Architecture and deployment of services of assistance to the person

Molham Darwish

UNIVERSITE BRETAGNE LOIRE

THESE / UNIVERSITE DE BRETAGNE-SUD
sous le sceau de l'Université Bretagne Loire
pour obtenir le titre de:
DOCTEUR DE L'UNIVERSITE DE BRETAGNE-SUD
Mention: STIC
Ecole doctorale SICMA

présentée par :
DARWISH Molham
Lab-STICC

Architecture et déploiement de services d'aide à la personne

Thèse soutenue le 12 septembre 2016,
devant le jury composé de :

M. Jean-Paul Haton
Professeur des Universités, Université Henri Poincaré, Nancy 1/ Rapporteur

M. Eric Zamaï
MCF-HDR, Institut national polytechnique de Grenoble / Rapporteur

M. Alexandre Abellard
MCF – Université de Toulon / Examinateur

M. Guy Gogniat
Professeur des Universités, Université Bretagne Sud / Examinateur

M. Alain Hillion
Professeur, Télécom Bretagne / Examinateur

M. Eric Senn
MCF-HDR, Université Bretagne Sud / Directeur de thèse

M. Christophe Lohr
MCF, Télécom Bretagne / Invité

M. Florent De Lamotte
MCF, Université Bretagne Sud / Invité

Abstract

The ageing of the European population encouraged the community to search for solutions to support this evolution. In this context, several issues (related to the expensive and limited health care services and health facilities capacities) need to be addressed.
Thus, several projects, research and industrial solutions have been proposed to address these issues. Most of these projects and industrial developments are based on the use of the latest ICT technical developments to provide solutions that ameliorate the well-being of the targeted ageing group and to guaranty their independence in their own living spaces. The provided technological solutions need to be guaranteed against potential faults which may lead to the systems failure and impact the users' needs and independency.

In this work, we propose a home automation system representation, based on the user's needs to provide continuous and viable solutions that meets the users' expectations, and ensuring the availability of system's services.
For this goal, we propose to develop an integrated modeling framework allowing the representation of the home automation reconfigurable system with the consideration of a fault tolerance approach (based on the alternative definition of scenarios of system services deliverance).
In the proposed workflow, we describe the system structural elements (described as services and components) in the design modeling view, and, we lead model transformation rules allowing generating an analysis model and a behavior model. The analysis model allows making a decision about the alternative elements selection in order to substitute the faulty elements. The analysis model definition is based on the notion of Fault Tree Analysis approach (adopting the probability of events failure in order to evaluate a given system status).
The behavior model is in charge of simulating the execution of the system services ensuring, thus, that the proposed scenarios lead to system services deliverance.
Moreover, we propose to define an expert based feature measuring the importance of a system's component within the service context. In this framework, we propose a new approach, based on the joint integration of the importance factor into the Fault Tree Analysis approach in order to study the criticality of the component, in case of failure, on the service continuity.
We propose an experimental validation framework, based on several validation objectives to evaluate the proposed work in this research.

Résumé

Le vieillissement de la population européenne a encouragé la communauté à favoriser la recherche de solutions pour soutenir cette évolution. Dans ce contexte, plusieurs questions (liées aux services de soins de santé et aux établissements de santé, coûteux, et à capacités limitées) doivent être abordées.
Ainsi, plusieurs projets de recherche et des solutions industrielles ont été proposés pour remédier à ces problèmes. La plupart de ces solutions sont basées sur l'utilisation des derniers développements techniques en matière de TIC permettant ainsi de fournir des solutions qui améliorent le bien-être des personnes âgées et de garantir leur indépendance dans leurs propres espaces de vie. Les solutions technologiques fournies doivent être garanties contre les défauts potentiels des équipements qui peuvent conduire à l'échec des systèmes et avoir un impact sur les besoins et l'indépendance des utilisateurs.
Dans ce travail, nous proposons une représentation du système d'automatisation de la maison, sur la base des besoins qui consiste à fournir des solutions continues et viables et répondant aux attentes des utilisateurs (tout en assurant la disponibilité des services des systèmes).
Dans cet objectif, nous proposons de développer un cadre de modélisation représentant le système reconfigurable domotique et permettant l'examen de l'approche de la tolérance de panne, sur la base de la définition de scénarios alternatifs (assurant la délivrance continue des services). Dans ce workflow, nous décrivons les éléments structurels du système (décrits comme des services et composants) en vue de la modélisation conceptuelle. Des règles de transformation de modèle permettent de générer un modèle d'analyse et un modèle de comportement. Le modèle d'analyse permet de prendre une décision à propos de la sélection des éléments alternatifs pour remplacer les éléments défectueux. Ce modèle d'analyse est défini sur la base de la notion d'approche d'arbre de défaillances, qui adopte la probabilité d'échec des composantes pour évaluer l'état global du système considéré. Le modèle de comportement est responsable de la simulation de l'exécution des services du système pour assurer que les scénarios conduisent à la délivrance du système.
Par ailleurs, nous proposons de définir une caractéristique permettant d'évaluer l'importance empirique (vue par le concepteur) des composantes d'un système dans le cadre d'un service donné. Ensuite, nous proposons une nouvelle approche, fondée sur l'intégration du facteur de l'importance dans l'approche d'arbre de défaillances pour étudier la criticité du composant, en cas d'échec, ainsi que la continuité de service. Un cadre de validation expérimentale, basé sur plusieurs objectifs de validation est finalement proposé pour conclure ce travail de recherche.

Remerciements

Contents

General introduction

This chapter provides a summary of the thesis presented in this document. To do this, an introduction to the context of this work, highlights the needs that the contribution of this thesis has sought to fill. Then, the contribution is described and discussed to meet the adopted research objectives and requirements. Finally, a number of highlighted results and conclusions, as well as some identified limitations end this chapter.

1. Research Context

The aging rates growth in the European population has prompted the community to seek solutions to accompany this growth. In this context, several problems must be considered. In fact, the health field is suffering from a shortage of workforces, which could result in a general degradation of the quality of care.

Furthermore, places in medical centers or retirement homes are limited and may become saturated in the coming years. In addition, hospital costing expensive, regardless of pathologies, and financial aid at this level tends to be limited.

One of the crucial questions that the European countries have to provide an answer as quickly as possible is the one concerning the management of dependent people, whether elderly, with disabilities or even both at once (the disabled aging).

Given the global financial crises, it is clear that the solutions consisting of investment in accommodation facilities or adapted care centers are not on the scope or the willingness of different governments. It is therefore essential to search alternative solutions promoting the autonomy of elderly people while staying at home without the help of any third party.

Several projects have already been launched to try to address these issues. The European collaborative program, Ambient Assisted Living (AAL), has been created to promote and finance projects, which highlights the interest of Europe to progress in this field. It is an assessment of the appropriateness of using Information and Communication Technology (ICT) to help elderly people.

In recent years, issues related to aging topics and assistance to individuals with disabilities are particularly among the topics, to which the region of “Bretagne” and the General Council of “Morbihan” bear high interests to finance projects and solutions, in order to tackle these concerns. In 2003, they have financed the project COHAND, which mainly focused on aspects related to provide a set of linked services / navigation to help elderly people, with the consideration of ensuring the quality of services even in the presence of hazards. Followed in 2005 by the project QUATRA which included several players in the field of domicile, such as the laboratory Lab-STICC, in which the research work of this thesis has been conducted, the laboratory VALORIA and the center of “Rééducation et de Réadaptation Fonctionnelles de Kerpape”. In the following years, up till now, several projects and researches have been launched in this context.

In Lab-STICC, the problem of “home support” for dependent people, whether elderly or disabled, is multidisciplinary. It involves the participation of different stakeholders in the sector

of human and social sciences as well as the information and communication technologies and sciences to ameliorate the well-being of the intended users.

In this context, activities conducted in this research area constitute a continuation of projects and researches that have been realized in Lab-STICC. Particularly, the works realized by Willy Allègre [Allègre, 2012] in terms of design and modeling of such home automation systems. Another promising research work was realized by Said Lankri [Lankri, 2009] which deals with the use of reconfiguration in a technical assistance context of disability, to guaranty the continuity of provided services to the disabled aging users.

System reconfigurability is the process of performing changes to system features to ensure correct suitability to the current context. Changes can be triggered due to either objective changes, or to system elements' failures. The reconfigurability in case of a failure is an essential feature that should be considered to preserve the continuity of the service delivery.

For example, when a device fails, the system, through the reconfiguration must be able to propose alternatives to obtain the requested service. When accessing a service is not possible, (for example: because a door is blocked), the system should provide an alternative path if it exists.

Following this formal definition of system reconfiguration, the objective, in this thesis, is raised from the need to establish a set of services (provided to the users) ensuring the reliability in case of potential failures of the system elements.

In this perspective, a design workflow has been developed to address the requirement of having reconfiguration mechanisms of home automation systems, based on the definition of a fault tolerance approach allowing system reconfigurability in case of failure, by making static and dynamic analysis of the system structural elements (defined during the design process) to verify that this structure does not lead to a failure of different services.

2. Problem raised in this research

Failure occurrence in critical real time systems (such as nuclear systems, telemedicine systems, as well as home automation systems) can have catastrophic consequences on the users of the system and maybe the surrounding environment. The consequence effect of a failure is raised from two aspects, the cost of uncorrected errors on the whole system performance and services continuity. This is particularly important for systems which are controlled and for which it is not accessible to human repair. Other aspects of failure occurrence exist when human lives may be affected by the system failure.

Redundancy of system elements is the principal mechanism allowing dealing with failures to ensure system continuity in terms of service deliverance. This mechanism is often expensive and sometimes ineffective for unexpected events.

Establishing a reconfigurable system requires the design of a flexible and tolerant architecture to overtake any potential failure of system elements. This architecture is based on the formalization of fault tolerance mechanisms to establish a reconfigurable home automation system, which ameliorates the design process of the system structural elements.

3. Objectives to be achieved in the thesis

The objectives of this thesis consist on proposing a system approach for the abstract definition and conception of system reconfiguration.

In this context, we propose a design flow of system elements, which respond to the reconfiguration criteria.

In this research, the reconfiguration is an essential criterion since it ensures better availability of services. Given the alternative propositions of faulty components, the proposed approach also aims at performing a computational analysis of the preformed strategy to choose which alternative to select and to justify this selection. This analysis is based on an adopted fault analysis approach to meet the reconfiguration criteria.

The prototyping of the system design flow is an essential objective in this research, to realize the proposed fault tolerance approach as well as to implement and demonstrate the concepts and abstracts related to the design flow analysis of system elements.

A proposition of a validation scenario is another main objective in the thesis to realize an experimental evaluation of the scientific contributions, as well as the technical part, in the context of existing platforms in Lab-STICC.

4. Organization of thesis manuscript

Figure 1 shows the general organization of thesis report. After presenting the context of the work and the set of requirements and criteria extracted from the bibliographic study of projects, which is faced in the first chapter, the second chapter introduces the motivation and direction of the research in terms of ensuring home automation systems continuity and details the approaches related to the system continuity and reconfigurability. The research contributions that attempt to address the targeted objectives are presented in the third and the fourth chapters.

The third chapter presents the design flow model architecture, which specifies the steps to follow to design a sustainable model of the system structural elements, through the analysis of these elements continuity and checks model viability ensuring the system reconfigurability, as well as the practical implementation of the design flow to simulate the proposed solutions.

The fourth chapter realizes a critical analysis of system fault tolerance approaches and presents the contribution to integrate a new risk indicator to the fault analysis approach to reveal in a new approach the influence of system elements in the continuity of the system.

The fifth chapter presents a framework of experimental validation scenario that needs to be realized, to validate the proposed solution presented in this research in terms of system reconfiguration. A conclusion chapter summarizes all the conducted work and contributions and presents promising perspectives from the conducted work.

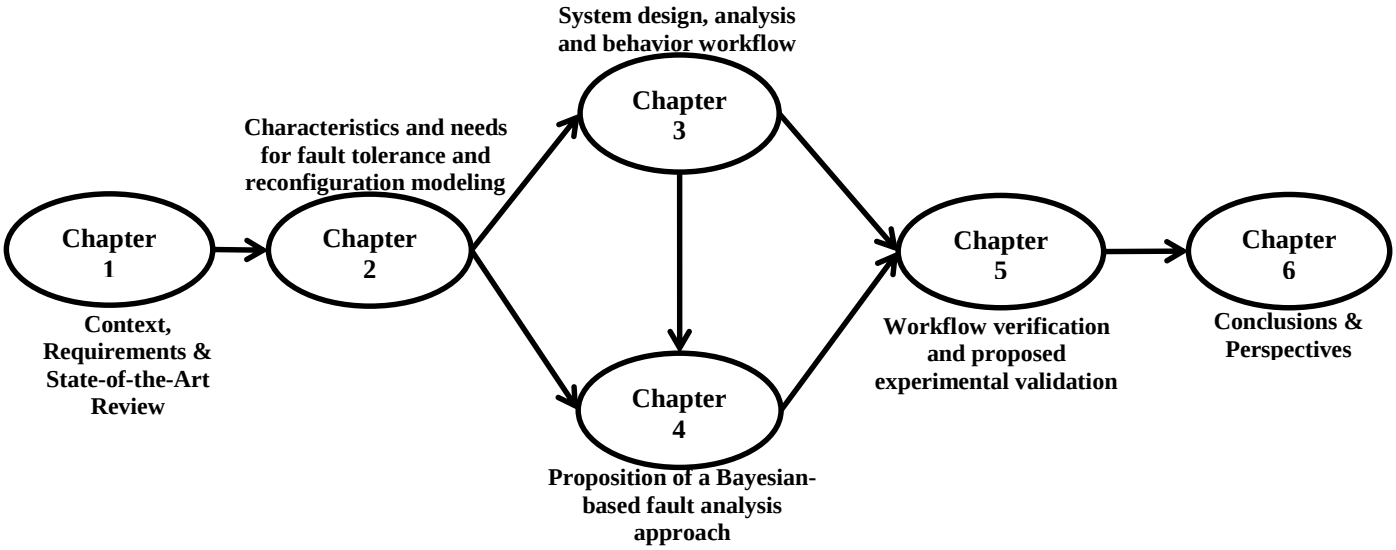


Figure 1. General organization of thesis report

Chapter 1

Context, Requirements and State-of-the-Art Review

Home Automation and Ambient Assisted Living (AAL) domains have been of major influence on this work. Home Automation technologies offer a superfluity of technical solutions with various constraints, while AAL brings extensive real life material in terms of requirements, needs, or use cases.

This chapter gives a general introduction to this important topic (section 1.1). AAL and Home Automation domains are presented in sections 1.2 and 1.3. This presentation enables section 1.4 to list some general requirements identified in these domains. Section 1.5 is devoted to outline the major contributions of this thesis. Finally, section 1.6 summarizes the state of the art and reviews different projects and conducted activities in these domains.

1.1. Introduction

According to Eurostat [Eurostat, 2016], the proportion of population aged between 50-64 years and 65-79 years of the European Union total population has been growing regularly. From a proportion of 19.1%, in 2010, it increased to 19.9% in 2014 for the group of 50-64 years, as shown on figure 1.1.a. (compared with the same proportions for the Belgium and the Romanian populations); and, from 12.8%, in 2010, to 13.4% in 2014 for the population group of 65-79 years (as shown on figure 1.1.b.). It is a real fact: the European population is getting older each year.

This ageing of the population is the result of the combination of several factors, among which are the ageing of baby-boomers, and the decrease of birth rates.

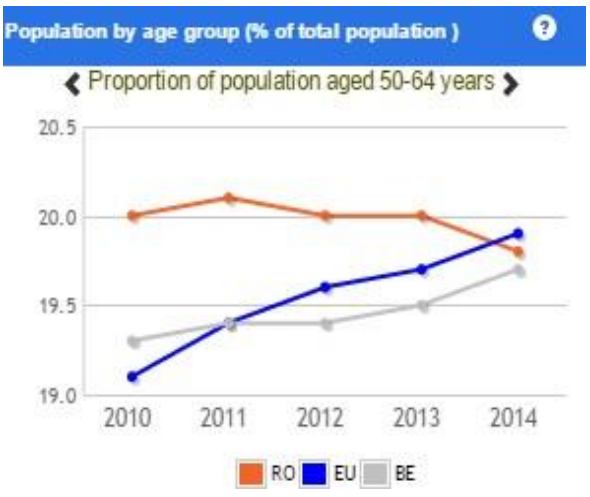


Figure 1.1.a. Proportion of Population by age group [50-64]

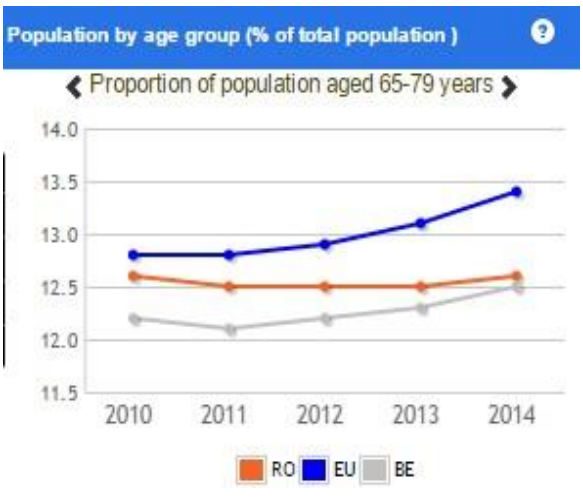


Figure 1.1.b. Proportion of Population by age group [65-79]

Figure 1.1. The European population age growths

The European Office for Statistics (EOS) has emphasized that the number of elderly people will rapidly grow according to the new European population projections for 2008–2060 [Eurostat, 2016]. A noticeable demographic development in the aging of the European populations has been highlighted in recent studies about age groups. The number of people aged 60 years and older will raise nearly 2 billion by 2050. This growing number of elderly people will cause considerable issues for most European countries, since these aging people generally live alone and need intensive care. This also means that European countries will spend huge investments on products securing and enhancing their wealth, safety and security and will have significant entertainment and communications needs.

The aging of the European population has prompted the community to seek solutions to assist this demographic change.

In this context, several problems must be considered. The health field is suffering from a shortage of work, which could result in a general deterioration in terms of care quality. Furthermore, hospitals and retirement homes have limited capacities and will become saturated in the coming years. In addition, stays in hospitals are expensive; regardless of the fact that pathologies and financial aid at this level tend to be limited.

At the same time, it is important to notice that in recent years, the raise of techniques and technologies dedicated to make housing "intelligent" can be clearly witnessed. “Intelligent houses” are equipped with devices of different kinds (sensors, PCs, interactive TV, tablets, smartphones ...). This allows a more comfortable life and a better safety. The most promising application area of these habitats is intelligent home support, which aims at providing services to people with disabilities (due to age or some kinds of diseases) in order to allow this population category to continue living with self-respect in their own homes. These services are based on residential facilities that can be traced back to events related to alarm systems, to automation of comfort, temperature control, monitoring physiological person, etc. Through the correlation of collected information, processing capabilities, storage and communication of new services with high added values for the user can be designed and deployed in individual homes.

Several projects have already been launched to try to address these issues. For instance, the European Collaborative program AAL [AALJP, 2016] has been launched to promote and to fund projects, highlighting thus, the European interest to progress in this field.

AAL systems aim to create improved conditions for elderly people and for people with disabilities. They also aim to reinforce the well-being of elderly persons through the use of Information and Communication Technology (ICT). These objectives are reached by providing different types of solutions for various users with disabilities, in addition to other users such as their relatives and caregivers. These solutions may be for the prevention and the management of chronic conditions, for the social interaction, or for elderly people mobility. There are also other solutions aiming to enhance the quality of life for aging groups and to take care of their health conditions.

The project called “Innovation-Domicile-Autonomie” (IDA) [IDA, 2016], initiated by the Rennaise metropolis, fits perfectly within this framework. It is an assessment of the suitability of using ICT to help elderly people.

In this manuscript, and after a precise inventory of needs of the elderly people, we have attempted to assess the relevance and adequacy of different industrial technologies, to assist people in their homes. Among other things, home automation technologies were evaluated in order to emphasize their potential contributions in assisting elderly people at home. Briefly speaking, the studies showed that a single and a general purpose solution cannot be implemented in all potential faced cases. Each person has his particular requirements and his own needs. This means that the solutions should be adapted to each person. Manufacturers admit their limitations here, where their manufacturing of customized products for each user is too expensive.

In this area, the developed technical solutions require software-based systems in order to fill the gap between the end products of the home automation market and the customized solutions. To fulfill their mission, these software systems must meet several requirements, related to the user needs, as well as how

these systems will be adapted towards user's requests. The question raised here concerns the variety of configurations available and the need to complete, or to better adapt, the existing configurations allowing thus the continuous delivery of the services for elderly people.

1.2. Ambient Assisted Living

As previously mentioned, the growth in the population aging will create significant problems for most industrialized countries. To tackle this issue, the AAL systems are appealed to strengthen the well-being of elderly people, providing emergency services, improving the autonomy and comfort. These systems will postpone the need for a medical environment and allow elderly people to stay longer at home.

The AAL is an approach that aims to create better conditions of life for elderly persons and to strengthen the industrial opportunities in Europe through the use of ICT. One of the key objectives of AAL systems is to extend the time people can live in their preferred environment by increasing their autonomy, self-confidence and mobility. Another objective of these systems is to support maintaining health and functional capability of the elderly individuals, as well as, to promote a better and healthier lifestyle for individuals at risk. AAL systems aim also to enhance the security, to prevent social isolation and to support maintaining the multifunctional network around the individual. For other stakeholders, AAL purposes to support caregivers, families and care organizations and to increase the efficiency and productivity of used resources in the ageing societies.

The AAL Joint Program [AALJP, 2016] defines the concept of AAL through six dimensions categorizing the provided solutions.

- Autonomy: By increasing the autonomy, the self-confidence and the mobility of elderly people, AAL tends to extend the length of time people can live in their preferred environment, and proposing comfort services which significantly improve their quality of life.
- Activities: Maintaining physical or intellectual exercise helps elderly people to remain in good health and prevents a decrease in terms of their intellectual and physical capacities.
- Assisting individuals at risk by promoting a better and healthier lifestyle, and providing emergency treatment services which are liable to predict, identify, and prevent unsafe situations by sending emergency alerts.
- Securing support and maintaining the network around the individual, including family, friends and social activities, to enhance security and prevent social isolation.
- Supporting caregivers, families and care organizations in their everyday activities.
- Streamlining the use of resources dedicated to elderly people by increasing their efficiency and productivity.

There are many solutions allowing addressing these dimensions. Automation of some tasks can enforce autonomy and the use of mechanical aids can improve mobility. Social workers and health professionals can propose activities, support and assistance. Unfortunately, healthcare associations or companies have difficulties for the recruitment of people for these jobs.

A promising solution, for assisting both helped people and helpers, could be to use Home Automation technologies in conjunction with ICTs and human interventions. The next section illustrates this domain.

1.3. Home Automation

Home Automation can be seen as the use of computer technology, electronically and automatically in the context of a habitat in order to assist people in their daily life needs, whether living home or in a special structure. These technologies can serve several purposes; among the most common include the safety of persons, compensation of handicap situations, comfort, energy management and maintaining the social link between single people and relatives.

Therefore, Home Automation refers to the situation where various electronic devices can communicate, cooperate and operate as an integrated system rather than as a collection of independent devices [Cooper & Keating, 1996]. In other words, Home Automation aims to integrate into a coherent whole different devices around the same interface adapted to the specific needs of each user.

The state of the art in the field of Home Automation covers the use of the technologies in this domain through research projects and intelligent habitats. The latter are used to support research teams to implement and to test, in real conditions, their developed methods and tools. Even that it is not intended to be exhaustive, this state of the art still allows interested researchers to list a number of projects that have had an impact in research in Home Automation in recent years and intending to make more "intelligent" the habitats of tomorrow. In section 1.7, some projects in the domain of Home Automation are listed and briefly presented.

1.4. Requirements categorization

The domain of Home Automation involves a new kind of technologies to simply manage installations and to meet specific user needs. The lack of tools (able to operate any Home Automation technology and to create solutions for specific needs of elderly people) makes the adoption and use of these technologies marginal. This absence of tools may be due to the complexity and to the number of requirements inherent to Home Automation systems.

This section aims to identify a set of required properties that Home Automation systems should ensure in order to meet elderly people needs and expectations.

To develop such a system, users and deployment environments specificities (that should be taken into account) include [Lézoray *et al*, 2010; Lézoray *et al*, 2011]:

- Users may have specific diseases: the Home Automation system must be customizable, depending on the final users capabilities;
- The acceptability of some services may be different: some services must be customizable depending on the final users;
- Target platforms: the system must be customizable depending on the capabilities of the available target platforms;
- Home configuration: the system should be customizable depending on the current configuration of the existing devices at home.

The previous specificities should be realized so that the desired systems for elderly people need to have certain criteria which are specific to the home automation assistance tasks. From the literature review of pervasive systems [Virone & Sixsmith, 2008] especially in the domain of AAL solutions, a set of requirements (ensuring efficiency and meeting the expectation of the final users) that an AAL system should meet has been concluded.

Some requirements are categorized as functional requirements, related to the technical functionalities of the solutions provided by the system, such as interoperability of heterogeneous technologies and the ability to

recover potential functional errors. Other requirements are considered to be nonfunctional requirements, related to the general acceptance of the final users, such as the cost and the ease of use [Schneider & Becker, 2008]:

1. ***Adaptability***:

Adaptability is defined as: the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed [IEEE, 1990]. As a crucial requirement for AAL systems, Home Automation services must address the specific needs of users (closely related to their disability) as well as the configuration of their living space. In [Lamprinakos, 2010], it was stated that when defining AAL services in a Home Automation system, different requirements of the various users, according to their capabilities and disabilities, in addition to the evolution of these requirements must be considered, as well as the evolution of the living space. This includes the ability of adapting the services dynamically without a user intervention. Home Automation systems must address the specific needs of users, closely related to their disability and the configuration of their living space. The ability of a service, or a software, to adapt its behavior to individual users, typically it is based on information acquired about its user(s) and its environment. This information includes the evolvement of the disease or the disability of a user, or according to the change in the environment.

2. ***Interoperability***:

The heterogeneity issue of technologies and communication protocols (found in a home) as well as those to be integrated should be addressed. Interoperability is defined as being the ability of two or more systems, or components, to exchange information and to use the information that has been exchanged [IEEE, 1990].

According to [Perumal *et al*, 2008], heterogeneous systems (in smart home environment) include mainly:

- ✓ Home Entertainment system;
- ✓ Surveillance and Access Control;
- ✓ Energy Management;
- ✓ Home Automation and
- ✓ Healthcare and Assistive Computing systems.

These heterogeneous systems and the devices that operate these systems are belonging to different middleware [Miori *et al*, 2006], totally isolated from each other, and running with different technologies such as OS and programming platforms.

Systems heterogeneity is considered as an essential challenge for the integration of all devices according to a standard mechanism that makes the heterogeneous systems interoperable and "speak the same language" apart from the used technology [Miori *et al*, 2006].

3. ***Availability***:

The availability, in case of a failure, is an essential feature that must be taken into account to ensure the delivery of a service. In other words, we talk about reconfiguration which is the ability to tackle the failure of service execution through the definition of different scenarios during the design time. Reconfiguration is triggered on activation failure. It performs compensation on an operation basis, using off-line defined reconfiguration rules in resource descriptions.

According to [Lankri & Philippe, 2009] reconfiguration is defined as: "A process to perform changes to the system to ensure correct suitability to the current context. Changes can be made to the software, the hardware, or to both, and are triggered due to either objective changes (adaptively), or to service failures (fault-tolerance)".

The availability in case of failure is an essential feature that must be taken into account to ensure the delivery of a service [Lankri *et al*, 2009]. In other words, it is the ability to overcome the failure of service implementation through the definition of different service execution scenarios during the design time. Two

reconfiguration modes can be considered: Off-line and on-line modes. Off-line (Static) reconfiguration performs compensation on an operation basis, using statically defined reconfiguration rules in resource descriptions [Lankri & Philippe, 2009; Lankri *et al*, 2009; Lankri, 2009]. Whereas, on-line or dynamic reconfiguration uses recent system status in order to compute new reconfiguration rules in a dynamic manner.

4. ***Distributed access points***:

This requirement refers to the degree to which a product, a device, or a service is available to as many people as possible. While the majority of Home Automation solutions are restricted to individual use, it is necessary to address the various multi-user large-scale (e.g. Clinic) environments. Accessibility includes also that the services or the Home Automation system are accessible to other users such as patient relatives or to the medical stuff.

The services provided by the AAL system need to be accessible from different access points, in order to make all users (medical staff, elderly people and their relatives) participate using the services according to their roles [Queirós *et al*, 2013]. For example, the medical stuff should have access to the services related to the health monitoring in order to analyze the collected data and to make the right decision and send, in-time, feedbacks to the elderly people at home, or to make on time intervention in case of emergency problems.

5. ***Maintainability***:

This requirement is defined as the ease with which a software system, or component is built to correct faults, improves the performances (or other attributes) or adapt to a changing environment.

6. ***Cost***:

This issue includes the cost of the electrical installation, software supervision and the intervention of an expert for the configuration and the adaptation of the system.

7. ***Acceptability***:

The user acceptability requirement is crucial. For instance, if cameras and other sensors can gather relevant monitoring activities information to adapt an "intelligent" behavior, their presence should, in no way, interfere with the privacy of an individual or even annoying the user.

8. ***Ease of deployment***:

The configuration must be accessible to non-experts in automation people to adapt the system to the dependent user whose needs can change rapidly.

In this section, important requirements allowing increasing Home Automation assistance systems' efficiency and effectiveness towards the users have been underlined. These requirements are crucial to attain solutions supporting elderly people to live both independently and actively, based on adding non-functional attributes to the ICT solution [Sun *et al*, 2010].

1.5. Expected Contribution of this thesis

Inspired by the achievements in the field of pervasive systems, this thesis helps to improve the flexibility of software systems while maintaining a high level of reliability. The expected contributions are at three levels.

1. Proposition of a new model that improves application flexibility and enables the connection of components and meets the desired requirements towards user needs and expectations;

2. Development of an application based on the concept of Model Driven Engineering (MDE), to create, edit, simulate and validate the structure and behavior of the component assemblies before their (re-) deployments;

3. Proposing a runtime environment built on the foundation of service-oriented software architecture, supporting adaptation properties, evolution and openness required by the new component model.

In the expected contributions of this thesis, the problem of service reconfiguration is the main concern of the home automation systems. In fact, the service reconfiguration (availability) had been chosen according to a detailed evaluation of several projects and systems that had been accomplished under different Home Automation approaches, such as AAL systems and the systems developed in the Lab-STICC research laboratory. These projects and the corresponding solutions will be presented in more details in the following section. The conducted evaluation, summarized in Table 1.1, is based on the set of functional requirements described in section 1.4.

System	Requirements			
	Adaptability	Interoperability	Reconfiguration	Distributed access point
GiraffPlus **[Coradeschi *et al*, 2013; Cesta *et al*, 2013]**	√	√	**X***	√
InCASA **[Prestileo and Fiore, 2012; Lioudakis, 2010]**	**X**	√	**X**	√
eCAALYX **[Boulos *et al*, 2009; Boulos, 2009]**	√	√	**X**	**X**
DANAH **[Lankri, 2009]**	**X**	√	√	**X**

Table 1.1 Evaluation of Home Automation systems

***GiraffPlus is performing a configuration planning that could be employed to ensure the reconfiguration, but it is not used for this goal**

In this table, we make an evaluation of various systems developed in different projects and labs. This evaluation is based on the defined criteria and requirements. We aims with this evaluation at identify the fulfilled criteria by the evaluated systems, to stands on and find out which criterion (criteria) that is (are) not considered in these systems.

1.6. State-of-the-Art Review

The review proposed in this section is made of three subsections. Subsection 1.6.1 starts with a description of the AAL domain (from which social requirements have emerged) and it presents some projects on this topic. Some AAL solutions and projects (their use and goals) are then introduced in subsection 1.6.2. The overview of this domain makes it possible to sense, more precisely, the needs for a new type of software for the field of Home Automation. Finally, subsection 1.6.3 presents some projects that have been conducted within Lab-STICC research teams.

1.6.1.Background on AAL

Ambient Assisted Living is a hot topic in Europe. Several projects have been conduct in order to address different aspects and to try covering the inherent needs to a home-keeping situation. Several projects and researches have been conducted in the domain of AAL systems for elderly persons and people with diseases (specifically in the industrial and developed countries). In recent years, the EU has funded a number of projects promoting independent living for elderly. As examples of these projects, the project funded by the ICT Policy Support Program under the Competitiveness and Innovation framework Program (CIP) [ICT PSP], and the Seventh (FP7-2007-2013) Framework Programs for Research and Technological Development [Cordis, 2016].

The AAL Joint Program [AALJP, 2016] is exactly focused on this topic. Several AAL projects address smart homes for the elderly. The solutions proposed by the AAL projects are based on created technologies. They are developed in a close loop with the users, and the projects are expected to result in commercialization in the near future. However, the proposed solutions tend to be specific for a certain class of illnesses and the scientific novelty can be limited. In many of them, services provided are mostly based on a one-to-one correspondence between sensor data and actions that the system performs.

The projects funded by this program are categorized, according to the afforded solutions, into six key groups [AALJP, 2016]:

- ICT based solutions for the prevention and management of chronic conditions for elderly people.
- ICT based solutions for advancement of social interaction of elderly people.
- ICT based solution to provide enhancement in the independency of elderly people.
- ICT based solution to provide mobility solutions to elderly people.
- ICT based solution to provide services that facilitate the self-management of daily life activities.
- ICT based solution to support occupation in life for elderly people.

1.6.2.Projects in AAL

The AAL Joint Program is a collaborative association of twenty European Union member states, in addition to three Associated States. They group together the AAL Association, whose main objective is to enhance the quality of life of elderly people through the use of Information and Communication Technology (ICT). Their main activity is to found R&D projects in the AAL domain and to publish annual calls for project proposals.

GiraffPlus Project:

GiraffPlus (January 2012- December 2014) is an EU project that was funded by the CIP and the FP7 [Coradeschi *et al*, 2013] in the area of ICT for ageing well. This project deals with enhancing the well-being of elderly people and extends their independency of living, by the early detection of possible health problems to decline them and providing services to assist those having deficiencies [Cesta *et al*, 2013]. The main components of GiraffPlus system consist of a network of sensors that are ubiquitously integrated in the home and categorized into physiological and environmental sensing devices. The collected data from these sensors is interpreted by an intelligent system which, in turn, prompts alarms or reminders for two kinds of users: the elderly users at home (primary users) and related caregivers and health professional (secondary users). The system consists also of telepresence robot, the Giraff robot, which is controlled over the internet by a remote user to move inside the home, in order to assist the elderly users to sustain their social contacts with other users. Giraff robot has an interface (similar to Skype) and is equipped with communication facilities [Cesta *et al*, 2013], as illustrated in figure 1.2.

The project newness, compared to other AAL systems, can be summarized in the following:

- Easy communication tool through the robot;
- The system uses a high level reasoning approach for context recognition, configuration planning and personalization and interaction services;
- The development of a system that contains both sensor network and a telepresence robot. Apart from other AAL projects, GiraffPlus has been evaluated in real homes, where it has been installed in at least 15 homes in three EU countries (Italy, Spain and Sweden).

The system developers have followed an evaluation strategy of several steps in order to develop a solution based on the acceptance of the users (by including both primary and secondary users in the experimentation of the system). An initial integrated version of the system has been implemented in order to develop a first version of the solution (including the robot, as well as the different used sensors) with the participation of the primary and secondary users.

Also, a first visualization module has been achieved to envisage different activities in the system. This visualization has been evaluated with secondary users (since the evaluation with primary users is still under development [Coradeschi *et al*, 2013]). The system, in general, is still under development according to the plan of the project work.

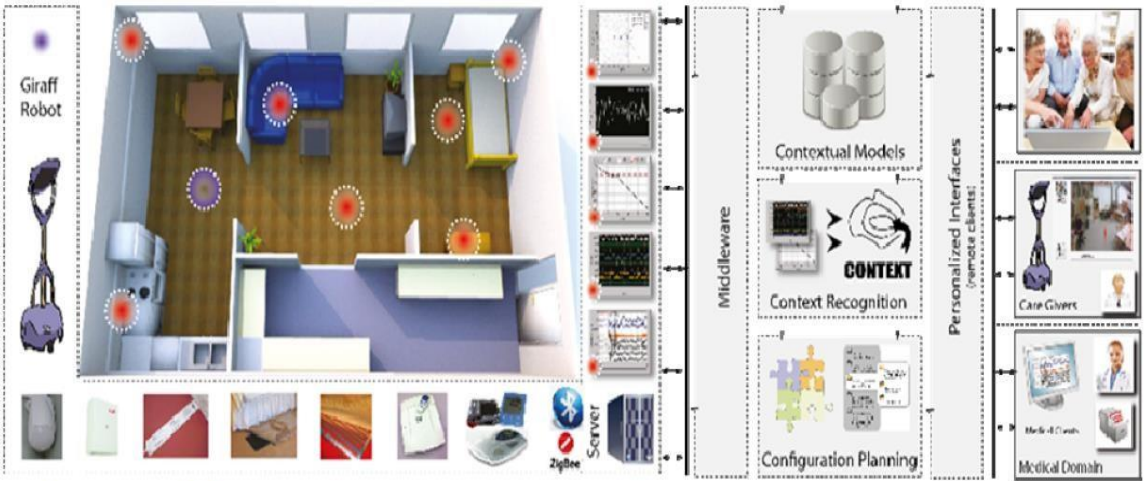


Figure 1.2. The GiraffPlus system overview

InCASA Project:

InCASA project ("Integrated Network for Completely Assisted Senior citizen's Autonomy, InCASA" (April 2010 - September 2012) deals with citizen- centric technologies and public/ private services networks, to help and protect independent elderly people, prolonging the time they can live well in their own home by increasing their autonomy and self-confidence [Lamprinakos, 2010; Prestileo and Fiore, 2012]. The main objectives of the InCASA project can be summarized as follows [Lamprinakos, 2010; Lioudakis, 2010]:

- Provide elderly people with solutions to monitor their health conditions when they are at their own homes.
- Provide General Practitioners (GPs) and health professionals with technological facilities to remotely analyze and control elderlies' health conditions.
- Ensure care continuity through the interaction between the elderly at home and his/her health professionals.
- Ensure the ability to control the electronic devices remotely through the architecture of the system.

The objectives of the InCASA project can be accomplished through the use of technologies and services for the elderly people to strengthen their well-being at home, which in turn increases their independence and minimizes the need for health assistance.

In order to meet its objectives, InCASA architecture is organized into three main tiers, as illustrated in figure 1.3, which purposes to provide integrated solutions and services to control the environment and health conditions for the elderlies and gives feedback and alerts to the users. The three tiers [Lioudakis, 2010] are:

- The Front-End which represents the home infrastructure and consists of both Body Sensor Network (BSN) and Home Sensor Network (HSN) that are responsible to sense different body metrics and the environment metrics respectively, and propose the collected information to a Home Base Station (HBS) that is responsible for the collection, analysis and communication of the data.
- The Back-End which reflects the center of the data evaluation comes from the HBS in order to take decision and send alerts to the elderly at home. The Back-End components are deployed at the health care service provider, and mainly consist of the Smart Personal Platform (SPP) (which "constitutes the information broker and the overall point of control with respect to the operational aspects of the system) [Lioudakis, 2010].
- The healthcare applications and practitioners user space which is managed internally by the health center or externally by third parties. The applications denote all clinical applications involved in the system.

The system platform is tested in six different locations in Italy, Spain, the UK, France, Greece and Denmark. Roughly, 200 patients participate in these tests. Extensive experience and data is collected within these real-world installations [M2M Journal, 2013].

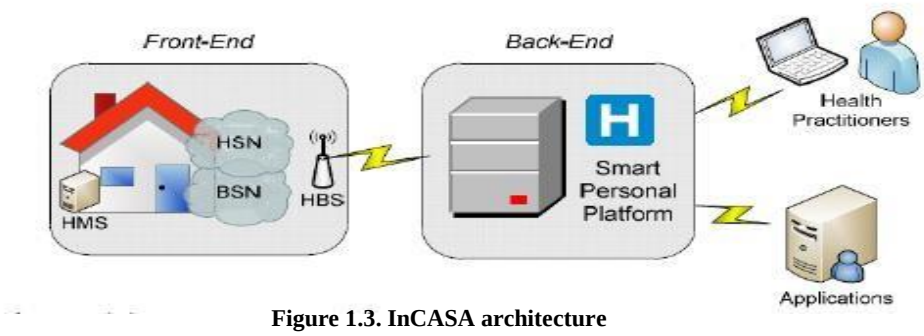


Figure 1.3. InCASA architecture

eCAALYX Project:

The project eCAALYX (Enhanced Complete Assisted Living Experiment) (2009 - 2012) was developed under the AAL Joint Program.

The project eCAALYX continued the advantages of the architecture and functionality that are developed by the project called CAALYX [Boulos *et al*, 2009; Boulos, 2009], and aims to provide a solution in order to improve the well-being of elderly people at home and extends their independency for longer periods, by developing home and outdoors 24/7 health telemonitoring / telehealthcare services for elderly people, including those with multiple chronic conditions. The key objectives of eCAALYX project are [Boulos *et al*, 2009; Boulos, 2009]:

- Monitor the health of elderly people and people with chronic conditions.
- Based on the health monitoring, improve the quality of life of elderly people by strengthening their autonomy and safety.
- Provide continuous health support for elderly persons that assists in prolonging the need for intervention from health professionals and avoiding the descent of health circumstance.

The project objectives could be accomplished by providing an ICT solution that meets all users' expectations.

The proposed ICT solution is based on architecture, as illustrated in figure 1.4, which consists of three main interconnected subsystems [Boulos *et al*, 2009]:

- Home Subsystem, which includes number of equipments and devices such as Customer Premises Equipment (CPE), Set-top-box, interactive TV and stationary home sensors.
- Mobile Subsystem, which consists of a "smart garment", integrated with sensors, wireless Body Area Network, Wearable Body Sensors and mobile phone.
- Caretaker Subsystem, which consists of the remote Caretaker Server and Auto Configuration Server.

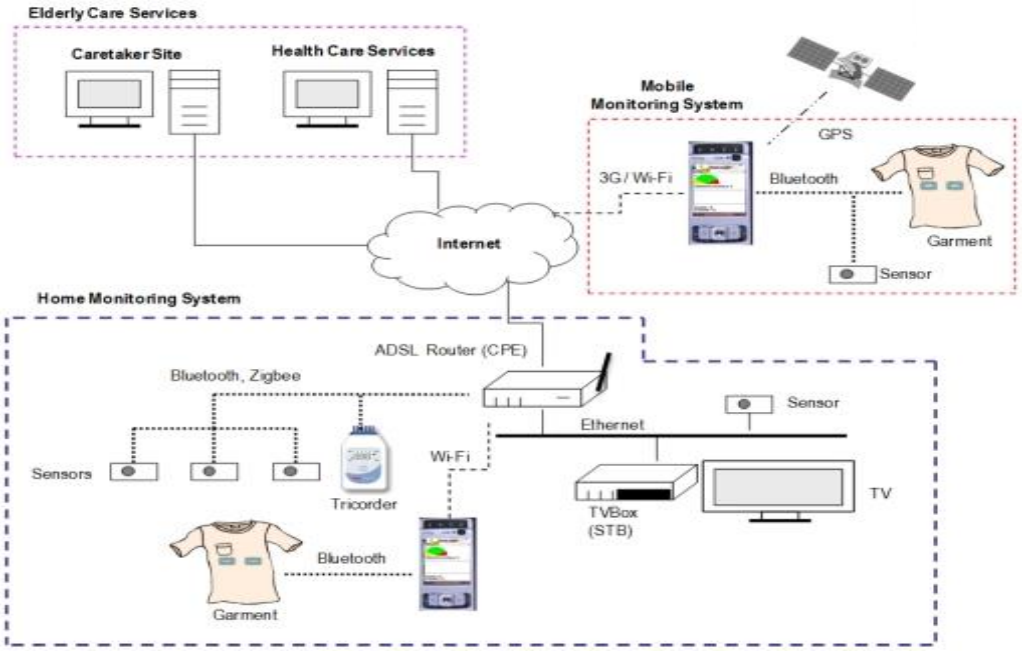


Figure 1.4. eCAALYX main components and subsystems

The eCAALYX system main process includes obtaining vital signs and data from different sensor types, in order to record this data locally, analyze it, send it (when an emergency case is detected) to a remote health care center in order to take the appropriate action by the health care specialists.

The eCAALYX system provided an ICT based solution for the prevention and the management of chronic conditions of elderly people. The project has been experimented in real situations and has developed a body of sensors and a Smartphone application which can remotely monitor different body movements and takes physiological measurements of heartbeat and body temperature.

1.6.3.Projects conducted in Lab-STICC

In recent years, the themes of aging and assistance to individuals with disabilities are particularly among the topics to which Brittany Region and the General Council of Morbihan bear interest and help funding projects aiming to find solutions. In 2003, they financed the project COHAND and in 2005 the project QUATRA which includes several stakeholders in the field such as the Lab-STICC, the IRISA laboratories and the center of rehabilitation and Functional Rehabilitation of Kerpape (CMRRF) [Lankri, 2009].

The team work MOCS (Méthodes, Outils pour Circuits et Systèmes) of Lab-STICC participated for several years in projects related to home automation to support dependent persons [Allègre, 2012]. The research area "pervasive systems for disability assistance" brings together the work to application purpose, methodological flows for the deployment of home automation systems that must be the least invasive as possible while ensuring an acceptable level of quality of services. This theme is based on a close cooperation of many years with the center of rehabilitation and Functional Rehabilitation of Kerpape that has plenty of home automation infrastructures. Brittany Region and the General Council of Morbihan have shown a special interest in the problem of dependency for several years and have funded projects that aim at bringing novel technological solutions.

COHAND Project:

In the context of Home Automation, COHAND [Belabbas, 2007] focused mainly on two aspects. The first is the linkage range of services / navigation, which is piloted by the system, in order to facilitate the user navigation, through the wheelchair, to the desired service place. The second is the quality of service that ensures that the system can best provide the service even in the presence of hazards (obstacles or service failures).

This project has focused on the environmental control and the mobility of people with disabilities in the studying of the services offered and the navigation of electric wheelchair [Belabbas *et al*, 2006a, Belabbas *et al*, 2006b].

When selecting a service, the system pilots the wheelchair and services related to navigation (e.g. open doors, turn on the lights) to finally activate the requested service (e.g. watching television). The system contains reconfiguration rules in case of failure. The used approach, to calculate the appropriate way to access the service, is based on topological models representing the environment. The topological model is a non-deterministic model of all the possible paths from any area to any other one. The system is able to reconfigure (change path and use of equipment with similar capabilities) to provide the requested service anyway. The system has the ability to respond quickly and efficiently to occurring changes. The intelligent wheelchair cooperates with the environment control through wireless communication, and the proposed approach provides several rules controlling the interaction between the wheelchair and the environment. These rules are based on types of transfer.

This project led to an experimental simulation platform. From a practical point of view, it is not deployable in reality, and therefore it was unusable as a whole for dependent people.

QUATRA Project:

Following the COHAND project, this project leads to a real experimental platform tested in CMRRF Kerpape [De Lamotte *et al*, 2008] in collaboration with the Lab-STICC. It targets people with severe disabilities. The architecture (connecting terminals) has been defined to ensure the delivery of home automation services. The user, equipped with a PDA can get connected 1) to a mobile terminal embedded in his chair to control it and, 2) to the fixed terminals distributed in the environment. Several experiments have been made for testing a set of scenarios that meet the needs of patients in the center.

In order to facilitate the access to everyday services, the users can control their home automation appliances (shutters, doors…) through a single interface as well as multimedia services through a suitable interface, where a set of basic services and scenarios are published. Upon activation of a scenario, the system automatically generates and activates the sequence of commands to meet the request of the user.

The advantage of this approach resides in the fact that the user does not have to make an effort as it is at his disposal to have full control on the equipment of his house with one or two selections via the adapted interface. QUATRA mainly aimed at using consumer electronic components.

This project has taken place in a real experimental platform deployed to users in real situations. The QUATRA project outputs have been the subject of experiments in the Kerpape Centre, in rooms for disabled patients equipped with home automation devices.

DANAH Project:

DANAH (Domotique, Aide à la Navigation et Assistance au Handicap) project started in 2006. The main objective of this project is to set out the different scientific issues from previous projects to improve the propositions that have been made. It followed the COHAND and QUATRA projects and aimed to refine the principles that have been developed and to overcome new appeared difficulties [Lankri, 2009].

The project proposed a middleware that integrates a process of multilevel reconfiguration, static and dynamic reconfiguration of home automation services, in addition to the topological reconfiguration of paths for the movement of an intelligent wheelchair.

The vital goal of the project was to achieve a system incorporating aspects services, navigation, reconfiguration and design flow that can be deployed rapidly in real situations.

DANAH middleware is based on client-server architecture (distributed system). Home Automation terminals (DANAH servers), are distributed in the environment [Lankri & Philippe, 2009]. A terminal can thus be connected to a Home Automation bus KNX-EIB [Lankri *et al*, 2009] to control equipments connected to it. Other terminals are responsible for monitoring equipment to local activation via infrared or Bluetooth such as multimedia equipment. DANAH is a system that works perfectly in a real situation. DANAH is a multi-platform system with a simple graphical user interface that can be deployed on embedded devices. It works on PC and NOKIA N800 smartphone. It supports Bluetooth and Wi-Fi technologies of communication and home automation technologies KNX/EIB and Infrared. The user, equipped with a PDA (Personal Digital Assistant) embedded in his wheelchair, has the opportunity to reach the services and methods published by these terminals.

DANAH combines existing technologies in order to provide appropriate assistance for elderly people and people with disabilities. It also provides its services through a set of domestic devices which control the environment and a navigation system to assist the elderly people to perform independently their everyday activities at home.

These services are provided to the user through a specific user interface (PDA connected to an intelligent wheelchair) responsible for controlling the environment devices as well as the wheelchair to navigate to the place of the service when it is called by the user.

DANAH middleware is developed to allow remote control and automation of different home automation equipments independently of the communication protocol used.

DANAH consists of two architectures; one for the hardware architecture in which DANAH is a distributed system consisting of remote hosts that collaborate in a peer to peer network to deliver services. Users have each a computer with which they know about the available offered services. DANAH uses KNX/EIB technology which is the only open standard for home control.

The other architecture is the client/server based software architecture (servers are running on remote hosts while clients are running on users computers).

Nevertheless, the point here in DANAH, is that user interfaces are general and there is no user's personalization, but the wheelchair automation is achieved through specific wheelchair drivers.

For the environmental control, DANAH proposes an environmental control model to represent devices and services, in addition to a model for scenarios of actions. From DANAH's view, controllable devices are called resources because each one can be used by a limited number of users at once. The resource model includes a user friendly name and a list of provided functions called operations which may success or fail during execution, a list of running modes and a list of states. Modes express the ability of a resource to have different running schemes, for instance, doors can have a normal and an emergency running modes. The resource model is associated to a physical device through a resource device model.

DANAH supports two types of resources: concrete resources and composite resources as aggregation of resources. Operations can be runtime controlled (atomic) or composite. Atomic operations are directly run by calling appropriate methods within the runtime. Composite components are operations described using literal expressions such as:

Operation "Enter" {expression="SEQ(Door.Open Light.On)"}

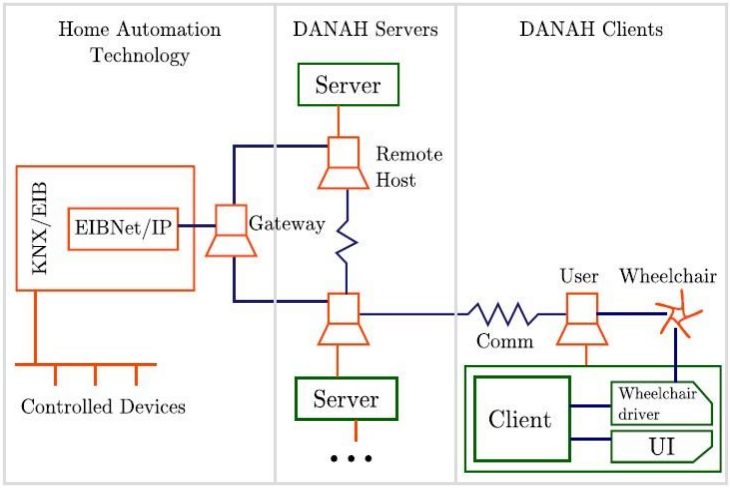


Figure 1.5. DANAH architecture

AIDOM Project:

The AIDOM (Aide et Intervention à Domicile) project is related to home control systems and was developed in the thesis of Truong [Truong, 2010]. This project focuses on data mining for the automatic detection of everyday life scenarios. The originality of this project lies in the analysis of automation activities of residences without additional equipment (e.g., sensors, cameras), by using the commands (requested services) that the user transmits to the home automation system. The evaluation criterion is the energy spent by the user and where the algorithm aims to minimize by proposing scenarios for automatic activation of services.

In this project, each service is considered as a repeating pattern over time. A frequency analysis of the requested services allows to propose scenarios as well as to launch an alert if the person deviates too much from his daily life activities.

SIGAAL Project:

The SIGAAL project (Inter-generational Support Services for seniors in their homes) provides home care to help dependent elderly people [SIGAAL, 2016]. This project aims to reduce the isolation of these individuals by providing personalized services to maintain or strengthen social ties. As part of this project, [Lézoray *et al*, 2011] proposes a design process enabling dynamic adaptation of services in an IT context broadcast.

Using development methods based on model driven engineering, the approach of "medium" is proposed. It is about the interconnections between software components that can be of different types according to the service concerned and to the type of components connected to it.

ASIM Project:

ASIM (Assistance by Intelligent Multi Protocols' Server) is a project related to "Health and autonomy of life through digital technology" as a future investment. Driven by the company Vity Technology and in collaboration with the company Kaptalia and the University of South Brittany, this project concerns the construction of a multi-protocol Home Automation server capable of interpreting data from bio-medical sensors with the aim to facilitate home support for vulnerable people.

Intelhome Project:

The project was developed in the thesis of W. Allègre [Allègre, 2012] following the AIDOM and DANAH projects. It aims to define a design methodology for home automation assistance systems. In this work, a model driven engineering flow for assistive home automation systems is planned. Firstly, integration had been proposed of a home automation non-expert (e.g. family member / caregiver, occupational therapist) in the design process to better take into account the requirements of disabled people. Also, a main target was to limit the costly interference of the expert on the development. A Domain Specific Modeling Language (DSML) is developed allowing one to describe home automation system by manipulating high-level abstraction concepts. This DSML is composed of two views: the environment view to model the home automation environment and the interaction view to model user-system interactions. Two types of interactions are defined, in the interaction model, to consider the greatest number of prerequisites. Thus, it is possible to define requests (i.e. services) and temporary prohibitions (i.e. modes) on home automation devices. The proposed work had been evaluated and assessed on the Kerpape Smart Home case study with students and occupational therapists which provided interesting results on both the proposed design flow and the DSML usability [Allègre *et al*, 2011].

Willy has developed an interaction model providing the notion of scenarios of a service execution. In the developed domain language, each service can be represented using different semantic levels, when the non-expert designer can use this definition to introduce the reconfiguration concept, but without any consideration of how to select these scenarios. Intelhome may provide the reconfiguration if the non-expert defines specific services scenarios in a reconfigurable manner, but without any mechanism of how to define the suitable scenarios tolerating the failure situation.

Platform proposed in Cédric's work:

The work conducted in Cédric's PhD work [Cédric, 2015] has attempted to realize several objectives. The global objective of this thesis is to develop a pervasive architecture that let the designer choosing and deploying healthcare services depending on the devices and networks available in a smart home. More precisely, the proposed solution needs to:

- Take into account the human context, more precisely the medical profile of the handicapped person;
- Take into account the hardware and the communication context, in terms of interoperability;
- Provide service continuity in terms of reconfiguration of system elements;
- Enable cooperation between different devices to provide a health service;
- Enable the sharing of the same living space between different users.

In this research work, a task for Disability Assistance is defined as the process of demanding and activating a resource (hardware or software) to deliver a service to the disabled users.

A task manager is developed in the system to control the resources available in the system. The task manager is responsible for defining which resource offers the most optimal execution of service. This application ensures the life cycle of the task, from the time the choice is made by the patient until his execution in the living space. Such an example of a task is the task turn on the light « _task_LampeEib1_On », which corresponds to the lighting of one of two lamps connected to a connection bus. Each task may have a degraded version, referring to delivering the service with degraded resource conditions (a light with less power). Additionally, each task has an alternative version to ensure the continuity of the service provided by the resource, in terms of reconfiguration.

The goal is to obtain all compatible solutions "task / resource", that ensuring the deliverance of optimal services to users. The optimal resource is selected based on the knowledge about the utilization rates of each resource. Before being deployed, the task manager ensures that the execution will not disrupt the functioning and integrity of the tasks already run. Once all these steps validated, the task is well deployed on the resource to meet the patient's needs.

XAAL Project:

The XAAL project, developed by the IHSEV/HAAL team of Telecom Bretagne [XAAL, 2016] aims to address interoperability issues of home automation systems. XAAL provides a functional distributed architecture, developed and continuously upgraded, addressing the issue of interoperability between different components and devices using different technologies. Many vendors offer home-automation solutions. There are almost as many home automation protocols as manufacturers (or alliances of manufacturers). The interoperability between home- automation solutions is a key issue nowadays: how to make a device from vendor A (e.g. a switch) talking with a device from vendor B (e.g. a lamp)? Existing solutions for interoperability issues play around the idea of gateways: a box (a small computer) is equipped with two or more modules, each module "talking" a given home-automation protocol; then a piece of software over those modules makes it possible to forward messages between protocol A and protocol B. XAAL proposes to reorganize and formalize architectures of those boxes (which are in fact all composed of the same functionalities). Functionalities

are cut into well-defined functional entities, communicating to each other via a messages bus over IP. Each functional entity may have multiple instances and be deployed in different ways over physical entities. Figure 1.6 shows the general architecture of XAAL bus, which consists of the following parts:

- Native Equipment: devices communicating using the XAAL protocol natively.
- Gateways: translate messages between a manufacturer protocol and the XAAL protocol.
- Schemas Repository: contain schemas describing devices.
- Database of Meta-data: actual configuration of devices.
- Cache: stores notifications of sensors about state changes.
- Automaton of Scenarios: advanced home automation services.
- User Interfaces: generate web pages or provides REST API for mobile applications or external servers [Lohr *et al*, 2015].

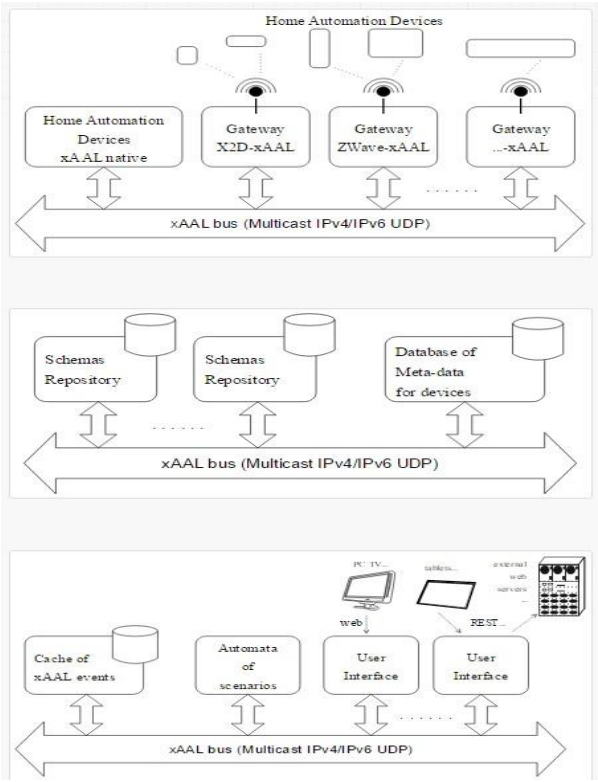


Figure 1.6. General Architecture of XAAL bus

1.6.4. Synthesis and Discussion

After introducing some existing projects and researches under the AAL joint program, as well as in Lab-STICC, this section is summarizing the main properties of these projects, which meet the requirements described in section 1.4. Additionally, this summary puts the expected contribution of this thesis in relation with the categorized requirements, to illustrate the direction and the main objectives to be realized in this research.

1. GiraffPlus:

GiraffPlus is configured in a way to allow the system to be adaptive (i.e. services and sensors can be easily added). In addition to this adaptability feature, the system provides its services to various users (including the elderly users, as well as the family members and caregivers) from different access

points. Therefore, the architecture of the system is designed in a way which hides the heterogeneous nature of its different components and the heterogeneous communication protocols. This adaptive nature in terms of the architecture and the user's requirements is "considered" as the mechanism for reconfiguration. This adaptive feature is implemented as a system's component called (the configuration planner). This component is in charge of generating the configuration of the connected sensors and actuators allowing, thus, providing the requested information (state variables) to perform the corresponding activity. This configuration is assumed be adapted to changing conditions, including the malfunction, without any details of how a configuration may be adapted to these malfunctions. To conclude, GiraffPlus does not consider any specific mechanism of errors tackling.

The system is performing a configuration planning, which should be adapted to changing conditions, including the malfunction, without any details of how the configurations may be adapted in case of failure. Moreover, during the experimental validations conducted on the system the configuration issue was not tested.

2. InCASA:

The InCASA system's architecture consists of different elements which are connected with each other using different heterogeneous protocols (such as Bluetooth, WiFi, and ZigBee). The functional architecture of InCASA is realized using four levels of communication at home, with a service center and with external users (such as relatives), caregivers and general practitioners. This fact allows to the different users of the system the ability to use the system and to interact with it from different locations. Nevertheless, InCASA considers the system structure as a static structure, which is not adaptable, or changeable, under any conditions such as any potential malfunction.

3. eCAALYX:

The eCAALYX system's architecture allows the customization of the provided services from user's perspectives, as well as from the deployment environment perspectives. Interoperability, between different system components, is also underlined. The risk of failure is introduced in eCAALYX in relation with the problem of heterogeneous technologies and subsystems of the whole system. This type of risk is underlined following interoperability guidelines. No specific reconfiguration has been proposed in this system.

4. COHAND:

The COHAND system proposes a service static reconfiguration allowing calculating the appropriate way to access the service and to ensure the availability of the service. The main point is that COHAND misses a design flow expressing the global system's approach. The service static reconfiguration proposed in this system is a redundancy-based approach.

5. QUATRA:

The QUATRA system focuses on providing the users with an "adapted interface" allowing facilitating the devices usage in the living space. QUATRA offers no reconfiguration mechanism.

6. DANAH:

The DANAH platform offers reconfiguration mechanism to ensure system continuity.
Additionally, it offers the interoperability of different heterogeneous connection technologies.

7. AIDOM:

The AIDOM system is developed to provide adapted services to elderly users from their perspectives and needs. AIDOM offers no reconfiguration mechanism.

8. SIGAAL:

The SIGAAL project aims at providing an adapted solution to reduce the isolation of elderly people and to help the users to interact with their relatives via an interactive TV services, rather than focusing on the continuity of the provided services. SIGAAL offers no reconfiguration mechanism.

9. ASIM:

ASIM offers no reconfiguration mechanism.

10. Intelhome:

Willy has developed an interaction model providing the notion of scenarios of a service execution. In the developed domain language, each service can be represented using different semantic levels, when the non-expert designer can use this definition to introduce the reconfiguration concept, but without any consideration of how to select these scenarios. Intelhome may provide the reconfiguration if the non-expert defines specific services scenarios in a reconfigurable manner, but without any mechanism of how to define the suitable scenarios tolerating the failure situation.

11. Work proposed in Cédric's Thesis:

This works offers reconfiguration of the tasks performed on the resources.

12. XAAL:

This platform is developed to undertake the interoperability issues in home automation systems. XAAL offers no reconfiguration mechanism in the devices defined in the schemas defining the device.

Table 1.2 illustrates different existing projects and systems presented in sections 1.6.2 and 1.6.3, with the categorized functional requirements fulfilled by each project, and how the expected work in this thesis is directed to accomplish the promising functional requirement(s).

This table clearly shows that the majority of works did not consider the service continuity requirement in case of system elements failure. As stated previously, system reconfiguration is a promising and critical criterion and should be considered in home automation systems, to ensure that the provided services are always delivered to the elderly people and guaranty their autonomy.

Existing projects	Fulfilled functional requirements			
	Adaptability	Interoperability	Reconfiguration	Distributed access point
GiraffPlus	√	√	X	√
InCASA	X	√	X	√
eCAALYX	√	√	X	X
COHAND	X	X	√	X
QUATRA	√	X	X	X
DANAH	X	√	√	X
AIDOM	√	X	X	X
SIGAAL	√	X	X	√
ASIM	X	√	X	X
Intelhome	X	X	X	√
XAAL	X	√	X	X
Contribution of this work	X	X	√√√	X

Table 1.2. Synthesis of projects and fulfilled requirements

1.7. Conclusions

In this chapter, introduction of the domain of this research thesis is proposed. State of the art on AAL systems and home automation solutions is presented, followed by identification and analysis of the requirements related to ideal solutions that meet user expectations and needs. This chapter also introduced a number of projects and researches that had been carried out under the AAL Joint Program, as well as in Lab-STICC.

After evaluating different projects and systems, it has been shown that few systems adopted service reconfiguration in case of failure. And in these systems, which realized reconfiguration in their solutions, the approach of redundancy of faulty components is the main solution to ensure system continuity.

Thus, the reconfiguration, as a manner to ensure system continuity, is clearly defined as an essential requirement, to guaranty the services reconfigurability in case of potential failures. The fault tolerance is the main mechanism to ensure the reconfigurability of home automation systems.

As a final point, the work in this research is directed to concentrate on system reconfiguration.

Chapter 2

Characteristics and needs for fault tolerance and reconfiguration modeling

After having reviewed some system approaches as well as existing projects and solutions, and identifying crucial requirements for AAL systems, it can be clearly concluded that most systems are providing almost the same set of services for their users. According to the stated requirements, it is important to outline that these systems meet partially these requirements. Thus, the attempt is to propose a system that meets all requirements so that one can govern that the system meets all users' expectations.

Several researches have carried out to study and to provide a solution for system's interoperability by the use of different solutions tackling, thus, the heterogeneity of different system components. It is worthwhile to notice that other systems (including distributed access points) have undertaken the system's adaptability issue.

On the other hand, very few existing systems take into account the system's availability and the services reconfiguration issues in the context of tolerance to failure.

Therefore, to increase the efficiency of an AAL system and to meet the user's expectations, it is a very critical matter to develop such an ideal Ambient Assisted Living system that delivers the services to the final users with consideration of the requirements already mentioned in chapter 1.

Given the fact that the reconfiguration requirement is an essential prerequisite for successful information pervasive systems (especially those related to ambient assistant systems), the objective of this chapter is focused on services reconfiguration and fault tolerance methodologies in order to propose a system architecture framework responding to the "reconfiguration" requirement.

Section 2.1 is devoted to present the promising points which have motivated this work and the expected contributions of this research work.

The contribution of a proposition starts by a specific definition of the reconfiguration requirement issue and fault tolerance concepts are detailed in section 2.2.

A detailed demonstration of the main characteristic of fault tolerance, in addition to a review of some methodologies related to fault tolerance paradigm are then given in section 2.3. Section 2.4 is devoted to realize a critical analysis of different approaches dealing with the service failure issue. Section 2.5 introduces the system abstract formalization with the proposed fault tolerance approach to tackle the service failure proportion issue. Whereas, section 2.6 is devoted to introduce the fault tree analysis approach and the research activities tackle this approach. Section 2.7 presents a proposition of a dynamic reconfiguration approach as a perspective for a future work. Finally, an illustrative example is presented in section 2.8 in order to demonstrate the concepts detailed in this chapter.

2.1. The expected direction of work

When establishing home automation systems, it is always wise to define the set of principles that makes the system ideally designed and is likely to be used by all users, regardless of their ability, age or culture. This design philosophy shares the evidence that the users should be shared in the design clue issues [Pruski, 2003]. Universal design aims to reverse this trend by offering the system (hardware or software) to be consistent with the physical and intellectual abilities of as many users as possible.

The systematic application of these principles gives a greater opportunity for people with disabilities to access the same functionality as the other users.

From the previous chapter, it was concluded that service's reconfiguration should be considered as a crucial requirement for ambient assisted systems. This specific feature is not widely considered in researches and the related literature. Important research activities focused on modeling the fault tolerance issue for computing systems related to real time solutions such as high-speed transportation or orbit systems, whereas, very few researches addressed the reconfiguration issue in the case of failure for distributed systems related to the medical domain and home automation systems (in which the failure of a system, or part of it, is considered as a critical issue specially for services related to health monitoring and environment controlling) [Rennels, 1987].

One of the main difficulties, often recognized, is that each individual focuses on his own characteristics, and therefore, expresses his own needs. This makes the design and the implementation of the developed solutions for "everyone" more difficult.

From the implementation's point of view of such complex systems (for dependent persons), it is clear that one of the difficulties with academic research activities in this field is to evaluate the efficiency of these research activities. In other words, there is a lack of full-scale facilities, and when they are available, there is no experimental protocol allowing conducting a "general purpose" evaluation.

Ideally, the validation of any AAL system should be conducted by immersing the considered system in a real apartment allowing, thus, the evaluation of the system by dependent persons in a real environment. In addition to the validation process, this process allows system's adjustment leading thus to a possible transfer of technology with real benefit.

Continued progress and cost reduction in information technologies and communication materials have made home automation systems, generally, feasible and cost effective. From a hypothetical point of view, the main challenges of the field of home automation systems (this also concerns the challenges of all researches and projects in this domain) can be summarized in the following [Chan *et al*, 2008]:

- Home automation systems require entire combination of the architecture and the computational infrastructure;
- Elderly people habits and objectives should be ideally considered and respected;
- Further research activities are required to be addicted to both legal and ethical issues, related to the user acceptance, requirements and fulfillment.

In this chapter, the main guidelines leading the direction of the expected contribution in this work are first presented. Main perspectives (that should be faced in the academic and research domains, especially in relation with the field of home automation systems) are also considered. In the remaining, of this chapter, the objective is to develop formalisms allowing the description of how an AAL system should be realized with the consideration of fault tolerance and reconfiguration of its services to ensure that the system provides its services to the users correctly and continuously.

The next section is devoted to introduce the contribution of this research. The idea consists on proposing a framework of a system assumed to meet user's expectations with the consideration of fault tolerance approach and the continuous availability of the delivered services.

2.2. Reconfiguration concept and system elements definitions

In this section, the main goal is to highlight what the reconfiguration or availability is and the related concepts and definitions that clarify the notion of reconfiguration. The issue of how the

reconfiguration concept adapts within the domain of home automation and AAL systems is also addressed.

2.2.1.Fundamental concepts definition

In this section, several fundamental concepts, related to the notion of reconfiguration and fault tolerance, are defined and briefly discussed.

Reliability:

Reliability concepts aim at constructing systems so that no fault may occur. This corresponds to a kind of idealism. In reality, reliability just aims to build systems that deliver services with a justified confidence, i.e. able to avoid failures more frequent or more severe than the presupposed acceptable service.

The reliability concept is, thus, considered as a key element in systems defining and conception. It is even described as critical in some domains such as avionics systems (life critical systems), financial transactions (business critical) or systems in orbit (mission-critical) whose failure can have serious human or financial consequences.

To ensure that the behavior of a system should not deviate from what is expected, it is worthwhile to recall that dependability aspects constitute an important issue of the system's development process.

Availability:

The availability is the intuitive sense of reliability. A system is available if it is able to perform its "intended function" at the moment the function is required. Formally, the availability of a system, as a function of time, is the probability that the system is operational at a given instant of time.

Fault tolerance of a system:

The term "Fault Tolerance" refers to the capacity of a system to keep on functioning and delivering the requested service besides the fact that one or several components are out of functioning.

Fault recovery:

Fault recovery refers to the ability of a system to detect and to "repair" faulty components in order to ensure the continuous system functioning. A system can fail and usually does. Even though, and due to dependability, other systems interacting with it can cause failures by error propagation. In this case, and in order to ensure the service continuity (while maintaining an acceptable level of confidence in the functioning of the system), fault recovery techniques must be used to cover the induced functioning errors. Thus, this will allow the system to continue to operate. A system having the capacity to continue operating, fully or in a degraded regime, in case of faulty components occurrence, is described as being fault tolerant. The faults origins can be diverse, including human errors caused by misuse of the system, design errors and errors due to the environment, e.g. sensor failures inducing errors in the readings.

Moreover, reconfiguration concept is related to the issue of how a system can cope with failures of services. By failures we mean any malfunction that prevents the execution (by the user) of a requested service to be complete successfully, i.e. with a return to success. Therefore, the goal is to initiate corrective actions in response to failures.

These failures can occur either at the level of operations activation or at the navigation level. In this sense, we are interested in the failures recovery issue. In other words, the major interest is focused

on determining failures correction procedures rather than on avoiding the failures occurrence (which is the issue concerned by the reliability concept).

A failure can be defined as an event that occurs when the delivered service deviates from performing the function it is supposed to conduct, or that, the service is no longer conform to the predefined specifications. This is also the situation when the service specifications do not meet adequately the system's function.

The mechanisms put in place to achieve a certain fault tolerance are also diverse. For example, in computer systems, mechanisms of data replication (backup, archive) are used to overcome the loss of storage servers. Distribution mechanisms are also used to limit data to a local area affected by the unavailability of a server. Load balancing mechanisms are also applied in order to cope with too much processing load, as is the case in high-traffic web servers.

To ensure correct suitability to the current context, fault recovery changes can be applied to the software, the hardware, or to both, and are triggered due to either objective changes (adaptively), or to the service failures (fault-tolerance).

According to a predefined reconfiguration mechanism and in order to overcome potential failures (ensuring, thus the continuous delivery of a service), different scenarios are generally defined during the system's design time.

Several types of reconfiguration mechanisms can be distinguished:

- ***Static (off-line) reconfiguration mechanism***:
  Static reconfiguration mechanism performs failure compensation on an operation basis, using statically predefined reconfiguration rules "stored" in the resource description;
- ***Dynamic (on-line) reconfiguration:***
  Dynamic reconfiguration mechanism computes, in a dynamic manner, a new reconfiguration rule using recent system status, in the absence of off-line reconfiguration of the failure;
- ***External reconfiguration:***
  This reconfiguration mechanism occurs when an intervention from a third party (i.e. an expert) is conducted in order to recover the failure.

For example, in DANAH's software architecture [Lankri & Philippe, 2009], to ensure service deliverance, a static reconfiguration depending on predefined alternative scenarios is adopted. Moreover, DANAH offers the possibility of multi-level reconfiguration from the light up to heavier reconfigurations. In fact, when an operation fails, the system searches, at first, whether an alternative has not been defined by the expert; this is what is called *reconfiguration by rules*. DANAH allows a higher reconfiguration level where the system itself calculates an alternative by compensation effects. Finally, when the alternatives of these two levels fail, the transaction may be issued and the result is definitely a failure. Notice that reconfiguration by means of rules is determined by the domain expert in order to fulfill the user's needs and the system may propose, by itself, an alternative service if there is no adequate rule.

2.2.2. Conception of system elements related with reconfiguration concepts

According to the definition of the reconfiguration concept, and based on the objective of developing a framework, allowing services reconfiguration, this section introduces the main system elements definitions to be realized in the high-level abstract representation of system reconfiguration

elements. To conduct this objective, we start by an example introducing the main notions that could be presented in a home automation system, and from the example, we can derive the main conceptions of the reconfigurable system elements.

In the example, we consider that a disabled elderly person living in an apartment which consists of a bedroom, a living room, a kitchen and a bathroom.

The apartment is equipped with a set of sensors and actuators that are distributed around in the rooms in order to sense both the environment and the resident, in addition to monitor, analyze and make interventions according to the data delivered by the sensors.

The proposed architecture here is the same architecture as the one in the system deployed in Kerpape apartment and consists of the following elements:

- Intelligent wheelchair equipped with a PDA (to control and connect with other smart devices in the apartment);
- Lights of two types: ceiling lights and wall lights that can have controls ON / OFF;
- Electric shutters with up / down controls;
- Doors equipped with electric door openers;
- Electrical Bed;
- Rails ceiling to move the bed to the bathroom and vice versa (with the Up / Down controls and forward / backward);
- Automated windows with the Open / Close commands;
- Controlled plugs;
- Calling facility (connected to a call center);
- Multimedia Facilities;
- Work table and removable electrical cooking elements controllable: Up/Down;
- Entry audio-video returning information on the TV screen;
- Contactors to indicate the doors open state;
- Detector in case of non-supply or fire alarm;
- Temperature Sensors;
- Brightness sensors;
- Presence detectors associated with the light sensors.

Several reconfiguration plans scenarios could be proposed for each element of the list above (a reconfiguration plan in the case of failure in the execution of the service related to each sensor/actuator).

Suppose that the resident is willing to go to his/her bedroom from the living room at night by using the wheelchair:

- The controller of the bedroom door should open the door to allow for the resident to move inside the room. In case of door opening failure, the system makes an emergency call to a caregiver, or to a relative, to assist the resident;
- When the door is opened, then automatically the ceiling light in the bedroom is switched on (or the wall lights in the room will be switched on as alternatives of the ceiling light);
- When the presence detector detects that there is nobody in the living room, the lights is switched off. If the lights in the bedroom are failed to switch on, thus the living room lights still on until they are switched off by the user;
- When the presence detector perceives that the resident is in the bedroom and that he/she is in the bed (by the bed sensor detector), then, the system switches off the light(s) and closes the bedroom door. In the case of failure in execution of the last commands, the system sends a command to the relative to assist in the termination of the commands.

The use-case diagram of the proposed scenario is illustrated in figure 2.1 where ellipses represent the service and the main components participating in the execution of the particular service as well as

the users participating in performing this service. In the use case-diagram, three types of arrow can be illustrated:

- An arrow connecting users to the service and functions;
- The arrow titled {include} (representing that the parent ellipse "move to bedroom" should comprise the child ellipse, e.g. switch on the light;
- The arrow titled {extend} represents that the parent ellipse, or function, comprises but not obligatory the child function (the function switch-on the light may comprise switching on the wall light).

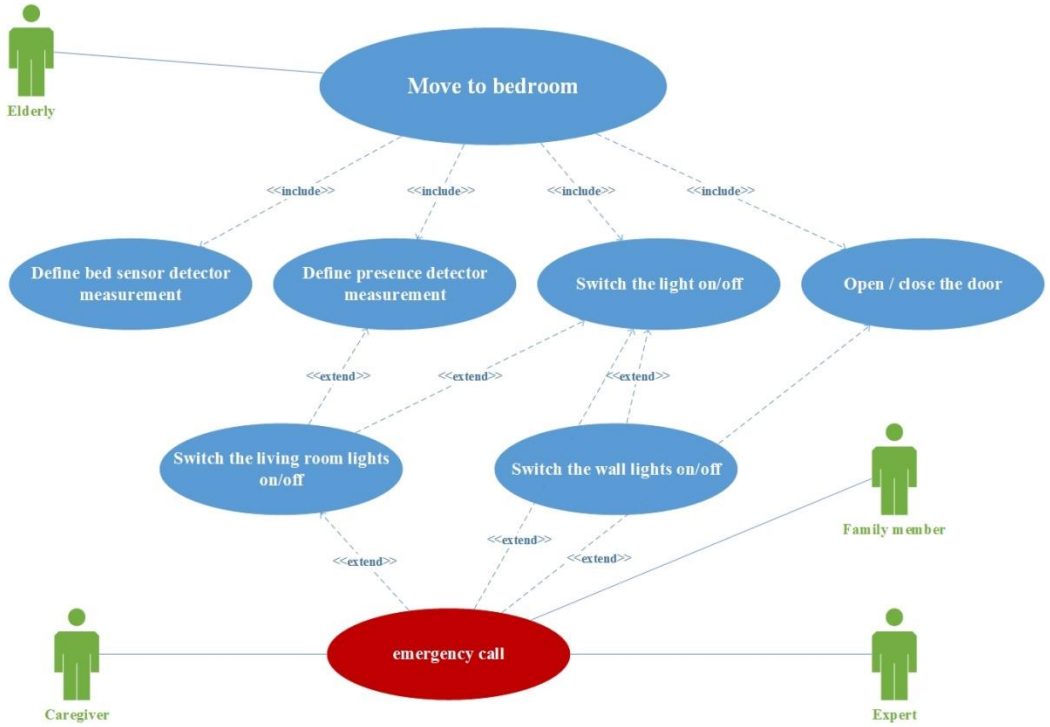


Figure 2.1. Use case diagram of the proposed scenario

After displaying an example of reconfigurable home automation system, we introduce the main system elements definitions to be realized in the high-level abstract representation of system reconfiguration elements. These definitions are considering the following issues:

1. As a starting point, to study and to develop the proposed abstract model of the system elements functionalities, the consideration should focus on the proposition of static reconfiguration. The dynamic services reconfiguration should be regarded in the expected system workflow as a perspective for the work;

2. The Ambient Assisted Living systems architectures are considered as distributed systems, as well as various assistive domains such as healthcare facilities [Truong *et al*, 2009a], aware-homes [Truong *et al*, 2009b] and assisted-living for the elderly [Deursen *et al*, 2000];

3. The Ambient Assisted Living systems are composed of a set of key elements making the system architecture responsible of performing all the system functionalities. These elements are as follows:

- Hardware resources: pervasive systems (including AAL systems) consist of different types of hardware devices such as network of sensors, set-top-box, PDAs, appliances, interactive TVs, actuators, smart wheelchair, etc.
- Software resources: AAL systems, as pervasive systems, include off-the-shelf software operating on the devices to perform the proposed services. Such applications include operating systems, web based applications, video and audio applications, etc…

- The hardware resources are connected to each other, as well as connected to the software resources via wired and wireless networks. This allows to exchange information between these resources and to perform the proposed related services.
- These resources, called components, are performing certain types of actions in order to provide certain services to the desired users. These actions are related to the operations activated by these components, such as *SWITCH ON a Light and OPEN CHANNEL1 in the TV to provide a service related to watch the TV at night*.
- Services: each AAL system performs a various number of services and facilities, which are provided to elderly people to assist them and guaranty their self-independence.

According to these components and the actions performed by each of them, several reconfiguration scenarios can be proposed. These scenarios are based on the services related to each bilateral (component – action).

4. Faults, in a distributed system, can be classified into: device, application and network failures [Goel and Okumoto, 1979]. Device failures are related to faults occurring at the hardware devices levels (such as sensors and actuators…). Notice that these faults may potentially contribute to the failure of the distributed system.

5. Software failures include software crashes due to bugs, operating system errors, unhandled exceptions and faulty usages. Network failures are related to the errors caused by low signal strength, devices going out of range and unavailability of communication channels due to a heavy traffic.

6. A proposed reconfiguration scenario is conceived so that it concerns a specific component failure type and specific related action(s) of this component. Such a proposed scenario aims at provide potential alternative(s) in order to tolerate the component failure ensuring, thus, the deliverance of the service performed by the considered component.

7. Each planned scenario reflects an action composed of the use-cases and actors leading the actions to be fired.

8. A failure is tolerated according to a mechanism which is assumed to deliver a feedback information/signal when an error is detected. In other words, the fault tolerance process should be preceded by an Error detection mechanism.

After having introduced the concept of assistive systems for elderly people, and after the definition of the service reconfiguration, availability and related concepts, the question we may raise is the following:

What are the specific characteristics that must have a home automation system in order to be able to tolerate potential faults?

In other words, how should we define a home automation system framework to be able to provide the available services to its users, and to meet user's expectations and needs, with the consideration of the reconfiguration and fault tolerance requirement?

2.3. Fault tolerance

2.3.1.Major characteristic of fault tolerance for AAL systems

Since AAL systems exist in the user's environment, the technology is maintainable if it is invisible to the user and does not interrupt the user's awareness. This requires that the functioning of the devices in the environment needs to be unconscious to the user. Therefore, the system has to be resilient to various kinds of faults and should be able to function despite possible occurring faults.

As previously mentioned, Ambient Assisted living systems architectures can be described as a distributed system in nature [Lézoray *et al*, 2010], because they constitute a distributed, heterogeneous and highly dynamic infrastructure and technologies.

To answer the question raised in the previous section, we suggest analyzing the main fault tolerance features and characteristics for distributed systems. This is justified by the similarity of those systems architectures with home automation system architecture (Ambient Assisted Living systems).

To characterize the main fault tolerance features in distributed systems, including AAL systems, it is wise to differentiate, first, between the terminologies related to fault detection, recovery and tolerance. This allows understanding how the system can tolerate faults and reconfigure the services in order to insure the correct deliverance of different services.

The association between the three basic concepts: component fault, component error and service failure is illustrated in figure 2.2.

In this figure:

- A ***Component fault*** refers to the cause of an error. When a component fault occurs, it leads to the component error;
- A ***Component error*** is the resulting consequence of a component fault where the component becomes out of functioning. Notice that a component error may, or not, lead to a service failure;
- A ***Service failure*** occurs when one or several component errors cause the delivered service to deviate from normal service deliverance and functioning.

Figure 2.2. Fault tolerance terminology

Fault tolerance in today's complex systems is considered as a key point for users' satisfaction. Following the literature, (with regards to fault tolerance requested features, regardless the system nature), major characteristics of fault tolerance generally include:

- No single point of failure: if a system experiences a failure, it must continue to operate without disruption during the repair process [Nelson, 1990];
- Avoid downtime: It is crucial to ensure correct operation even in the presence of faults;
- Allow the system to keep executing even with the presence of faults. These systems are usually classified as either highly reliable or highly available [Avizienis *et al*, 2004];

Faults origins can be diverse including human errors (caused by a misuse of the system), design errors and errors due to the environment, e.g. sensor failures inducing errors in the readings. Moreover,

achieving high levels of reliability and availability, in spite of service or infrastructure failures, poses distinctive set of challenges [Wang *et al,* 1999]. This raises the important question of how the system can cope with a service failure and initiate corrective actions in response to occurring failures.

Given that failures can occur either at the level of activations of hardware components or at the level of the software running these components, it is worthwhile to notice that we are interested in service fault tolerance in the sense that we do not seek avoiding failures but we should try "tolerating" these faults and retain the service delivery (i.e. our target is to seek system reliability). Gokhale [Gokhale *et al,* 1998] has shown that the critical requirement of fault tolerance mechanisms, in any system, arises from two points:

- The cost of uncorrected errors is especially severe in large time-shared computer service systems and in situations in which a computer controls a very valuable system, and is not accessible to human repair. Examples are a real-time control computer and Telemedecine systems;
- A second critical requirement for fault-tolerance exists when human lives may be affected by computer errors. e.g., in military defense systems, in control of high-speed transportation systems or of medical systems.

2.3.2.Fault Tolerance methodologies

Two major fault tolerance methodologies are generally considered for fault recovery: redundancy based methods and replication based methods.

In a replication based method, several identical components operate concurrently and a voting system is then used to select the adequate outcome (i.e. adequate component).

Whereas, in a redundancy based method, only one component is functioning at time while the remaining redundant components are standing by, awaiting the case where the active component ceases functioning.

Nevertheless, these methodologies need to be modified to fit the needs of systems related to Ambient Assistance [Chakraborty *et al,* 2002]. Achieving expected levels of reliability, especially when coupled with the ad hoc growth of devices that may be expected in smart homes, constitutes a great challenge [Lankri *et al,* 2009].

Existing fault tolerance methodologies are diverse. Based on an analysis of some existing works, here follows the main concepts to be highlighted:

- In computer systems, storage servers are crucial hardware system components. To overcome the loss of a storage server, a data replication mechanism (backup, archive, distributed servers) is applied. When a fault occurs, the needed data will be available on another storage server (i.e. alternative component). Another situation is also encountered when dealing with high traffic web servers. In this case, a load balancing mechanism (a replication approach) is applied in order to cope with such a huge processing load [Cheung, 1980]. The main drawback of this approach is the redundancy of the data in several locations, which needs to be synchronized simultaneously when the data is updated in one location. Stored data must be up-to-date in all servers, thus, the data must apply backup regularly.
- Most of the existing systems involved in services reconfiguration propose the online replacement of the faulty component with other equivalent components. The work described in [Gokhale *et al,* 1998] focuses on the ontological aspects of service, fault and recovery strategies. The ontology "service" describes all system services themselves (features and prerequisites). The ontology "fault" describes resources conditions that can lead to errors. The third ontology "recovery" outlines possible strategies to overcome different fault situations. When an exception occurs (i.e. a service failure) the system reports the current

service context and the service manager status. Then, the system attempts to match the exception into an instance of the ontology of faults. Once all possible errors identified, the research manager in the ontology (covering possible solutions to the problem) takes into account the prerequisites and features offered by the faulty service described by the ontology of services in order to substitute the failed service by an alternative similar service [Cheung, 1980]. This method can be similarly applied for AAL systems, taking into account all potential faults resources for any service provided by the system. Nevertheless, this method doesn't consider dynamic mechanisms to execute alternative service. The reason is that the recovery strategies are already defined as ontology and the system is assumed to select an alternative service from a predefined set of alternative services only. Some studies describe similar mechanisms for Web services [Gokhale *et al*, 1998] based on ontologies. But, this time instead of benefiting an ontology recovery considering faults, finding similar services takes place in the "catalog of services" by selecting the closest service to the faulty one.

- Previous methodologies assume that the component replacement (for the failed service) is available and that it is sufficient to identify and to replace the failed component. However, sometimes it is hard to find the adequate alternative component that offers a similar functionality with a high satisfactory degree. In this particular context, the work described in [Lee *et al*, 2008] suggests the application of a dynamically assembling services approach in order to meet a target goal. Two entities are considered: services (that provide the basic functionalities of the system), and, tasks (that describe the services necessary for its fulfillment). Taking into account the current context (i.e. the pre-requisites of the selected services are assumed satisfied by the context), the developed approach aims at dynamically selecting necessary services to accomplish a considered task.
- In [Lankri *et al*, 2009], the authors discussed dependability requirements of pervasive computing in a healthcare environment (based on the fact that pervasive computing technology is likely to create concerns about the security of healthcare systems, due to increased data aggregation, ubiquitous access, and increasing dependency on technical solutions). The authors have also discussed the challenge of securing medical devices by overriding actions for emergencies as well as protecting confidential patient's data from unauthorized access and modification. They have proposed a highly distributed architecture, in which access points to the servers are still available and can operate separately. In the case where the number of servers is unavailable, access points must provide manual override mechanisms (in case of emergencies). These override mechanisms contain devices that allow logging the required data for later audit or raising an alarm. An advantage of this architecture is that it is simple and fits the connectivity available to pervasive devices.
- In [Sen, 2010], the author discussed several mechanisms that can be employed in distributed systems for tolerating faults, including assisted living systems. These mechanisms include Alternate Application / Device Usage by the proposition of alternate module to surrogate the failed device; Alternate Notification Mechanisms by redirecting the message through a different channel of communication if the system discovers that a notification device has failed. As an example of the latter method is the notification media about healthcare personnel in an emergency case.
- In [Jeffers, 1988], the authors have introduced errors handling in sensing and inferring context based on the use of redundancy, either by using several sensors (acquiring the same information to overcome errors of the sensor's failure), or by using multiple algorithms to deduce the same higher-level context from sensed contexts (which helps overcoming wrong inferences made by one or more such algorithms).

- In [Kim *et al*, 2006], the authors also had discussed the redundancy based methodology. The proposed methodology may be applied at several levels:
    - Information redundancy seeking to provide fault tolerance through data replicating or data coding. For example, a Hamming code can provide extra bits into data allowing thus to recover a certain ratio of failed bits. Sample uses of information redundancy are parity memory, ECC (Error Correcting Codes) memory, and ECC codes on data blocks.
    - Time redundancy achieves fault tolerance by performing the same operation several times. Timeouts and retransmissions in reliable point-to-point and group communications are examples of time redundancy. This form of redundancy is useful in the presence of transient or intermittent faults. It is of no use with permanent faults. An example is the TCP/IP's retransmission of packets.
    - Physical redundancy (concerning devices, not data). Extra equipment is added to enable the system to tolerate the loss of some failed components. RAID disks and backup name servers are examples of physical redundancy.

Table 2.1 makes a synthesis of the presented fault tolerance methodologies conducted in major existing research activities.

	Fault tolerance methodologies	Drawback
Computer systems	• Data replication mechanism, to overcome the loss of storage servers. • Data replication to overcome the huge processing load in case of high traffic webservers.	• The huge redundancy of data in several locations. • Need to simultaneously synchronize these locations to make the data up-to-date. • A cost effective solution
Web services based on ontological aspects of "service"	• Define ontologies to describe the concepts: service, fault and recovery. • In case of service failure, the system replaces the whole service, based on predefined fault status, with an entire service.	• No consideration of dynamic mechanisms to execute the system (use static reconfiguration of the services), in case there is no alternative offered service.
[Lankri *et al*, 2009] research work	• Dependability needs to pervasive computing. • In case of the unavailability of some data servers, access points provide manual override mechanisms to overcome the problem of patient's data loss or unauthorized access and modification. • Provide devices to access to the data for later check.	• Cost effective solution, even it is simple.
[Jeffers, 1988] [Sen, 2010]	• Replication mechanisms of applications / devices / sensors	• The cost of replication of the same application / device / sensor, in case the system uses several resources.

[Kim *et al*, 2006]	• Redundancy based methodology of:
	  ○ Information (data replication);
	  ○ Time (perform the service several times);
	  ○ Physical redundancy of devices.
		• The cost of the redundancy of resources.
		• Some critical services (healthcare services in case of emergencies) could not be replicated several times.
		• No clear strategy of the mechanism of selecting the redundant resource(s).

Table 2.1. Summary of fault tolerance methodologies

2.4. Service failure approaches

After having reviewed major fault tolerance methodologies, this section details different approaches used to evaluate the failure occurrence degree.

Based on the uncertainty evaluation of fault occurrence (at different levels: the components, services and the system levels), two major probabilistic approaches are proposed when dealing with fault tolerance analysis: The Bayesian networks and the Markov chains approaches, and a logical based approach.

2.4.1.Bayesian Networks based fault tolerance approach.

A Bayesian network [Ben-Gal, 2007] is a probabilistic directed acyclic graphical model based on the use of a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network is used to compute the probabilities of the presence of various diseases. The application of Bayesian networks into fault tolerance context is based on the representation of the states of different system components considered as random variables, and, on computing conditional dependencies (i.e. conditional transition probability distributions) between these different random variables. The main idea is to evaluate the service failure and, thus, system functioning probability as a consequence of components failure probabilities.

Since Bayesian Networks use a probabilistic model to represent the knowledge of a system functioning states, one can formulate the following crucial questions (related to AAL fault tolerance mechanism, described in previous section):

- What is the probability of a component to fail?
- Assumed the set of component states for a given service, what is the most probable faulty component [Landstrom, 2010], which may lead to a service failure?

All these questions raise the key importance evaluation of the probability of component and service to occur. In fact, it is difficult to estimate the probability of a component to fail (or service to fail) since it is impossible to predict and classify all the defects of a component [Myllymäki, 2001] (due to the reason that the probability of a component to fail is unpredicted). For instance, a component may function correctly in all the experiments performed during the test of the system before putting the system in real investment. Even though, some technological vendors can estimate the Mean Time Between Failures (MTBF) of a component, but when it comes to the real environment of an application, this can be different from the theoretical estimation, due to different issues.

Major advantages of Bayesian networks concern the fact that it is intuitively easy for a human to understand [Richards *et al*, 2013], as well as the consistency, theoretically solidity mechanism of Bayesian networks for processing uncertain information [Aghie and Saeedi, 2009]. As probabilistic approaches provide a consistent calculus for uncertain inference, and given the input, all the alternative mechanisms for computing the output with the help of a Bayesian network model will produce exactly the same answer. Of course, this is a big advantage of Bayesian networks, but this is applied when we can predict or estimate precisely the probability of a component to fail. This shows the major constraints of Bayesian Networks: their inability to represent dynamic facts about the system [c. Kyprianidou, 2002]. Another constraint of Bayesian Networks is related to the quality and extends of the *prior* beliefs used in the Bayesian inference processing. In fact, a Bayesian Network is only as useful as the prior knowledge is reliable (since the quality of these prior beliefs will change to the entire network and invalidate the results). This is also applicable in knowledge related to fault tolerance of

system components, because we do not have reliable prior knowledge about the components functioning states. Another significant disadvantage of an approach involving Bayesian Networks is the fact that there is no universally accepted method for constructing a network from data.

2.4.2. Markov chains (or discrete-time Markov chains, DTMC) based fault tolerance approaches.

[Pfeifer & Carraway, 2000] constitutes another interesting approach that is widely used for computational systems knowledge representation and reasoning with the consideration of fault tolerance and reconfiguration in these systems [Norris, 1998].

First order Markov chains are mathematical probabilistic models for the future state estimation (of a component or a service) depending, only, on the present state of this component (or this service). Therefore, the basic knowledge representation tool within DTMC is the state transition probability measure (representing the probability of transition between two states within a given time interval). The states here could represent the states corresponding to each component of an AAL system, when the service is represented by the process, which moves successively from one state to another reflecting that a component used by this service, or reflecting the movement independently to a failure state representing the occurrence of a fault. The movement of the process from a state to another is fired using the transition probability value, i.e. what is the probability of the component to move from a correct state to the faulty state?, or, what is the probability of a component to participate in a particular service? The previous two questions and other similar questions related to the formalization of a system using Markov Chains raise the key issue of probability values estimation, which is a difficult task in the case of variant hardware and software components of an AAL system, as previously mentioned. Nevertheless, Markov chains offer significant advantages over other reliability modeling approaches.

Some of these advantages are:

- Simplistic modeling approach: The models are simple to be generated and flexible;
- Redundancy management techniques: System's reconfiguration required by failures is easily incorporated in the model;
- Coverage: Covered and uncovered failures of components are mutually exclusive events. These are not easily modeled using classical techniques, but are readily handled by the Markov approaches;
- Complex systems: Many simplifying techniques exist which allow the modeling of complex systems.

According to [Pfeifer & Carraway, 2000], the advantage of the Markov process is that it tidily describes the failure of an item. It develops the probability for an item being in a given state, as a function of the sequence of states through which the item has traveled. The Markov process can thus easily describe degraded states of operation, where an item has either "partially" failed or is in a degraded state (where some functions are performed while others are not). Nevertheless, the key disadvantage of Markov Chains is the explosion of the number of states as the size of the system increases. The diagrams for large systems are generally extremely large and complicated, difficult to construct and computationally excessive.

Other drawbacks of Markov chains are:

- In some areas, the data available will be insufficient to estimate reliable probability or transfer rates, values especially for rare transitions;

- Like all other sequential models, the validation of Markov models depends on predictions of the system behavior over time and is, therefore, difficult, and may even be impossible, for really long periods of time.

Besides the two previous probabilistic approaches (based on uncertainties computations of fault occurrence), an interesting logical based fault tolerance approach has been proposed in several applications

2.4.3.Boolean logic based fault tolerance computation approach:

A component, a service or a system are considered as having a binary functioning state: Fault and Success. A given service S_{k0} needs several components $\{C_{k0,1}, \ldots, C_{k0,K0}\}$, called main service components (allowing the service to be delivered successfully). Therefore, the K0 components should all assume the Success state of functioning in order for the service S_{k0} to be in the state of Success. Figure 2.5 illustrates the concept of Boolean logic.

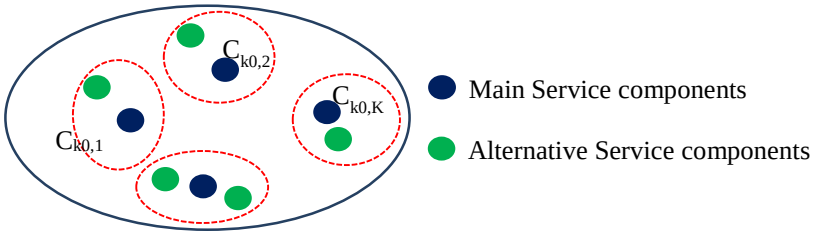


Figure 2.3. Boolean fault tolerance approach

In the case where no fault tolerance approach is conceived, the success state of the service functioning can be simply computed using the tautology of an AND Boolean function:

$$State(S_{k0}) = \bigcap_{l=1,\cdots,K0} State(C_{k0,l})$$

When component alternatives (statically or dynamically) are expected to be used (in a replication or a redundancy approach), the state of each component is computed using the tautology of an OR Boolean function:

$State(C_{k0,l}) = \bigcup$ State(alternative Components)

Therefore, the Boolean logic fault tolerance approach is based on combining components binary logical state values in order to compute a binary logical state value for the considered service.

An important advantage of this approach is that it does not use any *prior* or transition probability values of being in the success state of functioning for different components. Nevertheless, a major handicap of this approach is that when a component fails, and an alternative component is used, then the service state of functioning remains unchanged, i.e. no indicator is delivered in order to raise the fact that some components may have failed.

2.5. System Abstract formalization

In the previous section, we have defined the home automation system in terms of distributed systems, as well as we have introduced the existing fault tolerance methodologies and have detailed different approaches used to evaluate the failure occurrence degree. In this section, a formalization framework allowing representing service's reconfiguration elements is proposed. The objective is to facilitate the understanding of the relationship between these elements and to formalize how fault tolerance mechanisms can be achieved.

In fact, an Ambient Assisted Living system, Q, performs a set of predefined services in order to ensure the well-being of desired users of the system.

The system Q is composed of three major constituents:

$$Q=\{S, C, X\} \quad (1)$$

Where:

- S: is a set of K basic services:
  $$S = \{S_1, S_2 \dots S_K\} \quad (2)$$
  that the system aims to provide to the end user;

- C: is a set of N components,
  $$C = \{C_1, C_2 \dots C_n\} \quad (3)$$
  or devices, that the system uses in order to deliver different services; and

- $X = \cup_n X_n$: is the set of states that different components may assume.

Each component C_n assumes, when a particular service is executed, one of the states from the C_m-states set:

$$X_n = \{X_1^n \dots X_{sn}^n\} \quad (4)$$

(X_n denotes the number of states related to the component C_n).

It is important to notice that each service S_k, k=1, …, K, activates a subset of components from the components set C, and that, each component C_n, may be activated by several services as a "primary" component (i.e. fundamental for the execution of the service) or, as an alternative component for some other services for which a primary component fails.

In other words, the relationship between a given component C_n and a given service S_k can be "quantized" in terms of "*importance* or *criticality* into three levels: P, A and NC:

- P: The component is of primary importance for the service (i.e. if the component fails, then, we need to perform a dynamic reconfiguration to overcome its failure);

- A: The component is of an auxiliary need for the service and is considered as for alternative use if a corresponding component from another service fails;

- NC: The component is Not Critical for the considered service.

The global relationship between the Service and the Component sets can thus be resumed as a matrix, Figure 2.3.

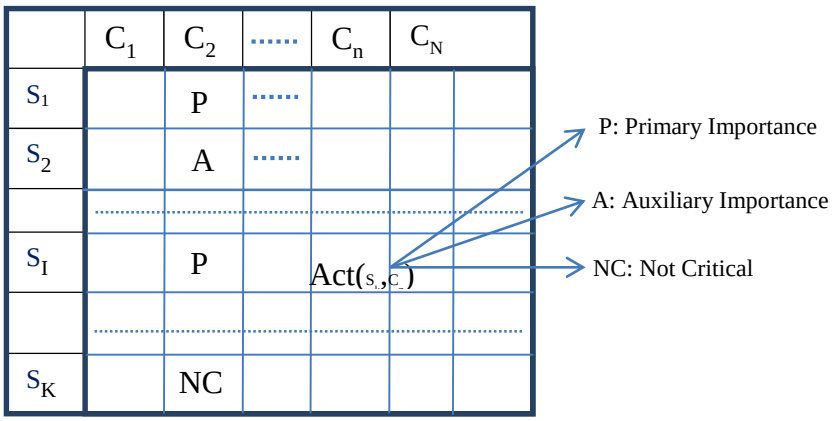


Figure 2.4. Importance relationship matrix between Service and Component sets

For instance, the component C_2 is of primary importance for the two services S_1 and S_I. This is the case of a light that should be used for two different tasks. Also, this component has an auxiliary importance for the service S_2 (if the light used for the service S_2 is down, then the light generated by the component C_2 may be a good alternative. Finally, the component C_2 is strictly not critical for the service S_K.

The state of the component C_n at a given time "t" is denoted $X_n(t)$.

Assuming that at time t_0, the component C_n assumes the state $X_n(t_0)$, then, the component state transition (due to the activation of a given service S_K) is governed by the following rules:

- If $Act(S_k, C_n) \equiv NC$; Then: $X_n(t_1) = X_n(t_0)$ (i.e. the state of the component remains unchanged;
- If $Act(S_k, C_n) \equiv P$; Then: $X_n(t_1) = X_n$, $S_K \in \{X_1^n \dots X_{sn}^n\}$ (i.e. the state of the component corresponding to the requested service is activated) ;
- If $Act(S_k, C_n) \equiv A$; Then:
  - If [no other service requests the use of C_n] ; Then $X_n(t_1) = X_n(t_0)$ (i.e. the state of the component remains unchanged);
  - If [An auxiliary service $S_{K'}$ requests the use of C_n]; Then $X_n(t_1) = X_n, S_{K'} \in \{X_1^n \dots X_{sn}^n\}$ (i.e. the state of the component corresponding to the auxiliary requested service is activated);
    This case refers to the situation where the component C_n is used as an alternative component for the service S_K (in an alternative scenario); whereas, this very same component C_n may be a primary component for another service $S_{k'}$ at the same time. This constitutes, a first indication related to the component C_n importance in the global system's framework (a component used by all the services of the system should have a higher importance degree than a component shared by a few number of services). This issue will be detailed in chapter 4.

$(t_1 - t_0)$ is the time needed for a component to move from an initial state at t_0 to a new state at t_1, (seconds, minutes, etc.).

Therefore, and in order to study the question of a service failure, we suggest, first, to model the component failure issue in terms of a component importance factor for a given service S_k and a given component C_n. This factor, denoted $I(S_k, C_n)$, assumes values from the set $\{a_0, a_1, a_2\}$ where:

- a_0: means that the component C_n is not critical to the service S_k;
- a_1: means that the component C_n is important to the service S_k, but there is an alternative component that can perform the same part of the service (i.e. static reconfiguration);
- a_2: means that the component C_n is very important for the service S_k and we need to perform dynamic reconfiguration to overcome the failure (by supposing another service to be executed in order to perform the essential service delivery).

At this modeling level, we can raise the following question:

"*How can we gather the knowledge related to the component importance factor with the state of a component (i.e. component failure or, component functioning with success) and to propagate this knowledge in order to analyze the service failure issue (based on components failure)*?"

To clarify this question, let us analyze the following figure where we have reported on the X- axis the three states of the component importance factor, and on the Y-Axis the two component functioning states (i.e. Failure and Success):

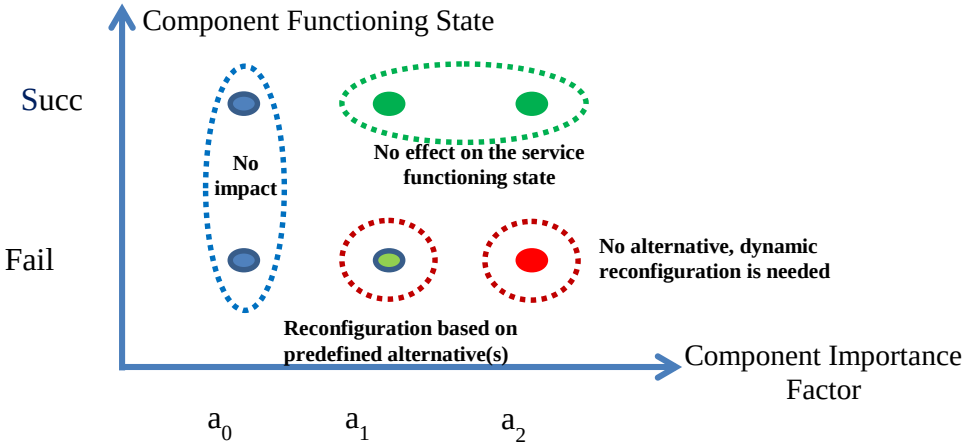


Figure 2.5. Components operational contexts

Using this figure, the case where the component C_n importance factor $I(S_k,C_n)$ (for a given service S_k) assumes the state a_0, then, whatever the component functioning state is, this will not affect the service S_k Failure/Success state.

When $I(S_k,C_n) = a_1$ or $I(S_k,C_n) = a_2$ and the component functioning state is (Success), then this will have no effect on the service global functioning state.

When $I(S_k,C_n) = a_1$ and the component functioning state is (Fail), then the system will perform a substitution of the failed component based on predefined alternative(s).

The most critical situation is clearly when $I(S_k,C_n) = a_2$ and the component functioning state is Failure, then, the service will certainly fail and a dynamic reconfiguration mechanism should be investigated to overcome this situation of failure.

This approach intends to formalize a model allowing considering different situations encountered in this component importance factor / Functioning state figure.

Moreover, another concern can be faced in this model. This concern is related to the interesting time (of the service), which may be varying the importance factor issue.

In fact, the proposed component importance factor / Functioning state figure is not static and may evolve through time (for instance, the light component does not have the same importance factor at day time and at night time). In this case, the importance factor should be simply considered as a time depending function.

2.5.1.Importance concept and system abstraction

At this level, an important question can be raised concerning the use of the components importance factors in fault tolerance mechanisms. In fact, as already mentioned, the main fault tolerance mechanism used major distributed systems is the redundancy mechanism where the expert is in charge of giving a prior set of alternatives to each component in order to tolerate occurring faults.

The proposed importance factor (using three levels of importance associated to each component) can be exploited in order to define a kind of priority for the selection of alternatives in the case of failure.

It is worthwhile to notice that the redundancy mechanism does not take into consideration the probability of failure of each component (or any other quantitative measure of the components performances).

To tackle this issue, the most widely used approach to investigate the performances of a system is based on the use of Fault Tree Analysis [Ramamoorthy *et al*, 1977], FTA. In FTA, the system is modeled as a hierarchical tree formed by the constituting elements (i.e. components). Given the basic probability of failure of different components, and using the probabilistic computational rules, the global performance of the system is evaluated as being the probability of success of this system. When a component fails, the FTA is applied, as a fault tolerance mechanism, as follows:

"the global performance of the system is computed taking into account each existing alternative to the faulty one. The final alternative to retain will be the one leading to the highest performance of the system (i.e. the highest success probability)".

In chapter 4, a definition of a continuous valued importance factor is detailed (i.e. the importance factor will not be restricted to the three quantized values, here introduced, but, may assume any value from an importance values interval). This "opening" from the three quantized values into a continuous value will allow to the designer, the use of an important degree of flexibility. Moreover, as the proposed importance factor will be a continuous value, chapter 4 proposes a global fault tolerance approach integrating the two existing features for each component: its failure probability as well as its importance factor. Moreover, the linkage between the abstract representation (previously introduced) and the FTA (as a fault tolerance mechanism) is realized by developing a design workflow to be detailed in the next chapter).

2.5.2.Proposed Fault Tolerance approach

As we have shown in the existing approaches (dealing with fault tolerance issue), the use of state *prior* and state transition probability values (in order to compute the global success/failure of a service conditionally to the constituting components states) suffers from precise statistical values estimation mainly in dynamic environments (besides the extreme difficulties of having representative amount of statistical data allowing to make high quality estimation of these probability values).

Another important issue concerns the inherent meaning of a service success or failure probability. In fact, having an extremely small probability value of a service failure does not prevent the service to fail (i.e. will not guarantee success).

The probabilistic approach is difficult to be applied when talking about dynamic representation of a component functioning states based on the component importance factor.

moreover, how could the importance factor of a component influence the probability of a component to fail?

On the other hand, the Boolean fault tolerance computation approach is so "rigid" that no indication is delivered when some components fail. It just targets to produce a binary state value for the functioning of the considered service.

In this work, the proposed model is positioned in between the two approaches: the probabilistic-based and logic-based approaches.

In fact, the proposed approach is inspired from the application of possibility theory in order to model different operational contexts at the components level and at the information combination process (in order to propagate the components functioning states into the service functioning state).

Possibility theory has been developed for modeling uncertainties (i.e. the same application context of probabilistic approaches), for which only subjective, or ambiguous, knowledge is available. This frequently occurs in the framework of approximate reasoning, where, for instance, the only available knowledge is given through linguistic descriptors. The word "possibility" can be interpreted in several ways; physical, epistemic and logical. In the physical sense, "possible" means easy to achieve or feasible. For instance, in the sentence "it is possible for the vehicle to hold 6 passengers", possibility means the physical capacity of the vehicle. In the epistemic point of view, "possible" means plausible. The logical view of possibility provides a means to deal with incompleteness. With given incomplete information about an event, the logical interpretation of possibility gives a degree of confidence in the occurrence of the event as in "it is possible that it will rain tomorrow".

The basic building blocks of possibility theory can be described as follows:

Let Ω denotes the frame of discernment (i.e. set of states of an event) where one and only one element may occur, but we don't know which element (i.e. this is the so called uncertain situation).

A possibility distribution is a mapping π from S to a totally ordered scale L, with top 1 and bottom 0, such as the unit interval. The function π represents the state of knowledge of an agent (about the actual state of affairs) distinguishing what is plausible from what is less plausible, what is the normal course of things from what is not, what is surprising from what is expected. It represents a flexible restriction on what is the actual state with the following conventions

- $\pi(S) = 0$ means that state S is rejected as impossible;
- $\pi(S) = 1$ means that state S is totally possible (= plausible).

For instance, let Ω = {S-Suc, S-Fai} be the two functioning states of a considered service.

At a given time, the service can assume one and only one of the two states of Ω, and we are seeking to know this state. All probabilistic based approaches exploits the available statistical sources of knowledge in order to attribute probability values to both states p1= Pr{S-Suc}, and P_2= Pr{S-Fail}, where $P_1 + P_2 = 1$; whereas, possibility theory defines a possibility distribution π given by two values π (S-Suc) and π (S-Fai).

Where, $0 \leq \pi(\text{S-Suc}), \pi(\text{S-Fail}) \leq 1$.

In other words, the constraint of unit total belief is relaxed. The possibility degree of the service success or fail may assume any value within the interval [0, 1].

In the case of total ignorance (i.e. no knowledge is available), then both states have a unit possibility degree π(S- Suc)=1 and π(S-Fail)=1.

The case of total knowledge, for instance success (resp. Fail), is represented by the possibility distribution: π (S-Suc)=1 and π (S-Fail)=0 (resp. π (S-Suc)=0 and π (S-Fail)=1).

Possibility theory offers a framework for preference modeling in constraint directed reasoning. Both prioritized and soft constraints can be captured by possibility distributions expressing degrees of feasibility rather than plausibility. Qualitative decision criteria are particularly adapted to the handling of uncertainty in this setting. Finding the potential of Possibilistic representations in computing conservative bounds for such probabilistic calculations is certainly a major challenge. Other applications of possibility theory can be found in fields such as data analysis, database querying, diagnosis, belief revision, argumentation, and case-based reasoning.

In the proposed approach, the first step consists of associating each potential operational context of a component, C_n, with a possibility distribution, πCn, defined over the functional service states set Ω = {S-Suc, S-Fai}, and then, combining different component possibility distributions in order to establish a global service functional possibility distribution.

This process can thus be expressed as follows: π_{Cn}: $\{a_0, a_1, a_2\}$ x $\{C_n\text{-Suc}, C_n\text{-Fai}\}$ → [0, 1] x [0, 1]

Where: $(a, b) \rightarrow \left(\pi_{Cn}(\text{S-Suc}), \pi_{Cn}(\text{S-Fail})\right)$

- a refers to the component C_n importance factor;
- b refers to the component C_n functioning state;
- π_{Cn}(S-Suc) is the possibility degree for the service S to function successfully conditioned by the component C_n operational context (a, b); and
- π_{Cn}(S-Fail) is the possibility degree for the service S to fail, conditioned by the component C_n operational context (a, b).

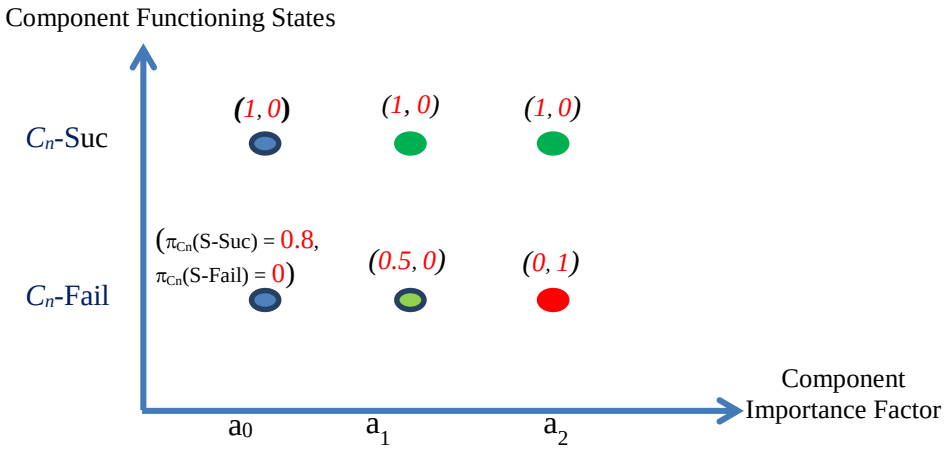


Figure 2.6. Service (S) possibility distributions induced by the component Cn operational context

Here follows the proposed attributions of possibility degrees:

$$\left(\pi_{Cn}(\text{S-Suc}), \pi_{Cn}(\text{S-Fail})\right)$$

Notice that, as figure 2.6 demonstrates:

- If the possibility degree for the service S to function successfully is less than the unit value (i.e. < 1), then this will indicate that the component failure has an important impact on the service functioning state;

- When the service functional possibility distribution is given by: (π_{Cn}(S-Suc) =1, π_{Cn}(S-Fail)=0), then the component optimally contributes to the service functioning;
- When the service functional possibility distribution is given by: (π_{Cn}(S-Suc)=0, π_{Cn}(S-Fail)=1), then the component impacts drastically the service functioning, forced to failure;
- In the case where the component operational context is: (a_1, C_n-Fai), this implies (π_{Cn}(S-Suc)=0.5, π_{Cn}(S-Fail)=0) indicating that the service is in the Success functioning state (due to the presence of the alternative component;
- In the case where the component operational context is: (a_0, C_n-Fai), this implies (π_{Cn}(S- Suc) =0.8, π_{Cn}(S-Fail) =0) indicating that the service is in the Success functioning state (the component is not critical for the service functioning). Nevertheless, forcing π_{Cn}(S-Suc) =0.8 indicates the presence of an erroneous component which is not critical for the service functioning).

After the critical analysis of fault tolerance approaches, and suggesting an intermediate proposition of an enhanced fault tolerance mechanism, with the consideration of importance factor, the next section is devoted to analyze different approaches of fault and failure of system elements. Mainly, these approaches depend on the Fault Tree Analysis FTA, which is generally used in several researches and projects that argue the reliability and availability concerns, in variant domains, not only in the domain of home automation systems.

2.6. Fault Tree Analysis

Behind every service failure there is one or more fault(s). So, the goal is to identify the causes of a system's potential failures. Fault Tree Analysis helps to identify the source of the failure(s) and to evaluate the influence of faults on the service functioning.

2.6.1.FTA concept definition

Fault Tree Analysis (FTA) constitutes an important symbolic logic analytical knowledge representation technique applied in the operations research discipline and in the design of several software fault tolerance based systems allowing handling potential failures.

In fact, FTA is a top down, deductive failure analysis approach targeting to analyze the effects of initiating faults and events on a complex system. An undesired state of a system is analyzed using Boolean logic to combine a series of lower-level events. This analysis approach is mainly used in the fields of safety engineering and reliability engineering to understand how systems can fail, to identify the best ways to reduce the service failure risk or to determine (or get a feeling for) event rates of a safety accident or a particular system level (functional) failure. FTA is used in various application domains including nuclear systems, and other risky areas; but is also used in fields as diverse as risk factor identification related to social service systems failure [Lacey, 2011].

In the domain of software engineering, FTA is mainly used for debugging purposes and is closely related to the cause-elimination technique used to detect bugs/errors. FTA, as a failure analysis approach, can also be used to monitor and to control the safety performance of complex systems, as well as system's functioning as an investigative tool to identify and correct causes of the top event failure. A Fault Tree (FT) is a graphical model representing the combinations of parallel and/or sequential fault events that can lead to the occurrence of the predefined undesired top event [Eri-05]. It shows logical relationships among basic events up to the top event. Boolean gates symbolize the relationship between inputs and the system output. The higher event is the output of the gate and the lower events are the inputs to the gate. The functioning of the desired top event strongly depends on the reliability data of primary events, also known as basic events.

The basic FTA structure, illustrated in figure 2.5, refers to the logical representation of the undesired event (i.e. the service under study) to more basic events (i.e. the components that comprise the service). The top event of the fault tree is the top event. The middle events are intermediate events. The bottom of the fault tree refers to the causal basic events, or primary events. The logical relationships of the events are shown by logical symbols or gates [NASA, 2002].

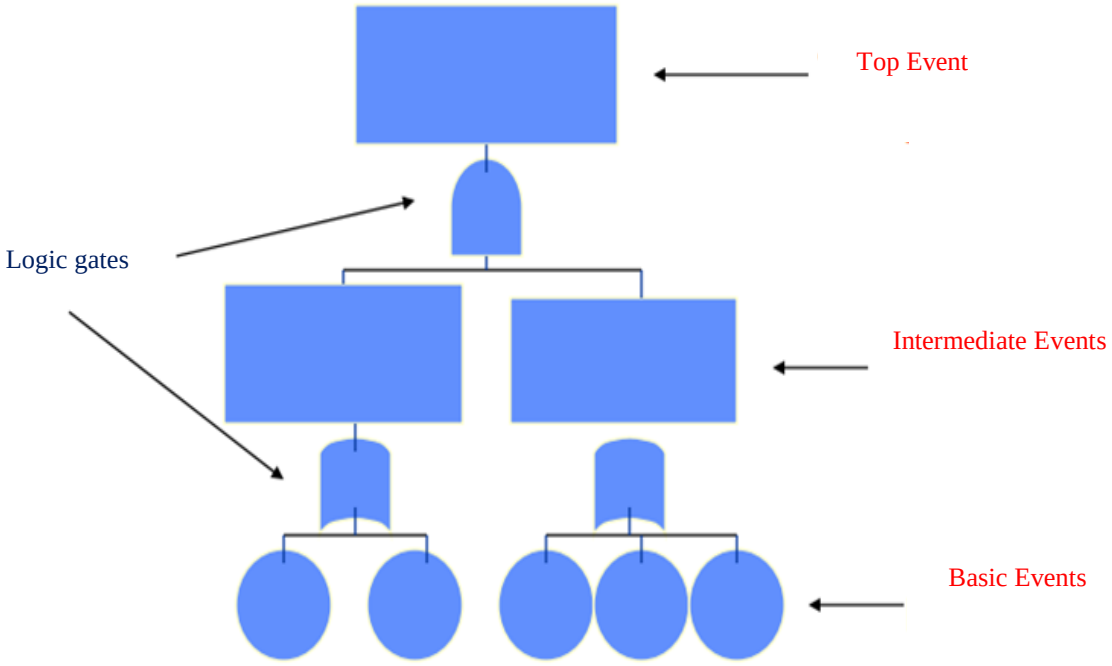


Figure 2.7. Basic Fault Tree Structure

In fact, the FTA is a widely used technique in the purpose of reliability and safety analysis of several types of complex systems. Fault tree analysis is one of many symbolic "analytical logic techniques" found in operations research and in system reliability.

FTA, as an analysis method, could be applied in the modeling process of a given system and used to authenticate the adequacy of the system design. [NASA, 2002] proposed the use of Fault Tree for a system is being designed as well as for a system is implemented and operating. They defined the fault tree as an important tool in assisting in the design of the system, as well as a tool for the evaluation and validation of system design.

In the following chapters, the use of FTA is proposed as decision aide mechanism for the selection of alternative components when potential failure occurs.

2.6.2.Fault Tree analysis and system life cycle

From a software engineering point of view, where does the FTA (as a tool and/or an approach) takes place during the system lifecycle of any system (Ambient Assisted Living system as well)?

From its own introductive name, FTA is considered as an approach to analyze the system in order to determine potential failures in a bottom-up analysis of the basic parts and components constituting the corresponding system.

As a technique, FTA is also considered in order to evaluate the weaknesses that could be revealed in the system design at any step during the system development. In fact, FTA provides a cost- effective

means of improving or verifying the consistency and efficiency of system's design [Hessian *et al*, 1990].

FTA techniques could be applied during the design process and used to authenticate the adequacy of the design. In [NASA, 2002, p.110], the authors proposed the use of Fault Tree for a system is being designed as well as for a system is implemented and operating. They defined the fault tree as an important tool in assisting in the design of the system, as well as a tool for the evaluation and validation of system design.

In the next subsection, ongoing research activities dealing with encountered problems for the application of FTA (using the Bayesian reason approach) are discussed. The main focus concerns the approaches dealing with imprecise probabilities and how to avoid this difficulty. The concept of events importance is also raised.

2.6.3.Ongoing research activities

In this section, a review of the main ongoing research activities dealing with the representation and processing of imprecise statistical data about failure of system components and services is proposed. Activities that highlighted the proposition of importance factor in Fault Tree Analysis and its influence into the decision related to which scenario to be followed, (when talking about different scenarios to deliver services in Human Ambient Assisted Living Systems) are also considered.

After a close analysis of the state of the art, it seems that most efforts are turned to deal with imprecise probabilities by substituting these probabilities by fuzzy membership values or possibility degrees. The reason behind this trend is to overcome the unavailability, or the small size sample base, of statistical data which lead to imprecise probabilities. Moreover, as the importance factor (also called importance measure) is mostly related to subjective evaluations and is a human related factor, it seems quite natural to model this measure/factor using fuzzy/possibility based concepts.

One of the very first studies, conducted in 1983, [Tanaka *et al*, 1983] is motivated by overcoming the difficulty of dealing with imprecise failure probabilities. The main idea of this approach consists on substituting precise failure probabilities of different components within the system by trapezoidal fuzzy numbers and using fuzzy set theory concepts, mainly the extension principle, in order to reach a final estimation of the top event possibility of failure, also given as a trapezoidal fuzzy number.

[Philip & Emad, 1995] introduced the concept of importance measures from fault finding point of view to determine the failure of a system. They discussed diverse approaches of importance measures. The proposed measures are differentiated by the consideration of system and components reliability, i.e. failure proportion. They presented the concept of structural measures. This concept evaluates the criticality of a component without taking into account the reliabilities of various components. In other words, failure probabilities are “not considered” for the definition of the importance measures. An enhancement of this measure is achieved- called time dependent measures of importance – to take into account component’s reliabilities at a fixed point of time, which are the probabilities that components are critical to the functioning of the system at particular points of time. A global view of component importance measures is discussed as an improvement of the previous measures, by considering reliabilities at various points of time, and is called time independent measures of importance.

[Suresh *et al*, 1996] proposed to transform numerical probabilities of failure into triangular fuzzy numbers membership functions to allow better representation of imprecise probability values. In this approach, instead of computing the global failure probability, the authors used a fuzzy importance measure, based on a distance measure between fuzzy numbers, at the level of each basic component, in order to compute an importance value for each basic component for the good functioning of the service. This distance measure is calculated based on the Euclidian Distance (ED) between two fuzzy numbers,

1) a component is fully available in the calculated top event. 2) A component is fully unavailable in the calculated top event, as shown in figure 2.8.

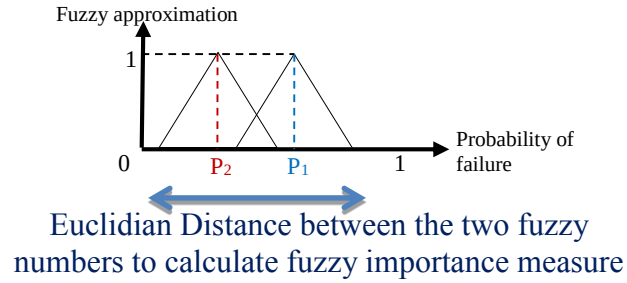


Figure 2.8. Calculating fuzzy importance measure by the use of Euclidian Distance between two membership functions

In 2002, NASA Labs addressed different topics related to Probabilistic Risk Assessment (PRA) based on the introduction and the use of FTA as one of the most important logic and probabilistic approach used in PRA and system reliability assessment today [NASA, 2002]. The concept of importance measure is discussed in details as one of the most important outputs of an FTA. A set of importance measures are calculated for the top event, to evaluate the impact for all the events in the fault tree in terms of their contributions to the top event probability. Independently, all events in a fault tree can be ranked according to their importance factor. Top importance measures can also be calculated in order to appreciate the sensitivity of the top event probability to an increase, or a decrease, in the probability of any event in the fault tree. In addition, the importance measure here is used to allocate various types of resources (testing maintenance resources, quality control requirements, etc.), in order to optimize the top event occurrence probability.

In 2011, [Aven, 2011] described uncertainties related to risk assessments and applications by using different existing alternative approaches for representing uncertainties rather than the common probability. A global perspective has been introduced for this representation to measure uncertainties within one system, including probability, possibility theory and evidence theory. By this approach, different processes are used to represent the uncertainties related to different components or sets within a system.

In a very recent research conducted by [Liping & An, 2010; Liping *et al*, 2013], imprecise probabilities of failure are transformed, using fuzzy concepts, into epistemic numerical risk indicators considered as possibilistic values. The risk indicator, as an importance index, is used to measure the level of reliability and risk.

The thesis work [Julwan, 2013] discussed the difficulty of obtaining components failure rates and how the fuzzy set theory can be applied to overcome the weaknesses of conventional fault tree analysis in nuclear systems. In his Ph.D. thesis, Julwan described a new intelligent hybrid fault tree analysis framework to overcome the weaknesses of conventional fault tree analysis, qualitative failure possibilities and their corresponding mathematical representations to clarify nuclear event failure likelihoods. In this thesis work, new software has been developed to realize an intelligence hybrid fault tree analysis framework to overcome the limitations of the existing fault tree analysis systems by accepting both quantitative failure probabilities and qualitative failure possibilities.

[Braham, 2013] presented an approach for dealing with the difficulty of making a decision under the joint presence of uncertainty and imprecision. It is hard, for experts, to make a decision about how to undertake system fault tolerance in case of errors in execution. In his approach, Braham proposed to implement the Bayesian model to handle the uncertainty factor.

Another approach was proposed by [Sallak *et al*, 2013] in order to overcome the difficulty in identifying the components that have more significant influence on the system's behavior with respect

to the availability. To achieve this approach, they have introduced a reliability model based on the transformation of epistemic uncertainties into epistemic uncertainty of importance measures of components. This transformation is based on the consideration of the evidence theory, or belief functions theory, as a framework for taking into account both aleatory and epistemic uncertainties. The idea of using evidence theory (Dempster – Shafer theory) is that it is considered as a "simplification" of probability theory.

In the same manner, [Anurag *et al*, 2014] proposed a reliability model to replace the probabilistic representation of basic events by possibilities and membership function. The use of fuzzy set theory is proved to be effective in handling many types of uncertainties. In that way, it leads to fuzzy fault tree analysis. In this model, they used triangular and trapezoidal type-2 fuzzy numbers in order to characterize the failure possibility of the basic events.

Chapter 4 is dedicated to present a technical contribution related to the proposed approach allowing dealing with both failure probabilities as well as expert-defined importance factors in the very same framework.

2.7. Proposition of a dynamic reconfiguration approach

Finding an alternative component to replace the faulty one in a dynamic manner requires the definition of the working context of a component, i.e. the property of the component for each particular service that may participate in performing it, the time period when the component would be activated for a specific service.

We are also interested in defining the spatial position of the component in the living space (X, Y, Z) dimensions, in order to specify the nearest component to it that may be used as an alternative component (or the nearest composite service to replace the faulty component).

In addition, it is wise to define (in the component's context) the operation of the component during the execution of the service, for instance: a light will be switched on for a specific service and will be switched off for another service.

In other words, we are interested in defining a metadata for each component and service in order to describe their main properties. The information provided by this metadata will be used by the system to define and to accomplish a set of queries allowing defining the most alike component or composite service to replace a faulty component in a dynamic manner instead of the classical static reconfiguration technique.

In the case of dynamic reconfiguration implementation, the system applies two levels of queries (nested queries). First, the system inquires (based on the properties of all components in the system) the most alike component and / or composite service to be used instead of the faulty component, then the system applies an additional query on the selected alike components, in order to inquire the alternative component with the most possibility alike.

The possibility alike (or possibility of similarity) of each component, based on the metadata of components, is illustrated in figure 2.6, where, in this matrix, the two dimensions, rows and columns, represent the set of components in the system.

For the considered service, the matrix element associated with the couple (C_I, C_J) representing the similarity possibility value of the component C_J in relation with the property of the component C_I.

For instance, in figure 2.6, C_1 is similar (in terms of substitution capacity) to the properties of C_I with an empirical value (*a priori* fixed by the expert) of 0.3 (for instance, a light with a property of

lightening and a window's shutter which also has the property of lightening in the same room in the afternoon), where C_I' is similar to C_I with a possibility of similarity 1 (i.e. fully redundant component, for instance, two lights).

The components similarity is evaluated using different parameters derived from the components Metadata, such as the context of each component, the location of the component, etc.

The similarity is a *priori* estimated (empirically by the expert) upon the use of different parameters derived from the component Metadata, such as the context the location… For instance, the similarity between adjacent light and a window (in the context of lightening) is higher than the similarity between the very same light and another light situated far away. Thus, the expert is appealed to give a similarity value "appreciation" taking into consideration the service usage context.

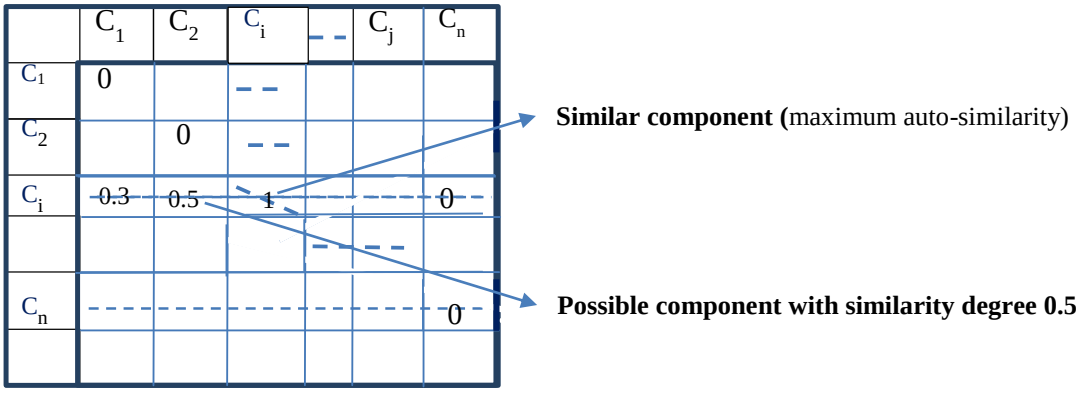


Figure 2.9. Similarity matrix in terms of substitution capacity

Therefore, in order to represent the dynamicity fault tolerance of our system, we suggest appending the Metadata for each component in order to express its main specifications as previously mentioned.

Table 2.2 illustrates an instance of the definition of Metadata component propagation.

Component Id	Component Name	Service1 id	Time slot for service1	Operation in service1	Usage in service1	Service2 id	Time slot for service1	Operation in service2	Usage in service2	Spatial position (x,y,z)
1001	Ceil Light1	11	Evening	Switch ON	Lightening	12	Evening	Switch Off	Darken	(100,100, 3)
1002	Door1	17	Evening	Open	Entrance to	35	Evening	Close	Get out of	(10,15,1)
1025	…	…	…	…		…	…	…		…

Table 2.2. An instance of the definition of Metadata components propagation

As shown in table 2.2, the set of services, that use each component, is defined, with the expression of the time periods of the execution of the service (morning, evening, night / winter, summer…).

For each service, the state of the activated component is selected. Additionally, the component's spatial position is localized to evaluate the closest component having the same property of the faulty component located in a compatible location to deliver the service (e.g. in the same room).

The presentation of the Metadata has more complexity as shown in Table 2.2. For instance, as mentioned previously, sometimes a component can be dynamically replaced by a composite of components instead of one component, or we should pre-test the state of the component before it is used by a service (a light, in a given service, must be switched on, but because the same light was

switched on previously by another continuous service, thus, no operational change should be performed on this particular light).

Another issue is related to the benefit of using a particular component in a specific condition. e.g. supposing a service which is using the heater during the winter, and for an individual day, the temperature is moderate so it is not mandatory to use the heater then. All these considerations need to be considered in the representation of the Metadata of the whole components.

2.8. Illustrative example

In this section, a summary of the notions and concepts, raised in this chapter, will be presented.

Considering a simple example of a service: "*having a shower in the morning*" (S_1). This service requires the use of a set of components, which are:

1. A "*sensor detector*" (C_1) to detect the presence of a person in the bathroom;
2. A "*Ceil light1*" (C_2) and
3. A "*water tap*" (C_3).

In this service, the sensor detector C_1 is considered as being not critical (NC). The Ceil light1 C_2 has an auxiliary importance (A) for the service. Whereas, the water tap C_3 has a primary importance (P).

Another service, in the same bathroom, is "*Brush teeth*" (S_2) which requires another component "*mirror light*" (C_4), in the bathroom there is also a "*window shutter*" (C_5).

For this service, the mirror light C_4 is considered of an auxiliary importance (A), whereas the window shutter C_5 is considered as not critical (NC).

In this simple example, we have:

1. $S = \{S_1, S_2\}$ the set of the two services: {*having a shower in the morning* ; *Brush teeth*};
2. $C = \{C_1, C_2, C_3, C_4, C_5\}$ the set of available components in the bathroom; and
3. $S_1 = \{C_1, C_2, C_3\}$, $S_2 = \{C_4, C_5\}$: the required components of each considered service.

In the first service "*having a shower in the morning*", the components operational context is given by $I(S_1, C_i)$; i = 1, 2, 3. The operational context of the components assumes the following issues:

$C_1 \rightarrow a_0$, because the component "*sensor detector*" is considered as being not critical to the service S_1;

$C_2 \rightarrow a_1$, the component "*Ceil light1*" is assumed being important to the service S_1, but it can be substituted by the component "*mirror light*" (the component C_4 from the service S_2) in case of a failure;

$C_3 \rightarrow a_2$, the component "*water tap*", is considered very important to the service S_1, and it has no alternative.

The same discussion is applied for the second service "*Brush teeth*". The components operational context is given by $I(S_2, C_j)$; j = 4, 5. The operational context of the components assumes the following issues:

$C_4 \rightarrow a_1$, the component "*mirror light*" is considered as being important to the service S_2, but it can be substituted by the component "*Ceil light1*" (the component C_2 from the service S_1) in case of a failure.

$C_5 \rightarrow a_0$, because the component "*window shutter*" is assumed being not critical to the service S_2.

The metadata for set of components of this example is shown in Table 2.3.

Component Id	Component Name	Service1 id	Time slot for service1	Operation in service1	Usage in service1	Service2 id	Time slot for service1	Operation in service2	Usage in service2	Spatial position (x,y,z)
1000	Sensor Detector	11	Morning	Sensing movement	Sensing movement	-	-	-	-	(50, 60, 3)
1001	Ceil Light1	11	Morning	Switch ON	Lightening	-	-	-	-	(100,100, 3)
1002	Water tap	11	Morning	Open warm	Shower	-	-	-	-	(90,90,1)
1025	Mirror Light	12	Morning	Switch ON	Lightening	-	-	-	-	(110, 110, 1)
1044	Window shutter	55	Morning	Open	Lightening	67	Evening	Close	Darken	(130, 130, 2.7)

Table 2.3. The Metadata of the example

When the component "*ceil light1*" doesn't work properly (i.e. faulty component) for performing the service S_1, the system should dynamically query, according to the metadata, to investigate the most similar component that can replace the faulty light component. From the Metadata table, the most similar components are (1025, *mirror light*) and (1044, *window shutter*). Even though the mirror light is positioned nearer than the shutter to the ceil light, but the system dynamically analyzes that the window shutter has more possibility of a similarity value than the mirror light (since the service is performed in the morning: daytime light).

2.9. Conclusion

In this chapter, the motivation, concepts and definitions of reconfiguration have been expressed including the research directions in terms of ensuring home automation systems continuity.

An abstract formalization of the home automation system structure is proposed. This formalization intends to provide effective and efficient services, and ensure the continuity of the proposed services (through achieving and including fault tolerance mechanisms with the provided services).

Concepts and terminologies describing service availability, reliability and reconfiguration are introduced, followed by a definition and study of characteristics of fault tolerance mechanisms applied in distributed systems (including home automation systems).

The relative relationship between system components and the provided services is quantized in terms of "*the criticality*" of a given component to the functional continuity of a service. This criticality concept is detailed deeply and identified in chapter 4 of this manuscript.

The abstract system model is further detailed in the next chapter, to lead the modeling process of system structural elements, as a detailed workflow, in terms of a Model Driven Architecture Approach.

Chapter3.

System design, analysis and behavior workflow

Taking into account the reconfiguration and fault tolerance mechanisms, the formalism (proposed in the previous chapter and allowing realizing the home automation system formalism) is studied and its technical implementation is proposed.

The key task consists on developing a workflow to describe the structural design of an Ambient Assisted Living (AAL) System which undertakes the fault tolerance and fault analysis approach to overcome potential errors in such a system.

The system's workflow underlines the design of the system by describing the structure, the behavior of the considered system and its analysis. Also, the system's design workflow allows providing a standard approach to envision the design of an AAL system.

After presenting a short introduction about the key objective to be carried by the workflow (section 3.1), the system's design and elements structuring are illustrated and discussed.

The main model blocks, handled by the workflow, are also clarified (section 3.2).

Section 3.3 is devoted to present and to describe the system design model structure, as well as the Meta-Model conforming to this model.

Section 3.4 is dedicated for detailing the system analysis model, in terms of the objective for performing the analysis phase, the structure of the analysis model and the conducted transformation rule process (to make transition from the design model into the analysis model).

In the same manner, section 3.5 is describing the behavior model target, structure and its dedicated transformation rule.

Finally, the workflow phases are coupled with an illustrative example in section 3.6.

3.1. Introduction

The main objective of this chapter is to present and to develop a workflow describing the structural design of an Ambient Assisted Living (AAL) system which undertakes the fault tolerance approach and fault analysis to overcome the potential errors in such a system.

In fact, a system workflow underlines the design of the system, describes the structure, behavior, and other views of the considered system and analysis. The system design workflow also allows providing a standard way to envision the design of an AAL system.

Moreover, the proposed workflow is assumed to offer a way to envisage a system's structural design including elements such as [OMG, 2014]:

- Activities to be achieved by the expert designer;
- Individual entities of the system and how these entities can interact with other system entities;
- How the system will run.

System's design and analysis is used to unravel internal problems, to improve efficiency and to expand opportunities. It has a direct impact on the system quality performances.

In order to propose a well-defined system design, it is important to define the main elements contributing to the system structure. In the particular case of home automation systems, the main element of the system concerns the services provided to the final user. Each service, involved in the

system, is mainly composed of a set of components (hardware devices, sensors and actuators, as well as "the piece" of software that is controlling the hardware part of the component). Besides the main components, to overcome the potential failures, alternative components need to be defined in a way allowing maintaining the context consistency of particular services.

To overcome potential failures and always choosing the best execution scenarios of the service, the structural system design needs to have an analysis phase following the design phase. This will allow to study and to investigate whether the system (with the stated configuration) will work correctly or some failures could appear and cause the system or a part of it, to stop working). For this reason, system's analysis based on fault analysis approach needs to be appended following the system design in order to analyze the system elements (including services and corresponding components).

The analysis approach, to be proposed in the next chapter, fits the analysis phase of the structural design of the system. It is related to the integration of a risk indicator (introduced in chapter 2) into the fault analysis approach (FTA).

In this chapter, we propose a workflow process to carry on the set of activities responsible for the development of a model representing the system structural elements. This model representation resumes the system abstract formalization (which is characterized in the previous chapter).

Moreover, to formalize the desired workflow architecture, the process and the tasks conducted by the designer are illustrated in figure 3.1. This workflow process states the major sequential tasks that should be performed, in order to define the desired system, with the consideration of fault tolerance mechanism, to meet the user's expectations and requirements. By these tasks, the expert designer is in charge to establish the system scheme, in terms of system services compositions and the relationship between the services and the components. The system design scheme is being developed according to users' needs and desires. The design of the system passes an analysis phase to assess the correctness of the system structure in terms of reconfigurability in case of potential failures. In the same manner, the system design is assessed using a behavior analysis stage to demonstrate the exactness of service execution to reach the deliverance point.

From the previous chapter, the system abstract formalization was conducted so that to tackle the fault analysis and reconfiguration ensuring, thus, the continuity of the system. The system abstract consists mainly of the components (hardware devices and the software that is controlling the hardware parts) participating to perform the set of predefined services. Moreover, to define a service, the designer has to specify the set of parameters related to the general context definition of the service. The system's structural design is a process that has to be conducted with a close participation of the final user so that the designer could configure the system services according to the user's opinion, requirements and expectations.

As the main goal is to propose a workflow allowing the development of a reliable Home automation system for users and ensuring the continuity of system's services (based on the functionality of different system's components); it is thus crucial for the designer to analyze the proposed structural design of the system. This analysis allows to identify the component(s) that cause the most critical problems to the system and to prioritize improvements in the system design [Weibull, 2006] (by making revision of the design and enhancing possible structural design configurations modification).

System's behavior is also proposed to mimic the process of how the system could be performed at the level of its services and its corresponding components.

As a final task, the proposed solution (service and constituting components) is evaluated from both the designer and the user points of views, in order to appreciate, or to redesign, the proposed configuration.

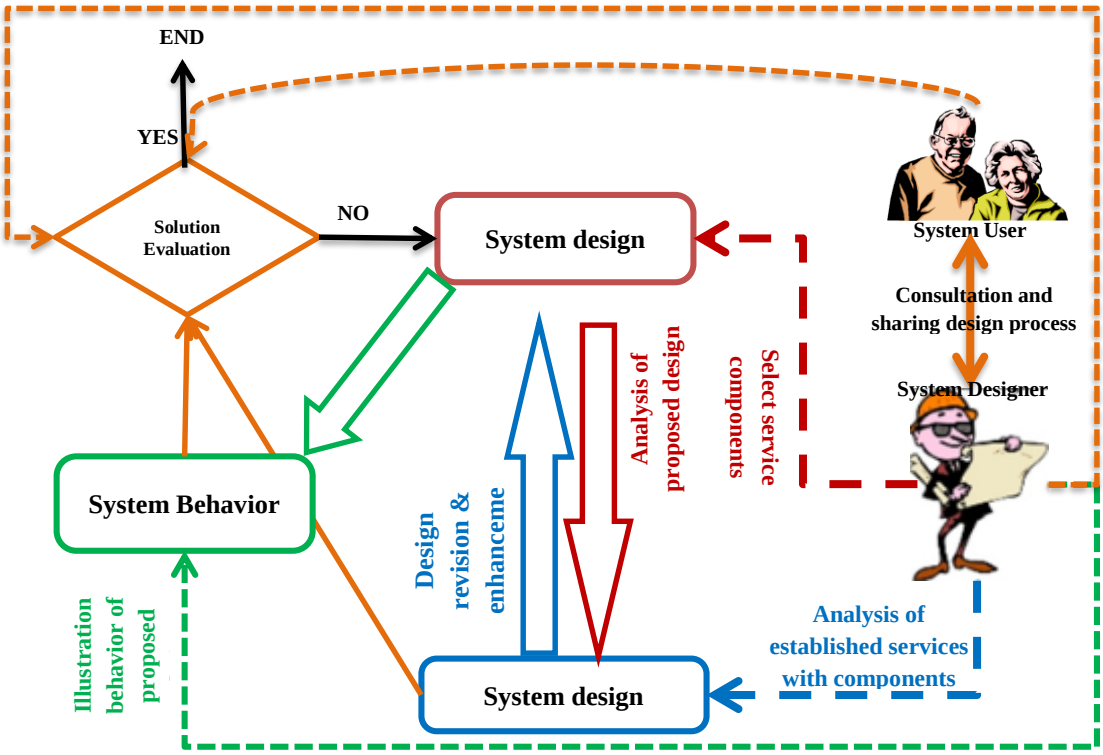


Figure 3.1. Workflow architecture process

3.2. System workflow

In this section, the system workflow is introduced. In this workflow, based on the activities to be undertaken by the designer, he/she is in charge of defining the representation of the system structural elements, including each service context and parameters. Therefore, for each service, the corresponding components and alternative scenarios will be defined (taking into account the consistency and the convenience between components contexts and service context).

To evaluate the system's design correctness (so that to avoid as much as possible the unexpected failure of the system or parts of it), a system analysis of potential failures will be conducted, using fault analysis approach, to overcome the system services from cutting down. The system analysis process is modeled in the workflow to represent the analysis part.

The use of fault analysis will allow the system to always choose a better collection of sound and available components (which have not a proportion of unavailability), to perform the considered service. In addition, the system analysis will always propose adequate alternatives of the component that respect the general context, as well as it has high ratio of availability (failure is in the smallest probability value).

To check the exact execution performance of each provided service, a behavior model is established to make a simulation of system execution, and verify that the selected scenarios (for each service) reaches the deliverance issue, and prove the viability of system design.

The overall system workflow processes and phases are illustrated in figure 3.2.

In fact, the system workflow consists mainly of three stages. Each stage refers to a sub process in the whole process of the system structure development, with consideration of the reconfiguration requirement.

The first phase consists of converting the system abstract formalization (proposed in chapter 2) into a design model of the system structural elements, in terms of services and components (hardware and software components).

The second phase aims at performing a fault analysis approach of the system's services structure, to conduct a fault tolerance mechanism of system elements, and propose alternative scenarios to ensure the continuity of system performance.

Whereas the third phase aims at leading an analysis process of the system behavior, to represent the ability of the proposed services' scenarios to lead the service to reach the deliverance issue.

In the proposed workflow, transformation rules are being proposed to conduct a transition process between the design phase and the fault analysis phase, as well as a transition between the design phase and the behavior analysis phase.

The three phases of the workflow, as well as the transformation rules interconnecting these phases, are involving the main technical contribution proposed in this research. Each phase of the workflow is being proposed following the model driven architecture paradigm, to model the system structural elements, as well as the analysis and behavior approaches.

The workflow phases are more detailed in the following sections. The design model phase is illustrated in section 3.3. The system's analysis modeling phase is detailed in section 3.4, in addition to the proposed transformation rule to lead the design model to the analysis model. The proposed approach used to conduct the analysis phase is detailed in chapter 4. Whereas, the system's behavior model is presented in section 3.5, in addition, the transformation rule from the design model to the behavior model is proposed.

System design, analysis and behavior workflow

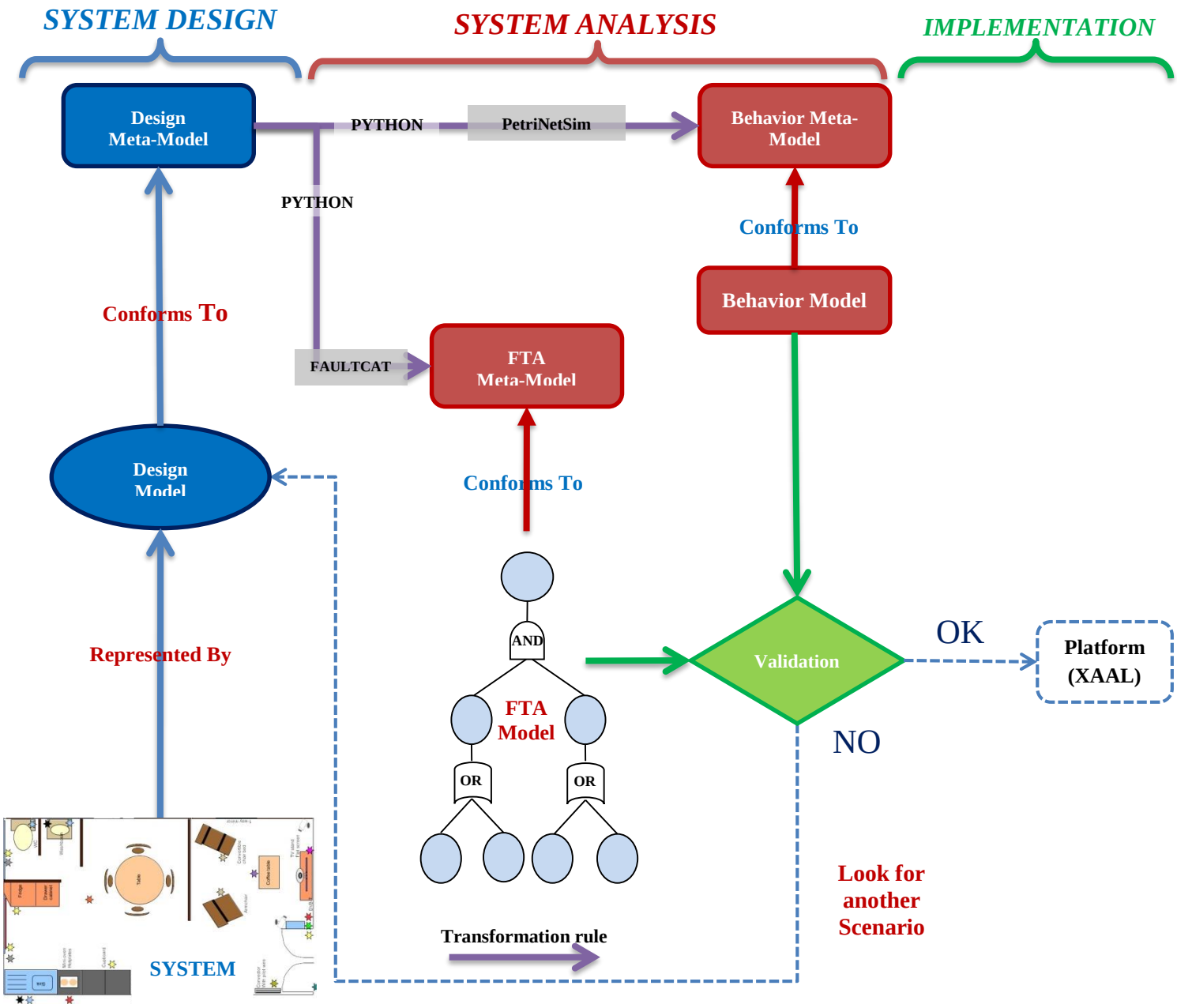


Figure 3.2. System workflow

3.3. System design model structure

An illustration of the scheme of an AAL system design is proposed. System's design is the process defining the style, modules, boundaries, and data for a system in order to satisfy *prior* specified requirements. System's design could be seen as the application of systems theory to product development.

There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. In fact, system's design is considered as an important phase in any system development life cycle (SDLC) [Pendharkar *et al*, 2008], which consists of a framework that is used to structure, plan, and control the process of developing a system.

System's development is composed of a number of clearly defined and distinct task phases which are defined by the systems expert designers to plan for, design, build, test, and deliver the desired systems. Like any manufactured object on an assembly line, system's development aims to produce high quality systems assumed to meet, or even to exceed, the user's expectations, based on the user's requirements.

In systems design (as a work phase), the design functions and operations are described in detail, including business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules (or subsystems).

To establish a well-designed system (including system's analysis of its components and services), the designer should first set the abstract tasks needed for achieving his work in clear and good steps, starting from establishing how the information about the system's elements will flow between each part of the desired system architecture. In other words, the designer ought to establish, first, a design workflow to clearly define and comprehend the main tasks of each part of the system's structural design. Moreover, in order to develop the workflow of any system, the design framework of the system needs, first, to become conscious.

As system's design is the process of defining the style, modules, boundaries, and data for a system to satisfy specified requirements, it is considered as an important phase in any system development life cycle (SDLC) [Pendharkar *et al*, 2008] (which consists of a framework used to structure, plan, and control the process of developing a system).

To have a well-defined system's design, the Model Driven Architecture (MDA) paradigm stands out from the different software engineering approaches. These approaches assume that all, or part, of the computer application is generated from models.

The Model Driven Architecture (MDA), [Miller & Mukerji, 2001] provides tools, concepts, and languages to create and to transform models.

MDA is presently making several promises about the potential benefits that could be reaped from a move from code-centric to model-based practices [Bézivin, 2004]. The MDA is based on a hierarchy of manipulated models involving two basic concepts: the model and the meta-model.

- The ***model*** is the abstraction of a system. Figure 3.2 gives the relationship between a model and a system [Bézivin, 2005]. Figure 3.3 is read as: a system is "represented by" a model;
- The ***Meta-Model*** represents the syntax used to define models. A Meta-Model is considered as a model that provides the structural elements required for defining a lower level model. It is said that any model is "conform to" a meta-model.

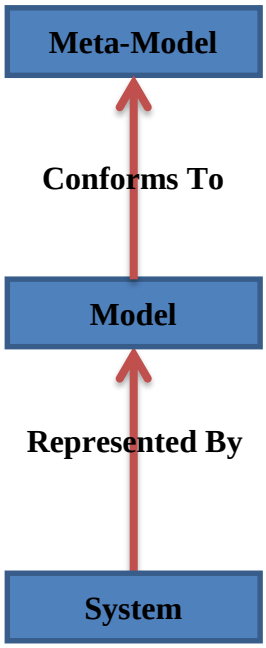


Figure 3.3. Fundamental concepts of the MDA: hierarchy of the models

The formalization of the Meta-Model is based on a static diagram structure describing the structure of a system by showing the system's structural elements, their parameters and the relationships among different objects.

The Meta-Model is a syntax representation allowing describing the structure of a system by showing the system's elements, their attributes, and the relationships among objects. The purpose of a Meta-Model is to illustrate the composition of a design model.

In the design of a system, several elements modules are identified and grouped together to determine the static relations between these elements. With a detailed modeling, the modules of the conceptual design are often split into several sub-modules.

Figure 3.4 shows up the Meta-Model, in which the association between services and their components will facilitate the structural design of the system. This will allow easily developing, managing and manipulating system's model by referring to the proposed Meta-Model.

In this Meta-Model, the relationship between the service and its corresponding main components is formed.

Each "*Service*" is represented using several attributes related to the context description and parameters characterizing each service. Each service could be either a simple service, in which it is composed of several components, or it could be composed of multi sub services. For instance, a service "going to bed" room requires first performing the sub services "Turn OFF the TV" and "urn OFF the heating system".

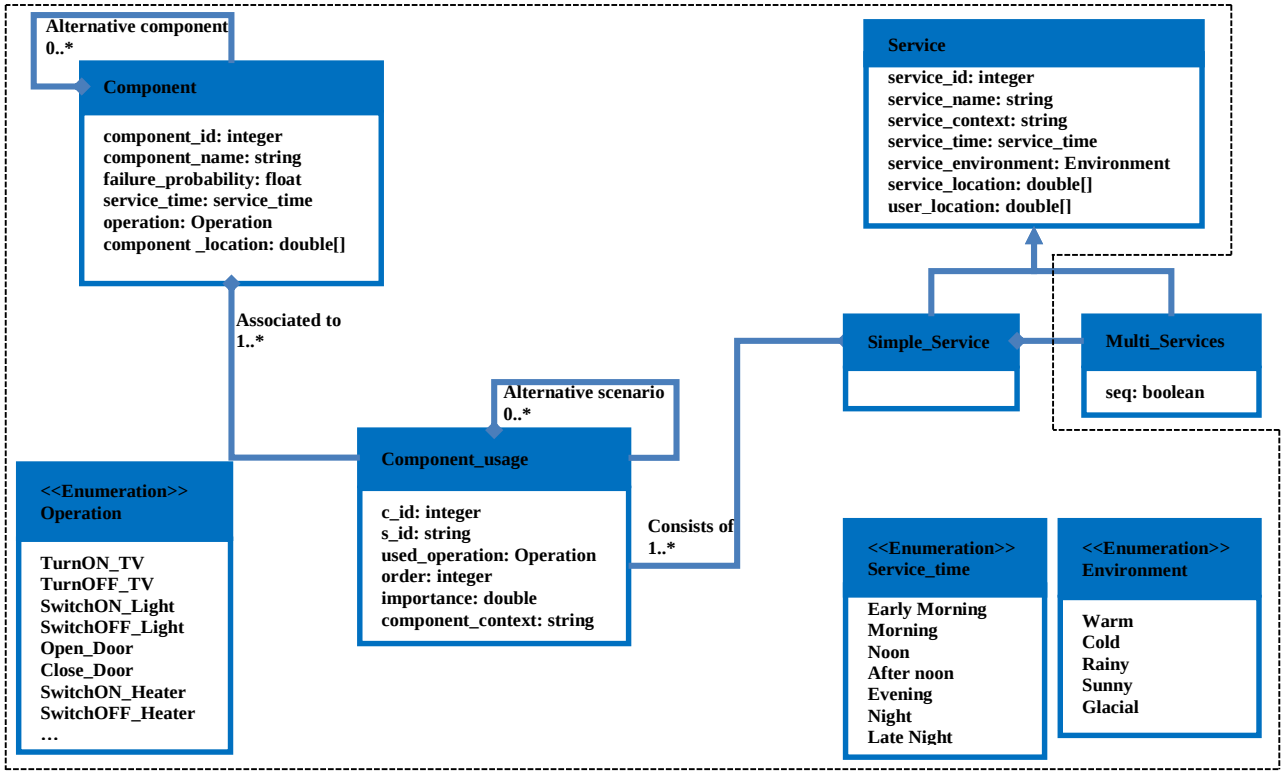


Figure 3.4. The Meta-Model of the system design

In turn, the "*Component*" is represented as an abstract component with several parameters referring to the basic characteristics of a components (the ID, name, the probability of failure, in order to estimate the degree of potential failure of the component and the chance to substitute it with an alternative(s) component(s) and the set of operations that could be performed by a given component).

The association, i.e. relationship, between the two notions, the *Service* and the *Component*, is formed by another notion representing the functionality context of a particular component within the service's context. In other words, the relationship between a *Service* and a *Component* will differentiate the component's context between different corresponding services. The main parameters related to the definition of the component's context are the degree of importance for a component in the contribution in the service (the importance concept introduced in chapter 2, and will be treated in chapter 4), as well as the activation order of the component in the service (for instance, the light needs to be switched ON before the TV to be used). This order parameter takes values from 0 to n where n is the number of the activated components in a given service. A zero order value is used to indicate the parallelism use of a component. Some components may be activated in parallel during the service execution (for instance, the smoke detector is activated in the service of the lunch cooking and it is activated during the whole period of the service execution rather than to have a sequence, after using the microwave). The significance of the activation order appears in the behavior model, when we have to simulate the execution behavior of the system services. In the case where a component has alternative(s), this case is represented through the formulation of a relationship between the component module itself, to represent the fact that a component may have an alternative, alternatives, or composite alternatives (the case of a ceiling light in a living room for example, it would have a composite alternative which consists of the living room door and the ceiling light placed in the adjacent dining room, as illustrated in figure 3.5).

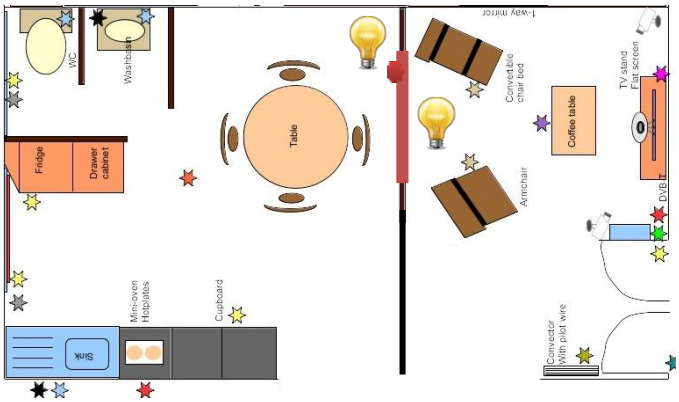


Figure 3.5. Example of composite alternative components

We chose to represent the system design model using an XML format conforming to the defined Meta-Model. Thus, the system is represented using an XML model to represent the system components and system services as well as the associations between the system elements. In this work, we modeled that part of the Meta-Model (figure 3.4) that is surrounded by dashed lines.

Using the XML modeling, each component in the global system is represented using an XML scheme with the basic parameters describing the component (such as the name, operations and location).

Each service is, also, represented by an XML scheme representing the service principal parameters, in addition to the list of components participating to the given service.

The components, in the service XML context, will have additional parameters that differentiate between different components (according to the service they belong to). Figure 3.6 illustrates an example of an XML model of a component (in the left) and a service (in the right).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>1</id>
    <name>Light1</name>
    <failure_probability>0.294</failure_probability>
  - <operation>
      <operation1>SwitchON</operation1>
      <operation2>SwitchOFF</operation2>
    </operation>
  - <location>
      <X_coordinate>3.0</X_coordinate>
      <Y_coordinate>2.5</Y_coordinate>
      <Z_coordinate>2.5</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
- <service>
    <id>1</id>
    <name>sitting_at_room_night</name>
    <context>sitting at room at night, cold weather</context>
    <time>Night</time>
    <environment>Cold</environment>
    <location>[3.0, 3.0, 3.0]</location>
    <user_location>[2.0, 2.0]</user_location>
    <component>&Light1</component>
    <component>&Heater</component>
</service>
```

Figure 3.6. XML model (a component and a service)

Additionally, figure 3.7 illustrates an extract of the XML scheme representing the component "Light1" model functioning in the service "sitting at room in the night".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component_service.dtd">
- <component>
    <id>1</id>
    <name>Light1</name>
    <priority>1</priority>
    <importance>2</importance>
    <operation>SwitchON</operation>
    <Component_context>lightening</Component_context>
    <service_Id>1</service_Id>
    <Alternative_Component>&Light2</Alternative_Component>
</component>
```

Figure 3.7. XML model for a component functioning in a given service

The component model represented in figure 3.6 (the left side) conforms to the component Meta-Model representation. In this model the main parameters that describe a given component are related to the component id, the component name, its probability of failure, the main operation(s) performed by the component and the location of the component in the living space.

In the same way, the service in modeled as illustrated in the right side of figure 3.6. This model conforms to the service Meta-Model representation, and it mainly has a set of parameters describe the service. These parameters are related to the service id, the service name, the service usage time, the service context, the location of demanded service and the corresponding principal components.

Whereas the model represented in figure 3.7 refers to the component functioning, and it conforms to the functionality notion in the Meta-Model. In this model the main parameters, which describe the functioning of a component in a given service, are related to the component id, the component name, the priority (order) of the component during the service, the component importance to the service execution, the component functional operation, the service id to whom the component belongs and the alternative component(s) that may substitute the given component in case of failure.

3.4. System's analysis modeling process

By the system analysis model, we aim at conducting an evaluation of the system design structure and at performing a fault tolerance mechanism, in order to ensure the continuity of system services. This mechanism is based on the introduction of alternative components (alternative scenarios) allowing to substitute the faulty component(s).

The system's analysis model is conducted using the Fault Tree Analysis approach, which is introduced in chapter 2. This tree structure is used as a decision aide to assess the service functioning state, depending on different scenarios.

In this section we introduce first the Meta-Model of the analysis phase, then the model which represents the tree structure of the Fault Tree Analysis tool, and finally we present the transformation rule conducted to make the transition from the design model-to-the analysis model.

3.4.1.The system analysis Meta-Model

The modeling approach (followed in building the design model) is assumed to facilitate and to make the creation of the analysis Meta-Model as simple as possible.

Following the MDA approach, we formalize the Meta-Model of the analysis phase, based on a static diagram representing the services and the components of the system in the tree structure.

In the analysis of the system, two main elements modules are identified and grouped together to determine the static relations between these elements.

As there is no standard Meta-Model for Fault Tree structure defined in advance, we built a fault tree Meta-Model for our transformation according to a tool called FaultCAT (Fault Tree Creation & Analysis Tool) [Zhao & Petriu, 2015]. The Meta-Model describes regular fault trees, which are fully supported by the FaultCAT tool used in this work.

Figure 3.8 illustrates the analysis Meta-Model, in which the association between the services and their components will facilitate the fault tree analysis structure of the system. This Meta-Model contains elements named FTelements, which can be either Events or Gates. Such an element can have children elements, for example an event can have a child gate, and a gate can have several children events. Every event has an attribute: the name of the event.

The Basic_Event, has also a probability attribute (Pf), which represents the component's probability of failure. In the Meta-Model, we represent a given service as the top event of the tree structure, and the components of the system as the basic events of the tree structure. The association

between the top event and the basic event is modeled as an intermediate event, and the gates represent the relationships between the events.

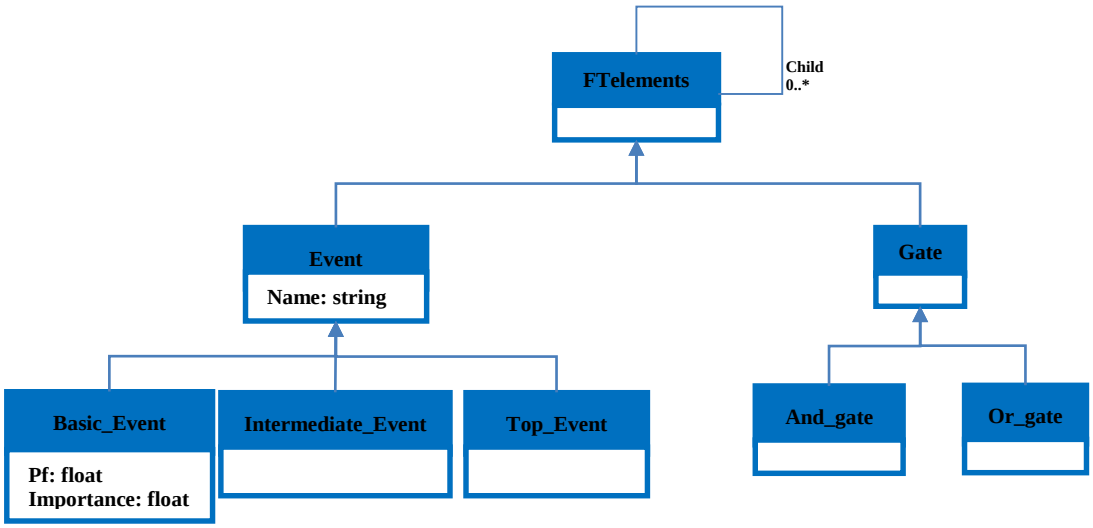


Figure 3.8. The Meta-Model of the Fault Tree Analysis

3.4.2.The system Fault Tree Analysis model

The system analysis model is based on the description of FaultCAT analysis tool. As we described the design model using XML schemes, we also represent the FTA model using XML schemes, which is converted to a graphical representation, as a tree structure, using FaultCAT.

From the definition of the system services and components, an algorithm is developed to generate an XML scheme. This XML scheme will represent the structure of each particular service, its components and alternative scenarios as a fault tree structure. To execute the transformation from the design to the analysis, an XML file will be generated as an input file to FaultCAT.

FaultCAT is an open source fault tree creation application developed in Java. A fault tree is a graphic help for analyzing a system for faults, and how these faults can affect other components in the system. It is designed to allow users combining several fault tree components into a single fault tree.

By attaching probabilities to certain components, a user can see how some parts of a system can affect other parts. There are also tools for leading further analysis on the created fault trees [FaultCAT, 2016].

Figure 3.9 illustrates an extract of the XML fault tree analysis model which conforms to the analysis Meta-Model. In this model, the top event of fault tree refers to the service to be analyzed. Between the And-Gate nodes, the basic events are modeled, which refer to the components activated by the given service. When an alternative component is found, it is modeled using an Or-Gate node to link it to the principal component. The fault tree analysis model conforms to the Meta-Model and it follows the XML scheme used in FaultCAT analysis tool.

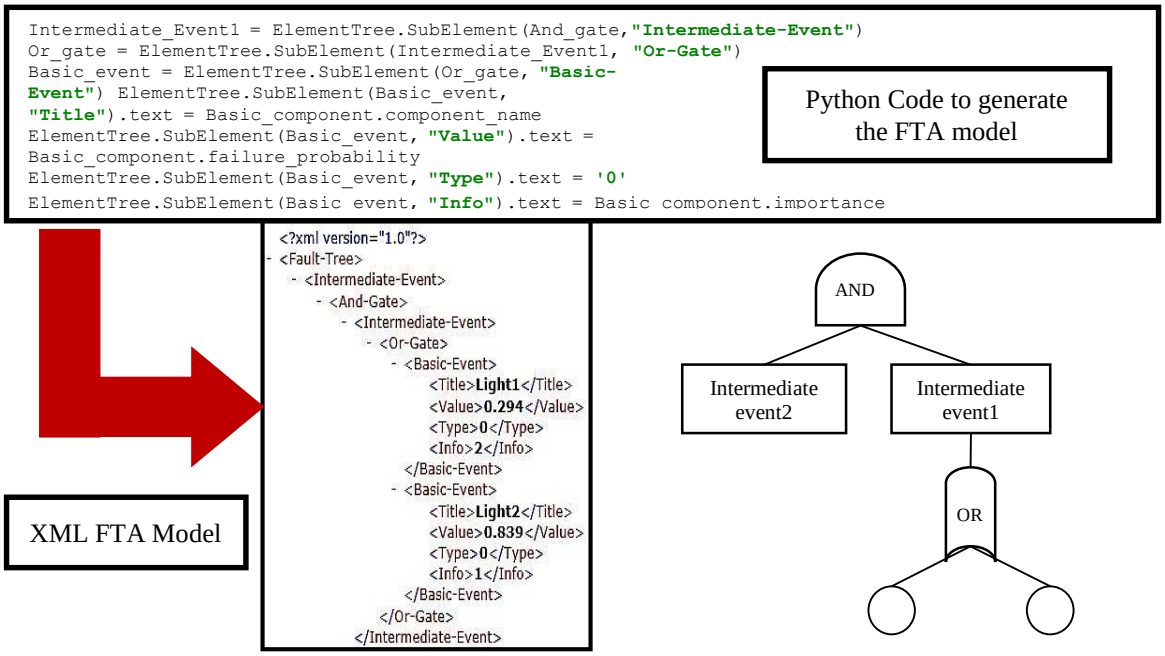


Figure 3.9. Extract of the generated FTA model, in XML format

By using the generated XML scheme for the Fault tree structure (as an input to the FaultCAT graphical tool), we can get the same structure of the fault tree, illustrated in chapter 2, as illustrated in figure 3.10. This tree structure diagram is the graphical representation of the FTA XML Model and conforms to the syntax represented by the analysis Meta-Model. In this figure, the XML scheme represents the Fault analysis model, and the tree structure (in the right) is the generated tree using FaultCAT tool.

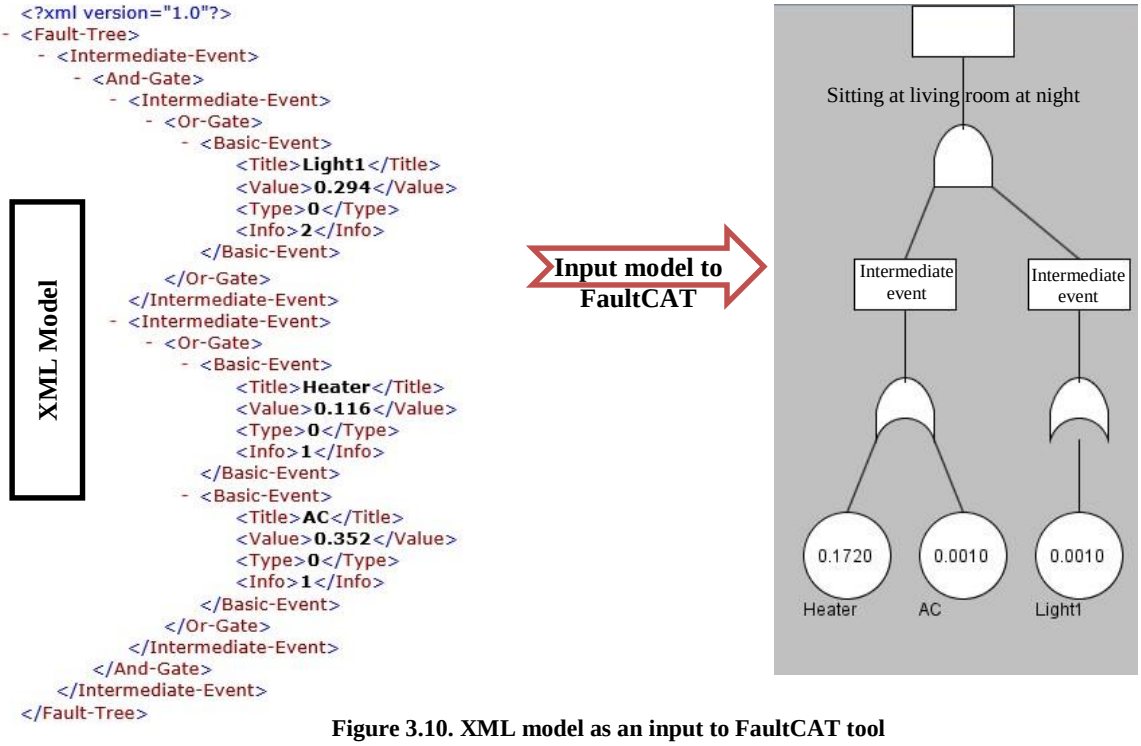


Figure 3.10. XML model as an input to FaultCAT tool

3.4.3.The Design model-to-FTA model transformation rule

The model transformation processes allow conducting transition between the different stages of the workflow and produce solutions conform to the predefined specifications.

Each transformation takes a model(s) as an input to the process and produces a model(s) as an output of the process. This approach, based on a hierarchical organization of the models and the applicable transformations, puts the model in the center of the expert designers concerns.

The model transformation is the core of the Model Driven Engineering software development methodology.

While this approach is widespread in the research community and begins to be widely used in the industry, there is no real standard for its implementation. There are different implementations proposals such as Atlas transformation language (ATL) [Jouault *et al*, 2006].

In this study, we proposed to realize the transformation rule, by developing a transformation rule algorithm using python framework, since the workflow models have been developed in Python framework.

To generate such a system analysis tool of services failure, a transformation tool has been developed to support this transformation from the system's design model to the fault analysis model which is based on the fault tree analysis approach FTA.

The transformation rule applied in the workflow, to achieve the transition from the design model-to- the analysis model, is illustrated in algorithm 3.1.

```
1.  Top_Event.name = Service.name
2.  Top_Event.Child = gate_and
3.  While (Component is not None):
4.      gate_and.And_gate.Child = intermediate
5.      intermediate.Intermediate_Event.name = Component.context
6.      intermediate.Child = gate_or
7.      gate_or. Or_gate.Child = basic1
8.      basic1.Basic_Event.name = Component.name
9.      basic1.basic_Event.Pf = Component.failure_probability
10.     basic1.basic_Event.Importance = Component. Importance
11.     if (Component.alternative is not None)
12.             gate_or. Or_gate.Child = basic2
13.             basic2.Basic_Event.name = Component.alternative.name
14.             basic2.basic_Event.Pf = Component. alternative .failure_probability
15.             basic2.basic_Event.Importance = Component. alternative. Importance
```

Algorithm 3.1. Transformation rule: Design Model-to- FTA Model

In this algorithm, the transformation rule matches the top event of the tree structure with the service element in line 1. Then, it establishes an and-gate (line 2) to link the top event to its children or basic events, which will represent the service's components. The loop (lines 3 -15) establishes the basic events and matches each basic event with a component from the design model. Each basic event is assigned with the probability of failure of the component, as well as the importance factor. The conditional loop (lines 11 – 15) checks if the component has alternative component, in order to establish basic event represents the alternative component, and linked with the principal component with an or-gate. The or-gate is established in line 6, and it will be linked to an intermediate event with the and-gate (to follow the tree graphical representation in the FaultCAT tool.

Considering the given service “sitting at the living room at night” which is composed of the two principal components, the “Light1” and the “Heater”. The component “Heater” has an alternative component “AC”. Figure 3.11 illustrates the application of the design model-to-FTA model transformation rule, given in the algorithm 3.1, to generate the FTA model of the service “sitting at the living room at night”, which conforms to the FTA Meta-Model, from the design model.

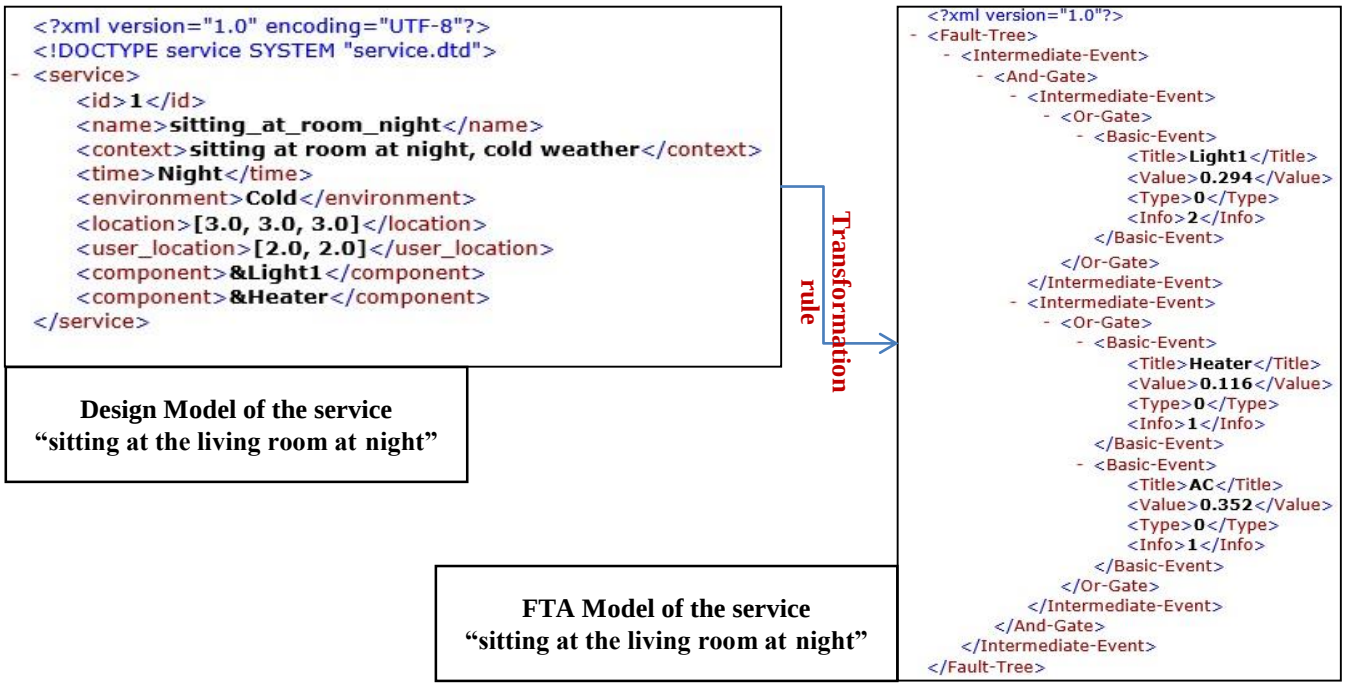


Figure 3.11. The application of design model-to-FTA model

After performing the analysis phase, a verification scenario is performed on the analysis model generated to check if the proposed system model is valid or if it needs to be revised.

The verification step is conducted by carrying out a model checking based on the generated analysis model of the FTA.

Model checking consists of verifying whether a model satisfies a given property (often expressed in temporal or modal logic). Model checking implies the fully automated property proving. When a property does not hold on a model, the user gets a counterexample. Model checking requires expressing models using formalisms.

Several possible approaches allow conducting this task [Hostettler *et al*, 2011].

Figure 3.12 illustrates an extract of the model checker of the generated FTA model. This model checker, developed in Python, performs a mathematical computation of the service global decision indicator, based on the information obtained about the failures probability of the corresponding components and the importance factors appended to each component.

The main functionality in this algorithm lies in lines 31-45, where the decision function is calculated, based on the calculation of the basic events probability of failure and the importance factor, to study the influence of the basic event on the top event (the service functioning state).

The computational equations are developed in the next chapter. Whereas, a detailed verification approach is proposed in chapter 5.

```
Pf_X_I = 0
and_Importance_Indicator = 1
QTE = 0
length = 0
or_Importance_Indicator = []
Quality_of_Success = 0

for intermediate_event1 in root:
    for And_gate in intermediate_event1:
        for intermediate_event2 in And_gate:
            for Or_gate in intermediate_event2:
                product_Pf_X_I = 1
                j=0
                i= i + 1
                print "Internediate Event:" + str(i)
                for Basic_event in Or_gate.findall("Basic-Event"):
                    j = j + 1
                    print "Basic Event:" + str(j)
                    print Basic_event.find("Title").text
                    print "Probability of failure for Basic Event " + Basic_event.find("Title").text + " = " + Basic_event.find("Value").text
                    print "Importance Factor for Basic Event " + Basic_event.find("Title").text + " = " + Basic_event.find("Info").text
                    Pf_X_I = float(Basic_event.find("Value").text) * float(Basic_event.find("Info").text)
                    print "Pf*I for Basic Event " + Basic_event.find("Title").text + " = " + str(float(Basic_event.find("Value").text) * float(Bas
                    product_Pf_X_I = product_Pf_X_I * Pf_X_I
                    print product_Pf_X_I
                or_Importance_Indicator.append(round(product_Pf_X_I, 4))
                print or_Importance_Indicator
while length < len(or_Importance_Indicator):
    and_Importance_Indicator = round(and_Importance_Indicator, 4) * (1 - round(or_Importance_Indicator[length],4))
    print and_Importance_Indicator
    length = length + 1
QTE = 1 - round(and_Importance_Indicator, 4)
Quality_of_Success = 1 / QTE

print "THE GLOBAL DECISION INDICATOR Q(TE) FOR THE SERVICE is: " + str(QTE)
print "THE QUALITY OF SUCCESS OF THE SERVICE is:" + str(round(Quality_of_Success, 4))
```

Figure 3.12. Extract of the FTA model Checker

3.5. System's behavior

Modeling the system and building its design helps the designer to understand how the system will work [Few, 1996]. The system's behavior introduces a conceptual model building, constructing a system's diagram to describe how the designer thinks the system will behave.

In the system behavior process, we aim at performing an examination of the system, by examining its services functioning states. By examining the service state, we mean to realize an assessment of the deliverance issue of the service, to the end user, following the predefining scenarios of service execution.

When the service is defined, and coupled with the corresponding components, as well as the alternative components (scenarios), we need to verify that this given service reaches the delivering issue, if it is executed, whatever the scenario it follows.

For this reason, we need to establish a relative representation of the service accomplishment trajectory, from the starting point, through each included component and alternative scenarios, to the deliverance point.

The best tool to build a conceptual model, representing and evaluating the success functioning of the system states, is based on the use of Petri Nets modeling language allowing to simulate how each system service will be executed through its constituting components (with the alternative components).

A Petri net is one of several mathematical modeling languages for the description of distributed systems behavior. A Petri net is defined as a directed bipartite graph in which the nodes represent the transitions (i.e. events that may occur, signified by bars) and places (i.e. conditions, signified by circles, representing the availability of the component during the service deliverance). The directed arcs describe which places are pre- and/or post-conditions for which transitions (signified by arrows) [Desel & Juhás, 2001].

As a mathematical model the Petri nets are commonly used to represent discrete event systems. Each space is associated with the activation of a home automation service discrete event (i.e. basic services or scenarios). Transitions, which must be defined between each place, may be of different types: a transition vector (Basic), a start marker (start) or an end marker (End).

Arcs allow linking places to transitions and vice versa. The designer can choose a type of arc following the behavior result (true, false, or unknown) of each service and, thereby, define alternative components to ensure the operational safety of the considered service. The resulting graph can then be interpreted at runtime by considering the possible alternative paths (as well as activation of alternative components) [Allègre, 2012]. Figure 3.13 illustrates an example of the use of a Petri net to show the behavior of the system (the example of a scenario of the lightening service, as the modeling language proposed by Allègre to represent the scenario, without tokens).

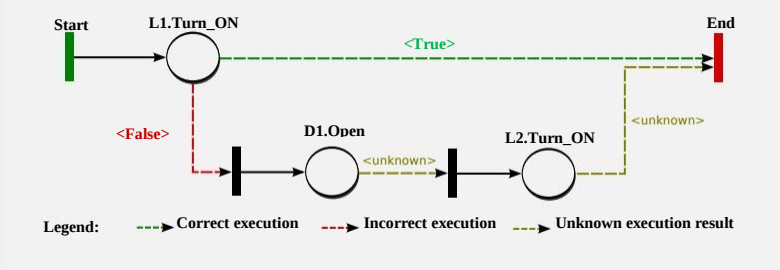


Figure 3.13. Example of the service behavior using petri net – alike [Allègre, 2012]

3.5.1.The system behavior Meta-Model

The modeling manner followed in building the design model, will facilitate and make the creation of the system behavior Meta-Model simple.

Following the MDA approach, we formalize the Meta-Model of the behavior process, based on a static diagram representing the services and the components of the system in the PetriNet structure.

Thus, in the behavior simulation of the system's services, two main elements modules are identified and grouped together to determine the static relations between these elements.

Figure 3.14 illustrates the system behavior Meta-Model, in which the association between the services and their components will facilitate the behavior simulation structure. This Meta-Model contains mainly a module named Place [Luis & Matteo, 2008], which will represent the service starting point, as well as the corresponding components (and alternatives). The transitions in the PetriNet structure are represented as Transition Module. Whereas two similar modules, named InputArc and OutputArc, will represent the main Arcs connecting the transition with the places.

Contrary to the analysis model, we are only here interested to include a feature representing the name of the service and the name of the corresponding components.

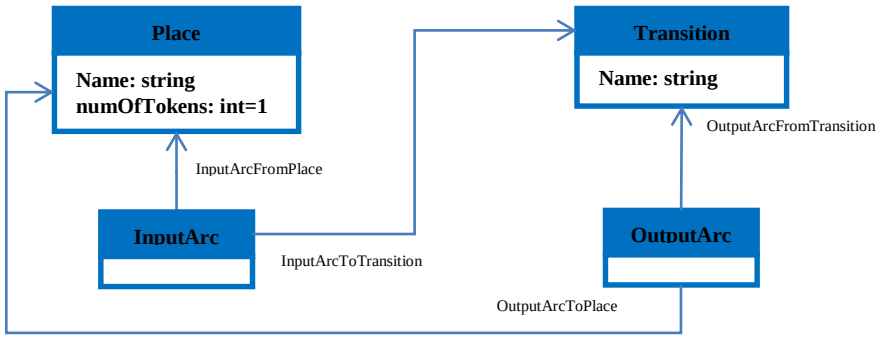


Figure 3.14. The Meta-Model of the system behavior

3.5.2.The system behavior model

The system behavior model is based on the general description of PetriNet structure. As we have described the design model, as well as the fault tree analysis model, using XML schemes, we also represent the behavior PetriNet model using XML schemes (which are converted to a graphical representation) as a tree structure, using a tool called PetriNetSim.

From the definition of system services and components, an algorithm is developed to generate an XML scheme. This XML scheme will represent the behavior of the system when performing each particular service as a Petri net modeling scheme.

To execute the transformation from the design to the behavior of the system, an XML file is generated as an input file to a tool called PetriNetSim. PetriNetSim is an open source tool developed in JAVA to design and simulate Colored simple/colored/timed Petri Nets in Java programming language.

Figure 3.15 illustrates an extract of the algorithm to generate the Behavior model which conforms to the Behavior Meta-Model.

Figure 3.15. Extract of the generated Behavior model, in XML format

In this model, the beginning place is reserved for the service to be simulated, as the start place of the Net. The beginning place is assigned by the name of the service. The transitions are represented also in the model, and the input arcs, as well as the output arcs are modeled to link the places and the transitions of the Net. Each place in the Net is assigned by the name of the component, corresponding to the given service. And for each principal component, an alternative place, is represented and assigned to an alternative component. In case where the activation order of a component equals to 0, a place in parallel is established to represent the component (which is being activated during the whole period of the service).

The transitions, in this model, are holding a Boolean value (i.e. true or false) related to the firing action of the preceding place (component). Thus, in the case of a component failure, represented in the preceding place, the next transition will have a false value which will force the transition not to fire. Thus, the alternative transition to the alternative component, represented by the alternative place, will have a true value which will activate the alternative component. This corresponds to the situation when a component represented in the place P_1, has an alternative component represented in the place P_2, and is connected to two transitions T_1 and T_2. In one time unit, T_1 will hold a true value (false value) and T_2 will hold a false value (true value). This simple example is illustrated in figure 3.16. Obviously, at the beginning of the behavior simulation, all the principal transitions (linked to principal places) held the true value, whereas the alternative transitions held the false value.

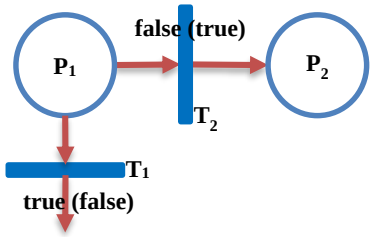


Figure 3.16. AN example of the behavior model

Since the presentation of the components failure is a run time aspect (rather than a design facet), thus, we need another proposition to represent the failure influence on the service behavior (during the design process). In other words, we need to represent the impact of the components' probability of failure on the simulation of the service behavior. For this reason, we may propose the use of an advanced form of Petri Nets to analyze the behavior of the service execution with the consideration of the components' probability of failure. A stochastic Petri net is a form of Petri nets where the transitions fire after a probabilistic delay determined by a random variable [Marsan, 1989]. These random firing delays (associated with the transitions) could be linked to the components probability of failure (those represented by the preceding places).

The firing delay of a transition is associated with each transition. It specifies the amount of time that must elapse before the transition can fire. Notice that this firing delay is a random variable with negative exponential probability distribution function. We can link this firing delay to each component's probability of failure simply by noticing that if the probability of failure is increased then, it will increase the firing delay. In this case, we can assume that this transition will not fire and the component may fail. Therefore, in this case we propose to fire the alternative transition.

For instance, in figure 3.17, we are analyzing the behavior of a given service at the point of a component C_1 given by the place P_1, which has an alternative component $C`_1$ given by the place $P`_1$. The transition T_1 is connecting P_1 with P_2, and it holds a firing delay α, where $\alpha = P^f(C_1)$. The greater the value of α, the more the firing delay for the transition T_1 (is increased) which may lead to the non-fire of the place P_1 to P_2. Thus, we assumes the firing of the alternative transition $T`_1$, which will activate the component $C`_1$ given by the place $P`_1$.

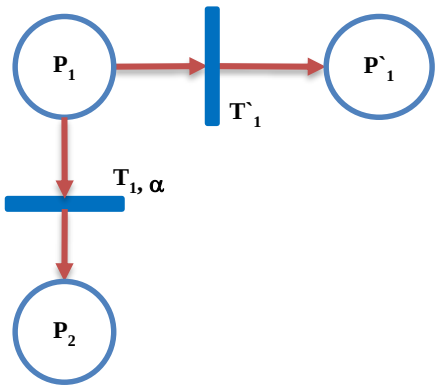


Figure 3.17. Stochastic Petri Nets example

An End place is modeled as the end of the simulated service, to check if the Net reaches this place (i.e. the service reaches the deliverance issue) following the proposed execution scenarios.

By using the generated XML scheme for the behavior model as an input to the PetriNetSim tool, we can get the Petri Net structure, as illustrated in figure 3.18. This Petri Net structure diagram is the graphical representation of the Behavior Model and conforms to the syntax represented by the Behavior Meta-Model. In this figure, the XML scheme represents the behavior model, and the Petri Net structure (in the right) is the generated Petri Net using PetriNetSim tool, which graphically represents the behavior model.

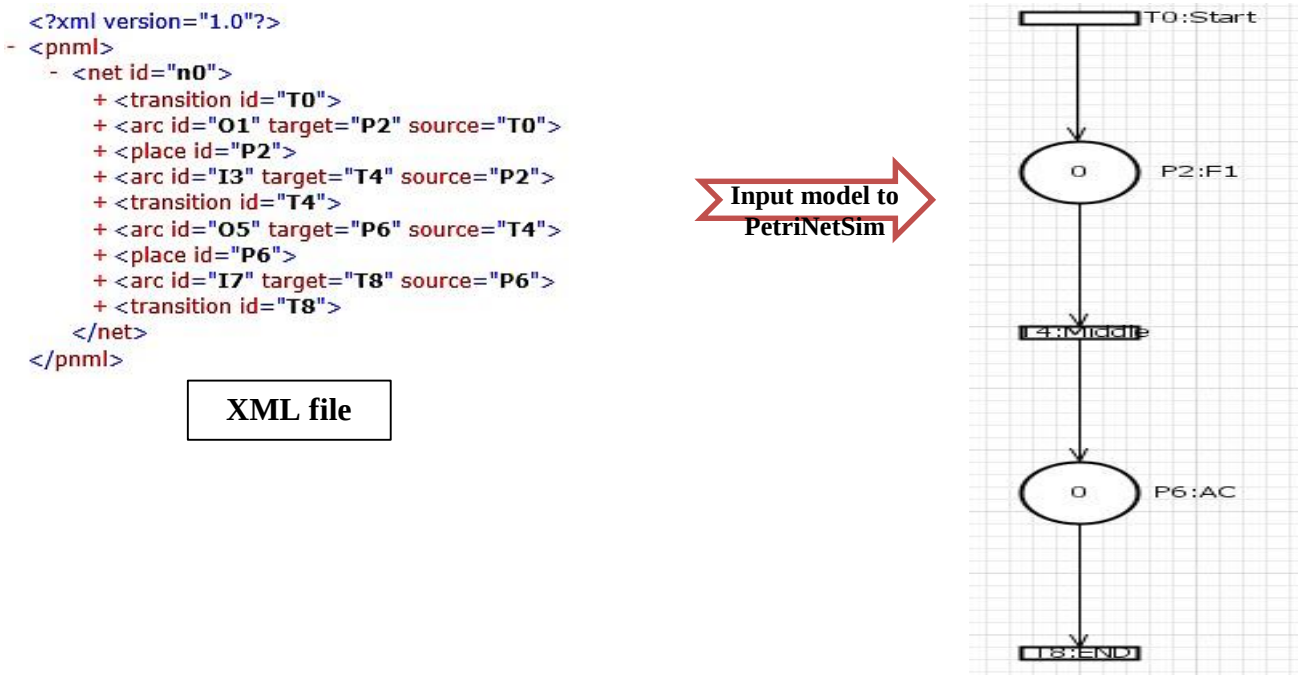


Figure 3.18. Extraction of system's behavior model in PetriNetSim

3.5.3.The Design model-to-Behavior model transformation rule

In the same manner followed in the analysis phase, a transformation process has been developed to generate a simulation of the system's behavior. This transformation intends to generate the behavior of the system starting from the system's design (after having conducted the analysis of the system's design), in order to present how the chosen service scenario will be executed.

In the case where a better scenario exists, the system will select it (based on the analysis of that scenario). The system behavior, as a result from system design and analysis, will give the opportunity to evaluate the system performance and, thus, to evaluate the design of the system.

To generate such a system behavior simulation tool of system functioning, a transformation tool has been developed to support this transformation from the system's design model to the behavior model which is based on the Petri Net.

The transformation process (from the system's design to the system's behavior) will be performed using the Python framework in order to achieve the transformation and to do a simulation of the system's behavior.

The transformation rule applied in the workflow, to achieve the transition from the design model-to- the behavior model, is illustrated in algorithm 3.2.

Considering the previous service "sitting at the living room at night" which is illustrated previously in figure 3.11. Figure 3.19 illustrates the application of the design model-to-behavior model transformation rule, given in the algorithm 3.2, to generate the simulation behavior model of the service "sitting at the living room at night", which conforms to the behavior Meta-Model, from the design model.

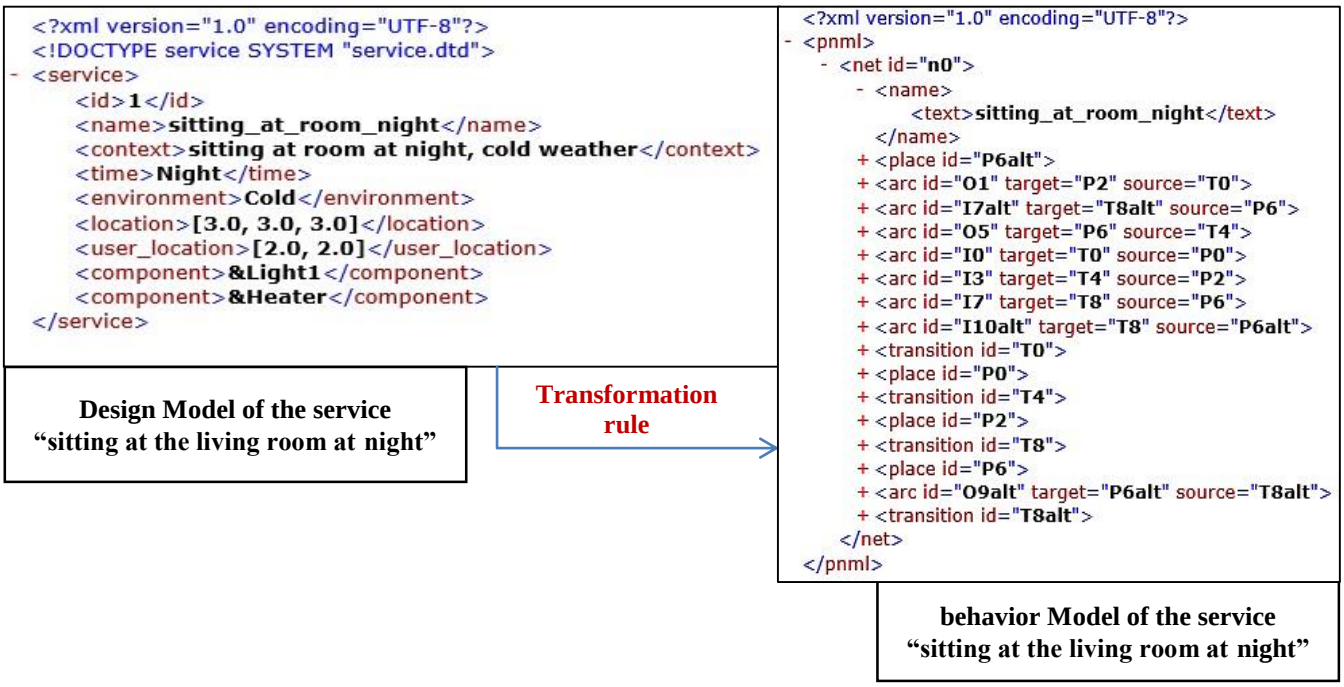


Figure 3.19. The application of design model-to-behavior model

```
1.  Start_place = Place
2.  Start_place.name = Service.name+ "Start"
3.  Strat_place.InputArcFromPlace = InputArc
4.  InputArc. InputArcToTransition = Start_transition
5.  Start_transition. Name = "t0"
6.  Start_transition. OutputArcFromTransition = OutputArc
7.  OutputArc .OutputArcToPlace = Component_place
8.  t =1
9.  While (Component is not None):
10.     Component_Place.name = Component.operation
11.     if Component.order == 0):
12.             Component _Place.InputArcFromPlace = InputArc
13.             InputArc. InputArcToTransition = transition
14.             transition. Name = t
15.             transition. OutputArcFromTransition = OutputArc
16.             OutputArc .OutputArcToPlace = END_place
17.     Component _Place.InputArcFromPlace = InputArc
18.     InputArc. InputArcToTransition = transition
19.     transition. Name = t
20.     transition. OutputArcFromTransition = OutputArc
21.     OutputArc .OutputArcToPlace = Component_place
22.     A_t = t + 4
23.     if (Component.alternative is not None)
24.             Component_Place.InputArcFromPlace = InputArc
25.             InputArc. InputArcToTransition = transition
26.             transition. Name = A_t
27.             transition. OutputArcFromTransition = OutputArc
28.             OutputArc .OutputArcToPlace = Alternative_Component_place
29.             Alternative_Component_Place.name = Component. alternative .operation
30.             Alternative_Component _Place.InputArcFromPlace = InputArc
31.             InputArc. InputArcToTransition = transition
32.             A_t = A_t+1
33.             transition. Name = A_t
34.             transition. OutputArcFromTransition = OutputArc
35.             OutputArc .OutputArcToPlace = Component_place
36.     else:
37.             Component_Place.InputArcFromPlace = InputArc
38.             InputArc. InputArcToTransition = transition
39.             transition. Name = A_t
40.             transition. OutputArcFromTransition = OutputArc
41.             OutputArc .OutputArcToPlace = Failed_place
42              Failed_place.name="Fail:
43.     t = t+1
44. END_place = Place
45. END_place.name = "END"
```

Algorithm 3.2. Transformation rule: Design Model-to- behavior Model

In this algorithm, the transformation rule matches the start place of the PetriNet structure with the service name, as the start of the behavior simulation (lines 1-2). The lines 3-7 establish a sequence of arcs/transition to link the service place to the first component place. Then, the lines 9-43 generate each place in the Net structure, which refers to the corresponding components of the design model. Each place is assigned by the name of the corresponding component. In between the places, a sequence of arcs and transitions are characterized to link the places. The lines 23-35 represent, in the same manner,

the alternative components of the principal components, as alternative places linked to the principal places in the Net. Whereas, the lines 11-16 check the parallelism feature of a given component, in order to be represented in the Petri Net as a parallel place. In case the principal component doesn't have alternative component, the lines 36-42 link the component place to a place refers to a "Fail", to represent the deadlock and the failure of the service. The whole elements in the generated Petri Net are given a sequential number. In the end of the algorithm, the line 38 generates a place "End" to finalize the Network, in which we will check the reachability of the model (i.e. to check that the service reaches the deliverance issue).

As a result, by following the workflow steps from the system's design to the system's behavior, the system development life cycle is partially performed. This will help enforcing the well- definition of the system's efficiency and soundness.

After performing the behavior phase, a verification scenario is performed on the behavior model generated to check if the proposed system model is valid or if it needs to be revised.

The verification step is conducted by carrying out a model checking based on the generated behavior model of the Petri Net.

Model checking is performed on the system's behavior in order to confirm the system model viability. Since the system behavior is modeled using a Petri net, model checking is assumed to verify the reachability of the model using reachability graph.

Reachability refers to the ability to get from one place to another within a Petri net. We say that a marking $M(P_1, P_2)$ exists if there exists a sequence of adjacent places (i.e. a path) starting from P_1 and ending at P_2 [Darondeau *et al*, 2012]. For the system structural model, we aim at check the behavior model by analyzing the reachability graph to check if the place "Start", which refers to the service, can reach the place "End", which refers to the good functioning of the service following the behavior of different scenarios represented by the Petri Net model. In this work, the reachability graph is analyzed using TINA (TIme Petri Net Analyzer) [TINA]. TINA is a set of toolboxes responsible for the editing and analysis of Petri Nets. TINA can construct and check reachability graphs from nets described in textual or graphical form, produces transition systems abstracting their behavior in human readable form for available model checkers and equivalence checkers [Berthomieu et al, 2004]. Moreover, TINA preforms a Stepper simulator allowing interactive simulating and the net model in step-by-step. In chapter 5, a detailed verification approach is proposed to verify the system's FTA modal, as well as the system's behavior model.

3.6. Illustrative example

In order to illustrate the proposed approach, let us consider the following simple example, illustrated in figure 3.20:

Service sitting in living room at night during winter time requires the use of the following two components: Light1 and Heater.

In this example, the heater component has an alternative component AC.

By demanding this service, the system will heat up the heater first (before a period of time of the entrance of the user to the living room) and, then, the main light Light1.

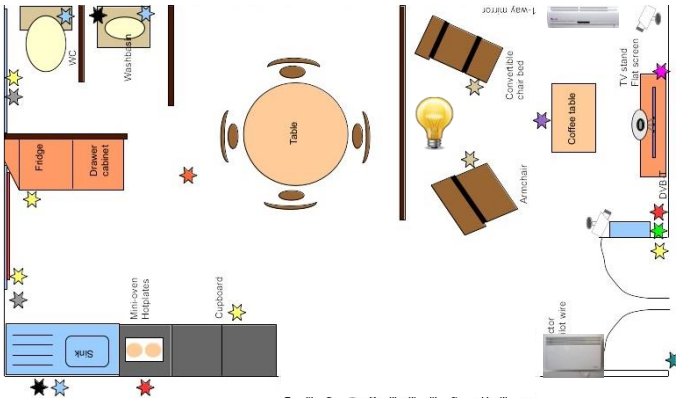


Figure 3.20. Illustrative example

In the case of failure in the heating system, the system will switch on the AC instead.

Following the workflow process phases, the designer starts by defining the main components of the XML models (which are contributing into this particular service, Light1, Heater and AC), as illustrated in figure 3.21.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>3</id>
    <name>Heater</name>
    <failure_probability>0.116</failure_probability>
  - <operation>
      <operation5>ON</operation5>
      <operation6>OFF</operation6>
    </operation>
  - <location>
      <X_coordinate>2.5</X_coordinate>
      <Y_coordinate>2.5</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

Heater

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>1</id>
    <name>Light1</name>
    <failure_probability>0.294</failure_probability>
  - <operation>
      <operation1>SwitchON</operation1>
      <operation2>SwitchOFF</operation2>
    </operation>
  - <location>
      <X_coordinate>3.0</X_coordinate>
      <Y_coordinate>2.5</Y_coordinate>
      <Z_coordinate>2.5</Z_coordinate>
    </location>
</component>
```

Light1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>4</id>
    <name>AC</name>
    <failure_probability>0.352</failure_probability>
  - <operation>
      <operation7>ON</operation7>
      <operation8>OFF</operation8>
      <operation9>VARIANT</operation9>
    </operation>
  - <location>
      <X_coordinate>6.0</X_coordinate>
      <Y_coordinate>6.0</Y_coordinate>
      <Z_coordinate>2.5</Z_coordinate>
    </location>
</component>
```

AC

Figure 3.21. Contributing components XML model

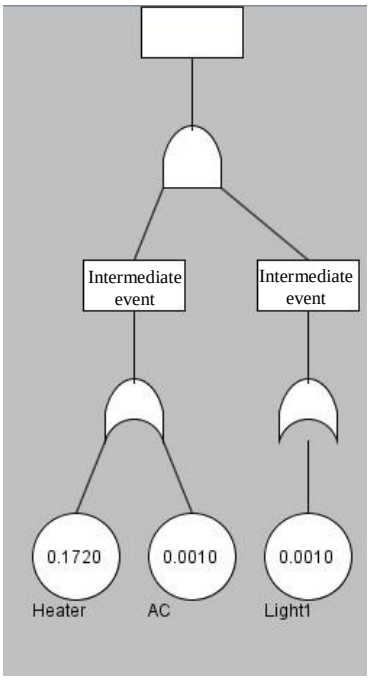
The service's XML model is defined with the related components in the next step as it is illustrated in figure 3.22. In this figure, the service "*sitting in living room at night*" model defines the service parameters. These parameters are the service id, the service name, the context of the service, the location of the service and the component called by the service, light1 and heater.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
- <service>
    <id>1</id>
    <name>sitting_at_room_night</name>
    <context>sitting at room at night, cold weather</context>
    <time>Night</time>
    <environment>Cold</environment>
    <location>[3.0, 3.0, 3.0]</location>
    <user_location>[2.0, 2.0]</user_location>
    <component>&Light1</component>
    <component>&Heater</component>
</service>
```

Figure 3.22. XML model of the service

The next step consists on performing the analysis on the proposed service, using the FTA approach. As previously mentioned, this would be done using the FaultCAT tool.

The generated FTA with a model checker of the analysis step is illustrated in figure 3.23. The components failure probabilities are shown in the bottom of the left figure. The results of the model checker are shown to the right. For each component, the analysis phase calculates the relative computation of the probability of failure with the importance factor. For the top event (the service) it calculates the relative global decision indicator *Q(TE)*. Thus, the cost function for the service is estimated to decide whether the selected scenario is viable to follow or we can select another scenario to follow.



Basic Event 1, Heater
Probability of failure for Basic Event F1 = 0.172
Importance Factor for Basic Event F1 = 2
Pf*I for Basic Event F1 = 0.344

Basic Event 2, AC
Probability of failure for Basic Event AC = 0.001
Importance Factor for Basic Event AC = 1
Pf*I for Basic Event AC = 0.001

Basic Event 1, Light1
Probability of failure for Basic Event Light1 = 0.001
Importance Factor for Basic Event Light1 = 1
Pf*I for Basic Event Light1 = 0.001

THE GLOBAL DECISION INDICATOR Q(TE) FOR THE SERVICE is: 0.0013
THE COST FUNCTION OF THE SERVICE is: 769.2308

Figure 3.23. FTA and model checker

The system's design, as a last step, is simulated using the system's behavior model, by generating a Petri Net model illustrating how the service could be executed, as shown in figure 3.24.

In this figure, the first place P0 refers to the beginning of the service and the name of the place is the name of the service. The places, P2 and P3, represent the principal components light1 and heater. Whereas, the place P4 refers to the alternative component AC for the principal component heater. At the end, the place P5 represents the place End, which we aim at analyze the reachability of the network, to ensure that the service is executed, following the defined scenarios.

In fact, this Petri Net graph is generated by the behavior model, which in turn is generated by applying the transformation rule from the design model.

The behavior model is also evaluated using the TINA toolbox to check the reachability graph of the service scenario.

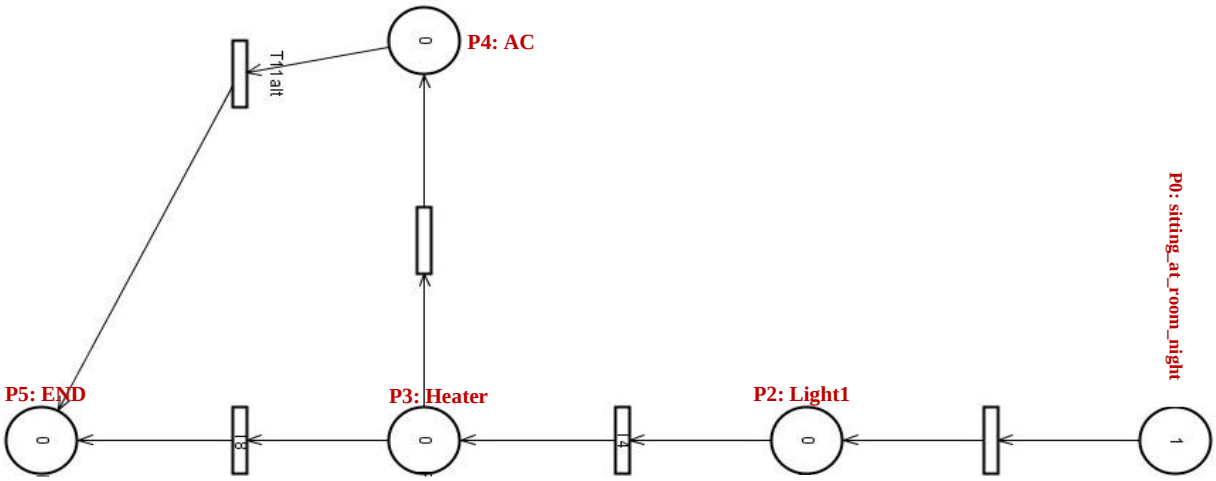


Figure 3.24. System behavior model using Petri Net

Moreover, a detailed verification process of the workflow is presented in chapter 5, to verify that we built the workflow, thus, the system in the right and correct way.

3.7. Conclusions

System's design and analysis is very crucial; most systems fail because of the shortcomings in this phase. With system's design, the designer will be able to formalize and to recognize how the system elements are structured and associated with each other within the system and; how they interact with the entities outside, as well as inside, the system. System's design will take the user's requirements into consideration and comes out with a high level and low level design that will form the blue print to the actual solution corresponding to the user's needs in hand.

Complex repairable systems, as home automation systems, are being increasingly witnessed, and issues related to analytical system reliability and availability solutions are well known. To overcome this intractability, the analysis of the system is an important step to always check out the best design of the system that keen with the reliability and the availability of system entities.

In the system's development life cycle, partial phases are proposed by the workflow. The goal is to design a system that describes the entities of the system as a set of services introduced to the final users.

Differentiation between services is done by establishing a context for each service. Components forming the services are also characterized using the component's context.

Component's context should be consistent with the service context in order to make the discrimination between different components, as well to be able to use the same components in different services (but for diverse purposes).

To perform the analysis of any proposed system's services structure, an analysis approach is developed allowing to analyze and to check if the proposed service scenario maintains the full availability of the system. System's analysis is doing revision and enhancement of the system's design, to always propose a sound system through its structural design.

System's behavior is proposed also as a phase to visualize how the services would be executed with selected reliable components and alternatives.

Transformation approaches and tools are used to perform the conversion from a development phase to another. Transformations are based on the use of XML based schemes generated by the Python language framework, in order to generate structures assumed to be compatible with other tools used in the transformation tasks.

Model checkers are performed to evaluate the system model viability. The evaluation of the analysis approach is performed using Python tool, and, the system's behavior model checking is realized using TINA toolbox.

In this chapter, a design flow based on Model Driven Architecture approach was presented. Three output blocks are generated leading to:

- A model representing the system structural elements (components/services), with Meta-Model conforms to the modeling;
- An FTA to analyze a service's continuity and
- A system behavior allowing simulating the execution of a scenario.

These blocks are interconnected and form the model checker of the design.

The proposed approach allows the expert designer to design each service that needs to be provided to the final users. Moreover, the resulting workflow allows the designer to realize offline analyses of the predefined scenarios for each service execution. When the designer analyzes the design model, he ensures that each service will be delivered in the run time, based on the verified and well-designed predefined alternative scenarios.

The analysis part of the workflow is based on the use of the Fault Tree Analysis approach. In the next chapter, a proposition for the integration of a risk indicator into the classical Bayesian FTA is conducted. This proposition allows analyzing each service in the system based on injecting the influence of component's importance factors, as well as the probability of failure of each component, into the success / failure of the service execution.

The developed workflow structure facilitates the design process of the system structural elements and reduces the time needed to fetch and tolerate potential failures. As a result, this design workflow ensures the continuity of the system and meets the requirement for a system reconfigurable towards the user's expectations.

System design, analysis and behavior workflow

System design, analysis and behavior workflow

Chapter 4.

Proposition of a Bayesian-based fault analysis approach

In the previous chapter, the workflow of the system design is proposed by describing the system structure, based on the abstract formalization of system elements (proposed in chapter 2). In order to evaluate the continuity of system's services, transformations rules from the system design model into the fault tree analysis model and to Petri Net model of fault tolerance and behavioral analysis were then applied.

In chapter 2, a critical analysis of existing solutions dealing with the service failure issue has been proposed. Moreover, the advantages and disadvantages (or weaknesses) of each approach are expressed. Several approaches have been proposed to formalize fault tolerance concepts allowing, thus, the representation of the failure of a component, and in turn, the failure of the related service.

This chapter is devoted to introduce a new proposition based on the joint use of the two notions (presented in chapter 2): the importance factors and the probabilities of failure (in order to study the relative influence of these two features in the functioning state of a given event). This proposition intends to employ the importance factor, mentioned in chapter 2, within the fault tolerance mechanism, as a bridge to the main contribution of this chapter.

The key objective of the proposed contribution is to tackle the joint use of the importance factors and the probabilities of failure in the literature and ongoing activities, as well as the imprecise, even the lack, of available data about the failure occurrence.

A recall of reviewed fault analysis research activities is introduced in this chapter, in order to come to the point of the contribution towards the objective (section 4.1). From the conducted review of fault analysis approaches, Fault Tree Analysis (FTA) approach is suggested to be used. Thus, FTA structure is recalled.

The application of new fault analysis approach, based on FTA and importance factors, is proposed (section 4.2), coupled with an illustrative example (section 4.3), with a close technical analysis of the proposed model results.

4.1. Proposition of a Bayesian-based fault analysis approach

The key objective of the proposed approach is issued from the need of the joint use of the importance factors and the probabilities of good functioning, to make full advantage of these two notions allowing, thus, to the expert designer to integrate his own appreciations of the importance factors of different components into the Bayesian fault tree analysis approach.

In this chapter, the objective is to present and to analyze a proposition allowing evaluating the success of services related to Ambient Assisted Living Systems. This proposition is based on the use of Fault Tree Analysis (FTA) approach of a service failure depending on two factors:

- A failure measure of different service components, and,
- A user defined degree of importance (that each component holds for the success of the corresponding service).

In conventional FTA based on Bayesian concepts, basic events (either they are basic components (light, door, TV…) or composite components seen as one component (door and light, window and window Shutter …)), are described by their associated "crisp" failure/success probabilities. Therefore,

quantitative analysis, in the framework of FTA, is mainly based on the evaluation of the fault/success probability of the top event in which the probability of failure/success of each of its contributing sub event is already known. In [Aven, 2011], these probabilities are interpreted either as a relative frequency based on the fraction of times the failure occurs and repeats; or as a subjective measure of uncertainty, conditional on the background knowledge (the Bayesian perspective). The majority of related research activities consider that the failure probabilities are "a priori" given. Using classical probabilistic computations (initiated in the systems reliability domain), allows direct evaluation of failure probability of an output event from two or more independent input events, as well as dependent input events where success/failure probabilities should be computed as being conditional probabilities, combined by a Boolean function (Gate AND / OR in figure 4.1).

Boolean AND gate Boolean OR gate

Figure 4. 1. Boolean gates

In fact, using two independent elementary constituting events for which the failure probabilities are denoted as $P^F(1)$ and $P^F(2)$, the resulting top event failure probability, PF, is simply obtained as follows:

- Using the Boolean AND function, the top event success is reached when ALL elementary constituting events have success. This implies that the failure probability can be simply obtained as follows:

$$\mathbf{P^F = 1 - [1 - P^F(1)] \, . \, [1 - P^F(2)]} \quad (1)$$

- Using the Boolean OR gate, the top event failure is reached when ALL elementary constituting events have failure. This immediately leads to the following failure probability:

$$\mathbf{P^F = P^F(1) \, . \, P^F(2)} \quad (2)$$

These simple computations can be generalized, straightforward, when more than two elementary constituting events are used.

A major difficulty encountered in FTA for the evaluation of the top event failure probability is related to the fact that precise and reliable failure probabilities of the elementary constituting events are difficult to obtain. Sometimes, these probabilities are not available, or their estimation is based on the use of a statistically small size sample base (leading to imprecisely or unreliable estimated probabilities).

Another important lack of FTA is related to the fact that resuming all the knowledge about different events functioning by a single measure, i.e. failure probability, does not reveal, by itself, all informative facets related to these events for the global top event (designate a service in a system) good functioning. This fact has already been pointed in several studies [Ericson, 2005; Borgonovo, 2007; Cheok *et al*, 1998] all reaching the same conclusion:

"Probability of failure is a good indicator but it does not indicate how far an event or a component contributes to the system failure in FTA".

The next section is dedicated to present a proposed approach allowing dealing with both failure probabilities as well as the expert-defined importance factors in the very same framework.

4.2. Proposed approach based on FTA

Following the FTA state of the art analysis, conducted previously, it was clearly shown that most efforts focus on imprecise failure probabilities and nearly most of these studies share the same conclusion: the use of fuzzy and possibility theories concepts seems to be the most adequate for modeling and processing imprecise failure probabilities. Nevertheless, the following important question is naturally raised: Does a Top event's probability of failure reveals all informative facets characterizing the good functioning and the long term service delivery? In fact, the probability of failure, by itself, does not indicate the importance functioning state of different components that compose the service. Generally, most research activities do not consider the relative importance of basic events (i.e. constituting components) for the good functioning of the global service.

To clarify the concept of importance indicators, let us consider the example illustrated in figure 4.2. In this example, seven elementary events are used in order to deliver two top events (i.e. two services). The particularity of this example is positioned on the fourth basic component (BE_4) assumed to participate into the good functioning of the two top events. In fact, considering the user subjective appreciation, BE_4 has a relative importance I_1 for the functioning of TE_1 (considered separately), and a relative importance I_2 for the functioning of the second service TE_2 (considered separately). Noticing that BE_4 is shared by the two services, then from a global system point of view, BE_4 has a global importance higher than both elementary ones (since, if it fails, then both services will fail).

If, for instance, BE_4 refers to an energy supply engine, then its importance is crucial for the global system, whereas, if BE_4 refers to a secondary lightening component, its importance is not as much crucial. This clearly shows that the importance factor of each component obeys a subjective user/expert-based appreciation. Also, it is worthwhile to notice that the importance factor reflects a different informative facet than the failure probability measure.

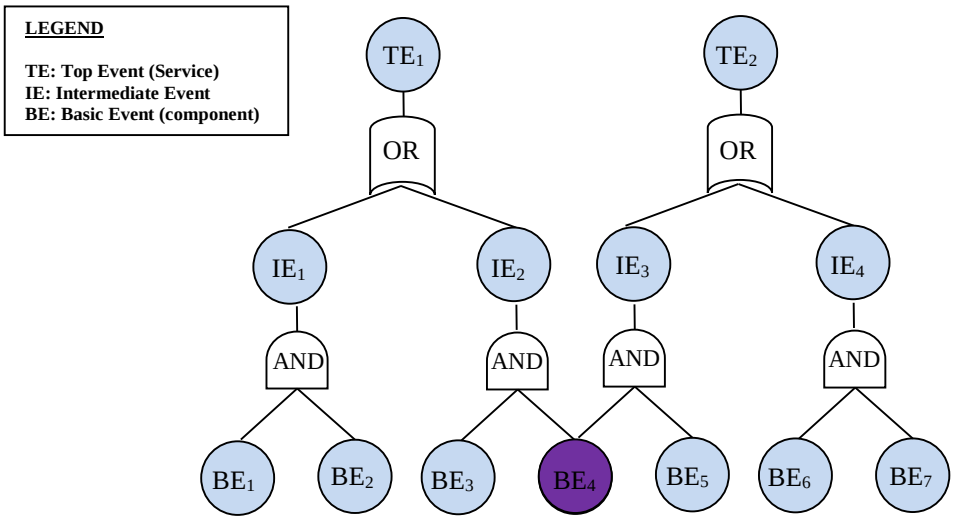


Figure 4. 2. Basic components importance factor concept

In this section, a proposition of an approach allowing the mixed-use of failure probabilities and *prior* expert-defined events importance factors is made. Moreover, if the importance factors are to be removed, then, the proposed approach is assumed to get reduced to the classical Bayesian Fault Tolerance Analysis.

The proposed approach is going to be detailed and evaluated using several simulation measures. An important opening of the proposed approach is that it can be easily coupled with the existing approaches dealing with imprecise probability values.

4.2.1.Importance factors

A risk factor, for each component in three levels and for each service was introduced in chapter 2. These three levels are as follows: not critical, primary importance and auxiliary importance. In this section, we propose to define a continuous value of importance factors (i.e. the importance factors will not be restricted to the three quantized values, but, may assume any value from an importance values interval). This "opening" from the three quantized values into a continuous value is assumed to allow to the designer, to have more flexibility in terms of importance degrees.

The first step in the proposed approach consists on defining and attributing, by the designer, importance factor values to each elementary event used by the system. This factor, denoted as $I(BE_n)$ for the elementary event BE_n, is assumed to be ranged in the interval $[\alpha, \beta]$ as illustrated in figure 4.3. $I(BE_n)$ is a function of number of available alternative events; as well as the critical usage of the BE_n for the considered top event(s) (both determined by the designer).

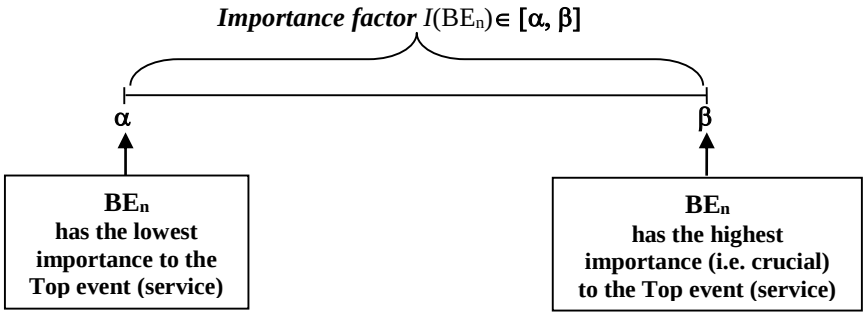


Figure 4. 3. Importance factor range

4.2.2.COST function

The second step in the proposed approach consists on defining a new global indicator ***Q(TE)*** to be used in the process of elementary events replacement decision (i.e. when an elementary event should be replaced and where multiple secondary events are potentially available to replace the faulty one). The global indicator ***Q(TE)*** is assumed to:

- Depend on the elementary events probabilities of failure as well as the importance factors of all basic events;
- Be reduced to the classical maximum likelihood decision criterion, when the importance factors are neutralized.

The proposed global indicator ***Q(TE)*** (assumed to substitute the top event failure probability used for decision purposes in the classical Bayesian decision making) is thus a function measuring the

service performance at the top event level. This indicator depends on the two vectors: $\mathbf{P^F}$ (probabilities of failure vector) and $\boldsymbol{I}$ (importance factors vector):

$$\boldsymbol{Q(TE)} = \boldsymbol{g}(\mathbf{P^F}, \boldsymbol{I}) = \boldsymbol{g}\begin{pmatrix} \left(\mathbf{P^F(BE_1)}, \ \ldots, \mathbf{P^F(BE_n)}\right), \\ \left(\mathbf{I(BE_1)}, \ \cdots, \mathbf{I(BE_n)}\right) \end{pmatrix} \quad (3)$$

[1/***Q(TE)***] is called: the ***Cost*** function.

To clarify the use of the new global indicator ***Q(TE)***, let us consider the following example shown in figure 4.4.

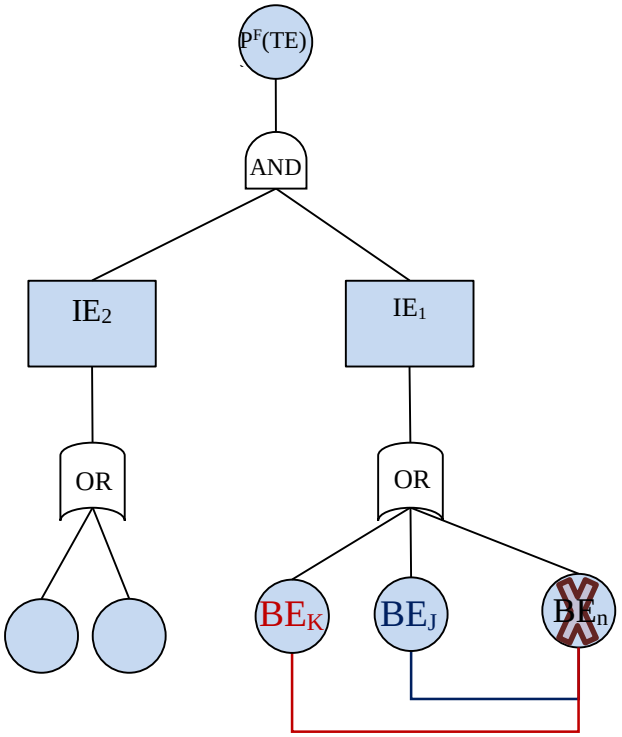


Figure 4. 4. Use of importance indicator for the replacement decision

Assume that the elementary component BE_n fails and that two potential components BE_j and BE_k may be used in order to replace BE_n. The question then is which of the two components should-we use instead of BE_n? The classical Bayesian FTA approach considers the top event new failure probabilities as the basis for the criterion to select the alternative BE.

In fact, if the basic component BE_n is replaced by the alternative component BE_i (resp. BE_k) then this will lead to a top event probability of failure $\mathbf{P^F(TE)}|_{BE_n \rightarrow BE_j}$ (res. $\mathbf{P^F(TE)}|_{BE_n \rightarrow BE_k}$). The maximum likelihood decision, allowing to choose the alternative component, consists on replacing the elementary component BE_n by the alternative component BE_j if and only if $\mathbf{P^F(TE)}|_{BE_n \rightarrow BE_j} < \mathbf{P^F(TE)}|_{BE_n \rightarrow BE_k}$. Otherwise, BE_k will be considered.

When importance factors are used, the replacement decision is conducted upon values obtained by the new global decision indicator ***Q(TE)*** instead of the top event failure probability $\mathbf{P^F(TE)}$. This means that we may encounter situations such that: $\boldsymbol{Q(TE)}|_{BE_n \rightarrow BE_k} < \boldsymbol{Q(TE)}|_{BE_n \rightarrow BE_j}$ (leading to consider the alternative component BE_k for the replacement) even if $\mathbf{P^F(TE)}|_{BE_n \rightarrow BE_j} < \mathbf{P^F(TE)}|_{BE_n \rightarrow BE_k}$ (supposed

leading to consider the alternative component BE_j for the replacement if the classical maximum likelihood decision criterion).

In other words, the use of importance factors by the global decision indicator ***Q(TE)*** as a new criterion, may change the replacement decision about which scenario will be followed to execute the service represented in the top event.

Several constraints, or assumptions, need to be verified by the ***Q(TE)*** function that we should consider:

1. ***Q(TE)*** should be directly proportional to $\mathbf{P^F(TE)}$ (i.e. $Q(TE) \propto \mathbf{P^F(TE)}$). This constraint means that when the top event probability of failure $\mathbf{P^F(TE)}$ is increased, ***Q(TE)*** should also be increased;
2. ***Q(TE)*** should be directly proportional to ***I*** (i.e. $Q(TE) \propto I$). This constraint means that when the event importance is increased, ***Q(TE)*** should also be increased.
3. When ***Q(TE)*** value of an event is increased, the less consideration should be adopted in the execution of the scenario (due to the increasing in the proportion of failure).

An interesting question may be raised at this level: What is the added value of dealing with importance factors?

This question is answered in the remaining of this chapter by showing how the use of importance factors may change the classical Bayesian replacement decision about which scenario (i.e. alternative components to choose) will be followed to execute the service represented in the TE.

4.2.3.Hybrid Probabilistic/Importance proposed approach

The proposed hybrid Probabilistic/Importance approach makes the following assumptions:

- The service is formed by three hierarchical levels: 1. Elementary (or Components) level (at this level, elementary events are referred to as basic components); 2. Intermediate level (composed by intermediate events IE_m), and 3. Top level composed by a single event, called top event (TE) representing the delivered service;
- Several fault tree schemes can be considered. These schemes may contain conjunctive AND or disjunctive OR gates linking the basic components into the intermediate events as well as those linking the intermediate events into the top event. Three of these possible schemes are given in figure 4.5.

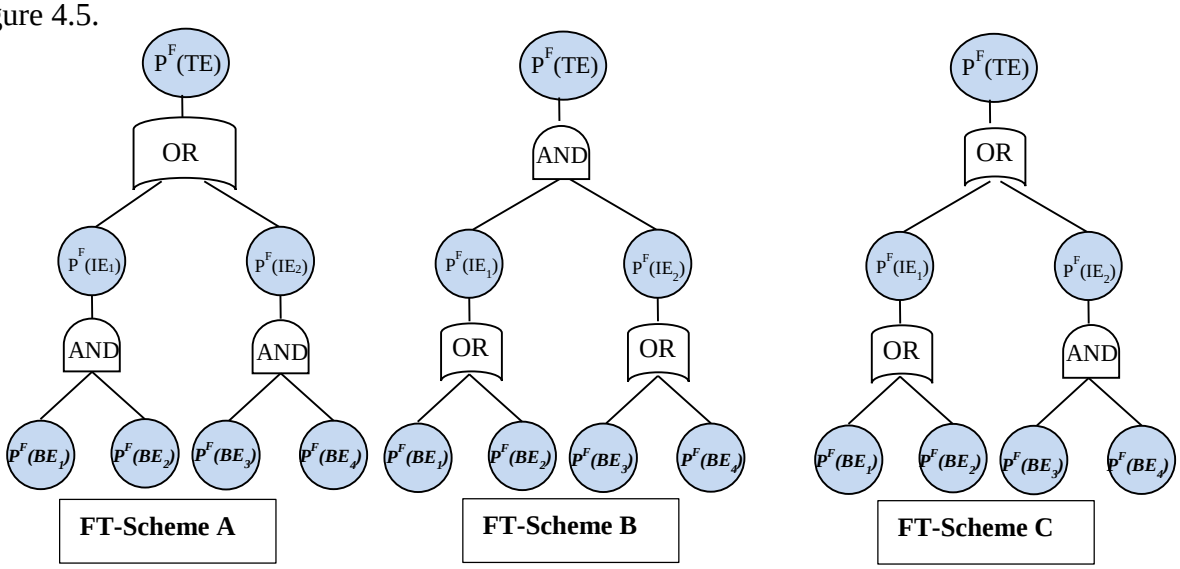


Figure 4.5. Three possible different fault trees schemes

All possible fault tree schemes can be simulated and studied in the same manner. In this chapter, only the FT-Scheme A is considered in order to illustrate and to study the behavior of the proposed approach. The reason for choosing this scheme is simply that it represents the most encountered cases where the service (TE) consists of one scenario (e.g. IE_1) and a single alternative scenario (IE_2). Each scenario is composed of two main components, BE_1 and BE_2 for the first scenario, as well as BE_3 and BE_4 for the second scenario:

- Each basic component, BE_n, is associated with two characteristic numerical features: the component probability of failure $P^F(BE_n) \in [0,1]$ and the component importance factor $I_{BE_n} \in [\alpha,\beta]$;
- The special unit value is assumed to belong to the importance factor range values (i.e. $1 \in [\alpha,\beta]$). This assumption is made in order to allow retrieving the classical Bayesian approach performances by imposing the unit value as the importance factor attributed to all basic components.

Under these assumptions, the proposed approach starts by defining an importance indicator (at the level of each basic component) using both the component probability of failure $P^F(BE_n)$ and the component importance factor I_{BE_n}.

Several approaches allowing the combination of these two performance indicators are already proposed in the reliability technical literature. In the proposed approach, the use of the direct product combination operator is used. This means that each basic component is being associated with the following importance indicator [NASA, 2002]:

$$\text{Importance\_Indicator}(BE_n) = P^F(BE_n) \,.\, I_{BE_n} \qquad (4)$$

At the intermediate level, the importance indicator of each intermediate event IE_m, is defined by extending the probability of failure at the output of a Boolean AND gate (after substituting the input basic components probabilities of failure by the importance indicators). In the case of an intermediate event having two input basic components BE_1 and BE_2, this leads to:

$$\text{Importance\_Indicator}(IE_m) = 1 - [1 - P^F(BE_1) \,.\, I_{BE_1}] \,.\, [1 - P^F(BE_2) \,.\, I_{BE_2}] \qquad (5)$$

Notice that in the case where the intermediate event is obtained by using a Boolean OR gate, the importance indicator is simply given as:

$$\text{Importance\_Indicator}(IE_m) = [P^F(BE_1) \,.\, I_{BE_1}].[P^F(BE_2) \,.\, I_{BE_2}] \qquad (6)$$

Finally, at the top level, the importance indicator of the top event TE (i.e. the global decision indicator ***Q(TE)***) is defined by extending the probability of failure at the output of a Boolean OR gate, after substituting the input basic components probabilities of failure by the importance indicators issued from the Boolean AND gates. In the case when a top event having two intermediate events input, each having two basic components (BE_1 and BE_2) and (BE_3 and BE_4), figure 4.9 (Scheme –B), this leads to the following quality of success function:

$$\boldsymbol{Q(TE)} = \left[1 - [1-P^F(BE_1) \,.\, I_{BE_1}] \,.\, [1-P^F(BE_2) \,.\, I_{BE_2}]\right] \times \left[1 - [1-P^F(BE_3) \,.\, I_{BE_3}] \,.\, [1-P^F(BE_4) \,.\, I_{BE_4}]\right] \qquad (7)$$

4.2.4.Hybrid approach behavior and simulation results (Conjunctive AND gate):

In this section, the behavior of the proposed approach is studied using the fault tree (scheme A) given in figure 4.9 (i.e. two stages: a first stage composed of two conjunctive AND gates linking 4 basic components and the second stage consists on combining the two intermediate events using the disjunctive OR gate linked to the Top event). Other tree schemes can be similarly evaluated (for instance using the disjunctive OR gate at the first stage, or, the conjunctive AND gate at the second stage; or even both conjunctive AND and disjunctive OR gates at the same stage).

In order to evaluate the behavior of the proposed approach, figure 4.6 shows the adopted visualization method of the importance indicator of an intermediate event IE for which two basic components BE_1 and BE_2 are used.

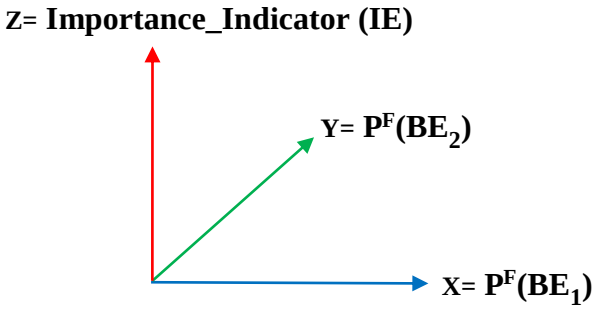


Figure 4. 6. Visualization approach to illustrate the importance indicator for an intermediate event (given two fixed basic components importance values (I_{BE1} and I_{BE2})

Figure 4.7, illustrates the importance indicator behavior of an intermediate event obtained by a conjunctive AND function of two basic events (similar results can be obtained in case where the event is obtained as a disjunctive OR function). In this figure, multiple values for the basic component importance are used (I_{BE_1} and $I_{BE_2} \in \{1, 1.2, 1.4, 1.6, 1.8, 2\}$, i.e. $\alpha = 1$ and $\beta = 2$).

For each couple of importance values (I_{BE_1}, I_{BE_2}) the importance indicator of the intermediate event is traced as a function of $(X,Y) = (P^F(BE_1), P^F(BE_2))$ following the proposed combination equation (5).

It is worthwhile to notice that the global shape of the traced importance indicator decision surface (as a function of $(X, Y) = (P^F(BE_1), P^F(BE_2))$) changes significantly depending on the variation of the basic components importance values. Nevertheless, when $I_{BE_1} = I_{BE_2}$, the symmetric aspect of the traced shape of the importance indicator decision surface shows the "equivalent" role of both basic components. This symmetric aspect is "reduced" upon the relative importance variation of these components.

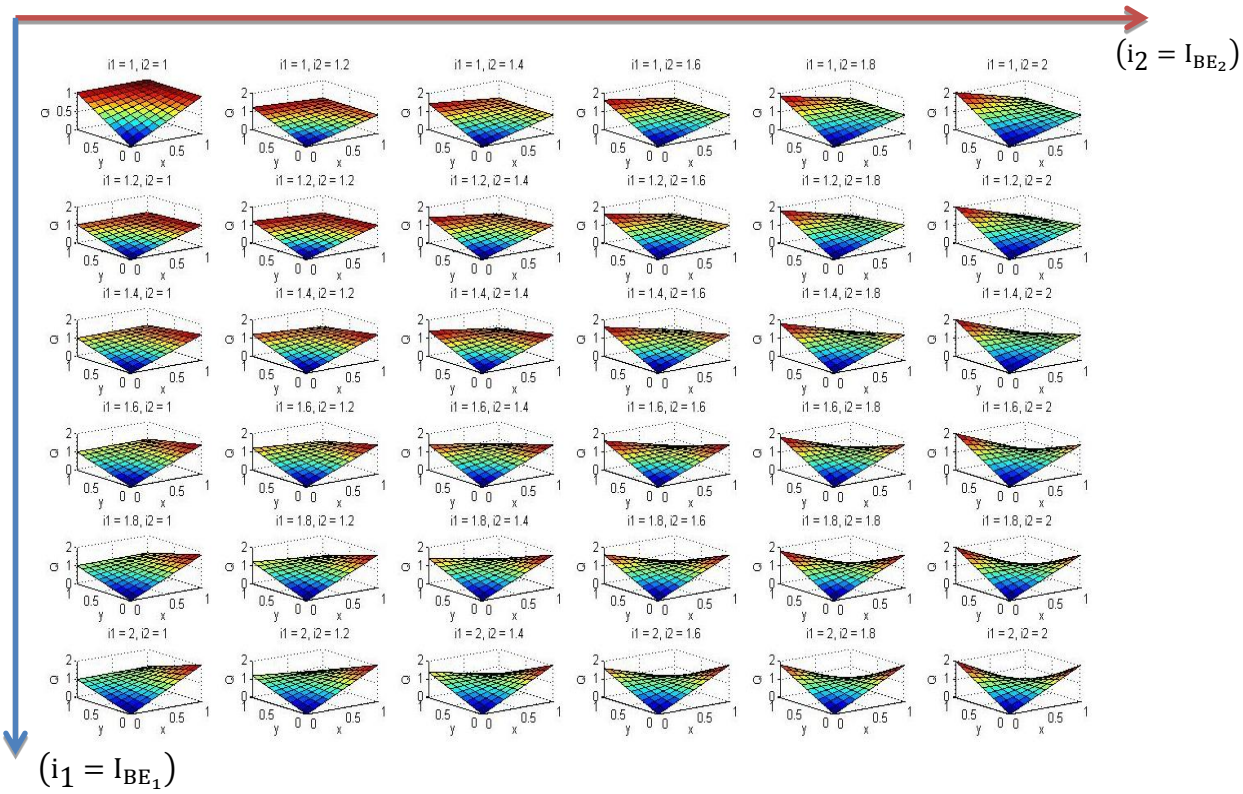


Figure 4. 7. The importance indicator behavior of an intermediate event obtained by a conjunctive AND gate of two basic events

For instance, if we consider the 4th surface from the 1st line (when i_1=1, i_2=1.6), we can obviously notice that the decision function *Q* is a function of y (since the second importance factor is more significant). Whereas, in the 1st surface from the 4th line, the opposite case (i.e. i_1=1.6, i_2=1) the decision function *Q* is a function of x (since the first importance factor is more significant). The same cases are applied respectively for the 6th surface from the 2nd line and the 2nd surface from the 6th line, when i_1=1.2, i_2=2 (resp. i_1=2, i_2=1.2).

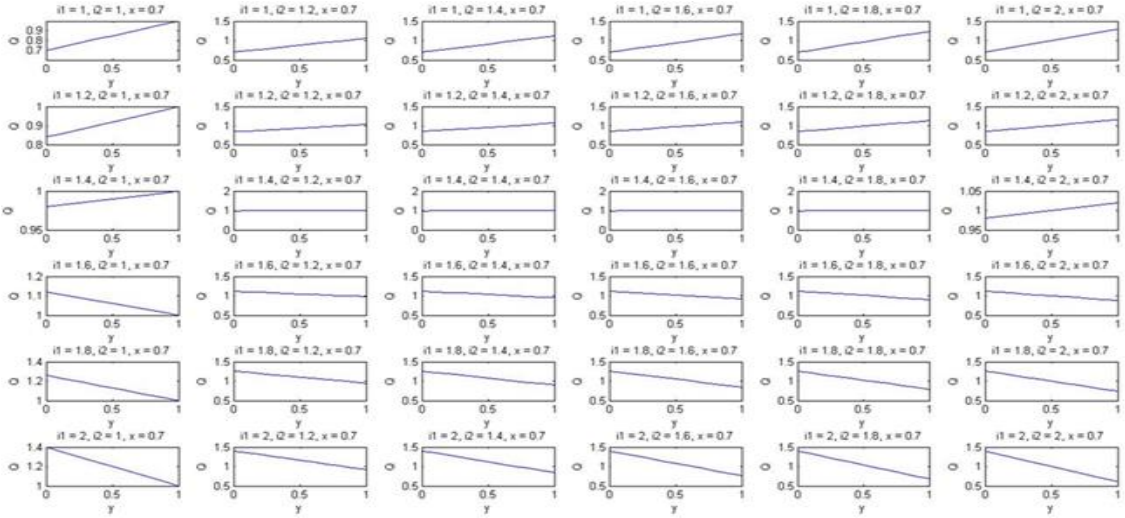


Figure 4. 8. A synthesis of how the importance values of basic components influence the importance indicator behavior

To get a closer analysis of the importance indicator behavior, figure 4.8 retraces the same configuration as it was in figure 4.7, but with a fixed probability of failure for the first component (fixed at X = $P^F(BE_1)$ = 0.7). In this figure, if we study the slopes when $i_1=i_2$ (the diameter of the figure), we can witness how the behavior of the decision is a function of y. In the other case, the behavior of the decision assumes increasing values when $i_1<i_2$ ($i_1=1$, $i_2=1.6$) depending on the value of y, and it assumes decreasing values when $i_1>i_2$ ($i_1=1.6$, $i_2=1$) depending on the value of y.

Figure 4.9 gives a synthesis of these results. The relative importance values of the basic components are separated into two zones.

In zone 1 (res. zone 2), I_{BE_1} is considered such as $I_{BE_1}<I_{BE_2}$(res. $I_{BE_1} > I_{BE_2}$). In zone 1, it is worthwhile to notice that the slope of the traced line representing the importance indicator (as a function of $P^F(BE_2)$) is an increasing function of I_{BE_2}. The minimum value is reached when Y=$P^F(BE_2)$ tends towards zero and is given by $P^F(BE_1)$ x I_{BE_1}. On the contrary, in zone 2, the importance indicator (as a function of $P^F(BE_2)$) is a decreasing function of I_{BE_2}. In this case, the maximum value is reached when Y=$P^F(BE_2)$ tends towards zero is given by $P^F(BE_1)$ x I_{BE_1}.

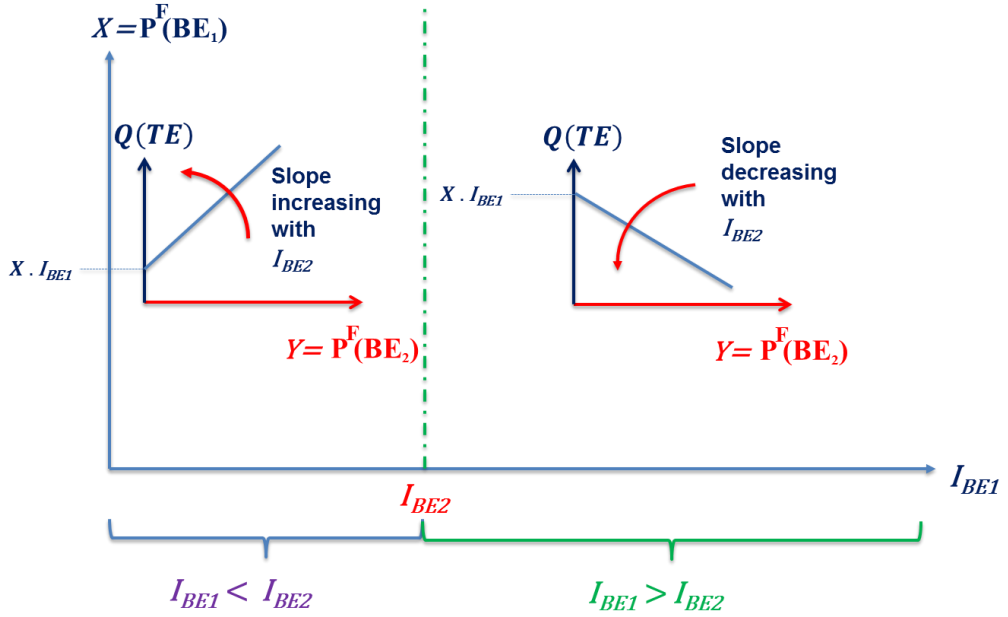


Figure 4. 9. A synthesis of the importance indicator behavior

In figure 4.10, the results shown in figure 4.8 are combined into one shape to illustrate how the importance indicator is an increasing or a decreasing function in relation with the importance factor of the basic events. This is illustrated with a fixed probability of failure for the first component (here, we have fixed at X=$P^F(BE_1)$ = 0.6). In this figure, it is shown how all lines (representing the importance indicator as a function of Y=$P^F(BE_2)$) cross the very same point.

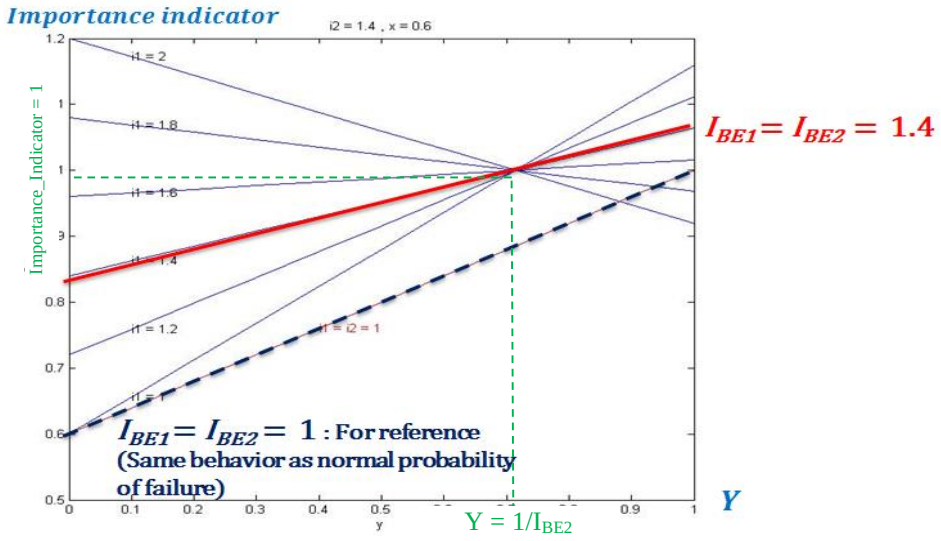


Figure 4. 10. The importance indicator behavior

The coordinates of intersection point of all these lines are given by: (Y = $1/I_{BE2}$, Importance_Indicator = 1). When the importance values for the two constituting basic components equal the unit value (i.e. $I_{BE_1} = I_{BE_2} = 1$), the importance indicator traces the values of failure probabilities of the conjunctive AND event (function of $P^F(BE_1)$ and $P^F(BE_2)$).

4.2.5.Hybrid approach behavior and simulation results (Disjunctive OR gate):

In order to evaluate the behavior of the global indicator ***Q(TE)***, figure 4.11 shows the adopted visualization method of the global importance indicator of two events importance indicators (IE_1 and IE_2 respectively). As illustrated in figures 4.7 and 4.8 of the previous sub-section, it illustrates the global importance indicator behavior of the top event obtained by a disjunctive OR function of two sub events.

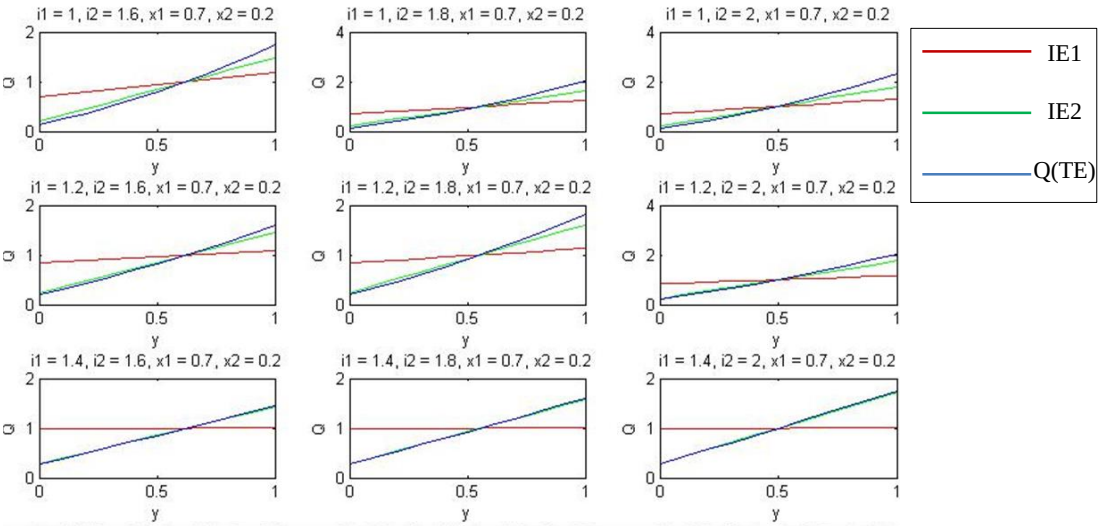


Figure 4. 11. A synthesis of how the Importance indicator values of sub events influence the global importance indicator behavior of a top event Q(TE)

In this figure, the two sub events are the two intermediate events IE_1 and IE_2. We study the behavior of the Top Event *Q*(TE), based on various values for the sub events (obtained by several values of i_1 less than several values of i_2).

To analyze the behavior of the disjunctive OR gate, we have to notice that the Top Event performance depends on 8 parameters (the 4 failure probabilities and the 4 basic components importance factors).

Let us consider the example illustrated in figure 4.12.

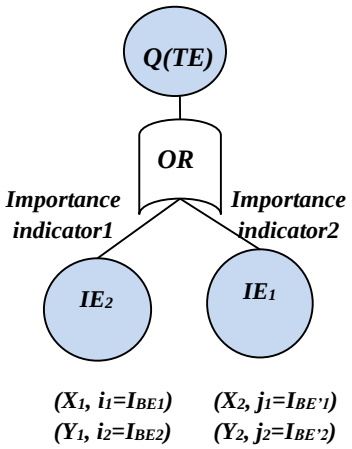


Figure 4. 12. An example to illustrate the behavior of the disjunctive OR

In this example, 7 parameters are fixed and only one basic component importance factor is considered as variable.

All the following values are considered as fixed (for illustration purposes):

$X_1 = P^F(BE_1) = 0.1$; $I_{BE_1} = 1.4$;

$Y_1 = P^F(BE_2) = 0.08$;

$X_2 = P^F(BE_3) = 0.05$; $I_{BE_3} = 1.2$;

$Y_2 = P^F(BE_4) = 0.06$; $I_{BE_4} = 1.3$.

The importance factor of the second basic component, I_{BE_2}, is considered as variable in the range [1, 2]. In figure 4.13, the following importance indicators are plotted (all function of the variation of I_{BE_2}):

- Importance Indicator for first intermediate event IE_1 is plotted in green $II(IE_1)$.
- Importance Indicator for second intermediate event IE_2 is plotted in blue $II(IE_2)$.

The global success factor for the top event ***Q(TE)*** is plotted in continues read line. ***Q(TE)*** is simply the direct product of both importance indicators of the two intermediate events.

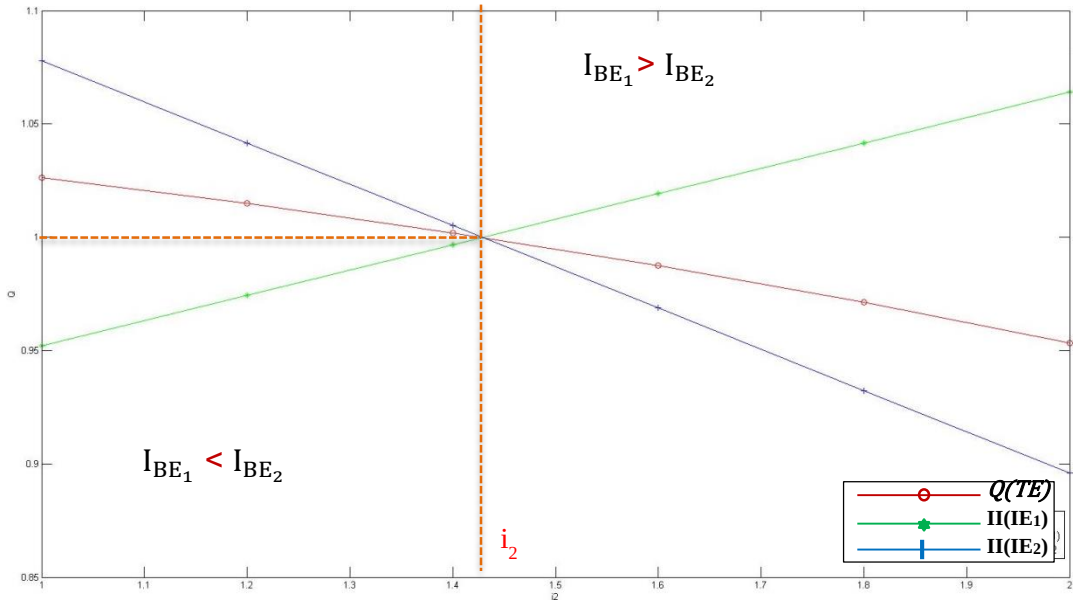


Figure 4. 13. A synthesis of how the behavior of the disjunctive OR

It is worthwhile to notice that if $I_{BE_2} < (1/Y_1 = 1/ P^F(BE_2))$, the disjunctive OR gate will select the first intermediate event as a selected scenario (since the importance indicator of this intermediate event is less than the importance indicator of the second intermediate event). On the contrary, this decision is reversed in case where $I_{BE_2} > (1/Y_1 = 1/ P^F(BE_2))$. This shows the advantage of using different basic components importance factors.

4.3. Illustrative example

In this section, we will consider the following illustrative example shown in figure 4.14. In this example, the basic event BE_3 fails, and the designer is supposed to have, a priory, defined two decision scenarios.

The first scenario consists on substituting the failed component with the component BE_2. In this case, the following parameter that we have to consider is: the new Top event quality function value Q`(TE).

The second scenario consists on substituting the failed component with the component BE_1. In this case, the following parameter that we have to consider is: the new Top event quality function value Q``(TE).

Let us assume the same *prior* fixed values of 7 parameters (as was considered and explained in the example of figure 4.12 and figure 4.13 respectively) and keep $I(BE_2)$ as a variable.

It is worthwhile to notice that in the case where the role of the basic components importance factors are annealed (i.e. all are set to the value = 1), then, if BE_2 (respectively BE_1) is used as an alternative of BE_3, the resulting top event failure probabilities ($P^F(TE_1)$ & $P^F(TE_2)$ respectively) are respectively given by 0.0233 and 0.0265. Thus, the expert designer should select BE_2 as the alternative for BE_3.

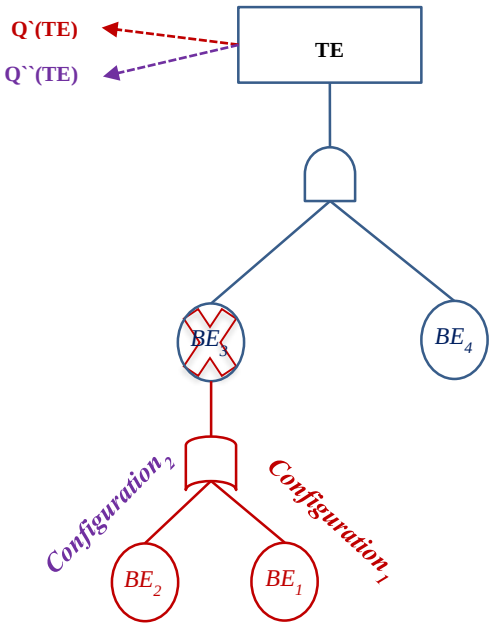


Figure 4. 14. Illustrative example

Supposing other numerical values of probabilities of failure (the same 7 parameters) and always keeping *I*(BE2) as a variable, we can show the obvious influence of the importance factor in the replacement decision, as illustrated in figure 4.15.

$X_1 = 0.1$	$Y_1 = 0.098$	$X_2 = 0.9$	$Y_2 = 0.06$
$i_1 = 1.4$	$i_2 = 1 : 2$	$i_3 = 1.2$	$i_4 = 1.3$

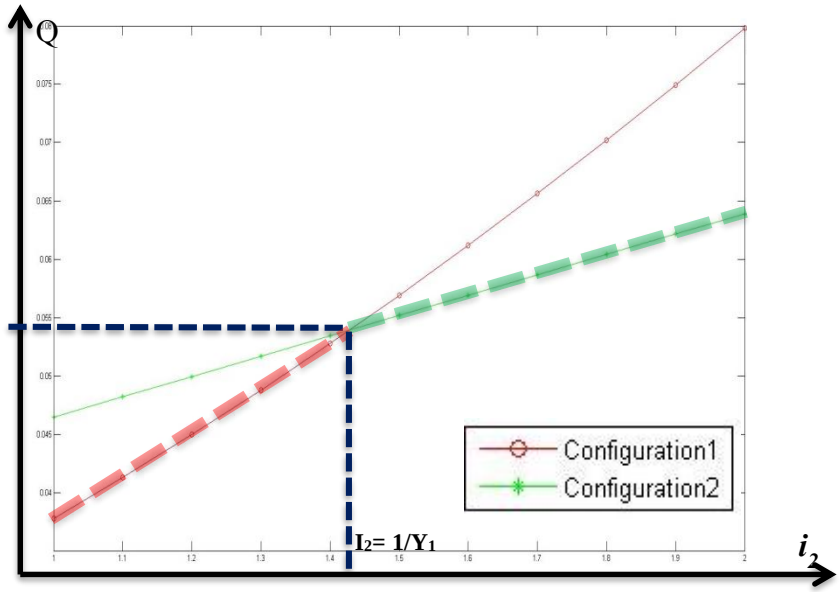


Figure 4. 15. Illustrative example with 7 fixed parameters and I(BE2) as a variable

In fact, the importance factor influences the decision about which scenario (configuration) to be selected. In this figure, it is shown how the two importance indicators cross the same point. This intersection point is given by: ($I(BE_2) = 1/Y_1$).

Below this threshold value, when the Importance Indicator $II_1 < II_2$, then, the system will select the configuration of replacing the failed component BE_3 with BE_2.

On the contrary, this replacement decision is reversed in the case where $II_1 > II`_2$. Here, the system will select the configuration of replacing the failed component BE_3 with BE_1.

This clearly shows the influence of different basic components importance factors on the final decision of components substitution.

In order to study the role of the importance factor range $[\alpha, \beta]$ in the replacement decision, let us consider the numerical example illustrated in figure 4.16, using the same fixed values as shown previously, but with a different importance factor $I(BE_1) = 5$ and $I(BE_2)$ is considered as a variable in the range $[\alpha = 1, \beta = 10]$.

As it is shown, whatever the range of the importance factor of the basic components, it always gives the same shape and thus the resulting replacement decision is related to the variation between the importance values between the alternative basic components rather than the range of importance factor.

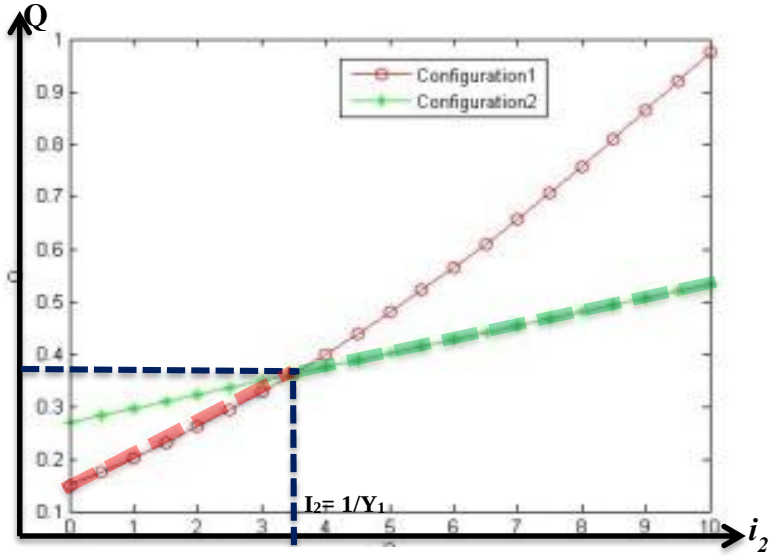


Figure 4. 16. Importance factor range [α, β]= [1, 10]

Last but not least, in some particular cases, the importance factors range should be designed as a function of the failure probabilities dynamics. For instance, in the example shown in figure 4.17, we tried to minimize as possible the values of the probability of failure of some basic components. With the consideration of P^F ($BE_1 = 10^{-4}$), P^F ($BE_2 = 0.0098$) and $P^F(BE_4=6*10^{-4})$, if the range $[\alpha, \beta]$ is "badly" chosen, configuration2 remains always the best configuration to be chosen without looking to the importance factors (the intersection point is out of the chosen range $[\alpha, \beta]$). The main reason of this issue is that the two importance indicators cross the same point given by: $I(BE_2) = 1/ P^F(BE_1)$. In other words, the importance factor value of the intersection point is inversely proportional to the probability of failure. Therefore, the range of the importance factor should be selected in an appropriate manner with the range of the failure probabilities. In order to resolve this problem, it consists on shifting the fault probabilities range in order to fit a fixed predefined range $[\alpha, \beta] = [1, 2]$ or to choose a very small range values of $[\alpha, \beta]$.

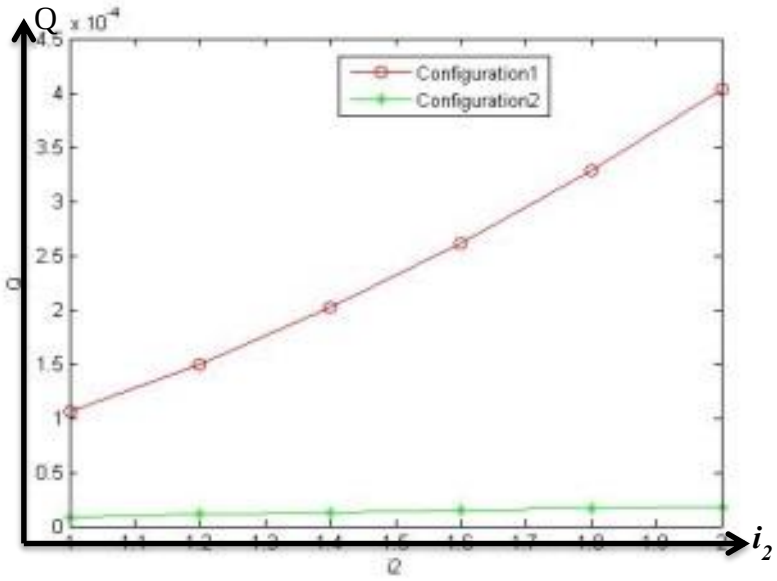


Figure 4.17. Tiny probabilities of failure

4.4. Conclusion

In this chapter, analysis and evaluation of existing approaches and studies related to Fault Tree Analysis (FTA) is conducted in order to appreciate the application of these approaches for the evaluation of the success of services related to Ambient Assisted Living Systems. After an analysis of the state of the art studies two major comments can be formulated.

- It is clearly shown that the Bayesian approach is the most widely used (for the evident reason that it allows the estimation of the Top Event (i.e. service) failure probability depending upon basic components fault probabilities. All encountered studies target to find solutions in the case of basic components imprecise failure probability due to the lake of knowledge of these probabilities or the very small size of statistical samples allowing estimating precise failure probability values. Based on this fact, nearly all encountered studies propose the use of mathematical theories dealing with imprecise probabilities as the evidence theory, the fuzzy sets theory and the possibility theory. After substituting basic components fault probabilities by the basic components failure possibility values (or fuzzy numbers), existing approaches use the Top Event possibility of success instead of the classical Top Event failure probability in order to evaluate the system performances or to make a decision related to the substitution of a basic component by an alternative one.
- None of the existing approaches offers the possibility of using a user defined degree of importance (that each component holds for the success of the corresponding service).

In the proposed work, the evaluation of a system services success/failure is conducted using failure analysis approach and integrating two sources of information: the basic components failure probabilities as well as the designer-defined basic components importance factors. The proposed approach can be simply reduced to the classical Bayesian one by affecting the unit value as the importance factor to all basic components.

In the proposed work, the computation of decision function is based on independent events. To consider dependent events, the decision function should be computed in terms of an event conditional probability.

The proposed approach was evaluated using a possible tree configuration of events organized in three conceptual levels (Basic/Intermediate/Top levels), with conjunctive AND gates linking the basic level into the intermediate one, and, disjunctive OR gates linking the intermediate level into the top level.

The result of the proposed approach can be resumed by confirming that the use of user-defined important factor may have a major role in the selection of alternative components and that the obtained decision may be different from the one obtained by the Bayesian approach.

Chapter 5.

Workflow verification and proposed experimental validation

Whereas chapter 3 has detailed the workflow processes, and chapter 4 has introduced the fault tree analysis approach, in this chapter, the experimental framework allowing the global verification of the proposed workflow (with the consideration of fault tolerance mechanism) as well as a proposition for the system validation are presented. After a short introduction concerning the conceptual verification and validation concepts in section 5.1, a proposition of a verification framework is presented in section 5.2. This section also presents, through the simulation results of an illustrative example, the viability of the proposed system structure modeling. Section 5.3 is devoted to a proposition of the proposed system validation approach. A conclusion, in section 5.4, concludes this chapter.

5.1. Introduction

It is worthwhile to differentiate between two notions (since they are often confused) in the system development phases: the verification and the validation notions.

In fact, the verification notion addresses the question: "Are we building the product right?" whereas, the validation notion tends to answer the question: "Are we building the right product?" [Barry, 1989].

According to [IEEE-STD-610, 2016], the distributed system verification issue entails the process of evaluating the developed software to determine whether the products of a given development phase satisfy the conditions imposed at the start point of this phase and to determine whether they satisfy the specified requirements. In other words, the software verification process is ensuring that the product has been built according to the requirements and design specifications. On the other hand, the software validation process tackles the issue of ensuring that the developed product actually meets the user's needs, and that the specifications were correctly defined in the first place.

Software verification and validation are difficult tasks because a developer cannot test forever, and it is hard to know how much evidence is enough.

[Collofello, 1998] introduced five common objectives of the verification and validation process. These objectives provide a framework within which it is possible to determine the applicability of various verification and validation approaches and techniques:

- Correctness: This objective refers to the extent to which the product is fault free;
- Consistency: Referring to the extent to which the product is consistent within itself and with other products;
- Necessity: This concept considers the extent to which everything in the product is necessary;
- Sufficiency: Referring to the extent to which the product is complete;
- Performance: This objective considers the extent to which the product satisfies its performance requirements.

Different approaches and methodologies are often followed, in order to perform the verification and validation process of the entire software, solution or system.

In the domain of AAL solutions, several projects and systems have been developed to provide assistive services to the elderly people. These projects have been validated in different locations and laboratories, to express the correctness of provided solutions from the user's point of view and to meet his satisfaction.

In the Lab-STICC research labs, several research activities had been conducted. Some of these researches are related to the domain of home automation solutions. In these activities, experimental validations had been realized to evaluate and to validate the conducted works.

Research works presented in [Willy thesis. 2012] had been coupled with a number of validation protocols. The experimental validations conducted in this work were of two types. First, Willy proposed to conduct a series of experiments to evaluate the Domain Specific Modeling Language, IntelHome, and the design flow, which correspond to the environment and interaction models. These experiments were performed on a real case study: "les appartements tremplins du centre de Kerpape". The conducted evaluation involved the application of three types of empirical methods, the case study allowing the evaluation of the expressiveness of the language IntelHome. The second empirical method consisted of leading controlled experiments to assess the effectiveness and suitability (i.e. accessibility) of the DSML and associated design flow. Whereas, the third method is based on conducted surveys to evaluate the effectiveness and adequacy (i.e. accessibility) of DSML and the associated design flow, collecting information related to the fulfillment of the participants. Secondly, Willy proposed to validate the proposed non-intrusive supervision model. Also, he proposed to validate the non-intrusive supervision model, using datasets derived from the "Domus" smart home of the University of Sherbrooke [Sherbrooke, 2016].

In [C. Seguin Thesis. 2015], C. Seguin devoted an experimental validation, by studying a concrete example to illustrate the notions and modeling concepts proposed in his research work. To present the modeling principles, he examined, first, the experimental platform and its origins. Second, he presented a proof of concept and specific scenarios to highlight the deployment strategies.

The validation protocol consisted of performing the experiments in different experimental platforms to validate the proposed system modeling approach. Following the projects conducted at the Lab-STICC Laboratory in Lorient, an environmental control platform was already developed. Starting from this base, C. Seguin adapted this control platform during field studies performed both in Kerpape center and in Domus laboratory, in order to fit this platform with the proposed approach.

In the following sections, the verification and the validation of the proposed reconfigurability system workflow are detailed. After a brief recall of the proposed workflow process, the verification of the implemented structure is conducted and illustrated using a representative example captured from a real living lab (Experimental' HAAL). The developed implementations are introduced as a background of the validation process. A proposition of a validation process is then given.

5.2. Proposition of a verification framework

As previously mentioned, to check and to verify the proposed system structure, two issues are of major interest: the conceptual system evaluation, i.e. the verification issue, (allowing to prove the soundness and the correctness of the proposed system structural design) and, the delivered services validation (in terms of user's acceptability of the proposed system as well as of the system reactivity when unexpected failures occur). In this section, the proposed system verification framework is detailed.

5.2.1.Conceptual system verification

The global AAL system architecture, already proposed in this study, is resumed in figure 5.1.

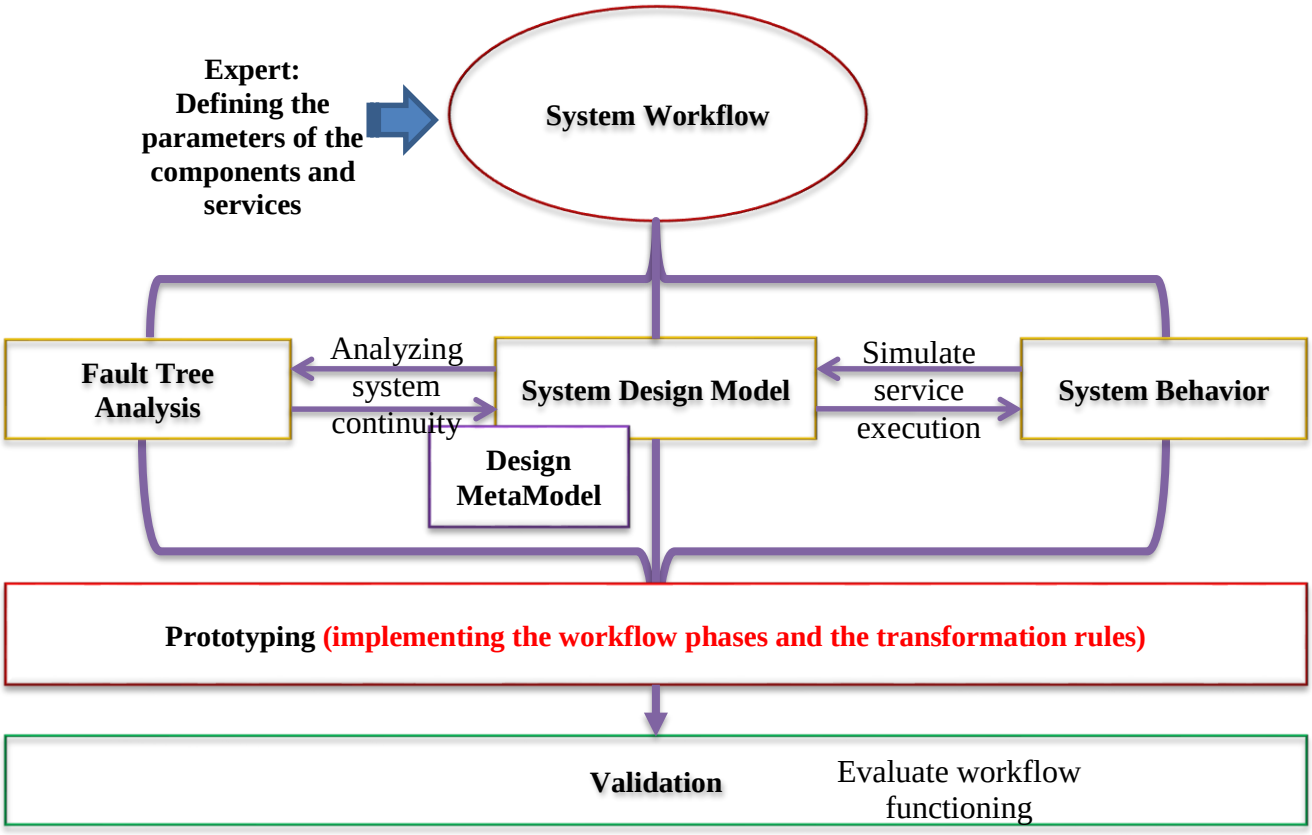


Figure 5.1. System workflow architecture

This AAL system workflow is composed of three main model blocks:

1. ***The system design model***: This block refers, in terms of services and components, to the representation of the system structural elements;
2. ***Fault Tree Analysis model***: This model refers to the block in charge of the decision aided approach (based on the analysis of the good functioning of the system services). The adopted model uses fault tolerance mechanisms;
3. ***System behavior analysis model***: This model block intends to simulate the scenario execution of a given service in order to ensure that the service is delivered to the user (following the predefined scenarios).

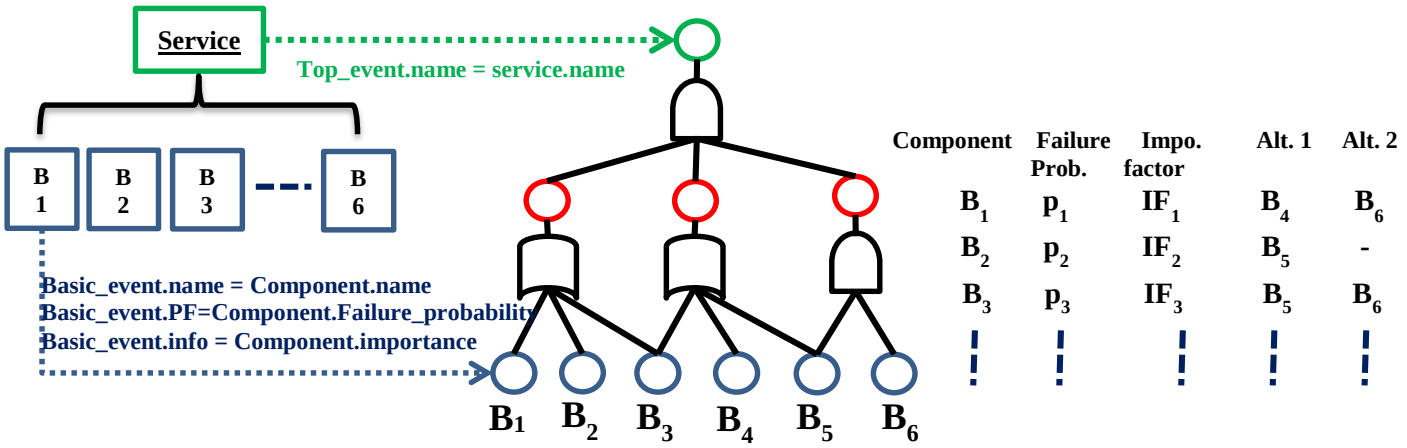
The expert designer is in charge of:

- Defining and adjusting the system elements parameters;
- Defining alternatives scenarios in case of components failure;
- Conducting the transformation rules allowing, to the system, to check the validity of the proposed system design model and the good functioning of each service.

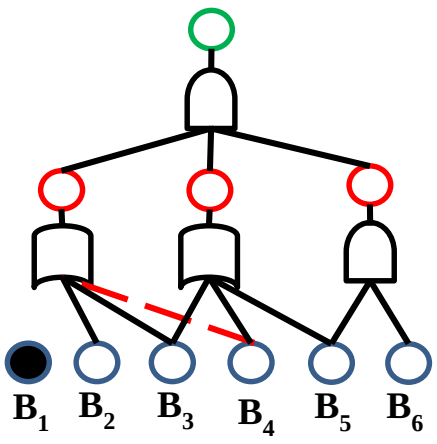
Two transformation rules are considered: the design model-to-the Fault Tree Analysis model rule, and, the design model-to-the behavior analysis model rule.

The design model-to-the fault tree analysis model transformation rule is applied to analyze the influence of a component failure on the functioning state of a given service. This rule will consider the service (as well as all its activated components) in order to establish the expert's predefined tree structure model. The target of this rule is to determine, from the expert's predefined scenarios set, the optimal service scenario delivering the service in an acceptable manner from the user's perspective.

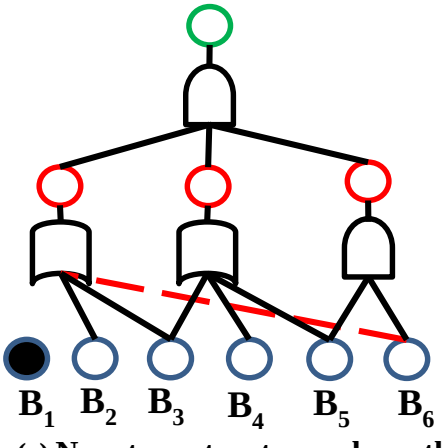
This transformation rule is illustrated by the workflow given in figure 5.1. In figure 5.2.a, the tree structure, of a service, is given as well as the characteristic parameters (i.e. failure probability and importance) for each component participating into the service functioning. Moreover, alternative scenarios are predefined for each component in order to face components failure situations. For instance, when the component B_1 fails, the design model to the analysis model transformation rule addresses this failure case into the FTA model. Using the components characteristic features, the two potential alternative scenarios are then evaluated by the FTA model (corresponding to the two possible alternative scenarios: substituting B_1 by B_4, or, substituting B_1 by B_6). The FTA helps the designer to propose the most adequate alternative scenario as well as the new tree structure resulting from the alternative replacement process. In other words, the tree structure given in figure 5.2.b (resp. figure 5.2.c) is regenerated by the transformation rule from the design model if the component B_1 is substituted by the alternative component B_4 (resp. B_6). This process of tree restructuring leads the expert to make a re-estimation of the importance factor for each corresponding component, because of the changing in the tree architecture.



(a) Basic Tree Structure and associated parameters



(b) New tree structure where the component B_1 is replaced by B_4



(c) New tree structure where the component B_1 is replaced by B_6

Figure 5.2. Illustrative example of the design model-to-the analysis model rule

Once the design model-to-the analysis model transformation rule is executed, the behavior model of the returned service structure (with the adopted alternative scenario) should be verified in order to check that the considered service reaches the deliverance issue.

Therefore, the second transformation rule aims to transform the system design model into the behavior model generating, thus, the simulation part in charge of checking the capability of the given service to be achieved and delivered (following the predefined scenarios, in an acceptable mode from the user perspective).

The key target of this rule consists on determining the execution behavior of each service and checking the reachability graph to evaluate the feasibility of the predefined scenarios.

The behavior model is appealed to consider the service (as well as its activated components and their alternatives) in order to establish the simulation structure model and to prove that the graph, represented as a Petri Net, is reachable from the start point to the end point, i.e. the deliverance of the service to the user.

Figure 5.3 illustrates the functioning of this transformation rule (i.e. defining the graph places referring to the service as well as to the components including the alternatives).

In fact, in figure 5.3, an example of the behavior structure of a given service as well as each component participating into the service functioning (principals and alternatives) is shown. In this example, each component is assumed to have a single alternative component.

In this structure, the alternative scenarios are included for each component, and the transitions in the graph will determine the scenario to follow, in case of failure.

For instance, when the place B_1 (representing the component B_1 in figure 5.4.a) does not fire a token to B_2, it refers to the failure of B_1. Thus, B_4 will be fired as an alternative scenario, to face this failure as illustrated in figure 5.4.b.

In Petri Net conceptions, when the transition between B_1 and B_2 does not fire, instead, the transition between B_4 and B_2 will fire as an alternative scenario of the failed component B_1. In this case, the behavior structure will show how the service is executed in a continuous manner.

When the token is fired to the place END, then this leads to conclude that the graph is reachable and the service reaches the deliverance issue following the returned scenario from the system structure, as show in figure 5.4.c.

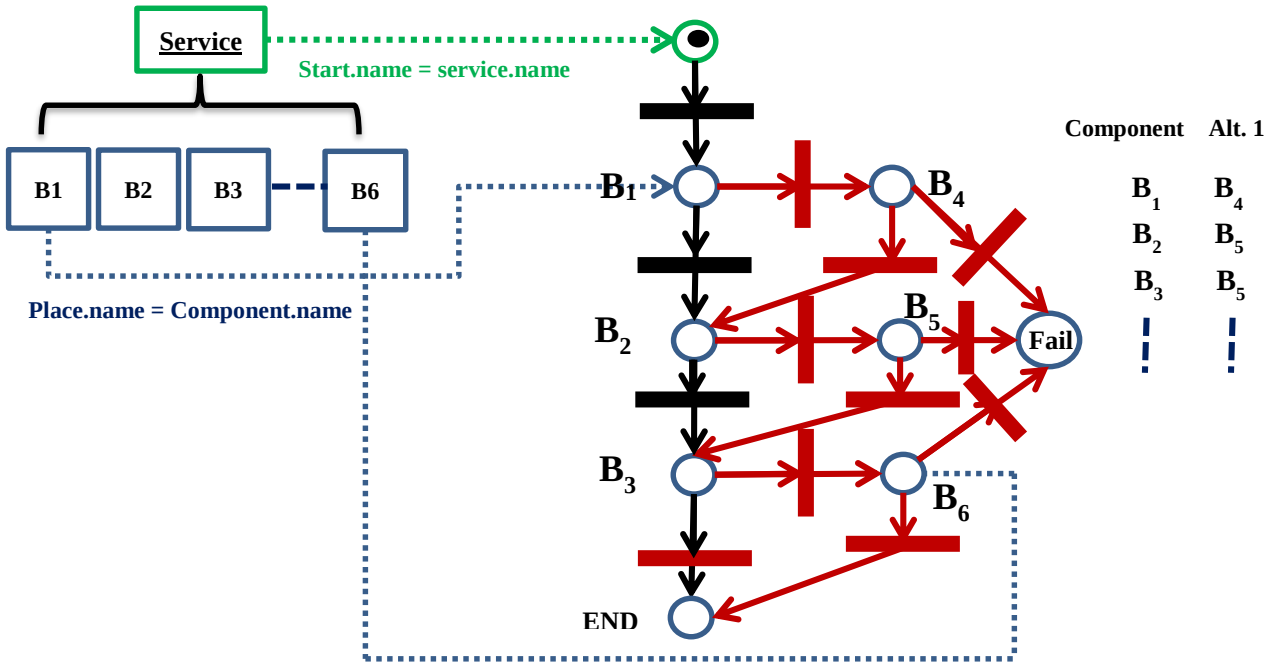
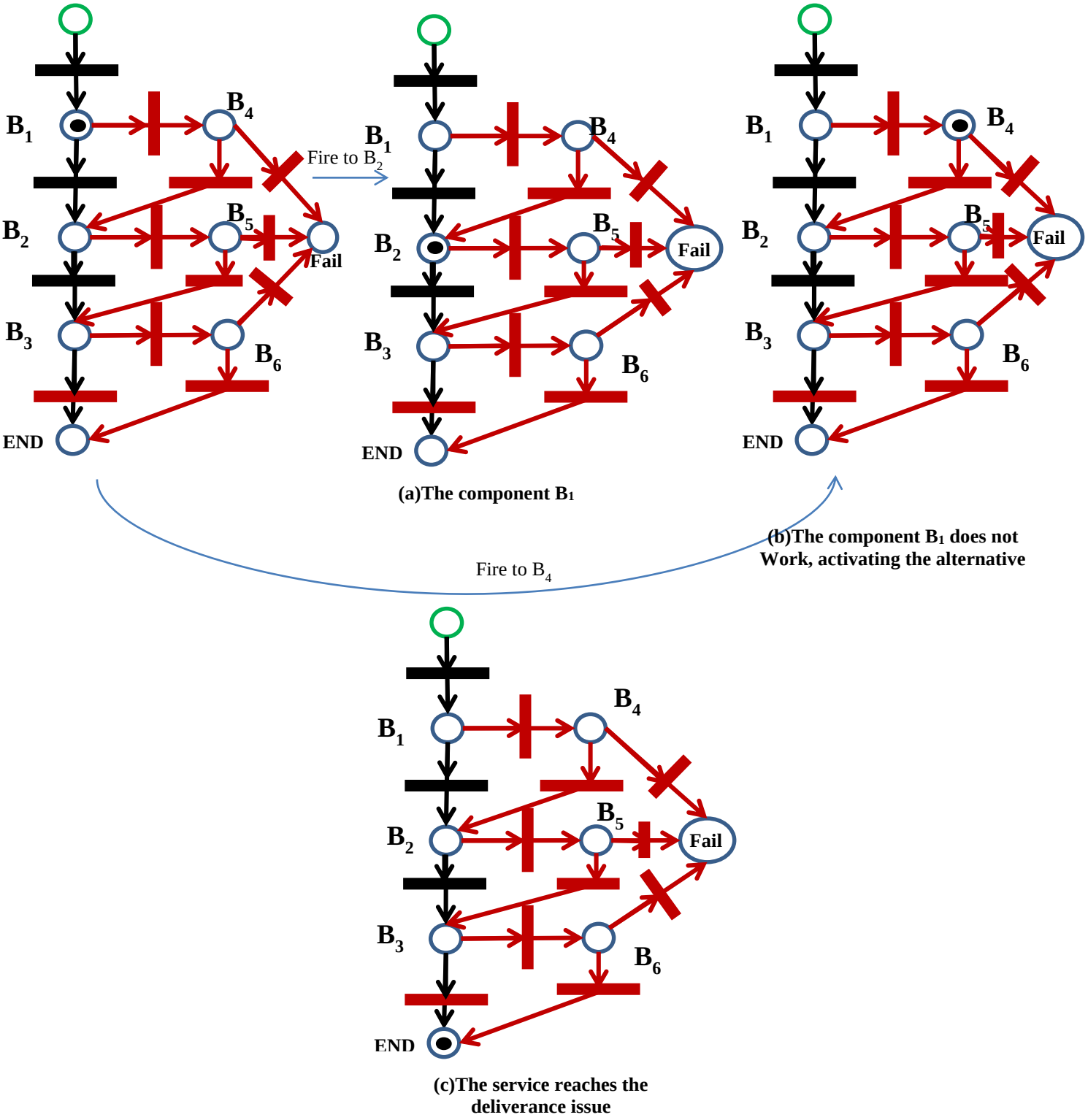


Figure 5.3. The behavior structure of a given service and its components generated by the transformation rules



(a)The component B_1

(b)The component B_1 does not Work, activating the alternative

(c)The service reaches the deliverance issue

Figure 5.4. Illustrative example of the behavior process

In order to check the correctness of the proposed system conceptual architecture given in figure 5.1, the implementation structure given in figure 5.5 is proposed in this study. In this figure, several steps are performed to verify the developed workflow. Following these steps, the expert designer will develop the workflow representing the system model elements.

In order to illustrate this verification structure, the living space context is considered in this section. First, the designer introduces the available components in the living space. Once the designer establishes the system components models, the designer starts to define, in a second step, each service needs to be provided by the system, including the selection of the components, as well as the alternatives composing the considered service.

The definition of the components, within the considered service, requires the designer to define additional parameters of the component model definition in order to establish the association between the service and the corresponding components.

The definition of the components, and each service provided by the system, is a conducted process in the implementation structure, using Python code, to represent the established models using XML schema.

Furthermore, each predefined service needs to be analyzed, in terms of continuity. Thus, each given service model is transformed into the fault tree analysis model. The transformation rule, which transforms the service design model into the analysis model, is an implementation conducted process in the illustrated implementation structure, using python framework. The tree structure is generated using XML model schema.

When the tree structure model is generated for a given service, the implementation structure conducts an analysis calculation, based on the adopted approach proposed in chapter 4, to evaluate the defined service functioning scenario.

The aim of the conducted calculation is to give a decision help about each adopted scenario to help the designer and to return (in a manual manner) to the design model to ameliorate the design structure of the system, in a static way.

The idea of injecting a faulty component to the system, in order to generate an alternative scenario, is not conducted in the implementation structure, and will be considered in the validation protocol.

The final step consists on conducting the transformation of the design model-to-the behavior model, in order to generate a simulation graph. The aim of this transformation is to establish a Petri Net simulation graph allowing verifying that the given service reaches the deliverance issue, following the predefined scenarios. This transformation rule is conducted in the implementation structure in Python framework. The generated graph and reachability verification are conducted using the Petri Net modeling framework.

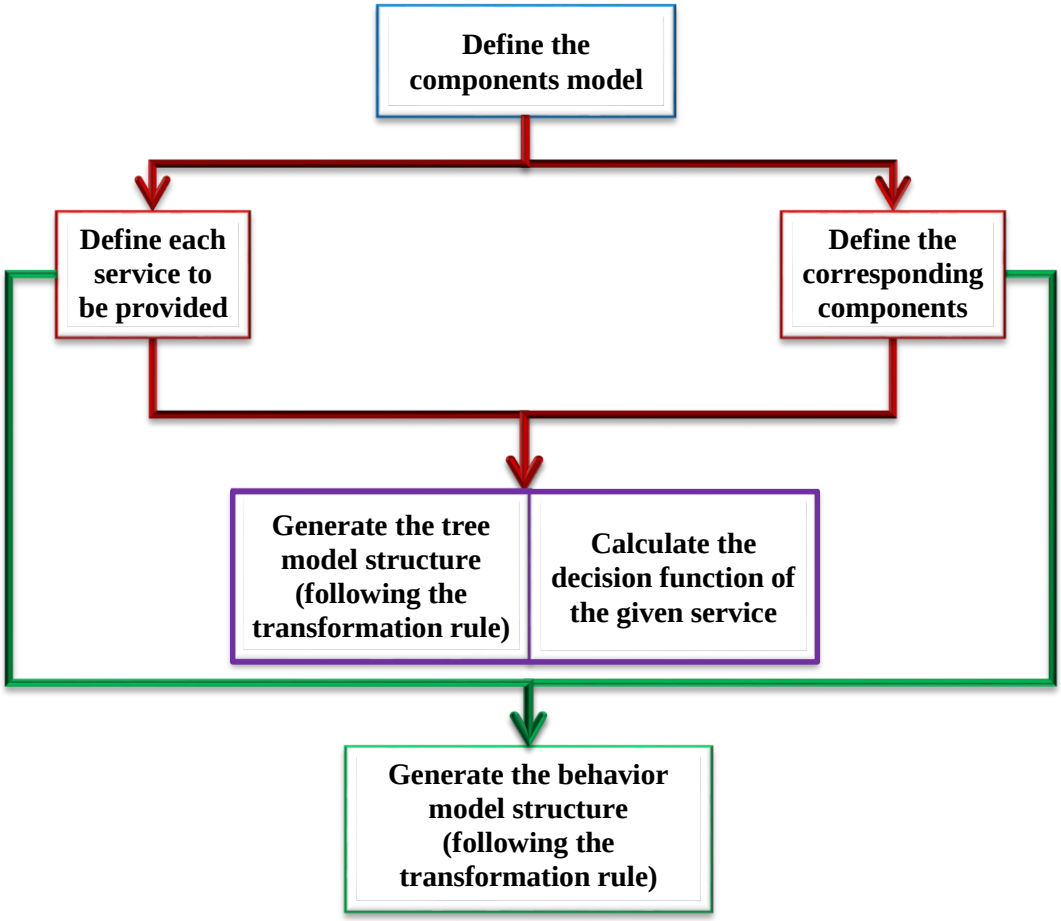


Figure 5.5. The implementation structure of the system

The main objective of the proposed verification framework is to check the correctness of the transformation rules and, in turns, the viability of the workflow in representing the system (with the fault tolerance mechanism) in an acceptable way from the user's point of view.

The proposed conceptual verification approach is coupled with a representative example from the Living Lab in order to check the correctness of the implemented tasks. The following subsection gives a brief introduction of the Living Lab environment followed by the illustration of the proposed example.

5.2.2.Experiment'Haal

In order to establish the verification experiment, the proposition is to define a framework, which is compatible with a real situation environment. In Lab-STICC, different platforms have been developed to conduct researches and projects in the domain of assisted living.

Experiment'Haal is a platform allowing accommodating the Lab-STICC projects related to personal assistance systems. Experiment'Haal is located in the Telecom Bretagne, campus of Brest, and it was inaugurated in 26-11-2012.

To foster an interdisciplinary approach, the laboratory Experiment'Haal intends to develop and to host experimental assistive technologies for usage tests. This platform can be used by academic or institutional stakeholders for experimentation and observations of habitability and acceptability.

This apartment-laboratory includes three rooms, covering an area of 50 m², including [experiment-Haal, 2016]:

- A room representing the residential housing including kitchen, bathroom, living room, dining room, hosts furniture, sensors, and actuators;
- A room of 15 m² is considered as the technical room that hosts the servers and workstations;
- A room of 24 m² is dedicated to the usability tests.

Figure 5.6 illustrates the apartment-laboratory.

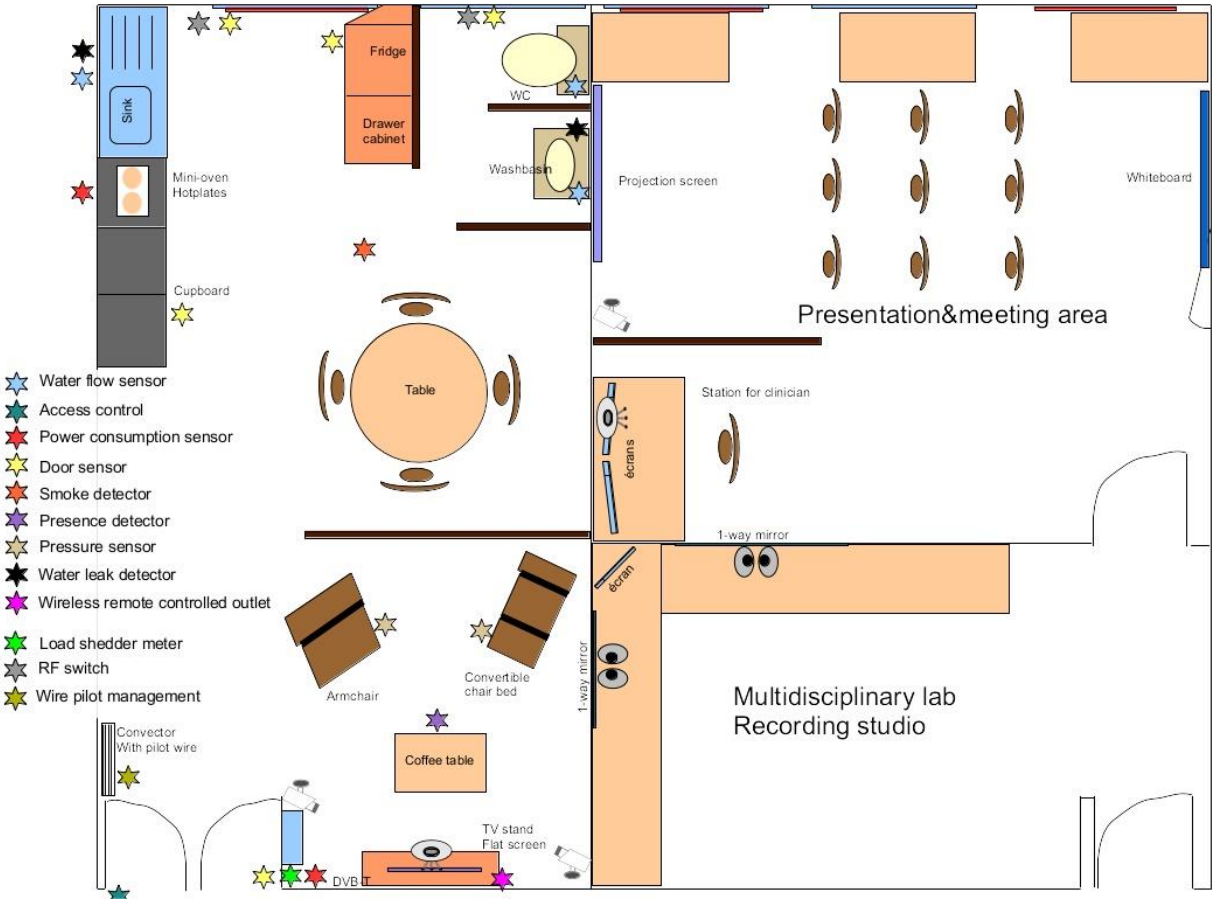


Figure 5.6. Experiment'Haal living space

Particularly, it allows the engineers to work with sociologists, ergonomists and computer scientists to test new products and services in the home automation domain, via the observation of practices in real conditions.

5.2.3.Illustrative example

In order to conduct the verification experiments, several basic examples were generated, and the resulting fault tree model structures and the behavior models are checked, in terms of the correctness of transformation rules.

In this section, a concrete example representing a realistic home automation system composed of several services using 8 equipments is proposed. This example is extracted from the living lab Experiment'Haal. This example can be extended to include several services and a greater number of basic equipments (i.e. components).

In fact all the used equipments (belonging to the platform) are represented in terms of components modeled as the first phase in the system workflow.

It is worthwhile to recall that (as it is the proposed workflow detailed in chapter 3):

- Each component may be a principal component and an alternative component for one or several services.
- Each component may be simultaneously used by several services.

This simply means that different components are assumed to be service independent.

In figure 5.7, a tree structure representation of the proposed system is given.

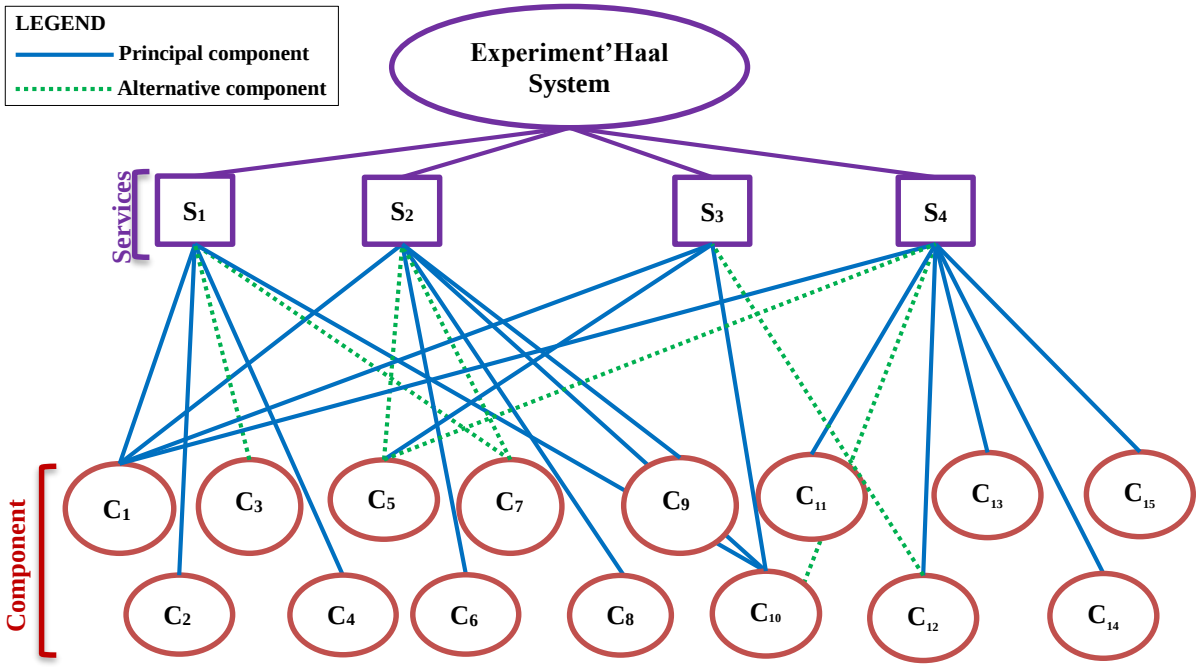


Figure 5.7. The Home automation system example

In this example, the basic equipments (i.e. components) used in the system, independent from the services, are given as follows:

C_1: *Presence detector*.
C_2: *Living room light*.
C_3: *Lampadaire.*
C_4: *TV*.
C_5: *Dining room light*.
C_6: *Bathroom light.*
C_7: *Window blind*.
C_8: *Water leak detector in the bathroom*.
C_9: *Water flow sensor in the bathroom.*
C_{10}: *Main power consumption sensor.*
C_{11}: *Power consumption sensor in the kitchen.*
C_{12}: *Kitchen Light.*
C_{13}: *Water leak detector in the kitchen.*
C_{14}: *Water flow sensor in the kitchen.*
C_{15}: *Smoke detector.*

Moreover, four services are considered in this example:
S_1: *Watching TV in the afternoon*.
S_2: *Using the washbasin*.
S_3: *Have a dinner*.
S_4: *Cooking the dinner*.

Each service in the system can be expressed using a tree structure representation.

For instance, the potential tree structure of the service S_1 "*Watching TV in the afternoon*" is illustrated in figure 5.8. This service uses the following components: S_1 = *{C_1: Presence detector, C_2: Living room light, C_4: TV, C_{10}: Main power consumption sensor}* as principal components and the components *{C_3: Lampadaire, C_7: Window blind}* as alternative components for the component C_2 (*Living room light*).

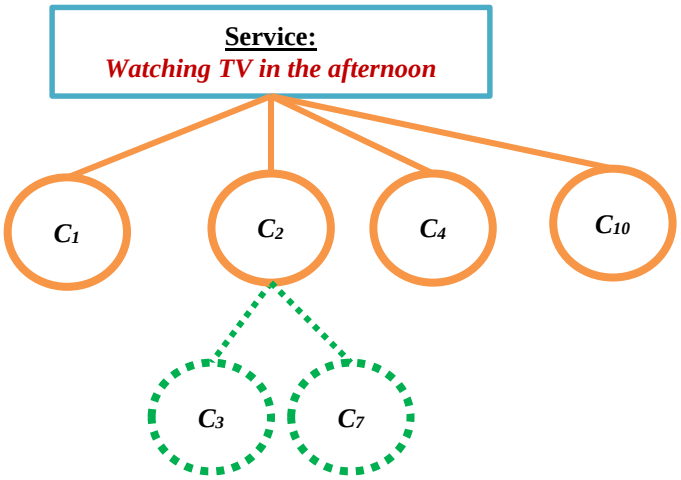


Figure 5.8. The service Watching TV in the afternoon

Similarly, the potential tree structure of the service S_2 *"Using the washbasin"* is illustrated in figure 5.9. This service uses the components: S_2 = *{C_1: Presence detector, C_6: Bathroom light, C_8: Water leak detector in the bathroom, C_9: Water flow sensor in the bathroom, C_{10}: Main power consumption sensor}* as principal components and the components *{C_5: Dining room light, C_7: Window blind}* as alternative components for the component C_6 (*Bathroom light*).

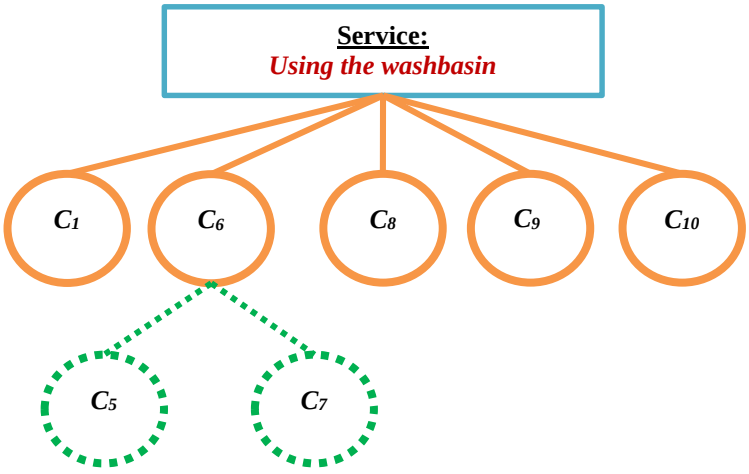


Figure 5.9. The service "Using the washbasin"

In the same manner, we can represent, as illustrated in figure 5.10, the services S3 "Have a dinner" which uses the components: S_3 = {C_1: *Presence detector*, C_5: *Dining room light*, C_{10}: *Main power consumption sensor*} as principal components and the component {C_{12}: *Kitchen Light*} as the alternative component for the component C_5 (*Dining room light*).

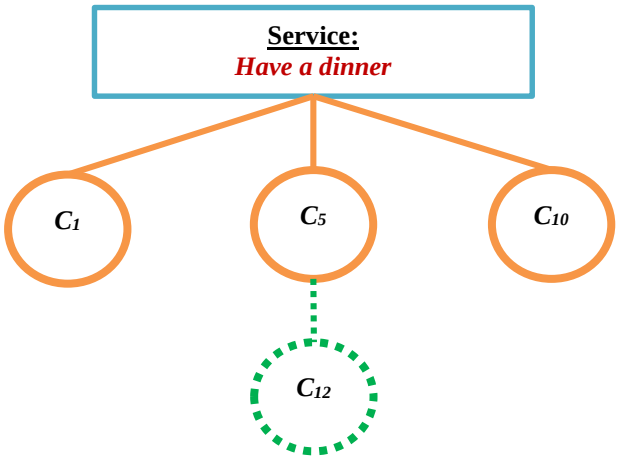


Figure 5.10. The service "Have a dinner"

The last service S4 "Cooking the dinner", which is illustrated in figure 5.11, uses the components: S_4 = {C_1: *Presence detector*, C_{11}: *Power consumption sensor in the kitchen*, C_{12}: *Kitchen Light*, C_{13}: *Water leak detector in the kitchen*, C_{14}: *Water flow sensor in the kitchen*, C_{15}: *Smoke detector*} as principal components and the component {C_5: *Dining room light*} as the alternative component for the component C_{12} (*Kitchen light*) and the component {C_{10}: *Main power consumption sensor*} as the alternative component for the component C_{11} (*Power consumption sensor in the kitchen*).

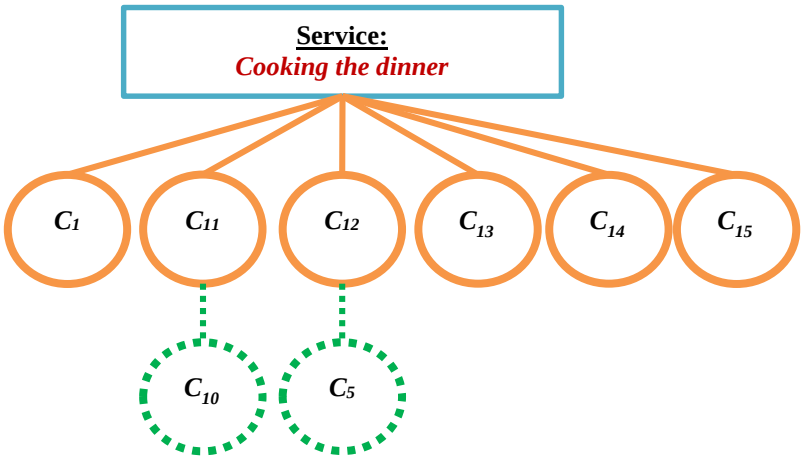


Figure 5.11. The service "Cooking the dinner"

To implement this example, the expert designer applies the following tasks:

A) System components models definition

The first task, conducted by the expert designer, consists on the model definition of all the system's components (not only the components of the considered service). The model definition of

each component includes: the component ID, Name, Functional operations, Location and the "proposed" failure probability value.

The XML model of the component C_2: *"Living room light"* is illustrated in figure 5.12. Moreover, the XML model representation of each component is illustrated in appendix (A).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>2</id>
    <name>Living_room_light</name>
    <failure_probability>0.01</failure_probability>
  - <operation>
      <operation1>ON</operation1>
      <operation2>OFF</operation2>
    </operation>
  - <location>
      <X_coordinate>1.5</X_coordinate>
      <Y_coordinate>0.9</Y_coordinate>
      <Z_coordinate>2.75</Z_coordinate>
    </location>
</component>
```

Figure 5. 12. The XML model of component C_2

B) System services model definition

In the next step for the example implementation, the expert designer defines the service that needs to be activated. The service model includes: the service ID, Name, Context, Time, Environment, Location and the corresponding activated components.

With the definition of each service of the system, the designer also establishes the relative model definition of the service activated components (principal and alternatives). This model includes the component priority, the component importance factor, the relative operation and the proposed alternative component(s).

The XML model of the first service S_1: *"Watching TV in the afternoon"* is illustrated in figure 5.13. Moreover, the XML model representation of the services and corresponding components are demonstrated in appendix (A).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
- <service>
    <id>1</id>
    <name>Watching_TV_in_the_afternoon</name>
    <context>sitting at room at afternoon</context>
    <time>afternoon</time>
    <environment>warm</environment>
    <location>[1.5, 1.0, 3.0]</location>
    <user_location>[1.5, 1.0]</user_location>
    <component>&presence</component>
    <component>&Light1</component>
    <component>&TV</component>
    <component>&power1</component>
</service>
```

Figure 5. 13. The XML model of service S_1

C) The design model-to-the fault tree analysis model transformation rule

After the definition of the service and the corresponding components, as well as the potential alternatives, the design model-to-the analysis model transformation rule is applied. The objective is to generate the analysis model, the fault tree model, to evaluate and calculate the decision function related to the service scenario (based on the main components), and to make the calculation of the decision function of the service alternative scenarios (based on the substitution of the components by the predefined alternative ones).

The ultimate goal is to verify whether the design model-to-the analysis model transformation rule generates the corresponding fault tree structure of the predefined scenario, with the correct information related to the service and components. This transformation rule should generate the tree structure illustrated in chapter 5.14 for the service S_3 *"have a dinner"*, according to the principal components *{*C_1: *Presence detector*, C_5: *Dining room light*, C_{10}: *Main power consumption sensor}*, and the alternative component *{*C_{12}: *Kitchen Light}*.

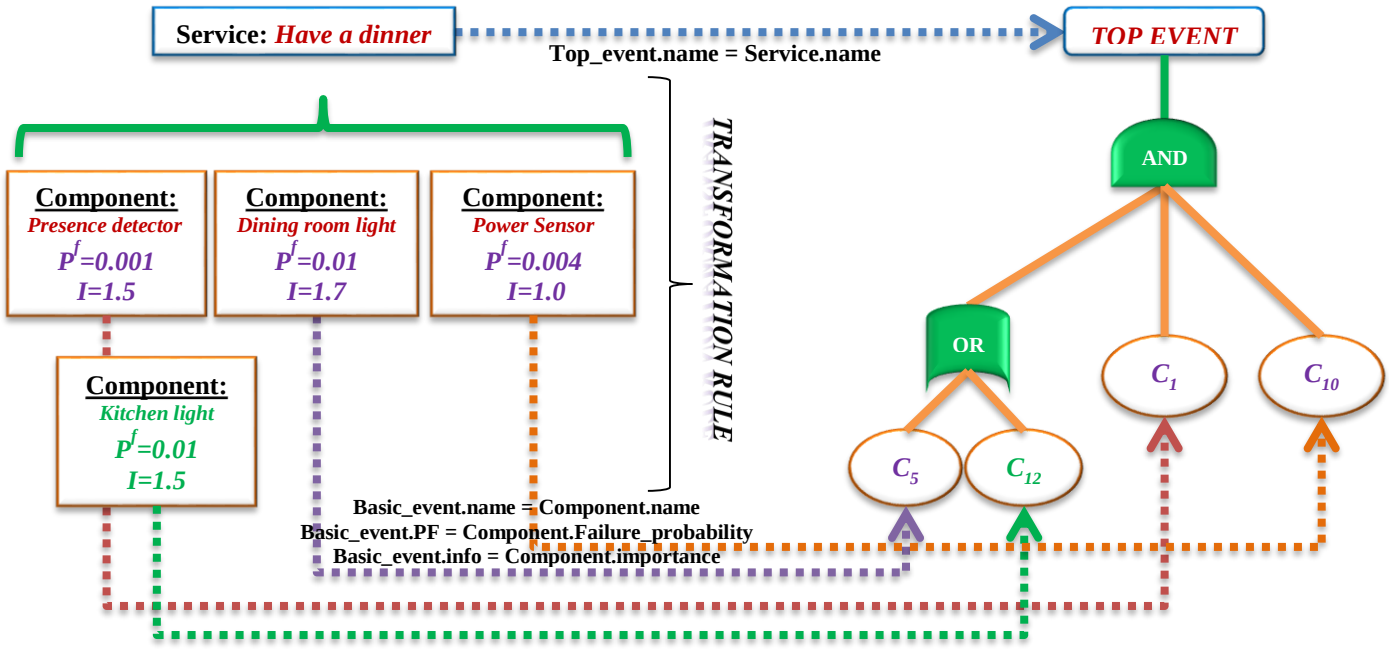


Figure 5.14. Transformation rules on the service

The transformation rule applied on the service generates the fault tree structure illustrated in figure 5.15. The resulting tree structure is in full harmony and is compatible with the rules illustrated in figure 5.14.

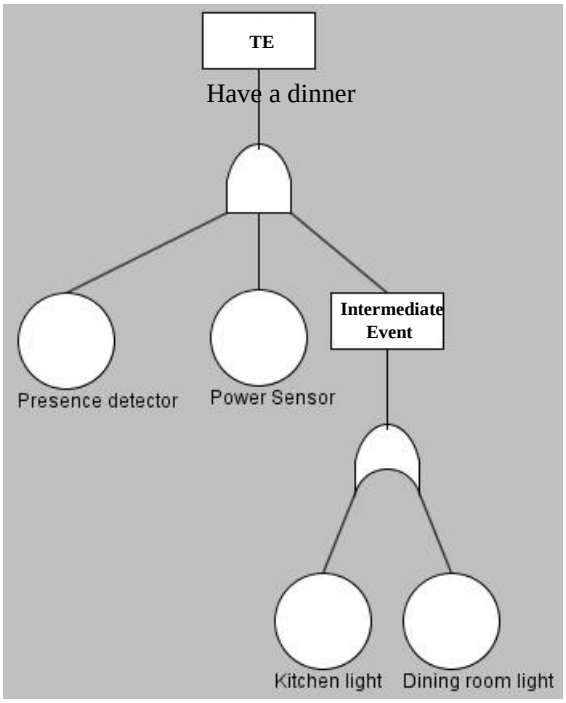


Figure 5.15. The generated fault tree structure of the service "Have a dinner"

D) The design model-to-behavior model transformation rule

The behavior of the system will be simulated using the PetriNet simulator in order to check that the service is successfully executed whatever the scenario followed.

The design model-to-behavior transformation rule is applied to generate the behavior model allowing, thus, to simulate the service execution scenario (based on the main components) and to ensure the execution of the service alternative scenarios (based on the substitution of the components by the predefined alternative components).

Therefore, the goal is to verify whether the transformation rule generates the corresponding behavior model structure of the predefined scenario (with the correct information related to the service and components, both principals and alternatives).

Using the considered illustrative example, this transformation rule generates the PetriNet structure illustrated in figure 5.16, according to the principal components, Dining room Light, Presence detector and Main power consumption sensor, as well as the alternative component, Kitchen Light.

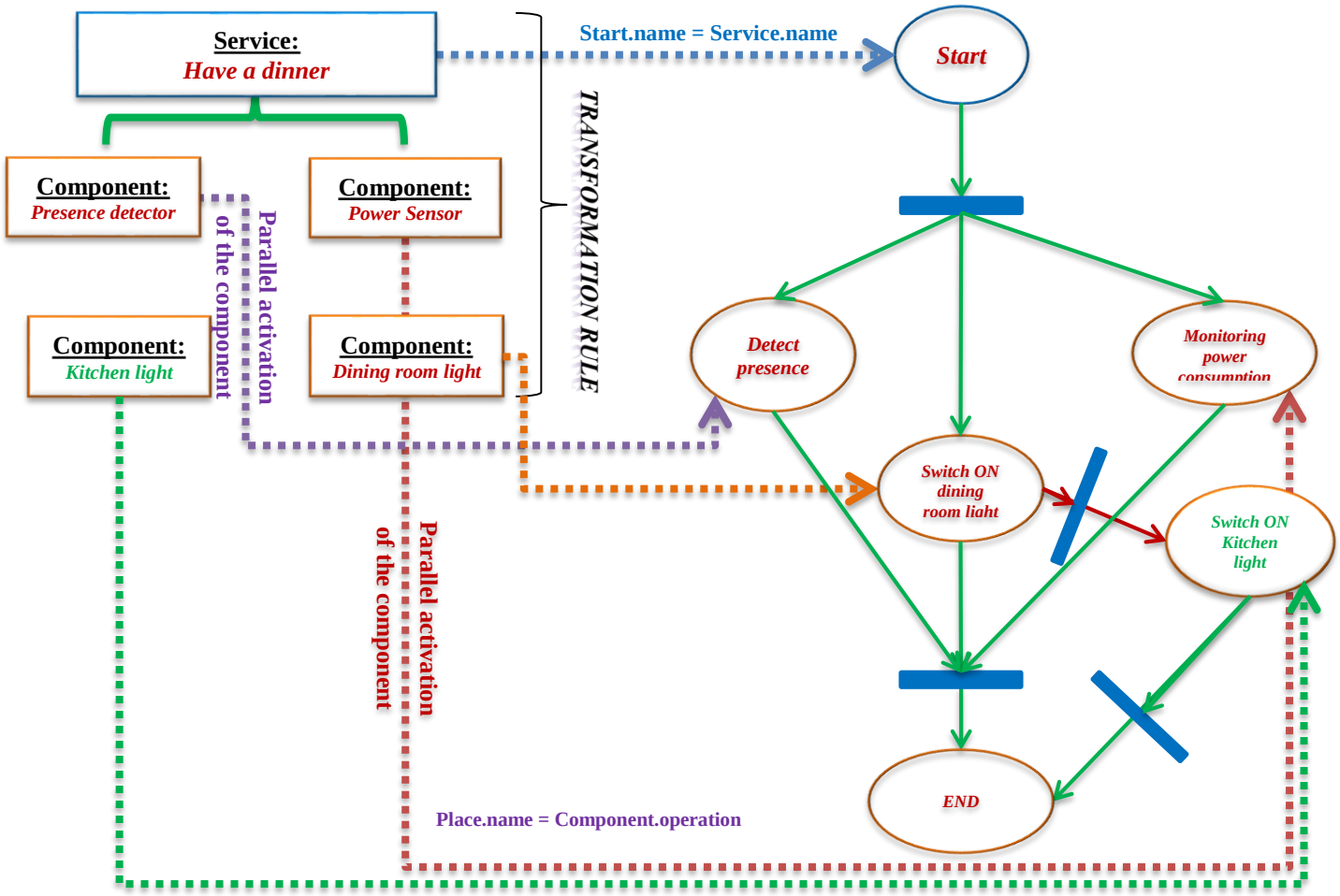


Figure 5.16. The generated PetriNet structure of a service

The behavior model illustrated using the normal form of Petri Net could be used for demonstrating the execution of the service scenarios.

When using the Stochastic Petri net form, it will be more significant to demonstrate how the components probabilities of failure may delay the firing of different transitions, which will influence the behavior of service execution (as detailed in chapter 3).

When the firing delay is increased as a ratio of the probability of failure, we assume the failure of the corresponding component (given by the preceding place), thus we select the alternative component (given by the alternative place) to guaranty the continuity of the service execution.

For the verification of the behavior process, we have dealt with the PetriNet simulator tool to represent the behavior model and conduct the simulation process. When we aim to realize a simulation of the behavior model, after the proposition of the Stochastic Petri net form, another tool may be used for this purpose. Such a tool is TimeNet 4.1 which is used as a package for the modeling and performance evaluation of standard and colored stochastic Petri nets [A. Zimmermann. 2012].

After the verification of implementation structure and showing that the workflow is correct and sounds right, i.e. delivering the right product, in the next section, we will open the door for a proposition of validation framework to address the question: “Are we building the right product?”

5.3. Proposition of validation framework

The target objective of the validation framework is to verify two main issues: the system meets the users' satisfaction and that the system is capable of resisting to unexpected events (the occurrence of errors and faults in the runtime context) which may breakdown the whole system functioning. In the domain of software engineering, there is no general purpose validation approach. Several validation methodologies are proposed in the literature. We can even say that each system suggests its own validation protocol.

In this section, the proposed validation approach intends to tackle the continuity of the system services in the framework of Home Automation reconfigurable system.

Generally speaking, a validation experimental framework in a home automation reconfigurable system should be defined according to:

- *The purpose of the considered system:*

The purpose of the considered home automation reconfigurable system is to resist to potential failures of the system components ensuring, thus, the services deliverance continuity towards user satisfaction;

- *objectives of the proposed validation:*

Evaluate the effectiveness of the proposed approach to carry out the purpose of the considered system. Recall the fact that the key objective of any validation process is to show that each modeling method is self-consistent. If there are several models of the system, then, the objective of the validation process consists on demonstrating that these models are internally and externally consistent, capable of and are suitable for their intended purposes.

The validation of fault tolerance mechanisms is often conducted by injecting faults that emulate the actual faults and ''stress'' the functionality of the resilience mechanisms [Hsu *et al*, 2010]. Thus, the validation protocol mainly should consist of the injection of faulty components into the system, and evaluating the response of the proposed system to face the potential failure (s).

In this research study, the proposed objectives of the validation approach are the following:

1. ***Single fault tolerance validation***: By injecting a faulty component to the system, the latter is assumed to pursue performing the correct way (using the proposed fault tolerance mechanism) and to tackle the leading failure and guaranty its continuity;

2. ***Multiple fault tolerance validation***: The system reaction to "multiple" faulty components should be checked. This objective intends to study the resistance of the proposed system when successive components failures occur.

3. ***Expert's importance factor tuning***: This validation objective intends to tackle the issue of the importance factor values determination by the expert. In fact, as already mentioned in the previous chapters, each component, in the system, is characterized by two features: its failure

probability, and its importance factor. The failure probabilities are "imposed" features and the expert designer has no influence on these values. On the contrary, the importance factors are "injected" by the expert designer (based on multiple targeted objectives). Nevertheless, an importance factors tuning phase is proposed in the validation framework, allowing to the expert designer to adjust these values through the observation of the alternative scenarios proposed by the system.

4. ***Dynamic importance factor tuning***: Throughout all this study, the use of the importance factors is proposed in a static way (i.e. once the values are defined by the expert designer, these values remain unchanged even when one or several components fail). It sounds more logical that when a component fails, then, the importance factors should be dynamically adjusted in order to allow to the system to react to potential other components failures (awaiting for the failure component change, and returning thus to the initial system configuration).

In order to realize the proposed objectives, the proposed validation framework consists on considering the global home automation system delivering multiple services and represented as a tree structure with three hierarchical levels: components, services and system (as illustrated in figure 5.17). This system tree structure is assumed to be generated by the design model-to-analysis model transformation task.

At this point of system structuring, the intermediate events are discarded for the validation purpose. In the context of this study (from a practical point of view), the system to be modeled should be the Lab-STICC apartment-laboratory Experiment'Haal system platform.

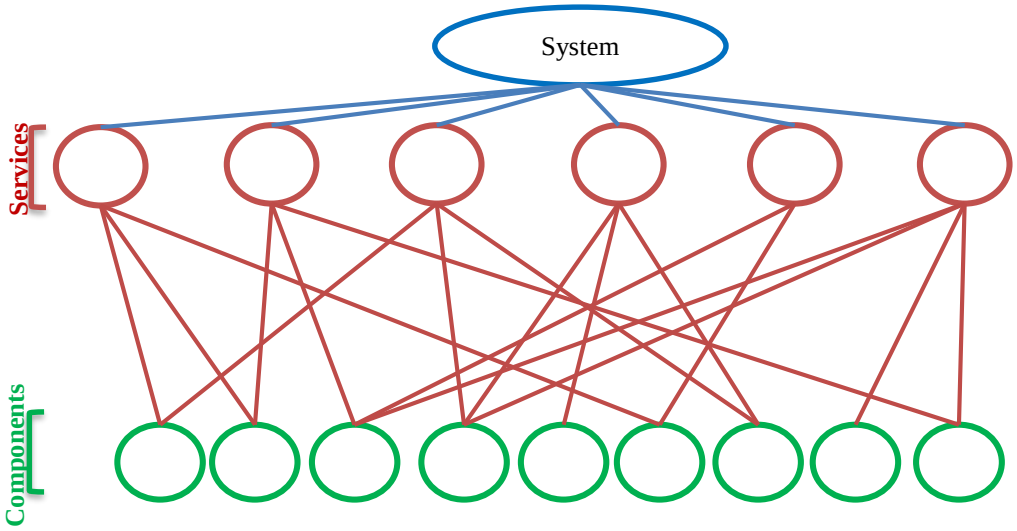


Figure 5. 17. The Home automation system tree structure

1. ***Single fault tolerance validation***:

The validation framework should prove the fault resistance, in terms of the capability of each service to interact in correct way to tolerate the fault to guaranty its continuity. In order to validate this objective, the proposed validation approach consists on the sequential injection of a faulty component and to evaluate the capacity of the fault tolerance applied mechanism in tackling the induced failure. The proposed single fault tolerance validation algorithm, at the level of each service, is given as illustrated in algorithm 5.1:

For a service S_m (m=1, …, Nb-of Services)**:**
  For each component C_n (n=1, …, Nb-of components used by S_m)
    C_n***_status = Fault***
    For each component (*C`, C`` …*) in the system:
      Determine the alternative component
      C`.context == Cn.context
      C`.location near C.location
      Redefine Design (S_m**)**
      Regenerate_tree(S_m**)**
      Calculate decision function (S_m**)** $f_{C`}(f_{C``})$
      Decide which scenario to follow according to the values of $f_{C`}(S_m)$, $f_{C``}(S_m)$**…**
  Test of service S_m **deliverance**

Algorithm.5.1. Single fault tolerance validation

The regenerated tree structure refers to the best scenario to substitute the faulty component and ensure the continuity of the service. Following the design model –to- analysis model transformation rule, this process will take place to replace the faulty component and regenerate the design model, thus the tree structure model (the good functioning of these steps has been checked in the system verification phase).
If the service S_m pursues performing the correct way after the sequential injection of all faulty components, then the service S_m is said to be "Single Fault Tolerance validated".
It is important to notice that this validation objective is crucial. In fact, if the service deliverance is not guaranteed for any component, then, the expert designer should reconsider the design of the whole service.

2. Multiple fault tolerance validation

When a component fails, the fault tolerance mechanism ensures the service deliverance continuity. Nevertheless, the faulty component should be replaced, and once the component replacement is realized, the system is reinitialized.
In practice, the time spent between the fault occurrence and the faulty component replacement may be important. During this wait-for-replacement time, other components may fail. At this level, the important issue of the system's resistance to this multiple failures has to be tackled. It is important to notice that this validation objective has to be positioned at the system level (not at the services level).
Assuming that the system is composed of N basic components, then,

- the number of Two-component failures is given by N (N-1);
- the number of Three-component failures is given by N (N-1)(N-2);
- ….

Depending on the number of components constituting the global system (i.e. N), two validation strategies can be considered: Exhaustive validation strategy, and, Random validation strategy.
The exhaustive strategy is applied when the number N is reasonable. It consists on the exhaustive test of all possible configurations of multiple components failures. For instance, if N=10, then the number of Two-component failures to be checked is 90. Similarly, the number of Three-component failures to be checked is 720….
The Random validation strategy is applied when the N is high enough to make the number of potential multiple components failure extremely high. For instance, if N= 100, then, the number of Two-component failures to be checked is 9900. Similarly, the number of Three-component failures to be checked is 970200…. Facing this important number of failure configurations, it sounds reasonable to randomly pick up a predefined number of failure configurations and to test the system services continuity limited to this predefined number.
For both validation strategies, if the system services continuity is verified for the considered multiple faulty components configurations, then the system is said to be 2-Fault tolerant, 3-Fault tolerant, etc.
The proposed multiple fault tolerance validation algorithm (K faulty components) at the system's level is given as illustrated in algorithm 5.2:

For a system *SY*:
Generate K faulty components C_{k_1}, C_{k_2}, …, C_{k_K}
(Following the Exhaustive or the Random strategies)
For (i=1, 2…K)
    C_{k_i} *.status = Fault*
    Determine the alternative component to C_{k_i}
    If (C`.context0 == C*n*.context) and (C`.location near C_{k_i}***.location)***
    For all services (*S*m) using C_{k_i}
        Redefine Design (*S*m)
        Regenerate_tree(*S*m)
        Calculate decision function (*S*m)
        Decide which scenario to follow
    End
End
For all services S_p composing the system *SY*
    Test of service S_p **deliverance**
End
If all services S_p composing the system *SY* have successful deliverance
    Then (The system *SY* is K-Fault tolerant)

Algorithm.5.2. Multiple fault tolerance validation

3. *Expert's importance factor tuning*

As previously mentioned, each component used in the system by the proposed fault tolerance approach is characterized by two features: its failure probability and its importance factor.
The failure probabilities are intrinsic features for which the expert designer has no influence.

On the other hand, the importance factors are expert-defined features where the expert is assumed to initialize the fault tolerance approach by "empirically" selecting adequate values for each component of the system. Nevertheless, this empirical values selection needs a tuning phase allowing to the expert designer to refine and to adjust his own values selection. This tuning phase is proposed to be conducted, at the service level, through the validation framework using the algorithm.5.3:

For a service S_m (m=1, …, Nb-of Services):
  For each component C_n (n=1, …, Nb-of components used by S_m)
    C_n_*status = Fault*
    For each component (*C`*, *C``* …) in the system:
      Determine the alternative component
      C`.context == C_n.context
      C`.location near C.location
      Redefine Design (S_m)
      Regenerate_tree(S_m)
      Calculate decision function (S_m) $f_{C`}$ ($f_{c``}$)
      Decide which scenario to follow according to the values of $f_{c`}(S_m)$, $f_{c``}(S_m)$…
      If (the selected scenario does not fit the expert's opinion)
        Then (Allow the expert to adjust the importance factor values)
    End
  End

Algorithm.5.3. Expert's importance factor tuning

To illustrate this validation objective, let us consider the example of the service "going to the bedroom" with two principal components, "door1" (of the bedroom) and the "ceiling light1" of the bedroom. The ceiling light has only one alternative component corresponding to the "wall light1" in the bedroom.

The corresponding service has the following parameters:
S: *Going to bedroom*.
C_1: *Door1* → P^f = 0.005, I = 1.2
C_2: *Ceiling light1* → P^f = 0.1, I = 1.5
C_2`: *Wall light1* → P^f = 0.05, I = 1.2

In the case of the failure of C_2, the system will propose to use the component C_2` instead, to substitute the faulty component, as illustrated in figure 5.18. In this case, because the component $C_2$` "wall light1" has no alternative, its importance factor should be increased from 1.2 to 2.0 (for instance).

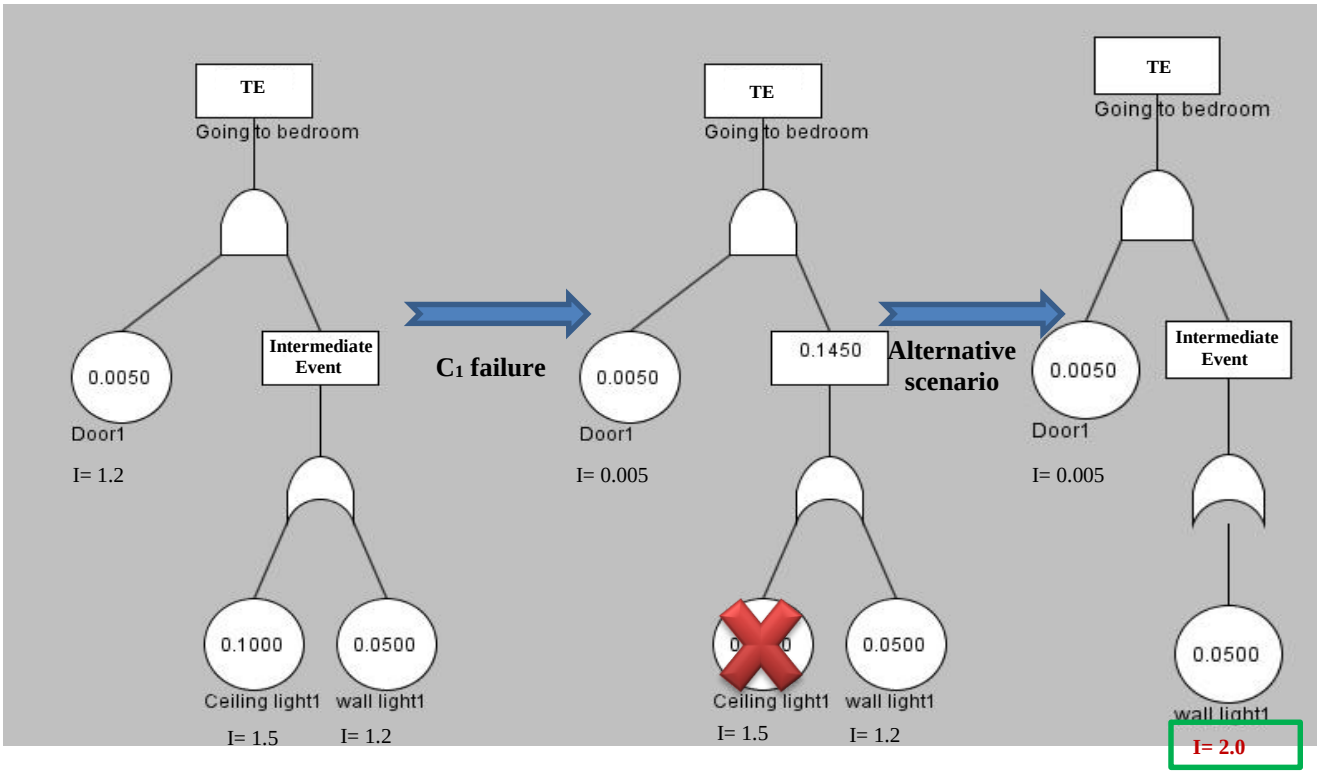


Figure 5.18. An alternative scenario changing

Thus, this validation objective needs to be achieved in order to make the system design consistent and efficient.

4. Dynamic importance factor tuning

This validation objective is in between the validation issue and the prospective research issue. In fact, the expert's importance factor tuning objective focuses on tuning the importance factor in order to select the optimal alternative scenario when a single component fails.

Nevertheless, it can be clearly understood that when a component fails in the home automation system, the whole set of importance factor values (attributed to non-faulty components) should be actualized in order to take into account the system functioning with a faulty component.

The reason behind raising this issue is that the time for the replacement of a faulty component may be important. Nevertheless, when the faulty component is replaced by the expert, the initial set of importance factors will be used again.

As a simple example of this issue, we can imagine a power source adopted as the alternative power source of another service. This power source will thus be shared by two services. Therefore, its importance factor cannot remain unchanged.

Two major approaches can be considered for the dynamic importance factor tuning: Empirical tuning and Rule-based tuning approaches.

The empirical tuning dynamic importance factor approach consists on the definition of the set of importance factors to be newly used when a component fails. Figure 5.19 shows an example of how an empirical tuning dynamic approach can be defined.

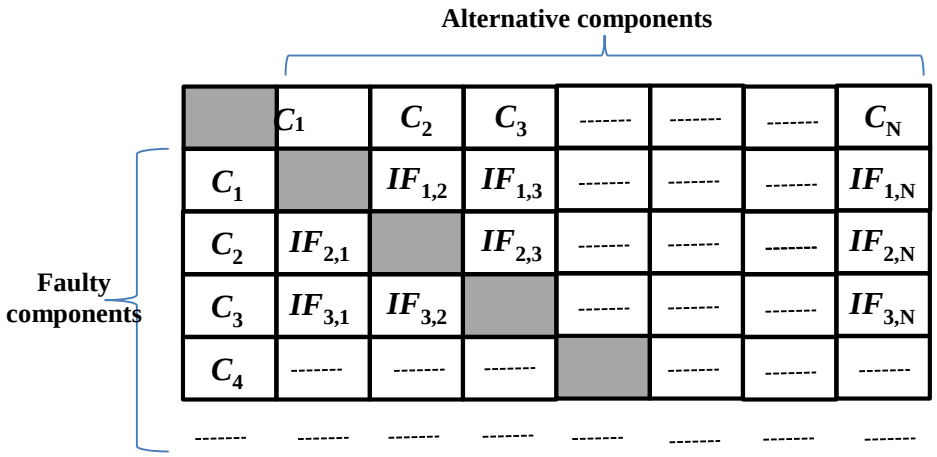


Figure 5.19. The empirical tuning dynamic importance factor ($IF_{n,k}$ refers to the new importance factor affecting the component C_k when the component C_n fails)

On the other hand, the Rule-based dynamic tuning approach consists on defining a set of rules allowing a dynamic computation of the new importance factors corresponding to the expert's vision when a component fails.

As it can be witnessed, it seems easier to use the empirical tuning dynamic approach. Nevertheless, it is important to recall the major limit of this approach which is the empirical approach can be used only when two components fail (i.e. it is not adapted to the case of multiple components fail, over than two).

5.4. Conclusions

In this chapter, an experimental framework allowing the workflow validation of the system (with the consideration of fault tolerance mechanism) has been introduced.

After the presentation of an introduction of the validation and verification concepts, the conceptual verification framework is detailed, based on the proposed and developed framework presented in chapter 3.

This verification framework is coupled with a simple example to illustrate how the verification could be achieved in real environment.

As perspectives concerning the real validation of the conducted work, the validation framework proposed, in this chapter, opens the door to develop several validation protocols. In fact, four major validation objectives are defined in this chapter and a validation protocol allowing conducting each validation objective is proposed.

These protocols have to be implemented and evaluated in a real living lab. This open proposition, could be done by evolving an implementation structure to inject, for each experiment, set of faulty components, to check how the transformation rules would regenerates the analysis tree structure model, as well as the behavior structure model, which will ameliorate the design structure of system elements, towards an optimal representation of the system, from the user's point of view.

Workflow verification and proposed experimental validation

Chapter 6

Conclusions and perspectives

This chapter globally summarizes the work carried out in this thesis. Starting from the requirements, this chapter goes through the proposed contributions, discusses the appropriateness to the context and ends by highlighting the perspectives that could enrich the work for future works. The first section makes a summary by going back over the research context and objectives, recalls the contributions and emphasizes its adequateness in relation with the requirements. The second section shapes some opening perspectives of future scientific and technical development based on the proposed contribution.

6.1. Conclusions

In this section, the thesis context and objectives are recalled. The proposed contributions, to address the raised objectives, are summarized as well as the obtained results are repositioned.

6.1.1.Context of the work

The ageing of the European population encouraged the community to search for solutions to support this evolution. In this context, several issues need to be addressed related to the expensive and limited health care services and health facilities capacities.

Thus, several projects, research and industrial solutions have been proposed to address these issues. Most of these solutions are based on the use of the latest ICT technical developments to provide dedicated services that ameliorate the well-being of the targeted ageing group and to guaranty their autonomy in their own living spaces.

6.1.2.Main objective of the work

In complex home automation systems, the solutions provided to the elderly people tend to offer assistive technologies to the target users, thus, these systems are described as “Ambient Assisted Living”, AAL, systems. These provided technological solutions mainly consist of interconnected hardware and software resources to form integrated automated home systems.

These systems are assumed to hold a set of predefined characteristics and criteria (functional and nonfunctional) to meet the users’ perspectives and satisfactions. Such characteristics could be related to the interoperability of the provided technologies, the adaptability to users’ needs, the acceptability, the accessibility, the maintainability and the availability in case of potential failures of some parts of the system.

The main addressed objectives in this work consist on realizing a state of art study to identify functional characteristics to insure the continuity of the provided services. Moreover, a design model is proposed. This model is assumed to allow the description of a system based on the needs in terms of

provided services continuity and, to conduct a system implementation to evaluate the services (and the related platform) in real situations.

The state of the art review has directed this research work to put the system reconfiguration requirement as being the key objective to be realized in this research work.

The reconfiguration issue, in terms of ensuring the provided system continuity (in case of potential failures occurrence), is an essential requirement for all Ambient Assisted Living systems that guaranty the continuity of services provided to the targeted users, and to secure their live and autonomy.

In this study, and to realize the system reconfiguration, the fault tolerance approach is proposed. This approach is based on the analyzed redundancy solution (which is the main mechanism usually applied to ensure the reconfigurability of the system).

6.1.3.Proposed contributions to reach the objectives

In order to address the predefined research objectives, several tasks have been identified for this goal. These tasks can be summarized in the following:

- The evaluation of home automation systems in order to identify the adopted requirements, in terms of ensuring the system's continuity;
- The proposition of an integrated modeling framework allowing to represent the system with the consideration of reconfiguration approach;
- The development of a workflow to represent home automation services that meet the reconfiguration requirement and developing a prototype representing the workflow processes;
- Proposing an experimental validation framework to validate the correctness of the proposed system representation.

Several contributions have been proposed to lead the research and to tackle the defined objectives:

6.1.3.1. Abstract model proposition of system elements in accordance with fault tolerance approach

This contribution tends to define an abstract formalization for the home automation system structure. This formalization targets to provide effective and efficient services and to ensure the continuity of the proposed services (through achieving and including fault tolerance mechanisms with the provided services). This formalization has been realized through the definition of the components, the operations, the alternative components and the concepts of component and service failure. Subsequently, these concepts have been clarified and the concept of the component importance factor for a particular service has introduced. A matrix representing the importance relationship between different service and component sets has been formulated.

Furthermore, the proposed approach was inspired from the use of possibility theory, which has been developed for modeling epistemic uncertainties (i.e. the same application context of probabilistic approaches), for which only subjective, or ambiguous, knowledge is available.

The main contribution of this task has been raised from the need to jointly use the proposed importance factor and the probability of good functioning in order to overtake the conflict use of these two notions (as well as to overcome the difficulty of having imprecise probability values). For this

reason, a risk factor has been introduced for each component in the three hierarchical levels defining each service. To resume, the importance factor has been modeled to be considered in the analysis process of the system structure.

6.1.3.2. Reinforce the notion related to components importance within a service

The evaluation of a system services success/failure was conducted using the failure analysis approach. The fault tree analysis (FTA) approach has been proposed to establish this proposition. In this contribution, the main proposition made in this thesis comprised the integration of the expert-defined events importance factors with the failures probability in order to enrich the Bayesian Fault Tree Analysis approach. The importance factor is "empirically" defined by the designer, and a new global decision function has been introduced. This function is supposed to rely on the basic events probabilities of failure as well as on the importance factors of all basic events. On important constraint (that we have fixed) concerning this decision function is that this function should be reduced to the classical maximum likelihood decision criterion when the expert designer decides to withdraw the use of importance factors (by considering all components having the same importance values). This global new decision function will influence the decision about which scenario the system will execute according to the importance of its constituting components and theirs failure probabilities values. In other words, the integration of importance factors within the decision function may change the replacement decision about which scenario will be followed to execute the service represented in the top event.

The key point in this contribution (that none of the existing approaches offers) consists on offering the possibility of using a user defined degree of importance (that each component holds for the success of the corresponding service) into the decision process. Additionally, the proposed approach has involved analyzing the influence of the use of the importance factor in the decision about the selected scenario, rather than the only use of the failure probability degrees.

6.1.3.3. Workflow representation of system design, analysis and behavior

This contribution consists on the development of a workflow allowing representing how the expert designer will be able to make abstraction of the system. The main idea of this contribution has been based on the ability to make a general framework of how the system will be designed, analyzed and its behavior will be presented, in a single framework. Moreover, this framework is assumed to allow the transition based on developed transformation rules, from the design stage to the analysis and the behavior stages, in order to evaluate the correctness and effectiveness of the system's structural design.

6.1.4.The prototype representation of the workflow

The main contribution, related to the technical implementation of the proposed workflow, consists on involving the fault tolerance mechanism to ensure the continuity of the constituting services and components in home automation systems. The key point of this contribution has included the development of a prototype representing the workflow phases, including the transformation rule from the system's design model into the system's fault tree analysis model, as well as the transformation rule from the system's design model into the system's behavior model.

The developed prototype allows to facilitate the evaluation of the system's model, by offering the evaluating capacity of the selected scenarios for each service in order to decide if a scenario is valid (or it needs to be enhanced/changed), based on the decision about the probability of failure of the constituting components as well the importance factor.

6.1.5.Experimental validation framework

An experimental framework has also been proposed. This framework allows the verification of the proposed system's workflow (with the consideration of the fault tolerance mechanism), as well as a proposition for the system's validation.

The verification of the implemented workflow phases has been conducted and illustrated using a representative example captured from a real living lab (Experimental' HAAL). The developed implementations have been introduced as a background of the validation process.

A proposition of a validation process has been proposed, based on the definition of several critical objectives summarized in the following:

- Single fault tolerance validation;
- Multiple fault tolerance validation;
- Expert's importance factor tuning;
- Dynamic importance factor tuning.

In order to realize the proposed objectives, several methodologies have been detailed based on the workflow processes and notions.

6.1.6.Obtained results

The keystone of the work presented in this thesis consists on the development of the reconfigurable home automation system's structure. This system structure has been developed and implemented, as a workflow, based on three modeling blocks and two transformation rules between these models:

- System structure design model.
- System analysis model.
- System behavior model.
- Design model – to – analysis model transformation rule.
- Design model – to – behavior model transformation rule.

These modeling blocks, as well as the transformation rules were implemented in Python framework and following the Model Driven Architecture (MDA) paradigm.

A new approach has been proposed based on the integration of the importance factor for a given component into the Bayesian Fault tree analysis approach, to evaluate the influence of components failure on the service functioning state.

6.2. Perspectives

The work conducted in this thesis opens many interesting perspectives that aim at investigating and answering questions that have not been yet addressed.

Concerning the workflow implementation, the implemented prototype using Python framework is developed in simple way, following the MDA approach and the object oriented programming paradigm, and it could be available for reuse and any future modifications or development. In the implemented workflow, the basic components are only considered to be presented, thus, it could be wise to consider (in the future developments) to evaluate the use of composite components (for instance a door + the ceiling light of adjacent room) to be considered in the implemented workflow.

In the same direction, the components' probability of failure has been proposed as a given precise value. In fact, these values are estimated by the technologies vendors based on different calculation issues, such as the "mean time to failure", the "mean time between failures" or the maximum consumption hours, etc. As it was indicated in the previous chapters, several promising studies have proposed to consider imprecise probability values and have suggested the use of possibility theory in order to overcome the problem of imprecise probability values. These studies did not consider any research direction in order to consider the expert's predefined importance factors. Therefore, it seems promising to pursue the proposed approach (integrating precise failure probability values and importance factors) into the case of imprecise failure probabilities jointly used with the expert's defined importance factors.

Concerning the proposed validation framework, several living labs are already established in Lab-STICC for the purpose of home automation systems. Thus, it is of great importance to lead a real experimental validation of the proposed methodologies (in chapter 5) on one or several living labs, such as the experimental' HAAL in Brest or the apartments of "centre de Kerpape" in Lorient.

Moreover, the proposed validation methodology could be experimented through some established platforms such as the XAAL bus solution. For instance, we can experiment the scalability of the proposed validation framework.

The simulation behavior model could be experimented to demonstrate, in real simulation, the scenario substitutions of the services' execution.

Another interesting perspective consists on leading a theoretical study concerning the automatic determination of different components importance factors in the FTA approach. This study may concern the static importance factors determination (i.e. determining these factors as a one shot); as well as the dynamic importance factors determination (i.e. automatic determination of these factors when some components become faulty ones).

Finally, the proposed dynamic reconfiguration approach (detailed in chapter 2) could be adopted in the future work to be included in the hybrid Bayesian approach to select the best alternative components, based on the use of components' Meta Data, in addition to the importance factor. The perspective is to develop an algorithm allowing calculating the degree of similarity between relative components. Thus, the selection of alternative component will depend on three features:

- The probability of failure;
- The importance factor;
- The Meta Data to compute the similarity.

Bibliography

[AALJP, 2016] Ambient assisted living joint programme. http://www.aal-europe.eu/about/

[Aghie and Saeedi, 2009] Aghie, A. and Saeedi, A. (2009). Using Bayesian Networks for Bankruptcy Prediction: Empirical Evidence from Iranian Companies. International Conference on Information Management and Engineering.

[Allègre *et al*, 2011] Allègre, W., Burger, T., and Berruet, P. (2011). Model-Driven Flow for Assistive Home Automation System Design. Proceedings of the 18th International Federation of Automatic Control (IFAC) world congress.

[Allègre, 2012] Allègre, W. (2012). Flot de conception dirigé par les modèles pour la commande et la supervision de systèmes domotiques d'assistance. PhD thesis 2012, UNIVERSITE BRETAGNE SUD, Lorient.

[Anurag *et al*, 2014] Anurag, S., Manoj, J., and Qureshi, M.F. (2014). Reliability Investigation of Series-Parallel and Components of Power System using Interval Type-2 Fuzzy Set Theory. International Journal of Innovative Research in Science, Engineering and Technology. Vol. 3, Issue 11.

[Aven, 2011] Aven, T. (2011). A risk concept applicable for both probabilistic and non-probabilistic perspectives. Safety Science; 49:1080–1086.

[Avizienis *et al*, 2004] Avizienis, A., Laprie, JC., Randell, B., and Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on dependable and secure computing, VOL. 1, NO. 1, January – March 2004.

[Barry, 1989] Barry, B. (1989). Software Risk Management. Proceedings of 2nd European Software Engineering Conference. ESEC'89. LNCS. pp. 1–19. doi:10.1007/3-540-51635-2_29. ISBN 3-540-51635-2. ISSN 0302-9743.

[Belabbas *et al*, 2006a] Belabbas, A., Berruet, P., Rossi, A., and Philippe, JL. (2006a). A modeling approach to control a handicap technical assistance system. WSEAS Transactions on Information Science and Applications, vol. 3, no. 8, pages 1460–1467.

[Belabbas *et al*, 2006b] Belabbas, A., Berruet, P., Rossi, A., and Philippe, JL. (2006b). Cooperative model generation for disabled people assistance. In Proceedings of the 5th WSEAS international conference on Telecommunications and informatics, pages 438–445.World Scientific and Engineering Academy and Society (WSEAS).

[Belabbas, 2007] Belabbas, A. (2007). Méthodes de conception de systèmes contrôle/commande reconfigurables en présence d'aléas: application au handicap. PhD thesis, Laboratoire d'Électronique des systèmes Temps Réel, Université de Bretagne-Sud, 2007.

[Ben-Gal, 2007] Ben-Gal, I. (2007). Bayesian Networks (PDF). In Ruggeri, Fabrizio; Kennett, Ron S.; Faltin, Frederick W. "Encyclopedia of Statistics in Quality and Reliability". Encyclopedia of Statistics in Quality and Reliability. John Wiley & Sons. doi:10.1002/9780470061572.eqr089. ISBN 978-0-470-01861-3.

[Berthomieu *et al*, 2004] Berthomieu, B., Ribet, P.-O., and Vernadat, F. (2004). The tool TINA - Construction of Abstract State Spaces for Petri Nets and Time Petri Nets, International Journal of Production Research, Vol. 42, No 14.

[Bézivin, 2004] Bézivin, J. (2004). In search of a basic principle for model driven engineering. Novatica Journal, Special Issue, pages 21–24.

[Bézivin, 2005] Bézivin, J. (2005). On the unification power of models. Software and Systems Modeling, vol. 4, no. 2, pages 171–188.

[Booch *et al*, 2005] Booch, G., Rumbaugh, J., and Jacobson, I. (2005). Unified Modeling Language User Guide, the (2nd Edition). Addison-Wesley, p. 496.

[Borgonovo, 2007] Borgonovo, E. (2007). Differential, criticality and Birnbaum importance measures: An application to basic event, groups and SSCs in event trees and binary decision diagrams. Reliability Engineering & System Safety, vol. 92, no. 10, pp. 1458-1467.

[Boulos *et al*, 2009] Boulos, M.N.K., Lou, R.C., Anastasiou, A., Nugent, C.D., Alexandersson, J., Zimmermann, G., Cortes, U., and Casas, R. (2009). Connectivity for Healthcare and Well-Being Management: Examples from Six European Projects. Int. J. of Environmental Research and Public Health.

[Boulos, 2009] Boulos, M.N.K. (2009). ECAALYX: Towards a Real-world Ambient Assisted Living Solution that Delivers in Non-technical Environments and Is Sustainable.

[Braham, 2013] Braham, M. (2013). An Approach to Build Software Based on Fault Tolerance Computing Using Uncertainty Factor. International Journal of Engineering Trends and Technology (IJETT), V6 (6):302-306.

[Cédric, 2015] Cédric, S. (2015). Infrastructure domotique pervasive pour l'assistance. PhD thesis 2015, UNIVERSITE BRETAGNE SUD, Lorient.

[Cesta *et al*, 2013] Cesta, A., Coraci, L., Cortellessa, G., Benedictis, R.D., Orlandini, A., Palumbo, F., and Stimec, A. (2013). Steps Toward End-to-End Personalized AAL Services. Workshop Proceedings of the 9th International Conference on Intelligent Environments.

[Chakraborty *et al*, 2002] Chakraborty, D., Perich, F., Joshi, A., Finin, T., and Yesha, Y. (2002). A reactive service composition architecture for pervasive computing environments. In PWC ’02 : Proceedings of the IFIP TC6/WG6.8 Working Conference on Personal Wireless Communications, pages 53–62, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V. ISBN 1-4020-7250-3.

[Chan *et al*, 2008] Chan, M., Esteve, D., Escriba, C., and Campo, E. (2008). A review of smart homes - Present state and future challenges. Computer Methods and Programs in Biomedicine, vol. 91, no. 1, pages 55–81.

[Cheok *et al*, 1998] Cheok, M.C., Parry, G.W., and Sherry, R.R. (1998). Use of importance measures in risk-informed regulatory applications. Reliability Engineering & System Safety, vol. 60, no. 3, pp. 213-226.

[Cheung, 1980] Cheung, R.C. (1980). A User-Oriented Software Reliability Model. IEEE Transactions on software engineering, VOL. SE-6, NO. 2.

[Collofello, 1998] Collofello, J.S. (1998). Introduction to Software Verification and Validation. SEI Curriculum Module SEI-CM-13-1.1.

[Cooper & Keating, 1996] Cooper, M. and Keating, D. (1996). Implications of the emerging home systems technologies for rehabilitation. Medical engineering & physics, 18(3). 176-118.

[Coradeschi *et al*, 2013] Coradeschi, S., Cesta, A., Cortellessa, G., Coraci, L., Karlsson, L., Furfari, F., Loutfi, A., Orlandini, A., Palumbo, F., Pecora, F., Rump, S.V., Stimec, A., Ullberg, J., and Otsland, B. (2013). GiraffPlus: Combining social interaction and long term monitoring for promoting independent living.

[Cordis, 2016] European Commission: Cordis: FP7: ICT: Home. http://cordis.europa.eu/fp7/ict

[Darondeau *et al*, 2012] Darondeau, P., Demri, S., Meyer, R., and Morvan, C. (2012). Petri Net Reachability Graphs: Decidability Status of First-Order Properties. Logical Methods in Computer Science (LMCS). Vol. 8(4:9), pp: 1-28.

[De Lamotte *et al*, 2008] De Lamotte, F., Departe, JP., Le Saout, F., Diguet, JP., and Philippe, JP. (2008). QUATRA: Final report (Technical report, in French). Lab-STICC. See demo here: http://www.youtube.com/watch ?v=T6GCFnkLTc0.

[Desel & Juhás, 2001] Desel. J. and Juhás. G. (2001). What Is a Petri Net? Informal Answers for the Informed Reader. Unifying Petri Nets. Lecture Notes in Computer Science Volume 2128, p. 1-25.

[Deursen *et al*, 2000] Deursen, A.V., Klint, P., and Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. ACM SIGPLAN Notices, Volume 35 Issue 6, Pages 26 – 36.

[Ericson, 2005] Ericson, C.A. (2005). Fault tree analysis. In Ericson (ed.), Hazard Analysis Techniques for System Safety, John Wiley & Sons, Virginia, pp. 183-221.

[Eurostat, 2016]. Eurostat Home.
http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home

[Experiment-Haal, 2016]
https://www.telecombretagne.eu/recherche/platesformes_technologiques/experiment-haal/

[FaultCAT, 2016]. http://www.iu.hio.no/FaultCat/index.htm.

[Few, 1996] Few. A.A. (1996). System behavior and system modeling. Department of space physics and astronomy. Rice University. University science books. Sausalito, California. 3rd edition.

[Goel and Okumoto, 1979] Goel, A. L. and Okumoto, K. (1979). A Markovian model for reliability and other Performance measures of software systems, in Proc. COMPCON, IEEE Computer Society Press, Los Angeles.

[Gokhale *et al*, 1998] Gokhale, S.S., Lyu, M.R., and Trivedi, K.S. (1998). Reliability Simulation of Component-Based Software Systems.

[Hessian *et al*, 1990] Hessian, R.T., Salter, B.B., and Goodwin, E.F. (1990). Fault-tree analysis for system design, development, modification, and verification. IEEE Transactions on Reliability. Vol. 39, Issue 1.

[Hostettler *et al*, 2011] Hostettler, S., Marechal, A., Linard, A., Risoldi, M., and Buchs, D. (2011). High-Level Petri Net Model Checking with AlPiNA. Fundamenta Informaticae, vol. 113, no. 3-4, p. 229-264.

[Hsu *et al*, 2010] Hsu, I., Gallagher, A., Le, M., and Tamir, Y. (2010). Using virtualization to validate Fault-tolerant distributed systems. Proceedings of the International Conference on Parallel and Distributed Computing and Systems Marina del Rey, California, pp. 210-217.

[IDA, 2016] http://www.loustic.net/ida

[IEEE, 1990] 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology.

[IEEE-STD-610, 2016] https://standards.ieee.org/findstds/standard/610-1990.html

[Jeffers, 1988] Jeffers, J.N.R. (1988). Practitioner's Handbook on the Modeling of Dynamic Change in Ecosystems. Published on behalf of the Scientific Committee on Problems of the Environment of the International Council of Scientific Unions.

[Jouault *et al*, 2006] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., and Valduriez, P. (2006). ATL : a QVT-like transformation language. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, page 720. ACM.

[Julwan, 2013] Julwan, H.P. (2013). Framework, Approach and System of Intelligent Fault Tree Analysis for Nuclear Safety Assessment. A Thesis Submitted for the Degree of Doctor of Philosophy, University of Technology, Sydney.

[Kim *et al*, 2006] Kim, Y., Kim, E-K., Kim, J., Song, E., and Ko, I-Y. (2006). Ontology based software reconfiguration in a ubiquitous computing environment. In CIT, page 260. IEEE Computer Society. ISBN 0-7695-2687-X.

[Kyprianidou, 2002] Kyprianidou, C. (2002). Analyzing Basic Genetics Using Bayesian Networks. (PhD) dissertation. Msc Actuarial science cases business school City University.

[Lacey, 2011] Lacey, P. (2011). An Application of Fault Tree Analysis to the Identification and Management of Risks in Government Funded Human Service Delivery. Proceedings of the 2nd International Conference on Public Policy and Social Sciences.

[Lamprinakos, 2010] Lamprinakos, G. (2010). An Integrated Architecture for Remote Healthcare Monitoring. 14th Panhellenic Conference on Informatics.

[Landstrom. 2010] Landstrom, E. (2010). Architecture for Fault-tolerant Control and Construction of Bayesian Network for Diagnosis. Gothenburg, Sweden.

[Lankri & Philippe, 2009] Lankri, S. and Philippe, JL. (2009). Multi- Level Reconfiguration in the DANAH Assistive System. In Proceedings of the IEEE International conference on Systems, Man and Cybernetics. San Antonio, USA.

[Lankri *et al*, 2009] Lankri, S., Berruet, P., and Philippe, JL. (2009). Service Reconfiguration in the DANAH Assistive System. In Proceedings of the 7th International Conference on Smart Homes and Health Telematics. France.

[Lankri, 2009] Lankri, S. (2009). Services et Navigation pour Personnes Dépendantes en Environnements Domotiques. PhD thesis 2009, UNIVERSITE DE BRETAGNE SUD, Lorient.

[Lee *et al*, 2008] Lee, J., Kim, J., Lee, B., and Wu, C. (2008). Utilizing semantic web 2.0 for self-reconfiguration of SOA based agent applications in intelligent service robots. In Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on, pages 784–789. Doi: 10.1109 /CIT.2008.4594774.

[Lézoray *et al*, 2010] Lézoray, J.B., Segarra, M.T., An, P.K., Thépaut, A., Gilliot, J.M., and Beugnard, A. (2010). A design process enabling adaptation and customization of services for the elderly. IWAAL-2010: International Workshop on Ambient Assisted Living, Valancia, Spain.

[Lézoray *et al*, 2011] Lézoray, J.B., Segarra, M.T., An, P.K., Thépaut, A., Gilliot, J.M., and Beugnard, A. (2011). A design process enabling adaptation in pervasive heterogeneous contexts. Personal and Ubiquitous Computing, vol. 15, no. 4, pages 353–363.

[Lioudakis, 2010] Lioudakis, G.V. (2010). Introducing Privacy-Awareness in Remote Healthcare Monitoring. Applied Sciences in Biomedical and Communication Technologies (ISABEL).

[Liping & An, 2010] Liping, H. and An, Z. (2010). Possibilistic Information Measure of Importance in Fault Tree Analysis. Seventh International Conference on Fuzzy Systems and Knowledge Discovery.

[Liping *et al*, 2013] Liping, H., Hong-Zhong, H., Yu, P., Li, Y., and Liu, Y. (2013). Importance identification for fault trees based on possibilistic information measurements. Journal of Intelligent & Fuzzy Systems 1013–1026.

[Lohr *et al*, 2015] Lohr, C., Kerdreux, J., and Tanguy, P. (2015). XAAL: A Distributed Infrastructure for Heterogeneous Ambient Devices. Journal of Intelligent Systems. Volume 24, Issue 3, Pages 321–331, ISSN (Online) 2191-026X, ISSN (Print) 0334-1860, DOI: 10.1515/jisys-2014-0144.

[Luis & Matteo, 2008] Luis, P. and Matteo, R. (2008). Metamodeling with Eclipse. SMV technical report series, No N/A. Centre Universitaire d'Informatique. Switzerland.

[M2M Journal, 2013] M2M Journal, ISSN 1868 – 9558, March 2013. Page 11.

[Marsan, 1989] Marsan, M.A. (1989). Stochastic petri nets: an elementary introduction. Advances in Petri Nets 1989, Springer, Berlin, pp. 1–29.

[Miller & Mukerji, 2001] Miller, J., Mukerji, J. (2001). Model driven architecture (MDA). Object Management Group, Draft Specification ormsc/2001-07-01.

[Miori *et al*, 2006] Miori, V., Tarrini, L., Manca, M., and Tolomei, G. (2006). An Open Standard Solution for Domotic Interoperability. Consumer Electronics, IEEE Transactions on (Volume: 52, Issue: 1).

[Myllymäki, 2001] Myllymäki, P. (2001). Advantages of Bayesian Networks in Data Mining and Knowledge Discovery.

[NASA, 2002] Fault Tree Handbook with Aerospace Applications version 1.1. Prepared for NASA Office of Safety and Mission Assurance NASA Headquarters Washington, DC 20546 August, 2002.

[Nelson, 1990] Nelson, V. P. (1990). Fault-Tolerant Computing: Fundamental Concepts.

[Norris, 1998] Norris, J. (1998). Markov chains. Cambridge University Press.

[OMG, 2014] "OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1". Object Management Group. Retrieved 9 April 2014.

[Pendharkar *et al*, 2008] Pendharkar, P.C., Rodger, J.A., and Subramanian, G.H. (2008). An empirical study of the Cobb–Douglas production function properties of software development effort. Volume 50, Issue 12 (Science Direct): Pages 1181–1188.

[Perumal *et al*, 2008] Perumal, T., Ramli, A.R., Leong, C.Y., and Mansor, S. (2008). Interoperability among Heterogeneous Systems in Smart Home Environment. IEEE International Conference on Signal Image Technology and Internet Based Systems.

[Pfeifer & Carraway, 2000] Pfeifer, P.E. and Carraway, R.L. (2000). Modeling Customer Relationships as Markov Chains. Journal of Interactive Marketing, 14(2), 43-55.

[Philip & Emad, 1995] Philip, J.B. and Emad, E.N. (1995). Measures of component importance in reliability theory. Computers & Operations Research, Vol. 22, No. 4, Pages 455–463.

[Prestileo and Fiore, 2012] Prestileo, A. and Fiore, R.D. (2012). inCASA project –Smart telemonitoring. TeleMediCare.

[Pruski, 2003] Pruski, A. (2003). Assistance technique au handicap (Traité IC2, série Systèmes automatisés). Recherche, vol. 67, page 02.

[Queirós *et al*, 2013] Queirós, A., Silva, A., Alvarelhão, J., Rocha, N.P., and Teixeira, A. (2013). Usability, accessibility and ambient-assisted living: a systematic literature review. Universal Access in the Information Society.

[Ramamoorthy *et al*, 1977] Ramamoorthy, C.V., Ho, G.S., and Han, Y.W. (1977). Fault tree analysis of computer systems. In Proceedings of the National Computer Conference. IEEE, New York, pp. 13-17.

[Rennels, 1987] Rennels, D.A. (1987). Fault Tolerant computing: issues, examples and methodologies. Lecture notes. Advanced fault-tolerant computing workshop. Bangalore-India. July 20-25.

[Richards *et al*, 2013] Richards, R., Sahin, O., and Sano, M. (2013). Bayesian (Belief) Networks. GCCRP Climate Change Early Career Research Network.

[Sallak *et al*, 2013] Sallak, M., Schon, W., and Aguirre, F. (2013). Extended component importance measures considering aleatory and epistemic uncertainties. IEEE Transactions on Reliability on 62 (1).

[Schneider & Becker, 2008] Schneider, D. and Becker, M. (2008). Runtime Models for Self-Adaptation in the Ambient Assisted Living Domain. 3rd Workshop on Models@run.time at MODELS. Toulouse, France.

[Sen, 2010] Sen, J. (2010). A Robust and Fault-Tolerant Distributed Intrusion Detection System. 1st International Conference on Parallel Distributed and Grid Computing (PDGC).

[Sherbrooke, 2016] Université de Sherbrooke: domus.usherbrooke.ca

[SIGAAL, 2016] http://sigaal.org/

[Sun *et al*, 2010] Sun, H., De Florio, V., Gui, N., and Blondia, C. (2010). The Missing Ones: Key Ingredients towards Effective Ambient Assisted Living Systems. Journal of Ambient Intelligence and Smart Environments, Volume 2, number 2. P.109-120.

[Suresh *et al*, 1996] Suresh, P.V., Babar, A.K., and Raj, V.V. (1996). Uncertainty in fault tree analysis: A fuzzy approach. Fuzzy Sets and Systems, Vol. 83, pp.135-141.

[Tanaka *et al*, 1983] Tanaka, H., Fan, L.T., Lai, F.S., and Toguchi, K. (1983). Fault tree analysis by fuzzy probability. IEEE Trans. Reliability, Vol.32, No.5, pp. 455-457.

[Truong *et al*, 2009a] Truong, T.B.T., De Lamotte, F., and Diguet, J-Ph. (2009a). Proactive remote healthcare based on multimedia and home automation services. In Proceedings of the 5th IEEE CASE, pages 385–390.

[Truong *et al*, 2009b] Truong, T.B.T., De Lamotte, F., Diguet, J-Ph., Said-Hocine, F. (2009b). Assisted living service identification based on activity patterns. In Pervasive Computing (JCPC), Joint Conferences on, pages 417–422.

[Truong, 2010] Truong, T.B.T. (2010). Home Automation Monitoring for Assisted Living Services and Healthcare. PhD thesis, Laboratoire en Sciences et Techniques de l'Information, de la Communication et de la Connaissance, Université de Bretagne-Sud, 2010.

[Virone & Sixsmith, 2008] Virone, G. and Sixsmith, A. (2008). Toward Information Systems for Ambient Assisted Living. In Proceedings of the 6th Int. Conference of the International Society for Gerontechnology, Pisa, Tuscany, Italy, June 4-7.

[Wang *et al*, 1999] Wang, W.L., Wu, Y., and Chen, M.H. (1999). An Architecture-Based Software Reliability Model. Dependable Computing.

[Weibull, 2006] Component Reliability Importance in System Reliability Analysis. http://www.weibull.com/hotwire/issue62/relbasics62.htm. Issue 62, April 2006.

[XAAL, 2016] http://recherche.telecom-bretagne.eu/xaal/

[Zhao & Petriu, 2015] Zhao, Z. and Petriu, D.C. (2015). UML Model to Fault Tree Model Transformation for Dependability Analysis. Proceedings of the International Conference on Computer and Information Science and Technology Ottawa, Ontario, Canada.

Personal publications

1. A comparison between Ambient Assisted Living Systems. DARWISH Molham, SENN Eric, LOHR Christophe, KERMARREC Yvon. ICOST 2014: 12th International Conference on Smart Homes and Health Telematics, Springer, 25-27 June 2014, Denver, United States, 2015, vol. 8456 - LNCS (Lecture Notes in Computer Science), pp. 231-237.
2. The integration of expert-defined importance factors to enrich Bayesian Fault. A journal article submitted and under revision in: Reliability Engineering and System Safety journal - Elsevier

Table of figures

Appendix A

In this section, we present the system elements models of the illustrative example presented in chapter 5, in order to realize the verification experiments.

To recall, this concrete example representing a realistic home automation system composed of several services using 15 equipments is proposed. This example is extracted from the living lab Experiment'Haal. This example can be extended to include several services and a greater number of components.

In fact all the used equipments (belonging to the platform) are represented in terms of components modeled as the first phase in the system workflow.

Figure A.1 presents a tree structure representation of the proposed system.

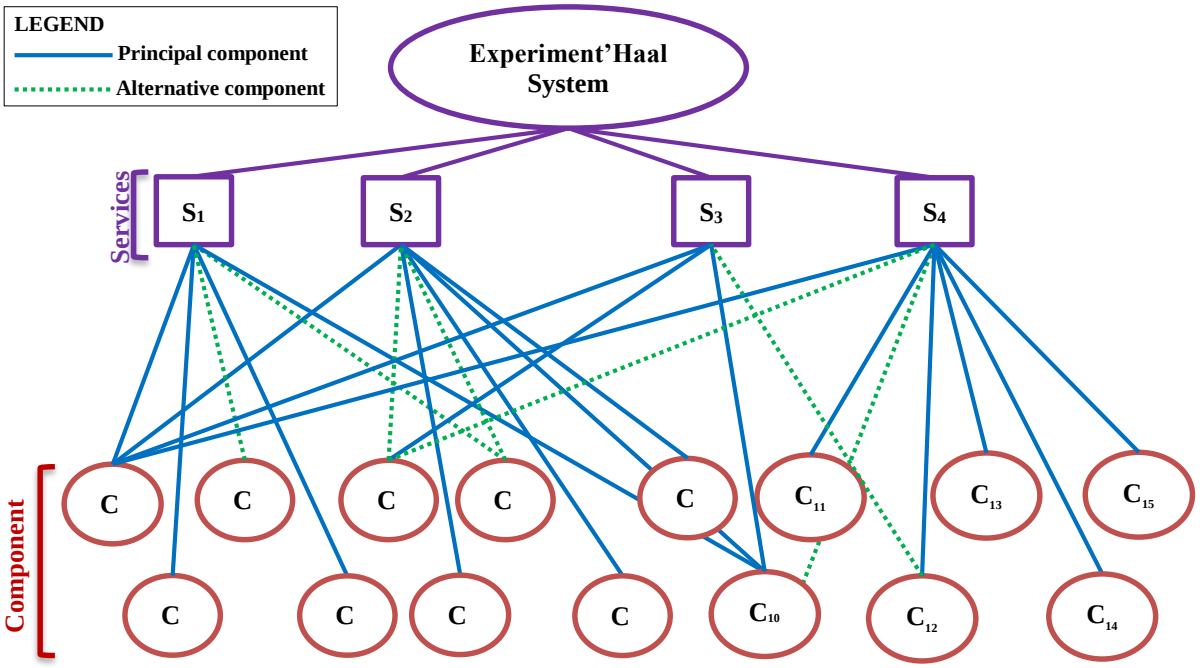


Figure A.1. The Home automation system example

In this example, the basic components used in the system, independent from the services, has been given as follows:

C_1: *Presence detector*.
C_2: *Living room light*.
C_3: *Lampadaire.*
C_4: *TV*.
C_5: *Dining room light*.
C_6: *Bathroom light*.
C_7: *Window blind*.

C_8: *Water leak detector in the bathroom.*
C_9: *Water flow sensor in the bathroom.*
C_{10}: *Main power consumption sensor.*
C_{11}: *Power consumption sensor in the kitchen.*
C_{12}: *Kitchen Light.*
C_{13}: *Water leak detector in the kitchen.*
C_{14}: *Water flow sensor in the kitchen.*
C_{15}: *Smoke detector.*

The XML models of components $C_1 \ldots C_{15}$ are illustrated in the figures A.2, A.3, A.4, respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
<component>
    <id>1</id>
    <name>Presence_detector</name>
    <failure_probability>0.001</failure_probability>
    <operation>
        <operation1>SENSE</operation1>
    </operation>
    <location>
        <X_coordinate>1.5</X_coordinate>
        <Y_coordinate>1.0</Y_coordinate>
        <Z_coordinate>3.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
<component>
    <id>2</id>
    <name>Living_room_light</name>
    <failure_probability>0.01</failure_probability>
    <operation>
        <operation1>ON</operation1>
        <operation2>OFF</operation2>
    </operation>
    <location>
        <X_coordinate>1.5</X_coordinate>
        <Y_coordinate>0.9</Y_coordinate>
        <Z_coordinate>2.75</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
<component>
    <id>3</id>
    <name>Lampadaire</name>
    <failure_probability>0.05</failure_probability>
    <operation>
        <operation1>ON</operation1>
        <operation2>OFF</operation2>
    </operation>
    <location>
        <X_coordinate>1.0</X_coordinate>
        <Y_coordinate>0.5</Y_coordinate>
        <Z_coordinate>1.5</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
<component>
    <id>4</id>
    <name>TV</name>
    <failure_probability>0.01</failure_probability>
    <operation>
        <operation1>CH1</operation1>
        <operation2>CH2</operation2>
        <operation3>CH3</operation3>
        <operation4>CH4</operation4>
    </operation>
    <location>
        <X_coordinate>1.5</X_coordinate>
        <Y_coordinate>0.6</Y_coordinate>
        <Z_coordinate>1.5</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>5</id>
    <name>Dining_room_light</name>
    <failure_probability>0.01</failure_probability>
  - <operation>
      <operation1>ON</operation1>
      <operation2>OFF</operation2>
    </operation>
  - <location>
      <X_coordinate>1.5</X_coordinate>
      <Y_coordinate>3.0</Y_coordinate>
      <Z_coordinate>2.75</Z_coordinate>
    </location>
  </component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>6</id>
    <name>Bathroom_light</name>
    <failure_probability>0.01</failure_probability>
  - <operation>
      <operation1>ON</operation1>
      <operation2>OFF</operation2>
    </operation>
  - <location>
      <X_coordinate>1.75</X_coordinate>
      <Y_coordinate>4.5</Y_coordinate>
      <Z_coordinate>2.75</Z_coordinate>
    </location>
</component>
```

Figure A.2. XML models for the components {C_1-C_2-C_3-C_4-C_5-C_6}

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>7</id>
    <name>Window_blind</name>
    <failure_probability>0.005</failure_probability>
  - <operation>
      <operation1>OPEN</operation1>
      <operation2>CLOSE</operation2>
      <operation3>VARIANT</operation3>
    </operation>
  - <location>
      <X_coordinate>1.8</X_coordinate>
      <Y_coordinate>5.0</Y_coordinate>
      <Z_coordinate>2.5</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>8</id>
    <name>Water_leak_detector_in_bathroom</name>
    <failure_probability>0.001</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>2.2</X_coordinate>
      <Y_coordinate>4.4</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>9</id>
    <name>Water_flow_sensor_in_the_bathroom</name>
    <failure_probability>0.001</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>2.2</X_coordinate>
      <Y_coordinate>4.3</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>10</id>
    <name>Main_power_consumption_sensor</name>
    <failure_probability>0.004</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>1.0</X_coordinate>
      <Y_coordinate>0.3</Y_coordinate>
      <Z_coordinate>2.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>11</id>
    <name>Power_consumption_sensor_in_the_kitchen</name>
    <failure_probability>0.004</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>0.2</X_coordinate>
      <Y_coordinate>4.2</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>12</id>
    <name>Kitchen_light</name>
    <failure_probability>0.01</failure_probability>
  - <operation>
      <operation1>ON</operation1>
      <operation2>OFF</operation2>
    </operation>
  - <location>
      <X_coordinate>1.0</X_coordinate>
      <Y_coordinate>4.5</Y_coordinate>
      <Z_coordinate>2.75</Z_coordinate>
    </location>
</component>
```

Figure A.3. XML models for the components {C_7-C_8-C_9-C_{10}-C_{11}-C_{12}}

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>13</id>
    <name>Water_leak_detector_in_kitchen</name>
    <failure_probability>0.001</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>0.2</X_coordinate>
      <Y_coordinate>4.7</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>14</id>
    <name>Water_flow_sensor_in_the_kitchen</name>
    <failure_probability>0.001</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>0.2</X_coordinate>
      <Y_coordinate>4.8</Y_coordinate>
      <Z_coordinate>1.0</Z_coordinate>
    </location>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component.dtd">
- <component>
    <id>15</id>
    <name>Smoke_detector</name>
    <failure_probability>0.001</failure_probability>
  - <operation>
      <operation1>SENSE</operation1>
    </operation>
  - <location>
      <X_coordinate>1.3</X_coordinate>
      <Y_coordinate>3.8</Y_coordinate>
      <Z_coordinate>3.0</Z_coordinate>
    </location>
</component>
```

Figure A.4. XML models for the components {C_{13}-C_{14}-C_{15}}

Moreover, four services are considered in this example:

S1: *Watching TV in the afternoon.*
S2: *Using the washbasin.*
S3: *Have a dinner.*
S4: *Cooking the dinner.*

The XML models for the services S_1, S_2, S_3 and S_4 are illustrated in the figure A.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
<service>
    <id>1</id>
    <name>Watching_TV_in_the_afternoon</name>
    <context>sitting at room at afternoon</context>
    <time>afternoon</time>
    <environment>warm</environment>
    <location>[1.5, 1.0, 3.0]</location>
    <user_location>[1.5, 1.0]</user_location>
    <component>&presence</component>
    <component>&Light1</component>
    <component>&TV</component>
    <component>&power1</component>
</service>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
<service>
    <id>2</id>
    <name>Using_the_washbasin</name>
    <context>Using the washbasin</context>
    <time>afternoon</time>
    <environment>warm</environment>
    <location>[1.7, 4.3, 3.0]</location>
    <user_location>[1.7, 4.3]</user_location>
    <component>&presence</component>
    <component>&Light3</component>
    <component>&Water_detector1</component>
    <component>&Water_sensor1</component>
    <component>&power1</component>
</service>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
<service>
    <id>3</id>
    <name>Have_a_dinner</name>
    <context>Have a dinner</context>
    <time>night</time>
    <environment>warm</environment>
    <location>[1.5, 3.0, 3.0]</location>
    <user_location>[1.8, 3.0]</user_location>
    <component>&presence</component>
    <component>&Light2</component>
    <component>&power1</component>
</service>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE service SYSTEM "service.dtd">
<service>
    <id>4</id>
    <name>Cooking_the_dinner</name>
    <context>Cooking the dinner</context>
    <time>night</time>
    <environment>warm</environment>
    <location>[0.5, 4.5, 3.0]</location>
    <user_location>[0.5, 4.5]</user_location>
    <component>&presence</component>
    <component>&power2</component>
    <component>&Light4</component>
    <component>&Water_detector2</component>
    <component>&Water_sensor2</component>
    <component>&smoke</component>
</service>
```

Figure A.5. XML models for the services {S_1-S_2-S_3-S_4}

For each service, the corresponding components will have a certain XML model, to describe the functionality of the component in the service, with the set of the alternative components for each principal component. For instance, the component "*Living room light*" has the functionality model for the service S_1 "*Watching TV in the afternoon*", illustrated in the figure A.6.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component_service.dtd">
<component>
    <id>2</id>
    <name>Living_room_light</name>
    <priority>2</priority>
    <importance>1.7</importance>
    <operation>SwitchON</operation>
    <Component_context>lightening</Component_context>
    <service_Id>1</service_Id>
    <Alternative_Component>&Lampadaire</Alternative_Component>
    <Alternative_Component>&Window_blind</Alternative_Component>
</component>
```

Figure A.6. XML model for the component C_2 functioning in the service S_1

In the same manner, the component "*Bathroom light*" has the functionality model for the service S_2 "*Using the washbasin*", illustrated in the figure A.7.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE component SYSTEM "component_service.dtd">
- <component>
    <id>2</id>
    <name>Bathroom light</name>
    <priority>2</priority>
    <importance>1</importance>
    <operation>SwitchON</operation>
    <Component_context>lightening</Component_context>
    <service_Id>2</service_Id>
    <Alternative_Component>&Light2</Alternative_Component>
    <Alternative_Component>&Window_blind</Alternative_Component>
</component>
```

Figure A.7. XML model for the component C_6 functioning in the service S_2

The generated fault tree analysis XML model for the service S_4 "*Cooking the dinner*", which is generated by the transformation rule from the design model to the fault tree analysis model, is illustrated in figure A.8.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Fault-Tree>
  - <Intermediate-Event>
      <Title>Cooking the dinner</Title>
      <Info/>
    - <And-Gate>
      - <Basic-Event>
          <Title>C1</Title>
          <Value>0.001</Value>
          <Info/>
          <Type>0</Type>
        </Basic-Event>
      - <Intermediate-Event>
          <Title/>
          <Info/>
        - <Or-Gate>
          - <Basic-Event>
              <Title>C11</Title>
              <Value>0.004</Value>
              <Info/>
              <Type>0</Type>
            </Basic-Event>
          - <Basic-Event>
              <Title>C10</Title>
              <Value>0.004</Value>
              <Info/>
              <Type>0</Type>
            </Basic-Event>
          </Or-Gate>
      </Intermediate-Event>
  - <Intermediate-Event>
      <Title/>
      <Info/>
    - <Or-Gate>
      - <Basic-Event>
          <Title>C12</Title>
          <Value>0.01</Value>
          <Info/>
          <Type>0</Type>
        </Basic-Event>
      - <Basic-Event>
          <Title>C5</Title>
          <Value>0.01</Value>
          <Info/>
          <Type>0</Type>
        </Basic-Event>
      </Or-Gate>
    </Intermediate-Event>
  - <Basic-Event>
      <Title>C13</Title>
      <Value>0.001</Value>
      <Info/>
      <Type>0</Type>
    </Basic-Event>
  - <Basic-Event>
      <Title>C14</Title>
      <Value>0.001</Value>
      <Info/>
      <Type>0</Type>
    </Basic-Event>
  - <Basic-Event>
      <Title>C15</Title>
      <Value>0.001</Value>
      <Info/>
      <Type>0</Type>
    </Basic-Event>
  </And-Gate>
</Intermediate-Event>
</Fault-Tree>
```

Figure A.8. The Fault tree analysis model of service S_4

Figure A.9 presents the FTA graphical representation of the service S_4.

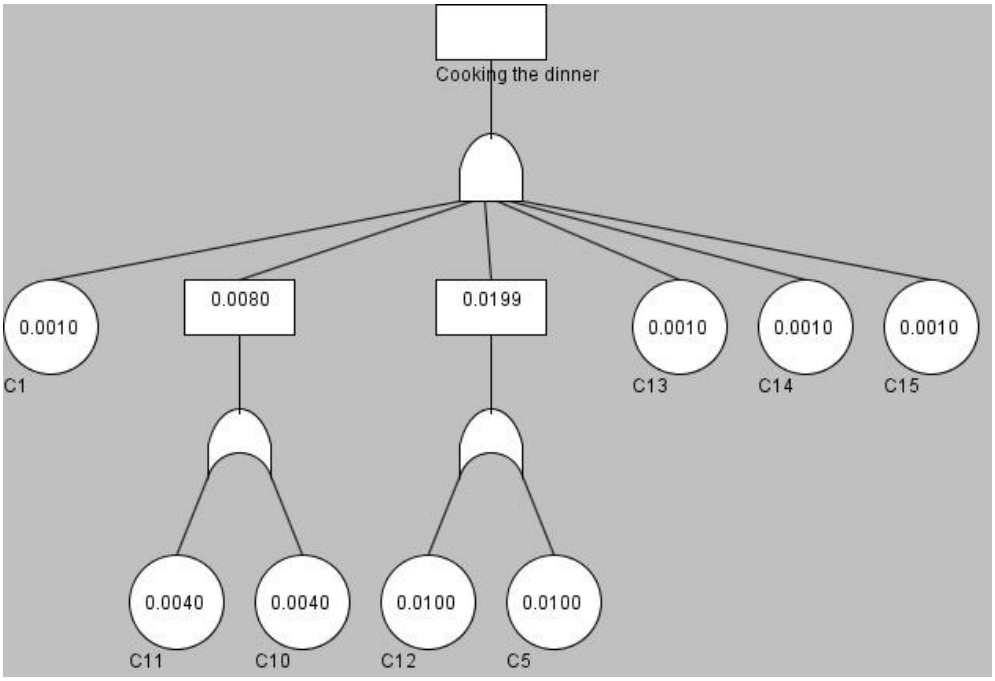


Figure A.9. The FTA presentation of the service S_4 using FaultCAT

The behavior analysis model generated by the transformation rule from the design model to the behavior model is illustrated in figure A.10 for the service S_3 "*Have a dinner*".

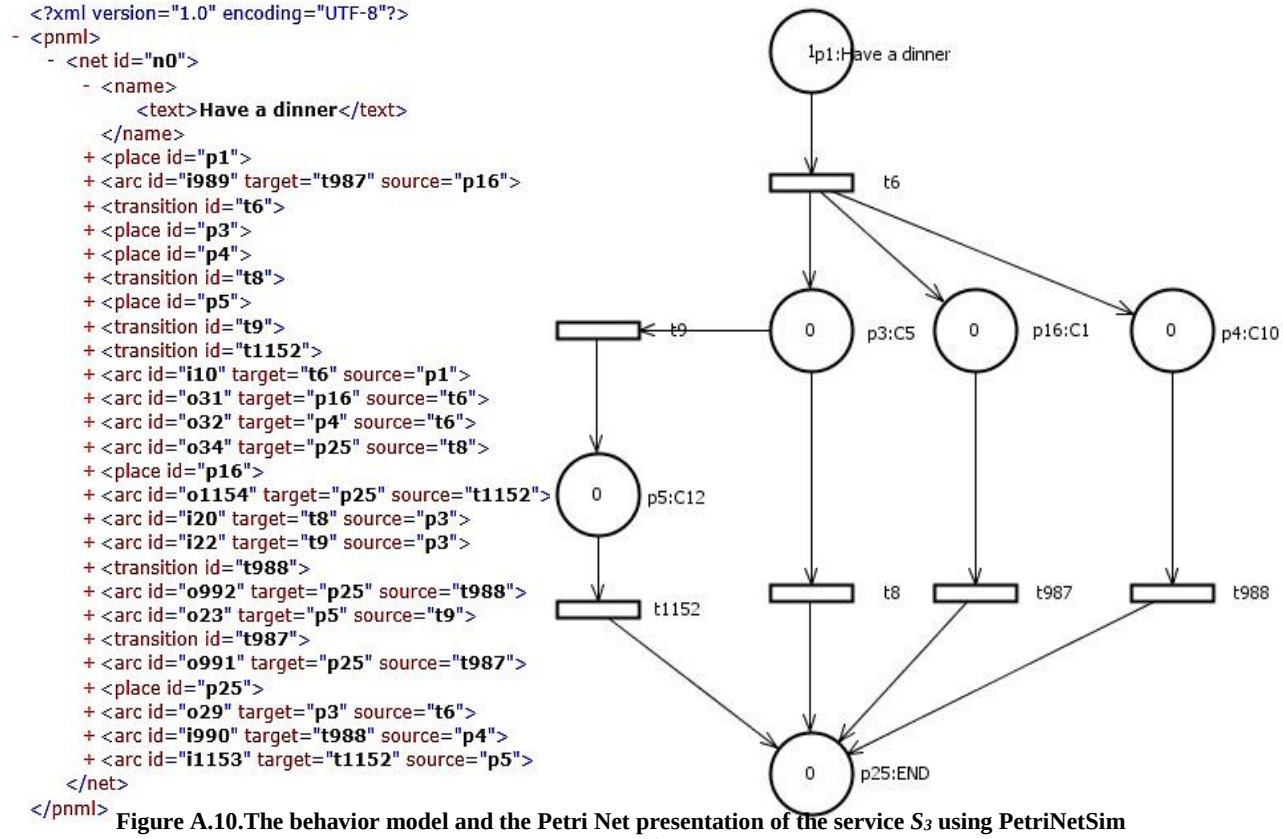


Figure A.10.The behavior model and the Petri Net presentation of the service S_3 using PetriNetSim