most acknowledged definition reported in [Abowd *et al.* 1999] does not fit very well with what is intended from a reconfigurable system. For these reasons we introduce the following definition to describe what is a context for context-aware reconfigurable systems:

New Context Definition for Context-Aware Reconfigurable Automated Control Systems
A context is the combination of specific conditions having the potential to make a system reconfigure itself, where conditions are data sensed and/or measured and/or read from the surrounding environment that reflect system reconfiguration parameters.

Now that what we intend by "context" in the course of this research work is defined, we can move to present the dynamics of the framework modules.

3.4 Framework Meta-Model

In modern reconfigurable systems, distribution of the software and intelligence are becoming more and more required to achieve better efficiency. This makes the development of applications more and more difficult. Hence, it is important to start with defining high level abstraction modeling for such huge and complicated systems. An abstract and generic meta-model will help in offering commonly understandable software structures and unified view to such systems.

However, defining a meta-model of software applications for such kind of systems is challenging due to many reasons that could be summarized in three points: (i) the multidisciplinary requirements, (ii) the big number of constraints, and (iii) the relation between the physical and computational components of the system. This is why, it is first required to define the directives of the definition of a meta-model through specifying what requirements and principles are to be considered.

3.4.1 Concept Principles

In order to easily define the software design, an overview of a set of examples (including the smart microgrids and the smart baggage handling) leads to recognize common characteristics of these systems [Fkaier *et al.* 2020b], [Fkaier *et al.* 2017]. In spite of the application field and the functional particularity, there are some recurrent components and processing tasks existing in the majority of them. In fact, all of them require first to acquire information from their environment, to check a set of constraints mainly functional and real-time ones, to satisfy some needs such as intelligence operations, etc.

At the aim of easily elucidating the requirements of an application and reducing the design complexity, abstraction models become really necessary. Also, in order to provide consistent modeling base, different perspectives should be taken into account to cover all system facets. For this, three modeling perspectives are provided in this work: the constraints, the behavioral, and the structural perspectives.

3.4.1.1 Structural Perspective

An analysis of the tasks to be fulfilled by smart software applications of reconfigurable automated control systems have led us to conclude that three abstract functional levels could be defined: (i) communication level, (ii) processing and control level, and (iii) functionality store level. Figure 3.1 presents the considered structure.

The definition of the three levels is built upon the class/type of topics and responsibilities to be performed by the application. In fact, all input/output operations could be classified in one level called *Communication Level*. Similarly, all algorithms from the different disciplines could be encapsulated together in one level called *Processing and Control Level*. Finally, all the code of the system particular functionalities could be organized in a separate level called *Functionalities Store Level*.

The first level, i.e., the communication level, has as topic the interaction with the environment that the application needs to behave within. The responsibilities of this level are: (i) the sending/reception of messages from/to other peers in the system, (ii) the reading of input information sensed by sensors, (iii) the reading of metering and measurements performed by metering units, and (iv) the application of decisions and the actuation of actuators.

The second level, i.e., the processing and control level, has as topic the checking of constraints such as the functional and timing ones, the satisfaction of requirements such as the intelligence, coordination, and security.

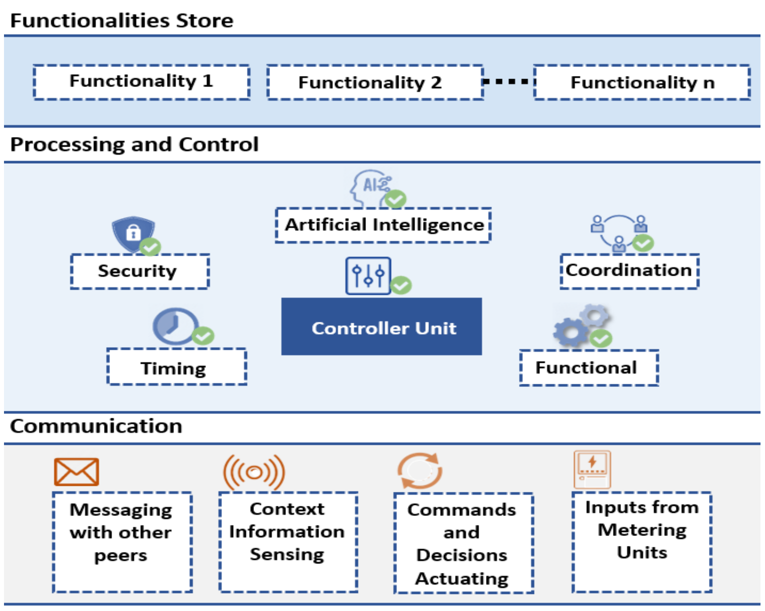


Figure 3.1: Generic structure of smart applications of reconfigurable systems.

The third level, i.e., the functionalities store level, has as topic the holding of the system functionalities in separate units.

3.4.1.2 Constraints Perspective

Reconfigurable systems are generally running under a set of constraints such as the timing constraints, the computational resources constraints (processing unit time, memory, and energy usage), functional constraints (priorities, dependencies, safety, coherence, etc.) and many other constraints. Since it devotes a considerable part of the computation to intelligence and awareness tasks, more constraints could arise. Therefore, more attention should be paid to satisfy all constraints.

Restricting the modeling on using the conventional modeling diagrams such as UML class or component diagrams may not be sufficient since not all relevant aspects can be represented. In fact, it is required, and especially when it comes to reconfigurable systems, to clearly specify constraints without ambiguities. For this, it is proposed in this work to define constraints using the formal language Object Constraints Language (OCL) to express constraints. OCL is integrated to UML and allows to easily read and write constraints.

3.4.1.3 Behavioral Perspective

An analysis of the behavior of smart reconfigurable systems allows to observe several shared states during its operation. Five main states are frequently used as depicted in the UML statechart diagram of Figure 3.2.

Systems need to read information from outside. *Reading Inputs* is an abstract

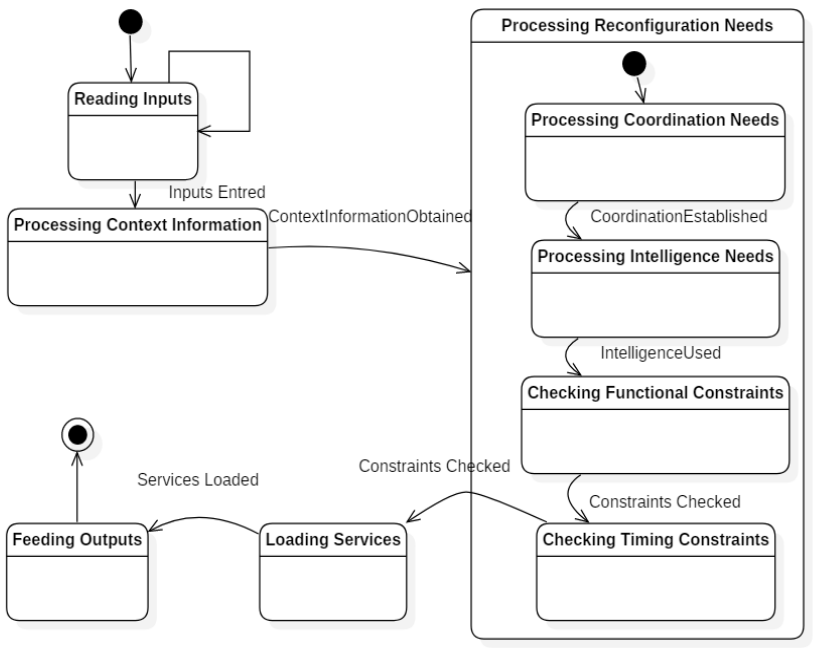

Reading Inputs
Inputs Entred
Processing Context Information
ContextInformationObtained
Processing Reconfiguration Needs
Processing Coordination Needs
CoordinationEstablished
Processing Intelligence Needs
IntelligenceUsed
Checking Functional Constraints
Constraints Checked
Checking Timing Constraints
Constraints Checked
Loading Services
Services Loaded
Feeding Outputs

Figure 3.2: Statechart diagram of generic context-aware reconfigurable system.

state that consists in other nested states as follows: the process starts with reading data which may be the reading of sensor's data, the measurements of measurement units, the messages sent from other devices, etc. Then, a reasoning about the received information is performed. Afterwards, conclusions of the previous step are extracted.

As offering a smart behavior is an important feature of future reconfigurable systems, checking some intelligence needs is necessary to provide. This can be achieved through two directives: (i) the collaboration and coordination with other distributed peers in the system, and (ii) the reasoning and perception of contextual information to help taking the proper decisions. The artificial intelligence concepts should also be considered for more smart processing. Hence, the *Processing Reconfiguration Needs* state includes two nested states: *Processing Coordination Needs* and *Processing Intelligence Needs* (as mentioned in Figure 3.2) which are responsible for intelligent behaviors.

Verifying the satisfaction of operational constraints is required to ensure systems efficiency. Functionalities of a system could have many functional constraints such as the coherence, the precedence, or the priorities constraints. Moreover, timing constraints are deemed to be very important ones. Therefore, checking constraints is performed through the *Checking Functional Constraints* and *Checking Timing Constraints* as presented in Figure 3.2.

The services (i.e., functionalities, operation modes) to be offered by the system are loaded in the state *Loading Services*.

Finally, the decisions made by the logic control unit are written to the output objects in the state *Feeding Outputs*.

3.4.2 Meta-Model

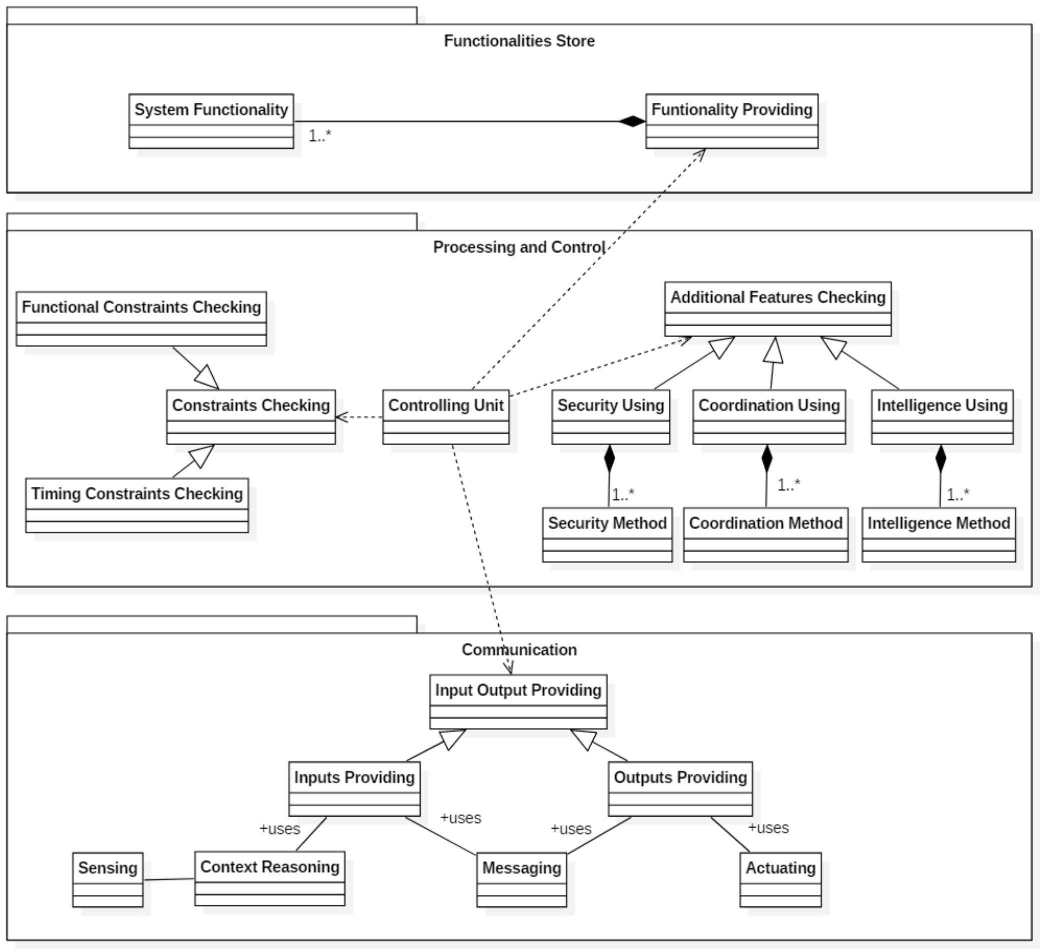


Figure 3.3: Generic meta-model of context-aware reconfigurable controlling software.

Figure 3.3 depicts a meta-model for the controlling software of generic reconfigurable systems where *Controlling Unit* plays the role of the most important element in the meta-model. It uses tow main meta-classes : *Constraints Checking* and *Additional Features Checking.*

Constraints Checking represents a generalized meta-class of the constraints under which an application can run such as the timing constraints (designed through the meta-class *Timing Constraints Checking*) or the functional constraints (designed through the meta-class *Functional Constraints Checking*).

Additional Features Checking represents a generalized meta-class of a set of additional features that an application may have such as the security operations (designed through the meta-class *Security Using* which is composed of a set of security methods), the coordination operations (designed through the meta-class *Coordination Using* which is composed of a set of coordination methods), and artificial intelligence operations (designed through the meta-class *Intelligence Using* which is composed of a set of intelligence methods).

The *Controlling Unit* uses the meta-class *Functionality Providing* to get some application functionalities and the meta-class *Input Output Providing* to interact with its environment.

3.5 Definition and Formalization

The core of the framework consists in a novel architecture offering efficient algorithms and allowing the communication with the applications environment. The framework architecture is designed to fit most of the reconfigurable applications that builds upon context-awareness and requires a set of miscellaneous requirements such as the needs to intelligence, security, timing, functional safety, and collaboration [Fkaier *et al.* 2016a] [Fkaier. *et al.* 2021b]. To present the concepts of the framework, an example of conveyors of airport baggage handling systems is considered in this chapter.

As the Unified Modeling Language (UML) is a semi-formal and internationally standardized modeling language, the framework concepts are introduced using UML (class, sequence, component, statechart diagrams).

3.5.1 Framework architecture

Figure 3.4 presents the proposed framework architecture. Based on the defined meta-model and in order to get loosely-coupled application components, the architecture consists of three layers where each is composed of a set of modules.

The first layer, Reconfiguration Layer, is responsible for the communication with the environment of the application and for reasoning contextual inputs. The second layer, Context Control Layer, is responsible for controlling the logic of applications. Finally, the third layer, Service Layer, is responsible for storing the application functionalities as services.

The framework is formally defined as a tuple

$$FW = (RL, CCL, SL) \tag{3.1}$$

where RL, CCL, and SL stand respectively for Reconfiguration Layer, Context Control Layer, and Service Layer.

In the following sections, definitions of the dynamics of the components of each layer are presented.

3.5.2 Reconfiguration Layer

The Reconfiguration Layer, denoted by RL, represents the interface between the framework and the external environment. Its main role can be generalized as han-

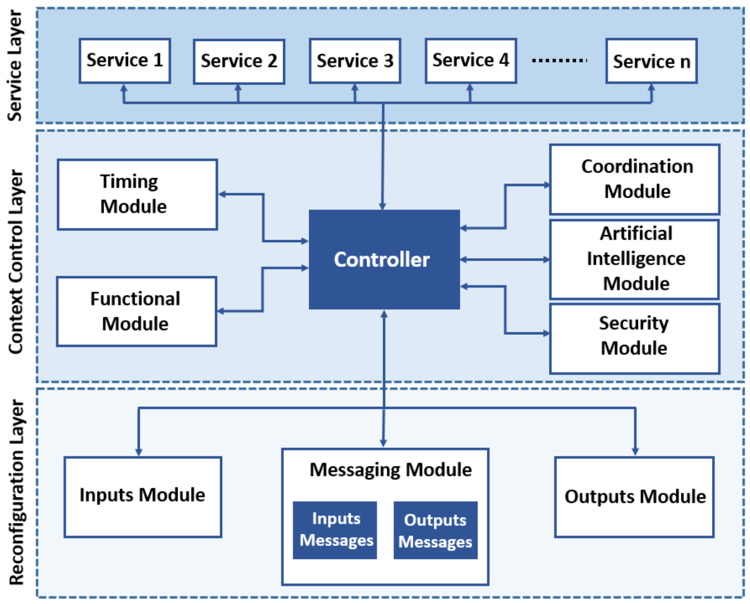


Figure 3.4: Architecture of the proposed framework.

dling the inputs/outputs. This layer is defined as

$$RL = (IM, OM, MM) \tag{3.2}$$

where IM being the Inputs Module, OM being the Outputs Module, and MM being the Messaging Module. In the following, details of each module are provided.

3.5.2.1 Inputs Module

The acquisition and recognition of context information from the surrounding environment is performed thanks to this module.

Composition of the Inputs Module

The inputs module IM is responsible for providing input data to the core logic of the framework, specifically to the controller (to be introduced later) in a form of a "context entry". The sources of input data are generally sensors and metering equipment. These input data are classified into critical and uncritical. A schematic description of IM is provided in Figure 3.5.

Data that are considered critical are sent directly to the controller whenever an event occurs without any further processing so that the controller can take rapid measures. For uncritical inputs, a reasoning is applied to data in order to determine the context and provide contextual recommendations to the controller. Reading inputs from uncritical data sources is performed in a periodic way. This additional processing and reasoning ensures the awareness of applications. A UML class diagram of IM is depicted in Figure 3.6.

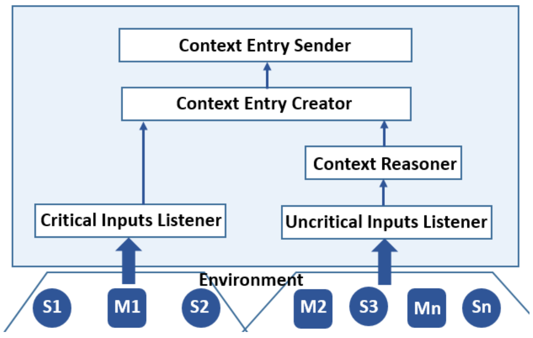


Figure 3.5: Inputs module scheme.

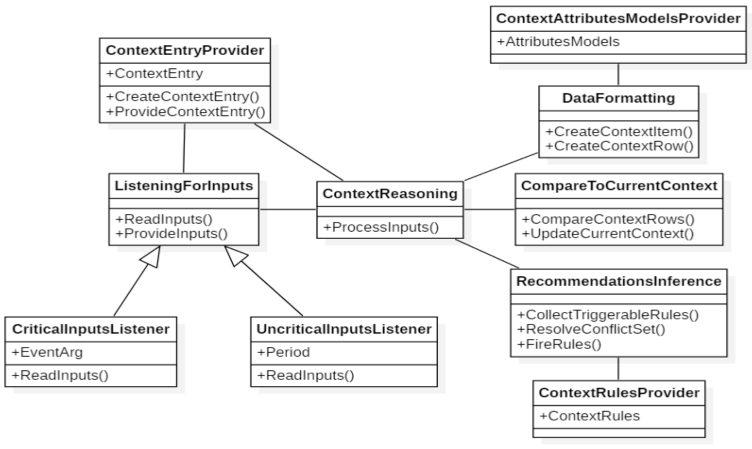
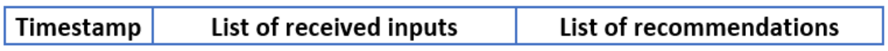


Figure 3.6: Inputs module class diagram.

The output of IM is a context entry which contains a time-stamp TS, a list of values of received data LRV, and a list of recommendations LR as depicted in Figure 3.7. The list of recommendations is left empty for the case of critical inputs.



Timestamp	List of received inputs	List of recommendations

Figure 3.7: Context entry structure.

Dynamics of the Inputs Module

The dynamic behavior of IM is provided through the statechart diagram presented in Figure 3.8. Ensuring the context-aware behavior to applications, is performed thanks to IM that has an important activity (i.e., state) which is the *Reasoning-Context*. This state is a composite state composed of three substates: *Formatting-Data*, *ComparingToCurrentContext*, and *InferringRecommendations*. In fact, these substates reflect a lightweight context reasoning process proposed in this research work to enable the context-awareness features [Fkaier. *et al.* 2020a]. Definition and details of the context-reasoning process are provided in the following section.

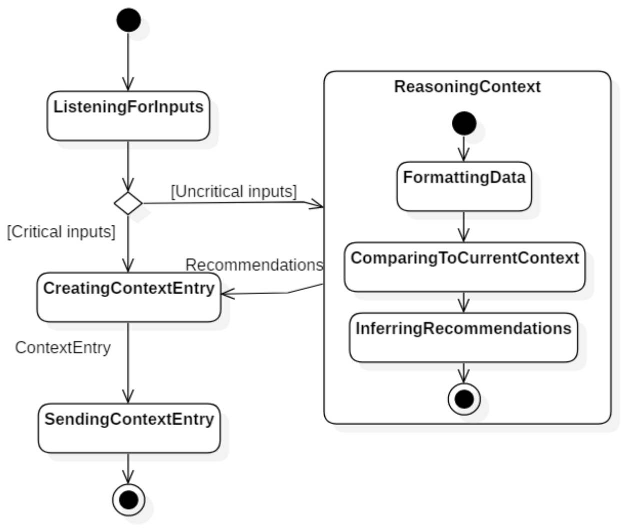


Figure 3.8: Inputs module statechart diagram.

3.5.2.2 New Lightweight Context Reasoning Approach for Reconfigurable Context-Aware Automation and Control Systems

Given the analysis of existing modeling and reasoning methods presented in Chapter 2, a hybrid context obtainment mechanism is proposed in this work. The mechanism builds upon an ontology-based context modeling and consists in a rule-based context reasoning, where an enhanced process is proposed to minimize resources usage. An example of use of the proposed mechanisms applied to microgrids is reported in [Fkaier. *et al.* 2020a].

Ontology-based Context Modeling: The adoption of ontology based modeling ensures a multitude of benefits. In fact, it offers the ability to build easy understandable models. More importantly, this technique is easily extensible. Extensibility is very important since modern technologies are continuously evolving and changes in a system environment are most likely to happen. Ontology-based modeling allows the derivation, addition, and/or modification of new semantics. The model can be easily created using the Web Ontology Language (OWL) tools.

Efficient Rule-based Context Reasoning: This research work proposes a lightweight process for context reasoning. The efficiency of the proposed rule-based process lies in smartly run the search for rules through diagnosing received data first. The reasoning process flow consists in three steps as depicted in Figure 3.9: (1) data formatting, (2) comparison to current context, and (3) search new context.

(1) Data Formatting

Sensed and measured data are generally provided in real numerical values, that without additional information are not meaningful and do not provide useful se-

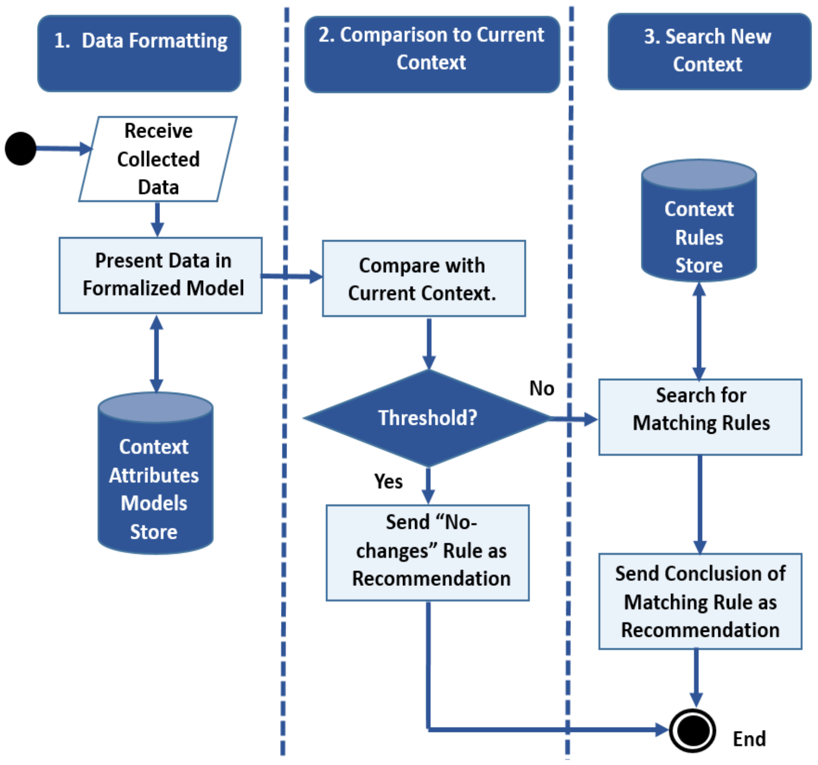

1. Data Formatting
2. Comparison to Current Context
3. Search New Context
Receive Collected Data
Present Data in Formalized Model
Context Attributes Models Store
Compare with Current Context.
Threshold?
No
Yes
Send "No-changes" Rule as Recommendation
Context Rules Store
Search for Matching Rules
Send Conclusion of Matching Rule as Recommendation
End

Figure 3.9: Proposed context reasoning process.

mantic. For this, the first step to do is formatting data in order to meaningfully present it.

Initially, developers or system owners need to define what kind of sensed and/or measured data the considered system is dealing with. This must be based on the defined ontological entities and their attributes defined in the modeling phase. Afterwards, a store containing the inputs values and their related meanings need to be defined in the form of a store called the Context Attribute Models Store (CAMS). Figure 3.10 depicts an example of context attributes that could be used to fill the store.

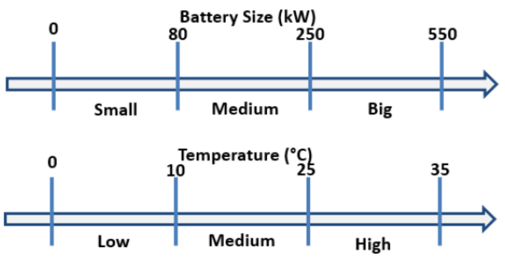


Figure 3.10: Context attributes models.

A context attribute model *cam* is defined as a tuple

$$cam = (cam_{type}, cam_{ranges}) \tag{3.3}$$

where cam_{type} stands for the semantic type of the input value and it is presented by its root in the ontology o, entity e, and attribute a. cam_{ranges} is a set of range of values such that $cam_{ranges} = \{range_i | i \in (1, ..., |cam_{ranges}|)\}$ where $range_i$ is defined as

$$range_i = (min, max, label) \tag{3.4}$$

with min is the minimum value of the range, max is the maximum value of the range, and $label$ is the label assigned to the range.

For example (see Figure 3.10), let us suppose that we have a model containing *temperature* as a context attribute of a *weather* ontology. The values in [0, 10] are labeled as low temperature, in [10, 25] as medium temperature, and in [25, 35] as high temperature; where *low, medium,* and *high* are defined as labels.

The $CAMS$ is used in this step to determine the context row CR which is defined as

$$CR = \{ci | i \in (1, ..., n)\} \tag{3.5}$$

with n being the number of inputs of the system, and ci is a context item defined by

$$ci = (ci_n, ci_v, ci_l) \tag{3.6}$$

with ci_n is the name of the item (i.e., input), ci_v is the input value, and ci_l is the label to be assigned to the item.

The pseudo-code of the *Data Formatting* step is the following:

Algorithm 1 Data Formatting Pseudo-code

```
Input: $LRV$
Output: $CR$
for each $value$ in $LRV$ do
  Search correspondent context attribute model $ccam$ from $CAMS$
  for each $range$ in $ccam$ do
    if $value \in [range_{min}, range_{max}]$ then
      Create a context item $ci$ such that $ci_n = ccam_{type}, ci_v = value, ci_l = range_{label}$
      Add $ci$ to $CR$
```

(2) Comparison with Current Context

In order to get efficient applications, context information should be available in real-time. Thus, calculations have to be performed continuously. However, in many cases the input data could be repeated or a little different from the current one. Thus, computations will return near results which means the same context. To overcome this limitation and to improve the performance of the computing units, we propose to create this second step, which is responsible for comparing the new input values LRV with the current ones, denoted by $V = \{v_1, ..., v_n\}$. The comparison is performed to a predefined similarity threshold that must be fixed based on the system particularity.

This difference calculation is performed through the following formula

$$D = \sum_{j=1}^{n} (|value_j - v_j| / [(value_j + v_j)/2] \times 100)/n. \tag{3.7}$$

For instance, let us consider the current context values $V = \{35, 1, 1100\}$ and let the new input values be $LRV = \{30, 1, 1000\}$. Let the threshold value be 10%. Calculating the difference D gives 8.3% which is less than 10%. In this case, there is no need to pursue the process since there is no context change. However, if the difference is more than the threshold then, new context must be determined.

(3) Search For New Context

To determine the new context, a Context Rule Store (CRS) must be set. CRS contains rules, where a rule has the form of *conditions* → *conclusion*, with *conditions* being a conjunction of premises. The labels of context items of the input context row (determined in Step 1) are used as conditions of the rules, while conclusions are considered as recommendations to be sent in the context entry to the controller of the upper layer of the framework.

The pseudo-code of the search for new context is given as follows:

Algorithm 2 Search For New Context Pseudo-code

```
Input: $CRS$, $CR$
Output: $LR$
repeat
  Select all triggerable rules from $CRS$.
  if more than one rule is found then
    Compare the number of premises of the rules.
    Start triggering from longest to shortest rule.
    if there are rules equal in length then
      Calculate Euclidean distance between the input values from $CR$ and the corresponding mid-range
      of context attributes.
      Start triggering from the rule having the minimal distance to the input.
      Add conclusions to $LR$
until no triggerable rules in $CRS$
```

3.5.2.3 Outputs Module

The role of the Outputs Module OM is conveying commands/decisions made by the controller of the upper layer to the connected output devices.

Composition of Outputs Module

After all necessary processing made by the upper layer (i.e., Context Control Layer), decisions or commands to be applied to the connected devices are managed through this outputs module. OM is responsible for feeding outputs devices -which are generally actuators- with new values considered as context consequences. Figure 3.11 depicts the class diagram of OM.

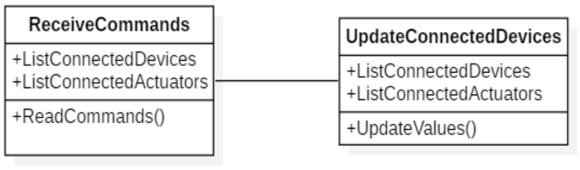


Figure 3.11: Outputs module class diagram.

Dynamics of Outputs Module

The dynamic behavior of OM is depicted through the statechart diagram provided in Figure 3.12. This module has a simple behavior that can be resumed in two important states *ReadingCommands* and *UpdatingDevices*.

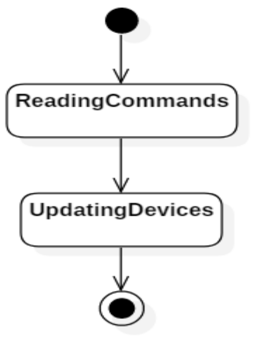


Figure 3.12: Outputs module statechart diagram.

3.5.2.4 Messaging Module

The proposed framework can also help in developing communicative software applications of distributed reconfigurable systems through enabling components of the system to communicate with each other using messages. Messaging allows components to be interactive and able to achieve self and/or other's objectives.

Composition of Messaging Module

This module MM is devoted to ensure: (1) sending messages from current controller to other parts in the system that the application behaves within, and (2) receiving messages from other parts. It is worth noting that, the transport of the messages between peers (i.e., outside the framework) can be performed using any technology and/or protocol relevant to the considered use case. Thus, regardless of the adopted technology, a mapping is necessary to convert messages in order to guarantee a conformance with the structure required by the framework.

A message exchanged between MM and the controller of the upper layer has a structure composed of four fields: a time-stamp, the source identifier, the destination identifier, and the content. Figure 3.13 depicts the class diagram of MM.

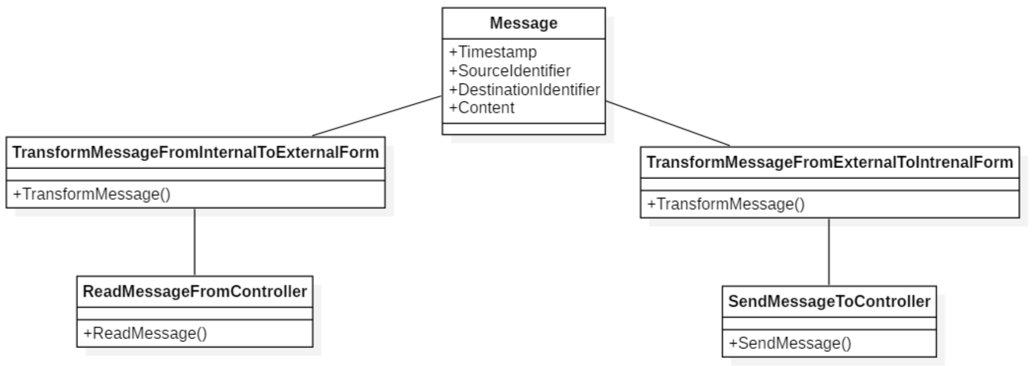


Figure 3.13: Messaging module class diagram.

Dynamics of Messaging Module

This module MM has two main behaviors, as depicted in Figure 3.14, receiving and converting messages from outside of the framework, and sending and converting messages to the outside.

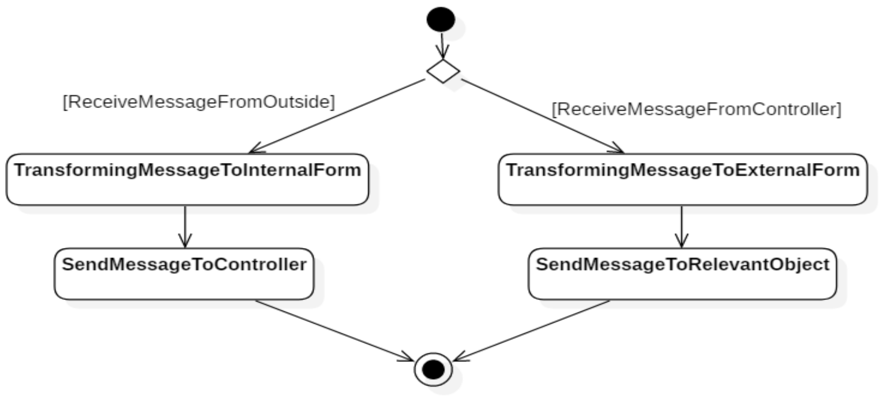

[ReceiveMessageFromOutside]
[ReceiveMessageFromController]
TransformingMessageToInternalForm
TransformingMessageToExternalForm
SendMessageToController
SendMessageToRelevantObject

Figure 3.14: Messaging module statechart diagram.

3.5.3 Context Control Layer

This layer, the Context Control Layer (CCL), is the central layer and it has a very important role since it contains a main part of the logic. CCL is defined as

$$CCL = (C, AIM, CM, FM, SM, TM). \tag{3.8}$$

To accomplish its roles, it relies on six modules: Controller (C), Artificial Intelligence Module (AIM), Coordination Module (CM), Functional Module (FM), Security Module (SM), and Timing Module (TM).

3.5.3.1 Artificial Intelligence Module

Complexity and increasing requirements of smart reconfigurable systems has manifested the need for robust entities that have the ability to handle new, unpredictable, intricate changes whenever it happens and also whenever it is prospective to happen. Artificial intelligence provides the ability to simulate the human intelligence in machines/applications that are programmed to resolve complex situations and to anticipate expected ones. This feature is becoming more and more in demand with the increasing attention paid to self-manageable systems which rely heavily on self-improvement issues. For this reason, we find it substantial to consider artificial intelligence as one of the mainstays of the software processes. Thus, we define this module AIM to be a container of a generalized artificial intelligence technique that can be used in multiple objectives depending on the system and the use case, for example to predict, plan, repair, etc.

Composition of Artificial Intelligence Module

The proposed artificial intelligence module AIM consists in an expert system, denoted by ES, composed of -as traditional expert systems- a knowledge base and an inference engine. Additionally, it contains a system history as depicted in Figure 3.15.

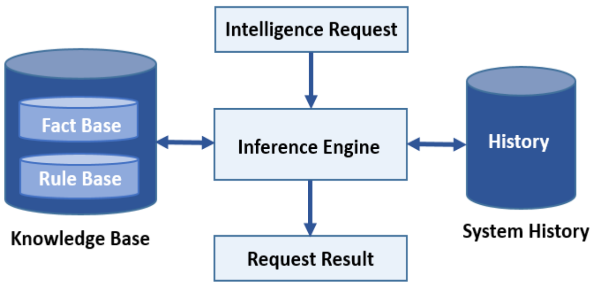


Figure 3.15: Artificial intelligence module scheme.

- Knowledge Base, denoted by KB, is composed of two bases: a facts and rules base.

  1. Fact Base (FB): a base containing the facts about the system. It is defined as $FB = \{f_i | i \in \{1, ..., n\}\}$, where a fact is defined by its name $< fact >$ and value $< value >$ such that $f_i :< fact >=< value >$.
  2. Rule Base (RB): a base containing rules which are logical forms enabling the deduction of new facts. It is defined as $RB = \{r_i | i \in \{1, ..., n\}\}$, where r_i is a rule having this form: $r_i : IF < premise > THEN < conclusion >$. The rules in RB must be defined by experts to resolve specific problems.

- History, denoted by H, contains the history of the system behavior defined as: $H = \{h_k | k \in \{1, ..., p\}\}$, where h_k is a previous demarche executed by AIM having this form $h_k = (IntReq, Cls)$ where $IntReq$ is an intelligence request sent by the controller module to AIM and Cls is the set of conclusions resulted after the processing of the $IntReq$ by AIM.

- Inference Engine, denoted by IE, is responsible for processing requests with the help of KB and H to infer knowledge. IE provides two reasoning methods, the forward and backward chaining, that could be used according to the specific cases.

Using a knowledge-base for expert systems allows developers to easily represent information since it is based on IF-THEN rules. It allows also to easily maintain knowledge thanks to the rapid process.

Dynamics of Artificial Intelligence Module

In order to improve the behavior of the inference process, we propose to add the history component to AIM to record all interactions with the controller. In fact, it might happen that some reconfigurations will be repeated during the life of an application. Thus, it is more efficient to save computational power and time with checking historical interactions first. Hence, the dynamic behavior of AIM is as represented in Figure 3.16. Based on the content of the request made by the controller $IntReq$, AIM reads the conditions, which are considered facts, and checks the history H to verify whether it is a new or repeated case. Then it decides whether to start off an inference process.

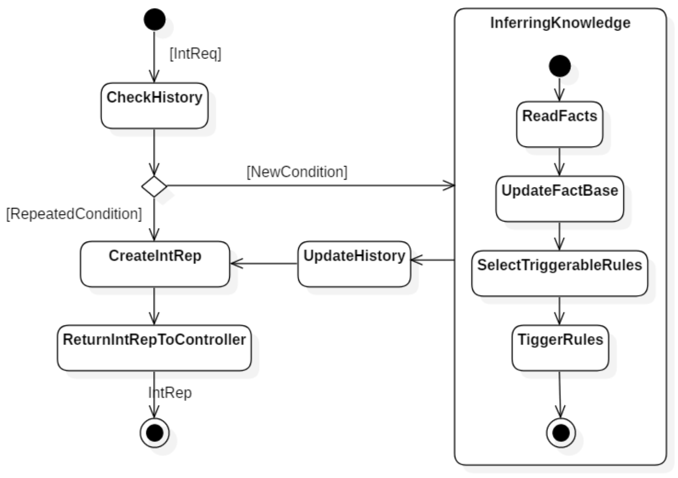
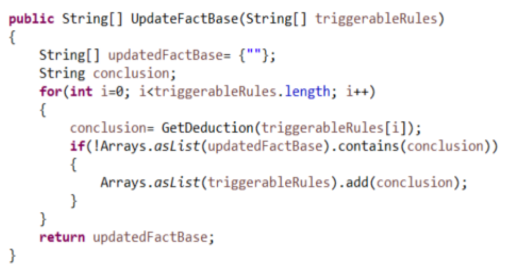


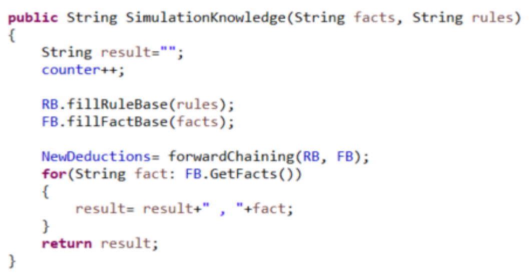
Figure 3.16: Artificial intelligence module statechart diagram.

A part of the implementation of the *InferringKnowledge* state is presented in Figure 3.19, Figure 3.20, Figure 3.17, and Figure 3.18.



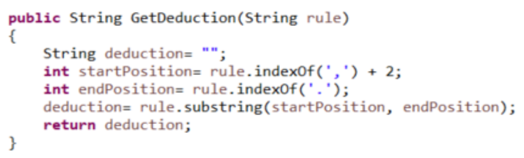
```
public String[] UpdateFactBase(String[] triggerableRules)
{
    String[] updatedFactBase= {""};
    String conclusion;
    for(int i=0; i<triggerableRules.length; i++)
    {
        conclusion= GetDeduction(triggerableRules[i]);
        if(!Arrays.asList(updatedFactBase).contains(conclusion))
        {
            Arrays.asList(triggerableRules).add(conclusion);
        }
    }
    return updatedFactBase;
}
```

Figure 3.17: Method allowing to update the fact base.



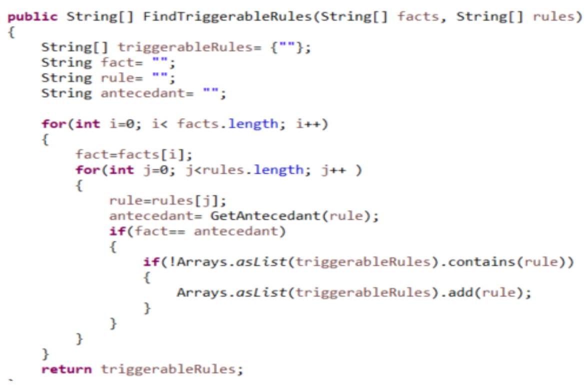
```
public String SimulationKnowledge(String facts, String rules)
{
    String result="";
    counter++;

    RB.fillRuleBase(rules);
    FB.fillFactBase(facts);

    NewDeductions= forwardChaining(RB, FB);
    for(String fact: FB.GetFacts())
    {
        result= result+" , "+fact;
    }
    return result;
}
```

Figure 3.18: Method allowing to simulate the knowledge base.



```
public String GetDeduction(String rule)
{
    String deduction= "";
    int startPosition= rule.indexOf(',') + 2;
    int endPosition= rule.indexOf('.');
    deduction= rule.substring(startPosition, endPosition);
    return deduction;
}
```

Figure 3.19: Method allowing to get the conclusion of a rule.



```
public String[] FindTriggerableRules(String[] facts, String[] rules)
{
    String[] triggerableRules= {""};
    String fact= "";
    String rule= "";
    String antecedant= "";

    for(int i=0; i< facts.length; i++)
    {
        fact=facts[i];
        for(int j=0; j<rules.length; j++ )
        {
            rule=rules[j];
            antecedant= GetAntecedant(rule);
            if(fact== antecedant)
            {
                if(!Arrays.asList(triggerableRules).contains(rule))
                {
                    Arrays.asList(triggerableRules).add(rule);
                }
            }
        }
    }
    return triggerableRules;
```

Figure 3.20: Method allowing to find a triggerable rule.

3.5.3.2 Coordination Module

The lack of autonomy, the heterogeneous technologies, and the single point of failure problems, have manifested the importance of considering the distributed approach. Distribution provide the possibility to overcome the aforementioned limitations through making multiple interconnected peers responsible for the system operation. Moreover, by considering the distributed approach many other advantages could be provided such as the possibility to add/remove some peers and the easy isolation/recovery of failures.

Generally, peers of distributed reconfigurable systems have partial, limited view of the system. Hence, there is a need to provide them with the ability to coordinate in order to perform some tasks. Figure 3.21 shows an example of a distributed system composed of peers where each group has a partial view of the system: c_{51} views c_{52} and c_{53}, while c_{13} views $c_{11}, c_{12}, c_{14}, c_{71}$, and c_{44}.

The coordination between peers is not an easy task to perform. This is why, a coordination module CM is added to the framework.

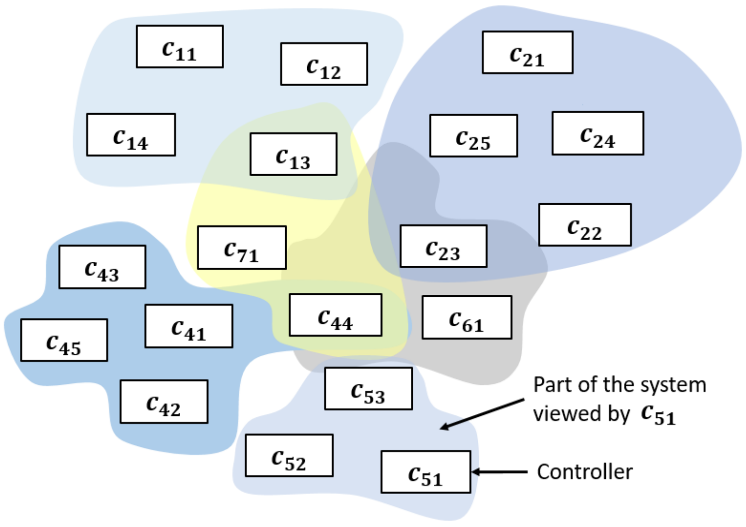


Figure 3.21: Distributed system scheme example.

Composition of Coordination Module

As we are interested in context-aware distributed reconfigurable systems, and as a context is defined as a system configuration, coordination processes, for this framework, are devoted to reach a consensus of a coherent reconfiguration process. Hence, the task of CM is performed based on a matrix, denoted by cm_{matrix}, defining all possible configurations that could take place between the software instances of the part of the system viewed by the current instance.

$$cm_{matrix_{m,n}} = \begin{pmatrix} I_{1,1} & I_{1,2} & \cdots & I_{1,n} \\ I_{2,1} & I_{2,2} & \cdots & I_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ I_{m,1} & I_{m,2} & \cdots & I_{m,n} \end{pmatrix}$$

where n represents the peers existing in the part of the system in which the current framework instance is used, and m represents the partial configurations that could take place in the considered part of the system, $I_{i,j}$ is a set of instructions/data to be used as configuration i of the peer j, such that $i, j \in \mathbb{N}$.

CM helps the controller of CCL to coordinate with other controllers in other peers of the system. This coordination/communication is granted by messaging through MM.

Dynamics of Coordination Module

The dynamic behavior of CM is presented through the statechart diagram depicted in Figure 3.22. The operation of CM starts when a coordination request, denoted by $CoordReq$, sent by the controller is received. Then, based on the defined coordination matrix, it searches all possible partial configurations and returns reply to controller.

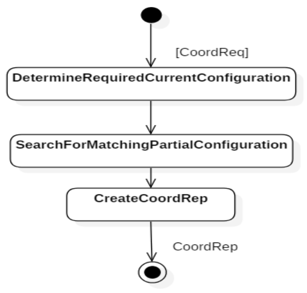

[CoordReq]
DetermineRequiredCurrentConfiguration
SearchForMatchingPartialConfiguration
CreateCoordRep
CoordRep

Figure 3.22: Coordination module statechart diagram.

3.5.3.3 Security Module

While significant advantages can be achieved thanks to its concepts, distribution paradigm gives rise to different challenges such as the security issues. In fact, distributed peers may need to collaborate with each other in multiple tasks. Some of these tasks could require the exchange of important information that should be secured (see Figure 3.23). Major security requirements that are often required in distributed system are the insurance of confidentiality and integrity.

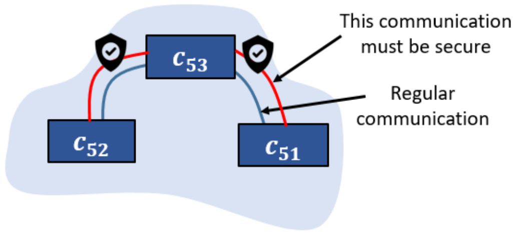


Figure 3.23: Secure communication for distributed peers.

For this, we propose this module SM to be a container of security tools that developers could use, extend, adapt in their applications in order to fit the considered use cases requirements. In this thesis, we propose a security protocol devoted to allow a secure communication between peers based on blockchain technology. The protocol is presented in Chapter 5.

Composition of Security Module

As mentioned above, the security module is devoted to store security techniques. Developers could add, for example, the required encryption methods or authentication tools. The role of this module is providing the infrastructure that allows the addition of techniques and the proper communication with the controller. Figure 3.24 shows a schematic presentation of the module components.

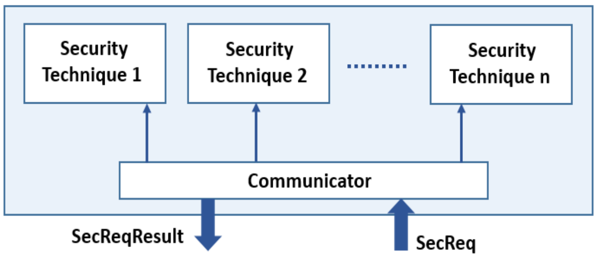


Figure 3.24: Security module scheme.

Dynamics of Security Module

The behavior of SM depends on the adopted security technique. But if we try to cover all steps required by the chosen technique in a submachine state, then we can simply resume the behavior in extracting data subject to security before starting the process and a submachine state including all necessary states (see Figure 3.25).

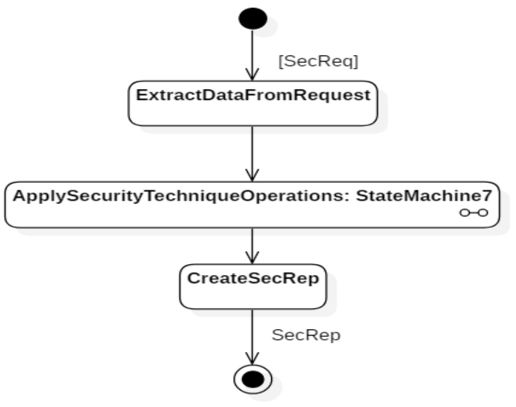


Figure 3.25: Security module statechart diagram.

3.5.3.4 Functional Module

As mentioned previously, the proposed framework is dedicated to the development of context-aware reconfigurable applications of distributed systems. Context-awareness and self-reconfiguration abilities of applications are coming from the fact that, based on the sensed/metered input data, the controller can automatically lead changes. The automated reconfigurations imply the modification or switch of the running functionality, which are defined as services in the framework (to be detailed in next sections).

Leading automated self-reconfiguration is eligible since it promotes applications adaptability and autonomy. However, the loading/unloading of a set of functionalities may give rise to some errors and/or problems. In fact, the relationships between functionalities need to be respected, especially, dependencies and inclusion/exclusion relationships. It is needed to provide a way to ensure that the addition/removal/change of functionalities does not cause any logical requirements violation or malfunctions.

To avoid this, we introduce the functional module FM to be responsible for functional constraints checking whenever reconfigurations need to take place.

Composition of Functional Module

In order to accomplish its task, this module FM relies on two types of functional constraints: (1) coherence constraint, and (2) precedence constraint.

The coherence constraint represents the ability of functionalities to operate together in harmony. In fact, with the multi-services that an application can provide, contradiction and conflict relationships could take place between some of the functionalities, such that the execution of the one(s) implies the suspension/cessation of the other(s). Hence, it is important to define the relation between functionalities in order to guarantee safe and coherent reconfiguration processes. For this, a Functional Exclusion Matrix, denoted by FEM, is introduced to specify these relations. FEM is given by

$$FEM[i][j] = \begin{cases} 1 & s_i \textit{ excludes } s_j. \\ 0 & \textit{otherwise.} \end{cases} \tag{3.9}$$

FEM is a square matrix of order n such that $i, j \in (1, ..., n)$, where n is the number of services of the considered application (stored in the services layer SL) and s_i is the i^{th} service of SL. s_i *excludes* s_j means that the operation of the service s_i implies the suspension of the service s_j.

The precedence constraint represents the logical order of the operation of a set of functionalities (i.e., services) or to all of them. In fact, for the operation of a set of services, it is required to use the outputs (e.g. signal, data, specific processing result, etc.) of other services. Thus, the precedence order should be preserved. To specify this precedence relationship constraint, a Functional Precedence Array, denoted by FPA, is defined in order to, for a given service s_i, it defines a set of pairs of its predecessors.

For a given service s_i, the search for predecessors is performed by:

1. Read the corresponding set, denoted by cs_i, from FPA,

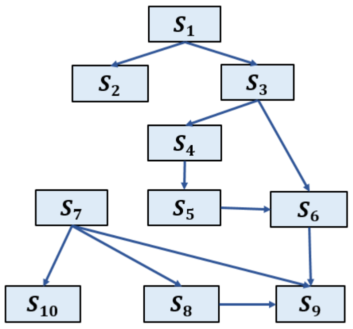


Figure 3.26: Services precedence relationship example.

2. Save the found cs_i,

3. Repeat steps (1) and (2) for each element in cs_i. The loop ends when cs_i is equal to the empty set.

4. Return a set of pairs (s_p, s_q) determining the order of operation, i.e., s_p must operate before s_q.

For example, let us consider the services mentioned in Figure 3.26, the corresponding FPA is given by

FPA:

s_1	$\emptyset$
s_2	$\{s_1\}$
s_3	$\{s_1\}$
s_4	$\{s_3\}$
s_5	$\{s_4\}$
s_6	$\{s_5, s_3\}$
s_7	$\emptyset$
s_8	$\{s_7\}$
s_9	$\{s_6, s_7, s_8\}$
s_{10}	$\{s_7\}$

The search for predecessors of s_6 gives $FPA[s_6] = \{s_5, s_3\}$, $FPA[s_5] = \{s_4\}$, $FPA[s_4] = \{s_3\}$, $FPA[s_3] = \{s_1\}$, $FPA[s_1] = \emptyset$. Hence, the predecessors of s_6 are $\{(s_1, s_3), (s_3, s_4), (s_5, s_6), (s_3, s_6), (s_4, s_5)\}$.

Dynamics of Functional Module

Each time the functional module receives a request from the controller, denoted by $FuncReq$, a check to the FEM and FPA is made by FM and the results are returned to the controller. The behavior can be modeled as depicted in Figure 3.27

A part of the implementation of the exclusion checking is presented in Figure 3.28.

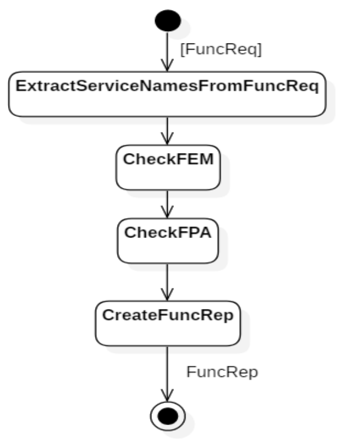


Figure 3.27: Functional module statechart diagram.

3.5.3.5 Timing Module

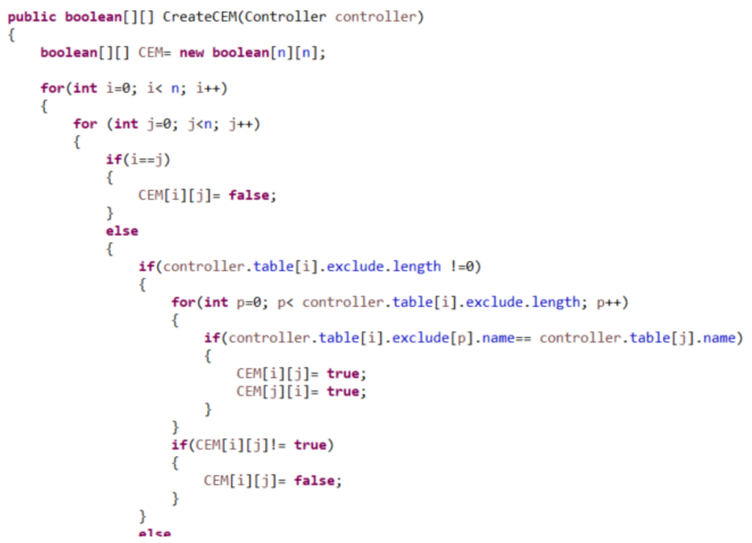
By the increasing functionalities that must be fulfilled, modern reconfigurable systems become more constrained with time, especially that event-driven reconfigurations must also be successfully handled. In order to ensure a timing efficiency, a timing module TM is proposed to be one of the mainstays of the framework.

This module TM has the role of temporal behavior analyzer of services to be applied. Analysis are conducted with the aim of checking the time feasibility in case of leading reconfiguration processes, i.e., whether tasks of the functionalities added due a reconfiguration respect their deadlines. To achieve this, the schedulability of tasks issued from reconfiguration processes must be conducted. Schedulability is defined as the ability of tasks to meet their deadlines.

Composition of Timing Module

This module reflects an estimation of a set of tasks that are schedulable and executable by the central processing unit (CPU) of a specific operating system running under a specific hardware. The tasks properties, such as the capacity (i.e., how many time slots necessary for execution), the priority, the period, the deadline, etc., must be determined by experts after a study of the considered system who muster all details pertained to it. Then, a set of schedulability algorithms that are based on the famous and widely used real-time scheduling protocols can be included in this module to help in evaluating the schedulability and feasibility.

Many real-time scheduling protocols are available and each is devoted for specific type of tasks [Fkaier *et al.* 2016b] periodic, non-periodic, and whether there is a resources sharing. For periodic tasks, there are protocols that handle tasks with static priority such as the Rate Monotonic or the Deadline Monotonic, or also tasks with dynamic priority such as the Earliest Deadline First or the Least Laxity First. Non-periodic tasks are generally handled with servers such as the Polling



```
public boolean[][] CreateCEM(Controller controller)
{
    boolean[][] CEM= new boolean[n][n];

    for(int i=0; i< n; i++)
    {
        for (int j=0; j<n; j++)
        {
            if(i==j)
            {
                CEM[i][j]= false;
            }
            else
            {
                if(controller.table[i].exclude.length !=0)
                {
                    for(int p=0; p< controller.table[i].exclude.length; p++)
                    {
                        if(controller.table[i].exclude[p].name== controller.table[j].name)
                        {
                            CEM[i][j]= true;
                            CEM[j][i]= true;
                        }
                    }
                    if(CEM[i][j]!= true)
                    {
                        CEM[i][j]= false;
                    }
                }
                else
```

Figure 3.28: Part of the implementation of the exclusion checking.

Server, Deferrable Server, and Sporadic Server. Sharing Resources is handled with protocols such as the Priority Ceiling Protocol or the Priority Inheritance Protocol.

In this module, an example of a scheduling according to Rate Monotonic for periodic tasks along with a Polling Server for non-periodic tasks is provided. This case is considered due to its simplicity, other schedulability tests can be added by users/developers since this module contains a representation of tasks.

In case of addition of a service as an aperiodic task, the use of the Polling Server is performed through a periodic task devoted to the execution of the aperiodic tasks. The server is characterized by a capacity, denoted by C_s, and a period, denoted by P_s. The aperiodic tasks are executed each period P_s for a maximum number of time slots equal to C_s. If there is no request waiting for the server, the task is suspended until the next period [Buttazzo 2011].

The condition for an isolated task (i.e., no other aperiodic task is waiting, denoted by J_a, arriving at an instant A_a, having a worst case execution time C_a, and a deadline D_a) is as follows:

- If $C_a \leq C_s$, the execution of J_a is finished at maximum after $2P_s$, and the acceptance condition is

$$2P_s \leq D_a \tag{3.10}$$

- Generally we do not have $C_a \leq C_s$, the execution of J_a is finished at $P_s + \lceil C_a/C_s \rceil P_s$, and the acceptance condition is

$$P_s + \left\lceil \frac{C_a}{C_s} \right\rceil P_s \leq D_a \tag{3.11}$$

Dynamics of Timing Module

This module is playing the role of: (1) taking the functionalities to be processed (services) and based on the estimation of experts/engineers about the properties (i.e., capacity, priority, period, deadline) of each functionality along with the available scheduler/hardware, made the proper calculations and return it to the controller.

Figure 3.29 depicts the behavior of the timing module. Two main scenarios can take place, the first when the request from the controller requires the addition of a new combination of services. The second is when the request from the controller requires the addition of one service as a non-periodic service.

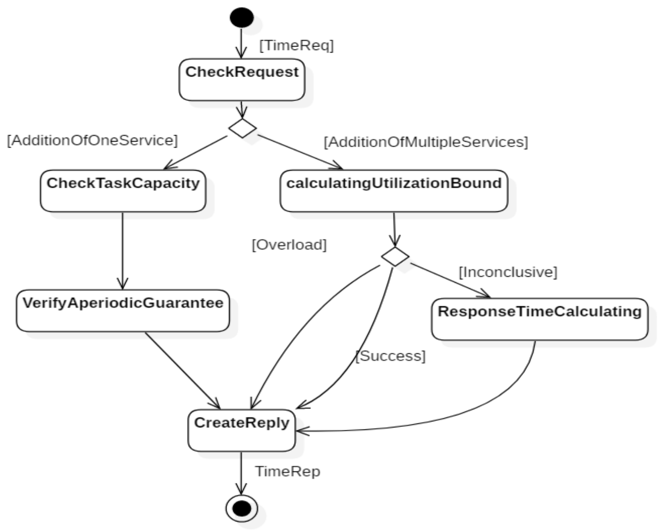
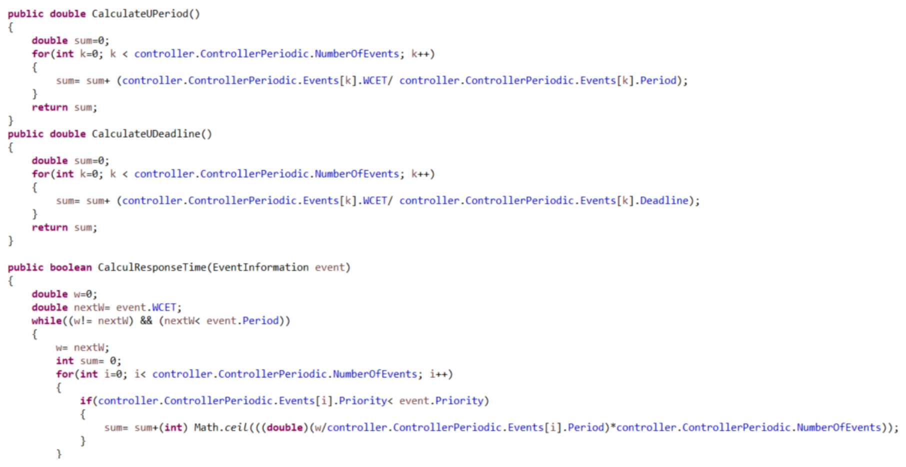


Figure 3.29: Timing module statechart diagram.

The implementation of some of the helper methods of the time-related analysis are presented in Figure 3.30.



```
public double CalculateUPeriod()
{
    double sum=0;
    for(int k=0; k < controller.ControllerPeriodic.NumberOfEvents; k++)
    {
        sum= sum+ (controller.ControllerPeriodic.Events[k].WCET/ controller.ControllerPeriodic.Events[k].Period);
    }
    return sum;
}
public double CalculateUDeadline()
{
    double sum=0;
    for(int k=0; k < controller.ControllerPeriodic.NumberOfEvents; k++)
    {
        sum= sum+ (controller.ControllerPeriodic.Events[k].WCET/ controller.ControllerPeriodic.Events[k].Deadline);
    }
    return sum;
}

public boolean CalculResponseTime(EventInformation event)
{
    double w=0;
    double nextW= event.WCET;
    while((w!= nextW) && (nextW< event.Period))
    {
        w= nextW;
        int sum= 0;
        for(int i=0; i< controller.ControllerPeriodic.NumberOfEvents; i++)
        {
            if(controller.ControllerPeriodic.Events[i].Priority< event.Priority)
            {
                sum= sum+(int) Math.ceil(((double)(w/controller.ControllerPeriodic.Events[i].Period)*controller.ControllerPeriodic.NumberOfEvents));
            }
        }
```

Figure 3.30: Timing module helper methods.

3.5.3.6 Controller Module

The controller C is the master element in the architecture. It is the component containing the main logic of applications. It performs the control task with the help of the other CCL modules as well as the other layers. Based on the received inputs it decides the modification level that should take place.

Composition of Controller Module

In order to accomplish its task, the controller needs to have three main mechanisms: (1) interaction with the rest of modules either from the same layer (i.e., CCL) or from other layers (i.e., RL and SL), (2) view and parameterization of the service layer (i.e., how many services in SL, properties of services, etc.), and (3) view and parameterization of the reconfiguration layer, specifically the input module IM and the output module OM (i.e., table of inputs containing their properties and table of outputs containing their properties).

To facilitate the handling of these mechanisms, a set of internal interaction requests, denoted by Req, a set of internal interaction replies, denoted by Rep, as well as a set of parameterization lists, denoted by $Parameters$, are defined such that,

$$Req = (IntReq, CoordReq, SecReq, FuncReq, TimeReq). \tag{3.12}$$

As mentioned earlier, $IntReq$ denotes the request to the artificial intelligence module, $CoordReq$ denotes the request to the coordination module, $SecReq$ denotes the request to the security module, $FuncReq$ denotes the request to the functional module, and $TimeReq$ denotes the request to the timing module. These requests can be used multiple times, as much as it is required according to the use case. The structure of each request is mentioned in Table 3.1.

Table 3.1: Structure of the requests used by the controller.

Request Name	Request structure
IntReq	*< list of premises > | < execution mode >*
CoordReq	*< list of services names >*
SecReq	*< type of the technique > | < data subject to security > | < data not subject to security >*
FuncReq	*< list of services names >*
TimeReq	*< list of services names > | < list of services properties >*

After finishing its processing, each module should return a reply to the controller containing the obtained result. The set of all replies usable by the controller is defined as;

$$Rep = (IntRep, CoordRep, SecRep, FuncRep, TimeRep). \tag{3.13}$$

As mentioned is previous sections, $IntRep$ denotes the reply of the artificial intelligence module, $CoordRep$ denotes the reply of the coordination module, $SecRep$ denotes the reply of the security module, $FuncRep$ denotes the reply of the functional module, and $TimeRep$ denotes the reply of the timing module.

The structure of each reply is depicted in Table 3.2.

Table 3.2: Structure of the replies returned to the controller.

Reply Name	**Reply structure**
$IntRep$	$< list\ of\ conclusions >$
$CoordRep$	$< list\ of\ possible\ configurations >$
$SecRep$	$< secured\ data > \mid < operational\ data >$
$FuncRep$	$< list\ of\ exclusion\ services > \mid < ordered\ list\ of\ services >$
$TimeRep$	$< schedulable >$

Figure 3.31 depicts the classes used to allow the interaction mechanism between the controller and the rest of the modules.

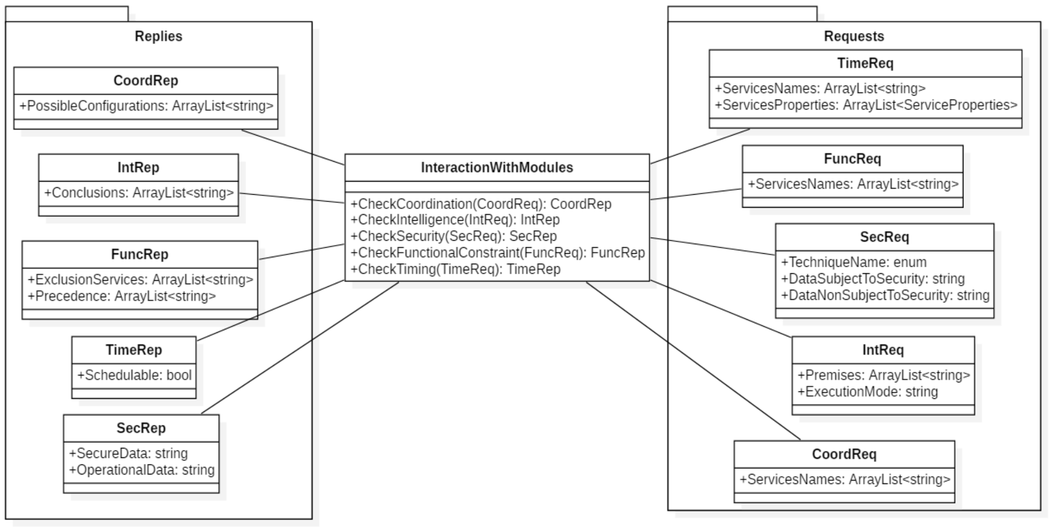

Replies
CoordRep
+PossibleConfigurations: ArrayList<string>
IntRep
+Conclusions: ArrayList<string>
FuncRep
+ExclusionServices: ArrayList<string>
+Precedence: ArrayList<string>
TimeRep
+Schedulable: bool
SecRep
+SecureData: string
+OperationalData: string
InteractionWithModules
+CheckCoordination(CoordReq): CoordRep
+CheckIntelligence(IntReq): IntRep
+CheckSecurity(SecReq): SecRep
+CheckFunctionalConstraint(FuncReq): FuncRep
+CheckTiming(TimeReq): TimeRep
Requests
TimeReq
+ServicesNames: ArrayList<string>
+ServicesProperties: ArrayList<ServiceProperties>
FuncReq
+ServicesNames: ArrayList<string>
SecReq
+TechniqueName: enum
+DataSubjectToSecurity: string
+DataNonSubjectToSecurity: string
IntReq
+Premises: ArrayList<string>
+ExecutionMode: string
CoordReq
+ServicesNames: ArrayList<string>

Figure 3.31: Interaction classes.

The controller interacts with the inputs module IM with two ways: (1) it receives data from IM periodically, or (2) it demands data from IM whenever necessary (e.g., after a reception of a message).

In addition, the controller "works" with the other layers through a set of general parameters defined as

$$Parameters = (T_{Services}, T_{CurrentServices}, T_{SecurityTechniques}, T_{Inputs}, T_{Outputs}). \tag{3.14}$$

where $T_{Services}$ designs a table describing the services of the service layer SL and contains the service identifier (i.e., name) and its timing properties, $T_{CurrentServices}$

designs a table containing the currently executed services, $T_{SecurityTechniques}$ designs a table enumerating the available security techniques provided by the security module, T_{Inputs} (*resp.* $T_{Outputs}$)designs a table listing the considered inputs (*resp.* outputs) of the system and their properties

Dynamics of Controller Module

The behavior of the controller cannot be limited to a specific statechart diagram because the logic is dependent on the applications implementations. However, to facilitate the task for developers, four main behaviors are provided: a collaborative behavior, a supervision behavior, a reconfiguration behavior, and an emergency behavior. The details of the four behaviors are described as follows:

1. **Collaborative Behavior**: this behavior is executed when the controller receive a message from other agents/peers/applications in the system that the current application behaves within. The collaborative behavior is reached in two scenarios: either a reconfiguration is taking place in other peers and it is required that the current peer reconfigure accordingly, so a negotiation can take place before decisions (see Figure 3.32), or it is required to update the information of the current peer (see Figure 3.33).

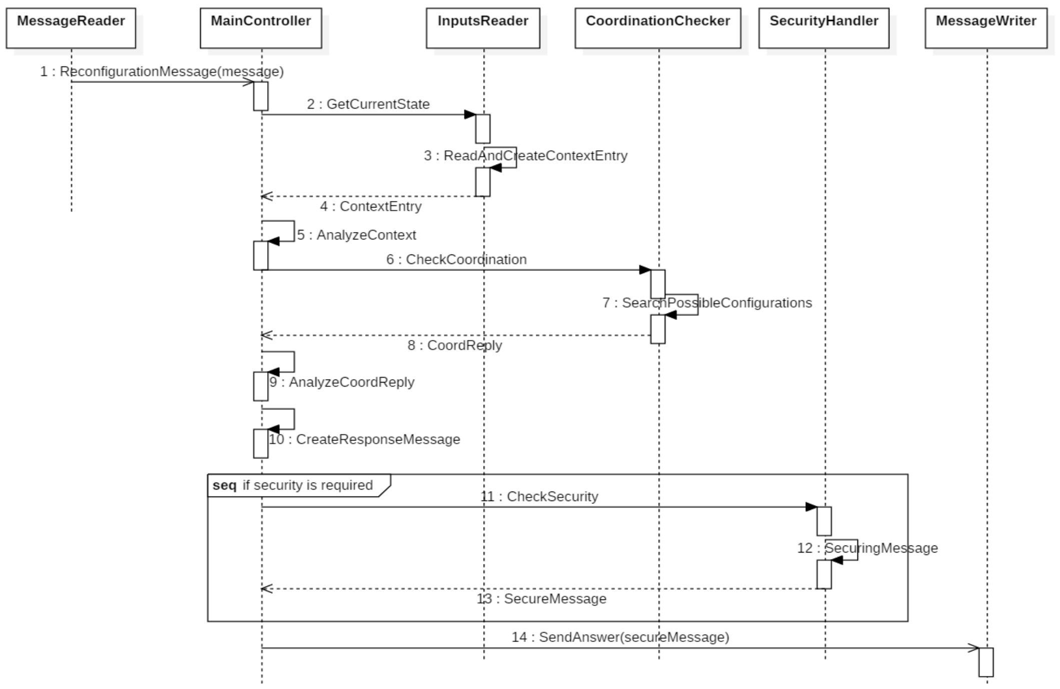


Figure 3.32: Collaborative behavior in case of negotiation.

2. **Supervision Behavior**: this behavior is executed most often and it is considered as the normal operational behavior where the application is executing

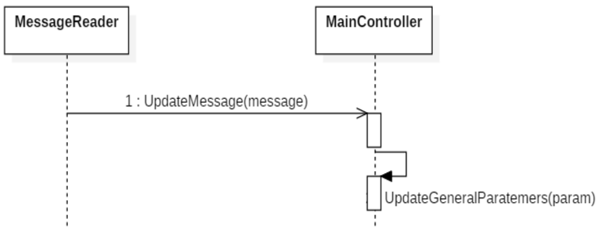


Figure 3.33: Collaborative behavior in case of update.

a set of services. Figure 3.34 depicts an example of supervision where no reconfiguration and no collaboration are required.

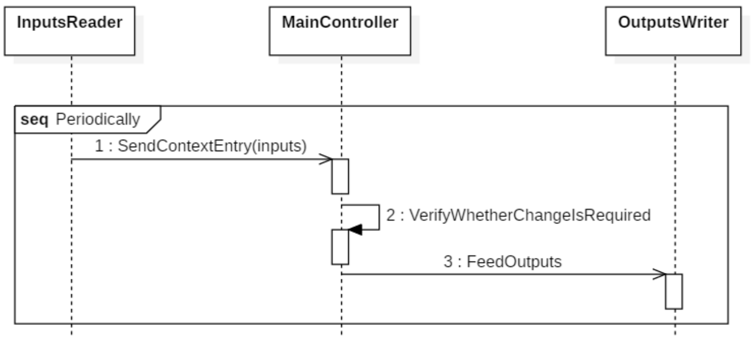


Figure 3.34: Supervision behavior.

3. **Reconfiguration Behavior**: this behavior is executed when the received input data and/or recommendations from the inputs module IM indicates that a context change is happening and an adaptation/reconfiguration must be conducted. Also this behavior can take place when the result of a negotiation process indicates that a reconfiguration is necessary. Figure 3.35 depicts an example of the main process of reconfiguration (updating other peers in the system and using security can be added).

4. **Emergency Behavior**: this behavior is executed when the controller receives alert and/or emergency events from the inputs module. The behavior is very similar to the reconfiguration one, except that some steps can be skipped such as the search for optimal configurations. In fact, in urgent situations it is important to eliminate the big risks first by finding any operational configuration then optimization can take place.

3.5.4 Service Layer

This layer contains the services of the system. A service, denoted by s_i, is considered as a functionality or an operation mode to be done by an application, i.e., the set of methods responsible for accomplishing a system activity. These methods are hold

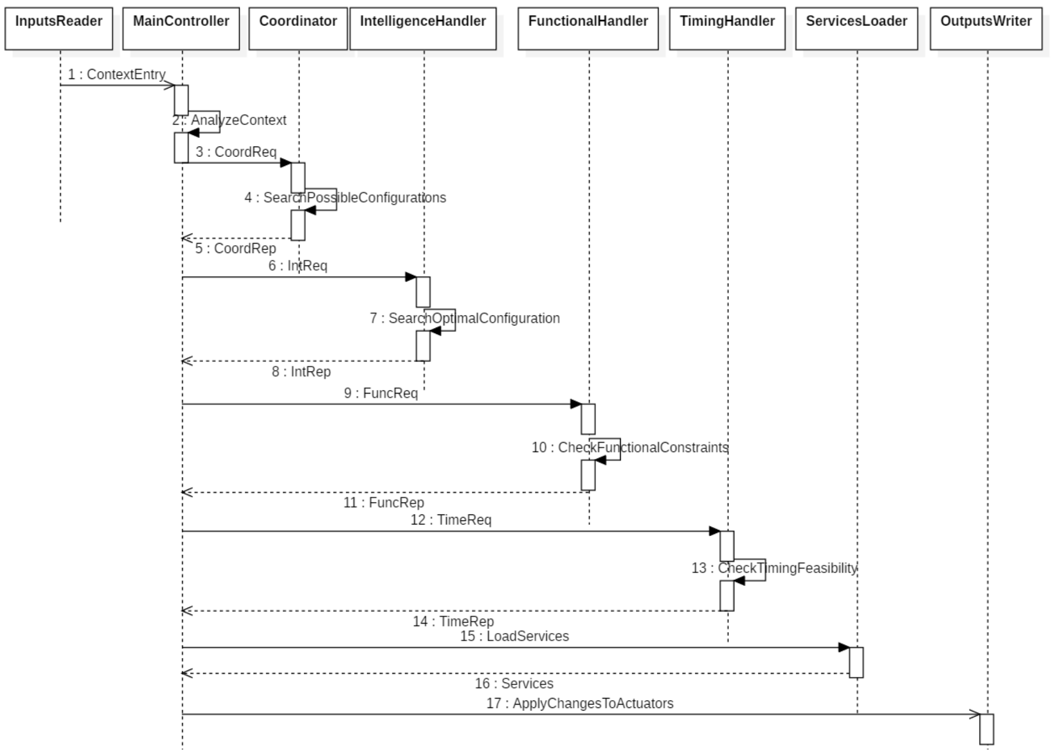


Figure 3.35: Reconfiguration behavior.

in separate elements (i.e., the services) in the purpose of allowing more clarity and avoiding possible conflicts arising from multiple activities of applications.

3.6 Running Example

In this section, we tackle a use case of the proposed framework applied to a formal example of an airport Baggage Handling System (BHS). We show how the framework features facilitate the software development, i.e., how to make developers focus on the specific requirements of the use case and not the infrastructure necessary for it (UML models, architecture, generic code). It is worth noting that, a second case study of microgrids software development is elaborated and reported in [Fkaier. *et al.* 2021b].

3.6.1 Case Study Presentation

Before demonstrating the details of the application development, it is necessary to introduce the considered case first. In the following, a presentation of the system and its requirements are provided.

3.6.1.1 Airport Baggage Handling System

BHS is one example of the goods transportation systems that have known a great expansion during the last years. Intelligence and awareness are the major characteristics that have been studied in order to be introduced into BHS [Nakagawa *et al.* 2014], [Sørensen *et al.* 2019]. In this context, we choose to apply the proposed framework concepts on BHS as a conveying system having to be more intelligent.

Composition: BHS components contains generally three parts: (1) computing unit processing the logic, (2) inputs/outputs elements ensuring the communication with the outer environment, and (3) a software application holding the operational logic.

Services: The BHS functionalities are generally conveying luggage (forward and backward directions), tracking bags, sorting, etc.

The layout of the considered BHS in this case study is depicted in Figure 3.36.

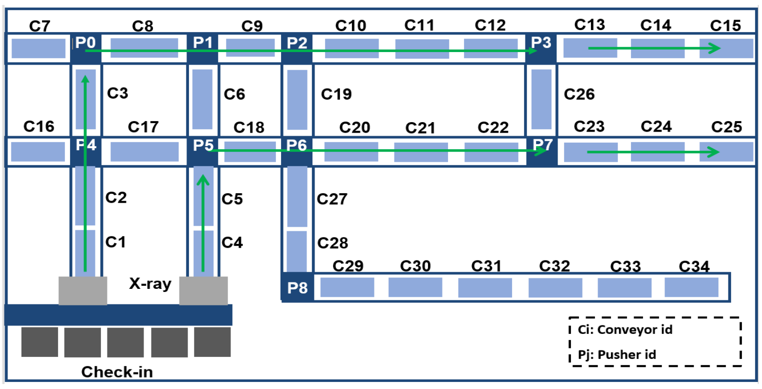


Figure 3.36: Baggage handling system layout example.

3.6.1.2 BHS Application Requirements

By the increasing of air transport, airports become more and more challenged with the quality of service especially in terms of reducing passengers waiting times including the baggage waiting. To cope with this challenge, modern and future baggage handling systems must provide better control and management solutions. As the BHS is an important subsystem in airports, developing intelligent context-aware software applications of conveyors may offer promising results.

BHSs have the need to smart controlling software applications that allow:

1. self-repair and extensibility with minimal or without human intervention,
2. energy saving through powering the conveyors down when no baggage is detected,
3. automatic speed adaptation in order to minimize the passengers waiting times,

4. behavior optimization mechanisms through using optimized routing (i.e., application that could reason taking into account the system layout including the shared and critical routes that might induce baggage bottleneck).

3.6.2 Case Study Development with the Proposed Framework

We show in the following the development of context-awareness, self-repair, and optimized route finding using the framework modules.

3.6.2.1 Application Model with UML

The first step to be done, and after analyzing the requirements of a BHS, is the design decisions of the software application. Determining the UML model of applications is very important in the development life-cycle since it allows to practically represent its structure and functionalities. The different diagrams offered by UML provide the opportunity to design and analyze different scopes of the application, e.g., the behavior can be modeled in detail with statechart diagrams, the high-level structure with the component diagrams, the deep structure with class diagrams, and the interactions with the sequence diagrams.

This step is facilitated thanks to framework model defined at the beginning of this chapter. The analysis of the target application leads us to define a set of services (functionalities), a set of input items, and a set of output items. For simplicity reasons, in this case study we consider four services that a BHS can do: track bags, move belts forward, move belts backward, and halt (sleep mode). Hence, the general component diagram is given as depicted in Figure 3.37.

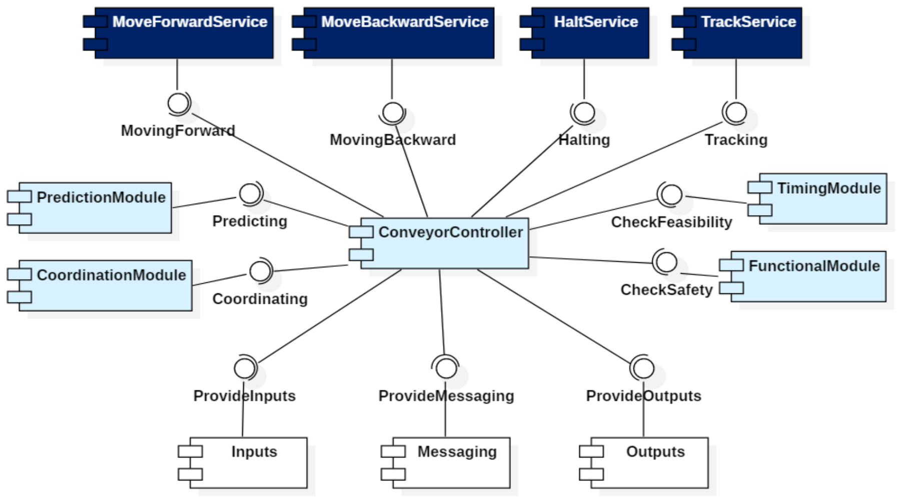


Figure 3.37: Component diagram of the BHS application model.

The four services are modeled with the four components *MoveForwardService*, *MoveBackwardService*, *HaltService*, and *TrackService*. The roles of the rest of modules will be as follows (it is worth noting that security is not considered in this case study):

- The timing module contains the timing feasibility analysis especially for the execution of reconfiguration events.
- The functional module contains the functional constraints under which the services of the conveyor have to work: exclusion relation between (move forward and move backward), (move forward and halt), (move backward and halt). Hence, the functional safety relationships are carefully defined.
- The coordination module contains the coordination matrix enabling the collaboration between the different devices (such as pushers) and conveyors in the BHS: the definition of the possible configurations of the system including the state of each device on it.
- The artificial intelligence module contains a knowledge base allowing a prediction task used to optimize the baggage routing.

The modules models are based on the models provided in previous sections. Whenever some tasks of the framework modules are not needed, the models can be modified accordingly. Otherwise, the application models will be similar to those of the framework. For example, the class diagram of the Coordination Module (CM) remains as depicted in Figure 3.38 and Figure 3.39.

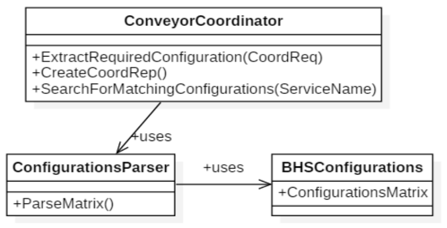


Figure 3.38: Class diagram of the coordination module.

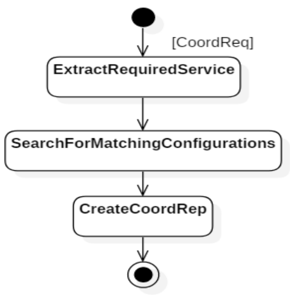


Figure 3.39: Statechart diagram of the coordination module.

It is also substantial in this modeling step to define the models of some elements necessary for the use of the framework: we need to model the context (necessary to enable the context-awareness feature of the Reconfiguration Layer). Moreover, we need to define the purpose of the intelligence module through determining the general rules upon which the inference builds. Operational constraints also need to be analyzed.

Context-Awareness: To enable the BHS with context awareness, and according to the framework first layer, it is required to define the ontology-based context model. Figure 3.40 depicts the defined context model of the most important elements that impact the context. The model is edited with the graphical ontology editor OWLGrEd Version 1.6.10 [1]. It is worth noting that this model can be enriched with more ontologies (and this an advantage of the selected approach, i.e., ontology context models).

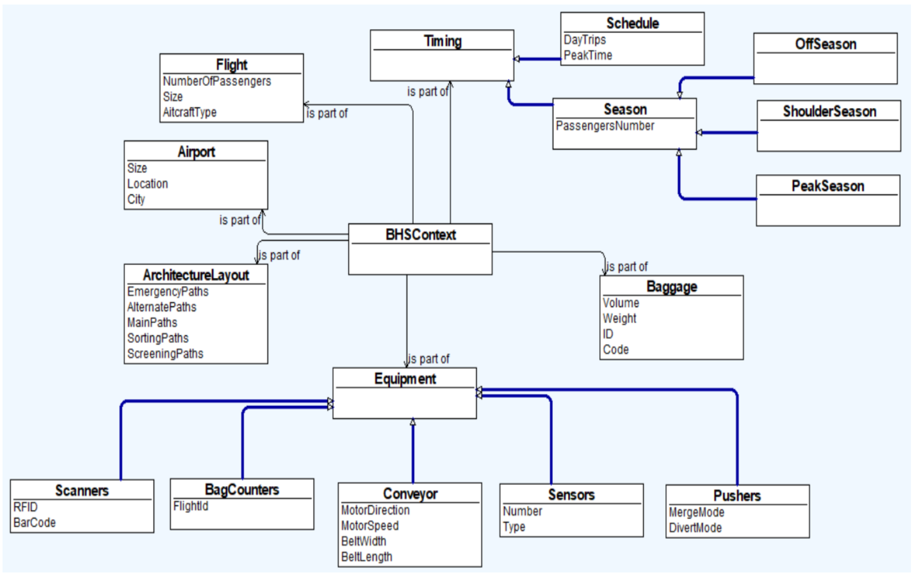


Figure 3.40: Baggage handling system context model with OWL.

For example, the context information depends on the physical layout of the BHS, this is why we define an ontology called *ArchitectureLayout*. It is important to the controller to know the available paths and their types (i.e., main, emergency, alternate, etc.) so that it can lead the proper coordination and routing processes.

Another example, is the one of *Conveyor* ontology which is a sort of *Equipment* substantial to the BHS. The properties of conveyors such as the motor speeds, directions, as well as the belt width, length, and others, are all important information that could impact the performance of the system, subsequently the software functionalities.

Timing ontology and its specialization *Schedule* and *Season* are very important entities that impact the quality of service and the performance of the whole baggage

[1]http://owlgred.lumii.lv/get_started

handling system. For example, a flight delay is a fact that the BHS must be aware of in order to behave properly and efficiently.

Routing and Prediction: To enable the conveyor system with proper prediction feature for better routing, it is necessary first to make it understand the architectural layout of the BHS. More importantly it is required to define the initial routes, the back-up routes, and the bridging conveyors. In this example, the initial routes are those marked with the green arrow in Figure 3.36 where the initial routes consists in conveyors running in forward direction. For simplicity reasons, two initial routes are defined: Route 1= (C1, C2, C3, C8, C9, C10, C11, C12, C13, C14, C15) and Route 2= (C4, C5, C18, C20, C21, C22, C23, C24, C25), one back-up route: Route 3= (C4, C5, C18, C27, C28, C29, C30, C31, C32, C33, C34), and a set of bridging conveyors: {C26, C19, C6, C3, C17}.

Constraints: It is in this step to design and analyze the probable constraints such as the timing and functional constraints. The Object Constraint Language (OCL) can be used to elucidate the constraints. For example, the services are constrained with coherence relationships that could be expressed according to the following invariant:

Context CoherenceRelation **inv:**
self. ForwardMove.isActive = true **implies self.**BackwardMove.isActive= false

More importantly, the controller which is the major element of the architecture and which its logic and behavior are left to the developers to perform, needs to be carefully modeled in this step. Examples of probable behaviors are provided (among the framework models) to help in developing the controller. Statechart and class diagrams of the final controller must be defined in order to facilitate its implementation later. In this example, we model the statechart diagram of the controller in case of reconfiguration in the context of a self-repair operation. The diagram is provided in Figure 3.41.

It is worth noting that the development of the controller logic is a delicate task that must produce reliable outputs, especially when it comes to critical use cases (where errors may induce big losses). For this, a modeling methodology is introduced in this research work that uses formal verification techniques to ensure model correctness. Details of this methodology are defined in Chapter 4. For the current chapter, we content with presenting the UML model before implementation but the complete modeling process contains also formal verification and other guarantees.

3.6.2.2 Scenario Example

The framework concepts are developed using Java programming language. In order to implement BHS applications, and after having idea about the framework mechanisms and services, developers need just to import the framework classes and start using its code. In the following, we present the scenario of reconfiguration and how the controller finds solutions.

A scenario of reconfiguration must take place when a failure (mechanical or

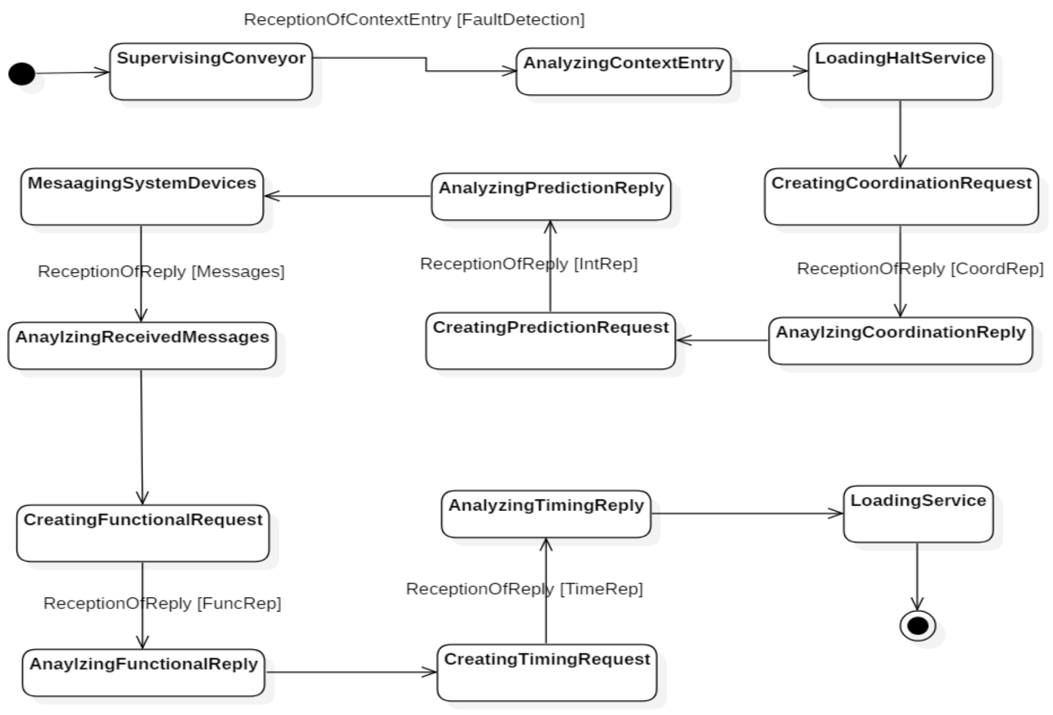

ReceptionOfContextEntry [FaultDetection]
SupervisingConveyor
AnalyzingContextEntry
LoadingHaltService
MesaagingSystemDevices
AnalyzingPredictionReply
CreatingCoordinationRequest
ReceptionOfReply [Messages]
ReceptionOfReply [IntRep]
ReceptionOfReply [CoordRep]
AnaylzingReceivedMessages
CreatingPredictionRequest
AnaylzingCoordinationReply
CreatingFunctionalRequest
AnalyzingTimingReply
LoadingService
ReceptionOfReply [FuncRep]
ReceptionOfReply [TimeRep]
AnaylzingFunctionalReply
CreatingTimingRequest

Figure 3.41: Controller statechart diagram.

electrical reasons) occurs in the conveyor C14 as depicted in Figure 3.42. Here, the application needs to take rapid measures to minimize the impact on the system throughput.

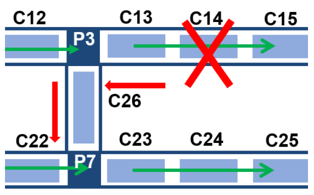


Figure 3.42: Conveyor C14 failure.

The failure information is detected by sensors of the conveyor C13. According to the Context Attributes Models Store (CAMS) defined in Table 3.3, specifically the attribute *Neighbor Activity Signal*, and using the context reasoning mechanism based on the Context Rules Store (CRS) defined in Table 3.4, specifically the rule R2, the input module IM sends a context entry to the controller containing recommendations to power-down the conveyor and reroute baggage.

The controller analyzes the context entry entry and starts to prepare for a reconfiguration. A reconfiguration decision should be taken efficiently through checking some conditions and constraints. In fact, as mentioned previously, a reconfiguration should be performed in coordination between the distributed elements of the BHS,

Table 3.3: Excerpt of the BHS CAMS.

Attribute Name	**Values**
Bag weight (kg)	[0, 24]-light, [25, 35]-medium, [36, 50]-heavy
Bag length (m)	[0, 0.4]-small, [0.41, 0.6]-medium, [0.61, 0.8]-big
Schedule	[00:00, 09:59]-regular, [10:00, 12:00]-peak, [12:01, 15:59]-regular, [16:00, 18:00]-peak, [18:01, 23:59]-regular
Neighbor activity signal	1-alive, 0-dead
Bag detection signal	1-bag exists, 0-no bag exists

Table 3.4: Excerpt of the BHS CRS.

Id	**Rules**
R1	(BagDetectionSignal is NoBagExist) → (Power-down conveyor).
R2	(NeighborActivitySignal is Dead) → (Power-down conveyor) and (reroute bags).
R3	(Schedule is PeakTime) → (Speed-up conveyors).
R4	(Flight is Delayed) and (Check-in is Completed) → (Update path state for routing).
R5	(Flight is Delayed) and (PlannedPath is Busy) → (Speed-up conveyors).
R6	(Flight is Delayed) and (PlannedPath is Busy) and (Season is Shoulder) → (Search alternate route).
R7	(Flight is Delayed) and (Season is Peak) → (Use emergency route).

therefore a coordination should take place. Then, the controller interacts with the coordination module CM to get possible configurations in which the conveyor C14 is out of service. Based on the coordination matrix, defined as depicted in Table 3.5, CM parses cm_{matrix} and returns $Config_8$ and $Config_9$ to the controller.

Table 3.5: Excerpt of the coordination matrix of CM.

	..	C13	C14	C15	C26	P3	..	C20	C21	C22	C27	C28	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..
$Config_7$	..	F	F	F	H	N	..	F	F	F	H	H	..
$Config_8$	..	B	H	H	F	A	..	F	F	F	H	H	..
$Config_9$	..	H	H	H	F	A	..	H	H	H	F	F	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..

In Table 3.5 F is the forward service, B is the backward service, H is the halt service, A is active state, and N is non-active state.

Having more than one possible configuration, the controller needs to choose the configuration ensuring better routing. For this, it interacts with AIM to predict probable behavior of each configuration. The failure information of C14 is considered as a fact and it is added in the fact base. The inference engine performs reasoning using the rule base depicted in Table 3.6. In the considered case, it is assumed that the two paths convey baggage for the same flight.

Table 3.6: Excerpt from the Rule Base of the AIM.

Id	Rules
R1	IF (C14 is in failure) AND (C12 is in forward) THEN (made C13 in backward).
R2	IF (C14 is in failure) AND (C12 is in forward) THEN (made C26 in forward).
R3	IF (C23 is in failure) THEN (made C26 in backward).
R4	IF (C26 is in forward) AND (C22 is in forward) THEN (bottelneck).
R5	IF (C26 is in forward) AND (C12 is in forward) THEN (bottelneck).
R6	IF (C5 is in forward) AND (C17 is in forward) THEN (bottelneck).
R7	IF (C18 is in forward) AND (C19 is in forward) THEN (bottelneck).

The AIM starts by checking in history whether such scenario was faced in the past and it finds that this scenario is new, then it starts triggering rules in order to find the path that does not lead to a congestion. Each possible configuration (i.e., $Config_8, Config_9$) is analyzed apart and the forward chaining is applied to check whether the considered configuration lead to bottlenecks.

Starting by $Config_8$, which has as facts *fact1: C26 is in forward* and *fact2: C22 is in forward*, so the rule R4 is triggered and we obtain a conclusion of possible bottleneck. Moving to $Config_9$, taking facts from the configurations mentioned in this route and triggering possible rules does not produce any congestion, therefore this configuration is selected to be better solution. AIM finishes by sending $Config_9$ to the controller which consists in stopping [C20..C22] and to run [C27..C34] on forward move.

The controller creates a functional request and sends it to FM to check the absence of incoherences. The Functional Exclusion Matrix (FEM) for the current case study is defined as depicted in Table 3.7. According to FEM, tracking bags and moving backward services can work together with no exclusion relationship. The Functional Precedence Array (FPA) is defined as depicted in Table 3.8, and it dictates that the tracking of bags should precede the backward service (i.e., acknowledgement of tracking service should be done before continuing with moving backward).

Table 3.7: Functional Exclusion Matrix (FEM).

	F	B	T	H
F	-	1	0	1
B	1	-	0	1
T	0	0	-	0
H	1	1	0	-

Table 3.8: Functional Precedence Array (FPA).

(T, F)	(T, B)	(T, H)

Finally, after having checked the different reconfiguration requirements, the controller sends a command related to the backward service to the outputs module to apply change in form of commands.

3.6.3 Discussions

In this section we highlight the contributions and originality of the framework through showing the efficiency of the context-awareness reasoning process and through discussing the use of the proposed framework in comparison to other frameworks.

3.6.3.1 Context Modeling and Reasoning Mechanism Performance

One of the challenges of modern smart applications is the resources consumption in terms of usage of computational time and physical resources such as the memory and energy. Therefore, providing a solution that can achieve the target with less resources consumption is always in demand.

The proposed context reasoning approach includes an intermediary step between the data formatting and the rules triggering concerned with comparing the sensed and current context values. A difference threshold must be defined to determine how much the difference is. The purpose of this step is to avoid unnecessary computations that are originated from the same or very near values that generally do not imply a context semantic change.

Let us consider the current context values given by $CC = \{35, 1, 1100\}$ and let us consider the four context measurements given by $V_1 = \{40, 1, 1200\}, V_2 = \{32, 1, 1020\}, V_3 = \{35, 1, 900\}, V_4 = \{35, 1, 1488\}\}$. The calculation of the difference between CC and $V_1, ..., V_4$ is presented in Figure 3.43.

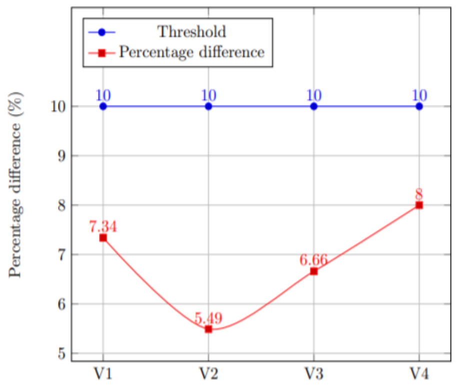


Figure 3.43: Comparison between the current and sensed values.

Figure 3.43 shows that the threshold is not reached with the four samples, therefore this step allows to save computational time four times.

3.6.3.2 Discussion about the use of the Proposed Framework

In this section, a discussion about the framework is tackled through highlighting its benefits and also mentioning what is missing and what needs improvement.

The framework consists in a programming tool of context-aware reconfigurable applications having an organized architecture. The clear architecture offers a complete view of the data flow from contextual data to the higher functionalities. The second layer of the framework includes a set of modules addressing the most prevalent requirements and constraints of smart systems. The contributions of the framework are implemented in a software tool using Java programming language as a proof of concept that helped to explore the framework assets. An example of a BHS also allowed to show in practice how the framework is fruitful.

Compared to the existing context-awareness frameworks, that are presented in detail in Chapter 2, our framework allows to develop not only context-awareness feature but also additional features such as the intelligence, timing, security, and functional safety needs. Unlike the majority of existing ones, our work provides the UML models of the framework which are very important to engineers, that have now clear structure (i.e., UML class and component diagrams). Secure coordination between distributed peers is also an original contribution tied to our framework since most of the existing works do not provide solutions to the coordination in distributed reconfigurable systems and the minority that consider it do not tackle security issues.

Despite its advantages, there is still some scopes that need to be improved. For example, the software tool is still in a proof of concept level and needs to be more robust/mature in order to be used in more complex real-life systems. This could be achieved through making more extensive tests (such as unit tests). Different case studies may also help to prove more the suitability of the framework.

3.7 Conclusions

In this chapter, we have introduced a software framework for the development of context-aware reconfigurable controlling software applications of distributed systems. First, a meta-model is defined to be the base of the framework concepts. Then, the concepts of the framework are introduced.

The framework consists in three layers representing the different functional levels of an application: a Reconfiguration Layer responsible for the interaction of applications with their environment especially the context-awareness and the initiation of reconfigurations, a Context Control Layer responsible for checking conditions of any reconfiguration procedure, and a Service Layer holding the functionalities of applications. The layers and modules have allowed to achieve the desired loose coupling between the different application scopes.

The concepts of the framework are applied to a formal case study of a BHS through which the benefit of using a programming model is exhibited. It is shown how developers can just focus on the requirements not how to ensure reading/writing inputs/outputs, how to express and define the different constraints, how to define intelligence and coordination strategies not how to define the infrastructure necessary for it.

Chapter 4

Modeling Methodology Based on the Proposed Framework

Contents

4.1 Introduction

This chapter introduces the proposed modeling methodology for correct and efficient controlling software of context-aware distributed reconfigurable systems. First, the background that has motivated the proposition of a new methodology is presented.

Then, a new UML profile, called GR-UML, is proposed in order to enable the specification of application's probabilistic features that need to run under memory and energy constraints. After that, transformation rules from GR-UML to GR-TNCES and to IEC 61499 function blocks are introduced. Right after, the methodology work flow that consists of three phases is proposed. A software tool chain that implements the methodology concepts is introduced. Finally, the methodology is applied to an example of a baggage handling system application is depicted. A second case study tackling the example of micrograids software is elaborated and reported in [Fkaier. *et al.* 2021a].

The contributions of this chapter are summarized in the following:

- The definition of a new UML profile called GR-UML aiming at specifying probability as well as memory and energy constraints.
- The definition of transformation rules that transform GR-UML to GR-TNCES and to IEC 61499 function blocks.
- The definition of a new modeling methodology aiming at creating clear, correct, and operable model.
- The implementation of a software tool for the proposed methodology.

4.2 Motivation

The modeling complexity of reconfigurable distributed systems is continually growing which prompts the necessity for finding more effective design methodologies and tools. IEC 61499 modeling standard is introduced to allow the design of distributed systems using the function block as elementary component. This standard provides a set of advantages that encourages its adoption. In fact, IEC 61499 provides portability, reconfigurability, and interoperability features. IEC 61499 allows also to model distributed agents communicating through messages thanks to its event-driven communication. It also offers the possibility of execution of agents directly on micro Programmable Logic Controllers (PLC). However, despite of its multiple merits, there is still a need for some stages and complementary processes in order for us to obtain clear and effective models and in effective way as well. In fact, using function blocks in the early stages of modeling does not provide the possibility to analyze the system/application requirements [Thramboulidis 2006], [Thramboulidis 2007].

Using UML instead allows rich analysis through different perspectives namely the structural and behavioral views. UML offers also a semi-formal modeling base for software applications. More importantly, some advanced application's features (especially that we talk about context-aware intelligent applications) could not be easily modeled with function blocks. Nevertheless, UML still has also some limitations when it comes to models correctness. UML does not offer formal semantics that enable to design probabilistic scenarios running under memory and energy constraints. Therefore, there is a need to extend the existing UML semantics with new semantics that enable to specify probabilistic and resources constrained scenarios.

As stated in Chapter 2, formal verification provides various benefits that improve the design efficiency through catching ambiguities and errors, and provide reliable results with less time and effort. To benefit from these assets, analyzing the application models with formal verification techniques such as Petri nets formalism might provide promising results.

In addition to all that, it is important to provide a systematic method that helps in improving the modeling efficiency and flexibility: a method that supports the incremental modification, refinement, and evaluation at multiple levels and that provides reliable results.

Considering these basis, we propose a new UML profile, called the Generalized Reconfigurable UML (GR-UML) which enriches the existing semantics with the ability to model probabilistic and resource constrained scenarios. Afterwards, we introduce a modeling methodology which consists in modeling applications using the framework models (models presented in Chapter 3 can be extended according to the use case) as well as the new GR-UML. In fact, the logic of the controller and the services can include probabilistic logic and applications can have resources constrained control. After that, formal analysis is conducted over the Petri nets models of the application. Finally, analysis and simulation of the function block models of the application on target hardware environment can be tested through the function block edition tools.

The accomplishment of the second and third phases is achieved thanks to a set of model transformation rules that map the models from GR-UML to GR-TNCES Petri nets extension and to IEC 61499 function blocks. A visual software tool that implements the defined transformations rules is developed.

It is worth noting that there are approaches that transform function block models to Petri nets. But not all application features can be modeled in the function block level [Thramboulidis 2008], e.g., the rule bases, the inference engine, etc. In real world implementations, applications can be deployed in different parts such as PLCs, networked servers, networked computers, and even the cloud [Lu *et al.* 2019]. Hence in order to cover the whole application model analysis, it is better to transform the UML application model to Petri nets rather than transforming the function block model to Petri nets model. Nevertheless, developers who are interested to verify the obtained function block behavior can do it using the existing approaches.

4.3 Preliminaries

In this section, we present the GR-TNCES formalism as well as our formal definition of the IEC 61499 concepts.

4.3.1 GR-TNCES

As presented in [Khlifi *et al.* 2019], the GR-TNCES formalism is defined as a network of R-TNCES given by: G= $\{\sum$ R-TNCES$\}$, where R-TNCES=(B, R) with B a behavior module and R a control module.

The behavioral module B is defined as a union of multiple TNCES and given by: $B = (P, T, F, QW, CN, EN, DC, V, Z_0)$

where;

- P is a finite non-empty set of places,
- T is a finite non-empty set of transitions,
- F is a finite set of flow arcs such that $F \subseteq (P \times T) \cup (T \times P)$
- $QW = (Q, W)$ with $Q : F \to [0, 1]$ is the probability of the arc and W is a mapper that maps a weight to a flow arc such that $(P \times T) \cup (T \times P) \to \{0, 1\}$, $W(x, y) > 0$ if $(x, y) \in F$ and $W(x, y) = 0$ otherwise, with x *and* $y \in P \cup T$,
- CN is a set of condition signals with $CN \subseteq (P \times T)$,
- EN is a set of event signals with $EN \subseteq (T \times T)$
- DC is a superset of time constraints on output arcs such that $F \subseteq (P \times T) \to [l, h]$
- $V : T \to \{\vee, \wedge\}$ indicates an event processing mode to each transition (AND or OR),
- $Z_0 = (T_0, D_0)$ where $T_0 : P \to \{0, 1\}$ is the initial marking position and $D_0 : P \to \{0\}$ is the initial clock position.

The control module R is defined as a set of reconfiguration functions $\{r_1, ..., r_n\}$ running under resource constraints (memory and energy). A reconfiguration function called r is defined as a structure given by: $r = (Cond, Q, E_0, M_0, S, X)$ where;

- $Cond : CN \to \{true, false\}$ is the precondition of r and is evaluated to true or false,
- $Q : F \to [0..1]$ is the TNCES probability that can be an internal (TNCES related) or a reconfiguration probability,
- $E_0 : P \to [0..max]$ is controlling the energy required by the TNCES,
- $M_0 : P \to [0..max]$ is controlling the memory required by the TNCES,

- $S : TN(\bullet r) \rightarrow TN(r\bullet)$ is the modification instruction of a reconfiguration,
- X : $laststate(\bullet r) \rightarrow initialstate(r\bullet)$ is the state processing function, where $laststate(\bullet r)$ indicates the last state of $\bullet r$ before the application of r and $initialstate(r\bullet)$ indicates the initial state of $r\bullet$ after the application of r.

4.3.2 Function Blocks Formal Definition

IEC 61499 standard provides a function block concept having three main function block types: basic, composite, and service interface function blocks, denoted respectively by BFB, CFB, $SIFB$. A function block, regardless its type, is defined by an interface, denoted by I, and an internal structure.

The interface I of all function block types is given by

$$I = (InE, OutE, InD, OutD, InW, OutW). \tag{4.1}$$

where

- InE (*resp.* $OutE$) is a set of input (*resp.* output) events.
- InD (*resp.* $OutD$) is a set of input (*resp.* output) data.
- InW (*resp.* $OutW$) is a set of With-associations for inputs (*resp.* for outputs).

4.3.2.1 Basic Function Block

A basic function block BFB is defined as

$$BFB = (I, ECC, A). \tag{4.2}$$

where I is its interface, ECC is its execution control chart, and A is the encapsulated algorithms.

A is the algorithms defining the encapsulated functionalities of a basic function block and it is given by

$$A = \{alg_i | i \in \{1, ..., |A|\}\}. \tag{4.3}$$

where alg_i is an algorithm.

ECC is the execution control chart supervising the operation of a function block. It is given by:

$$ECC = (ES, EA, ET, EF). \tag{4.4}$$

where,

- ES is a set of states of ECC.
- EA is a set of actions, an action is associated to an algorithm alg_i and output events of I.

- ET is a set of transitions between ECC states. Each transition has a guard condition which is the coming of an input event of I.
- EF indicates the flow between the different ECC states.

4.3.2.2 Composite Function Block

A composite function is defined as

$$CFB = (I, N). \tag{4.5}$$

where N is a network of basic and/or composite function blocks.

4.3.2.3 Service Function Block

$SIFB$ this type of function blocks represents an interface to services offered by the operating system or the device, e.g., interface to hardware (sensors, motors, controllers) or communication services (client/server communication).

4.4 New UML Profile: GR-UML

Thanks to the extensibility mechanisms offered by UML, we improve the class and statechart diagrams by means of the definition of new semantics at the aim of specifying the probabilistic features and memory and energy constraints.

4.4.1 Class Diagram Definition

UML class diagram is introduced as one of the structural view enablers of a system/application. It describes the system structure by showing the classes, their attributes, their methods, and the relations among objects.

In order to make the semantics of this diagram more suitable to the probabilistic reconfigurable systems that can operate under resources constraints (i.e., memory and energy), we extend its vocabulary by using new stereotypes to express the probabilistic property. Hence, we extend the solution proposed in [Salem *et al.* 2015a] by defining nine stereotypes of the attributes of classes as follow:

- $<<$ *probability* $>>$: stereotype used to determine that the given attribute is a probabilistic functionality/operation.
- $<<$ *memory* $>>$: stereotype used to determine that the given attribute represents memory resources of an operation.
- $<<$ *energy* $>>$: stereotype used to determine that the given attribute represents energy resources of an operation.
- $<<$ *in* $>>$: stereotype used to determine that the given attribute is a module input.

- $<<$ *out* $>>$: stereotype used to determine that the given attribute is a module output.
- $<<$ *input* $>>$: stereotype used to determine that the given attribute is system input.
- $<<$ *output* $>>$: stereotype used to determine that the given attribute is a system output.
- $<<$ *eventInput* $>>$: stereotype used to determine that the given attribute is an input event of a module.
- $<<$ *eventOutput* $>>$: stereotype used to determine that the given attribute is an output event of a module.
- $<<$ *integer* $>>$: stereotype used to determine that the given attribute is an integer.
- $<<$ *boolean* $>>$: stereotype used to determine that the given attribute is a boolean.

Whenever it is required to model equipment resources consumption especially memory and energy ones, it is required to have relevant methods for it. Thus, we define the hereafter methods:

- $checkEnergy(name : string) : bool$ - controls the energy resources mentioned by $name$.
- $checkMemory(name : string) : bool$ - controls the memory resources mentioned by $name$.

Based on this semantic extension, a class diagram can be defined as follows:

$$CD = \{Cl, At, Me, S, \psi, \omega\}. \tag{4.6}$$

where,

- $Cl = \{cl_1, cl_2, ..., cl_m\}$ is a finite set of classes.
- $At = \{at_1, at_2, ..., at_n\}$ is a finite set of attributes of classes.
- $Me = \{setInput, setOutput, resetInput, resetOutput, checkEnergy, checkMemory; setCeiling\}$ is a set of methods of the classes.
- $S = \{<<$ *probability* $>>, <<$ *memory* $>>, <<$ *energy* $>>, <<$ *in* $>>, <<$ *out* $>>, <<$ *input* $>>, <<$ *output* $>>, <<$ *eventInput* $>>, <<$ *eventOutput* $>>, <<$ *boolean* $>>, <<$ *integer* $>>\}$ is a finite set of stereotypes.
- $\psi : at_i \to cl_j$ is a function mapping the attribute at_i to the class cl_j.

- $\omega : s_i \rightarrow at_j$ is a function mapping the stereotype s_i to the attribute at_j.

Based on the new extended class diagram definition, we present an example of use of the added elements (i.e., the stereotypes $<< probability >>$, $<< memory >>$, $<< energy >>$ and the methods $checkEnergy$, $checkMemory$) as follows, let us consider these stereotyped attributes $<< Memory >> \rightarrow OperationMode1Memory : real$, $<< Energy >> \rightarrow OperationMode1Energy : real$ as well as the methods $checkMemory(\text{"}OperationMode1Memory\text{"})$ and $checkEnergy(\text{"}OperationMode1Energy\text{"})$ that can be used by classes that are responsible for resources control, and $<< probability >> \rightarrow Behavior1Probability : real$, $<< probability >> \rightarrow Behavior2Probability : real$, that can be used by classes responsible for the logic.

4.4.2 Statechart Diagram Definition

Statechart diagram is a tool proposed by UML to specify the behavior of objects (which are issued of a specific class). Some of the events of the statechart diagram can be specified with the $<< eventInput >>$ and $<< eventOutput >>$ stereotypes, and more importantly, guard conditions can now include the $<< probabilty >>$ stereotype that allow to express probabilistic transition from one state to another.

We define statechart diagram as follows

$$SD = \{St, Tr, Ev, G, Ac, Fr, Jn, Fl, Ch, \gamma, \delta, \varepsilon\}. \tag{4.7}$$

where:

- $St = \{st_1, st_2, ..., st_n\}$ is a finite set of states in an SD.
- $Tr = \{tr_1, tr_2, ..., tr_m\}$ is a finite set of transitions in an SD.
- Ev is a finite set of events in transitions of SD.
- G is a finite set of guards in SD.
- Ac is a finite set of actions in SD.
- Fr is a finite set of fork pseudostates in SD.
- Jn is a finite set of join pseudostates in SD.
- Fl is a finite set of the transitions flow, such that $Fl \subseteq (St \times Tr) \cup (Tr \times St)$.
- Ch is a finite set of choice pseudostates.
- γ: $ev_i \rightarrow tr_j$ is a function mapping an event ev_i of Ev to a transition tr_j of Tr.
- δ: $gr_k \rightarrow tr_j$ is a function mapping a guard gr_k of Gr to a transition tr_j of Tr.
- ε: $act_l \rightarrow tr_j$ is a function mapping an action act_l of Ac to a transition tr_j of Tr.

4.4.3 Component Diagram Definition

In the current research, only simple components are used in the UML component diagram. Hence, we define a component diagram $ComD$ as

$$ComD = \{Co, In, \alpha_{dep}, \alpha_{rea}\}. \tag{4.8}$$

where:

- Co is a finite set of components of $ComD$.
- In is a finite set of interfaces of $ComD$.
- α_{dep} is a finite set of dependency relationships of components to interfaces such that $\alpha_{dep} \subseteq (In \times Co)$.
- α_{rea} is a finite set of realization relationships of interfaces by components such that $\alpha_{rea} \subseteq (Co \times In)$.

4.5 Model Transformations

In this section, the transformation of the GR-UML models to GR-TNCES and IEC 61499 function blocks is provided.

4.5.1 Transformation of GR-UML to GR-TNCES

We propose the following list of transformation rules:

- Rule 1: A state $st \in St$ is mapped to a place $p \in P$ of an $R-TNCES$. For example, Figure 4.1 shows an example where the state st_b is transformed to a place p_1.
- Rule 2: A transition $tr \in Tr$ is mapped to a transition $t \in T$ of an $R-TNCES$. For example, Figure 4.1 shows an example where the transition tr is transformed to a transition t_1.

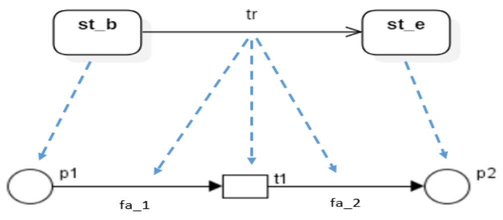


Figure 4.1: Transformation of simple statechart example.

- Rule 3: A transition tr in SD is given by a pair of states, st_b and st_e, where the first is the state from which tr is taken and the second is the next state if tr fires. A transition t and two places p_{out} and p_{to} are created using, respectively, Rule 2 and Rule 1. Rule 3 creates in the $R-TNCES$ a flow arc, $f_{a_1} \in F$, linking p_{out} to t, and another one, $f_{a_2} \in F$, linking t to p_{to}. Figure 4.1 shows an illustrative example of Rule 3.

- Rule 4: In SD, guard conditions can be assigned to some transitions. A guard gr is transformed to three condition arcs, ca, in an $R-TNCES$. A condition output signal, co, is added to the place guaranteeing the condition and a condition input signal, ci, to the related transition. Figure 4.2 shows an example of a guard condition mapping. In this example, the place guaranteeing the condition is p_3 belonging to $TNCES2$.

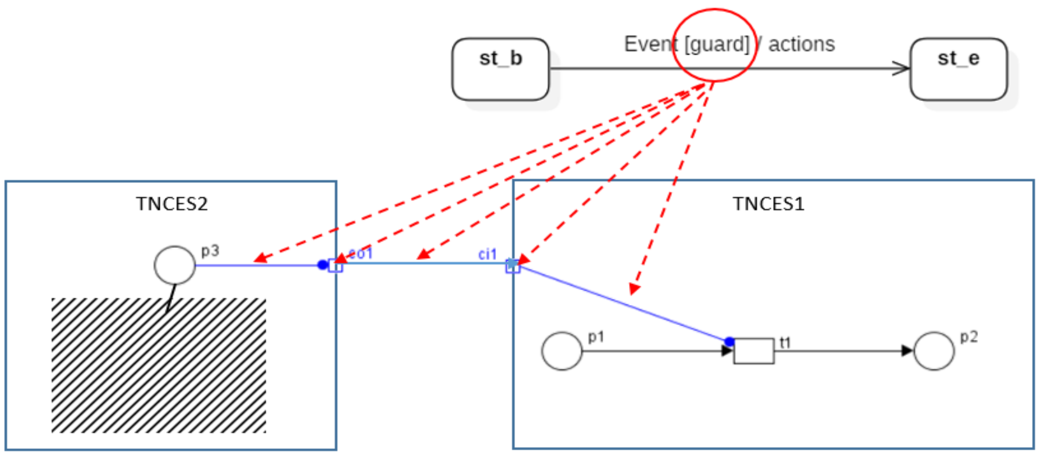


Figure 4.2: Transformation example of guard condition of transition.

- Rule 5: In an SD, guards stereotyped with $<< probability >>$ are mapped to the set QW of the behavioral module B of $R-TNCES$. A probabilistic guard gr has the form of $[Proba_i == x]$ where $Proba_i$ stands for the probability on the transition and x is the value of the probability which is a real number between 0 and 1. Figure 4.3 shows an example of a probabilistic guard condition mapping. In this example, $Proba_1$ is mapped to Q_1 which may take two values x_1 or x_2.

- Rule 6: In SD, actions can be assigned to some transitions. An action ac is mapped to four event arcs, ea, in an $R-TNCES$. An event output signal, eo, is added to the transition from which the event is triggered and an event input signal, ei, as well as an transition t_{ac} are added to the related transition t. Figure 4.4 shows an example of an action mapping.

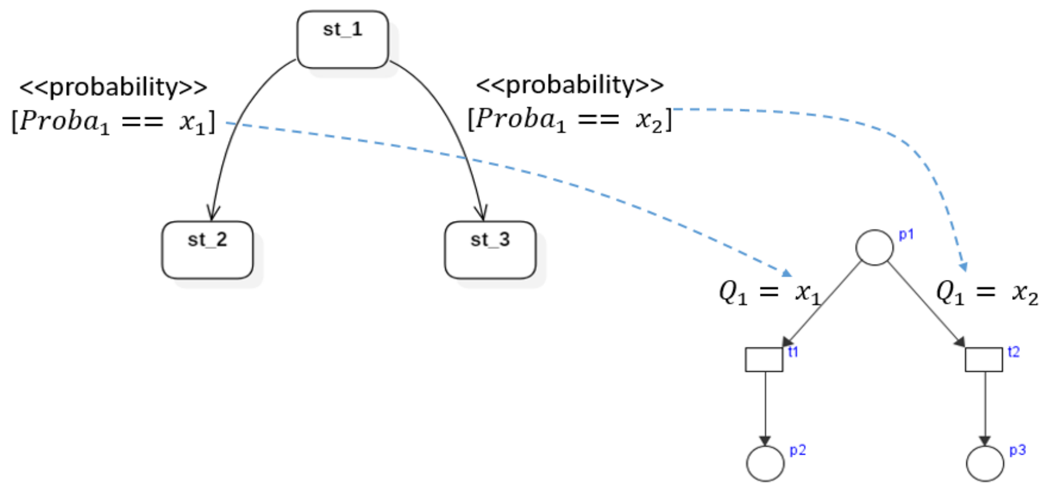


Figure 4.3: Transformation example of probabilistic guard condition of transition.

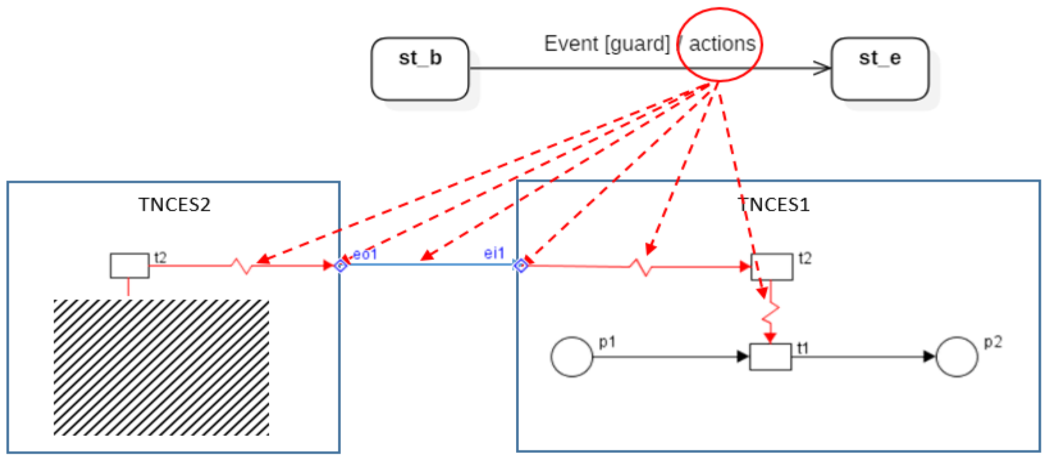


Figure 4.4: Transformation example of transition action.

- Rule 7: A fork pseudostate fr in an SD that splits an SD transition tr into several orthogonal regions of a composite state cst, is mapped in $R-TNCES$ to a transition t_{fr_1} linked to a place p_{cst} representing the composite state cst, and p_{cst} is linked to a transition t_{fr_2} that is linked to n places representing the internal simple states of a cst. Figure 4.5 shows an example of a fork pseudostate mapping.
- Rule 8: A join pseudostate jn in an SD that merges SD transitions from several orthogonal regions of a composite state cst, is mapped in $R-TNCES$ to a transition t_{jn_1} that is linked to n places representing the internal simple states of a cst, t_{jn_1} is linked to a place p_{cst} representing the composite state cst, and p_{cst} is linked to a transition t_{jn_2}. Figure 4.6 shows an example of a join pseudostate mapping.

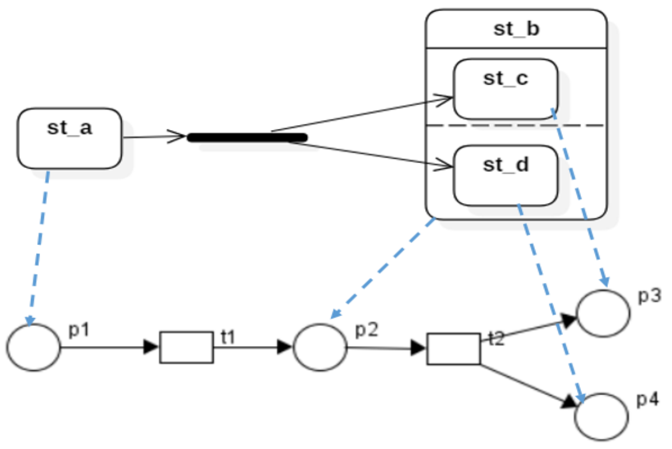


Figure 4.5: Transformation example of fork pseudostate.

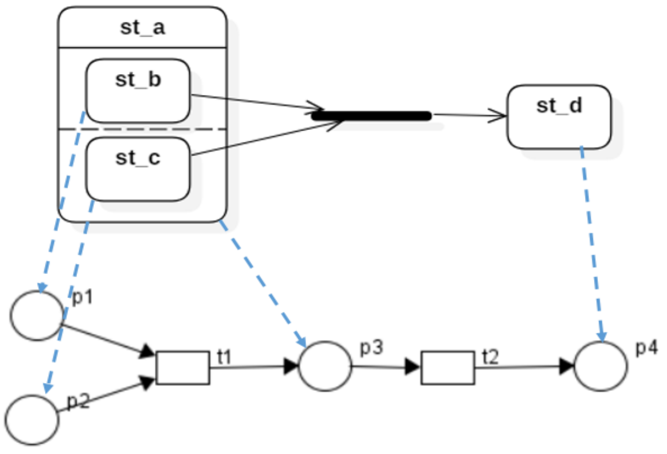


Figure 4.6: Transformation example of join pseudostate.

- Rule 9: A choice pseudostate ch in an SD that splits an SD transition to multiple conditional outgoings is mapped in $R-TNCES$ to a decision state along with transition t_{ch} that is linked to n places representing the target states, also n input/output condition signals are added according to Rule 4. Figure 4.7 shows an example of a choice pseudostate mapping.

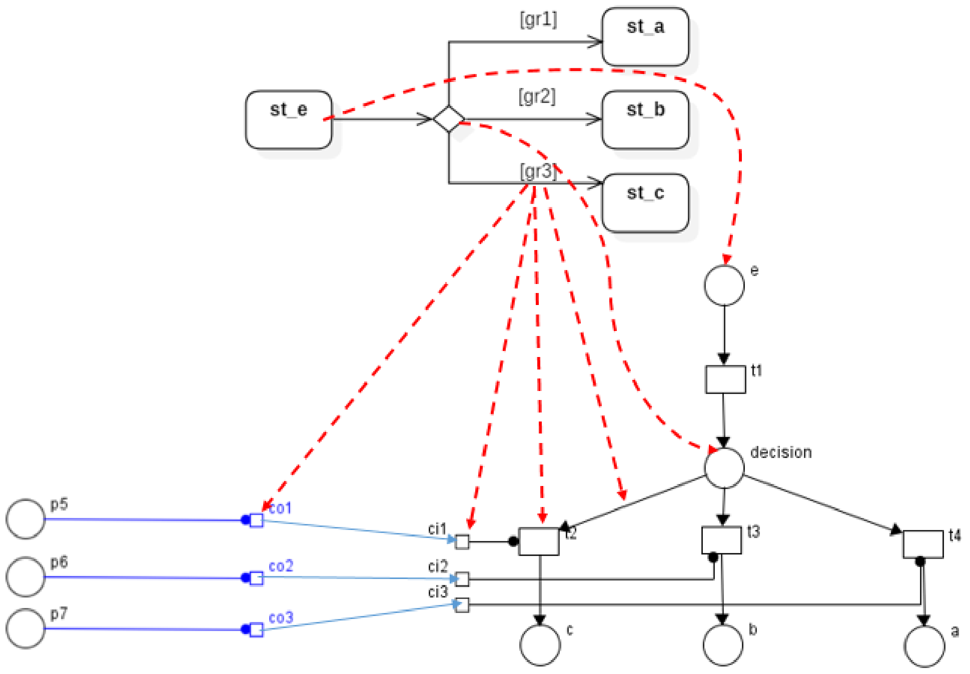


Figure 4.7: Transformation example of choice pseudostate.

The Table 4.1 depicts a summary of the proposed transformation rules.

Table 4.1: Summary of transformation rules of statechart diagram to GR-TNCES.

ID	**GR-UML Statechart Diagram**	**GR-TNCES**
Rule 1	St	P
Rule 2	Tr	T
Rule 3	Fl	F
Rule 4	$gr \in G$	$ci \in C^{in}$, $co \in C^{out}$, $\{ca\} \subset CN$
Rule 5	probabilistic $\{gr_i\} \subset G$	Q of QW
Rule 6	$ac \in Ac$	$ei \in E^{in}, eo \in E^{out}, t \in T, \{ea\} \subset EN$
Rule 7	$jn \in Jn$	$t \in T, \{fa\} \subset Fl$
Rule 8	$fr \in Fr$	$t \in T, \{fa\} \subset Fl$
Rule 9	$ch \in Ch$	$p \in P$, $\{t\} \subset T, \{fl\} \subset Fl$, $\{ci\} \subset C^{in}$, $\{co\} \subset C^{out}$, $\{ca\} \subset CN$

4.5.2 Transformation of GR-UML to IEC 61499 Function Blocks

In order to get the function blocks model of an application, both GR-UML component and statechart diagrams are used for transformation. Statechart diagram is used to transform the execution control chart of basic function blocks while component diagram is used to map the network of function blocks, i.e., relations between function blocks as well as function blocks interfaces. We extend the work reported in [Panjaitan & Frey 2007] to obtain the following transformation rules.

- Rule 10: A component $co \in Co$ of the component diagram $ComD$ corresponds to a composite function block CFB. Figure 4.8 shows an example of transformation of *Component1* of the GR-UML components diagram to a composite function block of the IEC 61499 function block.

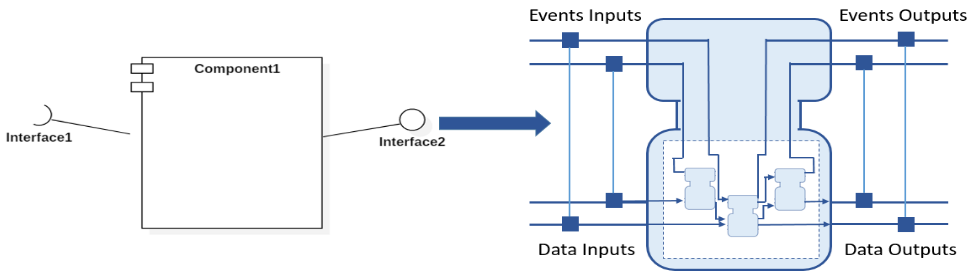


Figure 4.8: Transformation example of a GR-UML component to a composite function block.

- Rule 11: In GR-UML component diagram, as it is the case for class diagram, interfaces contain methods, where each method has a name, a set of input arguments defined by their name and type, and a return value defined by its type. An interface $in \in In$ realized by a component co_a and used by a component co_b of a component diagram $ComD$ is mapped to inputs of the interface I_b (i.e., subsets of InE, InD, InW) of function block fb_b and a part of the interface I_a (i.e., subsets of $InE, InD, InW, OutE, OutD, OutW$) of function block fb_a.

- Rule 12: Based on Rule 11, methods defined an interface in are mapped to I_a as follows:

  - Every provided method defined in in is mapped to an input event $ine_i \in InE$ with the proper associated with-association $inw_i \in InW$ as well as the relevant input data $ind^i \subset InD$, such that the name of the method is mapped to the name of event, the arguments of the method are mapped to the associated data, and the types of method arguments are the mapped to the type of input data.

  - The output of the provided method is mapped to an output event $oute_i \in OutE$ with the proper associated with-association $outw_i \in OutW$ as well as the relevant output data $outd^i \subset OutD$, such that the return type is the output data type.

For example, Figure 4.9 shows an example of transformation of *Method2* of *Interface1* to the input event *method2* associated with the input data *input2,*

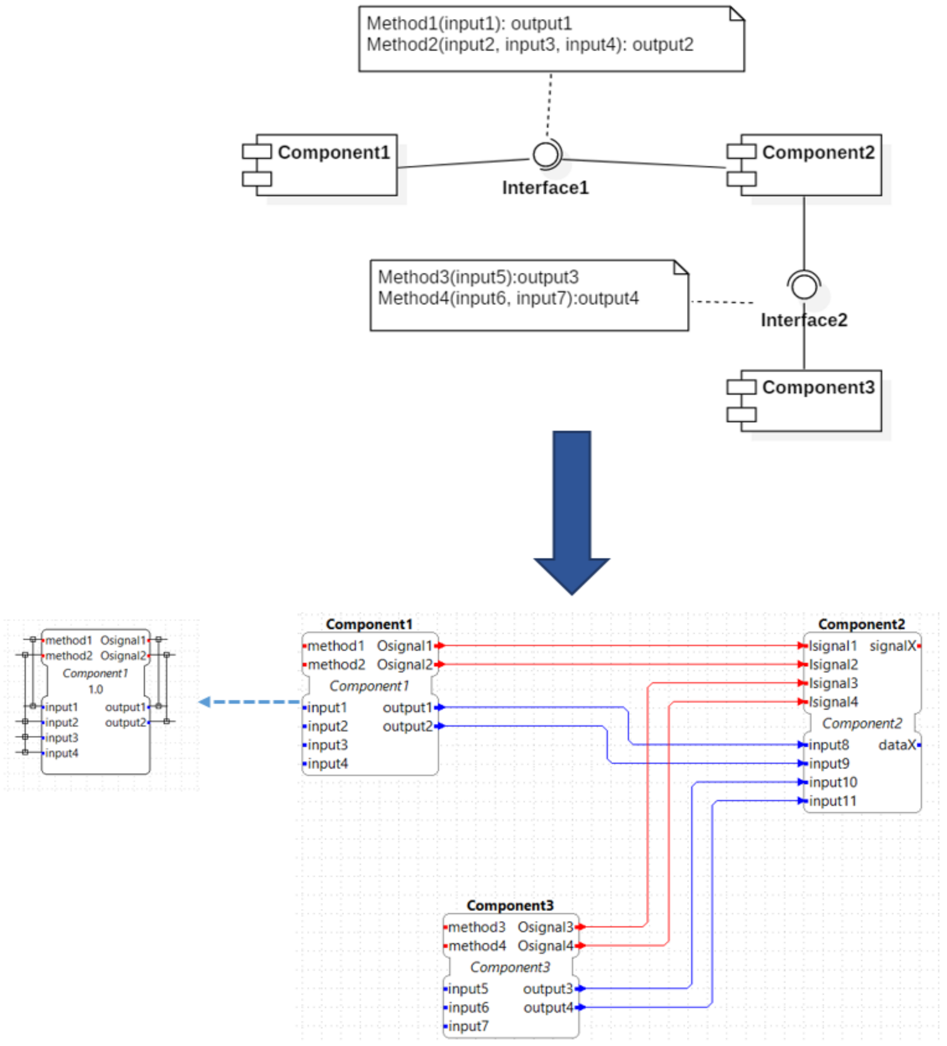


Figure 4.9: Transformation example of components to composite function blocks.

input3, *input4* and the return of *Method2* is transformed to the output event *Osignal2* associated with the output data *output2*.

- Rule 13: Based on Rule 11, required methods defined an interface *in* are mapped to I_b as follows: Every method defined in *in* is mapped to an input event $ine_i \in InE$ with the proper associated with-association inw_i as well as the relevant input data $ind^i \subset InD$, such that the name of the method is mapped to the name of event and the arguments of the method are mapped to the associated data. The types of method arguments are the mapped to the type of data. From Figure 4.9, *Component2* uses/requires *Interface1* and *Interface2*, then the outputs of *method1*, *method2*, *method3*, *method4* are mapped to inputs of *Component2*.

- Rule 14: The internal activity of each module (i.e., component) is transformed according to its statechart diagram as follows: each simple state is mapped to a basic function block while each composite state is mapped to a composite function block.

- Rule 15: A state requiring the use of a "data base" such as the rule base of the artificial intelligence module AIM, the system history H of AIM, context attributes model $CAMS$ of the inputs module IM, the context rule store CRS of IM, can be mapped to service interface function blocks $SIFB$ since such functionalities are generally performed by computers/servers networked with field devices not by function blocks.

- Rule 16: States in a GR-UML statechart diagram do generally hold a set of different internal activities (entry action, do activity action, exit action, deferrable trigger). These activities (i.e., actions) are transformed to the algorithms of a basic function block. The evolution of state activities are transformed to the execution control chart ECC as depicted in Figure 4.10, where each internal action ia_i is mapped to a state $es_i \in ES$ of ECC along with the algorithm associated to it.

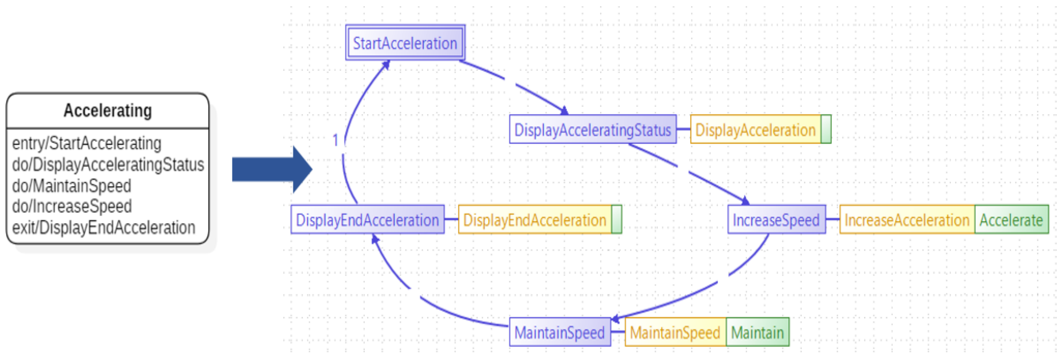


Figure 4.10: Transformation example of state internal activities to ECC of a BFB.

- Rule 17: Each transition $tr_i \in Tr$ in a statechart diagram that is leaving a state st_a and entering a state st_b is transformed to transitions linking an output event $oute_i \in OutE_a$ and the related output data $outd^i \subseteq OutD_a$ to the input event $ine_i \in InE_b$ and the related input data $ind^i \subseteq InD_b$ (i.e., blue and red links in Figure 4.9).

- Rule 18: Each guard gr_i associated to a transition tr_a linking a state st_a to a state st_b in the statechart diagram is transformed to output data $outd_i$ along with a with-association $outw_i$ of the interface of the source function block fb_a and to input data ind_i along with a with-association inw_i of the interface of the destination function block fb_b.

These transformation rules facilitate the generation of function blocks through creating the skeleton of the model. Details and precision of the model (for example the order of transitions in ECC) could be added manually.

The Table 4.2 depicts a summary of the proposed transformation rules from GR-UML component diagram to IEC 61499 function block network, while Table 4.3 depicts a summary of transformation rules from GR-UML statechart diagram to function blocks.

Table 4.2: Transformation rules of component diagrams to function blocks.

ID	**GR-UML Component Diagram**	**IEC 61499 Function Block**
Rule 10	$ComD$	CFB network
Rule 11	(1) required interfaces, (2) provided interfaces	(1) set of inputs of I, (2) part of the interface I
Rule 12	provided method	$ind^i \subseteq ind, outd^i \subseteq outd, ine^i \subseteq ine, oute^i \subseteq oute, inw^i \subseteq inw, outw^i \subseteq outw.$
Rule 13	required method	$ind^i \subseteq ind, ine^i \subseteq ine, inw^i \subseteq inw.$

Table 4.3: Transformation rules of statecharts diagrams to function blocks.

ID	**GR-UML Statechart Diagram**	**IEC 61499 Function Block**
Rule 14	(1) simple state, (2) composite state	(1) basic function block, (2) composite function block
Rule 15	states requiring the use of data storage	service interface function block SIFB
Rule 16	state internal activities	*ECC* of a basic function block and its algorithms
Rule 17	transition $tr_i \in Tr$	$oute_i \in OutE_a$, $outd^i \subseteq OutD_a$, $outw_i \in OutW_a$, $ine_i \in InE_b$, $ind^i \subseteq InD_b$, $inw_i \in InW_b$
Rule 18	transition $gr_i \in Gr$	$outd^i \subseteq OutD_a$, $outw_i \in OutW_a$, $ind^i \subseteq InD_b$, $inw_i \in InW_b$

4.6 Methodology Work Flow

The methodology consists of three phases and its work flow is presented in Figure 4.11.

- The first phase consists in modeling applications using/extending the framework UML models according to the considered case, and using GR-UML to model the controller as well as the services logic.
- The second phase consists in performing formal analysis using the mathematical tools of the GR-TNCES Petri nets extension, also model checking can

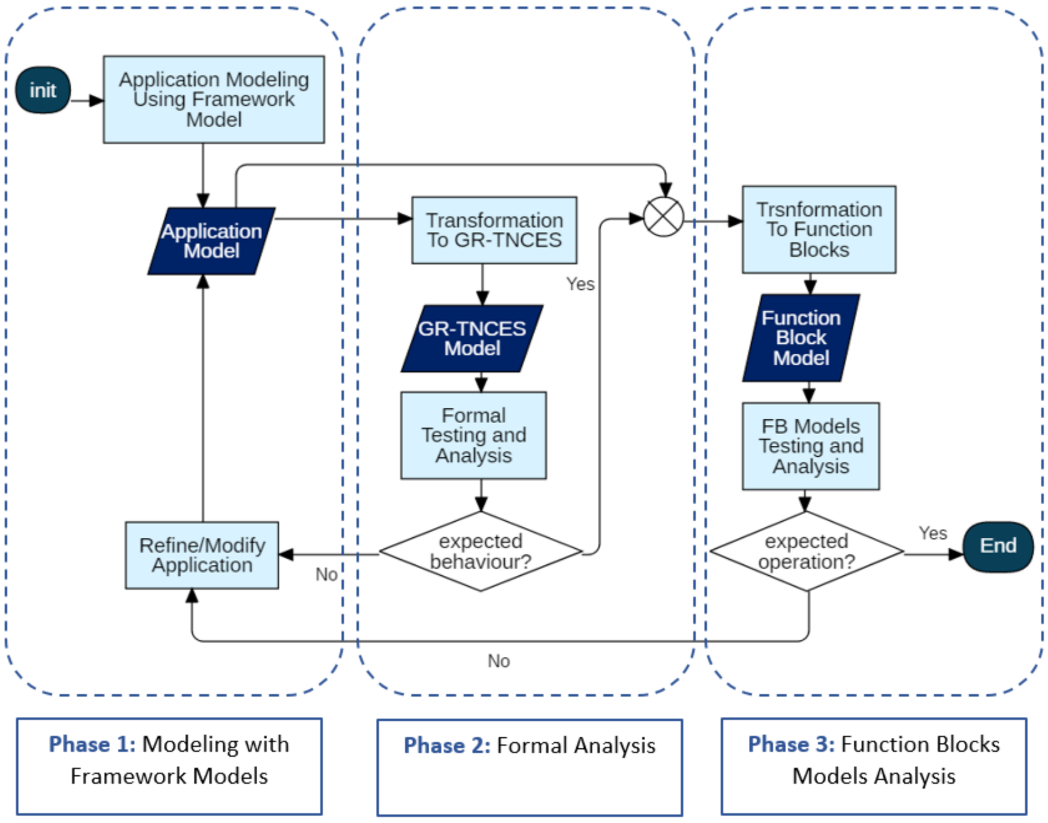

init
Application Modeling Using Framework Model
Application Model
Transformation To GR-TNCES
GR-TNCES Model
Formal Testing and Analysis
expected behaviour?
Yes
No
Refine/Modify Application
Trsnformation To Function Blocks
Function Block Model
FB Models Testing and Analysis
expected operation?
Yes
End
No
Phase 1: Modeling with Framework Models
Phase 2: Formal Analysis
Phase 3: Function Blocks Models Analysis

Figure 4.11: Modeling methodology flowchart.

be conducted since GR-TNCES provides the possibility to export models to PRISM model checker [Uddin *et al.* 2019].

- The third phase consists in analyzing the IEC 61499 function block models of applications. Deployment in target hardware environment analysis using the function block tools can also be performed.

The flow of the methodology provides the ability to analyze the output of each phase and to conduct modifications and refinements whenever required. The iteration through the three phases ends when the desired application model structure and behavior are obtained. The methodology makes the move from one phase to another easy thanks to the software tool chain that implements the defined transformation rules. In the following, more details about each phase are demonstrated.

4.6.1 Applications Modeling Using the Framework Models

Using abstract/high level modeling tools provides many advantages such as reducing time and effort, having unified understanding of the application, as well as conducting more sophisticated tests of particular applications [Thramboulidis 2004], [Schneider 2019]. In fact, defining the structural view (using component and class diagrams) helps to clearly model the application logic, parts, and components.

Further, to make this task easier for designers, this phase is performed based on the framework models as well as the GR-UML: designers have the possibility to use the concepts of the framework via extending it according to the considered use case. The framework facilitates the task through providing the "skeleton" of applications. Developers can take it as the starting base as it provides a well-organized software architecture, so there is no need to think about repetitive basic requirements, instead the effort is made on applications specificities. The framework architecture is defined in a way to reduce the complexity of application's design. It provides requirements presentation in a loosely coupled way. Thus easy to understand and easy to maintain. In addition, more sophisticated specification of probabilistic and resources constrained features of the controller or services logic can be performed.

4.6.2 Behavior Testing Using Formal Verification

Framework models facilitate the task of creating consistent and clear application models. However, for particular applications it is necessary to get more guarantees about the models correctness, especially for complicated reconfigurable systems in which ambiguities and errors can occur.

In this case, formal verification is required to get correctness guarantees based on mathematical basis. For this, a formal testing and analysis is proposed to be done as a second phase of the modeling of distributed reconfigurable applications. Models obtained in Phase 1 are transformed into GR-TNCES Petri nets models and formal analysis as well as model checking becomes possible.

Concerning the choice of the formalism, GR-TNCES provides a set of advantages over other Petri nets extensions as follows:

1. It ensures the modeling and verification of reconfigurable systems through providing a reconfiguration module which helps to make modeling of reconfigurable application simpler.
2. It is a "generalized" formalism where many aspects can be tackled including the probabilistic ones.
3. It allows to specify and verify resources constraints (memory and energy).
4. It allows optimizing the verification time [Hafidi *et al.* 2019].

The proposed framework as well as GR-UML are dedicated to develop reconfigurable systems, therefore GR-TNCES is more suitable than other formalism.

In this phase, Petri nets can be visualized in ZIZO [Salem *et al.* 2015b] the editor and simulator of GR-TNCES, places, transitions, input/output conditions, input/output events, arcs probabilities, TNCES modules and its networks can all be modeled and visualized. This graphical representation helps at first to recognize the model structure. Afterwards, manual mathematical analysis can be performed, moreover ZIZO provides the possibility to export models to ".pm" files readable by PRISM model checker, where easier properties checking is enabled.

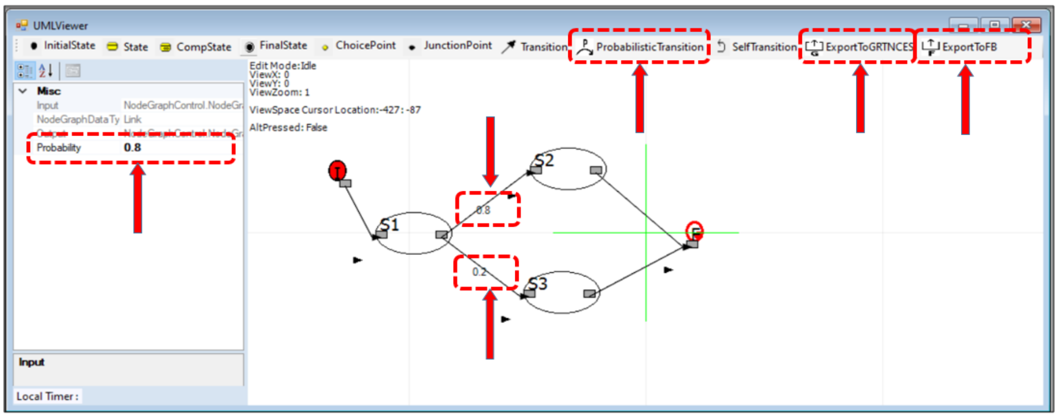
4.6.3 Function Blocks Models Analysis

Analysis of applications models according to IEC 61499 function blocks is performed in this phase. Guaranteeing compliance with IEC 61499, since it is a distributed systems reference modeling standard, is of great importance. Structural and behavioral aspects of applications developed in the first and second phases become more valuable if its analysis and even deployment is tested in a target environment. In this research, the open source PLC framework for industrial automation and control 4DIAC [1] is used for analysis. 4DIAC includes a development environment, a runtime environment, and a function block library that helps to easily set up tests. Other tools like the FBDK[2] can also be used.

4.7 Software Tool: ZiZo New Version

In order to facilitate the task of designers, we have created a visual software environment, called ZiZo, that implements the concepts of the proposed methodology. ZiZo was initially introduced in [Salem *et al.* 2015b], [Khlifi *et al.* 2019] as a editor, simulator, and verifier of R-TNCES as well as GR-TNCES models.

We have extended ZiZo with new features in order to allow the modeling of GR-UML. We have extended the statechart diagram viewer with the ability to draw probabilistic transitions as mentioned in Figure 4.12. We have also added an editor of component diagrams (see Figure 4.13).



Input	NodeGraphControl.NodeGr
NodeGraphDataTy	Link
Output	NodeGraphControl.NodeGr
Probability	0.8

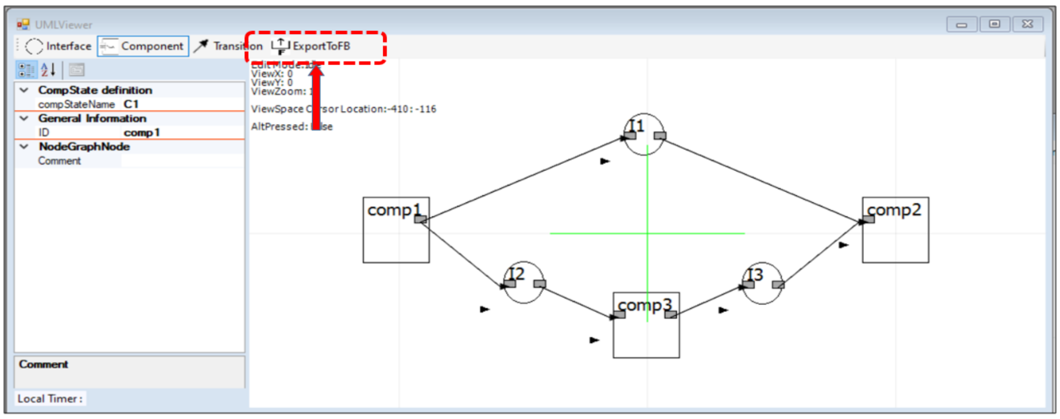
Edit Mode:Idle
ViewX: 0
ViewY: 0
ViewZoom: 1
ViewSpace Cursor Location:-427: -87
AltPressed: False

Figure 4.12: ZiZo: Probabilistic statechart diagram viewer.

In addition, we have created new buttons having the role of transforming the created models into GR-TNCES models in the form of ".zz" files (as mentioned in

[1] https://www.eclipse.org/4diac/

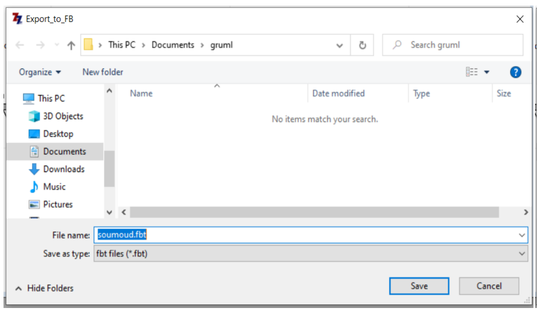
[2] http://ftp.holobloc.com/fbdk2/index.htm



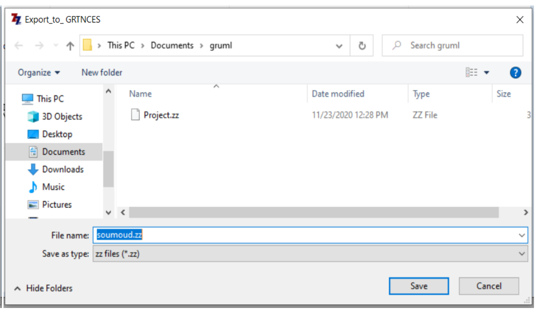
CompState definition
compStateName C1
General Information
ID comp1
NodeGraphNode
Comment
Comment
Local Timer :
ViewX: 0
ViewY: 0
ViewZoom:
ViewSpace Cursor Location:-410:-116
AltPressed: False

Figure 4.13: ZiZo: Component diagram viewer.

Figure 4.15) and into IEC 61499 function block models in the form of ".fbt" files (as mentioned in Figure 4.14). The logic behind these buttons contains the defined transformation rules.



File name: soumoud.fbt
Save as type: fbt files (*.fbt)

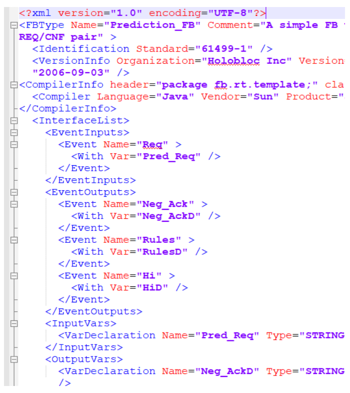


Project.zz 11/23/2020 12:28 PM ZZ File
File name: soumoud.zz
Save as type: zz files (*.zz)

Figure 4.14: Export to function blocks button.

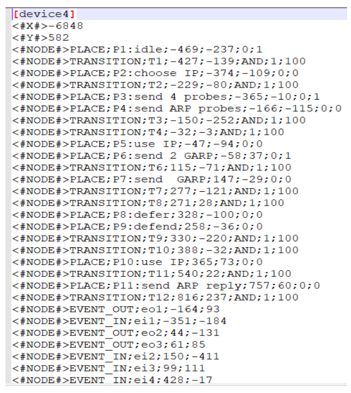
Figure 4.15: Export to GR-TNCES button.

The ".zz" files are readable by the part of ZiZo that contains the GR-TNCES modeling and simulation (the structure is mentioned in Figure 4.17), while ".fbt" files are readable by some tools of the IEC 61499 such as the 4DIAC and FBDK (the structure is like the XML as mentioned in Figure 4.16).



```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Prediction_FB" Comment="A simple FB
REQ/CNF pair" >
  <Identification Standard="61499-1" />
  <VersionInfo Organization="Holobloc Inc" Version
  "2006-09-03" />
<CompilerInfo header="package fb.rt.template;" cla
  <Compiler Language="Java" Vendor="Sun" Product="
</CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="Req" >
        <With Var="Pred_Req" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="Neg_Ack" >
        <With Var="Neg_AckD" />
      </Event>
      <Event Name="Rules" >
        <With Var="RulesD" />
      </Event>
      <Event Name="Hi" >
        <With Var="HiD" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="Pred_Req" Type="STRING
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="Neg_AckD" Type="STRING
      />
```

Figure 4.16: ".fbt" file readable by 4DIAC and FBDK.



```
[device4]
<#X#>-6848
<#Y#>582
<#NODE#>PLACE;P1:idle;-469;-237;0;1
<#NODE#>TRANSITION;T1;-427;-139;AND;1;100
<#NODE#>PLACE;P2:choose IP;-374;-109;0;0
<#NODE#>TRANSITION;T2;-229;-80;AND;1;100
<#NODE#>PLACE;P3:send 4 probes;-365;-10;0;1
<#NODE#>PLACE;P4:send ARP probes;-166;-115;0;0
<#NODE#>TRANSITION;T3;-150;-252;AND;1;100
<#NODE#>TRANSITION;T4;-32;-3;AND;1;100
<#NODE#>PLACE;P5:use IP;-47;-94;0;0
<#NODE#>PLACE;P6:send 2 GARP;-58;37;0;1
<#NODE#>TRANSITION;T6;115;-71;AND;1;100
<#NODE#>PLACE;P7:send  GARP;147;-29;0;0
<#NODE#>TRANSITION;T7;277;-121;AND;1;100
<#NODE#>TRANSITION;T8;271;28;AND;1;100
<#NODE#>PLACE;P8:defer;328;-100;0;0
<#NODE#>PLACE;P9:defend;258;-36;0;0
<#NODE#>TRANSITION;T9;330;-220;AND;1;100
<#NODE#>TRANSITION;T10;388;-32;AND;1;100
<#NODE#>PLACE;P10:use IP;365;73;0;0
<#NODE#>TRANSITION;T11;540;22;AND;1;100
<#NODE#>PLACE;P11:send ARP reply;757;60;0;0
<#NODE#>TRANSITION;T12;816;237;AND;1;100
<#NODE#>EVENT_OUT;eo1;-164;93
<#NODE#>EVENT_IN;ei1;-351;-184
<#NODE#>EVENT_OUT;eo2;44;-131
<#NODE#>EVENT_OUT;eo3;61;85
<#NODE#>EVENT_IN;ei2;150;-411
<#NODE#>EVENT_IN;ei3;99;111
<#NODE#>EVENT_IN;ei4;428;-17
```

Figure 4.17: ".zz" file readable by ZIZO.

4.8 Running Example

This section shows how the methodology allows to get reliable applications models, we consider especially the ones of the controller and the services. In fact, since the logic and behavior of the services and controller must be elaborated/terminated by developers, it is required to provide a way to get guarantees about the models correctness and efficiency. The controller has to interact with the rest of the modules and to make decisions of reconfigurations, etc. Therefore, proving the correctness of the controller and the whole application behavior is needed. In further step, the deployment testing in target hardware environment can be modeled/simulated using IEC 61499 function blocks.

To illustrate the methodology process, the same example of baggage handling system is considered. For simplicity reasons, we detail the modeling of one simple controller subroutine/subtask of loading services in the context of reconfiguration.

4.8.1 Phase 1 of the Methodology: Modeling using UML Models of the Framework

The first phase of the methodology consists mainly in using the ready/provided framework models and to define the models of the services and controller.

Figure 4.18 depicts a statechart diagram where the states of the controller are grouped in three main composite states: *SearchingForConfiguration* in which the interaction with the rest of modules can be accomplished, *DiscussingAboutConfiguration* in which a messaging with the distributed components of the BHS can be performed, and finally *LoadingServices* in which the controller needs to interact with the Service Layer (SL) in order to feed the outputs.

The focus of the current example is made on the last state, where Figure 4.18 depicts its detailed states. Four services are considered in this example, *Tracking Service* that must run in parallel with any other service, *Moving Backward*, *Moving Forward*, and *Halt*.

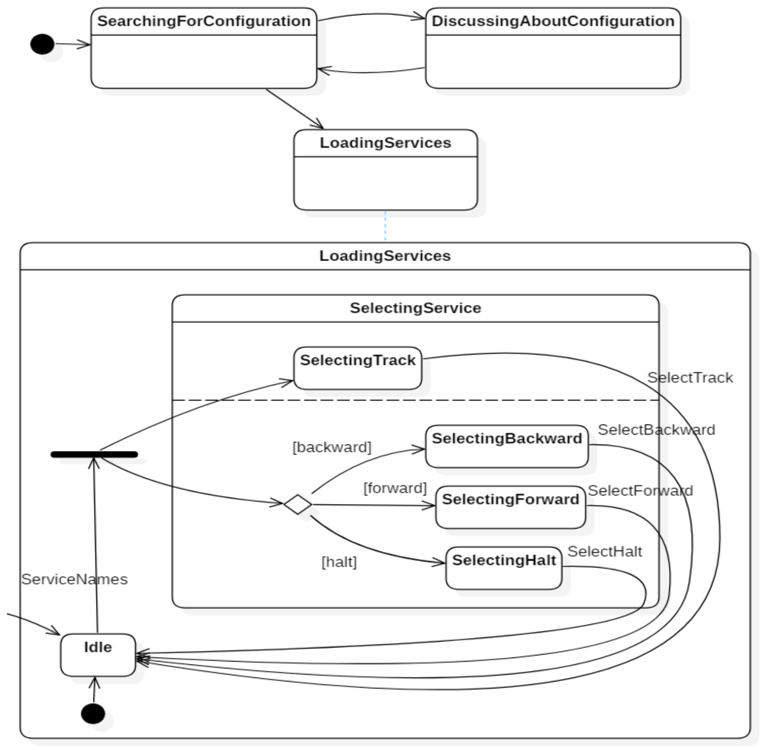


Figure 4.18: Statechart of the Loading Services subroutine of the controller.

The tracking service must run in parallel with any other service, this is why the services are designed using an orthogonal state, which is a kind of composite state where regions are assumed to be run in parallel, the first region contains the tracking service and the second region contains the other services.

Since the forward, backward, and halt service are excluding services, i.e., the execution of the one means the non execution of the others, a choice along with guard conditions are used to ensure the selection of only one of them.

4.8.2 Phase 2 of the Methodology: Formal Verification

One of the important advantages of the GR-TNCES formalism is that it provides modular representation and verification of complex applications, through decomposing/dividing the target into smaller modules, then a verification module-by-module can be accomplished. This feature helps to facilitate the task, make it clearer, and also to easily localize errors if they exist.

After transforming the GR-UML models, using the transformation rules presented earlier, we get the following GR-TNCES formal model. For example, Rule 1 results in seven places since we have seven states, Rule 4 and Rule 9 result in three condition inputs since we have a choice pseudostate with three guard conditions, Rule 8 results in having concurrent subnets since we have a fork pseudostate.

ZIZO tool allows to visualize and edit the GR-TNCES models. In this step,

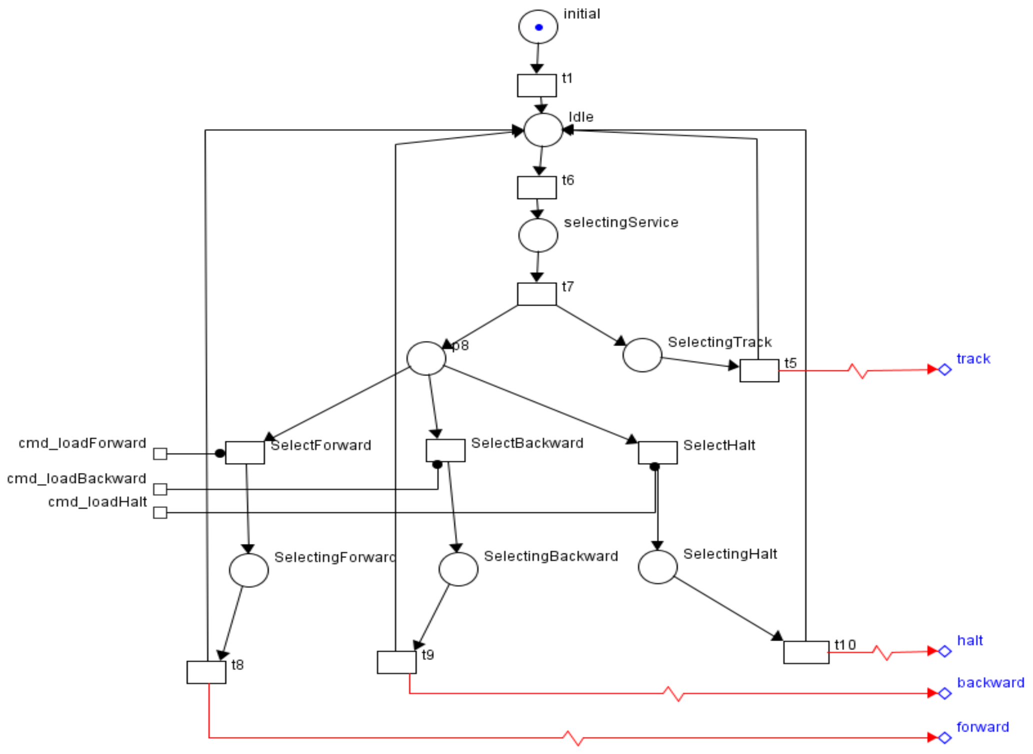


Figure 4.19: Transformation of the composite state Loading Services.

TNCES modules network can be visualized/created, etc. ZIZO enables also to compute the set of reachable states. Figure 4.19 depicts the visual Petri net of the considered example.

Now many properties can be checked, such as liveness and safety properties. It is recalled here that **safety** signifies that nothing undesired happens, and **liveness** signifies that a property is fulfilled by all the net's executions (deadlock free behavior).

It is noted that ZIZO allows the automatic code generation of PRSIM code in the form of ".pm" files. PRISM is a probabilistic model checker where Computation Tree Logic (CTL) and especially Probabilistic Computation Tree Logic (PCTL) properties can be used to analyze models of complex probabilistic scenarios.

In the current case, we show a simple verification example. At first, we have created the reachability graph of the given behavior model and verified that the model is deadlock free. Then, we checked a set of properties. As an example of the safety properties that can be checked for the *LoadingServices* module, we present the following,

1. Service forward and service backward should never happen simultaneously. This means that we need to check whether it is possible to reach a state where both places *SelectingForward* and *SelectingBackward* hold. For this we check this CTL formula

$$\mathbf{EF}(SelectingForward \wedge SelectingBackward). \tag{4.9}$$

The result returns false, then no contradictory states can happen.

2. Service forward and service halt should never happen simultaneously.

3. Service backward and service halt should never happen simultaneously.

As an example of the functional properties that can be checked for the *LoadingServices* module, we present the following,

1. Service track should work together with halt, backward, and forward services. This means that the place *SelectingTrack* holds often on every computation path. Hence, we verify this CTL formula

$$\mathbf{AG}(\mathbf{AF}SelectingTrack). \tag{4.10}$$

The obtained result is true, then we are sure that tracking will always be activated.

Probabilistic properties verification: Let us suppose that the delays of flights arrivals is on average about 20% per year for the considered airport example. Delays by 15 minutes or more can generally happen due to many reasons such as the adverse weather, air traffic control, connecting passengers, security clearance, and many other reasons. This fact incites the use of probabilistic logic to model the possible controlling scenarios at the aim of optimizing the waiting times [Frey 2014] and bags processing. To illustrate the transformation of probabilistic GR-UML statechart diagram to GR-TNCES and how it can be analyzed with PRISM, let us show a part of the logic of the composite state *SearchingForConfiguration* of Figure 4.18. Let us assume that the controller must lead two different processing "branches", the first is used whenever all flights are managed as planned and the second is used whenever delays happen (a delay may induce disturbances in the associated flight and in the successive ones as well). These two scenarios must not be used together.

Figure 4.20 shows that the probability of delays occurrence is characterized by $proba_1$ and consequently the likelihood to enter in a *ProcessingDelayDueScenario* is $proba_1$.

Thanks to the fifth transformation rule (Rule 5), we get two flow arcs leaving a place *WaitingForArrivalOfBagsAtCarousel* that are weighted with probabilities (0.2 and 0.8). The complete model can be visualized and edited with ZIZO and then probabilistic properties can be verified with PRISM. For example, to verify the confluence property it is possible to check that only one of the two scenarios is chosen at the same time. Let $x = 1$ be the state in PRISM code (which is based on a Markov chain model) indicating that the controller has to process baggage as planned and $x = 2$ be the state indicating the scenario issued by delays. The

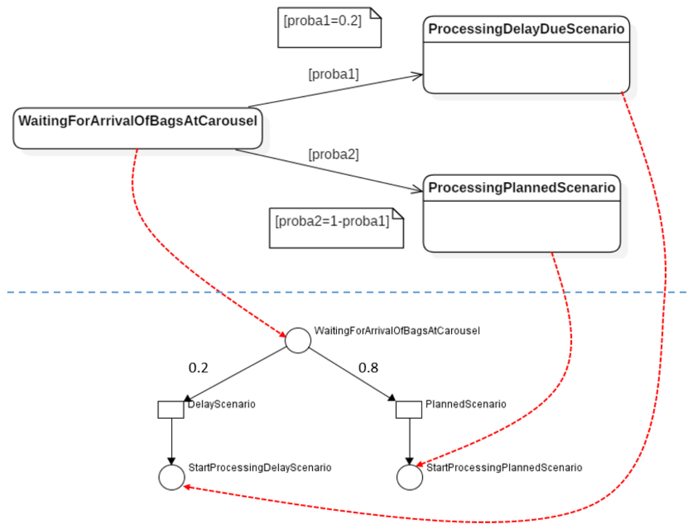

[proba1=0.2]
ProcessingDelayDueScenario
[proba1]
WaitingForArrivalOfBagsAtCarousel
[proba2]
ProcessingPlannedScenario
[proba2=1-proba1]
WaitingForArrivalOfBagsAtCarousel
0.2
0.8
DelayScenario
PlannedScenario
StartProcessingDelayScenario
StartProcessingPlannedScenario

Figure 4.20: Probabilistic part of the statechart of the controller.

verification of the following formula returns zero then never the two scenario will take place simultaneously.

$$P =?[F\ x = 1\ \&\ x = 2] \tag{4.11}$$

For further probabilistic analysis, and given that the BHS has a planned schedule where paths (i.e., conveyor sections) are reusable, we need to estimate the successful re-routing that might unpredictably take place (because of failures, or delays, etc.) and which gives rise to the need to handover bags from one path to another. Let us focus on the part which has the responsibility to analyze the efficiency of probable rerouting which is included in the composite state *ProcessingDelayDueScenario* as mentioned in Figure 4.21.

For simplicity reasons, let us assume that we have three successive paths P1, P2, and P3, and that P1 has only the ability to handover bags to P2, P2 can handover bags to P1 and P3, and P3 can handover bags to P2. Let us also assume that the capacity of a path is expressed in terms of bags weight.

In order to analyze the efficiency of rerouting, and after transforming the model to PRISM, we can apply PCTL formulas. Figure 4.22 shows the PRISM code used to analyze the routing analysis task, in which the probability of delay occurrence in the path P1 is once per week $p1 = 1/(7 * 24 * 3600)$, in P2 is twice per week $p2 = 2/(7 * 24 * 3600)$, and in P3 is three times per week $p3 = 3/(7 * 24 * 3600)$, also the probability to reroute bags from P1 to P2 is $p11 = 0.5$, from P2 to P3 is $p21 = 0.45$, from P2 to P1 is $p22 = 0.45$, and from P3 to P2 is $p31 = 0.5$.

As mentioned in Figure 4.22, the state $s = 0$ represents the idle state, $s = 1$ (respectively $s = 2, s = 3$) represents a delay in P1 (respectively P2, P3). The state $s = 11$ represents the final state. The rest of states are defined as follows,

- $s = 4$ means that bags are handed-over from P1 to P2 and a reconfiguration

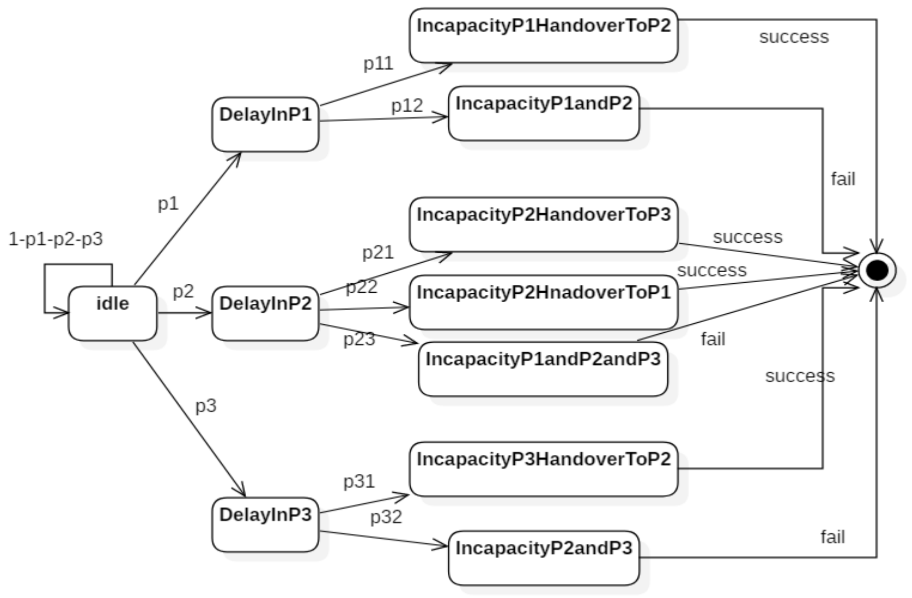


Figure 4.21: Probabilistic part of the statechart of the controller.

is successful (using different configuration for example with more speed, with activating deactivating merging and diverting pushers, etc.).

- $s = 5$ means that bags cannot be handed-over from P1 to P2 and a reconfiguration is failed.

- $s = 6$ means that bags are handed-over from P2 to P3 and a reconfiguration is successful.

- $s = 7$ means that bags are handed-over from P2 to P1 and a reconfiguration is successful.

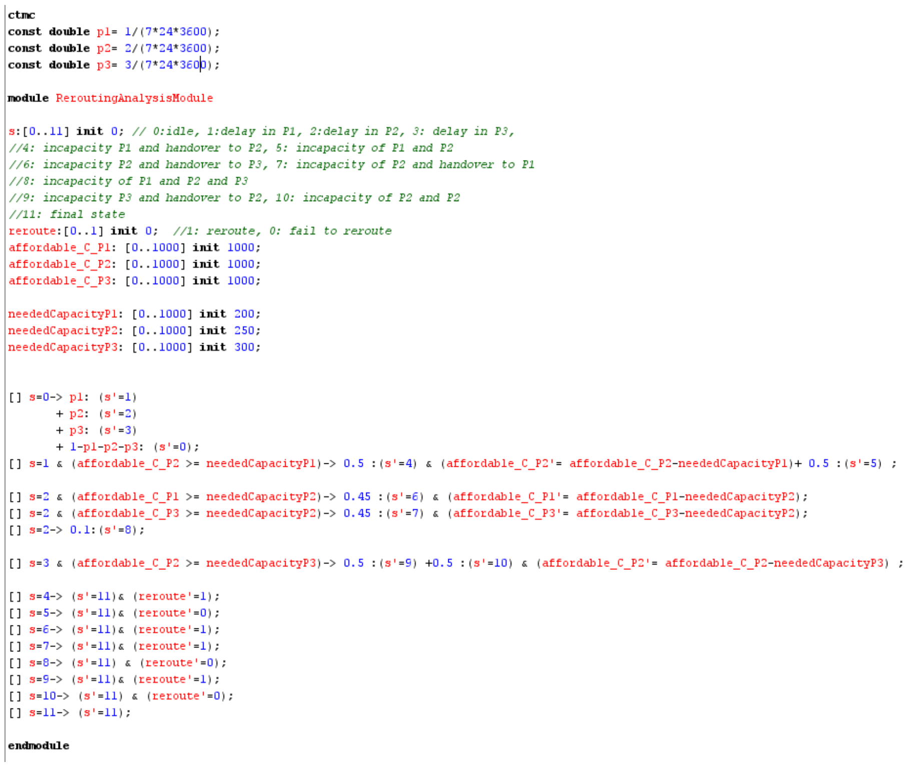
- $s = 8$ means that bags cannot be handed-over from P2 to P1 nor to P3 and a reconfiguration is failed.

- $s = 9$ means that bags are handed-over from P3 to P2 and a reconfiguration is successful.

- $s = 10$ means that bags cannot be handed-over from P3 to P2 and a reconfiguration is failed.

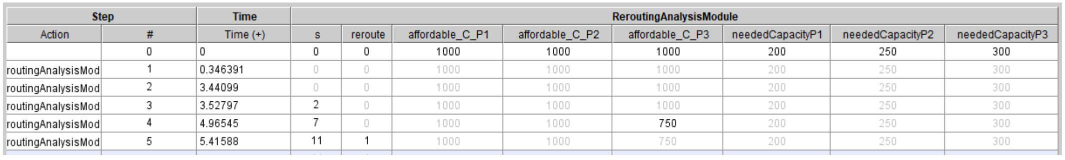
As first step, a simulation of the model is carried-out (see Figure 4.23).

Then verifying formulas can be done. First, we verify that the rerouting module does not contain any deadlock. This is achieved via the formula $E[F"deadlock"]$ which is proven to be false, i.e., the module is deadlock free (see Figure 4.24).



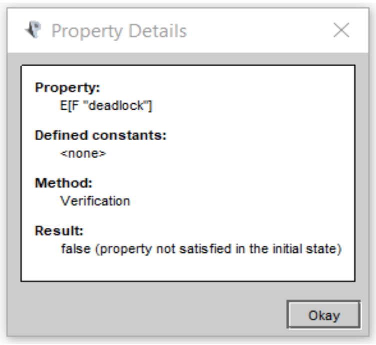
```
ctmc
const double p1= 1/(7*24*3600);
const double p2= 2/(7*24*3600);
const double p3= 3/(7*24*3600);

module ReroutingAnalysisModule

s:[0..11] init 0; // 0:idle, 1:delay in P1, 2:delay in P2, 3: delay in P3,
//4: incapacity P1 and handover to P2, 5: incapacity of P1 and P2
//6: incapacity P2 and handover to P3, 7: incapacity of P2 and handover to P1
//8: incapacity of P1 and P2 and P3
//9: incapacity P3 and handover to P2, 10: incapacity of P2 and P2
//11: final state
reroute:[0..1] init 0;  //1: reroute, 0: fail to reroute
affordable_C_P1: [0..1000] init 1000;
affordable_C_P2: [0..1000] init 1000;
affordable_C_P3: [0..1000] init 1000;

neededCapacityP1: [0..1000] init 200;
neededCapacityP2: [0..1000] init 250;
neededCapacityP3: [0..1000] init 300;


[] s=0-> p1: (s'=1)
       + p2: (s'=2)
       + p3: (s'=3)
       + 1-p1-p2-p3: (s'=0);
[] s=1 & (affordable_C_P2 >= neededCapacityP1)-> 0.5 :(s'=4) & (affordable_C_P2'= affordable_C_P2-neededCapacityP1)+ 0.5 :(s'=5) ;

[] s=2 & (affordable_C_P1 >= neededCapacityP2)-> 0.45 :(s'=6) & (affordable_C_P1'= affordable_C_P1-neededCapacityP2);
[] s=2 & (affordable_C_P3 >= neededCapacityP2)-> 0.45 :(s'=7) & (affordable_C_P3'= affordable_C_P3-neededCapacityP2);
[] s=2-> 0.1:(s'=8);

[] s=3 & (affordable_C_P2 >= neededCapacityP3)-> 0.5 :(s'=9) +0.5 :(s'=10) & (affordable_C_P2'= affordable_C_P2-neededCapacityP3) ;

[] s=4-> (s'=11)& (reroute'=1);
[] s=5-> (s'=11)& (reroute'=0);
[] s=6-> (s'=11)& (reroute'=1);
[] s=7-> (s'=11)& (reroute'=1);
[] s=8-> (s'=11) & (reroute'=0);
[] s=9-> (s'=11)& (reroute'=1);
[] s=10-> (s'=11) & (reroute'=0);
[] s=11-> (s'=11);

endmodule
```

Figure 4.22: Prism code of the rerouting analysis module.



Step		Time	ReroutingAnalysisModule							
Action	#	Time (+)	s	reroute	affordable_C_P1	affordable_C_P2	affordable_C_P3	neededCapacityP1	neededCapacityP2	neededCapacityP3
	0	0	0	0	1000	1000	1000	200	250	300
routingAnalysisMod	1	0.346391	0	0	1000	1000	1000	200	250	300
routingAnalysisMod	2	3.44099	0	0	1000	1000	1000	200	250	300
routingAnalysisMod	3	3.52797	2	0	1000	1000	1000	200	250	300
routingAnalysisMod	4	4.96545	7	0	1000	1000	750	200	250	300
routingAnalysisMod	5	5.41588	11	1	1000	1000	750	200	250	300

Figure 4.23: Prism module simulation.



Property Details

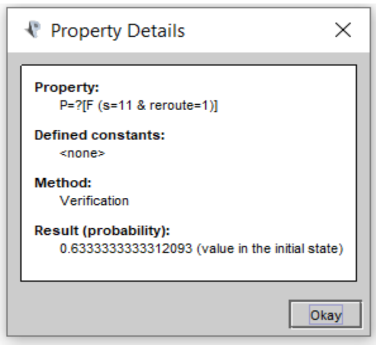
Property:
E[F "deadlock"]

Defined constants:
<none>

Method:
Verification

Result:
false (property not satisfied in the initial state)

Figure 4.24: Deadlock property.



Property Details

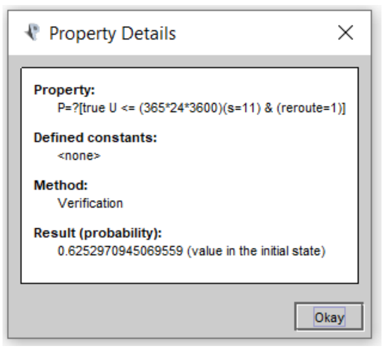
Property:
P=?[F (s=11 & reroute=1)]

Defined constants:
<none>

Method:
Verification

Result (probability):
0.6333333333312093 (value in the initial state)

Figure 4.25: Successful rerouting.



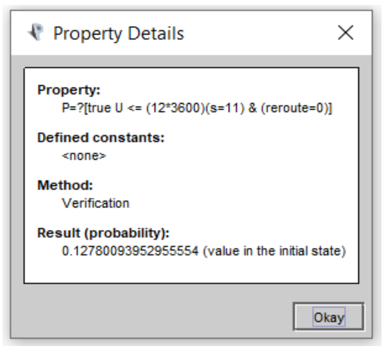
Property:
P=?[true U <= (365*24*3600)(s=11) & (reroute=1)]

Defined constants:
<none>

Method:
Verification

Result (probability):
0.6252970945069559 (value in the initial state)

Figure 4.26: Success for one year.



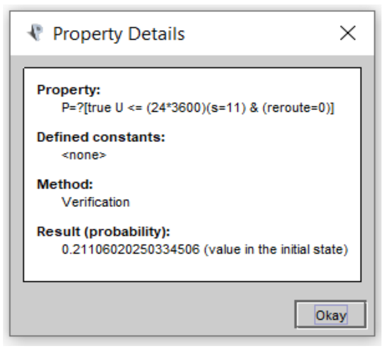
Property:
P=?[true U <= (12*3600)(s=11) & (reroute=0)]

Defined constants:
<none>

Method:
Verification

Result (probability):
0.12780093952955554 (value in the initial state)

Figure 4.27: Failure for next 12 hours.



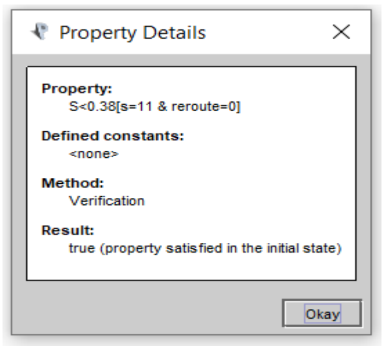
Property:
P=?[true U <= (24*3600)(s=11) & (reroute=0)]

Defined constants:
<none>

Method:
Verification

Result (probability):
0.21106020250334506 (value in the initial state)

Figure 4.28: Failure for one day.



Property:
S<0.38[s=11 & reroute=0]

Defined constants:
<none>

Method:
Verification

Result:
true (property satisfied in the initial state)

Figure 4.29: Long running failure.

Then, we want to know what is the probability of reaching a state where the rerouting is successful. Thus, we verify this formula $P=?[F(s=11\&reroute=1)]$ which returns 0.633 (see Figure 4.23), this means that in 63% of all cases we can have successful rerouting.

We verify the successful reconfiguration (i.e., rerouting) for one year via the formula $P=?[trueU <= (365*24*3600)(s=11)\&(reroute=1)]$ which returns 0.62 (see Figure 4.26). We also estimate the probable failure for next 12 hours (see Figure 4.27) and next day (see Figure 4.28).

We verify also, in the long run, the chance that failed reconfigurations are met is less than 0.38 using the long-run (steady-state) probabilistic operator S in the following formula $S < 0.38[s=11\&reroute=0]$ (see Figure 4.29).

After analyzing a set of properties, system designers can made assumptions about the satisfactory level they need. Models can be refined and its properties can be re-verified.

In this phase, all desired behaviors could be verified and model changes can take place whenever necessary. Once the considered model is analyzed and it is considered to be robust, then a move to next step becomes possible.

4.8.3 Phase 3 of the Methodology: Model Screening According to IEC 61499 Using a Function Block Tool

After checking the models of the controller, services, and the whole application with formal verification, it is now possible to move to the next type of "verification", where the models are transformed to IEC 61499 function blocks and a simulation on a particular platform becomes possible, for example using Raspberry-SPS as I/O, specific communication protocols such as HTTP, Modbus, or any other hardware provided by function block simulators.

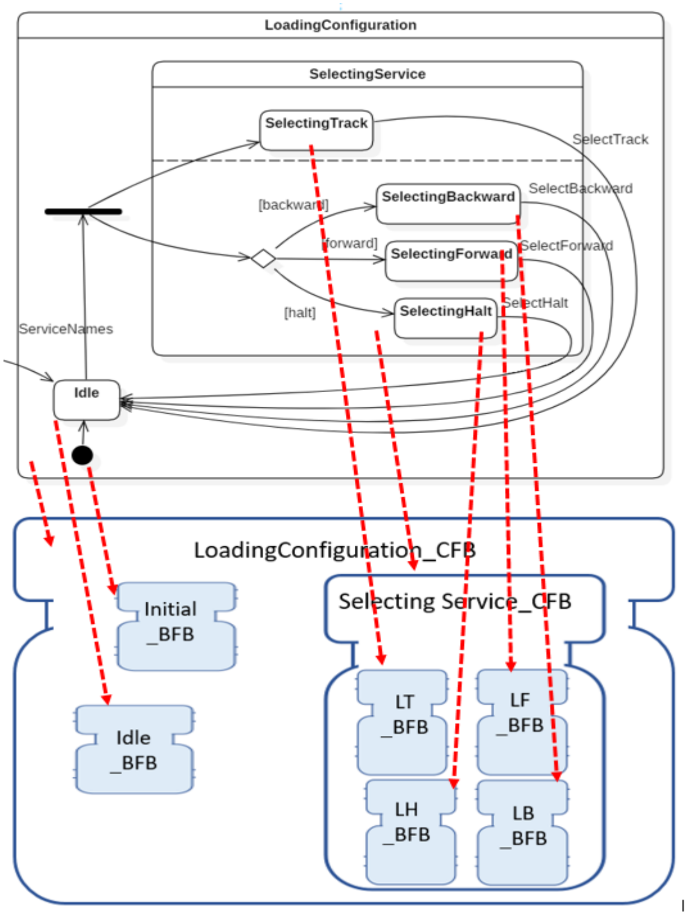


Figure 4.30: Transformation of statechart diagram of the controller loading service subtask to function blocks.

In this example, 4DIAC/FORTE is adopted to explore the deployment of the application. As an example, we continue to show how the model of the controller subtask, which is the *Loading Service*, is obtained. According to Rule 14, the composite state *LoadingServices* is transformed to a composite function block, i.e., the function block *LoadingServices* in Figure 4.30. The state *SelectingService* is transformed to a composite function block as orthogonal states are composite states where regions are supposed to work in parallel. The initial, idle, *LoadingForward*,

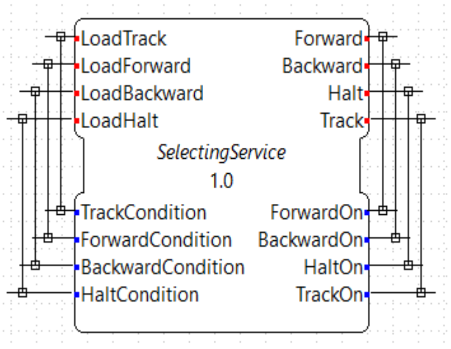


Figure 4.31: Selecting Service composite function block interface.

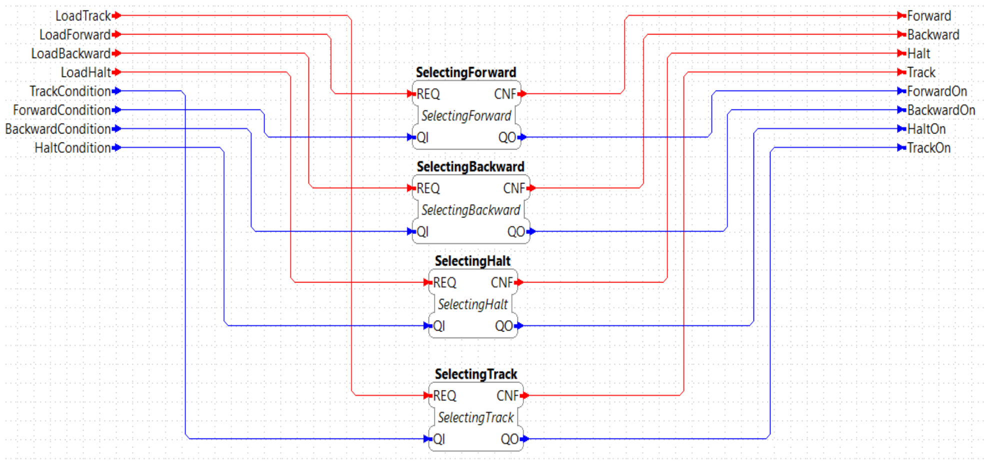


Figure 4.32: Internal composition of the Selecting Service function block.

LoadingBackward, *LoadingTrack*, and *LoadingHalt* states are transformed to basic function blocks called respectively *Initial-BFB*, *Idle-BFB*, *LF-BFB*, *LB-BFB*, *LT-BFB*, and *Starter-LH*.

Figure 4.31 and Figure 4.32 show the obtained interface as well as the internal composition of the composite function block *SelectingService*.

Applications models could then be analyzed: composition, encapsulation, and modular structures can be examined. Additionally, simulation on a target hardware environment, such as FORTE PC provided by 4DIAC using Ethernet for communication could take place.

Function blocks models analysis and refinement can be performed until obtaining the desired results.

Finally: At the end of the three phases, we obtain a reliable application model and implementation becomes straightforward. Implementing applications starts in basis of the provided framework code.

4.9 Comparison with Other Approaches

In this section, we compare the proposed modeling methodology with existing ones, where we highlight the advantages of our methodology. We build our comparison according to two scopes: (1) the modeling, and (2) the verification.

Concerning the modeling, three main methodologies are recognized for modeling control systems (such as goods handling systems, microgrid, etc.) requiring the intelligence and distribution:

First method: is to start the modeling of applications using function blocks, such as the works reported in [Yan & Vyatkin 2013]. In this approach, abstract models are not provided, even though an attempt to model the controller is made through defining two software elements, High Level Control (HLC) and Low Level Control (LLC), where HLC refers to the integration of software agent technology. Instead, the use of UML/GR-UML may provide promising results especially that it is considered the semi-formal software modeling language.

Second method: it consists in the use of UML only for clarifying the process of the development not the application itself, as stated in the research reported in [Panjaitan & Frey 2006]. As IEC 61499 does not provide complete development method, researchers have tried to use UML for better capturing requirements and better model component-based systems.

Third method: it consists in applying software design patterns, such as the Model/View/Control pattern, and software design approaches, such as the Multi-Agent Systems in combination with the function blocks to model systems, for example the work presented in [Black & Vyatkin 2009]. Again, this methodology does not provide the way to explore the different scopes of applications in comparison with the use of UML, rather it helps to model the software according the physical infrastructure of a system.

Despite the assets of the mentioned methodologies, it is still required to have the ability to provide applications UML models, especially to get a model that captures probable requirements and provides its abstract representation as well as the tool to its development.

Concerning the verification, there are works that use simulation of function blocks as testing, others simulate on testbeds, and some others transform function block model to Petri nets (NCES extensions). The problem with simulation, whether in some prototype testbed or emulated platform through simulation tools, is that results are tied to the considered scenarios, absolute guarantees cannot be obtained. Formal verification of function blocks model may not cover all the application features/functionalities, since it may be that some parts/components could not be modeled with function blocks (e.g. intelligence mechanisms), consequently verification will be made only on function blocks part, but we often need to get guarantees about an application as a whole. Besides, to the best of our knowledge, our methodology is the first to consider the three modeling scopes and with a software tool support.

In conclusion, given the limitations of the methodologies above-mentioned, our methodology is introduced to provide better process through allowing modeling using the conventional semi-formal modeling language of software systems (UML/GR-UML), to enable formal verification for all applications parts, and to explore software analysis in compliance with function block platforms, therefore to get better application's models before implementation.

4.10 Conclusions

This chapter has presented a modeling methodology for clear, correct, and operable models of controlling software applications of context-aware reconfigurable distributed systems. The chapter has also introduced a new UML profile called Generalized Reconfigurable UML (GR-UML). In addition, it has introduced a new software tool implementing the methodology concepts.

The methodology consists of three phases, applications modeling starting from the framework models and using GR-UML semantics, formally analyzing the behavior of applications and mainly the controller and services ones, and finally analyzing the models in target hardware environment in compliance with IEC 61499 standard. The methodology helps designers and developers to obtain reliable applications models before starting the implementation step, hence it helps in catching modeling errors and ambiguities. The process flow of the methodology offers flexibility to developers through enabling the iterative modifications and refinements. This process is made easier thanks to a set of transformation rules facilitating the transformation of UML models to GR-TNCES and function blocks models. We applied the contributions of this chapter to the controlling software application of BHS presented in Chapter 3.

Chapter 5

Security of a Network of Frameworks

Contents

5.1 Introduction

Traditional security techniques rely heavily on the concept of a trust third party and authorized agents to provide security solutions. Emerging security technologies are revolutionizing the field through introducing new concepts that get rid of classic limitations, namely the blockchain technology. In this context and with the undeniable need to security for computerized systems especially distributed ones, this chapter introduces a security protocol for transparent, privacy-preserving, and distributed communication between distributed peers. The protocol builds upon blockchain technology and the elliptic curve cryptography.

The outline of the chapter is organized as follows: First, the background and motivation of the protocol are presented. Then, the protocol concepts, phases, and mechanisms are defined. Lastly, an example of use of the proposed protocol in an electricity trading scenario among microgrids is depicted.

The contributions of this chapter are summarized as follows:

- Ensuring secure communication between distributed peers in trustless environment.
- Providing a solution to the privacy limitation of blockchain technology.
- Defining cost-effective solution to the scalability challenge inherent to the blockchain.
- Defining a method to calculate the confirmation time for the proof-of-work based blockchains.

5.2 Motivation

Establishing trust between communicating distributed parts of a same reconfigurable automation and control system is a paramount condition to ensure reliability and efficiency. Traditionally, this is provided by a trust third party (TTP) that plays the role of a trusted entity which two communicating parts entrust. The TTP verifies, on behalf of the relying parties, that transactions are safe against any malicious manipulation. Many application cases of this security paradigm are set up through a certificate authority. However, despite the assets of this approach, the risk of fraudulent actions and malicious use of relying parties' data is still there. In fact, the legitimacy and authority of this "trusted" part must be always reviewed, otherwise there is no guarantee that responsible parts may misuse data. Moreover, centralized techniques have an inherent risk of single point of failure, so an attack on TTP means a risk to leak/ endanger any user relying on it.

In this context, blockchain technology was introduced to overcome the need to use TTPs. It relies on a distributed ledger for transactions storage along with a consensus mechanism ensuring the validity of the data. Blockchain also guarantees an open access to all its users, in fact it is possible to any member in the network to read and consult any stored transaction. Hence it is a great tool for transparency insurance especially for applications where no trust can exist between users. This also can help in reducing costs spent for administration and infrastructure requirements. More importantly, blockchain provides a great solution for immutability since once a transaction is recorded it can never be deleted or even modified.

Given these assets, blockchain technology was introduced in many applications, especially in the economic field. Nevertheless, few are the works that study/apply its concepts in the automation and control fields. It has to be said that this technology is still in its infancy phase and many questions and challenges need to be alleviated, but its high potential to provide promising solutions also for intelligent reconfigurable systems lead us to propose a conversation model among peers that builds upon blockchain. In the proposed framework, we always try to benefit of the advances achieved in the software realm (context-awareness, artificial intelligence, blockchain based cyber security) to make reconfigurable systems smarter and more efficient.

As discussed in Chapter 2, blockchain technology is challenged with two main limitations: privacy and scalability. Concerning the privacy, the use of blockchain implies the revealing of identity. This fact can be problematic for many cases because of privacy issues. In fact, transparency is important but this can be in the same time a privacy threat as malicious tracking and use of others information can lead to dangerous consequences. From another side, immutability is also great but adding transactions without ever deleting anything gives rise to scalability problems. Storage in memory must be optimized in order to ensure long-living applications with mature blockchain.

5.3 Framework Security Technique

As mentioned in the defintion of the Security Module SM of the Context Control Layer (CCL), many security techniques (such as the encryption, certification, etc.) can be defined. In this thesis, we focus on the use of the secure conversation protocol based on a blockchain technique.

5.3.1 Composition of the Blockchain Technique (Security Module)

In order to use the proposed protocol that draws upon blockchain, we need to use different elements. Figure 5.1 depicts an overview of the class diagram of the elements necessary to use for the proposed protocol. We distinguish between classes related to the functioning of the security module through the package *Security-ModuleInteraction*, classes related to the protocol process through the package *ProtocolTransactionProvider*, and classes related to the operation of the blockchain through the package *BlockchainTools*.

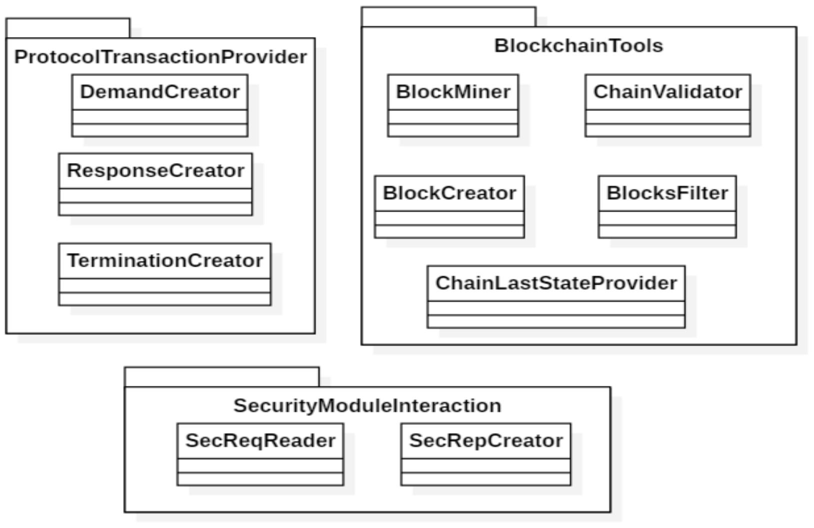


Figure 5.1: Composition overview of the proposed security technique.

5.3.2 Dynamics of the Blockchain Technique (Security Module)

The blockchain security technique component of the SM has mainly four behaviors: (1) create a protocol demand, (2) create a protocol response, (3) create a protocol termination, and (4) participate in the mining of others blocks. Figure 5.2 depicts a UML statechart diagram that describes the aforementioned behaviors.

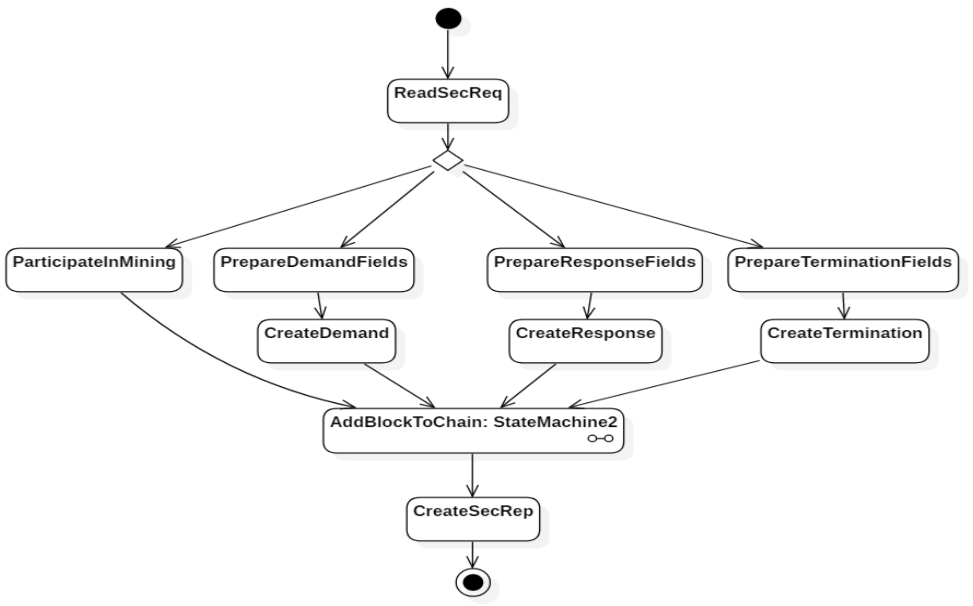


Figure 5.2: Proposed security technique behavior.

The first thing to do in the states related to the protocol phases is to prepare the content of each transaction then to create the transaction itself, finally put it in a block and publish it in the blockchain. The block addition has the same steps for all the behaviors, this is why we separate it in a reusable sub-machine state *AddBlockToChain*. This sub-machine is detailed in Figure 5.3.

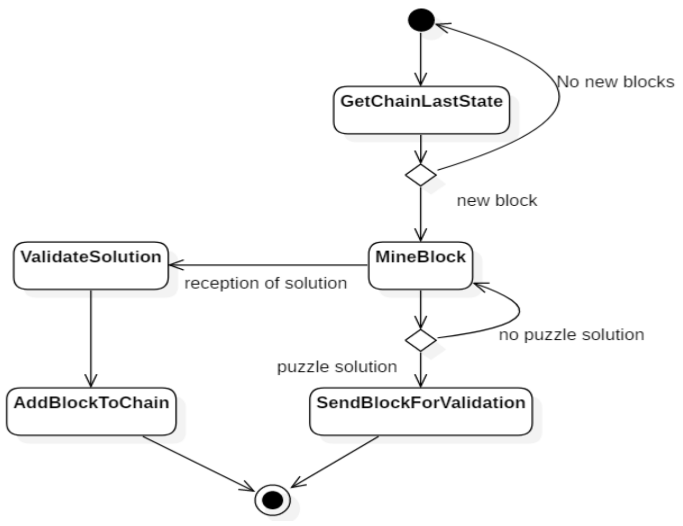


Figure 5.3: Block addition to chain behavior.

5.4 Secure Conversation Among Distributed Peers Protocol

We propose a secure conversation protocol to be used by distributed peers having software applications developed by the proposed framework at the aim of enabling secure exchange of data in the context of coordination or other communicative operations (e.g. trading process as it is mentioned later in the case study example). The peers are assumed to be equal (i.e., equal hierarchy, equal rights, and equal permissions).

Target: The protocol is devoted to group's interactions where in order to provide a service, "demanders" aggregate their demands in one single demand and "responders" may provide separate responses or also may aggregate their responses in one response. Such type of interaction can be required in many cases of the e-commerce such as the whole sale trading, electricity trading, e-auction, etc.

Overview: The protocol is composed of three phases:

- Phase 1 called "Demand Phase" in which peers needing to discuss/negotiate about one operation and/or coordination process initiate the conversation.
- Phase 2 called "Response Phase" in which other peers answer to the requests made in the first phase. The proposed protocol provides "responders" to make replies/responses in two different communication ways:

1. in the first way, **the response** is made by a set of peers that can satisfy the demand. These responding peers also unite their responses in a single reply.
2. in the second way, **the response** is made by individual peers (i.e., no gathering of replies).

- Phase 3 called “Termination Phase” in which the decisions/ measures issued from the conversation must take place.

Figure 5.4 depicts a process flow of the proposed protocol. The transactions made in each phase can be defined in a away to fit particular use cases as we detail in the running example. The protocol provides the infrastructure upon which coordinations/negotiations can be built.

Before moving to the process of the protocol, it is worth noting that peers need to arrange and agree about the ECIES setup in order to properly use key sizes, versions, etc. In the following sections, the symbol “||” denotes a concatenation.

In order to avoid the controversial property of blockchain, which is the revealing of peers identity, we propose to create identifier of transactions rather than its sources. For this, peers are asked to create a key pair of ECIES scheme and use the public key as a transaction identifier. This operation must be re-performed with each new transaction. Thus, we ensure also the avoidance of any possible binding of a set of transactions to one specific peer.

5.4.1 Protocol First Phase: Demand

This is the first phase of the protocol, where peers that have the need to a certain service gather their demands efficiently by means of the blockchain. This phase is composed of three steps: initial demand, gathering demands, and final demand. The technical description of each step is provided in the following.

5.4.1.1 Initial Demand

In this step, a peer A that needs a specific service (the service must be defined/agreed upon according to the use case) starts by publishing a demand in the blockchain. A demand is denoted by d and given by

$$d_i = (\lambda_i || content_i). \tag{5.1}$$

where λ_i is the identifier of d_i and it is the public key of an ECIES key pair, and $content_i$, as its name indicates, is the content of the demand. In order to identify new demands and not their source, i.e., peer A, new λ_i is newly created for each new demand. This way we preserve the anonymity of A and avoid possible malicious binding of demands.

The submission of d_i by peer A in the blockchain denotes the need of A to start a conversation with the rest of peers in the network.

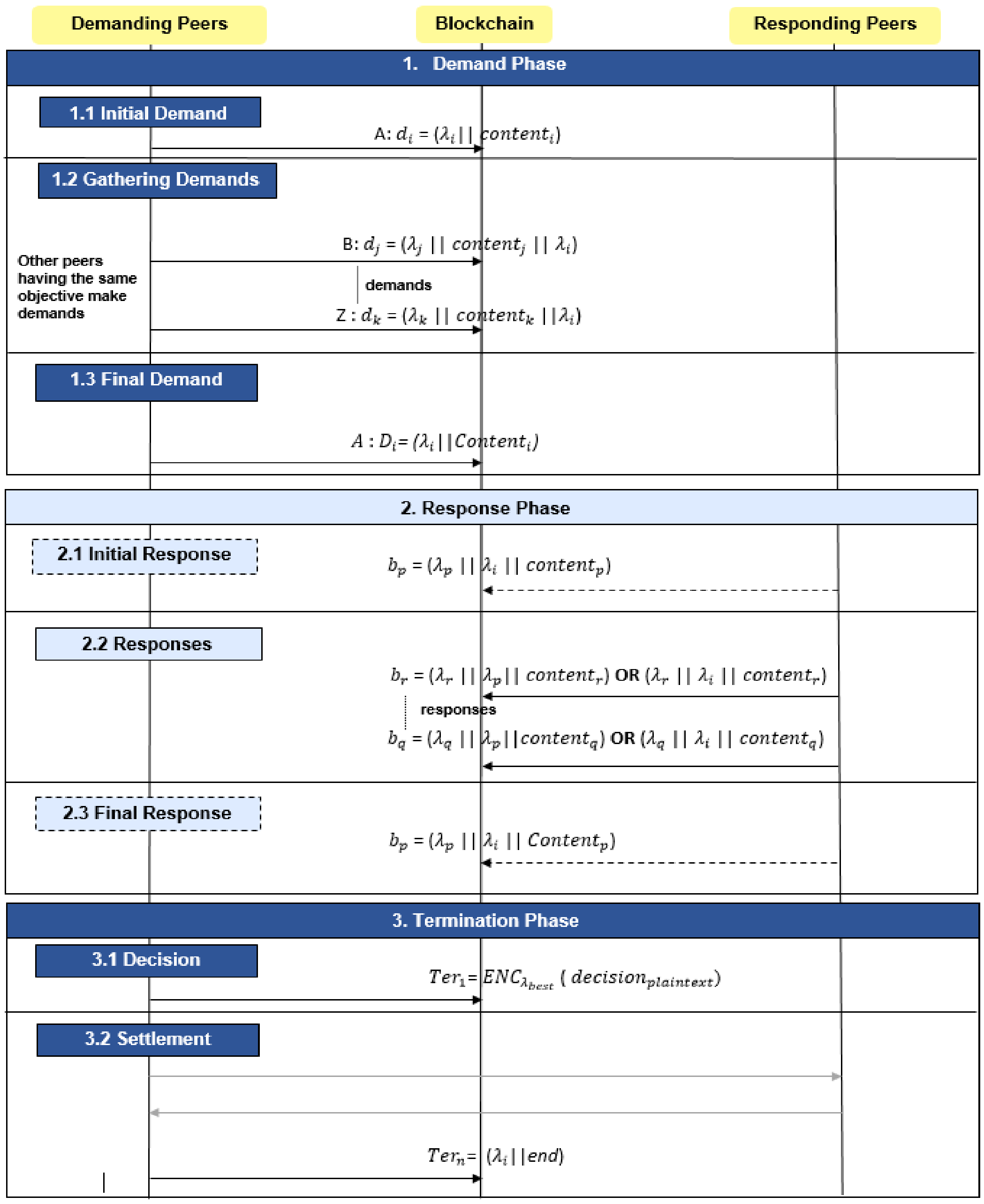

Demanding Peers
Blockchain
Responding Peers
1. Demand Phase
1.1 Initial Demand
A: $d_i = (\lambda_i || content_i)$
1.2 Gathering Demands
B: $d_j = (\lambda_j || content_j || \lambda_i)$
Other peers having the same objective make demands
demands
Z: $d_k = (\lambda_k || content_k || \lambda_i)$
1.3 Final Demand
$A : D_i = (\lambda_i || Content_i)$
2. Response Phase
2.1 Initial Response
$b_p = (\lambda_p || \lambda_i || content_p)$
2.2 Responses
$b_r = (\lambda_r || \lambda_p || content_r)$ OR $(\lambda_r || \lambda_i || content_r)$
responses
$b_q = (\lambda_q || \lambda_p || content_q)$ OR $(\lambda_q || \lambda_i || content_q)$
2.3 Final Response
$b_p = (\lambda_p || \lambda_i || Content_p)$
3. Termination Phase
3.1 Decision
$Ter_1 = ENC_{\lambda_{best}}(decision_{plaintext})$
3.2 Settlement
$Ter_n = (\lambda_i || end)$

Figure 5.4: Conversation protocol using Blockchain and ECIES.

5.4.1.2 Gathering Demands

Once d_i is added to the blockchain, it becomes readable by the rest of the peers in the network. Then, any peer that is concerned with the conversation and has the same need as A must create its demand which has the following structure,

$$d_j = (\lambda_j || content_j || \lambda_i). \tag{5.2}$$

where λ_j is the identifier of d_j and it is created by its owner similarly as λ_i of A, λ_i is the identifier of the initial demand and it is added in d_j to indicate that it is related to d_i, and $content_j$ is the content of the answer.

The gathering step lasts a period of time that must be defined by the system owners according to the requirements of the use case.

5.4.1.3 Final Demand

After a gathering time has elapsed, A which is the peer that has initiated the conversation process, must read all submitted demands and create a final demand D_i which has this form

$$D_i = (\lambda_i || Content_i). \tag{5.3}$$

where λ_i is reused to identify the final demand, and $Content_i$ is the final demand tenor.

The logic indicating how A should create $Content_i$ is a separate concern that is related to the use case, and since we are focusing on the procedural security aspect of the protocol, this logic is left proprietary.

5.4.2 Protocol Second Phase: Response

This is the second phase of the protocol, where peers that have read the final demand D_i and that have the ability to satisfy the request can make responses and post them in the blockchain.

This phase is composed of three steps where the first and last are optional steps, while the second step is obligatory. The flow of this phase can be accomplished in two ways depending on the requirements of the use case: **(1) collective responses:** performed by means of the three steps : initial response, responses, and final response, **(2) individual responses:** performed by means of one step, the obligatory one, which is the responses step.

5.4.2.1 Initial Response

If the first scenario, the collective responses scenario, is adopted by system owners, then the second phase starts with this step.

Once D_i is added to the blockchain, it becomes readable by the rest of the peers in the network. Then, any peer that is concerned with the conversation and wants to answer the demand, must create a reply, denoted by a_p and is defined as

$$b_p = (\lambda_p || \lambda_i || content_p). \tag{5.4}$$

where λ_p is the identifier of b_p and it is created by its owner similarly as λ_i of A, λ_i is the identifier of the initial demand and it is added in b_p to indicate that it is related to D_i, and $content_p$ is the content of the answer.

The submission of the first reply constitutes the first step and the initiator peer gets the responsibility of following and terminating the whole phase.

5.4.2.2 Responses

In this step, any peer that has the ability to satisfy the demand D_i issued from the first phase can create a response and add it to the blockchain. The response can have two structures according to the two responding scenarios.

For the collective responses, where responses are directed to the initiator peer, a response is defined as

$$b_r = (\lambda_r||\lambda_p||content_r). \tag{5.5}$$

where λ_r is the identifier of b_r and it is created by its owner similarly as λ_i of A, λ_p is the identifier of the initial response and it is added in b_r to indicate that it is related to b_p, and $content_r$ is the content of the answer.

For the individual responses, where responses are directed to A peer, a response is defined as

$$b_r = (\lambda_r||\lambda_i||content_r). \tag{5.6}$$

where λ_r is the identifier of b_r and it is created by its owner similarly as λ_i of A, λ_i is the identifier of the final demand and it is added in b_r to indicate that it is related to D_i, and $content_r$ is the content of the answer.

This step starts from the moment when b_p is added to the blockchain and lasts a period of time specified by the systems owners.

5.4.2.3 Final Response

If the first scenario, the collective responses scenario, is adopted by system owners, then the second phase ends with this step.

After the responding time is elapsed, the initiator peer must read all the submitted responses and creates a final response having this form

$$b_p = (\lambda_p||\lambda_i||Content_p). \tag{5.7}$$

The logic indicating how $Content_p$ can be created must also be defined by the owners according to the use case.

5.4.3 Protocol Third Phase: Termination

After the demands and responses phases are performed, we come finally to the last phase, in which decisions can be made. This phase is composed of two steps: decision and settlement.

5.4.3.1 Decision

To close the conversation, since A is the initiator of the conversation, it is the one who submits a decision based on the replies provided in the previous phase.

Decision's specific logic must be defined separately according to the use case. Then, once a decision is made, its value is presented in a field called $decision_{plaintext}$, encrypted using the ECIES encryption scheme, and submitted in the blockchain as follows

$$Ter_1 = ENC_{\lambda_{best}}(decision_{plaintext}). \tag{5.8}$$

where $ENC(.)$ is the encryption function of the ECIES scheme that uses the public key of the best response λ_{best} to encrypt the decision plaintext $decision_{plaintext}$.

Once the decision is added to the blockchain, all peers that have added replies (all peers in case of individual responses and first peers of collective responses) must try to decrypt the decision. Here, only one peer succeed to decrypt the cryptogram which is the one having the private key associated to the public key that is chosen by A to encrypt the decision. Hence, the fact that a decision have been made is known by all the blockchain peers but the identity of the chosen peer is kept anonymous and details of the decision are kept secret. Therefore, we preserve privacy of the chosen responding peer and the decision details.

5.4.3.2 Settlement

In this step may include multiple interactions inside and outside of the blockchain depending on the use case. The most important thing is that at the end A must submit a transaction mentioning the end of the conversation defined as follows:

$$Ter_n = (\lambda_i || end). \tag{5.9}$$

where λ_i is used by A to indicate that this transaction is related to the first demand, and *end* as its name indicates, it is a field/flag mentioning the end of the conversation.

5.5 Protocol Implementation

In this section, the implementation of the protocol and the necessary technologies are presented.

5.5.1 Distributed Messaging Platform

In order to allow the implementation of the secure conversation protocol, the first thing to provide is the communication platform or the messaging technique. This part can be ensured by any distributed technique selected by system developers.

The necessary thing is to prepare the connection between the chosen technology and the messaging module of the framework.

In the case study (to be presented in detail in the following section), the Java Agent Development Framework (JADE) framework is adopted to simulate the protocol usage scenario. JADE [1] allows to simulate the communication among distributed peer using messages. It provides the possibility for execution in different hardware systems comprising embedded ones such as Raspberry Pies and supporting various operating systems such as Android, Windows, and Linux.

5.5.2 Private Blockchain Implementation

A private blockchain is implemented to be the base of the conversation protocol. Microgrids software is implemented using the proposed framework where the security pool contains the methods necessary to use the trading protocol. In order to simulate the network of microgrids, the instances of the framework (i.e., microgrids software) are hosted in a set of virtual machines created on a VirtualBox[2].

In this thesis, blockchain is used as secure distributed communication and storage ledger, not for payment or crypto-currencies. The implementation is performed using also Java programming language and a set of libraries such as Bouncy Castle [3] are used for cryptography.

Chaining: The blocks are chained chronologically through calculating the hash of the new block based on the hash of the previous block. This chained hash represents the digital signature of a block and it is created using the SHA-256 algorithm. The structure of blocks is depicted in Figure 5.5.

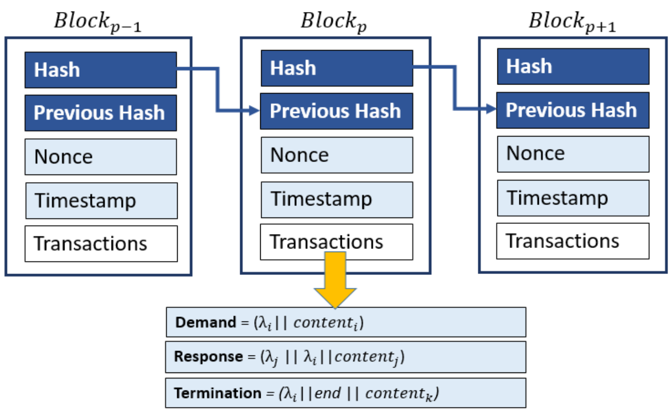

$Block_{p-1}$
$Block_p$
$Block_{p+1}$
Hash
Previous Hash
Nonce
Timestamp
Transactions
Demand = $(\lambda_i \,||\, content_i)$
Response = $(\lambda_j \,||\, \lambda_i \,||\, content_j)$
Termination = $(\lambda_i \,||\, end \,||\, content_k)$

Figure 5.5: Blocks data structure.

Consensus: The validity of the chain is ensured through the Proof-of-Work (PoW) consensus. This consensus mechanism guarantees that every node (i.e., framework) spends enough time and computational effort to add a block in the

[1] https://jade.tilab.com/
[2] https://www.virtualbox.org/
[3] https://www.bouncycastle.org/documentation.html

blockchain. Here it is assumed that all nodes are miners. All nodes must solve a difficult puzzle before adding a block to the chain, the puzzle in this thesis is finding a hash that starts with five zeros “00000”. In every new attempt to calculate a hash, a new random value, the nonce, is modified. In this paper the nonce is an integer incremented by one.

Confirmation Time: Defining confirmation times of a blockchain is an important step that facilitates the analysis and evaluation of use cases efficiency. Figure 5.6 depicts a graphical presentation that help us explain the calculation of the proposed blockchain confirmation time.

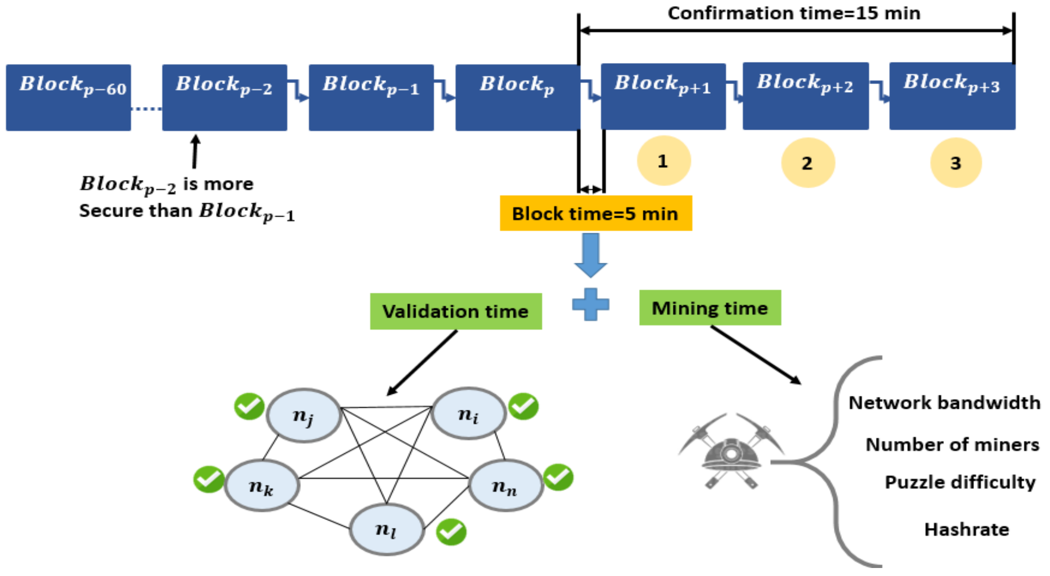


Figure 5.6: Blockchain confirmation time.

Before starting the explanation, it is worth to note that the calculation of the confirmation time is not yet standardized and in this thesis we propose the following method to calculate it.

Preliminaries: we define the terminology use in the calculation of the confirmation time as follows:

- *Average Mining Time:* denoted by x_{amt}, is the time taken by a node to solve the puzzle. This time slot depends on many factors such as the hashrate (hardware power), the number of nodes in the network, the puzzle difficulty (5 in this thesis), and the network infrastructure (e.g., bandwidth).

- *Average Validation Time:* denoted by x_{avt}, is the time taken by nodes to verify the solution found by the first miner that succeed to solve the puzzle. This time slot also depends on the size of the network and the network infrastructure.

- *Average Block Time:* denoted by x_{abt}, is the time taken to mine and validate a new block. and it is given by

$$x_{abt} = x_{amt} + x_{avt}. \tag{5.10}$$

- *Subsequent Blocks:* denoted by x_{sb}, is the number of blocks created after the block subject to confirmation time calculation. This parameter is also not yet fixed by standardization establishments nor by the first blockchain implementation (i.e., Bitcoin). The number definition is left to be proprietary and to be defined based on the assumptions of system owners. In this thesis, as we do not transfer money over blockchain, rather some important data, three is defined to the number of subsequent blocks.

Based on the aforementioned parameters, the blockchain confirmation time, denoted by x_{ct}, is calculated as follows:

$$x_{ct} = x_{sb} \times x_{abt}. \tag{5.11}$$

Note: the values of the parameters influencing the confirmation time calculation are controllable. In fact, if we need to get minimal times it is possible to minimize the difficulty, to strengthen miners with more powerful hardware, to improve the network connections, etc.

5.5.3 ECIES Encryption

ECIES is used to allow encryption/decryption of secret data whenever necessary and also to serve as public identifier provider for protocol transactions (i.e., demand, response, and termination). The ECIES operations are used following the IEEE P1363a [4]. The used curve is the “secp256r1” where “256” is the size of the prime curve and “r” is the type random of the curve parameters. AES with CBC is used for the symmetric encryption and 128-bit MAC as well as 128-bit block cipher key size are adopted.

5.6 Running Example

In this section, we tackle a use case of the proposed secure conversation protocol applied to a formal example of electricity trading among microgrids. We demonstrate how the protocol helps to achieve security and efficiency required for a trading protocol.

5.6.1 Case Study Presentation

Microgrids are local groups of distributed electricity loads and resources that operate in on and off modes with the traditional power grid. Microgrids are promoted with ICTs in order to enable smart operation through a bidirectional flow of information as well as electricity that helps to perform advanced functionalities such as renewable energies integration and electricity trading.

[4]http://www.secg.org/sec1-v2.pdf

5.6.1.1 Context of the Study

The integration of ICTs in the electricity grid brings the cyber-vulnerabilities to the computerized grid. In fact, many attacks can be oriented to the communication among microgrids in the context of trading. In fact, revealing the identities of trading parts can serve attackers with information about the activity as well as the purchasing power of particular entities so they can attack in critical moments through making denial of service attacks, malicious binding, and/or transaction integrity attacks.

5.6.1.2 Challenges

Some existing works have tried to solve the problem through using a trust third part (TTP) as a mediator to trade electricity on behalf of microgrids. This approach poses several problem apart from the centralized/decentralized fact which is weak face to security. It has been proven that this approach has a weakness in terms of single point of failure, i.e., attacks on the TTP may threaten all its linked parts. Moreover, additional costs in the financial as well as the computational sides must be dedicated.

Some other existing works, have tried to solve the problem through using the blockchain technology. Blockchain allows to provide a solution to the distribution and avoidance of TTP. It enables to establish trust among non-trustful parts without the need to a third authority. However, this approach poses a problem of privacy. In fact, using blockchain implies revealing the identity of all participants since all transactions will be public and accessible by all parts. In addition, using blockchain means continuously add new blocks to the chain without deleting anything. This poses a problem of growing stored data that require additional costs for storage.

In this context, we propose a trading process based on the conversation protocol to resolve the aforementioned challenges.

5.6.2 Microgrids System Model

The smart grid system is designed as a network of distributed microgrids where each microgrid is governed with a software application developed using the framework and having the possibility to trade electricity. The focus here is on the security side of the trading process not in the operational details. In the aim of providing a solution to the scalability challenge and to the costs minimization, we propose to divide microgrids into families of microgrids and the families to group of families.

5.6.2.1 Architecture and Membership Concept

Two main types of classification are proposed:

Family of microgrids: a family of micorgirds, denoted by fm, is defined as a set of geographically near microgrids. A family provides a price discount for its members when a trading process is happening between its members (i.e., buyer and seller belong to the same family). Figure 5.7 shows an example of a smart grid having seven families. For example, $fm_1 = \{m_1, m_2, m_3, m_4\}$ where m_i denotes a microgrid.

Group of families: a group of families, denoted by g, is defined as a set of families that have a pricing privilege when a trading process takes place among its members. For example, Figure 5.7 shows an example of a group of families: $g_1 = \{fm_1, fm_6, fm_2, fm_5\}$.

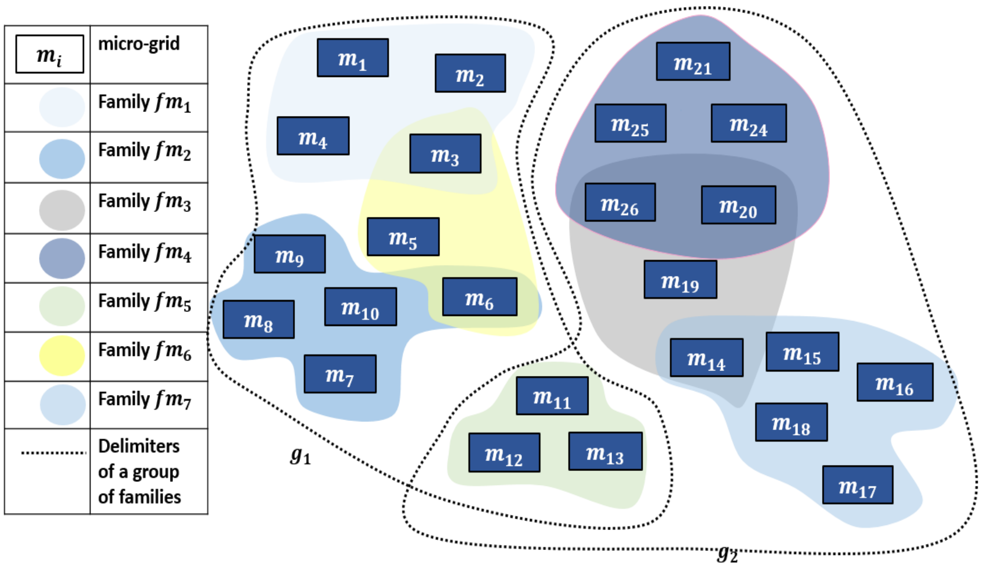


Figure 5.7: Architecture and membership concept.

5.6.2.2 Communication Model

The communication among micrograds can be performed through any networking solution defined by the system owners. For example, it can be guaranteed thanks to the Wide Area Network (WAN) solutions based on Internet. Virtual Private Networks (VPN) can be deployed for more security, IP addresses control, and secure channels. From the application level point of view, data is exchanged between microgrids through the Hypertext Transfer Protocol (HTTP). Figure 5.8 depicts the communication model of the considered system. The communication between sensing and metering infrastructure setup in the physical fields and the control applications is guaranteed thanks to ICTs (the standard IEC 61850 presents more details about such technologies).

It is also assumed that intelligent equipment are used in the electricity handling such as remotely operated circuit breakers, remotely operated switches. In addition, basic and emergency electric lines.

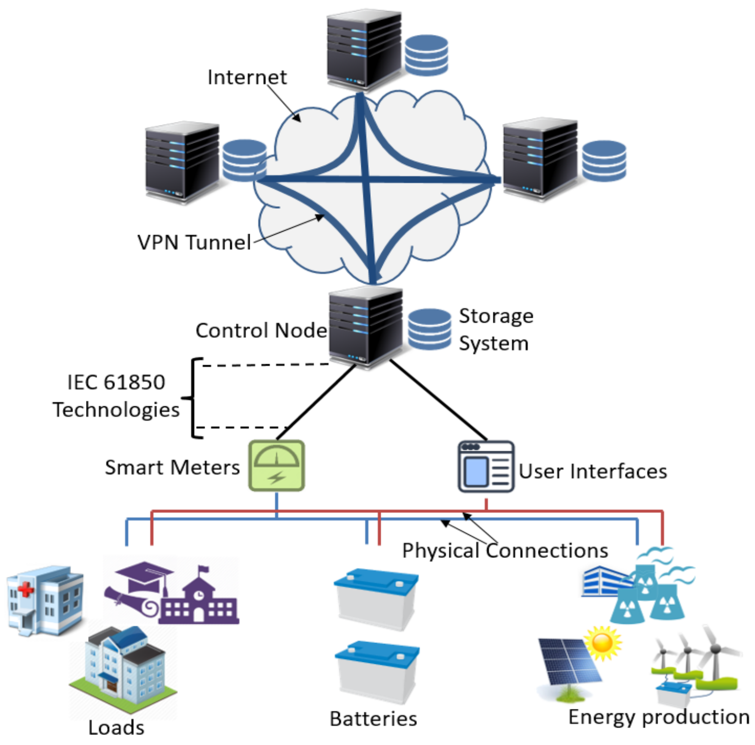


Figure 5.8: Microgrid network communication model.

5.6.3 Electricity Trading Protocol

In this section we show how the proposed conversation protocol can serve as a base for a trading process. In the considered scenario, the demand phase is used as an aggregation phase, the response phase is used as an auction phase, and the termination phase is used as settlement phase.

The proposed trading protocol offers a fully distributed security solution, ensures reliable process (through the characteristic of transparency and immutability of the blockchain), allows an anonymous participation in the trading, preserves the privacy of the sale information, and it reduces the costs in terms of number and size of exchanged data. Figure 5.9 depicts the phases and steps of the trading protocol.

5.6.3.1 Demand Phase: Aggregation

In order to reduce the number of exchanged transactions between microgrids, a coalition creation is proposed as a form of gathering in the first phase of the trading process. In fact, gathering requests of energy buying in one request in place of triggering many trading conversations may help considerably in reducing the data storage burden.

A microgrid coalition is a temporary joining together of different microgrids that belong to the same family for the purpose of buying electricity in one sale. This

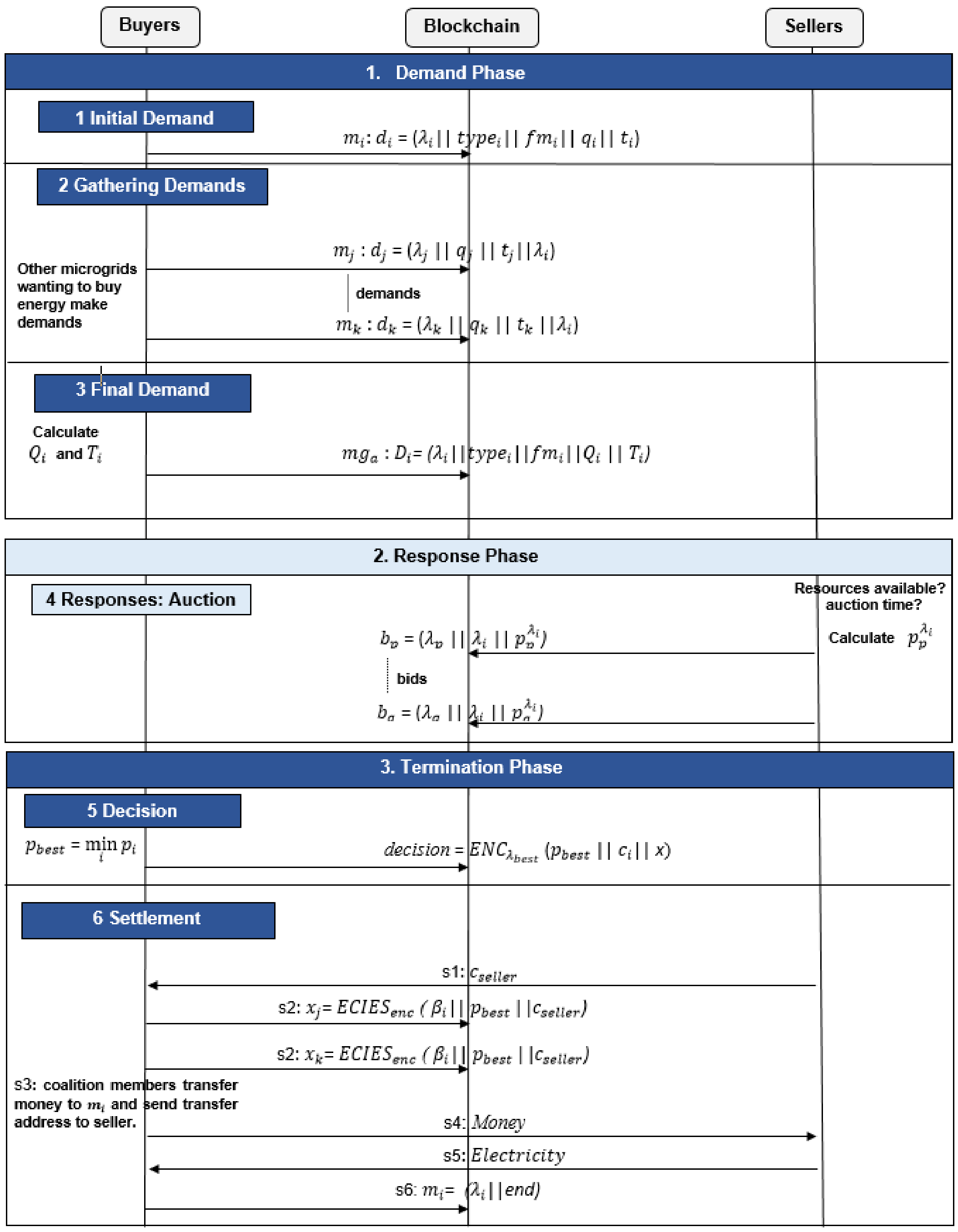

Buyers
Blockchain
Sellers
1. Demand Phase
1 Initial Demand
m_i: d_i = (λ_i || $type_i$ || fm_i || q_i || t_i)
2 Gathering Demands
m_j : d_j = (λ_j || q_j || t_j || λ_i)
Other microgrids wanting to buy energy make demands
demands
m_k : d_k = (λ_k || q_k || t_k || λ_i)
3 Final Demand
Calculate Q_i and T_i
mg_a : D_i= (λ_i || $type_i$ || fm_i || Q_i || T_i)
2. Response Phase
4 Responses: Auction
Resources available? auction time?
b_v = (λ_v || λ_i || $p_v^{\lambda_i}$)
Calculate $p_p^{\lambda_i}$
bids
b_a = (λ_a || λ_i || $p_a^{\lambda_i}$)
3. Termination Phase
5 Decision
$p_{best} = \min_i p_i$
decision = $ENC_{\lambda_{best}}$ (p_{best} || c_i || x)
6 Settlement
s1: c_{seller}
s2: x_j= $ECIES_{enc}$ (β_i || p_{best} || c_{seller})
s2: x_k= $ECIES_{enc}$ (β_i || p_{best} || c_{seller})
s3: coalition members transfer money to m_i and send transfer address to seller.
s4: Money
s5: Electricity
s6: m_i= (λ_i || end)

Figure 5.9: Trading protocol among microgrids.

coalition creation phase is performed through the three steps of the first phase: (1) initial demand, (2) gathering demands, and (3) final demand.

Step 1: Initial Demand

A microgrid m_i that needs to buy electricity begins with adding a demand d_i in the blockchain,

$$d_i = (\lambda_i || content_i). \tag{5.12}$$

$$content_i = type_i || fm_i || q_i || t_i. \tag{5.13}$$

where the content of the demand $content_i$ contains four fields: (1) $type_i$ is the type of the demand. In fact, two types of demands are defined: the type "initial" indicating the that the demand is the one triggering the start of a trading process, and the type "final" indicating the final electricity demand of the first phase. (2) fm_i is the family to which m_i belongs. (3) q_i is the electricity amount. (4) t_i is the required time of supply.

Step 2: Gathering Demands

After d_i is added to the blockchain, microgrids that belong to the same family and that need to buy energy can create demands and submit it to the blockchain too,

$$d_j = (\lambda_j || content_j). \tag{5.14}$$

$$content_j = q_j || t_j || \lambda_i. \tag{5.15}$$

where λ_j represents the identifier of d_j, q_j is the required electricity quantity, t_j is the desired time of supply, and λ_i is used to mention that the current demand is related to d_i.

This gathering step lasts a period of time defined by the system owners.

Step 3: Final Demand

Once the aggregation time is elapsed, the microgrid m_i must read all demands related to its own one and then calculate the global quantities as well as the total supply time,

$$D_i = (\lambda_i || type_i || fm_i || Q_i || T_i). \tag{5.16}$$

where λ_i is the same identifier used for d_i, $type_i$ this time the type contains "final", fm_i is the family to which the buyer microgrids belong, Q_i is the sum of the demanded electricity quantity, T_i is the total desired supply time.

Details of scenarios that may happen with the choice of large single desired supply time, resulted big quantity that may not be available by one seller, and many other possible scenarios related to the variations of the demand/response details are out of the scope of this protocol. We focus on the security side taking as example a simple trading scenario (without taking into account the exceptions and energy profile details).

5.6.3.2 Response Phase: Auction

This phase takes place among seller microgrids. In this case study, the individual responses communication way is chosen. Therefore, this phase is composed of one step: the auction step.

Step 4: Auction

Once the finial demand is posted in the blockchain, all microgrids in the system must verify their resources and plans and decide whether to participate in the auction or not. Afterwards, if it is possible to provide the total required electricity amount, a price might be calculated. The details of the pricing function are out of the scope of this protocol, however, the protocol dictates a price discount for traders belonging to the same group of families, and a higher discount for traders belonging to the same family.

Similar to the demands, bids made in the auction step are identified through the public key of the ECIES key pair. Thus, the bid, denoted by b, has the following structure,

$$b_q = (\lambda_q || \lambda_i || p_q^{\lambda_i}). \tag{5.17}$$

where λ_q is the identifier of the bid b_q, λ_i is used to refer to D_i, and $p_q^{\lambda_i}$ is the proposed price. Hence, the bid is public to all microgrids in the system, so microgrids bidding at the end of the auction step have the possibility to propose lower prices if they want to win the sale. Also, this pricing transparency help to establish trust and help to get liberal competitive prices.

5.6.3.3 Termination Phase: Purchase

This is the last phase where the seller knows that it is the chosen bidder. Then the agreement about details concerning the payment and electricity transfer can be performed. This phase is accomplished in two steps: (1) decision, and (2) settlement. First microgrids need to safely exchange contact data then all steps can be accomplished.

Step 5: Decision

After the end of the auction time, m_i must read all bids proposed in the auction step and select the best offer. In this thesis, the first-price auction model is used, where the best price to be paid is the lowest proposed price.

In order to protect seller microgrid against any attack, the decision should be secret, but in the same time we need to inform the rest of the microgrids that a decision is made and with minimal resources usage. For this, we propose the following solution: m_i must encrypt the identifier of the best bid using the public key mentioned in the bid itself with adding a variable containing the contact data and post it in the blockchain. Thus the decision is defined as follows:

$$decision = ENC_{\lambda_{best}}(p_{best}||c_i||x). \tag{5.18}$$

where p_{best} is the best offered price, c_i is the contact of m_i (may be a phone number/email/address, etc.), and x is random variable used to make the decision more secure (e.g. against brute force attack).

Thanks to the difficulty of the elliptic curve discrete logarithm problem and the Diffie-Hellman problem, the cipher text cannot be decrypted except by the microgrid having the private key related to the public key used in encryption, i.e., no microgrid can decrypt the cipher except the chosen bidder.

Step 6: Settlement

In this step, microgrids need to exchange the contact data that allows the payment. For this, six steps are defined as follows:

- **s1:** The seller sends its contact data (bank data/address) denoted by c_{seller} to m_i using the contact c_i included in the decision. This step does not involve the blockchain.

- **s2:** The buyer posts in the blockchain its bank data, the price, and the contact of the seller c_{seller} using every public key of coalition members (that have participated in the gathering step). The creation of these transactions is performed in the same order of appearance in the gathering step.

$$x_j = ECIES_{enc}(\beta_i||p_{best}||c_{seller}). \tag{5.19}$$

  where x_j denotes the notification to the coalition member m_j, β_i is the bank data of m_i, p_{best} is the chosen price, and c_{sller} is the contact of the seller which will be used to send details of which electric lines to consider for electricity transfer.

- **s3:** Coalition members send money to m_i using β_i and send electricity transfer addresses to seller using c_{seller}. This step does not involve the blockchain.

- **s4:** m_i sends the sum of money to the seller. Also this step does not involve the blockchain.

- **s5:** Seller transfer electricity to concerned microgrids.

- **s6:** Microgrid m_i confirms the end of the trading process by adding the following termination transaction to the blockchain,

$$Ter_n = (\lambda_i||end) \tag{5.20}$$

5.6.4 Performance Evaluation

In this section, we analyze the performance of the proposed electricity trading protocol via three scopes: first we compare the current protocol to existing trading protocols, then a security and privacy analysis is conducted, finally we analyze the memory consumption costs.

5.6.4.1 Comparison to Existing Works

Compared to existing works, the current trading protocol provides a secure, distributed, and cost-effective protocol.

Concerning the security, the proposed protocol guarantees privacy for both buyers and seller not only buyers. Compared to the use of RSA (the largely used encryption scheme), the use of ECIES is more secure. In fact, for the same security level, the keys of ECIES are thousands of times harder to break than RSA keys [5].

Concerning the distribution, the current protocol finds a solution to make the process fully distributed not like other works where TTP is not used but some of the nodes are considered trustful, which make the process decentralized rather than distributed.

Concerning the cost efficiency, the definition of the gathering step helps to considerably reduce the number of exchanged transactions, thus to reduce the data to be stored in the blockchain. Compared to other works, in which aggregations are made through an aggregator or a TTP, the current paper achieves the advantages of aggregations without the use of any TTP.

5.6.4.2 Security and Privacy

In order to demonstrate the security and privacy of the protocol, the honest-but-curious adversary model is used to prove that each peer knows only what it is allowed to know and curious peers fail to know any secret information.

In the demand phase (aggregation), microgrids that need to buy electricity post in the blockchain a demand identified by the public key of the ECIES key pair. The demands are considered anonymous since they are posted in the chain but no one knows its source. Thus, the identity of microgrids is kept private. Additionally, new keys are newly generated with each new transaction, hence no binding can happen.

In the response phase (auction), the auction step is guaranteed thanks to the blockchain, on the contrary to existing works that relies on TTP that plays a role of a trusted auctioneer, the proposed approach make the trust under the responsibility of all microgrids. In addition, man-in-the-middle attack has less chances in this type of architecture. From another side, the immutability of the blockchain guarantees that bids are never modified, hence we avoid the price modification attack or any other sort of integrity attack.

[5]http://www.secg.org/sec1-v2.pdf

In the termination phase (settlement), in the decision step, the use of ECIES encryption scheme makes it extremely hard to decrypt the cryptogram given the difficulty of Diffie-Hellman problem and the elliptic curve discrete logarithm problem. In addition, settlement data are exchanged also as encrypted messages, so it is hard to apply attack in this level also.

5.6.4.3 Cost-Efficiency

In order to show the efficiency in terms of memory usage, let us consider the following scenario. A formal case study is considered in which six brother microgrids $\{m_1, ..., m_6\}$ need to buy the following amounts of energy (see Table 5.1).

Table 5.1: Microgrids required electricity quantities.

Name	Electricity quantity (MWh)
m_1	0.4
m_2	0.55
m_3	0.65
m_4	0.6
m_5	0.2
m_6	0.35

Let us consider 14 bidders $\{m_7, ..., m_{20}\}$ that can provide the requested energy. $m_1, ..., m_{10}$ belong to a same family fm_1. $m_{11}, ..., m_{15}$ belong to fm_2, $m_{16}, ..., m_{20}$ belong to fm_3. fm_1 and fm_2 belong to the group g_1, and fm_3 belong to the group g_2.

Now it is demonstrated that the adoption of the proposed protocol allows better resources usage specifically the optimization of the size of the stored data in the blocks of the blockchain. Let us consider the same formal case study and suppose that for every demand from a buyer, there will be only five bids.

Scenario 1 denotes the case when m_1,..., m_6 apply individually.

Scenario 2 denotes the case when m_1 applies for a demand of the aggregation of the six microgrids.

Before presenting the calculation of the costs in the two scenarios, let us list the sizes of every field in the demands, responses (i.e., bids) and termination transaction.

Using the information in Table 5.2, a demand stored in the blockchain d_i consumes the size of $\lambda + type + fm + q + t = 124 + 2 + 2 + 6 + 11 = 145$ bytes, a reply consumes the size of $\lambda + q + t + \lambda = 124 + 6 + 11 + 124 = 265$ bytes, a bid consumes the size of $\lambda + \lambda + p = 124 + 124 + 5 = 253$ bytes, and finally the decision d consumes 181 bytes (summarized in Table 5.3).

Now an analysis and comparison of the storage burden in the blockchain is presented.

In Scenario 1, every microgrid applies individually for power, six final demands, thirty bids, and six decisions will be stored in the blockchain. This means that $(6 \times 145) + (30 \times 253) + (6 \times 181) = 9546$ bytes have to be stored in the blockchain.

Table 5.2: Component sizes of the used transactions.

Name	Size
λ	ECIES public key having the size 124 bytes (3072 bits/564 bytes for RSA)
$type$	is of size 2 bytes
fm	is of size 2 bytes
q, Q	quantities are of size 6 bytes
t, T	service times are of size 11 bytes
p	the price is of size 5 bytes
x	the random variable is of size 2 bytes
β	the bank data is of size 45 bytes
c	the contact variable is of size 45 bytes
$decision$	cipher of ECIES of size 181 bytes (384 bytes for RSA)

Table 5.3: Sizes of the demands/responses with ECIES.

Name	Size
initial demand d	$\lambda + type + fm + q + t = 124 + 2 + 2 + 6 + 11 = 145$ bytes
demand d	$\lambda + q + t + \lambda = 124 + 6 + 11 + 124 = 265$ bytes
final demand D	$\lambda + type + fm + Q + T = 124 + 2 + 2 + 6 + 11 = 145$ bytes
bid b	$\lambda + \lambda + p = 124 + 124 + 5 = 253$ bytes
decision $decision$	181 bytes

In Scenario 2, where the six microgrids are united in a coalition, one initial demand, five demands, one final demand, five bids, and one decision will be stored in the blockchain. This means that $(2\times145)+(5\times265)+(5\times253)+(1\times181) = 3061$ bytes have to be stored in the blockchain. Therefore the use of the coalition concept reduces considerably the storage burden. For the considered case it saves 6485 bytes which is around 67% of the size of data to be stored in the blockchain.

For the same security level 128, adopting ECIES encryption also helps to reduce the storage burden compared with RSA encryption. When using RSA encryption, a demand needs $\lambda+type+fm+q+t = 564+2+2+6+11 = 585$ bytes, a coalition demand consumes the size of $\lambda + q + t + \lambda = 564 + 6 + 11 + 564 = 1145$ bytes, a bid consumes the size of $\lambda + \lambda + p = 564 + 564 + 5 = 1133$ bytes, and finally the decision consumes 384 bytes (summarized in Table 5.4).

If RSA is used in stead of ECIES in Scenario 2 (with coalition), then $(2 \times 585) + (5 \times 1145) + (5 \times 1133) + (1 \times 384) = 12944$ bytes have to be stored in the blockchain. Hence the adoption of ECIES reduces considerably the sizes of the data to be stored in the blocks. For Scenario 2 the use of ECIES encryption in place of RSA encryption reduces around 75% of the size of data.

Figure 5.10 depicts the contribution of the proposed protocol in terms of reducing the amount of data to be stored in the blocks of the blockchain compared with the non-use of the aggregation approach and compared with the use of RSA

Table 5.4: Sizes of the demands/responses with RSA.

Name	Size
initial demand d	$\lambda + type + fm + q + t = 564 + 2 + 2 + 6 + 11 = 585$ bytes
demand d	$\lambda + q + t + \lambda = 564 + 6 + 11 + 564 = 1145$ bytes
final demand D	$\lambda + type + fm + Q + T = 564 + 2 + 2 + 6 + 11 = 585$ bytes
bid b	$\lambda + \lambda + p = 564 + 564 + 5 = 1133$ bytes
decision *decision*	384 bytes

encryption.

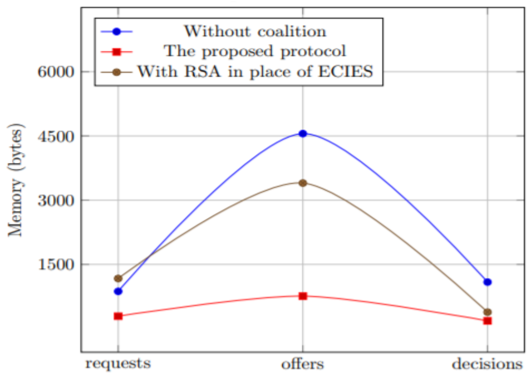


Figure 5.10: Comparison of memory size.

Figure 5.10 shows that the adoption of the coalition phase along with the ECIES encryption scheme highly reduce the size of the data to be stored in the blockchain.

5.7 Discussion

The proposed secure conversation/coordination protocol has multiple assets that can be summarized in four main points: distribution, security, privacy-preserving, and cost-efficiency. These features are achieved thanks to the characteristics of the blockchain as well as the Elliptic Curve Integrated Encryption Scheme (ECIES):

- **Security and Privacy-preserving.** (a) The security of the protocol lies in the immutability of stored data hence data cannot be falsified once it is added to the chain. (b) The protocol guarantees anonymous participation for peers through a mechanism identifying transactions rather than their source. This also improve peers privacy.
- **Full Distribution.** In the proposed protocol any peer of the system have the possibility to participate in a public conversation without the need to go through a trusted mediator. In addition, validation and mining are made by all peers not authorized ones (decentralized approach) or a TTP (centralized approach).

- **Cost-efficiency.** Secret information is encrypted with the ECIES scheme which guarantees better security level with less memory usage compared with RSA encryption [Gayoso Martínez *et al.* 2010]. In addition, the concepts of groups conversation helps to minimize the number of exchanged data.

In addition to the aforementioned assets, the proposed protocol can have more improvements such as including more functional steps enabling the withdrawal of transactions whenever unpredictable changes occur. Such problem involves the timing and procedural issues. Therefore more details would provide more guarantees and improve the quality of the protocol.

5.8 Conclusions

This chapter has dealt with the secure conversation among a network of frameworks. First, we defined a generic secure conversation process that builds upon the blockchain technology as well as the elliptic curve cryptography. Then, the concepts are implemented using Java programming language on a formal case study of an electricity trading protocol among microgrids.

The security and efficiency of the case study are analyzed and the analysis proves the robustness of the protocol. The protocol ensures a transparent communication between peers and respects the full distribution paradigm. It provides the participants with the ability to preserve anonymity and privacy of important information. More importantly, the steps and phases of the protocol promote the costs minimization, especially the memory usage one, which contributes considerably in the scalability of the system.

Chapter 6

Conclusions and Perspectives

Contents

This chapter concludes the dissertation with a summary of the obtained research contributions and presents outlooks and perspectives for future work.

6.1 Context and Problems

Intelligence and smartness are fundamental features for the efficient and successful operation of future reconfigurable systems. In order to enable such features, software advances in the field of context-awareness and intelligence computing techniques should be taken into account. From another side, the development of software of these systems includes a set of repetitive requirements where a reusable software infrastructure may help in improving the development process through minimizing the time and effort of modeling/implementing of these complex and repetitive features. Hence, an ultimate objective is to provide a software solution ensuring the software robustness all with improving developers productivity.

However, achieving such objectives is not easy given the set of problems associated to the engineering of these systems. Problems are related to the intricacy of reconfigurable systems and its miscellaneous requirements as well as constraints. To reduce the complexity of these problems, we first classify them into three types. Then a set of solutions is defined to resolve them one by one:

Requirements satisfaction problems. This category tackles the problems of finding solutions that fits the case of reconfigurable systems. Many requirements need to be provided such as the coordination in distributed systems, the respect of timing and functional constraints, the satisfaction of intelligent and predictive needs. More importantly, answers to these questions need to be addressed: **How to allow context-awareness and dynamic interaction with the environment? How to guarantee secure interaction between distributed peers against tampering, eavesdropping, and malicious tracking threats?**

Software infrastructure problems. This category tackles the problem of developing a variety of features that must operate together in harmony using one single support. The problems are formulated through the hereafter questions: How to promote reusability especially that many of the aforementioned features are frequently required by different systems? How to satisfy a variety of requirements from a single architecture and in clear way?

Methodological problems. This category tackles the problems of helping designers to get guarantees about the correctness of applications models in early development stages. UML semantics do not cover all features especially those related to probabilistic as well as resource-constrained scenarios. Analyzing the behavior of applications could not guarantee sureness if only UML models are adopted, hence there is a need to formal methods to get better guarantees. Equally important to formal methods, analyzing the applications models using the distributed systems standard, the IEC 61499, may allow to analyze the behavior in target hardware environment using one of its editors/simulators. But which analysis method to adopt and is it possible to define a process that takes advantages from the three mentioned techniques (UML, formal verification, and IEC 61499 function blocks)? How to facilitate the task of using different application modeling perspectives to developers?

In response to the above-mentioned problems, the current thesis has introduced a set of novel contributions presented in the following section.

6.2 Contributions and Outputs

This dissertation presents a new software framework dedicated to the development of context-aware reconfigurable applications of distributed automation and control systems. The aim of this project was to address the challenges associated with the development of context-aware, distributed, secure applications based on: (1) sound applications model, (2) efficient modeling and verification methodology, as well as (3) consistent software infrastructure.

The project provides solutions to develop context-aware applications from design to implementation. **At design time,** a meta-model as well as UML framework models are defined to help in conceiving systems/applications. Furthermore, a context modeling and reasoning mechanism is introduced to help define context-aware behaviors. A secure conversation protocol between distributed peers in a trustless environment is also introduced. For more reliability and efficiency, formal verification and deployment testing methodology is introduced. **At implementation time,** using the framework programming software tool represents a base of any customized applications. A protocol software that builds upon the blockchain technology is also provided.

The contributions achieved through this project are summarized in the following.

Chapter 2 has surveyed the context-awareness computing state-of-the-art by defining the most important mainstay concepts as well as presenting the most important works in the community. This survey has motivated the need for a new definition of the term context in the field of reconfigurable automated systems as well as the need for a suitable software tool. From another side, this chapter has presented in nutshells each related topic to the project contributions. In response to the limitations and motivations recognized in this chapter, the third, fourth and fifth chapters have introduced the framework and its techniques, the modeling methodology, and the security protocol.

Chapter 3 has started with defining the term context for the use in context-aware reconfigurable applications of distributed systems. Then, a meta-model from which the framework architecture was hailed is introduced. This meta-model has helped to identify the different functional levels as well as the necessary compartments, which are defined as layers and modules in the framework architecture. Afterwards, the framework concepts and mechanisms are defined. The framework provides developers with the ability to seamlessly develop multi-disciplinary applications through its architecture that covers a set of different topics related in a loosely coupled way. A context reasoning approach was introduced in the first layer. A running example was also used to prove the suitability of the concepts.

Chapter 4 has first introduced a new UML profile named GR-UML extending the existing semantics with probabilistic and resource control ones. Then, a modeling methodology is introduced to allow the modeling of applications using GR-UML, the formal verification of applications models, and the modeling according to IEC 61499 function blocks. The methodology has an iterative incremental process ensuring a flexibility in terms of modification and refinement during each phase. The process is made easier thanks to a set of model transformation rules. A software tool implementing the mapping rules of GR-UML models to GR-TNCES models as well as to IEC 61499 function block models is developed.

Chapter 5 has introduced a secure conversation protocol among a set of applications developed according to the framework. The security of the protocol draws upon the blockchain technology and the elliptic curve cryptography. A private blockchain is created where trustless nodes interact with transparency and reliability. The validity of the chain is maintained through a proof-of-work consensus mechanism where all nodes are considered as miners. The concepts of the protocol were extended and applied to an electricity trading scenario among distributed microgrids.

6.3 Perspectives

This research project aims at contributing towards resolving the identified problems. Nevertheless, questions for further research opened through the current thesis are defined.

The concepts of the proposed framework are implemented using Java programming language. This tool needs some improvements such as more testing through applying more case studies. An improvement of the quality of the code would make the tool more powerful.

Using the framework concepts in the development of other examples may prove more its suitability. Future smart factories is an example of reconfigurable systems requiring context-awareness and intelligence mechanisms. Therefore, using the framework may provide promising results.

In order to improve the data storage and access during the deployment of smart reconfigurable systems, the enrichment of the framework features with the ability to use the cloud computing [Mahmud *et al.* 2020] techniques need to be studied. Cloud computing may provide solutions for cost savings and promoting collaboration of distributed parts.

Providing a solution to analyze the quality of service [Abid *et al.* 2020] of applications code would also offer better efficient systems. Software quality parameters or metrics corresponding to the context-aware reconfigurable software applications may help to assess the quality of resulted software, hence future research activities should include this perspective.

Current deep learning techniques and tools embodies some limitations as presented in Chapter 3. However, given the numerous effective features offered by the deep learning, more research would lead to efficient results that could improve the context reasoning process.

Bibliography

[Abdellatif *et al.* 2019] Alaa Awad Abdellatif, Amr Mohamed, Carla Fabiana Chiasserini, Mounira Tlili and Aiman Erbad. *Edge computing for smart health: Context-aware approaches, opportunities, and challenges.* IEEE Network, vol. 33, no. 3, pages 196–203, 2019. (Cited on page 19.)

[Abid *et al.* 2020] Chaima Abid, Marouane Kessentini and Hanzhang Wang. *Early Prediction of Quality of Service Using Interface-level Metrics, Code-level Metrics, and Antipatterns.* Information and Software Technology, page 106313, 2020. (Cited on page 144.)

[Abowd *et al.* 1999] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith and Pete Steggles. *Towards a better understanding of context and context-awareness.* In International symposium on handheld and ubiquitous computing, pages 304–307. Springer, 1999. (Cited on pages 18 and 39.)

[Addouche *et al.* 2006] Nawal Addouche, Christian Antoine and Jacky Montmain. *Methodology for UML modeling and formal verification of real-time systems.* In 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), pages 17–17. IEEE, 2006. (Cited on page 32.)

[Aid & Rassoul 2017] Aicha Aid and Idir Rassoul. *Context-aware framework to support situation-awareness for disaster management.* International Journal of Ad Hoc and Ubiquitous Computing, vol. 25, no. 3, pages 120–132, 2017. (Cited on pages 24 and 26.)

[Alegre-Ibarra *et al.* 2018] Unai Alegre-Ibarra, Juan Carlos Augusto and Carl Evans. *Perspectives on engineering more usable context-aware systems.* Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 5, pages 1593–1609, 2018. (Cited on page 22.)

[Alegre *et al.* 2016] Unai Alegre, Juan Carlos Augusto and Tony Clark. *Engineering context-aware systems and applications: A survey.* Journal of Systems and Software, vol. 117, pages 55–83, 2016. (Cited on pages 17 and 23.)

[Alhamid *et al.* 2016] Mohammed F Alhamid, Majdi Rawashdeh, Haiwei Dong, M Anwar Hossain, Abdulhameed Alelaiwi and Abdulmotaleb El Saddik. *ReCAm: a collaborative context-aware framework for multimedia recommendations in an ambient intelligence environment.* Multimedia Systems, vol. 22, no. 5, pages 587–601, 2016. (Cited on pages 24 and 26.)

[Alhanahnah *et al.* 2018] Mohannad Alhanahnah, Peter Bertok, Zahir Tari and Sahel Alouneh. *Context-aware multifaceted trust framework for evaluating trustworthiness of cloud providers.* Future Generation Computer Systems, vol. 79, pages 488–499, 2018. (Cited on page 24.)

[Alsafi & Vyatkin 2010] Yazen Alsafi and Valeriy Vyatkin. *Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing.* Robotics and Computer-Integrated Manufacturing, vol. 26, no. 4, pages 381–391, 2010. (Cited on page 31.)

[Andren *et al.* 2017] Filip Andren, Georg Lauss, Roland BrUndlinger, Philipp Svec, Christian Seitl and Thomas Strasser. *Smart Grid Laboratory Automation Approach Using IEC 61499.* Distributed Control Applications: Guidelines, Design Patterns, and Application Examples with the IEC 61499, page 463, 2017. (Cited on page 30.)

[Arfaoui *et al.* 2020] Amel Arfaoui, Omar Rafik Merad Boudia, Ali Kribeche, Sidi-Mohammed Senouci and Mohamed Hamdi. *Context-aware access control and anonymous authentication in WBAN.* Computers & Security, vol. 88, page 101496, 2020. (Cited on page 19.)

[Belkadi *et al.* 2020] Farouk Belkadi, Mohamed Anis Dhuieb, José Vicente Aguado, Florent Laroche, Alain Bernard and Francisco Chinesta. *Intelligent assistant system as a context-aware decision-making support for the workers of the future.* Computers & Industrial Engineering, vol. 139, page 105732, 2020. (Cited on page 19.)

[Black & Vyatkin 2009] Geoff Black and Valeriy Vyatkin. *Intelligent component-based automation of baggage handling systems with IEC 61499.* IEEE Transactions on Automation Science and Engineering, vol. 7, no. 2, pages 337–351, 2009. (Cited on page 112.)

[Brown *et al.* 1997] Peter J Brown, John D Bovey and Xian Chen. *Context-aware applications: from the laboratory to the marketplace.* IEEE personal communications, vol. 4, no. 5, pages 58–64, 1997. (Cited on page 18.)

[Bucchiarone *et al.* 2017] Antonio Bucchiarone, Annapaola Marconi, Marco Pistore and Heorhi Raik. *A context-aware framework for dynamic composition of process fragments in the internet of services.* Journal of Internet Services and Applications, vol. 8, no. 1, page 6, 2017. (Cited on pages 3, 24 and 26.)

[Buttazzo 2011] Giorgio C Buttazzo. Hard real-time computing systems: predictable scheduling algorithms and applications, volume 24. Springer Science & Business Media, 2011. (Cited on page 62.)

[Cardoso & Sibertin-Blanc 2001] Janette Cardoso and Christophe Sibertin-Blanc. *Ordering actions in sequence diagrams of UML.* In Proceedings of the 23rd

International Conference on Information Technology Interfaces, 2001. ITI 2001., pages 3–14. IEEE, 2001. (Cited on page 32.)

[Chaqfeh & Mohamed 2012] Moumena A Chaqfeh and Nader Mohamed. *Challenges in middleware solutions for the internet of things.* In 2012 international conference on collaboration technologies and systems (CTS), pages 21–26. IEEE, 2012. (Cited on page 23.)

[Chen *et al.* 2004] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich and Dipanjan Chakraborty. *Intelligent agents meet the semantic web in smart spaces.* IEEE Internet computing, vol. 8, no. 6, pages 69–79, 2004. (Cited on page 23.)

[Chen 2017] Hong Chen. *Applications of cyber-physical system: a literature review.* Journal of Industrial Integration and Management, vol. 2, no. 03, page 1750012, 2017. (Cited on page 4.)

[Cheng *et al.* 2018] Xiufeng Cheng, Jinqing Yang and Lixin Xia. *A service-oriented context-awareness reasoning framework and its implementation.* The Electronic Library, 2018. (Cited on pages 24 and 26.)

[Choi *et al.* 2018] Chang Choi, Christian Esposito, Haoxiang Wang, Zhe Liu and Junho Choi. *Intelligent power equipment management based on distributed context-aware inference in smart cities.* IEEE Communications Magazine, vol. 56, no. 7, pages 212–217, 2018. (Cited on page 19.)

[Dai *et al.* 2015] Wenbin Dai, Valeriy Vyatkin, James H Christensen and Victor N Dubinin. *Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability.* IEEE Transactions on Industrial Informatics, vol. 11, no. 3, pages 771–781, 2015. (Cited on page 30.)

[Dai *et al.* 2016] Wenbin Dai, Victor N Dubinin, James H Christensen, Valeriy Vyatkin and Xinping Guan. *Toward self-manageable and adaptive industrial cyber-physical systems with knowledge-driven autonomic service management.* IEEE Transactions on Industrial Informatics, vol. 13, no. 2, pages 725–736, 2016. (Cited on page 30.)

[Devaraju *et al.* 2007] Anusuriya Devaraju, Simon Hoh and Michael Hartley. *A context gathering framework for context-aware mobile solutions.* In Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology, pages 39–46, 2007. (Cited on page 23.)

[Donohoe *et al.* 2013] Michael Donohoe, Brendan Jennings and Sasitharan Balasubramaniam. *Context-aware microgrid storage using electric cars.* In IEEE PES ISGT Europe 2013, pages 1–5. IEEE, 2013. (Cited on page 19.)

[Dubinin *et al.* 2005] Victor Dubinin, Valeriy Vyatkin and Thomas Pfeiffer. *Engineering of validatable automation systems based on an extension of UML combined with function blocks of IEC 61499.* In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 3996–4001. IEEE, 2005. (Cited on page 32.)

[Ekaputra *et al.* 2017] Fajar Ekaputra, Marta Sabou, Estefanía Serral Asensio, Elmar Kiesling and Stefan Biffl. *Ontology-based data integration in multi-disciplinary engineering environments: A review.* Open Journal of Information Systems, vol. 4, no. 1, pages 1–26, 2017. (Cited on page 16.)

[El Khaddar *et al.* 2015] Mehdia Ajana El Khaddar, Mhammed Chraibi, Hamid Harroud, Mohammed Boulmalf, Mohammed Elkoutbi and Abdelilah Maach. *A policy-based middleware for context-aware pervasive computing.* International Journal of Pervasive Computing and Communications, 2015. (Cited on page 23.)

[Fkaier *et al.* 2016a] Soumoud Fkaier, Mohamed Romdhani, Mohamed Khalgui and Georg Frey. *Enabling reconfiguration of adaptive control systems using real-time context-aware framework.* In 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), pages 1–8. IEEE, 2016. (Cited on pages 19 and 44.)

[Fkaier *et al.* 2016b] Soumoud Fkaier, Mohamed Romdhani, Mohamed Khalgui and Georg Frey. *R2TCA: New tool for developing reconfigurable real-time context-aware framework—Application to baggage handling systems.* In Proc. Int. Conf. Mobile Ubiquitous Comput., Syst., Services Technol.(UBICOMM), pages 113–119, 2016. (Cited on pages 19 and 61.)

[Fkaier *et al.* 2017] Soumoud Fkaier, Mohamed Romdhani, Mohamed Khalgui and Georg Frey. *Context-awareness Meta-model for Reconfigurable Control Systems.* In Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,, pages 226–234. INSTICC, SciTePress, 2017. (Cited on pages 19 and 40.)

[Fkaier. *et al.* 2020a] Soumoud Fkaier., Mohamed Khalgui. and Georg Frey. *Hybrid Context-awareness Modelling and Reasoning Approach for Microgrids Intelligent Control.* In Proceedings of the 15th International Conference on Software Technologies - Volume 1: ICSOFT,, pages 116–127. INSTICC, SciTePress, 2020. (Cited on pages 19, 22 and 47.)

[Fkaier *et al.* 2020b] Soumoud Fkaier, Mohamed Khalgui and Georg Frey. *Meta-Model for Control Applications of Microgrids.* In 2020 6th IEEE International Energy Conference (ENERGYCon), pages 945–950. IEEE, 2020. (Cited on page 40.)

[Fkaier. *et al.* 2021a] Soumoud Fkaier., Mohamed Khalgui. and Georg Frey. *Modeling Methodology for Reconfigurable Distributed Systems using Transformations from GR-UML to GR-TNCES and IEC 61499.* In Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE,, pages 221–230. INSTICC, SciTePress, 2021. (Cited on page 82.)

[Fkaier. *et al.* 2021b] Soumoud Fkaier., Mohamed Khalgui. and Georg Frey. *A Software Framework for Context-aware Secure Intelligent Applications of Distributed Systems.* In Proceedings of the 16th International Conference on Software Technologies - ICSOFT,, pages 111–121. INSTICC, SciTePress, 2021. (Cited on pages 44 and 68.)

[Flatt *et al.* 2015] Holger Flatt, Nils Koch, Carsten Röcker, Andrei Günter and Jürgen Jasperneite. *A context-aware assistance system for maintenance applications in smart factories based on augmented reality and indoor localization.* In 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), pages 1–4. IEEE, 2015. (Cited on page 19.)

[Forkan *et al.* 2014] Abdur Forkan, Ibrahim Khalil and Zahir Tari. *CoCaMAAL: A cloud-oriented context-aware middleware in ambient assisted living.* Future Generation Computer Systems, vol. 35, pages 114–127, 2014. (Cited on page 23.)

[Forkan *et al.* 2015] Abdur Rahim Mohammad Forkan, Ibrahim Khalil, Ayman Ibaida and Zahir Tari. *BDCaM: Big data for context-aware monitoring—A personalized knowledge discovery framework for assisted healthcare.* IEEE transactions on cloud computing, vol. 5, no. 4, pages 628–641, 2015. (Cited on pages 19, 23 and 26.)

[Franklin & Flaschbart 1998] David Franklin and Joshua Flaschbart. *All gadget and no representation makes jack a dull environment.* In Proceedings of the AAAI 1998 spring symposium on intelligent environments, pages 155–160, 1998. (Cited on page 18.)

[Frey 2014] Markus Frey. *Models and Methods for Optimizing Baggage Handling at Airports.* PhD thesis, Technische Universität München, 2014. (Cited on page 105.)

[Garcia *et al.* 2017] Marcelo V Garcia, Edurne Irisarri, Federico Perez, Elisabet Estevez and Marga Marcos. *An Open CPPS Automation Architecture based on IEC-61499 over OPC-UA for flexible manufacturing in Oil&Gas Industry.* IFAC-PapersOnLine, vol. 50, no. 1, pages 1231–1238, 2017. (Cited on page 30.)

[Gayoso Martínez *et al.* 2010] Víctor Gayoso Martínez, Luis Hernández Encinas and Carmen Sánchez Ávila. *A survey of the elliptic curve integrated encryption scheme.* 2010. (Cited on pages 12 and 139.)

[Ghribi *et al.* 2018] Ines Ghribi, Riadh Ben Abdallah, Mohamed Khalgui, Zhiwu Li, Khalid Alnowibet and Marco Platzner. *R-codesign: Codesign methodology for real-time reconfigurable embedded systems under energy constraints.* IEEE Access, vol. 6, pages 14078–14092, 2018. (Cited on pages 6 and 8.)

[Gochhayat *et al.* 2019] Sarada Prasad Gochhayat, Pallavi Kaliyar, Mauro Conti, Prayag Tiwari, VBS Prasath, Deepak Gupta and Ashish Khanna. *LISA: Lightweight context-aware IoT service architecture.* Journal of cleaner production, vol. 212, pages 1345–1356, 2019. (Cited on page 19.)

[Grichi *et al.* 2017] Hanen Grichi, Olfa Mosbahi, Mohamed Khalgui and Zhiwu Li. *New power-oriented methodology for dynamic resizing and mobility of reconfigurable wireless sensor networks.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 7, pages 1120–1130, 2017. (Cited on page 8.)

[Grimm *et al.* 2018] Tomás Grimm, Djones Lettnin and Michael Hübner. *A survey on formal verification techniques for safety-critical systems-on-chip.* Electronics, vol. 7, no. 6, page 81, 2018. (Cited on page 27.)

[Grobelna *et al.* 2010] Iwona Grobelna, Michał Grobelny and Marian Adamski. *Petri Nets and activity diagrams in logic controller specification-transformation and verification.* In Proceedings of the 17th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2010, pages 607–612. IEEE, 2010. (Cited on page 32.)

[Guellouz *et al.* 2016] Safa Guellouz, Adel Benzina, Mohamed Khalgui and Georg Frey. *Zizo: A complete tool chain for the modeling and verification of reconfigurable function blocks.* In Proc. 10th Int. Conf. Mobile Ubiquitous Comput., Syst., Services Technol.(UBICOMM), pages 144–151, 2016. (Cited on page 28.)

[Guellouz *et al.* 2018] Safa Guellouz, Adel Benzina, Mohamed Khalgui, Georg Frey, Zhiwu Li and Valeriy Vyatkin. *Designing efficient reconfigurable control systems using IEC61499 and symbolic model checking.* IEEE Transactions on Automation Science and Engineering, vol. 16, no. 3, pages 1110–1124, 2018. (Cited on pages 28 and 33.)

[Hafidi *et al.* 2019] Yousra Hafidi, Laid Kahloul, Mohamed Khalgui and Mohamed Ramdani. *New Method to Reduce Verification Time of Reconfigurable Real-Time Systems Using R-TNCESs Formalism.* In International Conference on Evaluation of Novel Approaches to Software Engineering, pages 246–266. Springer, 2019. (Cited on page 99.)

[Horcas *et al.* 2019] Jose-Miguel Horcas, Mónica Pinto and Lidia Fuentes. *Context-aware energy-efficient applications for cyber-physical systems.* Ad Hoc Networks, vol. 82, pages 15–30, 2019. (Cited on page 19.)

[Hu *et al.* 2008] Peizhao Hu, Jadwiga Indulska and Ricky Robinson. *An autonomic context management system for pervasive computing.* In 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 213–223. IEEE, 2008. (Cited on page 23.)

[Hynes *et al.* 2009] Gearoid Hynes, Vinny Reynolds and Manfred Hauswirth. *A context lifecycle for web-based context management services.* In European Conference on Smart Sensing and Context, pages 51–65. Springer, 2009. (Cited on page 19.)

[Iqbal *et al.* 2018] Razi Iqbal, Talal Ashraf Butt, M Omair Shafique, Manar Wasif Abu Talib and Tariq Umer. *Context-aware data-driven intelligent framework for fog infrastructures in internet of vehicles.* IEEE Access, vol. 6, pages 58182–58194, 2018. (Cited on pages 3 and 24.)

[Ivanova-Vasileva *et al.* 2008] Ioanna Ivanova-Vasileva, Christian Gerber and Hans-Michael Hanisch. *Basics of modelling IEC 61499 function blocks with integer-valued data types.* IFAC Proceedings Volumes, vol. 41, no. 3, pages 169–174, 2008. (Cited on page 33.)

[Kayes *et al.* 2019] ASM Kayes, Wenny Rahayu, Tharam Dillon, Elizabeth Chang and Jun Han. *Context-aware access control with imprecise context characterization for cloud-based data resources.* Future Generation Computer Systems, vol. 93, pages 237–255, 2019. (Cited on page 19.)

[Khalgui *et al.* 2019] Mohamed Khalgui, Olfa Mosbahi and Zhiwu Li. *On reconfiguration theory of discrete-event systems: From initial specification until final deployment.* IEEE Access, vol. 7, pages 18219–18233, 2019. (Cited on page 6.)

[Khan *et al.* 2020] Aftab Khan, Aakash Ahmad, Anis Ur Rahman and Adel Alkhalil. *A Mobile Cloud Framework for Context-Aware and Portable Recommender System for Smart Markets.* In Smart Infrastructure and Applications, pages 283–309. Springer, 2020. (Cited on page 19.)

[Khlifi *et al.* 2017] Oussama Khlifi, Christian Siegwart, Olfa Mosbahi, Mohamed Khalgui and Georg Frey. *Specification Approach using GR-TNCES: Application to an Automotive Transport System.* In ICSOFT, pages 105–115, 2017. (Cited on page 28.)

[Khlifi *et al.* 2019] O Khlifi, O Mosbahi, M Khalgui, G Frey and Z Li. *Modeling, simulation and verification of probabilistic reconfigurable discrete-event systems under energy and memory constraints.* Iranian Journal of Science and Technology, Transactions of Electrical Engineering, vol. 43, no. 2, pages 229–243, 2019. (Cited on pages 28, 84 and 100.)

[Kim *et al.* 2016] Kwanho Kim, Hyunjin Kim, Sang-Kuk Kim and Jae-Yoon Jung. *i-RM: An intelligent risk management framework for context-aware ubiquitous cold chain logistics.* Expert Systems with Applications, vol. 46, pages 463–473, 2016. (Cited on pages 24 and 26.)

[Krupitzer *et al.* 2018] Christian Krupitzer, Martin Breitbach, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele and Christian Becker. *A survey on engineering approaches for self-adaptive systems (extended version).* 2018. (Cited on pages 2 and 16.)

[Lee *et al.* 2018] Tae-Dong Lee, Byung Moo Lee and Wonjong Noh. *Hierarchical cloud computing architecture for context-aware IoT services.* IEEE Transactions on Consumer Electronics, vol. 64, no. 2, pages 222–230, 2018. (Cited on page 19.)

[Li *et al.* 2015] Xin Li, Martina Eckert, José-Fernán Martinez and Gregorio Rubio. *Context aware middleware architectures: survey and challenges.* Sensors, vol. 15, no. 8, pages 20570–20607, 2015. (Cited on pages 17, 18 and 23.)

[Lindgren *et al.* 2016] Per Lindgren, Johan Eriksson, Marcus Lindner, Andreas Lindner, David Pereira and Lus Miguel Pinho. *End-to-end response time of IEC 61499 distributed applications over switched ethernet.* IEEE Transactions on Industrial Informatics, vol. 13, no. 1, pages 287–297, 2016. (Cited on page 30.)

[Liu *et al.* 2017] Yang Liu, Yu Peng, Bailing Wang, Sirui Yao and Zihe Liu. *Review on cyber-physical systems.* IEEE/CAA Journal of Automatica Sinica, vol. 4, no. 1, pages 27–40, 2017. (Cited on page 2.)

[Lu *et al.* 2019] Huimin Lu, Qiang Liu, Daxin Tian, Yujie Li, Hyoungseop Kim and Seiichi Serikawa. *The cognitive internet of vehicles for autonomous driving.* IEEE Network, vol. 33, no. 3, pages 65–73, 2019. (Cited on page 83.)

[Luo & Feng 2015] Jianchao Luo and Hao Feng. *A Framework for NFC-based Context-aware Applications.* International Journal of Smart Home, vol. 9, no. 1, pages 111–122, 2015. (Cited on page 19.)

[Luo *et al.* 2020] Chu Luo, Jorge Goncalves, Eduardo Velloso and Vassilis Kostakos. *A Survey of Context Simulation for Testing Mobile Context-Aware Applications.* ACM Computing Surveys (CSUR), vol. 53, no. 1, pages 1–39, 2020. (Cited on page 19.)

[Mahalle & Dhotre 2020] Parikshit N Mahalle and Prashant S Dhotre. *Context-Aware Pervasive Systems.* In Context-Aware Pervasive Systems and Applications, pages 49–66. Springer, 2020. (Cited on pages 2 and 17.)

[Mahmud *et al.* 2020] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao and Rajkumar Buyya. *Profit-aware application placement* *for integrated Fog–Cloud computing environments*. Journal of Parallel and Distributed Computing, vol. 135, pages 177–190, 2020. (Cited on page 144.)

[Meskina *et al.* 2018] Syrine Ben Meskina, Narjes Doggaz, Mohamed Khalgui and Zhiwu Li. *Reconfiguration-based methodology for improving recovery performance of faults in smart grids*. Information Sciences, vol. 454, pages 73–95, 2018. (Cited on page 8.)

[Mousavi & Vyatkin 2015] Arash Mousavi and Valeriy Vyatkin. *Energy Efficient Agent Function Block: A semantic agent approach to IEC 61499 function blocks in energy efficient building automation systems*. Automation in Construction, vol. 54, pages 127–142, 2015. (Cited on page 31.)

[Murukannaiah & Singh 2014] Pradeep K Murukannaiah and Munindar P Singh. *Xipho: Extending Tropos to engineer context-aware personal agents*. In Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pages 309–316, 2014. (Cited on page 23.)

[Nakagawa *et al.* 2014] Elisa Yumi Nakagawa, Rafael Capilla, Francisco J Díaz and Flávio Oquendo. *Towards the dynamic evolution of context-based systems-of-systems*. In 8th WDES Workshop, Maceió, Brazil, pages 45–52, 2014. (Cited on page 69.)

[Nakamoto 2019] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. Technical report, Manubot, 2019. (Cited on pages 3 and 33.)

[Nassar *et al.* 2020] Mohamed Nassar, Khaled Salah, Muhammad Habib ur Rehman and Davor Svetinovic. *Blockchain for explainable and trustworthy artificial intelligence*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 10, no. 1, page e1340, 2020. (Cited on page 6.)

[Nikolakis *et al.* 2018] Nikolaos Nikolakis, Konstantinos Sipsas and Sotiris Makris. *A cyber-physical context-aware system for coordinating human-robot collaboration*. Procedia CIRP, vol. 72, pages 27–32, 2018. (Cited on page 31.)

[Nokovic & Sekerinski 2013] Bojan Nokovic and Emil Sekerinski. *pState: A probabilistic statecharts translator*. In 2013 2nd Mediterranean Conference on Embedded Computing (MECO), pages 29–32. IEEE, 2013. (Cited on page 32.)

[Noulamo *et al.* 2018] Thierry Noulamo, Emmanuel Tanyi, Marcellin Nkenlifack, Jean-Pierre Lienou and Alain Djimeli. *Formalization Method of the UML Statechart by Transformation Toward Petri Nets*. IAENG International Journal of Computer Science, vol. 45, no. 4, 2018. (Cited on page 32.)

[Ortiz *et al.* 2019] Guadalupe Ortiz, Alfonso Garcia-De-Prado, Javier Berrocal and Juan Hernandez. *Improving resource consumption in context-aware mobile* *applications through alternative architectural styles.* IEEE Access, vol. 7, pages 65228–65250, 2019. (Cited on page 19.)

[Oueslati *et al.* 2018] Raja Oueslati, Olfa Mosbahi, Mohamed Khalgui, Zhiwu Li and Ting Qu. *Combining semi-formal and formal methods for the development of distributed reconfigurable control systems.* IEEE Access, vol. 6, pages 70426–70443, 2018. (Cited on page 8.)

[Pang & Vyatkin 2008] Cheng Pang and Valeriy Vyatkin. *Automatic model generation of IEC 61499 function block using net condition/event systems.* In 2008 6th IEEE International Conference on Industrial Informatics, pages 1133–1138. IEEE, 2008. (Cited on page 33.)

[Panjaitan & Frey 2006] Seno Panjaitan and Georg Frey. *Combination of UML modeling and the IEC 61499 function block concept for the development of distributed automation systems.* In 2006 IEEE Conference on Emerging Technologies and Factory Automation, pages 766–773. IEEE, 2006. (Cited on pages 32 and 112.)

[Panjaitan & Frey 2007] SD Panjaitan and Georg Frey. *Development process for distributed automation systems combining UML and IEC 61499.* International Journal of Manufacturing Research, vol. 2, no. 1, pages 1–20, 2007. (Cited on pages 30 and 94.)

[Patil *et al.* 2013] Sandeep Patil, Valeriy Vyatkin and Bruce McMillin. *Implementation of FREEDM Smart Grid distributed load balancing using IEC 61499 function blocks.* In IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society, pages 8154–8159. IEEE, 2013. (Cited on page 30.)

[Perera *et al.* 2013] Charith Perera, Arkady Zaslavsky, Peter Christen and Dimitrios Georgakopoulos. *Context aware computing for the internet of things: A survey.* IEEE communications surveys & tutorials, vol. 16, no. 1, pages 414–454, 2013. (Cited on pages 18 and 23.)

[Psyche *et al.* 2020] Valery Psyche, Ben K Daniel and Jacqueline Bourdeau. *Learning Spaces in Context-Aware Educational Networking Technologies in the Digital Age.* In Educational Networking, pages 299–323. Springer, 2020. (Cited on page 19.)

[Salem *et al.* 2015a] Mohamed Oussama Ben Salem, Olfa Mosbahi, Mohamed Khalgui and Georg Frey. *Transformation from R-UML to R-TNCES: New formal solution for verification of flexible control systems.* In 2015 10th International Joint Conference on Software Technologies (ICSOFT), volume 2, pages 1–12. IEEE, 2015. (Cited on pages 32 and 86.)

[Salem *et al.* 2015b] Mohamed Oussama Ben Salem, Olfa Mosbahi, Mohamed Khalgui and Georg Frey. *ZiZo: Modeling simulation and verification of reconfigurable real-time control tasks sharing adaptive resources.* In Proc. Int.

Conf. Health Inform.(HEALTHINF), pages 20–31, 2015. (Cited on pages 99 and 100.)

[Salman *et al.* 2018] Tara Salman, Maede Zolanvari, Aiman Erbad, Raj Jain and Mohammed Samaka. *Security services using blockchains: A state of the art survey.* IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pages 858–880, 2018. (Cited on pages 3 and 34.)

[Schilit *et al.* 1994] Bill Schilit, Norman Adams and Roy Want. *Context-aware computing applications.* In 1994 First Workshop on Mobile Computing Systems and Applications, pages 85–90. IEEE, 1994. (Cited on page 17.)

[Schneider *et al.* 2017] Georg Ferdinand Schneider, Pieter Pauwels and Simone Steiger. *Ontology-based modeling of control logic in building automation systems.* IEEE Transactions on Industrial Informatics, vol. 13, no. 6, pages 3350–3360, 2017. (Cited on page 16.)

[Schneider *et al.* 2019] Georg Ferdinand Schneider, Hendro Wicaksono and Jivka Ovtcharova. *Virtual engineering of cyber-physical automation systems: The case of control logic.* Advanced Engineering Informatics, vol. 39, pages 127–143, 2019. (Cited on page 16.)

[Schneider 2019] Georg Ferdinand Schneider. *Semantic Modelling of Control Logic in Automation Systems-Knowledge-Based Support of the Engineering and Operation of Control Logic in Building and Industrial Automation Systems.* 2019. (Cited on page 99.)

[Shulsky & Schmitt 2002] Abram N Shulsky and Gary James Schmitt. Silent warfare: understanding the world of intelligence. Potomac Books, Inc., 2002. (Cited on page 19.)

[Sikder *et al.* 2019] Amit Kumar Sikder, Hidayet Aksu and A Selcuk Uluagac. *A context-aware framework for detecting sensor-based threats on smart devices.* IEEE Transactions on Mobile Computing, 2019. (Cited on pages 24 and 26.)

[Singh 2020] Pradeep Kumar Singh. Proceedings of icric 2019: Recent innovations in computing. Springer Nature, 2020. (Cited on page 19.)

[Smida *et al.* 2019] Moncef Ben Smida, Khaoula Miled, Mohamed Khalgui, Georg Frey and Zhiwu Li. *Modeling and Verification of a Reliable Multi-Agent Solution Promoting the Autonomy and Self-Sufficiency of Microgrids in an Isolated Location.* IEEE Access, vol. 7, pages 55090–55107, 2019. (Cited on pages 3 and 29.)

[Sørensen *et al.* 2019] René A Sørensen, Michael Nielsen and Henrik Karstoft. *Deep Reinforcement Learning for Route Optimization in Baggage Handling Systems.* In Proceedings of the 1st International Conference on Advances in

Signal Processing and Artificial Intelligence. International Frequency Sensor Association Publishing, pages 29–33, 2019. (Cited on page 69.)

[Tang *et al.* 2014] Lei Tang, Zhiwen Yu, Hanbo Wang, Xingshe Zhou and Zongtao Duan. *Methodology and tools for pervasive application development.* International Journal of Distributed Sensor Networks, vol. 10, no. 4, page 516432, 2014. (Cited on page 26.)

[Thramboulidis *et al.* 2017] Kleanthis Thramboulidis, P Bochalis and J Bouloumpasis. *A framework for MDE of IoT-based manufacturing cyber-physical systems.* In Proceedings of the seventh international conference on the internet of things, pages 1–8, 2017. (Cited on page 3.)

[Thramboulidis 2004] Kleanthis C Thramboulidis. *Using UML in control and automation: a model driven approach.* In 2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004, pages 587–593. IEEE, 2004. (Cited on pages 32 and 98.)

[Thramboulidis 2006] Kleanthis Thramboulidis. *Design alternatives in the IEC 61499 function block model.* In 2006 IEEE Conference on Emerging Technologies and Factory Automation, pages 1309–1316. IEEE, 2006. (Cited on page 82.)

[Thramboulidis 2007] Kleanthis Thramboulidis. *IEC 61499 in factory automation.* In Advances in Computer, Information, and Systems Sciences, and Engineering, pages 115–124. Springer, 2007. (Cited on pages 30 and 82.)

[Thramboulidis 2008] Kleanthis Thramboulidis. *Facts and Fallacies in the IEC61499 Function Block Model.* 2008. (Cited on page 83.)

[Thramboulidis 2015] Kleanthis Thramboulidis. *A cyber–physical system-based approach for industrial automation systems.* Computers in Industry, vol. 72, pages 92–102, 2015. (Cited on page 29.)

[Tranoris & Thramboulidis 2003] Christos Tranoris and Kleanthis Thramboulidis. *Integrating UML and the function block concept for the development of distributed control applications.* In EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 03TH8696), volume 2, pages 87–94. IEEE, 2003. (Cited on page 32.)

[Uddin *et al.* 2019] Riaz Uddin, Ali S Alghamdi, Muhammad Hammad Uddin, Ahmed Bilal Awan and Syed Atif Naseem. *Ethernet-Based Fault Diagnosis and Control in Smart Grid: A Stochastic Analysis via Markovian Model Checking.* Journal of Electrical Engineering & Technology, vol. 14, no. 6, pages 2289–2300, 2019. (Cited on page 98.)

[Veichtlbauer *et al.* 2016] Armin Veichtlbauer, Manuel Parfant, Oliver Langthaler, Filip Prostl Andren and Thomas Strasser. *Evaluating XMPP communication in IEC 61499-based distributed energy applications.* In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8. IEEE, 2016. (Cited on page 30.)

[Verginadis *et al.* 2017] Yiannis Verginadis, Antonis Michalas, Panagiotis Gouvas, Gunther Schiefer, Gerald Hubsch and Iraklis Paraskakis. *Paasword: A holistic data privacy and security by design framework for cloud services.* Journal of Grid Computing, vol. 15, no. 2, pages 219–234, 2017. (Cited on page 19.)

[Wang & Varghese 2020] Nan Wang and Blesson Varghese. *Context-aware Distribution of Fog Applications Using Deep Reinforcement Learning.* arXiv preprint arXiv:2001.09228, 2020. (Cited on page 19.)

[Weyns 2019] Danny Weyns. *Software engineering of self-adaptive systems.* In Handbook of Software Engineering, pages 399–443. Springer, 2019. (Cited on pages 2 and 16.)

[Yan & Vyatkin 2013] Jeffrey Yan and Valeriy Vyatkin. *Distributed software architecture enabling peer-to-peer communicating controllers.* IEEE Transactions on Industrial Informatics, vol. 9, no. 4, pages 2200–2209, 2013. (Cited on pages 30 and 112.)

[Yang *et al.* 2019a] Chen-Wei Yang, Victor Dubinin and Valeriy Vyatkin. *Automatic Generation of Control Flow from Requirements for Distributed Smart Grid Automation Control.* IEEE Transactions on Industrial Informatics, 2019. (Cited on pages 3 and 30.)

[Yang *et al.* 2019b] Ruizhe Yang, F Richard Yu, Pengbo Si, Zhaoxin Yang and Yanhua Zhang. *Integrated blockchain and edge computing systems: A survey, some research issues and challenges.* IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pages 1508–1532, 2019. (Cited on page 34.)

[Yuan & Wang 2018] Yong Yuan and Fei-Yue Wang. *Blockchain and cryptocurrencies: Model, techniques, and applications.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 9, pages 1421–1428, 2018. (Cited on page 33.)