Using timed automata formalism for modeling and analyzing home care plans

Kahina Gani

N° d'ordre: D. U : 2628
E D S P I C: 724

UNIVERSITE Blaise Pascal-Clermont II

**ECOLE DOCTORALE
SCIENENCES POUR L'INGENIEUR DE
CLERMONT-FERRAND**

T H È S E

Présentée par

Mme. Kahina GANI

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Spécialité : Informatique

Titre de la thèse :

Using Timed Automata Formalism For Modeling and Analyzing Home Care Plans

Soutenue publiquement le 02 decembre 2015 devant le jury:

M. Kamel Berkaoui	***Rapporteur et Président du jury***
M. Claude Godart	***Rapporteur***
M. Elyes Lamine	***Examinateur***
Mme. Marinette Bouet	***Co-encadrante***
M. Farouk Toumani	***Directeur de thèse***
M. Rémy Bastide	***Co-directeur de thèse***
M. Michel Schneider	***Invité***

In loving memory of my parents
Mouloud & Hania

Acknowledgment

Foremost, my sincere thanks go to my advisors, Pr. Farouk Toumani, Pr. Rémi Bastide, Dr. Marinette Bouet and Pr. Michel Schneider, without whom this thesis would not have been achieved.

I would like to thank Pr. Claude Gaudart and Pr. Kamel Berkaoui for taking the time to review my thesis report. I also thanks Dr. Elyes Lamine for having accepted to participate in my thesis committee as an examiner.

Thank to my friends, Doutoum, Yahia, Lakhder, Karima, Kawther, Nader, Maxime, Raksmey, Libo, Diyé and Marie. I also would like to thank my friends who are having become meantime as my little family here in Clermont-Ferrand. Meriem and salsabil my little heart sisters and Djelloul Mameri, thank you for everything you did for me, without forgetting his wife.

Finally, I want to thank my family, my wonderful and adorable sister Lynda to who I owe everything and my dear brothers Khaled, Nadir and Samir and the man of my life Mustapha for his love and support.

Abstract

In this thesis we are interested in the problems underlying the design and the management of home care plans. A home care plan defines the set of medical and/or social activities that are carried out day after day at a patient's home. Such a care plan is usually constructed through a complex process involving a comprehensive assessment of patient's needs as well as his/her social and physical environment. Specification of home care plans is challenging for several reasons: home care plans are inherently non-structured processes which involve repetitive, but irregular, activities, whose specification requires complex temporal expressions. These features make home care plans difficult to model using traditional process modeling technologies. First, we present a DSL (Domain Specific Language) based approach tailored to express home care plans using high level and user-oriented abstractions. DSL enables us through this thesis to propose a temporalities language to specify temporalities of home care plan activities. Then, we describe how home care plans, formalized as timed automata, can be generated from these abstractions. We propose a three-step approach which consists in (i) mapping between elementary temporal specifications and timed automata called Pattern automata, (ii) combining patterns automata to build the activity automata using our composition algorithm and then (iii) constructing the global care plan automaton. The resulting care plan automaton encompasses all the possible allowed schedules of activities for a given patient. Finally, we show how verification and monitoring of the resulting care plan can be handled using existing techniques and tools, especially using UPPAAL Model Checker.

Keywords: Timed Automata, Domain Specific Language, UPPAAL Model Checker, Home Care Plan.

Résumé

Dans cette thèse nous nous sommes intéressés aux problèmes concernant la conception et la gestion des plans de soins à domicile. Un plan de soins à domicile définit l'ensemble des activités médicales et/ou sociales qui sont menées jour après jour au domicile d'un patient. Ce plan de soins est généralement construit à travers un processus complexe impliquant une évaluation complète des besoins du patient ainsi que son environnement social et physique. La spécification de plans de soins à domicile est difficile pour plusieurs raisons: les plans de soins à domicile sont par nature des processus non structurés qui impliquent des activités répétitives mais irrégulières, dont la spécification requiert des expressions temporelles complexes. Ces caractéristiques font que les plans de soins à domicile sont difficiles à modéliser en utilisant les technologies traditionnelles de modélisation de processus. Tout d'abord, nous présentons l'approche basée sur les DSL (Langage spécifique au domaine) qui permet d'exprimer les plans de soins à domicile en utilisant des abstractions de haut niveau et orientées utilisateur. Le DSL nous permet à travers cette thèse de proposer un langage de temporalités permettant de spécifier les temporalités des activités du plan de soins à domicile. Ensuite, nous décrivons comment les plans de soins à domicile, formalisés grâce aux automates temporisés, peuvent être générés à partir de ces abstractions. Nous proposons une approche en trois étapes qui consiste à: (i) le mapping entre les spécifications temporelles élémentaires et les automates temporisés appelés "pattern automata", (ii) la combinaison des "patterns automata" afin de construire les automates d'activités en utilisant l'algorithme de composition que nous avons défini, et aussi (iii) la construction de l'automate de plan de soins à domicile global. L'automate de plan de soins à domicile résultant englobe tous les schedules autorisés des activités pour un patient donné. Enfin, nous montrons comment la vérification et le suivi de l'automate du plan de soins à domicile résultant peuvent être faits en utilisant des techniques et des outils existants, en particulier en utilisant l'outil de verification UPPAAL.

Mots clés : Automates temporisés, Langage spécifique au domaine, Outils de verification UPPAAL, Plan de soins à domicile.

Contents

List of Figures

List of Tables

General introduction

A general trend that can be observed these recent years in health domain is the development of the home care. Home care refers to health care or supportive care delivered by health care professionals at patients' homes. The target objective for social and economic reasons is to enable people who require care to remain at home instead of having long-term stays in hospitals or health establishments. Several types of care may be provided at patient's home including health services, specialized care such as parenteral nutrition or activities related to daily living such as bathing, dressing, toilet, etc. All the medical and social activities delivered for a given patient according to certain frequencies are scheduled in a so-called home care plan. In [51], the home care plan is defined as "a provisional list of treatments to achieve which may include activities and treatments, prescribed or not, associated with the frequency and schedules". Hence, the notion of a home care plan is a key concept in home care. As a part of the project Plas'O'Soins[1](Plateforme d'Aide au Suivi et à la cOordination des activités de Soins à domicile), we are interested by the problems underlying the design and management of the home care plans in order to bring some comfort to the coordinator, avoid the collision when planning activities or to ensure the feasibility of the home care plan, etc.

The design of a home care plan is challenging for several reasons. First, process modeling in the medical field is in general not an easy task [44]. Indeed, medical processes require usually complex coordination and interdisciplinary cooperation due to involvement of actors from various health care institutions. Moreover, home care plans display the following features that make them difficult to handle with traditional Business Process Management (BPM) technologies:

- Home care plan can be viewed as a collection of repetitive activities (e.g., medical activities) which are repeated during a given period. The activities are however enacted according to an irregular schedule. The irregularities of an activity have two main causes. First, the activity very often has to follow the evolution of the needs which is usually irregular. This type of irregularity is characterized by strengthening or weakening in the rhythm of realization. Then, an activity which requires human resources has to respect life cycles appropriate to this type of resources and in particular rest time of the weekend. This type of irregularity induces interruptions

[1] http://plasosoins.univ-jfc.fr/

in the rhythm of the realization. Specification of irregular activities requires the use of suitable temporal constraints.

- Home care plans are essentially unstructured processes in the sense that each patient has his/her own specific care plan. Indeed, the home care plan for each patient is developed on an individual basis because each patient is unique whether at his/her pathology or needs. Therefore, it is simply not possible to design a unique process capturing in advance the care plans of all the patients. In other words, the traditional approach *"model once, execute many times"* is not sustainable in our context.

- Home care plans are associated with complex temporal constraints. Indeed, the design of a home care plan requires the specification of the frequencies of the delivered home care activities. Such specifications are expressed by healthcare professionals in natural language, using usually a compact repetitive form: *everyday in the morning, once every 2 days in the evening during 15 days,* etc. But they may also have irregularities or exceptions: *Mondays and Fridays, except public holidays*.

Given the crucial role played by temporal constraints in home care plans, it appears clearly that such specification could take benefit from existing theory and tools in the area of timed systems. There exist in the literature many formal models enabling explicit representation of timed systems [1][5][6][8][9][23]. Among the well-known formalisms, we have time extensions for Petri Nets(PNs) [34] and Timed Automata(TA) [6][39].

Petri Nets were introduced in the 1960's by Carl Adam Petri as a graphical and mathematical modeling tool. It consists of four elements: places (represented by circles), transitions (represented by rectangles), edges also called arcs (represented by directed arrows) and tokens (represented by small solid (filled) circles). The initial model of Petri Nets was atemporal. There exist several extensions that take into account time constraints among which: *Time Petri Nets*(TPNs) [79] where a time interval is assigned to each transition, *Timed Petri Nets*(TdPNs) [100] whose transitions have exact duration. Other representations of time may involve places, called *timed places Petri nets* [105] or arcs, called *timed arc Petri nets* [55]. Among all these time variants, TPNs are most widely used for the expression of a large majority of temporal constraints [24][36][117].

In addition to Petri Nets, another well-known model has been extended to represent the time constraint: Timed Automata were introduced in [3][6] as an extension of finite state automata that enables explicit modeling of time. Informally, Timed Automata

are finite states automata enriched with *clock variables*. Moreover, transitions of timed automata are annotated with *guards*, expressed as time constraints, and *clock resets*.

Several contributions aiming to compare TPN and TA have been made [25][32][33][38]. Formalisms of TPN and TA show lots of common characteristics such as: they allow both to model real-time systems and have specific tools for properties verification [107]. Also they show the extreme difficulties to handle algorithmic analysis of timed models, for example, Timed Automata suffer from undecidability problems of language inclusion and complementation [6], and Time Petri Nets from undecidability problem of reachability [63]. Even though both are likely to meet our expectations, we use in our approach *Timed Automata*. The advantage of TA formalism is that the problem of reachability is decidable in polynomial space [6]. Also Timed Automata benefit from a very active theoretical research, which among other things allowed to develop very efficient tools such as Uppaal [18] or Kronos [35]. In our work, Timed Automata will be used as a basis to develop a formal framework to analyze home care plans.

As mentioned, we adopt as referring formal model for the specification of home care plans, the model of Timed Automata. However, due to the complex features of home care plans, it is not feasible to ask home care professionals to construct such formal specifications nor to use traditional process modeling technologies [123], usually based on GUI (Graphical User Interface), to design a home care plan process and then to automatically generate the corresponding timed automaton. To cope with such difficulties, we describe below the main features of our approach:

- The cornerstone of our approach lies in the definition of a Domain Specific Language (DSL) and a user centred specification language tailored to express home care plans using high level abstractions. We conducted a thorough analysis of current practices of home care professionals, in particular home care coordinators, in order to identify the main elementary temporal expressions used in this context. The DSL framework incorporates a GUI which is intended to be used in particular by the coordinator, to specify patient's home care plans.

- In order to ensure the feasibility of the home care plan and the non collision of defined activity, we propose an automatic transformation of user specifications into timed automata. This approach enables home care professionals to continue using a constrained form of a natural language to express complex temporal constraints (through a GUI), while being able to generate automatically timed automata-based

specifications of home care plans. In fact, the home care plan consists of a set of activities which are at their turn defined by a set of elementary temporal specifications that specify their scheduling. Thereby, building the home care plan timed automata involves the following steps:

1. *Pattern automaton construction*: The identification of the elementary patterns that are useful to specify home care plans. Each elementary temporal expression is formalized in the form of a timed automaton called *pattern automaton*. This pattern captures the temporal constraints of an elementary temporal specification and also the duration of the associated activity.
2. *Activity automaton construction*: The use of these elementary patterns to specify atomic activities. In fact, pattern automata are combined together using specific composition operator tailored to take into account the specific semantics of health care activities. The composition generates the activity automaton (i.e., the timed automaton specifying the schedule of a given activity).
3. *Home care plan automaton construction*: The automatic construction of a global home care plan by composition of the basic activities. The global *home care plan automaton* is generated using the composition operator. A timed word of such an automaton corresponds to a possible *legal schedule* of the activities and hence a *care plan automaton* describes all the possible *legal schedules* of a home care plan for a specific patient.

- The obtained formal specifications are exploited to support automatic verification and monitoring of home care plans. For example, we show that *realizability* of home care plan, which asks wether there is a schedule of activities that satisfies the plan, is reduced to the emptiness of the corresponding timed automaton while some monitoring issues are reduced to the membership or language recognition. Interestingly, the important task of the coordinator consists in grouping together the activities of home care plan in a set of *Interventions* while trying to minimize the total number of interventions at patient's home and also to reduce the overall cost of home care. Interventions generation can be computed automatically using a specific composition operator while ensuring some required properties (e.g., interleaving interventions are not allowed, *idle time* between the activities of a same intervention is controlled, etc.).

The thesis is organized as follows. **Chapter 1** describes the general context of this work and its main challenges. **Chapter 2** introduces main concepts of Domain Specific Languages (DSL) and Timed Automata and application of these approaches in the medical field. **Chapter 3** presents the contribution of this thesis which consists in modeling and analyzing home care plans. First, we describe the DSL based approach in which we identify the main building blocks in the home care plan and we propose a language that enables to express regular or irregular repetitions of an activity within some period in a condensed form similar to that used by doctors. Then, the general modeling process is presented together with the construction of the proposed automata, i.e., pattern automata, activity automata and home care plan automata. We also present some verification and monitoring issues using UPPAAL Model Checker. The validation step of our work is presented in the prototyping part (**Chapter 4**) in which we show the architecture of the prototype and we describe the implementation details. Finally, we conclude this manuscript by recalling the contribution of this thesis and discuss some future research directions.

Chapter 1

Context and problematic

Contents

The home care is a starting point of this thesis. The trend is to consider the home care as a unified way, even if it encompasses intervention of different types of structures. To ensure a good organization, quality and efficiency of care, coordinators organize and translate the various treatments to achieve in a corresponding *home care plan*. The home care plan is the main component of the home care, since it allows to coordinate different activities being performed and the various actors (physician, nurse, nurse auxiliary, liberal helper, etc.) involved in the home care as well as it's the input element to the intervention planning.

In this chapter, we present the context in the field of home care, highlighting its major challenges which prompted in this context the development of projects based on Information and Communication Technologies (ICT) among which the Plas'O'Soins project [1]. This latter will be described in the second part. Finally, we present the contribution of this thesis associated with this project.

1.1 Home care area

In recent years, home care landscape has undergone a significant evolution. This is mainly due to the major concerns of actual societies which aim to manage ageing population and to cope with the congested hospitals and centers [64]. Indeed, elderly people with loss of autonomy are especially vulnerable and often exhibit chronic diseases requiring continuous care for a long term. Keeping such a population in its social environment is a key point to avoid any risk of depression or exclusion, especially, when other factors are involved such as: children living further away, or family dispersal [47][58]. Thus, alternative structures to traditional hospitalizations have emerged. For example: (a) Hospitalization At Home, a sanitary alternative which is generally provided for patients with serious diseases; (b) Nursing Home Service, a medico-social logic which provides nursing care and general hygiene to elderly people; and (c) Maintaining At Home, a set of home services, which provides help to the addict people to allow them staying at home as long as possible, help (such as domestic help, etc.).

The development of home care is motivated by economical and political reasons [51][91] that try to control the health care costs; the demographic issue related to the ageing population; the social aspects related to the patient's wishes, as well as by technology advances where the progress on ICT contributes to the improvement of the home

[1] http://plasosoins.univ-jfc.fr/

care area.

1.1.1 Home care in France

1.1.1.1 Overview

According to INSEE [2](Institut National de la Statistique et des Etudes Economiques) in Figure 1.1, France counts 12.19 million of persons over 65 years old at January 1st 2015 that is 1.4 million more than in 2013. The ageing French population led to an increase number of fragile and dependent people. The home care concept appeared in France since 1950s through the development of some structures. Recently the National Federation of the Home Health Care Structures (Fédération Nationale des Etablissements d'Hospitalisation A Domicile- FNEHAD) underlined the increasing number of home care structures [51]. In fact, between 2005 and 2008, there was an increase of more than 120% in the number of home care and an increase of nearly 148% in the number of admitted patients.

	% Men	%Women	Both
Total population (million)	100,00	51,60	65,53
Under 20 years	24,60	48,90	16,09
From 20 to 64 years	57,90	50,80	37,93
65 years and over	17,60	57,90	**11,51**

Age group at January 1st 2013

	% Men	%Women	Both
Total population (million)	100,00	51,60	66,32
Under 20 years	24,70	48,90	16,37
From 20 to 64 years	56,90	50,80	37,76
65 years and over	18,40	57,50	**12,19**

Age group at January 1st 2015

Figure 1.1: Age group in France at January 1st 2013 and January 1st 2015. source: INSEE

Moreover, home care structures differ in their technical and medical ability (punctual care, rehabilitation care at home, palliative care, etc.) and their status: public establishment, private PHPS (Participating in the Hospital Public Service), associative private, profit-making private [51] as well as in their sizes which can be variable (small structures in general). This difference is mainly due to the clinical and social patient's needs. Indeed, home care structures try to satisfy the needs of the admitted patients on an individual basis. This implies an organizational complexity involving actors from different structures [20]. We describe in what follows some considered aspects in the home care.

[2]http://www.insee.fr/fr/default.asp

1.1.1.2 Home care actors

The home care area implies an interdisciplinary cooperation due to the involvement of actors from various health care institutions. We can distinguish several types of actors [14][20] [61][98]:

- **Treating physician**: he is freely chosen by the patient. Her agreement is a necessary condition for the patient's home care acceptance.
- **Home care coordinator**: he is the physician who is in charge of the control activities and care coordination between different health professionals. Her main tasks are [17]:
  - Elaborating the patient's home care plans and updating them according to the evolution of the patient's health state on the advice of the treating physician.
  - Monitoring the execution of the home care activities: from the information provided by different involved actors, the coordinator can be informed about the duration in the performance of activities.
- **External or internal actors**: actors can be full time employees (internal actors of the home care structure) or part-time employees (external actors). These actors are regularly involved at the patient's home such as:
  - **Nurse**: she evaluates the conditions of the patient receiving home care services and provides skilled nursing care and verifies if other services are required.
  - **Nurse auxiliary**: she is involved in the general hygiene care (toilet, dressing, bathing, etc) in the oversight of treatment taking, transfer, displacement inside home, etc.
  - **Nutritionist**: he supports the patient for the nutrition phases.
  - **Therapist**: such as pathologist who takes into account therapy needs.
- **Social workers**: they assist the patient and his/her family in their formalities to obtain material and financial support.
- **Patient's family, friends, etc.**: they contribute in the patient's home care for free. Family members or neighbors participate regularly or occasionally to provide various services to overcome the patient's disability.

1.1.1.3 Home care services

The medical and social care delivered for a given patient during the home care can be listed as follows [98]:

- **Punctual care**: it represents technical and complex care adapted to a not stabilized pathologies for determined period.
- **Continuous care**: it represents technical care of varying complexity. This type of care can vary from nursing care to maintain functioning levels for progressive pathologies.
- **Rehabilitation care**: this care is intended for patients after the acute phase of a neurological disease, orthopedic, cardiology or poly pathology for a fixed duration.
- **Home help services**: these services are provided by social service structures such as household duties (shopping, cooking, cleaning, etc) and general hygiene services (bathing, dressing, toilet, etc).

Figure 1.2 illustrates the actors types (roles) involved in the Plas'O'Soins project with their different goals.

1.1.2 Home care processes

Home cares involve complex and various processes and information exchanges. Figure 1.3 shows a generic vision of the home care processes proposed by FNEHAD [51]. This cartography follows the ISO 9000 standard which classifies the home care processes into three types: (1) Management processes, these processes contribute to set the target objectives of the home care, (2) Support processes (human resources, purchasing and logistics, IS management, etc.); these processes are essential for the proper functioning of the home care, they allow the allocation of necessary resources to ensure that the home care meets the patient's needs, (3) Operational processes (processing of admission request, delivery of care, coordination and monitoring of care, etc), these are the most important processes in home care; indeed they manage the care in order to ensure the patient's satisfaction. Operational processes have already been considered in the literature [99][125]. One of their most important features is the "singularity", i.e. each process is specific to each patient. This is mainly reflected by the design of *Home care plan* which is specific to each patient [125].

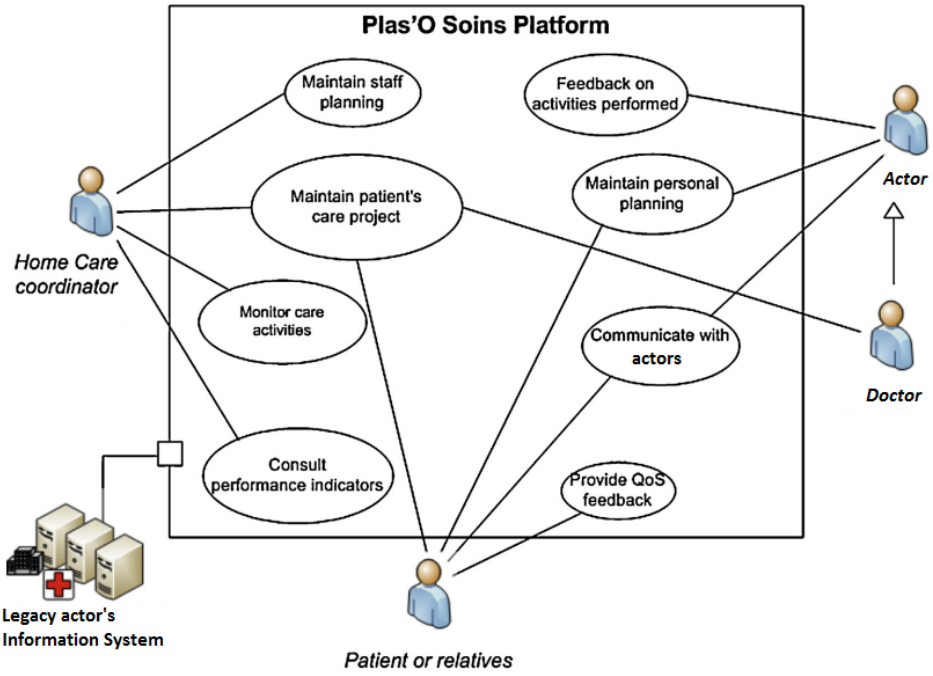

Figure 1.2: Use cases of the Plas'O'Soins platform

Indeed, with this configuration we can say that the home care plan is the core element, since it appears at the pre-admission phase and is used by the realization, the coordination and the monitoring phases. These phases may lead to its modification, which can vary from the addition of an activity to a complete overhaul of the home care plan.

1.1.3 Home care challenges

In the current situation, the structures that provide care and home services are faced with a main challenge related to the monitoring and coordination of activity, as described in the white paper published by the FNEHAD [51]. These challenges are: the coordination of cares and the continuity of cares.

Coordination of cares

The care coordination is defined in [67] as follows: "*The care coordination is a well thought out organization of patient care activities between two or more participants (including the patient) involved in the patient's care in order to ease the provision of appropriate health care services*".

Thus, care coordination requires essentially the coordination and communication be-

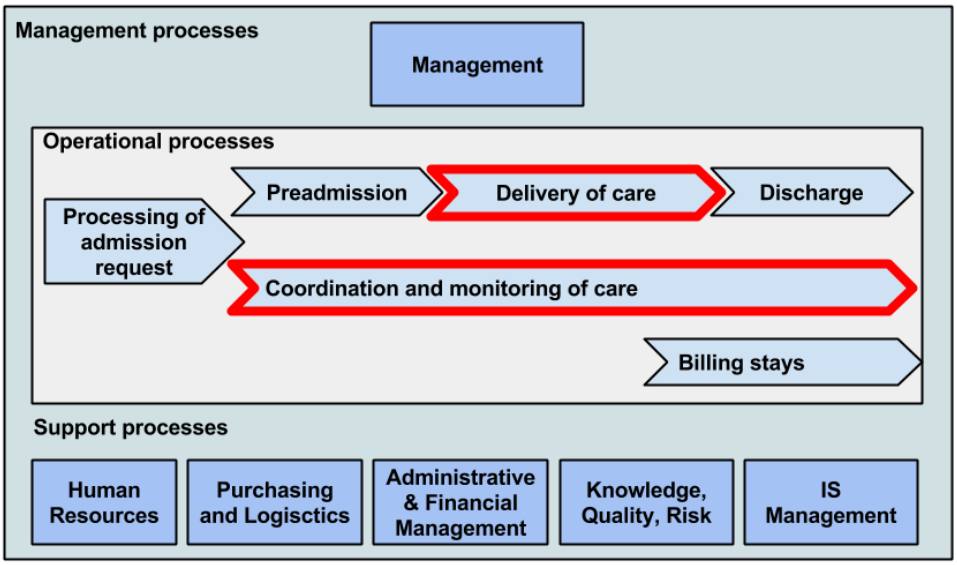

Figure 1.3: Main processes of home care (adapted from [51])

tween the different actors (care actors from different institutions, family, etc.) involved in the home care. Therefore, with the lack of inter-organizational communication and also between involved actors in a same structure. It is then important to provide a system which allows to assist the coordination of care, in order to ensure the fluidity of data transmission between involved actors in the patient's home care.

Continuity of cares

There are several definitions of the concept of the continuity of cares among which [45]:" *continuity of care consists in avoiding any interruption in the patient monitoring. It is ensured by coordination between practitioners (physician, nurse, etc) and it's an essential criterion of quality of care*".

Therefore, continuity of care deals with the ability to handle unexpected events and, hence, requires efficient communication between home care actors, implying a need for traceability of care provided and the history of the treated patients.

Many studies pointed out the lack of coordination and continuity of care in the home care area [20][99][125]. A common conclusion of these works lies in the need of automatic support to ensure the connection and information flows through the different involved actors or organizations. In response to this challenges, there is a considerable interest in exploiting the ICT solutions to enhance the quality and safety of the home care. The European Union undertaken a number of research projects in this direction: (1) Mobil-

ity (I2HOME), a solution allowing the elderly or disabled people to control the various electronic devices of their homes using their mobile phones or other objects [99]. (2) Remote monitoring (CAALYX, eCAALYX), solution dedicated to older or elderly persons at their home, it consists to collect five various vital signs and detect anomalies [99]. (3) COPLINTHO [3], a communication platforms allowing the interaction between all involved actors in the care process and which puts the patient in a central position, etc. We will present in what follows some ICT tools used in the home care area.

1.1.4 Overview on main software tools

There are a multitude of tools based on the ICT domain that improve the quality of the patient home care. Some are based on remote monitoring systems PROSAFE [28]), others on telealarm systems (presence verte[4] [125]), or systems enabling to remotely establish the medical diagnosis (ViSaDom [88], DIATELIC [5] [125]). In the following we present some tools in order to identify their specificity, their covered activities, their operating mode and their limitations.

- AtHome of ARCAN[6]: ARCAN is a software editor for managing personal home care coordination. AtHome is a software solution specific to each home care structure. AtHome manages all home care information (patient, billing, purchasing, inventory, communication and administration, etc.).

- Apozem[7]: is a software designed for some home care structures. Apozem manages all forms of home care support of the elderly people. It also manages the patient and his/her home care support (home care plan, transmission, etc.), and the edition of regulatory statements (medical register, annual activity report, etc.).

- Medlink[8]: is an integrated IT solution, specific to each home care structure. Medlink allows to improve the quality of care while reducing administrative tasks and costs. It also provides an access control (read, write, view) according to the role of involved actors.

[3] http://www.iminds.be/en/projects/2014/03/07/coplintho
[4] hhttp://www.presenceverte.fr
[5] http://www.diatelic.com
[6] arcan.fr
[7] http://www.medisys.fr/
[8] www.med-link.org

- Hippocad[9]: offers an application architecture based on a process approach and focuses on the operational home care of the patient's therapeutic project. Hippocad includes control services in the realization of home services, manages emergencies, unforseen and delay in a reactive manner, and integrates automatically the various reports.

In spite of the fact that these tools have the advantage of improving the home care support by providing means of traceability, driven alerts to the involved actors and information restitution in a real time (e.g., personalized report), etc., they are still far from meeting the identified difficulties. Indeed, these tools are not suitable, they are difficult to handle. They are not able to deal with patient constraints (patient constraints in a textual form and are unexploited, no patient agenda, etc.). The home care plan description is made in an unstructured manner, and the planned interventions are adjusted manually. And generally these tools lack of patient-centred sharing infrastructure.

1.2 Plas'O'Soins Project

Plas'O'Soins is a cooperative project financed by the ANR[10], the French National Research Agency (Agence Nationale de la Recherche). The project consortium brings together academic and industrial of the ICT and health domain and end-users from different home care structures in France. The objective of Plas'O'Soins is to set up a hardware and software platform designed to address the needs of the home care structures. The needs in terms of coordination, planning and monitoring of patient's care and actor's activities ensuring continuity of care for a patient in his overall care support.

The proposed hardware and software platform allows to store all useful information (patient's profile, protocols, prescriptions, interventions, analysis results, etc) and their evolution, in order to provide at any moment :

- The restitution of the current state of the patient's home care process, globally or with a specific vision tailored to the concerned actor's profile.
- The timely reporting of information (alerts, arrived on site, delays, schedule changes, etc) towards the appropriate actor whether a service emergency, a social actor, doctor, family, etc.

[9] http://hippocad.com/
[10] http://www.agence-nationale-recherche.fr/

- The planning and coordination of activities, and their dynamic re-planning.
- The consistency check of information integrity, traceability of activities and interventions.
- The help to the decision process by providing static data.

As depicted at Figure 1.3, the Plas'O'Soins platform is mainely concerned with the "delivery of care" and "coordination and monitoring of care" processes, while being interoperable with the tools that support the other processes, such as "billing" or "human resources". The platform features are grouped around the following processes(see Figure 1.4):

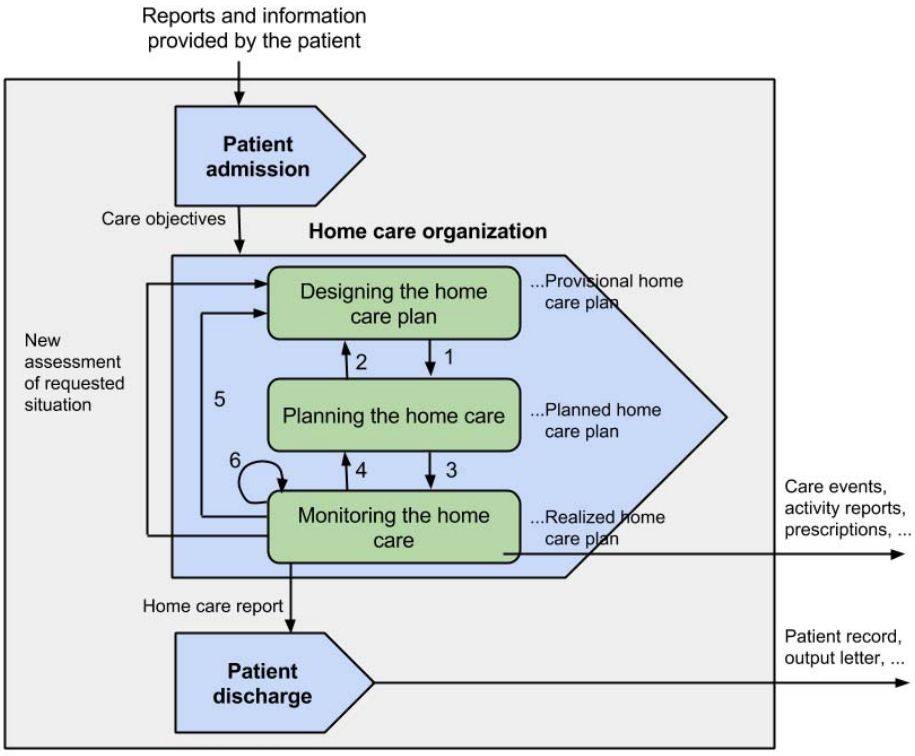

Figure 1.4: Processes managed by Plas'O'Soins platform

- **"Patient admission" process:** it handles patient-related information and his/her treatment that enable the creation or updating of patient records by the Plas'O'Soins platform.

- **"Patient discharge" process**: it allows the edition of the exit documents, as well as archiving and transmission of eventual information tailored to another concerned structure.

- **"Home care organization" process**: this central process of the platform is specific to each patient. Indeed the patient's home care is carried out through three main activities: the design of the home care plan, the planning of the home care and the monitoring of the home care. The design of the home care plan is the initial activity that determines the two others. Collaborations between these activities can be defined as shown in Figure 1.4 (the numbers refer to those in Figure 1.4):

  - **1**: First, the platform assists the coordinator in the design of the home care plan (provisional list of activities, involved actors, etc) which corresponds to the provisional home care plan. The information which compose the provisional home care plan are made available to the planning activity;
  - **2**: The planing activity indicates that the home care plan was planned within a determined time (e.g., week 51 of the current year). If a solution is not found, the home care plan should be reviewed and re-planned or manual planning is imposed by the coordinator;
  - **3**: The planned home care plan is available at any moment by the monitoring activity;
  - **4**: An identified problem at the care execution can lead to the modification of current planning, but keeping the same home care plan;
  - **5**: An identified problem at the care execution can require the modification of the current home care plan (and thus the creation of a new version of the home care plan) thus, a new planning is necessary;
  - **6**: An identified problem at the care execution may require warnings for continuing the home care support without trigger a change in the planning or a modification in the home care plan.

This decomposition into three activities induces three visions of the home care plan: (1) **the provisional home care plan**, which consists of scheduled activities to do at patient's home, (2) **the planned home care plan**, which consists of planned activities to do at patient's home, and (3) **the realized home care plan**,which consists of activities

already realized at patient's home. These three home care plans VISIONS must be managed simultaneously.

1.2.1 Our contribution within Plas'O'Soins project

Our work focuses on the design and analysis of the home care plan. To achieve this goal, we conduct the following steps:

- The Requirement analysis;
- The Definition of the Domain Specific Language (DSL);
- The definition of the formal model using Timed Automata theory;
- The implementation of the proposed model.

Note that, In the rest of this thesis, we will talk about the home care plan referring to the provisional home care plan. In the following, we will describe the requirement analysis. The other steps will be presented in the next chapters.

1.3 Requirement analysis

We conducted an on site analysis to understand the main concepts underlying the home care plan in the patient's home care. More precisely, we carried out interviews and observation phases with two home care structures: *UMT-Mutualité Tarnaise* and *Centre Hospitalier d'Albi*. During this phases, we were able to access and to collect several anonymized home care plans.

Thanks to the collected information, we could understand the manipulated data, the work of the involved actors especially the coordinators and the various difficulties related to the design of the home care plan.

Thus, home care plan is defined as a provisional list of care activities to be performed in the patient's home care. Home care plan encompasses all the services provided for a given patient and is essential to scheduling the delivery of such services at patient's home by health care professionals. More precisely, home care plan is a personalized list of care to achieve in a considered home care. It regroups *activities* which comprise a *frequency* (e.g., 3 times a week, everyday, on June 12th, etc), a *period* (e.g., from 01/01/2015 to 05/31/2015), an *interval* (e.g., morning, afternoon or evening), a *duration* in terms of

minutes (e.g., 30mn, 15mn, etc.), and a required *qualification* (e.g., nurse auxiliary, nurse, etc.). During the patient's home care period, the home care plan may change or evolve according to the evolution of the patient's health conditions.

In the following, we will present the results concerning this requirement analysis.

1.3.1 Involved actors in home care plan

Once the patient is supported by a home care structure, a comprehensive assessment of his/her needs must be taken into account, e.g., medical, social and psychological needs. This evaluation which usually takes place at patient's home, can be realized in one or more stages and involves various actors. Each actor brings his/her skills and contributes to the evaluation depending on the patient's health complexity. Among these actors we find social worker, therapist, attending physician, nurse, etc. Patient's family and helpers can contribute in this evaluation stage in order to be involved in the home care plan.

All the collected information will allow to the home care coordinator producing the home care plan.

1.3.2 Home care plan activities

A home care plan can be viewed as a collection of repetitive activities. A home care plan activity corresponds to the notion of medical or social activity (e.g., nursing activity or dressing, bathing activities). The care activity duration can vary from few minutes to hours. A same activity may occur several times in the same day. Some activities can take place at any time (e.g., monitoring of some parameters), others must be conducted in a fairly specific interval (e.g., between 8am and 9am for mobilization activity) or in a precise time (e.g., at 6pm for example for the lovenox injection activity). A home care plan activity is also defined by a period, which can be the same as the period of the home care plan or different (e.g., from 01/01/2013 to 03/31/2015) and the required qualification (e.g., nurse , nurse auxiliary, etc).

Irregular activities are inherent to the home care plans because the activity very often has to follow the evolution of needs of the patient. For example, a given activity *toilet* may be associated with complex frequency, e.g., every two days except Sunday evening.

1.3.3 Example of a home care plan

The home care plan is used in all home care structures and is mainly expressed through a paper format. Figure 1.5 shows a real example of home care plans that have been made anonymous for confidentiality reasons. The home care plan content may vary form one home care structure to another. Here we give the general elements that can be described in a home care plan.

- *Period* of the home care plan: corresponds to the period of validity of the home care plan (e.g., from 01/01/2015 to 01/01/2016).
- *Activities*: in the home care plan all the activities are described with their temporalities (e.g., Toilet every day in the morning, dress on Monday Wednesday and Friday evening, etc). Each activity is associated with a duration and an actor type (in some cases the number of involved actors is specified).
- *Medicine*: drug prescription by the attending physician is managed by the home care structure through the home care plan. Each drug is described by a name and temporalities which correspond to the administering time.
- *Material*: in the home care structures, the patient must have a material prescription in adequacy with his/her pathology. e.g., anti-bedsore mattress, medical bed, etc. Each material is associated with a temporal information.
- *Consumable*: a set of consumable is used to support the patient's home care. e.g., syringes, examination gloves, thermometers, etc.

As a part of this thesis, we focus on the activities.

1.3.3.1 Elaboration of home care plan

The home care plan is present in all home care structures. Designing and monitoring of the patient's home care plan are fundamental activities for a successful patient's home care. Figure 1.6 which is based on the corpus of on-site information analysis, shows how the patient's home care plan is carried out:

1. **Design the home care plan**. Firstly the coordinator creates the home care plan for each patient using a comprehensive assessment of the patient needs as well as his/here social and physical environment. The care plan is generally written in a paper format and can be updated at any moment by the coordinator;

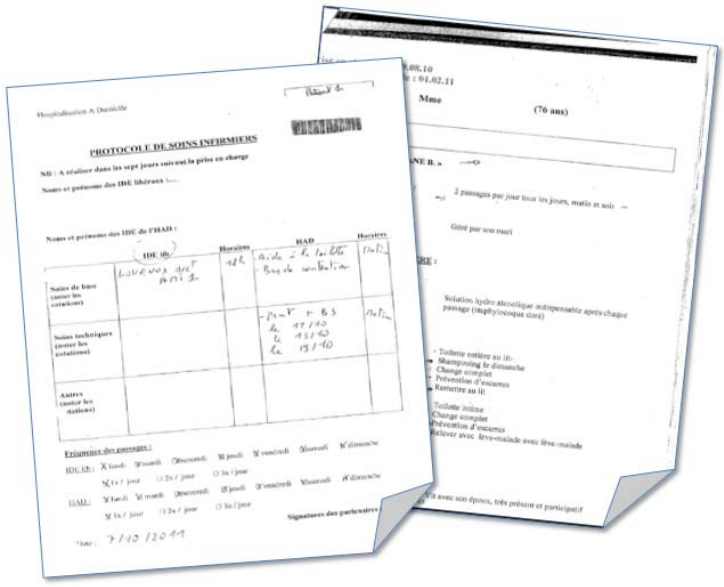

Figure 1.5: Example of home care plan

2. **Planning interventions.** Once the home care plan is created, it is used as an input for the planning activity. The latter is in charge to determine the intervention (start intervention time, end intervention time and the involved actor type). Planning activity must ensure that the patient and the involved actor are available in the defined time interval and that the required materials and consumable have been delivered at patient's home. In addition, to avoid too many movements of the actors, planning will try to group together activities that can be performed in the same time intervals. Some constraints occur in the planning activity such as: *involved actors constraints* (respect actor's agenda, respect zone wish, respect patient's wish, etc), *patient constraints* (respect patient's personal agenda, respect of the desired actor, etc), and *organism constraints* (respect of work schedule, respect of maximal duration of interventions, etc);

3. **Execution and monitoring of the home care plan**. Here the involved actors can inform the coordinator about these problems at the execution phases. The identified problems may either cause a modification at the planning phases, or modification of the home care plan (which leads to the creation of a new version of the home care plan), or lead to warning without causing modification in the home care plan.

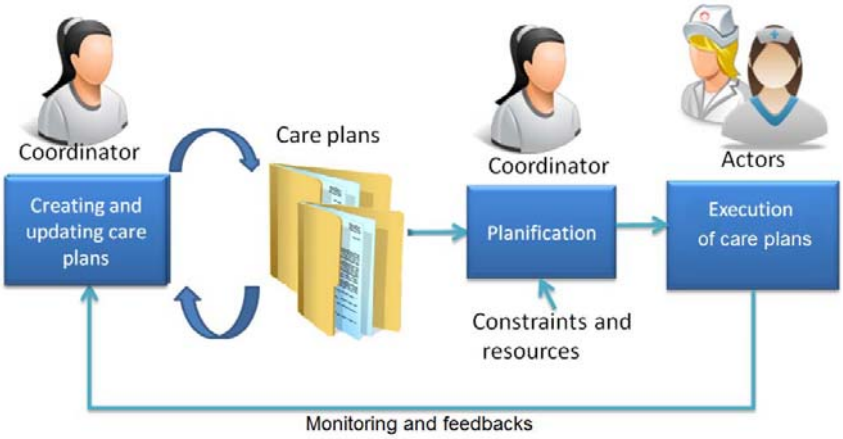

Figure 1.6: Process for the creation and the management of the home care plan

1.3.4 Problems related to the design and monitoring of the home care plan

As a conclusion of the on site analysis, we identified the following problems related to the design and monitoring of the home care plan. which are:

- A home care plan is usually elaborated in a paper format and so its handling is becoming more complicated. First, the number of admitted patients in home care structures increases which leads to an increasing number of home care plans that become difficult to manage such as the difficulty to update or to maintain multiple versions of a given patient's care plan. Another consequence is that the paper format lacks formal structure. Therefore, a written information may be not understood or may be redundant, etc [5]. Finally, in a long-term patient's home care, the involved actors accumulate knowledge about their patients without always plotted on paper. In this case, there are obvious risks for the safety of the patients.

- A home care plan is constructed through a complex process. Indeed, the realization process requires the coordination of an interdisciplinary team involving different actors coming from diverse medical institutions [44].

- A home care plan is an unstructured process in the sense that each patient must have his/her own specific home care plan. Indeed, a home care plan is always specified on an individual basis because each patient possesses specific characteristics which are necessary to take into account. It is not possible to design a unique process

capturing in advance the home care plans of all the patients.

- A home care plan is associated to complex temporal constraints. Indeed, it is necessary to specify the frequency (more or less regular) of every home care plan. This specification is expressed by health care professionals in natural language by using specific expressions: every day in the morning and in the evening, once every 2 days in the evening during 15 days, etc.

- A home care plan is needed for planning. In fact during the existing analysis in different home care structures, we noticed the absence of interventions generation mechanism. Everything is done manually by the coordinator. Manual support of the planning activity can in some cases (for a complex home care plan) increase the error risk. For example, planning an intervention that will be impossible because the coordinator forgot to take into account the involved actor's unavailability. Another possibility is the difficulty to propose an optimal planning. This difficulty is mainly due to the fact that the human capacity can not compute all possible combinations of interventions in order to choose the best configuration.

In this context, and within Plas'O'Soins project, we are interested in the problem underlying the elaboration of the home care plan by the coordinator.

1.4 Objective of the thesis

To cope with the aforementioned problems, we propose to adopt the following approach:

- The definition of the user-centred DSL, used to describe the main concepts of the home care plan at a high abstraction level. DSLs are already widely used in the fields of computer science an mathematics, and are recently used in the medical area [53][56][83][84][109][118]. The DSL will describe the home care plan activities and their relationships. It will be defined through a temporality model that will allow to specify intervals of realization for repetitive activities of the home care plan, where the repetitions can present irregularities and exceptions. This model will allow the coordinator to describe by himself the specificities of the home care plan. The DSL framework incorporates a GUI which is intended to be used in particular by the coordinator in their daily activities, which consists to manage and to evolve the home care plan, and to monitor the execution of the programmed activities by the involved actors at home.

- The definition of an automatic transformations of user specifications into formal model based on timed automata. This transformation is indispensable to perform any kind of verification and monitoring of the home care plan.
- The verification and monitoring execution of the home care plan. Among the verification that we have identified, there is a realizability checking of the home care plan, the intervention generation, and the monitoring of the home care plan. While checking the realizability is proper to our work, the monitoring and interventions generation are identified within the project. However, the last two are developed separately from our work using a traditional approach based on existing algorithm and on business rules. As part of this thesis we propose an approach based on timed automata.

1.5 Conclusion of Chapter 1

In this chapter, we presented the home care area and highlighted its main modeling challenges. Much work has already been achieved in this area, but there is still not a satisfactory solutions. We described the conducted requirement analysis which consist in carrying a set of interviews and collect and analyse some scripts documents. This requirement analysis has led us to define problems faced by the coordinator during the home care plan elaboration. Response elements were outlined, especially by addressing two scientific aspects regarding DSL and timed automata.

The domain specific language and the timed automata formalism will be subject of a detailed presentation in the next chapter.

Chapter 2

State of the art

Contents

In section 2.1 of this chapter, we present an overall view of the DSL concepts, its definition, its main advantages that have strengthened our choice together with some disadvantages, its development process steps, as well as some related work that used DSL in the medical field.

The section 2.2 is dedicated to the presentation of timed automata. We begin with some background knowledge on timed automata. Then we describe some extensions and subclasses of timed automata that we consider useful in the context of this thesis. We consider mainly the extension that takes into account activities with duration. Also, we present the software tools that enable model checking using timed automata. Before concluding, we present some existing works that use timed automata in the medical field.

2.1 Domain specific language

With the increasing of human needs, software engineering becomes more and more complex and sophisticated. Applications must be at the height of performance expected by the users. Over the years the trend of software engineering is to always increase the abstraction level of the programming languages [87].

The level of programming languages can be summarized as follows (1) low-level programming languages, which are more machine oriented languages (e.g., Assembler, etc.), (2) third-generation programming languages, which are more problem-oriented languages that try to answer to a wide class of problems (e.g., C, C++, etc.), and more recently (3) Domain Specific Language. While the (1) and (2) do not always permit optimal way to answer to a given problem, the DSL on the contrary allows to offer a solution more specific and less complex [78] [115].

Indeed, by focusing on a particular domain, it is easier to learn about the standard concepts of the domain and thus optimize and specialize the offered solution. Both for the programmer during development or modeling, than for the end user (who may not have *a priori* knowledge in computing) when using the solution.

2.1.1 Definition of DSLs

Several definitions of a DSL exist in the literature [52] [62] [73] [80] [121], depending on the application domain. In the technical domain for example we usually talk about domain specific programming languages, because they involve computer science experts and require programming competences. In software engineering or business domain, we

talk rather about domain specific modeling languages, because they are user-centred and require domain competences only [72] [103]. Among the domains where we find DSL, there are [121]: drawing and 3D animation [48] [66], financial products [13], telecommunications and telephony [71] [72], protocols [40], driver device [101], robotic [26] [96], and more recently the medical domain [77] [84] [118]. This variety of the applications domains shows the increasing importance of DSLs in the software engineering landscape. The development of a DSL in a domain is often argued by profits such as reliability, productivity and flexibility [68].

In the context of this work, we use DSL emphasizing more on the modeling language rather than the programming language. Because the solution that we offer is user-centred for the home care domain. Briefly, the domain specific modeling language (DSML) is a particular type of the DSL that is user-centred, and allows to propose a solution using a high level of abstraction. The solution corresponds to directly model the concepts domain. The DSML development enables users to fully engage in the proposed solution [50] [72]. Thus, users do not have to worry about the code, because it's generated automatically from the model using specific DSM tools[1] [114].

2.1.2 Main features of a DSL

Here we give the main advantages that prompted and still prompt today to development of the DSLs [121]:

- **Easier programming**: DSLs are defined by a syntax that is similar to that used by the domain experts, which enables these (usually) non-programmers to be able to fully engage in the language programming. Among the advantage of this implication, we find the reduction of the language learning effort by the domain expert and the ease of its understanding. In addition, DSLs unlike general-purpose programming languages are known to be concise, this is mainly due to the offered abstraction level. This features have the advantage to facilitate the developed programs analysis and thus to detect and correct errors earlier and quickly.

- **Improved safety/quality**: Unlike general-purpose programming languages, the use of DSLs enables automatic verification of a set of properties [111]. Thanks to the high level abstraction offered by the DSL, it is possible to use the traditional program analysis by reusing some verification tools [101]. These verifications allow

[1] http://www.dsmforum.org/tools.html

to quickly detect problems in the verified application and thus increase productivity. Moreover, focusing on specific domain enables both the optimization at compile time of the DSL as well as the development of specific tools such as, editors, debuggers, etc [72].

- **Systematic reusing**: Unlike general-purpose language that allows to abstract the common operations in libraries that are not always automatically reused, the DSLs are designed to lead to the systematic reuse. Indeed, DSL is characterized by a set of abstraction corresponding to concepts and knowledge domain. These abstractions are each of them associated with a code pattern that will be automatically generated and thus lead to the reuse of code at design time of the DSL. Besides, the code reuse in term of DSL also involves all the domain expertise that is captured during the design time of the DSL [101].

These features are introduced to ensure gains in productivity and quality in DSL-based approach [81]. Hence, drawbacks associated with this approach are identified in the literature

- **The cost**: Much effort should be given in the design and the implementation processes. It is also necessary to add cost of training and education for the end-users.

- **Portability and Compatibility**: Since DSL is designed for a particular domain, it lies in a non-standard and redundant language inducing hard problem in compatibility and portability.

Knowing the advantages and drawbacks facilitates the decision to define the DSL. Among the tools that are used to define DSLs there is Meta-modeling tools. Meta-modeling tools are development tools that allow to generate code from domain specific model [65]. Example of the most often used tools: (1) Eclipse modeling project [2], which provides development tools based on models. It consists on three frameworks (Eclipse Modeling Framework (EMF[3]), Graphical Editing Framework (GEF [4]), Graphical Modeling Framework (GMF [5])). (2) MetaEdit+ [6], which is an environment used for

[2] http://www.eclipse.org/modeling/
[3] http://www.eclipse.org/modeling/emf/
[4] http://www.eclipse.org/gef/
[5] http://www.dsmforum.org/tools.html
[6] http://www.metacase.com/products.html

creating modeling language and code generation [95]. And (3) XMF, a meta-modeling environment for language design. It combines a set of extended standard (e.g., MOF (Meta-Object Facility)[49], OCL (Object Constraint Language) [75], EBNF(Extended Backus-Naur Form)[104], etc.) [41]

2.1.3 DSL development

Figure 2.1 shows the two necessary steps to develop a DSL, namely the design and the implementation processes. In what follows, we will describe each step in more details.

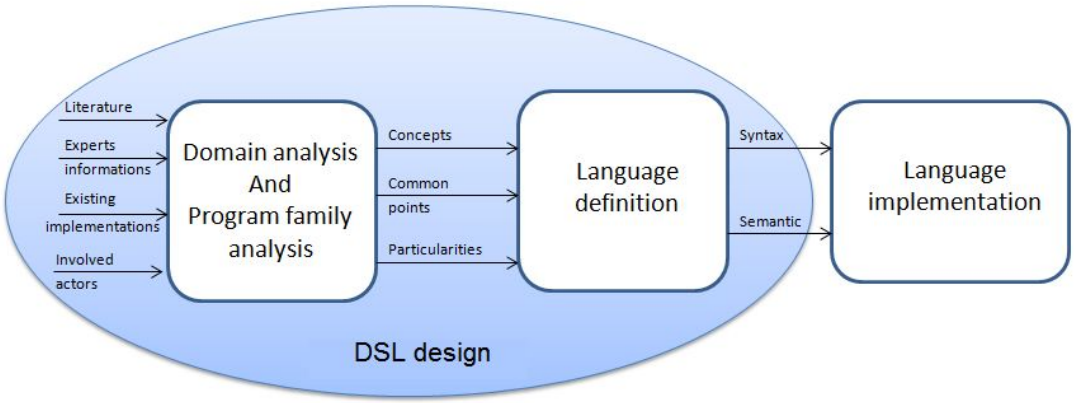

Figure 2.1: Design process of a DSL (adapted from [72])

2.1.3.1 DSL design

Designing a language based on DSL is a difficult task. Indeed, in the design process of a DSL much time and efforts are devoted to the analysis of the different domain elements. Especially, commonalities analysis between those elements, as well as the particularity of each of them. All these in order to design a language which is simple, readable, sufficiently expressive and easily understood.

In what follows, we present necessary steps to assist in the definition of the Domain Specific Language namely: *domain analysis*, *program family analysis* and *language definition*.

Domain analysis

The domain analysis concept is the first step in a DSL design process. It is considered

as an essential and major step in the domain identification approach. This concept has been defined for the first time by Neighbors [86] and was later revisited by McCain [76], Arango [12] and Pietro-Diaz [97].

The domain analysis consists in studying an application domain in a precise manner and determine its common information elements in order to reuse them systematically. This domain study is done in order to collect relevant information using different information sources, such as: interviews with domain experts, existing tools (systems), various documents (reports, manuals, etc.). Besides the common characteristic of the domain, the domain analysis identifies also the terminology, the key concepts of the domain and the various involved actors in the domain as well as their needs and objectives [43][111]. Even if in some cases the domain analysis allows to design a DSL, it remains insufficient in general.

Program family analysis

Program family analysis corresponds to a complementary approach of the domain analysis. Indeed, domain analysis is interested only to the commonalities between domain element and not to particularities. In [94], a program family analysis is defined as follows: "We consider a set of programs to be a program family if they have so much in common that it pays to study their common aspects before looking at the aspects that differentiate them". Thus, a program family in addition to being defined as a domain [86], it also has the particularity to take into account specific properties of the domain elements [111].

Several methodologies exist to develop a program family. [94] provides for example a methodology to develop a program family based on the successive refinement. Another newest work proposes an approach called FAST (Family-oriented Abstraction, Specification, and Translation) [122] that allows to define program families from identifying the domain terminology, commonalities among all members of the programs and variations.

The gathered information in the domain analysis and program family analysis are then used for the definition of the DSL. We will see in the following how this information are exploited to design a dedicated language.

Language definition

DSL design or more precisely language definition corresponds to define the syntax and semantic of the language.

Abstract syntax/Concrete syntax: Using the information gathered during the domain analysis and program family analysis phases, DSL designer should be able to define the syntax of the dedicated language. there exist two types of syntax: (1) The abstract syntax, which corresponds to the vocabulary and grammar of the DSL. And (2) the concrete syntax, which corresponds rather to the specification of the notation of DSL elements [65]. There are several notations for representing the DSL syntax, such as BNF (Backus-Naur Form), meta-modeling tools, etc.

Semantic: Once the syntax is defined it is necessary to define the semantics of the language in order to give it a sense. There are two types of semantics: *static semantic* and *dynamic semantic* [82]. Static semantic is interested in the verification program or model before its execution (e.g., type verification, etc.). Dynamic semantic is rather interested in the program or model behavior during its execution.

2.1.3.2 Implementation

There are several implementations techniques of DSL. The choice of one technique depends mainly on the DSL type whether internal or external [52]. Briefly, an external DSL is an independent language develop from scratch and it's not based on any other language. An internal DSL is a language developed on the basis of an existing language called the host language. External DSL implementation requires the creation of tools and support such as: the interpreter and compiler [81]. Internal DSL implementation requires the creation of a pre-processor [81] or extension of the compiler or interpreter of the host language [90].

2.1.4 Using DSL in medical field

Here, we cite two recent works concerning the use of a DSL-based approach in the medical field.

In the context of Medical Imaging System, [119] propose iDSL, a domain specific language that enables performance evaluation of the medical imaging system. Authors defines the grammar of the iDSL language based on a set of concepts (process, resource, system, etc.) relative to the domain. And its semantic through an automatic transformation into MoDeST (Modeling and Description language for stochastic Timed Systems) model [27]. The transformation into MoDest enables automatic performance evaluation

for model checking and simulation.

[37] proposes an approach for modeling the Clinical pathway using domain specific modeling languages. Clinical pathway is defined as a plan that enables to describe a set of patients goals, and how to achieve them in an efficiency manner [74]. Authors propose to develop the domain specific language considering some requirements such as, temporal dependencies and indefinite order relations between treatments steps, etc. The modeling language was implemented using MetaCase-tool and was tested on the example of "Wisdom tooth treatment".

2.1.5 Discussion with respect to DSL

Studying and analyzing the DSL development steps allow us to position our work as follows: The domain analysis and program family analysis correspond to our requirement analysis. The language definition corresponds to the definition of the temporalities language expressed using a BNF notation (we will give more detail in the next chapter). For verification and monitoring purpose, we opted for another DSL implementation approach based on formalizing using timed automata.

With respect to related work in the medical field, the approach proposed in [37] is close to our work. The proposed DSL may be associated in our case to the home care plan. However, authors have used the meta-modeling tools for implementing the DSL, which is not our case. In fact, as a part of this thesis, the DSL concepts allow us to describe the home care domain concepts through the UML class diagrams in collaboration with end-users, in order to define temporalities specification language which can assist the coordinator during the elaboration of the home care plan. The language implementation is achieved using among others HCI framework (Vaadin[7]).

2.2 Timed automata

Timed automata were introduced in [6] as a finite state automata enriched with a clock variable. Figure 2.2 shows a sample example of timed automaton A which is composed of a set of states (e.g., s_0 and s_1), where s_0 the initial state and s_1 the accepting state (final state), as well as transitions with labels over the alphabet $\Sigma = \{a, b\}$ and a clock which is a continuous variable over the set of real-valued numbers $\mathbb{R}^{\geq 0}$. Initially, the

[7] https://vaadin.com/home

clock is set to 0. While transitions are instantaneous, in every state, the time may elapse. In fact, the clock can be used to define constraints attached to the transitions and/or states. Constraints attached to the states are called *invariants*. They allow to specify that the automaton can remain in the state while the invariants conditions (boolean functions) are satisfied. Also, Constraints attached to the transitions are called *guards*. They allow to specify that the automaton can move from one state to another, depending on the attached constraints result. Here, the clock t is used in the guard of the transition labeled with b, means that b cannot be recognized when (b $\leq$ 3) is true. In addition, clocks are grow over the time and can reset upon triggering of transitions. For example here, the clock t is reset to 0 on the a-labeled transition from the state s_o to s_1.

Timed automata recognize *timed words* in the form of (a, t_0), ...,(a, t_n) where a is a symbol of the automaton and t_i is the time attached to each symbol a. The occurrences of times increase monotonically, i.e., $t_0 \leq t_0 \leq ... \leq t_n$. As an example w= (a, 0)(b,4) is a timed word which is accepted by the automaton of the Figure 2.2, where b has been recognized four units of time after a, while w= (a, 0)(b, 1) is not recognized by this automaton, since b can not be recognized before 3 units of time after a.

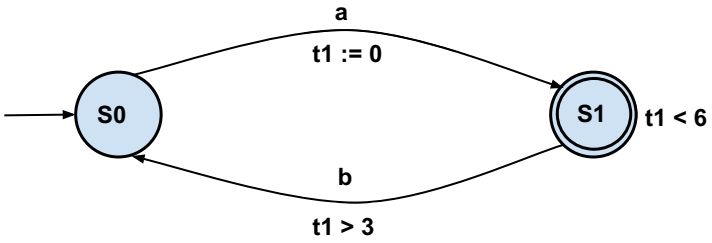

Figure 2.2: Timed automaton

2.2.1 Clock valuations

Based on the definition of [3] and [6], let X be a set of clocks over $\mathbb{R}^{\geq 0}$. A clock constraint is a formula x $\bowtie$ c, with x $\in$ X, $\bowtie \in \{=, <, \leq, >, \geq\}$ and c $\in \mathbb{N}$. We note ϕ(X) the set of clock constraints over X.

A valuation v for clocks of X is a function (v: X $\rightarrow \mathbb{R}^{\geq 0}$) that associates to each clock x the time value $v(x)$. We note $\mathbb{R}^{X}_{\geq 0}$ the set of valuation of X. Given d $\in \mathbb{R}^{\geq 0}$, v+d denotes the valuation that associates $v(x)$+d to every x $\in$ X. Also, given a subset of clocks r $\in$ X, [r $\leftarrow$ 0]v denote the clock valuation v' such that $v'(x)$=0 if x $\in$ r (i.e., a valuation

associates 0 value to clocks of r) and $v'(x)= v(x)$ if x ∈ X\ r.

Definition 2.2.1 ***(Timed automaton)*** *is a tuple* $A = (S, S_0, \Sigma, X, Inv, T, F)$ *where:*

- S *is a finite set of locations or states of the automaton;*
- $F \subseteq S$ *is a set of final states;*
- $S_0 \subseteq S$ *is a set of initial states;*
- Σ *is a finite set of transition labels;*
- X *is a finite set of clocks;*
- $Inv: S \rightarrow \phi(X)$ *associates an invariant to each state of the automaton;*
- $T \subseteq S \times \Sigma \times \phi(X) \times 2^X \times S$ *is a set of transitions. A transition (s, a,* ϕ*,* λ*,* $s^{'}$*) represents an edge from location s to location* $s^{'}$ *on symbol a.* ϕ *is a clock constraint, and the set* $\lambda \subseteq X$ *gives the clocks to be reset after firing such a transition.*

The set of timed words recognized by the automaton of Figure 2.2 constitutes the timed language noted L(A).

2.2.2 Semantic of timed automata

The semantics of timed automaton is defined as a transition system where each state of the automaton is represented by a pair (s, v) where $s \in S$ and v is a valuation over X s.t. v satisfies the invariant $Inv(S)$. The semantic of a timed automaton is then expressed in terms of two types of transitions [6]:

- **Action transition**: this type of transition can be triggered instantaneously if the current value of the guard is satisfied. Formally:

For a state $(s, v)$and a real-valued time increment d $\succeq 0$, (s, v) $\rightarrow^d$ $(s, v + d)$ if for all 0 $\preceq$ d' $\preceq$ d, v+ d' satisfies the invariant Inv(s).

- **Time transition**: this type of transition consists of staying in the same state and increasing the values of clocks respecting the invariant. Formally:

For a state (s, v) and a transition (s, a, ϕ, λ, $s^{'}$) such that v satisfies ϕ, $(s, v) \rightarrow^a$ ($s^{'}$, v[λ :=0]).
Resuming the timed automaton shown in Figure 2.2, a possible run of the automaton is: $(s_0, 0) \rightarrow^{0.5} (s_0, 0.5) \rightarrow^1 (s_0, 1.5) \rightarrow^a (s_1, 0) \rightarrow^1 (s_1, 1) \rightarrow^3 (s_1, 4) \rightarrow^b (s_0, 0)$.

2.2.3 Properties of timed automata

We recall briefly in this section some interesting properties on timed automata. Note that we use as a reference the original model proposed by Alur and Dill [6]. Indeed, decidability and complexity results vary depending on the considered timed automata model [6].

- **Closure of timed automata**

  Timed automata are closed under the union, projection and intersection however they are not closed under the complementation [6]. The union and intersection closure of timed automata are based on a same closure as for untimed automata [59]. The closure under projection[8] is reduced to replace transitions with ε (empty word) on the not mapped alphabet. Concerning the non closure of the complementation, a detailed proof was presented in [6] and a new proof can also be found in [10].

- **Language inclusion**

  The *timed language inclusion problem* consists, given two timed automata A and B in deciding whether or not L(A) $\subseteq$ L(B). This problem has been shown undecidable since it can be reduced to A $\cap\bar{B}=\emptyset$ and knowing that timed automata are not closed under complementation [6].

- **Langage equivalence**

  The *timed language equivalence problem* consists to check if L(A)= L(B). This problem has been shown undecidable since it can be reduced to L(A) $\subset$ L(B) and L(B)$\subset$ L(A) [6].

- **Universality**

  The *universality problem* of timed automata consists to check, if a timed automaton A defined over an alphabet Σ, can recognize all possible timed words over Σ. In [6] it is shown that this problem can be reduced to the inclusion problem, making the universality problem undecidable.

- **Reachability and langage emptiness**

[8] Projection enables to map a timed automaton A to timed automaton B by replacing the alphabet which is not in B by ε.

The *rechability/emptiness problems* are fundamental properties that make easy the verification of some systems behavior. Indeed, the *emptiness problem* consists to check if L(A)= $\emptyset$ and the *reachability problem* consist to determine if there is a run that allows to reach some identified states of the automaton. These problems are equivalent since, checking if the language L(A) is empty, consists to ask if there is an execution (run) that allows to reach a final state (or some other states) from the initial state of the automaton. These problems have shown to be *Pspace*-complet and the detailed proof can be found in [6][10].

These decision problems, especially the non closure under complementation, make difficult the use of timed automata to solve some verification problems. This situation has thus prompted the emergence of some variants and extensions of timed automata that allow to remedy to this problematic.

2.2.4 Variants of Timed automata

In order to solve the decision problems that are undecidable, several studies and researches have been realized on the model of Alur and Dill [6]. We expose in the following some interesting classes and extensions of timed automata which will be useful in the context of this work.

2.2.4.1 Deterministic timed automata

A timed automaton is called deterministic if it satisfies the following conditions:

1. Only one start location;

2. Given two transitions from the same source and with the same alphabet, their clock constraints are disjoint. i.e., for all s $\in$ S, for all a $\in \Sigma$, for every pair of transitions (s, a, ϕ_1, λ, s') and (s, a, ϕ_2, λ, s''), the clock constraints ϕ_1 and ϕ_2 are mutually exclusive.

In [6], it has been proved that the deterministic timed automata is closed under the complementation (can be complemented), thus languages inclusion is decidable. Deterministic timed automata are strictly less expressive than (non-deterministic) timed automata [10]. The problem of the determinization of non-deterministic timed automata is undecidable [116].

2.2.4.2 Event-clock automata

The class of *Event-clock Automata* has been defined in [7]. This model which englobes *Event-recording automata* and *Event-predicting automata* has been introduced to make decidable universal problem and the languages inclusion problem of timed automata. An *Event-recording automaton* is a timed automaton containing, for each symbol a, a clock that records the time of the last occurrence of a. An *Event-predicting automaton* is timed automaton containing clocks which provide the time of the next occurrence of a symbol. The problem of the determenization [9] of the event-recording automata is decidable.

2.2.4.3 Timed automata with silent transitions

Timed automata with silent transitions correspond to timed automata augmented with a *non observable action*-labeled transitions, also called ε-transitions. Among the properties of timed automata with ε-transitions is that they are more expressive than timed automata without ε-transitions, more specifically when ε-transitions reset clocks, i.e., $s \rightarrow^{\varepsilon,g,\{x\}} s'$ [22][31][46]. Several methods were proposed to remove ε-transitions when they do not reset clocks [21][46]. These methods are complex and can only be applied when ε-transitions do not reset clocks [22]. Timed automata with ε-transitions are not close under complementation [6][21].

2.2.4.4 Timed automata with action duration

There exist a limited works that propose modeling action with duration using timed automata. Among the few existing models we found: *Timed Automata With non Instantaneous Action Transition* and *Durational Action Timed Automata* (DATA). Timed Automata With non Instantaneous Action Transitions was introduced for the first time in [102] in order to specify the actions of systems which can take some time to be completed. Thus to model a non instantaneous actions in this type of model, each transition must be equipped with two kinds of constraints, *initiation-constraint* (to specify the beginning of the action) and *completion-contraint* (to specify the end of the action). It has been shown in [16][102][110] that Timed Automata With non Instantaneous Actions are more expressive than timed automata and less expressive than timed automata with $\varepsilon - transition$.

[9] To make deterministic a non-deterministic timed automaton

DATA were introduced in [19]. The major difference of this model compared to the model of timed automata is found in the associated states invariants. Indeed each state of the automaton (except the start state) has a termination condition of the action potentially in execution. This condition is of the form $\{x \geq t\}$, where $x \in X$ and $t \in \mathbb{R}^+$. It allows the system to control the action termination without forcing it to leave this state once action is executed (the system can stay there indefinitely). It has shown in [70] that DATA is closed under all boolean operations.

2.2.5 Software tools

In order to support verification and analysis of the modeled systems using timed automata, several model checkers have been proposed. Among the best known, Kronos[35], Hytech [112], and UPPAAL [18]. We explain in what follows the general principle of model checking, and describe the model checker that we have chosen as a part of this work.

2.2.5.1 Model checking

Model checking consists in testing a model, i.e. if it satisfies or not some properties. Figure 2.3 illustrates the principle of model checking. The model checker can be defined as the tool that allows to realize the model checking, since it receives as input the model and the property to verify, and provides as an output the answer regarding the property (if it's satisfied or not) [15].

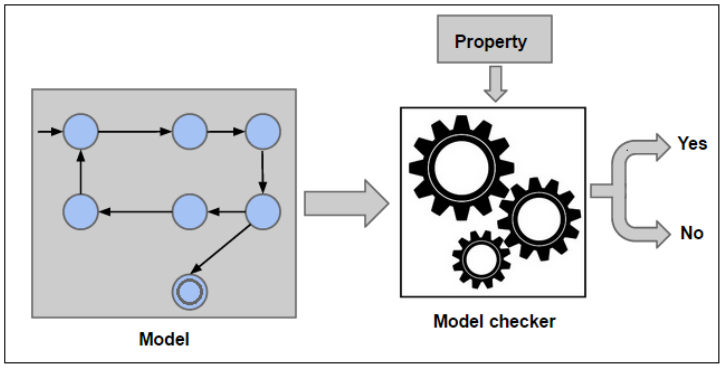

Figure 2.3: Principle of model checking

Several properties have been defined in the literature. They are generally classified in the following categories: *Reachability properties* which verify if a given state can be

reached or not; *Safety properties* which state that under some conditions, something bad will never happen in the system; And finally *liveness properties* which state that under some conditions, something Good will happen in the future [15].

There are several tools for model checking, among them, Kronos, Hytech and UPPAAL. Kronos is a tool that allows to verify the timed automaton. Among works that use kronos [29][89]. Hytech is a tool that allows to verify an hybrid extension of the timed automata [10] and reachability and safety properties [57][112]. UPPAAL is a tool that allows to verify timed automata characterized by a set of clocks, set of channels for synchronisation automata, urgent actions, etc., and checks properties such as reachability, safety, liveness and deadlock [31].

As a part of this thesis we will use the UPPAAL Model Checker that happens to be the most used in the timed system verification. Its main advantage over Kronos or Hytech is that it's easy to use thanks to its graphical user interface, in particular a simulation module that facilitates the testing phase and helps to faster detect eventual errors.

2.2.5.2 UPPAAL Model Checker

UPPAAL is a tool for modeling, simulating and verifying real time systems. UPPAAL uses an automaton extension. Among the elements of UPPAAL model:

(1) *Start state*: Each automaton is composed of a single start state represented by a double circle.

(2) *Guards*: Guards allow to express the conditions that must be satisfied on transitions. These conditions are timed conjunction constraints and constraints of integer variables.

(3) *Reset operation*: It corresponds to an initialization of the value of the clock.

(4) *Channels, synchronization and urgent*: UPPAAL allows to verify a networks of timed automata, communication between automata is done through communication channels enabling a message exchange. Synchronization is the communication element between the automata. It enables the simultaneous crossing transitions between automata (e.g., (send) a! and (receive) a? corresponds to a normal synchronization between one transmitter and one receiver). UPPAAL allows to declare a channel as being urgent in order to prevent automata to linger in a state.

(5) *Invariant*: It corresponds to conditions associated with the states of the automaton, expressed as a constraint on the clock values.

[10] An hybrid timed automata combines both continuous and discret behavioral [2]

UPPAAL tools

The UPPAAL graphical user interface (UPPAAL GUI) comprises three parts:

1. A modeling environment;
2. A simulation environment;
3. A verification environment.

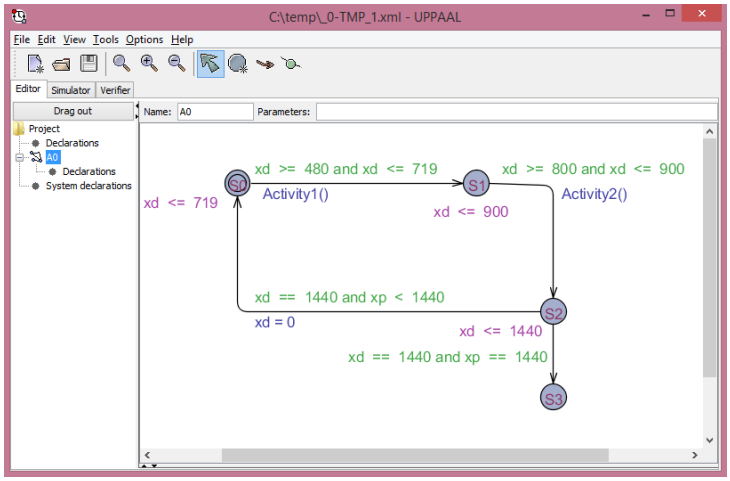

Figure 2.4: UPPAAL: modeling environment

The UPPAAL editor as shown in Figure 2.4 allows to construct a set of automata, define a tree automata, and configure and declare variables.

The UPPAAL simulator as shown in Figure 2.5 allows to simulate the execution of automata and to follow the execution evolution. Simulation with UPPAAL can be controlled by the user, in this case he may even choose by himself the transition to follow, or the simulation can be automatic. In this case the tool selects randomly transitions to follow.

The UPPAAL verifier as shown in Figure 2.6 allows to perform the verification of some properties. For example, can the automata reach the state f from the initial state? These properties must be written in a formal manner in order to be understood by the tool. The user can add or remove properties through this interface and at the end of the verification an answer is displayed to indicate whether or not the property is satisfied.

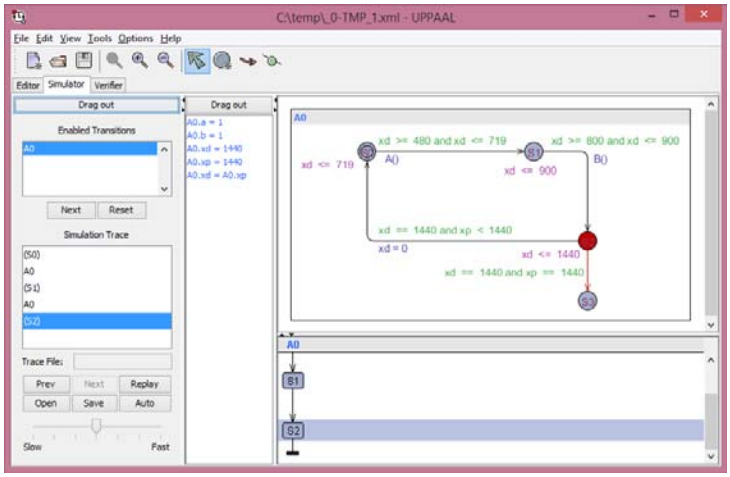

Figure 2.5: UPPAAL: simulation environment

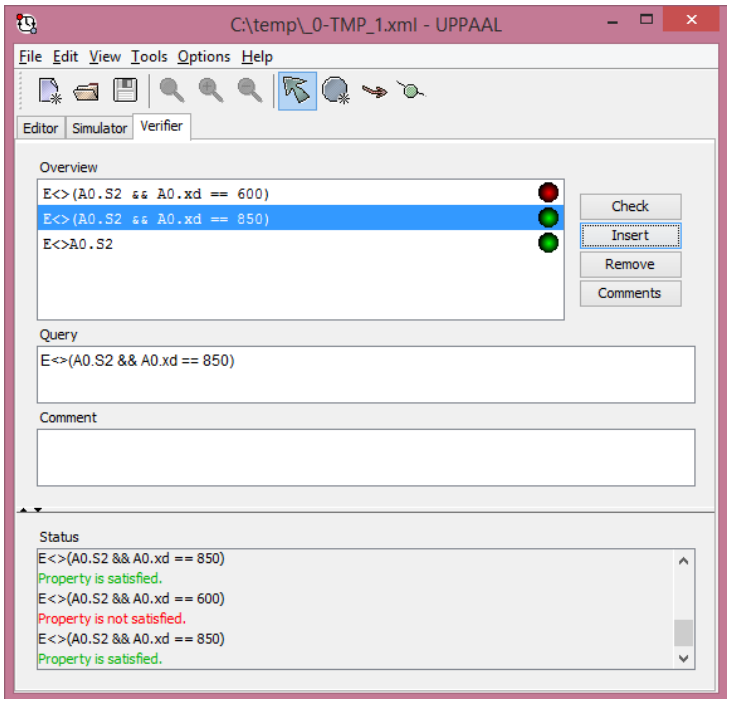

Figure 2.6: UPPAAL: verification environment

UPPAAL property specification language

UPPAAL uses a subset of (Timed Computational Tree Logic) TCTL logic [8] to specify the properties to verify. Briefly TCTL logic is a timed extension of the Computational Tree Logic (CTL) [42]. TCTL was defined in [4], it corresponds to a logical tree which

is interpreted on timed transitions systems [31][108]. The TCTL formulas that are supported by UPPAAL [54]:

- $A[]\psi$: For any execution, the property ψ is always verified;
- $A <> \psi$: For any execution, there is a state where ψ is verified;
- $E[]\psi$: There exist an execution where ψ is always verified;
- $E <> \psi$: There exist an execution leading to a state where ψ is verified;
- $\varphi \rightsquigarrow \psi$: Whenever φ is satisfied , then eventually ψ will be satisfied.

2.2.6 Timed automata in the medical field

Several studies involving methods based on formal model exist in the literature. Up to our knowledge there are only few works which concern the medical domain. We review in what follows some of these works.

In the context of Ambient Assisted Living (AAL)[120], [85] proposes a model based on a network of timed automata (i.e., parallel composition of several timed automata) to tackle the problem of modeling a risk detection system for elderly's home care and allows to generate alarm if there is evidence of problem. The model enables to represent the patient's environment, characterized by a house equipped with a wireless sensor network in each room, each bed/armchair and a magnetic sensor at the entrance door, as well as patient's behavioral, and decides to generate an alarm if it detects some evidence of problem. The critical properties (e.g., reachability, safety and liveness) were verified using UPPAAL Model Checker.

[69] proposes a formal method based approach to the development of the Generic Patient Controlled Analgesic (GPCA) infusion pump. The objective was to develop methodologies that ease the safety property verification. Authors use the timed automata model to approach the problem of safety-assured development of the pump software. Thus, they propose to transform manually the (GPCA) model expressed in Simulink[11] and Stateflow[12] into a network of timed automata, which was then verified with respect to a set of generic safety properties.

[11] http://fr.mathworks.com/products/simulink/
[12] http://fr.mathworks.com/products/stateflow/

[106] proposes a formal approach to model procedure of the medical treatment described in the Medical Guidelines (GLs) in the case of *Imatinib* (drog) dose adjustment protocol. Authors propose to formalize medical treatments described in GLs using extended timed automata with Task (TAT)[11]. Briefly TAT corresponds to timed automata where each state is associated with a piece of code. Semantic of TAT is the same as for TA extended with queue. The resulting formal model was verified using TIMES toolbox [113].

As part of Model Driving Design (MDD), [92] proposes a model translation tool, which allows to convert automatically a verification model based on timed automata to a model which can be simulated and tested using Simulink/ Stateflow. Knowing the strong needs of testing and verification of a medical device, authors propose to experiment their tool on an implementable cardiac pacemaker. They propose among others to model the pacemaker software using timed automata. Each timing cycle of the pacemaker was modeled using timed automata with a local clock. The model was then verified using UPPAAL and translated to be tested by Simulink. The experiment showed that the tool preserves the behavior of the pacemaker model from UPPAAL to Stateflow.

Always using implementable cardiac medical devices, [126] proposes a methodology for testing and verifying the proper functioning of a medical device within a *closed loop of the patient*[13]. To do so, they define a real time Virtual Heart Model (VHM) to model the proper functioning and malfunctioning of the heart. The VHM was modeled using a network of timed automata which allows to capture the timing properties of the heart. As use case, a pacemaker device has also been modeled using a network of timed automata. Tests and experimentation showed a clinically-relevant response.

2.2.7 Discussion with respect to timed automata

The study of some extension and subclasses of timed automata have allowed us to better understand what exists, in order to reuse or redefine the model that is most suited to the constraints that we want to model in this thesis. In fact, as a part of this thesis, we opted for timed automata with $\varepsilon - transitions$ and invariants in the states. The different results of expressiveness [22][31][46] have strengthened our choice.

Given a non-atomicity of activity that we will handle, it was necessary to make an overview of what is being done in the modeling of activity with duration using timed au-

[13] Closed-loop context of the patient is " a function of both the environment and the input from the device controller and must be captured by the device evaluation process" [126]

tomata. The two works above, both enable to represent the activity with duration, nevertheless they fail to respond to our expectations as they are represented. Indeed, Timed Automata With non Instantaneous Action Transitions make heavier the constraints on transitions and modifies the timed language acceptance by breaking this latter into *initiation* and *completion* language acceptance. Add to this the fact that it's less expressive than timed automata with $\varepsilon - transitions$.

Regarding DATA, they can control the duration of an activity. However, if two activities (or more) follow each other, there is no way to control the time that elapses between these two activities. This feature is completely excluded in our case, since, we must not only control the duration of the executed activity, but also control the time interval between activities. Add to this, the two Timed automata models with non Instantaneous Action Transitions and DATA can not be used directly in the existing verification tools but they must be transformed first.

From this finding, we propose in the next chapter a formal model which will allow us, among others, to control the execution time of activities to respect the time interval between activities, and which can be directly used by verification tools such as UPPAAL [18].

Concerning the related work, all described works use the timed automata formalism in the medical field. But while some of them simply reuse the timed automata as it is defined in Alur and Dill, others propose extension (e.g., TAT) to better adapt to their case studies. However any of these works have been interested in modeling semi-structured systems, none has taken into account the duration of the activity and also, except for the verification (e.g., reachability, safety), monitoring was not mentioned in these works.

2.3 Conclusion of Chapter 2

In this chapter we introduced the paradigms of domain specific languages. We have presented its various benefits and the design steps necessary to its definition. Then we exposed the formalism of timed automata on which this thesis is based. In the last part of this chapter, we mentioned the few works that use timed automata in the medical field. These works ignore several aspects of modeling such as the activity duration and the monitoring. In the next chapter, we will present the considered formal model which is based on timed automata formalism. We try through this formalism to take into account properties which we consider absent in other works.

Chapter 3

Modeling and analyzing home care plan using timed automata

Contents

As mentioned before, the design of a home care plan highlights several difficulties. To tackle such difficulties, we propose in this thesis an approach based on three specific components (Figure 3.1): (1) A DSL which provides a user centred specification language for the involved actors; (2) A formal model based on timed automata which defines an automatic transformation from user centred specifications to timed automata, and (3) A checking module which uses the resulting timed automata from (2) to support an automatic verification and monitoring. This approach will give rise to a tool that will be described in the next chapter.

This chapter is organized as follows. Section 3.1 describes the DSL based approach in which we mainly identify elementary temporal expressions. The general modeling process is presented at section 3.2 together with the construction of the proposed automata, i.e., pattern automata, activity automata and care plan automata. Section 3.3 presents some verification and monitoring issues. We conclude at section 3.4.

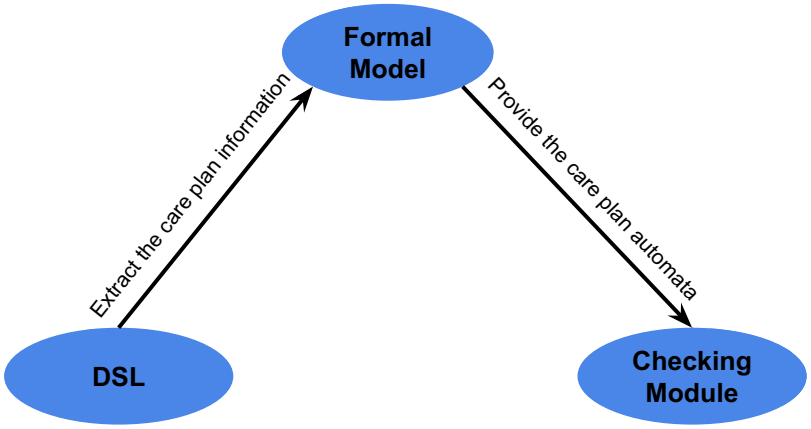

Figure 3.1: Underlying processes to the elaboration of the home care plan

3.1 A DSL-based approach for specifying home care plans

The design of a home care plan is a complex collaborative process, managed by a primary medical coordinator and carried out by an interdisciplinary team. In order to understand such a design process and also to understand how a medical coordinator approaches the problem, we conducted in the context of the *Plas'O'Soins* project a thorough on-sites

analysis of current practices in the field of home care. In particular, we carried out interviews with different professionals of home care institutions and we realized several analyzes of key documents and procedures. This study showed the central role played by care plans as primary components of effective care coordination at patient's home. Indeed, the proposed DSL approach deals with inherent concepts to the home care area. That's why we have defined the DSL for home care as being the set of objects that may be involved in the home care plan and manipulation that can be done on these objects. In the following, we define the main building blocks of the DSL, as well as the temporal specification used to express regular or irregular repetitions occurring in the home care plan.

3.1.1 The main building blocks

The proposed DSL provides high level abstractions that can be used by a care coordinator to design a home care plan for a given patient. Figure 3.2 summarizes the main building blocks of the DSL in the UML class diagram. This figure highlights the main concepts of a DSL tailored to express home care plans. It corresponds to only concepts that interest our work, which are:

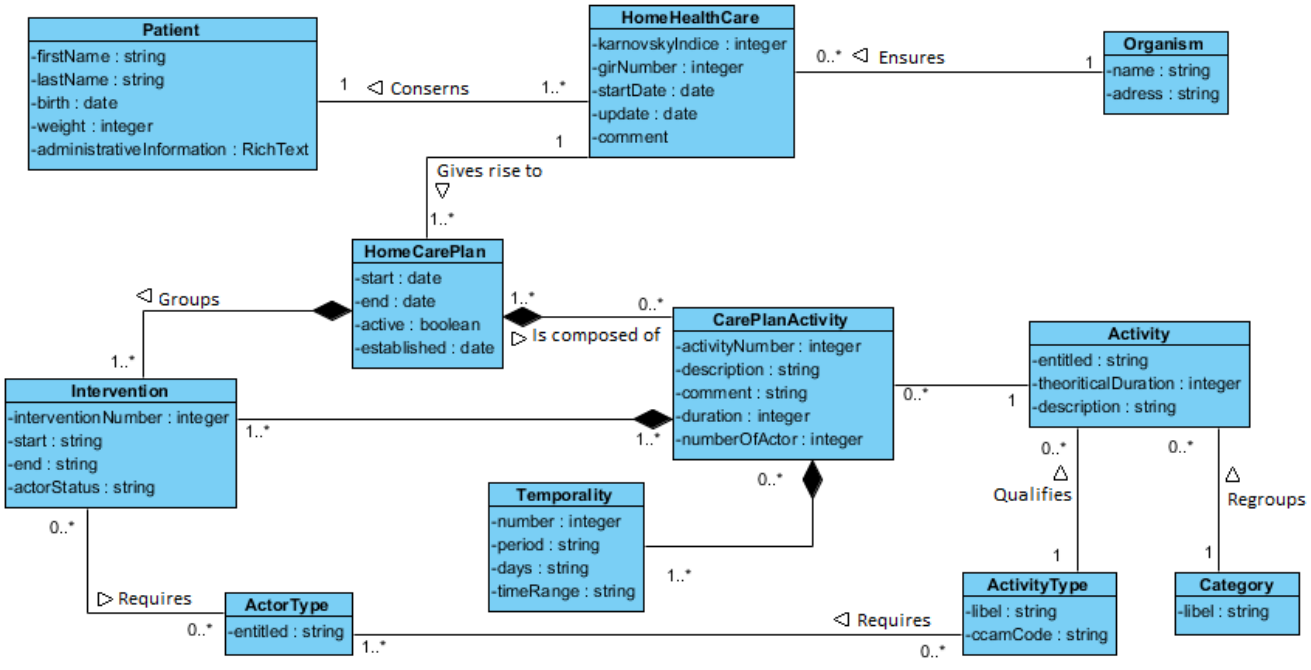

Figure 3.2: UML class diagram for a home care plan (partial)

- **Patient**: it is important to have the main features of the patient involved in the home care plan elaboration, such as: personal information (last name, first name, phone number. etc.), physical characteristics (weight to take into account eventual weight overload), comments (wish of the patient about the actor type, etc.). The input of this information is provided by the coordinator.

- **Home health care**: includes everything about medical prescription, the health care type, the therapeutic project, etc.

- **Care plan activity**: an activity denotes a medical or social service provided to persons in their own homes. The proposed DSL includes several predefined activities identified by our analysis of the application domain. Examples of the predefined activities are: (i) Health services: monitor medications, drug injection, aftercare, etc. (ii) Activities of daily living: bathing, assist with meal planning and preparation, dressing, maintain clean household, etc. Each activity is associated with description which provides additional information about the activity, in particular, a description of an activity includes the required qualification of the actors who are allowed to carry out the activity as well as the temporal constraint expressed as quadruplet (Period, Days, Time ranges, Duration), where:

  - **Period** specifies the time period during which the activity is defined;
  - **Days** indicates the days within a period in which an activity must take place;
  - **Time ranges** indicates the time slots in which the activity can occur;
  - **Duration** specifies theoretical duration which corresponds to the average of already observed durations.

- **Intervention**: an intervention is a collection of activities that can be scheduled together. Interventions are defined by grouping together activities that can be performed by a same actor type and which occur in the same time range. Interventions may be specified manually by the coordinator or computed automatically from the specifications of the activities and then proposed to the coordinator for validation.

Besides the aforementioned concepts, the proposed DSL is enriched with additional constraints derived from the domain knowledge (e.g., medical knowledge represented in ontologies, etc).

3.1.2 User centred specification of home care plan

Each activity of the care plan is associated with a set of elementary temporal specifications. These specifications provide the information about the time when the activity should be performed, expressed as a quadruplet (Days, Time ranges, Period, Duration), and allow us to provide a language that can express regular or irregular repetitions of an activity in some period under a condensed form, similar to that used by doctors (e.g. Everyday morning and evening).

Thus, the elaboration of the home care plan is mainly based on the specification of the time which characterizes activities. In order to make the specification of the time much easier to the coordinator, we propose a temporal specification language which better capture the most often used temporal expressions.

Indeed, the conducted analysis in the home care area led us to suggest a temporal specification language with a high level of abstraction based on three forms of elementary specifications [30]. Each instance of the quadruplet (Days, Time ranges, Period, Duration) corresponds to an elementary specification.

For the formal definition of the temporal expression, we use a BNF notation. Briefly, an element inside embraces associated with one of the following symbols ? $*$ + means respectively that the element may be present or absent, that it is present an indefinite number of times and that it is present at least once.

Temporal expression

We define below each element of the temporal specification, i.e., Period, Days, Time ranges. Knowing that Duration corresponds to the time duration of home care plan activity.

- **Period**
  1. <period> ::= <starting-date> "-"<ending-date>?
  2. <starting-date>::=/a date with the format mm/dd/yy/
  3. <ending-date>::=/a date with the format mm/dd/yy/
- **Days**
  1. <days-expression-form1>::= <date>+|"holiday" | <date>+"holiday"
  2. <days-expression-form2>::=<day-of-week>+["except("<date>*["holiday"]")"]

3. <days-expression-form3>::="everyday"|"every(n)days("<date>")"["except("<date >* <day-of-week>* ["holiday"]")"]
   where:
4. <date>::=/a date with the format mm/dd/yy/
5. <day-of-week>::= "monday"| "tuesday" |"wednesday" | "thursday" |"friday"| "saturday" | "sunday"

- **Time ranges**

1. <intervals>::= (<integer> "times") |<interval>+
   where:
2. <integer>::= /number of occurrences of the activity in the day/
3. <interval>::=<string-form>|<pair-form>|<hour-form>
4. <string-form>::="morning"|"midday"|"afternoon" |"evening"|"night"
5. <pair-form>::=<starting-hour>"-"<ending-hour>
6. <hour-form>::=<hour> /an hour with the format hh:mm

Note that Days and Time ranges can take different forms (patterns). Patterns associated with Days can be: (a) Absolute dates, expressed as <days-expression-form1>, specifies the specific date in a time range (holidays is used to describe all the dates of public holidays); (b) Relative days, expressed as <days-expression-form2>, specifies days of a week; or (c) Everyday, expressed as <days-expression-form3>, specifies that the activity occurs repetitively everyday or every n days from a given date. Patterns associated with Time ranges are periods in the day (e.g. morning, afternoon, evening, night), time interval (e.g. 10h-11h), or a starting hour of the activity (e.g. 10h).

Combination of elementary specifications allows to express specifications of irregularities and exceptions. In fact, to avoid ambiguity, when a same day occurs in different elementary temporal specifications, the coordinator must use exceptions. An exception is introduced via the keyword except.

Table 3.1 shows a simple example of a specification using this language. Each row of the table corresponds to an elementary temporal specification. The first two for the activity Toilet and the last two for the activity Dress. The case of the activity Dress shows an example where an exception is needed. In fact, we have an exception on 04/20/13, since on 04/20/13 the activity takes place only in the morning. The general form of a

Activity	Days	Time ranges	Period	Duration
Toilet	Monday Wednesday Friday	morning evening	01/01/13-03/31/13	30
	Sunday	morning	01/01/13-03/31/13	
Dress	Everyday except(04/20/13)	morning evening	01/01/13-03/31/13	20
	04/20/13	morning	01/01/13-03/31/13	

Table 3.1: Specification of activities **Toilet** *and* **Dress**

specification is therefore (E1) except (Day1 Day2 Day3 ...) where (E1) is a specification that can be expressed by the quadruplet (days, time range, period, duration) and Day1 Day2 Day3... are days to exclude from E1 (we call them "exception Days").

Roughly speaking, the notion of a legal schedule of a care plan activity is defined as a sequence of allowed instances of this activity which satisfies the set of temporal specifications. An example of a legal schedule for the activity Toilet of Table 3.1 corresponds to the sequence: Toilet at 09h00 on 01/01/13; Toilet at 18h on 01/01/13; ...; Toilet at 10h00 on 03/31/13.

The proposal of temporality language has led to the development of a GUI to support a coordinator in designing a home care plan. We will present in the next chapter this GUI.

3.2 General modeling process with timed automata

From the graphical user interface, the coordinator can specify the home care plan which consists on different activities associated with their temporal specifications. Recall that our main objective is to build the care plan timed automata. To achieve this objective, we propose a three steps approach (Figure 3.3): (Step 1) *Pattern automata construction*, which consists in mapping between elementary temporal specifications and timed automata called *Pattern automata*, (Step 2) *Activity automata construction*, which consists in combining patterns automata to build the *activity automata* using our composition algorithm, and (Step 3) *Home care plan automata construction*, which consists in constructing the global *care plan automata*.

This modular approach allows an incremental definition of a home care plan (which may be complex or not) by progressively aggregating, at first time, *Pattern automata*, and at the second time, *Activities automata*.

Among the advantages of this approach, is to facilitate the reuse of some components, in particular*Patterns automata* which makes the construction of the care plan more easier than if it is built directly from the specified temporal specifications, as well as to facilitate

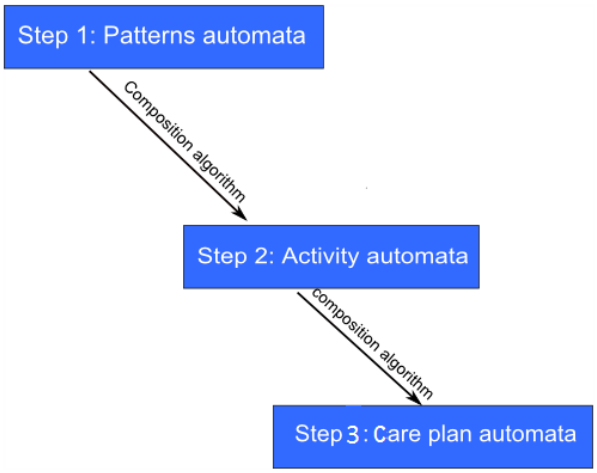

Figure 3.3: The three steps approach

the control and monitoring of the various home care plan components. To achieve this tasks we rest on timed automata theory to propose a new composition operator suitable to our context. This operator enables to capture some specific properties of activities, example duration of the activities of the home care plans. Dealing with a duration raises significant challenges, since one has to ensure that some specific properties are satisfied such as: prevent interleaving of activities, ensure continuing of activities executions from the beginning till the end of the specified duration, etc. This makes the definition automata much more complex. In fact, the notion of duration is not well mastered, since most of the existing works considered timed automata without duration. In our approach, we handled the problem of duration as follows:

- We extend the basic model of timed automata with a notion of the execution state which is particular state where the automaton waits for the execution of the activity according to specified duration. It should be noted that the extension does not increase the expressive power of the basic automata, and hence, does not impact their computational properties.
- We designed carefully the composition operator in order to take into account duration in the composition process.

In the remainder of this chapter, we present the formal model based on timed automata and we provide a detailed presentation of *pattern automata*, together with the construction of the *activity automata* and *care plan automata*.

3.2.1 A formal model based on timed automata with duration

The formalism of timed automata is used to model home care plans. As already said, several variants of timed automata have been proposed in the literature. We consider in our work timed automata with ε-transitions (i.e., silent transitions) and *invariants* (i.e., guards on the states). The activities of care plan are not instantaneous but have a duration. This is why we need to capture the notion of duration of an activity in our timed automaton. This is achieved through the identification of the notion of *execution states* of an activity automaton. More precisely, we distinguish between three kinds of states in a given activity automaton: the *start states* where the automaton stays before the activity starts, the *execution states* where the activity is executed, and the *waiting states* where the automaton indicates that the activity has been terminated. Note that to achieve this goal, we slightly modified the definition of timed automata to introduce this set of states namely: *waiting states* (denoted W), *execution states* (denoted E) and *start states* (denoted St). It is worth noting that this set is used to construct "pattern automata" and by the composition operator to generate the activity automata and the care plan automata. The proposed modification doesn't impact the semantics of the generated automaton and hence standard tools, such as UPPAAL can still be used to handle the verification task. A formal definition of the extended timed automaton is given below:

Definition 3.2.1 ***(timed automata (with an implicit duration))***
Let $A = (S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ *be timed automaton where:*

- S *is a finite set of locations or states of the automaton with* s_0 *the initial state,* $F \subseteq S$ *is the set of final states,* $W \subseteq S$ *is the set of waiting states,* $E \subseteq S$ *is the set of execution states, and* $St \subseteq S$ *is the set of start states;*
- Σ *is a finite set of transition labels including* $\{\varepsilon\}$*;*
- X *is a finite set of clocks. W.l.o.g., we assume that the time unit is in minute, because there is no precision beyond the minutes in the home care plans;*
- $Inv: S \rightarrow \phi(X)$ *associates an invariant to each state of the automaton;*
- $T \subseteq S \times \Sigma \times \phi(X) \times 2^X \times S$ *is a set of transitions. A transition* $(s, a, \phi, \lambda, s')$ *represents an edge from location* s *to location* s' *on symbol* a*.* ϕ *is a clock constraint, and the set* $\lambda \subseteq X$ *gives the clocks to be reset after firing such a transition.*

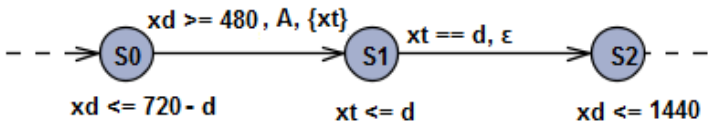

Figure 3.4: Example of a part of timed automaton for a task A

As an example, Figure 3.4 shows the timed automaton corresponding to an activity A that must be done in the time interval 8h-12h (i.e., the activity can be at least started at 8h or at most at (12h-d)) and having a duration d. At the beginning, the automaton is at the state $s_0 \in St$ then it starts the execution of the activity A when it enters the state $s_1 \in E$. The automaton stays at this state for the whole duration d of the activity A then moves to $s_2 \in W$. The automaton uses the clocks $\{x_d, x_t\}$, invariants and transitions guards to control the execution of the activity A. The clock x_d is used to control the execution of the activity within a day, the value $x_d <= 720 \quad d$ of the invariant means that the activity Toilet must begin before 12am-d. The clock x_t is used to control the activity duration, when the transition labeled with $x_t == 0$ is fired this means that the activity A has been executed with respect to the fixed duration.

This timed automaton recognizes timed words. For example *timed word* = (A, 480)...(A, 2400).(A, 3840) is an execution which is accepted by the automaton of Figure 3.4, where A belongs to Σ and the occurrences of time increase monotonically, i.e., $t_0 \leq ... \leq t_n$.

3.2.2 From elementary temporal specifications to pattern automata

Each elementary temporal expression is formalized in the form of a timed automaton called *Pattern automaton*. There are several temporal expressions that can be used by care professionals, such as: once a day, every n days, every another day, etc. However, in our work we focus on the most used temporal expressions i.e., absolute dates pattern (days-expression-form1), relative days pattern (days-expression-form2) and everyday pattern (days-expression-form3). We present in the following each pattern automaton in detail.

3.2.2.1 Relative days pattern

Relative days pattern is used to express a regular repetition of the activity of the care plan. For each row of temporality expressed as (Relative days, Time ranges, Period, Duration) defined for an activity a of the care plan, the corresponding timed automaton pattern A_{RD}= $(S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ is defined as follows :

- S is a finite set of states, with s_0 the initial state. The total number of states is: $NbStates = 3+(NbTimeRanges - 1) * 2 * NbDays+ NbDays$. Where $NbTimeRanges$ is the number of times ranges and $NbDays$ is the number of specified Days;
- F is the set of final states. We always have one final state;
- $\Sigma = \{\text{Activity name}\} \cup \{\varepsilon\}$ is the set of transition labels, where ε defines the silent transition;
- $X = \{x_d, x_t, x_p, x_w\}$ is the set of clocks, where x_d is used to control the execution of the activity within a day, x_t is used to control the activity duration, x_w is used to control the execution of the activity in a day of the week and x_p is used to control the execution of the activity in a day of the period. W.l.o.g., we assume that the time unit is the minute;
- Inv =$\{\forall s \in S, Inv(s) = (x_d \leq EndTimerange - d)$ and $s \in St, Inv(s) = (x_d \leq 24h)$ and $s \in W, Inv(s) = (x_t \leq d)$ and $s \in E\}$;
- $T \subseteq S \times \Sigma \cup \{\varepsilon\} \times \phi(X) \times 2^X \times S$ is the set of transitions. Each transition corresponds to a day of the week. The number of transitions is: $NbTransitions = 3+(7 - NbDays)+NbDays * 2 * NbTimeRanges$. Where 3 corresponds to the two return transitions to the initial state (the first one to reset x_d in the same week, another one is to reset x_w in order to move to the next week in the period) and the transition to the final state. ϕ is a clock constraint.

The timed language accepted by the automaton A_{RD} is the set of timed words associated with accepting runs. It is formulated as follows:

$L(A_{RD}) = \{w/w = \prod_{k=0}^{((end(P)-start(P))/(7*24h))-1}(\prod_{i=0}^{m-1}(\prod_{j=0}^{n-1}(a, t_{i,k}^j))), t_{i,k}^j \in [d_i + k * 24h * 7 + start(P) + start(I^j), d_i + k * 24h * 7 + start(P) + end(I^j)]\}$.

This timed language $L(A_{RD})$ consists in concatenating successively the timed word (a, $t^j_{i,k}$) according to three parameters: the existing time ranges in a day j $\in$ [0,n-1]; the different days of the week i $\in$ [0,m-1] and finally the different weeks contained in a period k $\in$ [0,((end(P)-start(P))/(7*24h))-1] where Period is defined by start(P) and end(P), Time ranges by start(I^j) and end(I^j). The d_i are the different specified relative days and k is the index of the week.

Hence, $L(A_{RD})$ corresponds to all the schedules of a given activity which satisfy the corresponding relative days constraint.

Activity	Days	Time ranges	Period	Duration
Toilet	Thursday except(05/01/14)	morning	05/05/13-05/05/14	30

Table 3.2: Specification of activity Toilet

Consider for example the specification of the activity Toilet (Table 3.2). According to this specification, the activity Toilet must be performed every Thursday except on 05/01/14 which also corresponds to Thursday (i.e., the activity Toilet must be repeated every Thursday of the period except on Thursday 05/01/14). The corresponding timed automaton A_{RDE}=$(S, s_0, \Sigma, X, Inv, T, F, W, E, St)$, depicted in Figure 3.5, is defined as follows:

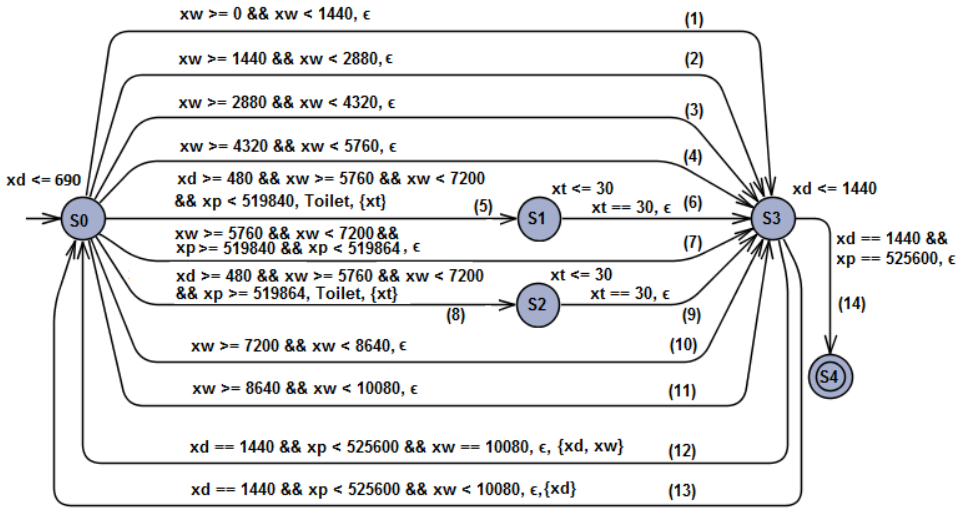

Figure 3.5: Example of a timed automaton in case of relative days (here Thursday except(05/01/14))

- S= $\{s_0, s_1, s_2, s_3, s_4\}$ with s_0 the initial state (which is also the start state for the activity Toilet), $W = \{s_3\}$ and $E = \{s_1, s_2\}$;

- F= $\{s_4\}$;
- $\Sigma = \{\text{Toilet}\} \cup \{\varepsilon\}$;
- X= $\{x_d, x_t, x_p, x_w\}$;
- T is the set of transitions where:
  - Transitions (1, 2, 3, 4, 10 and 11) express the fact that the activity Toilet does not take place during the other days of the week except Thursday ie. (Saturday, Sunday, Monday, Tuesday, Wednesday and Friday). However, Thursday is represented by three transitions labeled differently: the transition 5 expresses the fact that the activity Toilet is done every Thursday that are in the period before the 05/01/14, transition 6 is used to respect the fixed duration of the activity. Same thing for the transitions 8 and 9 but this time the activity Toilet is done every Thursday that are in the period after 05/01/14. At the end, the transition 7 expresses the fact that on Thursday 05/01/14 the activity Toilet is not performed.
  - Transitions 13 and 12 are respectively used to reset the clock of days in the same week of the period and to reset the clocks of days and weeks in the same period.
  - Transition 14 is used to express the end of the period and terminates the execution of the automaton.

For example a *timed word* = (Toilet, 6240).(Toilet, 16320).(Toilet, 26400).(Toilet, 36480).(Toilet, 46560).(Toilet, 56640).(Toilet, 66720).(Toilet, 76800).(Toilet, 86880).(Toilet, 96960).(Toilet, 107040).(Toilet, 117120).(Toilet, 127200).(Toilet, 137280).(Toilet,147360). (Toilet, 157440).(Toilet, 167520).(Toilet, 177600).(Toilet, 187680).(Toilet, 197760).(Toilet, 207840).(Toilet, 217920).(Toilet, 228000).(Toilet, 238080).(Toilet, 248160).(Toilet, 258240).(Toilet, 268320). (Toilet, 278400).(Toilet, 288480).(Toilet, 298560).(Toilet, 308640). (Toilet, 318720).(Toilet, 328800).(Toilet, 338880).(Toilet, 348960).(Toilet, 359040).(Toilet, 369120).(Toilet, 379200).(Toilet, 389280).(Toilet, 399360).(Toilet, 409440).(Toilet, 419520). (Toilet, 429600).(Toilet, 439680).(Toilet, 449760).(Toilet, 459840).(Toilet, 469920). (Toilet, 480000).(Toilet, 490080). (Toilet, 500160).(Toilet, 510240). is an execution which is accepted by the automaton of Figure 3.5

3.2.2.2 Absolute dates pattern

Absolute dates are used to define precise dates for an activity of the care plan, e.g. 12/12/2013, 12/14/2013, etc. For each elementary temporal specification expressed as (Absolute dates, Time ranges, Period, Duration) defined for an activity a of the care plan, the corresponding timed automata A_{AD}= $(S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ is defined as follows :

- S is a finite set of states, with s_0 the initial state. The total number of states is: $NbStates = NbDates + 3 + (NbTimeRanges - 1) * 2 * NbDates$. Where $NbTimeRanges$ is the number of times ranges and $NbDates$ is the number of specified dates;
- F is the set of final states. We always have one final state;
- $\Sigma = \{\text{Activity name}\} \cup \{\varepsilon\}$ is the set of transition labels;
- $X = \{ x_d, x_t, x_p\}$ is the set of clocks expressed in terms of minutes where x_d is used to control the execution of the activity in a time of the day, x_t is used to control the activity duration and x_p is used to control the execution of the activity in a day of the period;
- $Inv = \{\forall s \in S, Inv(s) = (x_d \leq EndTimeRange - d)$ and $s \in St, Inv(s) = (x_d \leq 24h)$ and $s \in W, Inv(s) = (x_t \leq d)$ and $s \in E\}$;
- $T \subseteq S \times \Sigma \cup \{\varepsilon\} \times \phi(X) \times 2^X \times S$ is the set of transitions. The number of transitions between two states of time range depends of the number of the specified dates for an activity. The number of transitions $NbTransitions = 2 + NbTimeRanges * 2 * NbDates + NbIntervals$. Where $NbDates$ is the the number of specific dates when the activity will be performed during the period, $NbIntervals$ is the number of intervals [Start date, End date] where the activity will not be performed, and 2 corresponds to the return transition to the start state and the transition to the final state.

The timed language accepted by the automata A_{AD} is the set of timed words associated with accepting runs. It can be formulated as follows:

$L(A_{AD}) = \{w/w = \prod_{i=0}^{m-1}(\prod_{j=0}^{n-1}(a, t_i^j)), t_i^j \in [D_i + start(I^j), D_i + end(I^j)]\}$.

This timed language $L(A_{AD})$ consists in concatenating successively the timed word (a, t_i^j) according to two parameters: the existing time ranges in a day j $\in$ [0,n-1] and each date i $\in$ [0,m-1]. Time ranges are defined by start(I^j) and end(I^j). D_i are different specified dates.

Hence, $L(A_{AD})$ corresponds to all the schedules of a given activity which satisfy the corresponding absolute dates constraint.

The example bellow illustrates the mapping of an elementary temporal specification into timed automaton using absolute dates pattern. Table 3.3 shows the specification of the activity Toilet that takes 30 minutes.

Activity	Days	Time ranges	Period	Duration
Toilet	05/06/13,05/08/13,05/09/13	morning	05/05/13-05/05/14	30

Table 3.3: Specification of activity Toilet: Absolute dates pattern

The corresponding timed automaton A_{AD}= $(S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ depicted in Figure 3.6 is defined as follows:

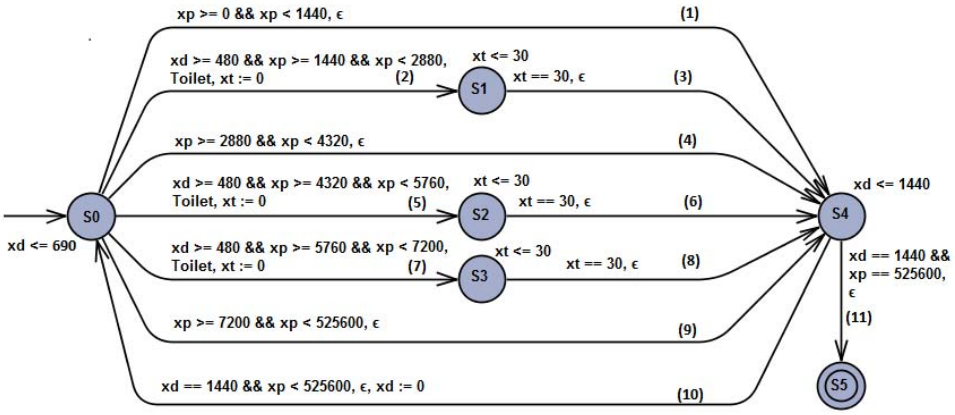

Figure 3.6: Example of a timed automaton in case of Absolute dates

- S= $\{s_0, s_1, s_2, s_3, s_4, s_5\}$ is a finite set of states with s_0 the initial state,W= $\{ s_4\}$ and E=$\{s_1, s_2, s_3\}$;
- F=$\{s_5\}$;
- $\Sigma = \{Toilet\} \cup \{\varepsilon\}$;

- X= {x_d, x_t, x_p};
- T is the set of the following transitions:
  - Transition (2) specifies that when the automaton is at state s_o, it can move to state s_1 to trigger the execution of the activity *Toilet* in the state s_1 on condition that the conjunction of state invariant ($x_d \leq 690$). Where 690 minutes = (12h*60minutes)-30 minutes of activity duration at s_0 and the transition guard ($x_d \geq 480$ && $x_p \geq 1440$ && $x_p < 2880$) allow it. ($x_d \geq 480$) ensures that the activity *Toilet* can be performed only in the morning and ($x_p \geq 1440$ && $x_w < 2880$) ensures that the activity *Toilet* takes place in the date 05/06/2013 of the period. The transition (3) is used to respect the fixed duration of the activity. The state invariant ($x_t \leq 30$) ensures that the automata stays at state s_1 during 30 minutes to perform the activity *Toilet*. Same principle for transition (5 and 6) for 05/08/2013, and transitions (7 and 8) for 05/09/2013.
  - Transitions (1, 4, 9) are triggered when x_p is outside the specified dates. Transition (1) represents the interval between the beginning of the period and the first specific date. Transition (4) represents the interval between the two specific dates. Transition (9) represents the interval between the last specific date and the end date of the period. For those transitions the automata can move from s_0 to s_4 at any moment where ($x_d \geq 0$ && $x_d < 690$), the guard is composed of a single condition on the clock x_p, it ensures the validity of the interval where the activity should not be done.
  - Transition (10) enables the automaton to move back from s_4 to s_0, without performing any activity, at the end of the day (i.e., when x_d equals 1440) within the specified period (i.e., $x_p < 525600$). Upon this transition, the variable x_d is reset to record the beginning of a new day.
  - Transition (11) is fired at the end of the period (i.e., when x_p equals 525600) and enables the automaton to move to the final state s_5 and terminate the execution.

For example the *timed word*= (Toilet, 1920).(Toilet, 4800).(Toilet, 6240) is an execution which is accepted by the automaton of Figure 3.6

3.2.2.3 Everyday pattern

Everyday combined with time ranges and period can be used as a particular case of Relative days. There are two ways to represent the corresponding temporal specification pattern. One is based on the representation of Relative days where all the seven transitions of a time range will be labelled with the activity name. And the other way is illustrated in the following example.

Consider the following elementary temporal specification of the activity Toilet that takes 30 minutes:

Activity	Days	Time ranges	Period	Duration
Toilet	Everyday	morning	05/05/13-05/05/14	30

Table 3.4: Specification of activity Toilet: Everyday pattern

The corresponding elementary temporal timed automaton $A_{ED} = (S, s_0, \Sigma, X, Inv, T, F, W, E, St)$ depicted in Figure 3.7 is defined as follows:

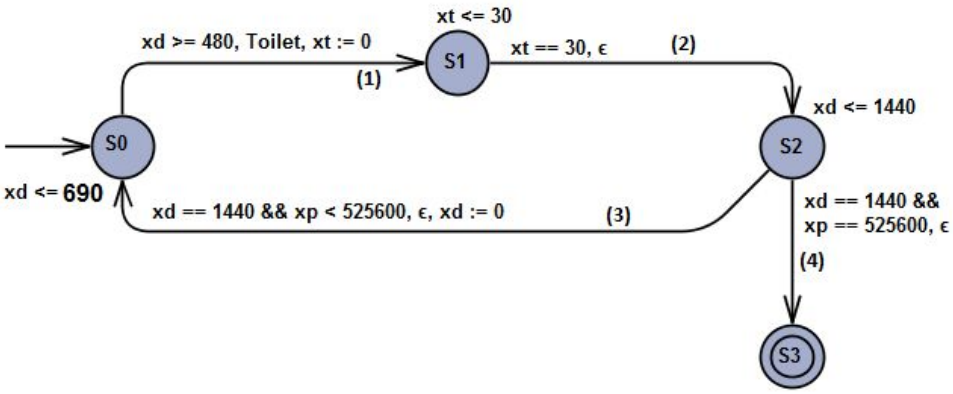

Figure 3.7: Example of a timed automaton in case of Everyday

- S is a finite set of states, with s_0 the initial state. The total number of states is: $NbStates = NbTimeRanges * 2 + 2$ where $NbTimeRanges$ is the number of times ranges. W= $\{s_2\}$ and E= $\{s_1\}$.
- F=$\{s_3\}$;
- $\Sigma = \{Toilet\} \cup \{\varepsilon\}$;
- X= $\{x_d, x_t, x_p\}$;

- T is the set of transitions where $NbTransitions = NbTimeRanges * 2 + 2$. We will see the meaning of each transition:

  - Transition (1) specifies that when the automaton is at state s_0, it can move to state s_1 to trigger the execution of the activity Toilet in the state s_1. The conjunction of state invariant ($x_d \leq 690$) at s_0 and the transition guard ($x_d \geq 480$) ensure that the activity $Toilet$ can be performed only in the morning (i.e., when the value of x_d is between 480 and 690). Transition (2) is used to respect the fixed duration of the activity. The state invariant ($x_t \leq 30$) ensures that the automata stays at state s_1 during 30 minutes to perform the activity Toilet.
  - Transition (3) enables the automaton to move back from s_2 to s_0, without performing any activity, at the end of the day (i.e., when x_d equals 1440) within the specified period (i.e., $x_p < 525600$). Upon this transition, the clock x_d is reset to record the beginning of a new day.
  - Transition (4) is fired at the end of the period (i.e., when x_p equals 525600) and enables the automaton to move to the final state s_3 and terminate the execution.

The timed language accepted by the automata A_{ED} is the set of timed words associated with accepting runs. It can be formulated as follows:

$L(A_{ED}) = \{w/w = \prod_{i=0}^{((end(P)-start(P))/24h)-1}(\prod_{j=0}^{n-1}(a, t_i^j)), t_i^j \in [i * 24h + start(P) + start(I^j), i * 24h + start(P) + end(I^j)]\}$.

This timed language $L(A_{ED})$ consists in concatenating successively the timed word (a, t_i^j) according to two parameters: the existing time ranges in a day j $\in$ [0,n-1] and the day i $\in$ [0,((end(P)-start(P))/24h)-1]. Period is defined by start(P) and end(P), Time ranges by start(I^j) and end(I^j).

Hence, $L(A_{ED})$ corresponds to all the schedules of a given activity which satisfy the corresponding Everyday constraint.

3.2.3 Activities Automata

Now that we have seen how to build the "Elementary temporal specification timed automata" using temporal specification patterns, we will see here how to construct an activity automaton. As said previously, care plan activity is characterized by a set of

temporal specifications. In order to build activity automaton we have to combine its associated patterns automata. For this purpose, we define a specific composition operator noted "$\bigodot$". This operator mixes the asynchronous product (or shuffle) on some states and a specific synchronisation product on other states (waiting states) in addition to blocking actions (in the execution states). Blocking is used to prevent the interleaving of activities. Indeed, since the activities are not instantaneous (i.e., they have duration), the composition operator must ensure that at a specific instant of time t, there is only one activity being executed (i.e., only one activity automaton is in an execution state). Synchronous is needed when the activity automaton is at waiting states in order to synchronize the reset of the day and week clocks (respectively, the variables x_d and x_w). A formal definition of the proposed composition operator is given below.

Definition 3.2.2 ***(Composition of timed automata)*** *Let* $A_1 = (S_1,\ s_0^1,\ \Sigma_1,\ X_1,\ Inv_1, T_1, F_1, W_1, E_1, St_1)$ *and* $A_2 = (S_2, s_0^2, \Sigma_2, X_2, Inv_2, T_2, F_2, W_2, E_2, St_2)$ *be two timed automata. The composition of* A_1 *and* A_2*, denoted* $A_1 \bigodot A_2$*, is the timed automaton* $(S_1 \times S_2,\ s_1^0 \times s_2^0,\ \Sigma_1 \cup \Sigma_2,\ X_1 \cup X_2,\ Inv,\ T,\ F_1 \times F_2\ ,\ W,\ E,\ St)$*, where* $Inv\ (S_1, S_2) = Inv\ (S_1) \wedge Inv\ (S_2)$ *and the set of transitions* T *is the union of the following sets:*

1. $\{((s_1, s_2), \varepsilon, \phi, \lambda, (s_1', s_2')) : (s_1, \varepsilon, \phi_1, \lambda_1, s_1') \in T_1$ *and* $(s_2, \varepsilon, \phi_2, \lambda_2, s_2') \in T_2$*,* s_1 *and* s_2 *are both waiting states and* $\lambda = \lambda_1 \cup \lambda_2$*,* $\phi = \phi_1 \wedge \phi_2\ \}$.

2. $\{((s_1, s_2), a, \phi, \lambda, (s_1', s_2'))$*:* $((s_1, a, \phi, \lambda, s_1') \in T_1,\ s_2 = s_2')$ *or* $((s_2,\ a,\ \phi,\ \lambda,\ s_2') \in T_2,\ s_1 = s_1')$*,* s_1 *and* s_2 *are both start states, or,* s_1 *is start state and* s_2 *is waiting state, or,* s_1 *is waiting and* s_2 *is start state* $\}$.

3. $\{((s_1, s_2), a, \phi, \lambda, (s_1', s_2'))$*:* $((s_1,\ a,\ \phi,\ \lambda,\ s_1') \in T_1,\ s_2 = s_2',$ s_2 *is a waiting/start state,* s_1 *is an execution state) or* $((s_2,\ a,\ \phi,\ \lambda,\ s_2') \in T_2, s_1 = s_1', s_1$ *is a waiting/start state,* s_2 *is an execution state)* $\}$.

The sets W, E and St are defined as follows:

- $W = \{(s_1, s_2) \in S_1 \times S_2$*:* $s_1 \in W_1$ *and* $s_2 \in W_2\ \}$.
- $St = \{(s_1, s_2) \in S_1 \times S_2$*:* $(s_1 \in St_1$ *and* $s_2 \in St_2)$ *or* $(s_1 \in St_1$ *and* $s_2 \in W_2)$ *or* $(s_1 \in W_1$ *and* $s_2 \in St_2)\ \}$.
- $E = \{(s_1, s_2) \in S_1 \times S_2$*:* $(s_1 \in W_1 \cup St_1$ *and* $s_2 \in E_2)$ *or* $(s_1 \in E_1$ *and* $s_2 \in W_2 \cup St_2)\}$.

Observe that the composition operator is closed w.r.t the definition 3.2.1 in the sense that it takes as an input two timed automata as defined in 3.2.1 and generates as an output an automaton compliant with this definition. Hence, it is possible to use the result of a composition as an input for another composition, thereby, enabling incremental composition.

Note that, when using the composition operator to construct activity automaton, the alphabet Σ_i, $i = 1, ..n$ of each temporal specification automaton (associated to activity automaton) has a same activity label (i.e. $\Sigma_1 = \Sigma_2 = \Sigma_3 = ...$).

Table 3.5 illustrates an example of the activity Toilet composed of two temporal specifications.

Activity	Days	Time ranges	Period	Duration
Toilet	Monday Thursday	Morning	01/01/14-12/31/14	30
	Sunday	Evening	01/01/14-12/31/14	

Table 3.5: Elementary temporal specifications

Given a set of elementary temporal specifications of a given activity (expressed in rows of Table 3.5), we first build the pattern automaton for each elementary temporal specification. Figures 3.8 and Figures 3.9 show the pattern automata A_{r1} and A_{r2} associated respectively with the first and the second elementary temporal specifications of Table 3.5. Timed automaton A_{r1} recognizes timed words corresponding to the execution of the activity Toilet on Monday and Thursday morning.

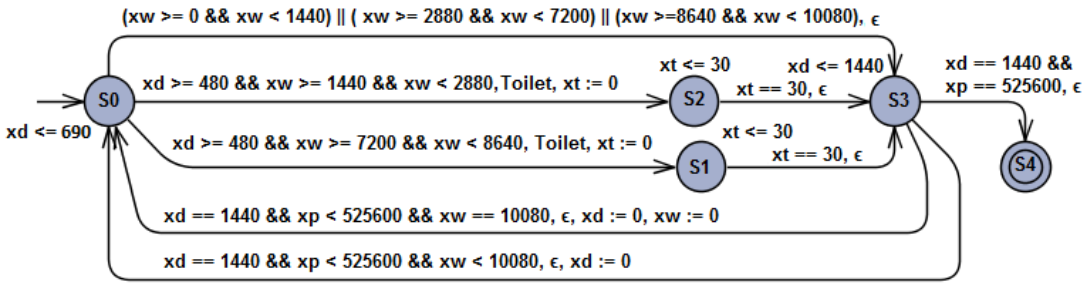

Figure 3.8: Timed automaton A_{r1}.

Timed automaton A_{r2} recognizes timed words corresponding to the execution of the activity Toilet on Sunday evening.

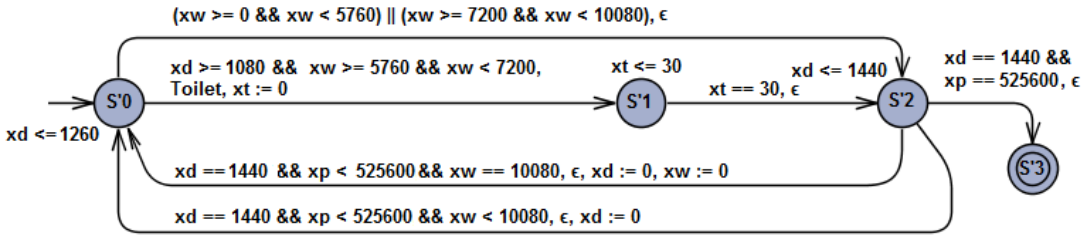

Figure 3.9: Timed automaton A_{r2}.

Having A_{r1} and A_{r2} automata, we construct the activity automaton using the defined composition operator. Figure 3.10 shows the composition result. The resulting automaton encompasses all the possible schedules of the activity Toilet specified in Table 3.5.

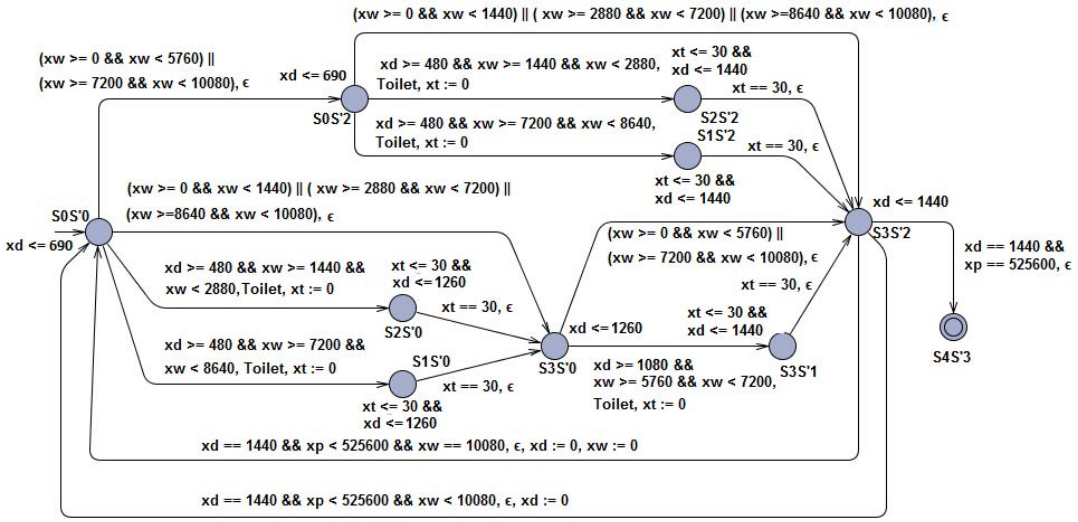

Figure 3.10: Activity (Toilet) timed automaton.

3.2.4 Care Plan Automata

Recall that the care plan is composed of a set of activities. Thus, in the same spirit as for the activity automaton, the care plan can also be described by means of timed automaton, but this time, by combining its associated activities automata.

Note that, when using composition operator to construct care plan automaton, Σ_i, $i = 1, ..n$ of each temporal activity automaton have a different activity label (i.e. $\Sigma_1 \neq \Sigma_2 \neq \Sigma_3 \neq ...$).

Table 3.6 illustrates an example of care plan composed of two activities: Toilet and Injection.

Activity	Days	Time ranges	Period	Duration
Toilet	Everyday	Morning	01/01/14-12/31/14	30
Injection	Everyday	09h00	01/01/14-12/31/14	20

Table 3.6: Example of home care plan

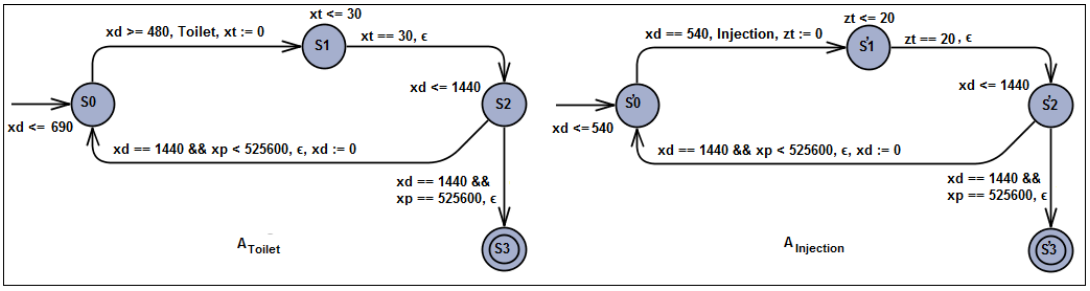

Figure 3.11: Toilet and Injection automata.

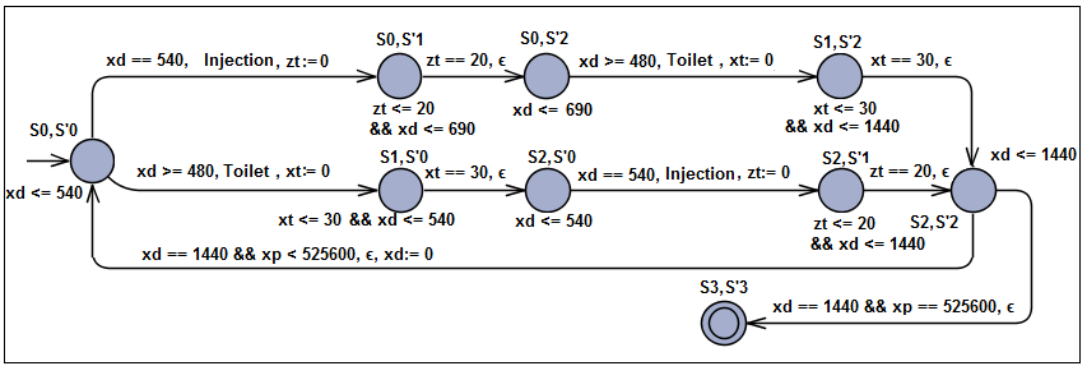

Figure 3.12: Home care plan timed automaton

Figure 3.12 shows the result of composition of the Toilet and Injection automata (A_{Toilet} and $A_{Injection}$) of Figure 3.11. The resulting automaton encompasses all the possible schedules of the activities Toilet and Injection. For example, a *timed word* of the form: (Toilet, 480).(Injection, 540).(Injection, 1980).(Toilet, 2001)...(Injection, 524700), which is accepted by the care plan automaton of Figure 3.12, corresponds to a legal plan within the specified period.

3.3 Formal analysis of home care plan using timed automata

With a formal model describing the behavior of home care plans at hand, it becomes now possible to handle automatic verification and monitoring of home care plans. Here we focus on realizability verification, monitoring and interventions generation. These verification tasks are important in the sense that a planning error or a bad monitoring can have dramatic consequences.

We discuss below how to use the proposed framework to verify and monitor the home care plans using UPPAAL Model Checker.

3.3.1 Realizability of home care plans

It is important to check the realizability of a home care plan, i.e., to check whether or not the activities included in the home plan can be effectively scheduled and performed according to the constraints specified in the plan. In other words, a home care plan is realizable when each activity can be performed without interruption in the imposed time range in any specified period. Checking realizability of a home care plan can be reduced to the emptiness problem of the corresponding timed automaton. In fact, the emptiness problem for a timed automaton A consists in verifying whether or not $L(A)$ is empty. A standard way to achieve such a test is to check reachability of the final state from the initial state.

Consider again the timed automaton shown in Figure 3.12. We check in the following if the final state (s_3, s'_3) is reachable from the initial state (s_0, s'_0). The emptiness checking is performed using the following query[1]: E <>(Process.FinalStateName).

Figure 3.13 shows the emptiness checking of the home care plan timed automaton of Figure 3.12 using UPPAAL. The status box shows the query result (satisfied or not). In our example, the query is satisfied, which means that the home care plan is realizable.

3.3.2 Monitoring of home care plans

Note that most of the activities of a home care plan are manual, i.e., performed manually by professionals. In current state of affairs, the activities that have been performed are

[1]Note that UPPAAL model checker uses a subset of TCTL (Timed CTL) as query language to express properties.

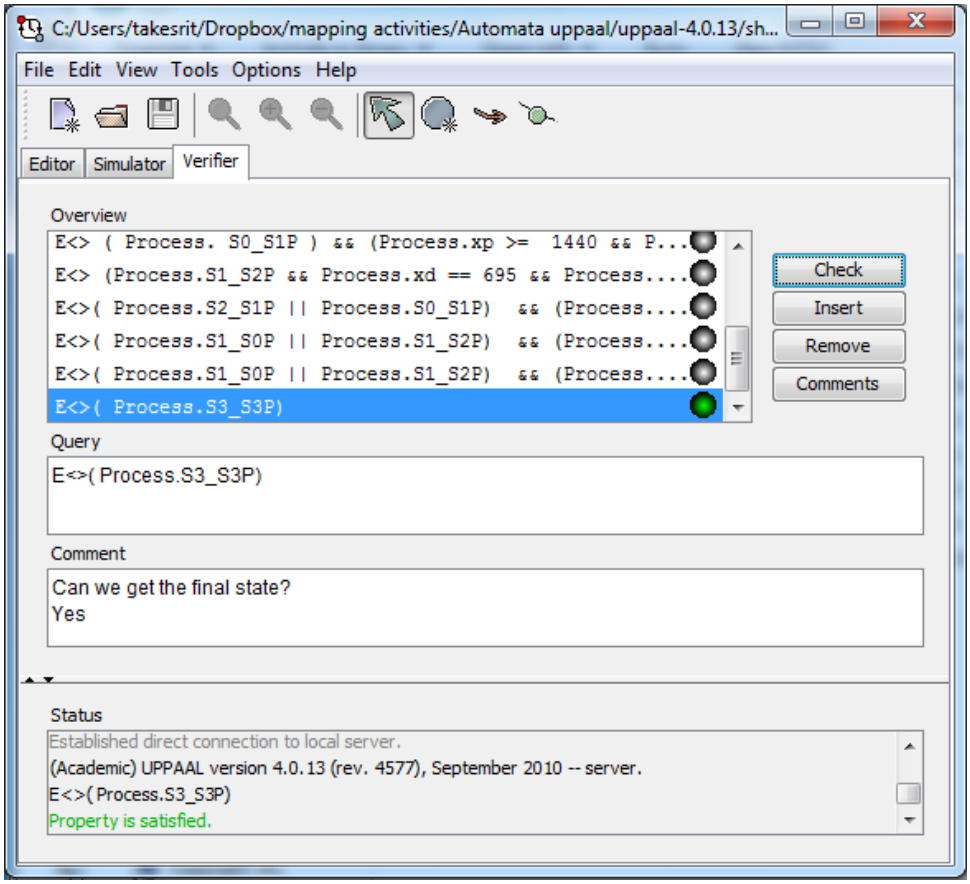

Figure 3.13: Emptiness checking timed automata with UPPAAL.

often recorded manually on paper. Our goal is to enable electronic recording of executed activities in order to keep track of the execution of home care plans. Such information can then be used to monitor home care plans. For example, compliance of execution traces w.r.t. a home care plan may be checked by reducing this problem to the membership problem in the timed automata framework. Also, the monitoring system may be used to detect executions that deviate from the specification. For example, if in the scheduled home care plan the activity Injection is provided at 09h00 and takes 30 minutes, and after execution we find out that it was in fact done at 10h00, it is important to be able to detect this type of deviation. Below an example of query:

E<> (Process.S0S'1) && (Process.xp >= 1440 && Process.xp < 2880 && Process.xd >= 660 && Process.xd <= 720).

This query checks that the activity Injection which was done on January 2 between 11h00 and 12Hh00 is well complying with a scheduled home care plan timed automaton

of Figure 3.12. In other words, is there an execution which satisfies this activity ? The answer is property not satisfied since Injection was scheduled starting from 9h00. More generally, a monitoring system can be enhanced with rules that enable to trigger alerts when particular deviations are detected.

3.3.3 Grouping activities into interventions

Grouping together activities that can be performed by a same type of actor (nurse, nurse auxiliary, etc.) and which occur in the same time range is called Intervention. Interventions may be specified manually by the coordinator or computed automatically from the specifications of activities and then proposed to the coordinator for validation. The concept of intervention is really important in the sense that it allows to reduce the waiting time between activities in order to avoid multiple movings at the patient's home. The analysis of activities to specify interventions is a complex task since it requires to ensure compatibility of time ranges by taking into account the duration of each activity (multiple configurations are possible). It is also necessary to ensure that the grouped activities can be made by a same type of actor. The composition operator can be modified in order to incorporate the interventions in the computed care plan automaton (the obtained automaton is called interventions automaton). This is achieved by modifying the activity automaton in order to take into account the Intervention state. More precisely, we modify Definition 3.2.1 to consider an automaton as a tuple A= (S, s_0, Σ, X, Inv, T, F, W, E, St, Int) where Int denotes the set of intervention states. In addition, a specific time parameter, denoted Tmin, is added to control the *idle time* between the activities within the same intervention. Tmin can be defined as " the minimum waiting time between two activities before starting a new intervention". In fact, the value of Tmin can be used as a parameter that can be defined by the coordinator and given as input to the composition operator to compute the interventions automaton (i.e., care plan annotated with interventions).

Figure 3.14 shows how the automaton of Figure 3.11 is modified to illustrate the Toilet automaton with interventions which will be used to generate interventions automaton. Indeed, let A_{Int} be the automaton corresponding to Toilet automaton with interventions. $A_{Int} = (S, s_0, \Sigma, X, Inv, T, F, W, E, St, Int)$ is defined as follows:

- $S = \{s_0, s_1, s_2, s_3, s_4\}$ with s_0 the initial state;
- $W = \{s_3\}$;

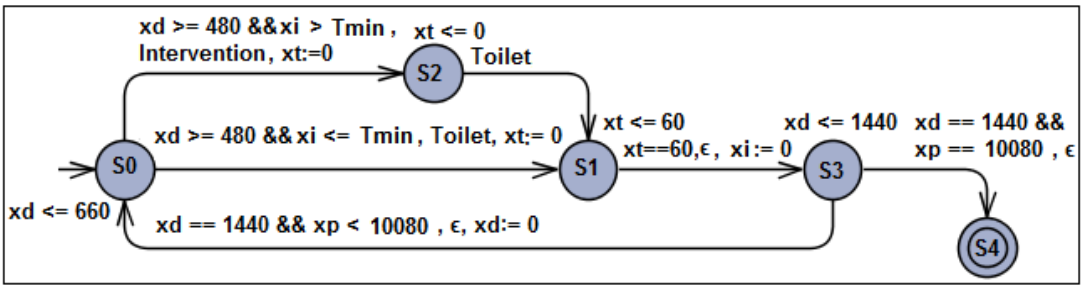

Figure 3.14: Toilet automaton with interventions.

- $E = \{s_1\}$;
- $St = \{s_0\}$;
- $Int = \{s_2\}$, is an intervention state which is instantaneous;
- $F = \{s_4\}$;
- $\Sigma = \{\text{Toilet, Intervention}\} \cup \{\varepsilon\}$;
- $X = \{x_d, x_t, x_p, x_i\}$ is the set of clocks, where x_i is used to control the minimum idle time fixed by the coordinator;
- T is the set of transitions.

In order to build the interventions automaton, it is necessary to add an additional rule in the definition of the composition operator "$\odot$" to take into account the intervention state. The new rule is defined as follows:

- $\{((s_1, s_2), a, \phi, \lambda, (s'_1, s'_2))$: $((s_1, \text{a}, \phi, \lambda, s'_1) \in T_1, s_2 = s'_2, s_2 \in W \cup E \cup St, s_1 \in Int)$ or $((s_2, \text{a}, \phi, \lambda, s'_2) \in T_2, s_1 = s'_1, s_1 \in W \cup E \cup St, s_2 \in Int)\}$.

And the set Int is defined as follows:

- Int$=\{(s_1, s_2) \in S_1 \times S_2$: $(s_1 \notin Int$ and $s_2 \in Int)$ or $(s_1 \in Int$ and $s_2 \notin Int)\}$.

Note that the new composition operator remains closed w.r.t. the modified Definition 3.2.2. Let us consider the example of Table 3.7 which illustrates an example of home care plan composed of two activities Toilet and Dress. The Toilet automaton with interventions is depicted at Figure 3.14.

Activity	Days	Time ranges	Period	Duration
Toilet	Everyday	08h00-12h00	01/11/15-01/17/15	60
Dress	Everyday	10h00-12h00	01/11/15-01/17/15	30

Table 3.7: Example of home care plan

Figure 3.15 shows the resulting care plan automaton with interventions using the added rule to the composition operator. The interventions automaton encompasses all the possible interventions that can be performed at patient's home. A timed word w= (intervention,540).(Toilet,540).(Dress,600).(Intervention, 1980).(Toilet,1980).(Dress, 2040).(Intervention, 3420).(Toilet, 3420).(Dress, 3480).(Intervention, 4860).(Toilet, 4860).(Dress, 4920).(Intervention, 63000).(Toilet, 63000).(Dress, 6360).(Intervention, 7740).(Toilet, 7740).(Dress, 7800).(Intervention, 9180).(Toilet, 9180).(Dress, 9240), where intervention corresponds to the start of an intervention, is accepted by the interventions automaton, and illustrates possible interventions within the specified period.

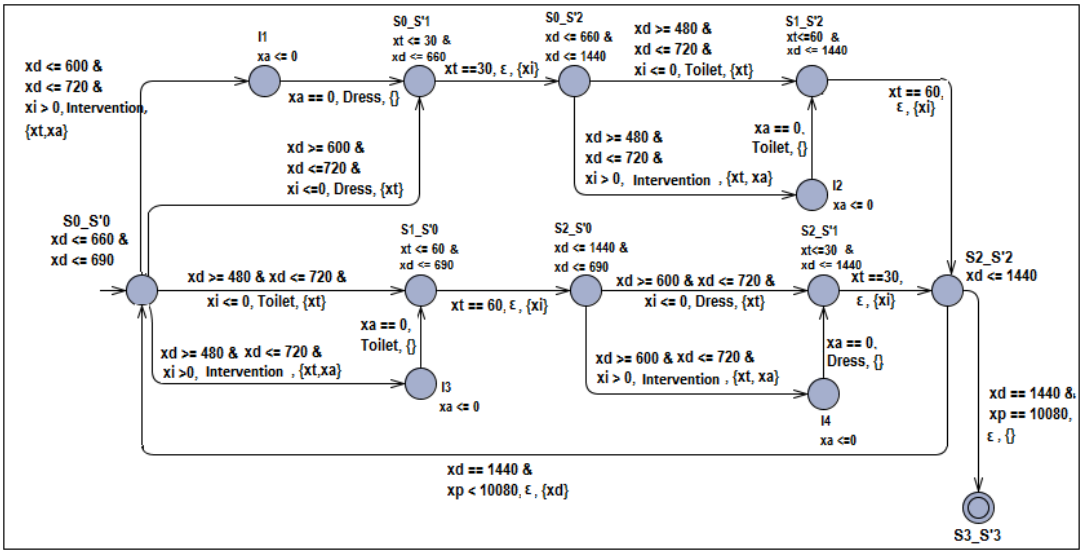

Figure 3.15: Example of interventions automaton.

3.4 Conclusion of Chapter 3

We described in this chapter our approach to generate formal specifications of home care plans, expressed as timed automata, from a set of high level and user oriented

abstractions. The resulting care plan encompasses all the possible legal schedules of activities for a given patient. We describe then how verification and monitoring of the resulting home care plan can be handled using UPPAAL Model Checker. The next chapter will deal with the implementation of the proposed approach.

Chapter 4

Prototype and experimentations

Contents

In order to test and validate our work, we have implemented the approach that we proposed in this thesis. We implemented our prototype as a part of the project Plas'O'Soins, which aims to provide a software platform for modeling, planning and monitoring home care plans. Pointing out that our work is in parallel with Plas'O'Soins project and we address identified problems with another vision. We have the same starting point, namely, temporal specifications. But we bring other performance such as realizability check, monitoring and interventions generation using UPPAAL Model Checker. As part of this thesis, we developed four main components: (i) **The graphical user interface** that allows to assist the coordinator for editing home care plans, (ii) **The temporality automata generator** that enables to generate temporality automata of each activity from extracting information about the temporal specifications in the database, (iii) **The care plan timed automata generator** which consists on constructing the home care plan timed automata using the result of (ii), this construction is an implementation of our composition algorithm, and (iv) **The interventions generator** which consists in grouping together activities that can be performed in a same time range and by a same type of actor. These four components have been implemented using JavaTM platform version 8, Netbeans[1] framework and Vaadin[2] framework. Here, we describe the technical details of the prototype implementation and present some tests on real home care plans.

4.1 Overview of the prototype

Before going into the details of the prototype architecture and implementation, it is necessary to have the intuition and the general view of the prototype functioning.

Figure 4.1 illustrates through a BPMN [3] process, a structured overview of the general functioning of our prototype, since the elaboration of the home care plan by the coordinator, till its verification phase.

Upon admission of a new patient, the coordinator creates for him/her a personalized home care plan, taking into account his/her medical needs as well as his/her social and physical environments. This activity is represented by the collapsed subprocess Create Home care plan (see Figure 4.1). An expended version of the decomposable activity Create Home care plan is shown in Figure 4.2.This latter presents the set of elementary human

[1] https://netbeans.org
[2] https://vaadin.com/home
[3] http://www.omg.org/spec/BPMN/2.0/PDF/

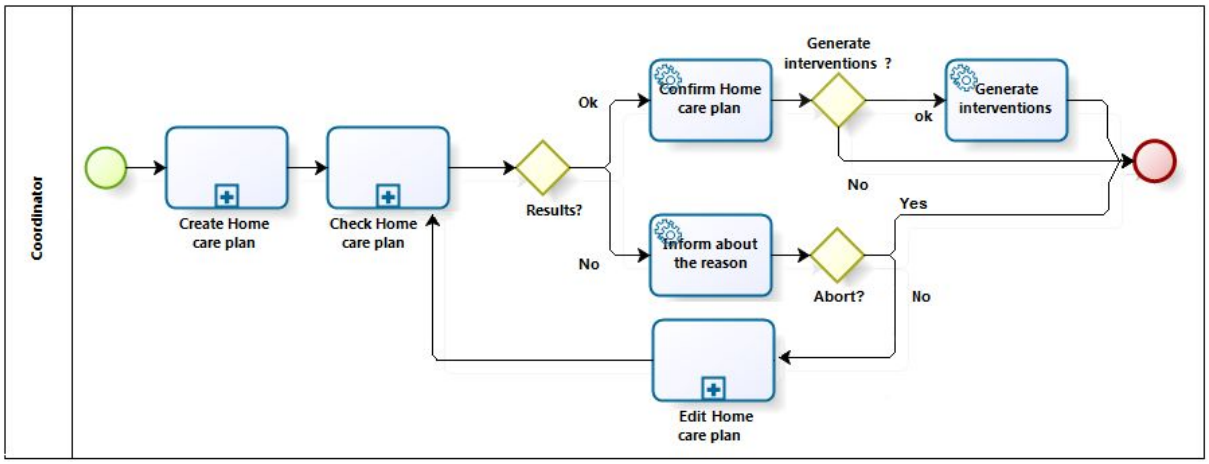

Figure 4.1: BPMN process of prototype functioning

activities, such as:

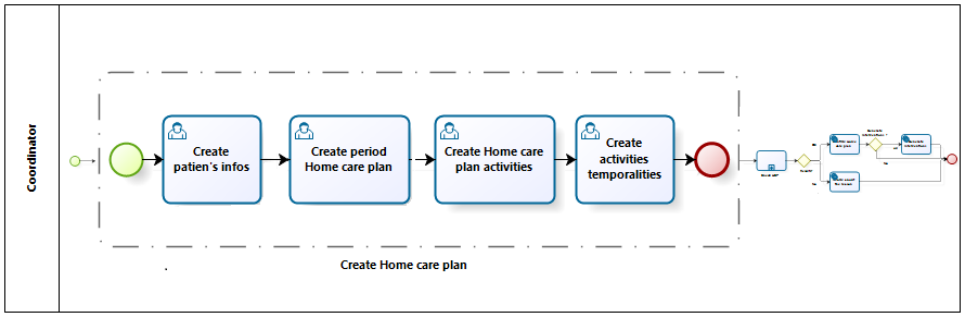

Figure 4.2: Expended view of the subprocess Create Home care plan

- *Create patient's infos*: create general information about the patient (Last name, first name, address, weight, etc.);
- *Create period Home care plan*: create the start date and end date of the home care for the created patient;
- *Create Home care plan activities*: create the required home care plan activities for the given patient;
- *Create activities temporalities*: specify the repetitiveness of the care plan activities.

Once the home care plan created, the coordinator can trigger a set of analysis which will be executed automatically by the system. The coordinator must check the validity

of the home care plan. The collapsed sub process Check Home care plan in Figure 4.1 encompasses a set of automatic activities which contribute to the realizability verification. Figure 4.3 shows a corresponding expended version which comprises of a set of automatic activities, which are:

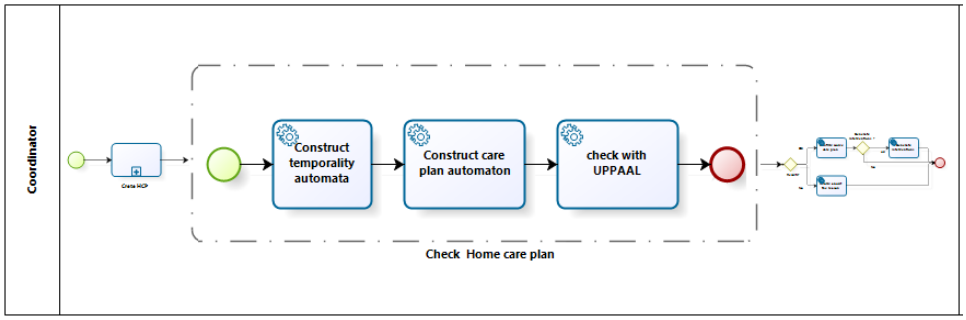

Figure 4.3: Expended view of the subprocess Check Home care plan

- *Construct temporality automata*: generate temporalities timed automata using activities frequencies;

- *Construct Care plan automaton*: compute the composition of all temporality automata in order to construct care plan automaton;

- *Check with UPPAAL*: check the realizability using UPPAAL Model Checker.

After checking the home care plan, if it is realizable, the coordinator can, if he wishes, initiate the interventions generation using UPPAAL Model Checker. Figure 4.1 illustrates the corresponding automatic activity Generate interventions. If the care plan is not realizable, an error message is displayed on the GUI to inform the coordinator about the reason. The coordinator must then re-edit the home care plan.

Another analysis which can also be made by the coordinator is monitoring. After the execution of the home care plan by the actors, the coordinator can check for a given period if the executed activities correspond (in terms of time ranges) to the scheduled activities. The monitoring is a very important step for the proper conduct of patient care.

4.2 Architecture of the prototype

Our architecture, depicted in Figure 4.4, comprises several modules, which are used for modeling and analyzing home care plans.

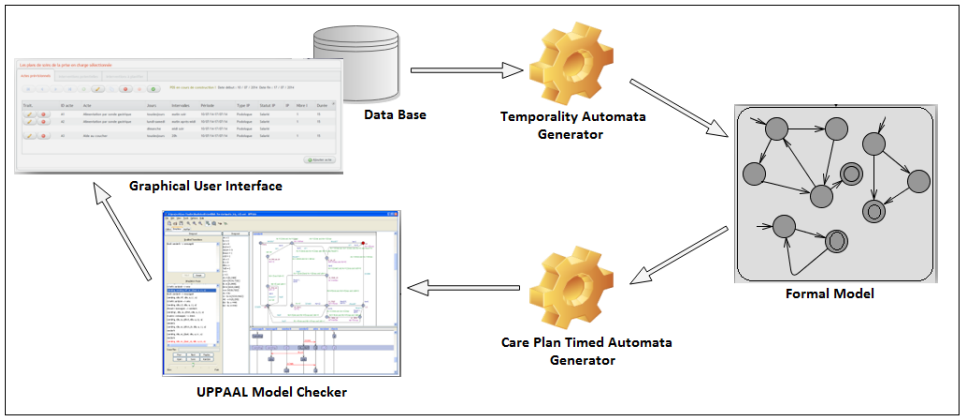

Figure 4.4: Architecture of the prototype

In what follows, we present in details the different components of our architecture.

4.2.1 GUI for editing home care plans

Developing a graphical user interface (GUI) for editing home care plans is the starting point in the process of modeling and analyzing home care plans. This editing represents a major challenge for the successful management of care and is essential to propose ways that enable greater efficiency for end-users.

After observing the practices and collecting opinions of end-users, we have developed a GUI that has been performed using Vaadin framework. It provides a set of windows and dialogue boxes offering the possibility to better attend the coordinator in the elaboration of the home care plan.

After editing patient's information and specifying the period of the home care plan, the coordinator performs two necessary steps in order to elaborate the home care plan.

4.2.1.1 Editing home care plan activities

In order to edit the home care plan, the coordinator opens a window as shown in Figure 4.5 which allows to enter the activities with their repetitiveness for the given patient. The order of the entries is irrelevant. The labels of the activities are chosen from a list

of allowed labels. Actor type for each activity and duration are initialized by the system by default (we associate for each activity a nominal duration and an actor type), but the coordinator has the possibility to change their values. He may even impose a name of actor (when the coordinator wants to assign an actor with specific skills).

We see for example in Figure 4.5 four different planned activities. Temporal specifications are edited into four main fields: Days, Ranges, Period and Duration. The values of these fields can be entered through a dialogue box (4.2.1.2) to allow an assisted entry. The value of the period is by default the same as that of the home care plan. The main actor type is associated with each activity (this is the type of actor who is responsible for the activity). The activity ID in the first column (A1, A2, ...) is attributed by the system. It is specific to each entry.

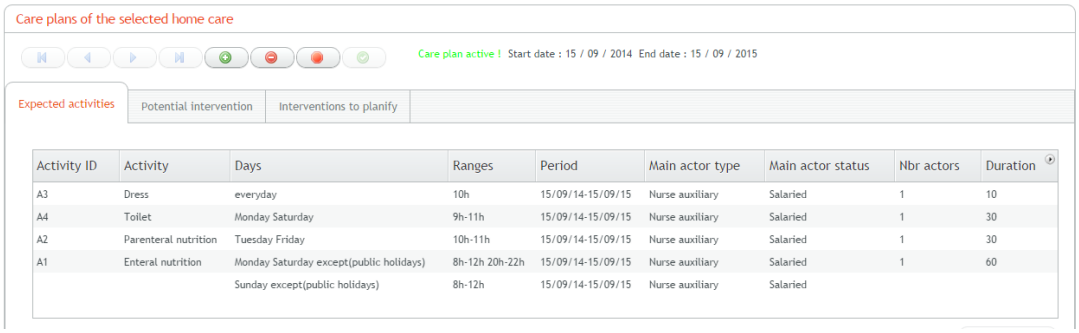

Figure 4.5: Editing home care plan activities

4.2.1.2 Editing activities temporalities

Dialogue boxes have been designed to guide the entry of temporal specifications. Figure 4.6 shows the editing of a temporality for an activity.

Indeed, expressing activities temporalities is essential to specify their repetitiveness. We can see for example in Figure 4.6 several fields to be completed: Date (which refers to a calendar), Time ranges (which refer to times of the day, time slots or fixed times), the Period and the Principal actor type. However, activities repetitiveness is not always fully regular but there may be exceptions. The button "except public holidays" allows to insert exceptions.

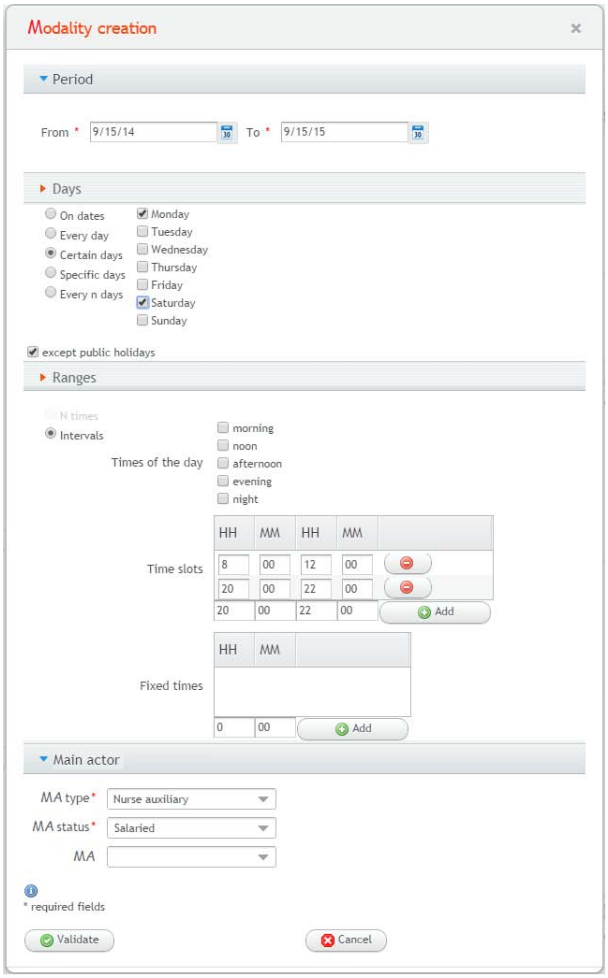

Figure 4.6: Dialogue box for editing activities temporalities

4.2.2 Storage in the database

The various data handled on the GUI are spread over the database. We used in our case Mysql[4] database version 5.6. This later memorizes all factual data handled to manage the home care of patients as plain texts. The connection between the GUI and the database is done through Java Persistence API (JPA)[5].

[4]http://www.mysql.fr
[5]https://eclipse.org/eclipselink/jpa.php

4.2.3 Temporality automata generator

Temporality automata generator enables to generate temporalities timed automata according to several input temporalities corresponding to each activity in the database for a given patient. Indeed, temporality automata generator consists in three main components as shown in Figure 4.7.

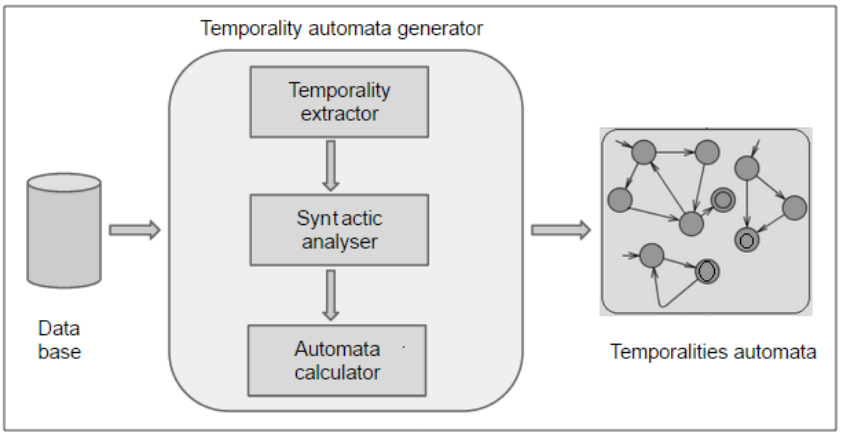

Figure 4.7: Components of temporality automata generator

- **Temporality extractor**: We have implemented a parser which extracts, for each patient, the corresponding care plan temporal specifications as plain strings. Temporality extractor browses several tables in the database to extract temporal specifications (Days, Time ranges, Duration and Period) of each activity corresponding to the ongoing home care plan of the patient.
- **Syntactic analyser**: It allows to structure the different temporality fields (Days, Time ranges, Period and Duration) and activity name, in class objects. Figure 4.8 shows the UML class diagram of the resulting class objects. It highlights the necessary concepts to express Elementary specification, such as:
  - <Activity>: this class corresponds to the activity of the home care plan, characterized by its duration and name;
  - <Day>: this class defines the pattern type that will be used for each elementary specification. Days type can take only one type among the three defined in the diagram (AbsoluteDate, SpecificDay, or Everyday). Exception class allows to express exceptions as dates;

- <Range>: this class represents the start and the end time of the elementary specification expressed in hours and minutes;
- <Period>: this class represents the start and the end date for the validity of each specification;
- <Actor>: this class represents the actor involved in a specification and his/her type. The coordinator may in some cases precise the actor name.

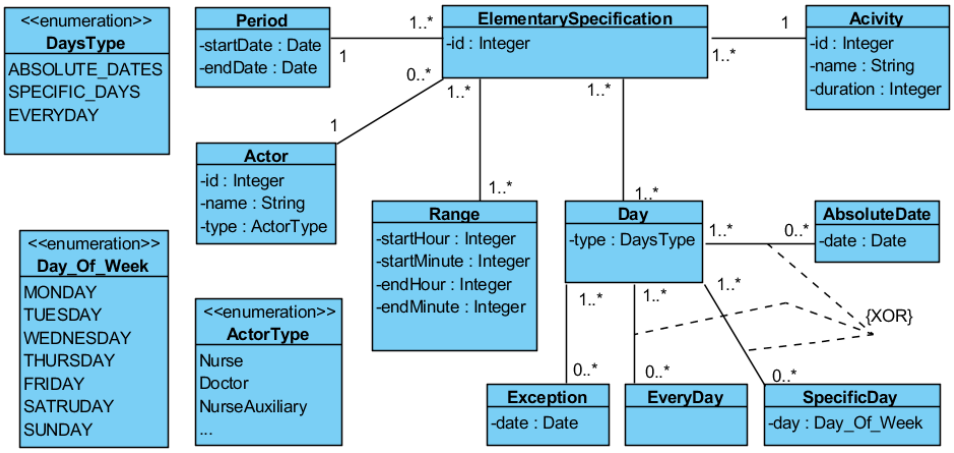

Figure 4.8: UML class diagram for temporal specifications

- **Automata calculator**: It is in charge to build temporality automata according to the result of the syntactic analyser, especially the type of Days (Everyday, Absolute dates or Relative days). Figure 4.9 shows the UML class diagram which includes all the class objects necessary for the construction of the temporality automata.
  - <Automaton>: this class is used to identify the temporality automaton;
  - <State>: this class describes a state of the automaton associated with its name, id and the type (start state, waiting state or execution state, etc.);
  - <Invariant>: this class defines the clock conditions in each state of the automaton;
  - <Transition>: this class describes a transition of the automaton, by specifying the source and the target state. Transition class is linked to three other classes: Update class, Label class and Guard class. They allow to describe the details of temporality automaton transitions;
  - <Clock>: this class defines a clock of the automaton with its name and type.

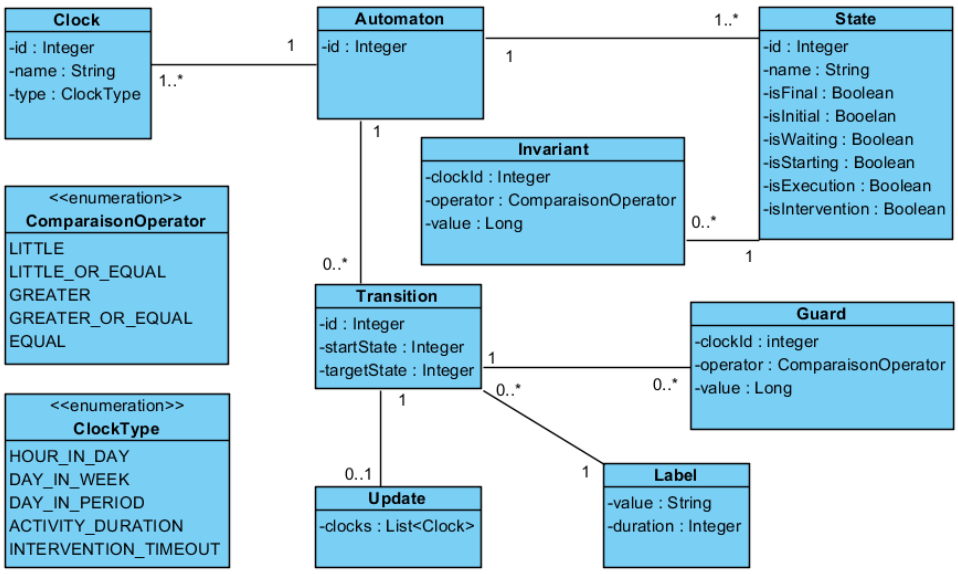

Figure 4.9: UML class diagram for temporality automata

4.2.4 Care plan automata generator

Care plan automata generator provides an implementation of our composition operator 3.2.2. Indeed, the care plan automata generator computes the composition of all temporality automata built by the temporality automata generator corresponding to the ongoing care plan of the patient. Once the care plan automata is built, appropriate XML files (sample in Figure 4.10) are generated in order to be read by UPPAAL Model Checker. Each element of this XML file corresponds to an information on the constructed care plan automaton. The XML file has as a root tag <nta> which regroups a set of tags, such as:

- <Declaration>: this tag describes the global declarations of the automaton. In our work, it is used to declare the clocks defined in the automaton.
- <name>: it is used to identify the automaton name. In Figure 4.10, the automaton name is A;
- <Location>: this tag allows to define each state of the automaton by a unique identifier, a name and an invariant;
- <Transition>: this tag is used to define the source and the target state, the label, the guard and the clock updates of a transition.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN'
'http://www.it.uu.se/research/group/darts/uppaal/flat-1_1.dtd'>
<nta>
<template>
    <name>A</name>
<declaration>
    clock xd0; clock xp0; clock xt0; clock xd1; clock xp1; clock xt1;
    int nbDress = 0;
    void Dress()
    {
        nbDress++;
    }
    int nbToilet = 0;
    void Toilet()
    {
        nbToilet++;
    }
</declaration>
    <location id="id0">
        <name>S0_S0</name>
        <label kind="invariant">xd0  &lt;=  660 and xd1  &lt;=  675</label>
    </location>
    <location id="id1">
    ...
    <transition>
        <source ref="id0"/>
        <target ref="id1"/>
        <label kind="guard">xd1  &gt;=  600 and xd1  &lt;=  720</label>
        <label kind="assignment">Dress(), xt1 = 0</label>
    </transition>
    ...
</template>
<system>// Place template instantiations here.
    system A;
</system>
</nta>
```

Figure 4.10: Part of an XML file generated by care plan automata generator

4.2.5 Connection with UPPAAL

Once the construction of the home care plan automaton is completed, the analyzing step of the care plan automata and the interventions generation is achieved using UPPAAL Model Checker. UPPAAL uses, as input, the XML file generated by the care plan automata generator, and provides as output the answer about analyzed properties (realizability or monitoring) and contributes for the interventions generation.

For example, Figure 4.11 shows how UPPAAL Model Checker displays the care plan timed automata using the XML file of Figure 4.10. Figure 4.12 shows how UPPAAL Model Checker uses TCTL queries.

We will see in more details in section 4.3 a real example of the care plan verification and how the answers are displayed to the coordinator through the graphical user interface.

4.2.6 Interventions generator

Interventions generator enables to provide to the coordinator a complete list of interventions to be performed by actors, day after day, during all the period of the home

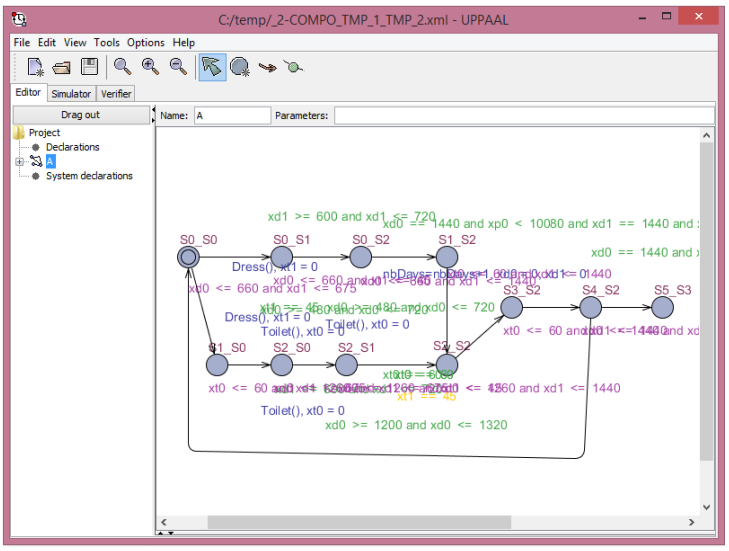

Figure 4.11: UPPAAL uses XML file of Figure 4.10 to display the corresponding care plan timed automaton

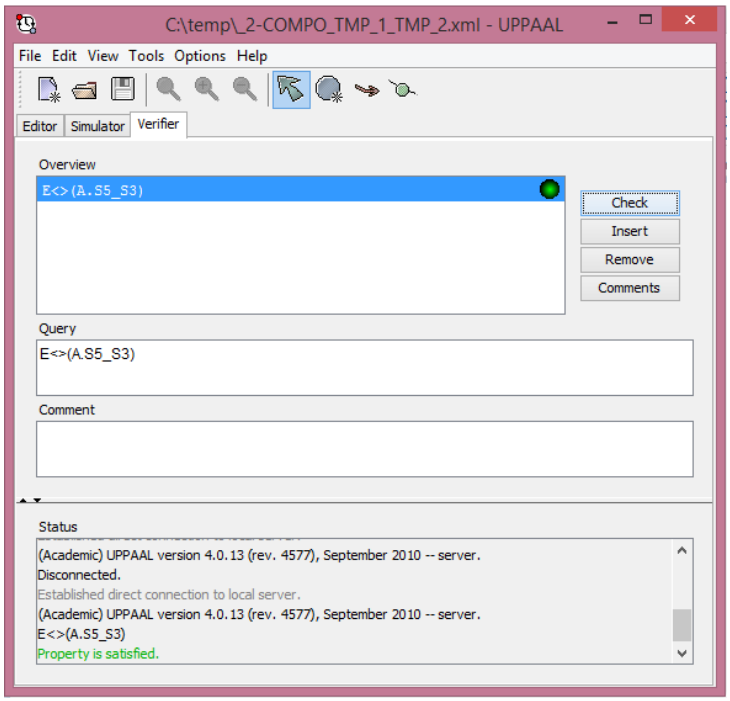

Figure 4.12: UPPAAL uses TCTL queries to verify the care plan automaton

care plan. Indeed, once the home care plan is validated, the coordinator can trigger the interventions generation. In order to generate interventions, the system follows the steps

below:

1. Construct the pattern automata by adding intervention transitions and intervention states.

2. Construct the activities timed automata with interventions using the modified composition operator on the corresponding pattern automata. An example of an activity automaton with interventions is depicted in Figure 3.14.

3. Construct the home care plan timed automaton with interventions using the modified composition operator on the constructed activities automata. An example of care plan timed automaton with interventions is depicted in Figure 3.15.

4. Before building the interventions list to be provided to the coordinator, different requirements must be taken into account, which are:

   - The *idle time* between the activities of the home care plan;
   - The non-interleaving interventions;
   - The selection of actor type who intervenes in each intervention;
   - The time range of each intervention.
   - Different optimisation criteria such as minimize the global waiting time of interventions. Minimize the total number of interventions and different costs problems.

   Execute the home care plan automaton with interventions using UPPAAL. This enables to generate all possible schedules of interventions to perform at patient's home. Timed words of such automaton satisfy the first two requirements. Time range of each intervention and different optimisation problems have not been considered in our current prototype. Implementation of the optimisation problems require a specific investigation.

5. Once the interventions are generated, the most qualified (having the highest skills) actor type is designed to execute the corresponding activities.

4.2.7 Monitoring

Another important step supported by our prototype is the monitoring. Once the care plan validated and the activities performed at patient's home, the coordinator can check the consistency between scheduled activities and those executed. This verification is done with respect to certain criteria such as: Has the activity X been given? Has the activity time range been well respected? The developed prototype can answer such questions using UPPAAL Model Checker and following the two steps below:

1. Once the care plan executed, the system retrieves the log file. The log file contains the execution trace of the care plan by the actor (at patient's home). It describes each executed activity with its time range (start time and end time).

2. Using TCTL requests, the system queries the care plan timed automaton (elaborated home care plan) to report anomalies. As mentioned before, this procedure involves the membership question, which consists on taking the care plan timed automaton, and some information about the executed care plan from the log file, and determine if there is an execution of the care plan automaton which satisfies the tested information.

4.3 Test and experimentation

We now show how the prototype can be used to facilitate the analysis and interventions generation of home care plans. In fact, our prototype will assist the coordinator in checking the care plan by validating it, if there is non error, or by identifying the problem if it exists. It will also assist the coordinator for generating interventions which are an important step for the planning.

To better understand and see concretely the prototype functioning, we propose in what follows treating a real example of care plan of an anonymous patient (for confidentiality reasons). We will test different analysis and verification steps (realizability, interventions generation and monitoring) that we have identified previously.

4.3.1 Realizability test

Testing realizability of the home care plan consists in testing whether or not each activity can be performed without interruption in the determined time ranges throughout the

specified period of the home care plan.

In order to assess the efficiency of our prototype, we apply two case studies illustrating different situations, for which we will check realizability properties.

Case study 1

We demonstrate the usefulness of the application by illustrating prototype running results on an anonymous patient case. Figure 4.13 shows his/her care plan. We see in this example that, for this patient, the care plan is composed of three activities: Toilet, Dress and Injection. Each activity is characterized by temporal assertions. For example, the activity Toilet is described by a temporal specification which contains an exception on public holidays in Days field. The Period is the same for all the activities and the Durations varies from 20 to 60 minutes.

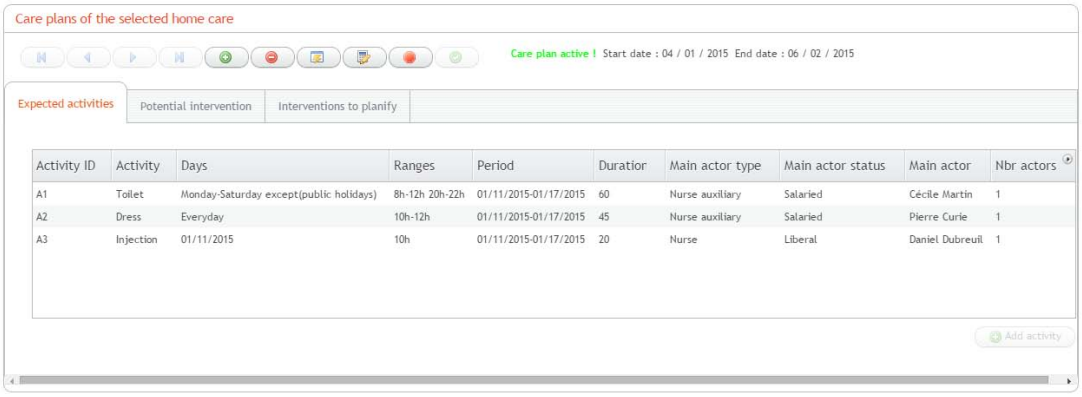

Figure 4.13: Example of a home care plan: Use case 1

Once the care plan created, the coordinator clicks on the saving button to record the care plan elements in the database. The coordinator then checks the care plan by clicking on the corresponding button to check the realizability of the care plan.

The concerning modules of our prototype construct the care plan automaton which is then verified using UPPAAL Model Checker. We use the TCTL query: E<> (Process.FinalStateName). This request means: is the care plan realizable?

Figure 4.14 shows the UPPAAL answer through the GUI. The care plan of the patient is realizable in all circumstances. The coordinator can then validate the care plan and trigger interventions generation if needed (see subsection 4.3.2).

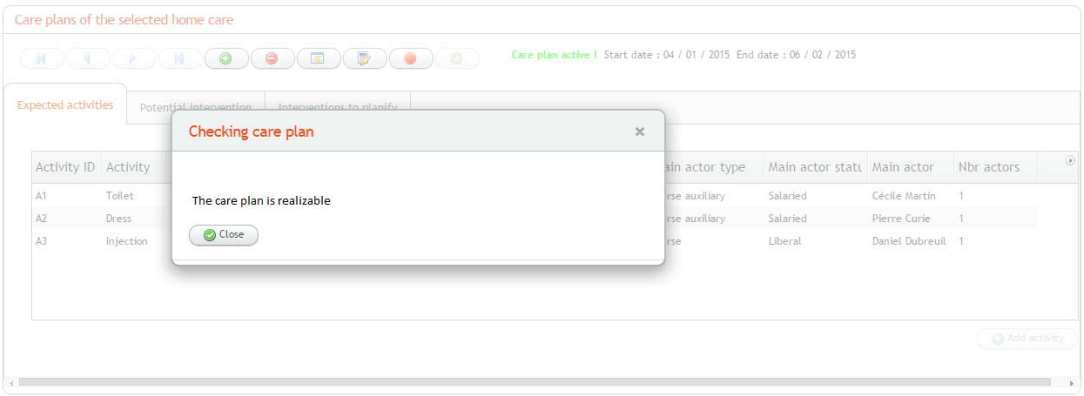

Figure 4.14: Dialog box showing the realizability of a care plan

Case study 2

We illustrate here a second example of a home care plan of another patient in Figure 4.15. The care plan comprises three activities: Toilet, Dress and Parenteral nutrition. Each of these activities is specified by one temporality row. We find the three types of patterns, Everyday, Absolute dates and Relative days. The Period is the same for all activities and the Duration varies from 20 to 60 minutes.

In the same way as in the first case study, the coordinator clicks on the saving button to record the care plan after creating, and then clicks on the corresponding button to check the realizability of the care plan.

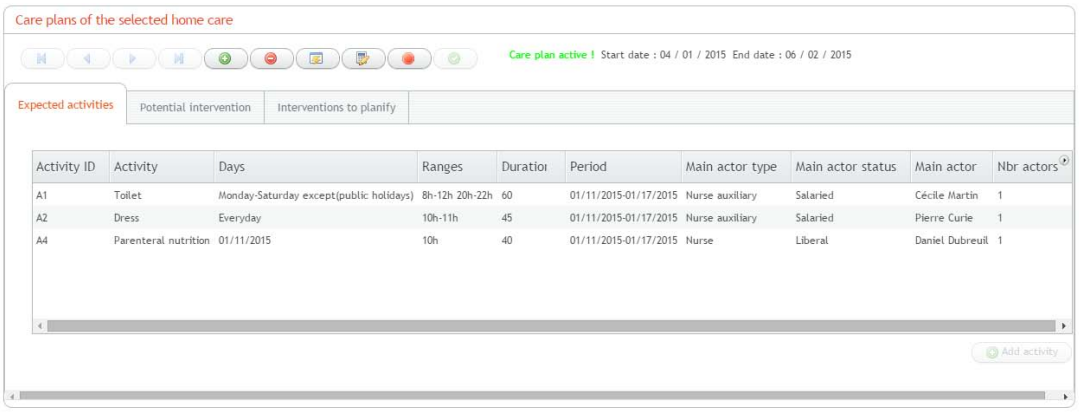

Figure 4.15: Example of a home care plan: Use case 2

Figure 4.16 shows the UPPAAL answer through the GUI. The care plan is not realizable and the problem is detected on Sunday, January 11th, 2015 for the activity Parenteral nutrition.

Indeed, we observe that the activity Parenteral nutrition is scheduled on 01/11/2015 at 10H00 and takes 40 minutes. If it starts execution and ends after 40 minutes, the following activity Dress which is also scheduled from 10H00 to 11H00 on 01/11/2015 and takes 45 minutes will not be performed and finished before 11H00. For this reason, a deadlock in the care plan automaton execution is detected. The coordinator must then re-edit the care plan.

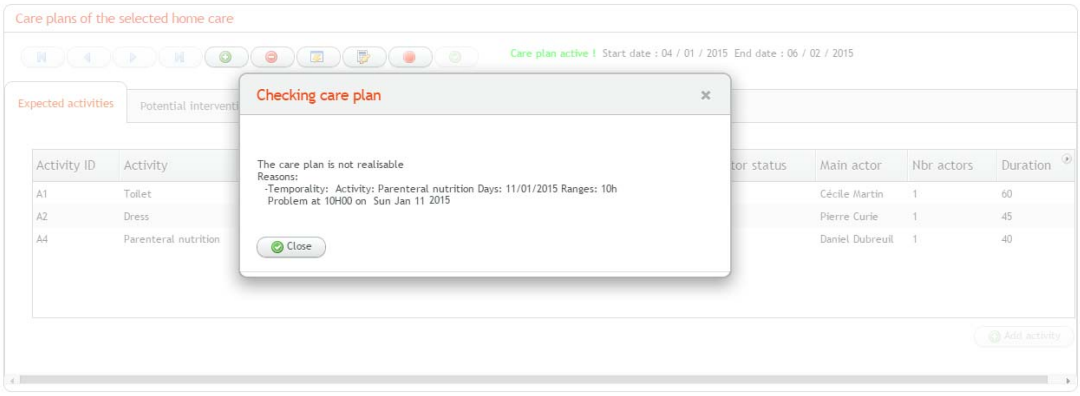

Figure 4.16: Dialog box showing the non realizability of a care plan

4.3.2 Interventions generation test

We have underlined the necessity of grouping compatible activities. A general processing for grouping activities on the basis of various criteria and constraints would be very useful for an efficient planning of these activities.

We take back the example of Figure 4.3.1. After having tested the realizability, the coordinator triggers interventions generation by clicking on the corresponding button. The system uses the modified composition operator (considering the interventions states) to generate an exhaustive list of interventions for each day of the period of the care plan.

Figure 4.17 shows the list of interventions through the GUI for the coordinator.

4.3.3 Monitoring test

We defined monitoring as a way to detect executions that deviate from the specifications. We illustrate in this part the same example as the one we saw in 4.3.1 for which we test the monitoring.

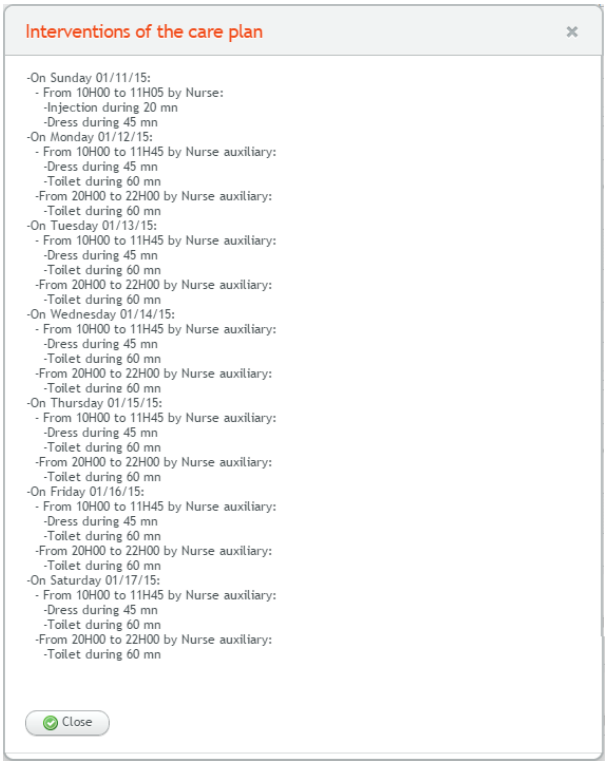

Figure 4.17: List of interventions

Once the care plan executed at patient's home, a log file is retrieved by our application. This log file contains information concerning executed activities at patient's home associated with their date, start and end hour. Figure 4.18 shows a sample of a log file corresponding to the execution of the home care plan illustrated in 4.3.1.

```
1  2015-01-11 20:08:37 [Activity] Injection [Date] 11/01/2015 [StartHour] 10h00 [EndHour] 10h20
2  2015-01-11 20:10:37 [Activity] Dress [Date] 11/01/2015 [StartHour] 10h20 [EndHour] 11h05
3  2015-01-12 22:00:30 [Activity] Dress [Date] 12/01/2015 [StartHour] 10h00 [EndHour] 10h45
4  2015-01-12 22:09:03 [Activity] Toilet [Date] 12/01/2015 [StartHour] 10h45 [EndHour] 11h45
5  2015-01-12 22:12:37 [Activity] Toilet [Date] 12/01/2015 [StartHour] 20h30 [EndHour] 21h30
6  2015-01-13 22:00:45 [Activity] Dress [Date] 13/01/2015 [StartHour] 10h00 [EndHour] 10h45
7  2015-01-13 22:01:03 [Activity] Toilet [Date] 13/01/2015 [StartHour] 10h45 [EndHour] 11h45
8  2015-01-13 22:05:30 [Activity] Toilet [Date] 13/01/2015 [StartHour] 21h20 [EndHour] 22h20
9  2015-01-14 21:50:15 [Activity] Dress [Date] 14/01/2015 [StartHour] 10h00 [EndHour] 10h45
10 2015-01-14 21:55:01 [Activity] Toilet [Date] 14/01/2015 [StartHour] 10h45 [EndHour] 11h45
11 2015-01-14 21:58:25 [Activity] Toilet [Date] 14/01/2015 [StartHour] 20h30 [EndHour] 21h30
12 2015-01-15 21:40:45 [Activity] Dress [Date] 15/01/2015 [StartHour] 10h00 [EndHour] 10h45
13 2015-01-15 21:42:11 [Activity] Toilet [Date] 15/01/2015 [StartHour] 10h45 [EndHour] 11h45
14 2015-01-15 21:46:22 [Activity] Toilet [Date] 15/01/2015 [StartHour] 20h10 [EndHour] 21h10
15 2015-01-16 22:10:45 [Activity] Dress [Date] 16/01/2015 [StartHour] 10h00 [EndHour] 10h45
16 2015-01-16 22:12:11 [Activity] Toilet [Date] 16/01/2015 [StartHour] 10h45 [EndHour] 11h45
17 2015-01-16 22:16:22 [Activity] Toilet [Date] 16/01/2015 [StartHour] 20h40 [EndHour] 21h40
18 2015-01-17 21:40:45 [Activity] Dress [Date] 17/01/2015 [StartHour] 10h00 [EndHour] 10h45
19 2015-01-17 21:42:11 [Activity] Toilet [Date] 17/01/2015 [StartHour] 10h45 [EndHour] 11h45
20 2015-01-17 21:46:22 [Activity] Toilet [Date] 17/01/2015 [StartHour] 20h00 [EndHour] 21h05
```

Figure 4.18: Log file example

Using this log file and UPPAAL Model Checker, our application will now test the

elaborated care plan through TCTL queries in order to detect activities that have been executed without respecting the scheduled time ranges. The TCTL queries are of type:

Q1: E<>(Se && xd == startHour && xt == (endHour - startHour) && xp >= date && xp < (date + 24h))

Q2: E<>(Sw && xd == startHour && xt == (endHour - startHour) && xp >= date && xp < (date + 24h)).

These queries mean: Are there an execution state Se and its waiting state Sw of the label Activity where these 2 queries are satisfied?

```
1  2015-01-11 20:08:37 [Activity] Injection [Date] 11/01/2015 [StartHour] 10h00 [EndHour] 10h20.....OK
2  2015-01-11 20:10:37 [Activity] Dress [Date] 11/01/2015 [StartHour] 10h20 [EndHour] 11h05.....OK
3  2015-01-12 22:00:30 [Activity] Dress [Date] 12/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
4  2015-01-12 22:09:03 [Activity] Toilet [Date] 12/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
5  2015-01-12 22:12:37 [Activity] Toilet [Date] 12/01/2015 [StartHour] 20h30 [EndHour] 21h30.....OK
6  2015-01-13 22:00:45 [Activity] Dress [Date] 13/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
7  2015-01-13 22:01:03 [Activity] Toilet [Date] 13/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
8  2015-01-13 22:05:30 [Activity] Toilet [Date] 13/01/2015 [StartHour] 21h20 [EndHour] 22h20.....NOT OK
9  2015-01-14 21:50:15 [Activity] Dress [Date] 14/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
10 2015-01-14 21:55:01 [Activity] Toilet [Date] 14/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
11 2015-01-14 21:58:25 [Activity] Toilet [Date] 14/01/2015 [StartHour] 20h30 [EndHour] 21h30.....OK
12 2015-01-15 21:40:45 [Activity] Dress [Date] 15/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
13 2015-01-15 21:42:11 [Activity] Toilet [Date] 15/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
14 2015-01-15 21:46:22 [Activity] Toilet [Date] 15/01/2015 [StartHour] 20h10 [EndHour] 21h10.....OK
15 2015-01-16 22:10:45 [Activity] Dress [Date] 16/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
16 2015-01-16 22:12:11 [Activity] Toilet [Date] 16/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
17 2015-01-16 22:16:22 [Activity] Toilet [Date] 16/01/2015 [StartHour] 20h40 [EndHour] 21h40.....OK
18 2015-01-17 21:40:45 [Activity] Dress [Date] 17/01/2015 [StartHour] 10h00 [EndHour] 10h45.....OK
19 2015-01-17 21:42:11 [Activity] Toilet [Date] 17/01/2015 [StartHour] 10h45 [EndHour] 11h45.....OK
20 2015-01-17 21:46:22 [Activity] Toilet [Date] 17/01/2015 [StartHour] 20h00 [EndHour] 21h05.....NOT OK
```

Figure 4.19: Result

Figure 4.19 shows the result of the monitoring. OK means that the activity was performed respecting the scheduled temporalities and not OK means an anomaly.

4.3.4 Experimental evaluation

Here we present the performance evaluation on the proposed approach for the generation of the home care plan automaton, the interventions generation and also evaluation on realizability and monitoring tests. This experiment was done on four home care plans expressed using the different temporality patterns defined in this thesis: (i) Everyday pattern, (ii) Absolute dates pattern, (iii) Relative days pattern, and (iv) all the three patterns together. Each home care plan is composed of four activities and planned for one week period.

The experiment has been achieved on Intel Core i3 1.90GHZ with 4Go of RAM and UPPAAL-4.1.19. Figure 4.20 shows the evolution of the running time (w.r.t care plan automaton generation, realizability check, interventions generation and monitoring) and depending on the used pattern type.

We give more explanation depending on each pattern:

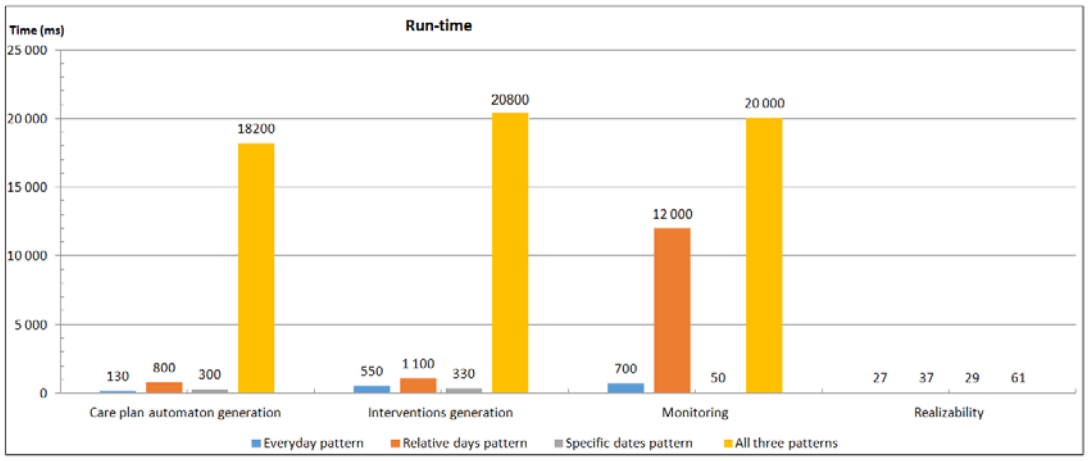

Figure 4.20: Execution time

- Care plan automaton generation: in this case, we note that the running time increases with the number of states and transitions in different timed automata. Indeed, when the home care plan is expressed using Everyday pattern or Absolute dates pattern, the running time is quite low because the number of states and transitions is quite reduced in contrast with Relative days pattern. In this last one, the number of transitions is multiplied approximatively by 7 (number of week days). In the same way, if the care plan is expressed in term of a composition of the three patterns the running time is even greater.

- Realizability: we note that UPPAAL Model Checker supports well the scalability. Proof, a reduced running time whatever the size of the care plan timed automaton. Only one TCTL query is necessary to check the realizability of the care plan.

- Monitoring: the running time is still also quite reasonable for the monitoring. The slight variation in the running time is due to the number of tested information from the log file. This corresponds to execute many times same queries related to the different activities.

- Interventions generation: As we can see, for a Relative days pattern automaton, interventions generation takes more time than the two other patterns automata. This is due to the increased number of transitions and states in this kind of automata. However, the running time becomes higher when generating interventions for an automaton combining the three patterns. This is due to the possible combinations of activities resulting from the composition operator (with interventions).

4.4 Conclusion of Chapter 4

We described in this chapter the prototype which allows to generate the care plan timed automata from the input of temporal specifications defined for a set of activities using the GUI. The resulting care plan automata were verified using the UPPAAL Model Checker through XML files and TCTL queries. We exposed the technical details of the prototype implementation which we exploited in order to conduct tests and experiments. In fact, we tested the prototype using real home care plans. These tests allowed us to assess efficiency of the prototype regarding the realizability, interventions generation and monitoring. In our experiments, we have found out that the application works well for the tested home care plans.

Conclusion

We investigated in this thesis the issues related to the design and analysis of the home care plan. Home care plans are inherently unstructured processes. This is due to the specificity of each patient, i.e., a specific care plan is required for each specific patient, as well as to the irregularity of repetitive activities within a plan. Nevertheless, we showed that using appropriate temporal expressions, it is possible to specify the schedule of such activities. In fact, we describe in this thesis a detailed presentation of the approach to construct temporal specification of the home care plan using a DSL. The obtained specification plays the role of a process model that must be enacted by a business process management or a workflow system.

In order to formally describe the home care plans constructed using the proposed DSL, we rested on a formal framework based on timed automata. Our proposal includes an approach to generate a formal specification of the home care plan, expressed as timed automata. We propose a three steps approach which consists in (i) the identification of the elementary patterns that are useful to specify the home care plans,(ii) the proposition of a specific composition algorithm that enables to combine patterns automata to build the activity automata, and then (iii) the construction of the global care plan automaton. The resulting care plan automaton encompasses all the possible allowed schedules of activities for a given patient. Once the care plan automaton is constructed, we can benefit from model checkers to make different analysis in a declarative way. We have illustrated realizability verification, monitoring of a care plan and grouping of activities into interventions using UPPAAL Model Checker.

Our specification language can easily be extended in order to increase its expressivity and its usability. Extensions are performed by introducing other patterns for defining elementary temporal expressions. For example, patterns such as "n times per day or per week" would be useful in a medical context. For preserving our modeling, it is necessary to ensure that each of these supplementary patterns can be transformed into timed automata.

The elaboration of a home care plan is an instance of the MTSPTW (Multiple Traveling Salesman Problem With Time Windows) with a time windows constraint for each activity and the optimization of interventions. In this type of problem, precedence constraints between tasks are very often considered. Introducing such constraints between activities into a care plan is sometimes necessary (for example, an injection of Lovenox

must be followed by a blood control before 15 days). Since it is possible to model precedence constraints with automata, we envisage an extension of our specification language to express this kind of constraint and to adapt our modeling into timed automata.

The interest of timed automata to model and solve planning and scheduling problems has been addressed in previous works [124][93]. Planning determines the possible plans while scheduling assigns resources to activities in order to respect the various constraints. Our approach deals only with planning preparation through the generation of interventions. An interesting issue will be to study how our care plan automata can be used to facilitate scheduling. For example, it has been shown [124] that optimal schedules correspond to shortest paths in timed automata. In our context, such an approach can be bind to optimal rounds after introducing distances between patients' homes. Extension of timed automata into priced timed automata [60] can also be worthwhile to consider since it permits to assign costs to activities. It is important to note that, when assigning resources to interventions, availability of human actors can be expressed with our temporal expressions.

In all these works for modeling planning and scheduling problems by automata, the specification of the problem is generally made by experts directly with the automata formalism. It would be necessary that end users can specify directly the main constraints of the problem. We think that our specification language can be enriched to offer such a possibility.

Bibliography

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems, Jun 1990.

[2] R. Alur, T.A. Henzinger, and H. Wong-Toi. Symbolic analysis of hybrid systems. In *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, volume 1, pages 702–707 vol.1, Dec 1997.

[3] Rajeev Alur. Timed automata. In Nicolas Halbwachs and Doron A. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99)*, volume 1633 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[4] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking in dense real-time. *INFORMATION AND COMPUTATION*, 104:2–34, 1993.

[5] Rajeev Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the Seventeenth International Colloquium on Automata, Languages and Programming*, pages 322–335, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[6] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[7] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. In *In Proceedings of the Sixth Conference on Computer-Aided Verification, number 818 in LNCS*, 1994.

[8] Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey, 1992.

[9] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–203, 1994.

[10] Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *In Proceedings of SFM'04, Lect. Notes Comput. Sci. 3185*, pages 1–24. Springer, 2004.

[11] Tobias Amnell, Elena Fersman, Paul Pettersson, Wang Yi, and Hongyan Sun. Code synthesis for timed automata. *Nordic J. of Computing*, pages 269–300, 2002.

[12] A. Arango. Domain analysis: From art form to engineering discipline. In *Proceedings of the 5th International Workshop on Software Specification and design (IWSSD)*, 1985.

[13] B.R.T Arnold, A. Van Deursen, and M. Res. An algebraic specification of a language for describing financial products. In *ICSE-17 Workshop on Formal Methods Application in Software Engineering*, pages 6–13. IEEE, 1995.

[14] Rym Ben Bachouch. Pilotage opérationnel des structures d'hospitalisation à domicile. *Thèse de doctorat de l'université de Toulouse*, 2011.

[15] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Represe ntation and Mind Series)*. The MIT Press, 2008.

[16] Roberto Barbuti, Nicoletta De Francesco, and Luca Tesei. Timed automata with non-instantaneous actions. *Fundam. Inf.*, 2001.

[17] R. Bastide, P. Bardy, B. Borrel, C. Boszodi, M. Bouet, K. Gani, E. Gayraud, D. Gourc, E. Lamine, P.-H. Manenq, M. Schneider, and F. Toumani. Plas'o'soins: A software platform for modeling, planning and monitoring homecare activities. 2014.

[18] Gerd Behrmann, Re David, and Kim G. Larsen. A tutorial on uppaal. pages 200–236. Springer, 2004.

[19] Nabil Belala. Formalisation des systémes temps-réel avec durées d'actions. *Thése de Magistére de l'Universié Mentouri-Constantine, Algerie*, 2005.

[20] Emna Benzarti. Home health care operations management : Applying the districting approach to home health care. *Thése de doctorat de l'Ecole Central Paris*, 2012.

[21] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundam. Inf.*, 36:145–182, 1998.

[22] Gastin P. Bérard, B. and A. Petit. On the power of non observable actions in timed automata. In *In Proceedings of the 13th Annual Sympsium on Theoretical Aspects of Computer Science (STACS'96)*, vol. 1046 of Lecture Notes in Computer Science, pp.257-268. Springer-Verlag, 1996.

[23] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE Trans. Softw. Eng.*, 17:259–273, 1991.

[24] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time petri nets. In *Proceedings IFIP*, pages 41–46. Elsevier Science Publishers, 1983.

[25] Bernard Berthomieu, Florent Peres, and François Vernadat. Bridging the gap between timed automata and bounded time petri nets. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS'06, pages 82–97, Berlin, Heidelberg, 2006. Springer-Verlag.

[26] Elizabeth Bjarnason. Applab: A laboratory for application languages. *Proceedings of the 7th Nordic Workshop on Programming Environment Reasearch (NWPER'96)*, 1996.

[27] H. Bohnenkamp, P.R. D'Argenio, H. Hermanns, and J. Katoen. Modest: A compositional modeling formalism for hard and softly timed systems. *Software Engineering, IEEE Transactions on*, 32(10):812–830, Oct 2006.

[28] Estéve D. Guennec J. Bonhomme S., Campo E. Prosafe-extended, a telemedicine platform to contribute to medical diagnosis. *journal of telemedecine and telecare*, 2008.

[29] Tripakis S. Bouajjani, A. and S Yovine. On-the-fly symbolic model-checking for real-time systems. In *In Proceedings of the 18th Real-Time SystemsSymposium (RTSS'97), IEEE Computer Society Press*, 1997.

[30] M. Bouet, K. Gani, M. Schneider, and F. Toumani. A general model for specifying near periodic recurrent activities - application to home care activities. In *e-Health Networking, Applications Services (Healthcom), 2013 IEEE 15th International Conference on*, pages 207–211, Oct 2013.

[31] Patricia Bouyer. Modéles et algorithmes pour la vérification des systémes temporisés. *Thése de doctorat de l'Ecole Normale Suérieure de Cachan*, 2002.

[32] Patricia Bouyer. Extended timed automata and time petri nets, 2006.

[33] Patricia Bouyer, Serge Haddad, and Pierre alain Reynier. Timed petri nets and timed automata: On the discriminating power of zeno sequences, 2006.

[34] F.D.J Bowden. A brief survey and synthesis of the roles of time in petri nets. *Mathematical and Computer Modelling*, 31:55 – 68, 2000.

[35] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification, 10th International Conference, CAV 98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer, 1998.

[36] Giacomo Bucci and Enrico Vicario. Compositional validation of time-critical systems using communicating time petri nets. *IEEE Transactions on Software Engineering*, pages 969–992, 1995.

[37] Hannes; Burwitz, Martin; Schlieter and Werner Esswein. Modeling clinical pathways - design and application of a domain-specific modeling language. In *Wirtschaftsinformatik Proceedings*, 2013.

[38] B. Bérard F. Cassez. Comparison of the expressiveness of timed automata and time petri nets. In *In Proc. FORMATS'05, vol. 3829 of LNCS*, pages 211–225. Springer, 2005.

[39] Franck Cassez and Olivier-H. Roux. Structural translation from time petri nets to timed automata. *Electron. Notes Theor. Comput. Sci.*, 128(6), 2005.

[40] Satish Chandra, Michael Dahlin, Bradley Richards, Randolph Y. Wang, Thomas E. Anderson, and James R. Larus. Experience with a language for writing coherence protocols, 1997.

[41] Tony Clark, Paul Sammut, and James S. Willans. Applied metamodelling: A foundation for language driven development (third edition). *CoRR*, abs/1505.00149, 2015.

[42] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, pages 244–263, 1986.

[43] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[44] P. Dadam, M. Reichertand, and K. Kuhn. Clinical workflows - the killer application for process-oriented information systems ? *Business*, 2000.

[45] Ministère de la santé. Glossaire des termes utilisés dans le cadre des réseaux de santé. *Observation nationale des réseaux de santé*, 2007.

[46] Gastin P. Diekert, V. and A. Petit. Removing ε-transitions in timed automata. In *In Proceedings of the 14th Annual Sympsium on Theoretical Aspects of Computer Science (STACS'97)*, vol. 1200 of Lecture Notes in Computer Science, pp.583-594. 1997.

[47] Bergman H. Ducharm F., Lebele P. Vieillissement et soins, l'urgence d'offrir des services de santé intégrés aux familles du xx1ème siècle. *revue transdisciplinaire en santé,*, 2:110–121, 2001.

[48] Conal Elliott. An embedded modeling language approach to interactive 3d and multimedia animation. *IEEE Trans. Softw. Eng.*, 25:291–308, 1999.

[49] Matthew J. Emerson, Janos Sztipanovits, and Ted Bapty. A mof-based metamodeling environment. *Journal of Universal Computer Science*, 10(10):1357–1382, 2004.

[50] Julien Mercadal Fabien Latry and Charles Consel. Processing domain-specific modeling languages: A case study in telephony services. In *In Generative Programming and Component Engineering for QoS Provisioning in Distributed Systems*, 2006.

[51] FNEHAD. La télésanté: un nouvel atout au service de notre bien-être, 2009.

[52] Martin Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

[53] Arjan Mooij-Boudewijn Haverkort Freek van den Berg, Anne Remke. Performance evaluation for collision prevention based on a domain specific language. In *Proceedings of the 10th European Workshop, EPEW, on Computer Performance Engineering*, pages 276–287. Springer Berlin Heidelberg, 2013.

[54] Kim G. Larsen Gerd Behrmann, Alexandre David. A tutorial on uppaal. In *Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, pages 200–236, Springer-Verlag, Septembre 2004.

[55] Hans-Michael Hanisch. Analysis of place/transition nets with timed arcs and its application to batch process control. In *Proceedings of the 14th International Conference on Application and Theory of Petri Nets*, pages 282–299. Springer-Verlag, 1993.

[56] John Hatcliff, Gary T. Leavens, K. Rustan M. Leino, Peter Müller, and Matthew Parkinson. Behavioral interface specification languages. *ACM Comput. Surv.*, 44(3):16:1–16:58, 2012.

[57] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. Hytech: A model checker for hybrid systems. *Journal of Software Tools for Technology Transfer*, 1:460–463, 1997.

[58] Gagnon M. Herbert R., Tourigny A. Intégrer les services pour le maintien de l'autonomie des personnesprisma (prog. de recherche sur l'intégration des services pour le maintien de l'autonomie). *EDISEM-Canada*, 2004.

[59] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Introduction to automata theory, languages, and computation, 2nd edition. *SIGACT News*, 32(1):60–65, March 2001.

[60] K. Subramani J. I. Rasmussen, Kim G. Larsen. Resource-optimal scheduling using priced timed automata. In *10th International Conference, Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, 2004.

[61] Salma Chahed Jebalia. Modelisation et analyse de l'organisation et du fonctionnement des structures d'hospitalisation á domicile. *Thése de doctorat de l'Ecole Centrale des Arts et Manufactures*, 2008.

[62] Luís Ferreira Pires Luciana Tricai Cavalini Antônio Francisco do Prado João Luís Cardoso de Moraes, Wanderley Lopes de Souza. A novel approach to developing applications in the pervasive healthcare environment through the use of archetypes. In *Proceedings of the 13th International Conference, ICCSA*, pages 475–490. Springer Berlin Heidelberg, 2013.

[63] Neil D. Jones, Lawrence H. Landweber, and Y. Edmund Lien. Complexity of some problems in petri nets. *Theoretical Computer Science*, 4(3):277 – 299, 1977.

[64] Wynn A. Jagger C. Spiers N. Parker G. Jones J., Wilson A. Economic evaluation of hospital at home versus hospital care: cost minimisation analysis of data from randomised controlled trial. *British Medical Journal*, 319:1574–1550, 1999.

[65] Abdelilah Kahlaoui. Méthode pour la définition des languages dédiés basée sur le métamodéle iso/iec24744. *Thèse de doctorat de l'école technologie supérieure université du Québec*, 2011.

[66] Samuel N. Kamin and David Hyatt. A special-purpose language for picture-drawing. In *Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL), 1997*, DSL'97, pages 23–23, Berkeley, CA, USA, 1997. USENIX Association.

[67] Dena M. Bravata Robyn Lewis Nancy D. Lin SA Kraft Moira McKinnon H Paguntalan Douglas K. Owens Kathryn M. McDonald, Vandana Sundaram. Closing the quality gap: A critical analysis of quality improvement strategies. *Stanford-UCSF Evidence-based Practice Center, for the Agency for Healthcare Research and Quality*, Vol. 7:210, 2007.

[68] Richard B. Kieburtz, Laura McKinney, Jeffrey M. Bell, James Hook, Alex Kotov, Jeffrey Lewis, Dino P. Oliva, Tim Sheard, Ira Smith, and Lisa Walton. A software engineering experiment in software component generation. In *Proceedings of the 18th International Conference on Software Engineering*, ICSE'96, pages 542–552. IEEE Computer Society, 1996.

[69] Baek Gyu Kim, A. Ayoub, O. Sokolsky, Insup Lee, P. Jones, Yi Zhang, and R. Jetley. Safety-assured development of the gpca infusion pump software. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 155–164, Oct 2011.

[70] Ilham Kitouni, Hiba Hachichi, Kenza Bouarroudj, and Djamel eddine Saidouni. Article: Durational actions timed automata: Determinization and expressiveness. *International Journal of Applied Information Systems*, 2012. Published by Foundation of Computer Science, New York, USA.

[71] David A. Ladd and J. Christopher Ramming. Two application languages in software production. In *Proceedings of the USENIX 1994 Very High Level Languages Symposium Proceedings on USENIX 1994 Very High Level Languages Symposium Proceedings*, VHLLS'94, pages 10–10, Berkeley, CA, USA, 1994. USENIX Association.

[72] Fabien Latry. Approche langage au développement logiciel : Application au domaine des services de téléphonie sur ip. *Thèse de doctorat de l'université de Bordeaux 1*, 2007.

[73] Paulo A. M. Silveira Neto Dhiego A. O. Martins Vinicius Cardoso Garcia Silvio R. L. Meira Leandro Marques do Nascimento, Daniel Leite Viana. A systematic mapping study on domain-specific languages. *The Seventh International Conference on Software Engineering Advances ICSEA*, 2012.

[74] F. Di Stanislao M. Panella, S. Marchisio. Reducing clinical variations with clinical pathways: do pathways work? *International Journal for Quality in Health Care*, 2003.

[75] Martin Gogolla Mark Ritchters. A metamodel for ocl. In *"UML"'99- The Unified Modeling Language*, 2003.

[76] Ron McCain. Reusable software component construction - a product-oriented paradigm. In *Proceedings of the 5th AiAA/ACM/NASA/IEEE Computers in Aerospace Conference*, 1985.

[77] A. L. Menezes, C. E. Cirilo, J. L. C. de Moraes, W. L. de Souza, and A. F. do Prado. Using archetypes and domain specific languages on development of ubiquitous applications to pervasive healthcare. IEEE Computer Society, 2010.

[78] Julien Mercadal. Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire. *Thèse de doctorat de l'université de Bordeaux*, 2011.

[79] Philip Meir Merlin. *A Study of the Recoverability of Computing Systems.* PhD thesis, University of California, Irvine, 1974.

[80] M. Mernik. *Dave Wile: Supporting the DSL Spectrum: paper review for the Journal of computing and information technology.* 2001.

[81] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 2005.

[82] Peter D. Mosses. Formal semantics of programming languages: An overview. *Electronic Notes in Theoretical Computer Science*, 148(1):41 – 73, 2006. Proceedings of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques (FoVMT 2004) Foundations of Visual Modelling Techniques 2004.

[83] J. Munnelly and S. Clarke. A domain-specific language for ubiquitous healthcare. In *Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on*, volume 2, pages 757–762, 2008.

[84] Jennifer Munnelly and Siobhán Clarke. Alph: A domain-specific language for cross-cutting pervasive healthcare concerns. In *Proceedings of the 2Nd Workshop on Domain Specific Aspect Languages*, New York, NY, USA, 2007. ACM.

[85] Isabel Navarrete, José A. Rubio, Juan A. Botía, José T. Palma, and Francisco J. Campuzano. Modeling a risk detection system for elderly's home-care with a network of timed automata. In *Proceedings of the 4th International Conference on Ambient Assisted Living and Home Care*, pages 82–89, Berlin, Heidelberg, 2012. Springer-Verlag.

[86] James Milne Neighbors. Software construction using components. *Doctorat Dissertation, University of California, Irvine*, 1980.

[87] Suzy Helène Germaine Temate Ngaffo. Des langages de modélisation dédiés aux environnements de métamodélisation dédiés. *Thèse de doctorat de l'université de Toulouse*, 2012.

[88] Provost H. Amico L. Berenguer M. Lombard F. Tyrrell J. Couturier P. Bosson JL. Wernert S. Schnee D. Basset D. Chemarin A. Frossard M. Nicolas L., Franco A. Téléassistance en hospitalisation à domicile: le programme visadom. *Lapresse médical*, 2005.

[89] A. Olivero and S Yovine. Kronos: a tool for verifying real-time systems - user's guide and reference manual. In *VERIMAG, Grenoble, France*, 1993.

[90] J.K. Ousterhout. Scripting: higher level programming for the 21st century. *Computer*, 31(3):23–30, Mar 1998.

[91] Lasbordes P. Livre blanc des systèmes d'information en hospitalisation à domicile, 2009.

[92] M. Pajic, Zhihao Jiang, Insup Lee, O. Sokolsky, and R. Mangharam. From verification to implementation: A model translation tool and a pacemaker case study. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, pages 173–184, April 2012.

[93] Sebastian Panek, Sebastian Engell, and Olaf Stursberg. Scheduling and planning with timed automata. In W. Marquardt and C. Pantelides, editors, *16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, volume 21, pages 1973 – 1978. Elsevier, 2006.

[94] Parnas.D.L. On the design and development of program families. *IEEE Transactions on Software Engineering*, 1:1–9, 1976.

[95] Juha pekka Tolvanen, Risto Pohjonen, and Steven Kelly. Advanced tooling for domain-specific modeling: Metaedit+, 2007.

[96] John Peterson, Paul Hudak, and Conal Elliott. Lambda in motion: Controlling robots with haskell. In *Proceedings of the First International Workshop on Practical Aspects of Declarative Languages*, PADL '99, pages 91–105, London, UK, UK, 1998. Springer-Verlag.

[97] Rubén Prieto-Díaz. Domain analysis: An introduction. *SIGSOFT Softw. Eng. Notes*, 15:47–54, 1990.

[98] Tarricone R. and Tsourous A.D. The solid facts: Home care in europe. world health organization. 2008.

[99] Olfa Rajeb. Proposition d'un cadre méthodologique pour le management de la continuité d'activité : application à la prise en charge à domicile. *Thèse de doctorat de l'université de Toulouse*, 2013.

[100] C. Ramchandani. Analysis of asynchronous concurrent systems by timed petri nets. Technical report, 1974.

[101] Laurent Révéillère. Approche langage au développement de pilotes de périphérique robustes. *Thèse de doctorat de l'université de Rennes 1*, 2001.

[102] Nicoletta De Francesco Roberto Barbuti and Luca Tesei. Timed automata with non-instantaneous actions. In *In Proceedings of the Concurrency, Specification and Programming Workshop CSP'2000*, 2000.

[103] Daniel A. Sadilek. Prototyping domain-specific language semantics. In *Companion to the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications*, OOPSLA Companion '08, pages 895–896. ACM, 2008.

[104] Roger S. Scowen. Extended bnf – a generic base standard. *Software Engineering Standards Symposium*, 1993.

[105] Joseph Sifakis. Use of petri nets for performance evaluation. In *Proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems*. North-Holland Publishing Co., 1977.

[106] A. Simalatsar and G. De Micheli. Medical guidelines reconciling medical software and electronic devices: Imatinib case-study. In *Bioinformatics Bioengineering (BIBE), 2012 IEEE 12th International Conference on*, pages 19–24, Nov 2012.

[107] Jiri Srba. Comparing the expressiveness of timed automata and timed extensions of petri nets. In Franck Cassez and Claude Jard, editors, *Formal Modeling and Analysis of Timed Systems*, volume 5215 of *Lecture Notes in Computer Science*, pages 15–32. Springer Berlin Heidelberg, 2008.

[108] Sergio Yovine Stavros Tripakis. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Desing*, 18:25–68, 2001.

[109] Kasper Osterbye Steen Brahe. *Business Process Modeling: Defining Domain Specific Modeling Languages by Use of UML Profiles*. Springer Berlin Heidelberg, 2006.

[110] Luca Tesei. Specification and verification using timed automata. *Doctorat Dissertation, University of Pisa*, 2004.

[111] Scott Thibault. Domain-specific languages: Conception, implementation and application. *Thèse de doctorat de l'université de Rennes 1*, 1998.

[112] Howard Wong-Toi Thomas A. Henzingr, Pei-Hsin Ho. Hytech: The next generation. In *In Proceedings of the 16th Real-Time SystemsSymposium (RTSS'95), IEEE Computer Society Press*, 1995.

[113] Leonid Mokrushin Paul Pettersson-Wang Yi Tobias Amnell, Elena Fersman. Times: A tool for schedulability analysis and code generation of real-time systems. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2003.

[114] Juha-Pekka Tolvanen. Domain-specific modeling: How to start defining your own language. *http://www.devx.com/enterprise/Article/30550*, 2006.

[115] Juha-Pekka Tolvanen. Domain-specific modeling for full code generation. *Model-Driven Development*, 2011.

[116] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, pages 222–226, 2006.

[117] Jeffrey J. P. Tsai, Steve Jennhwa Yang, and Yao-Hsiung Chang. Timing constraint petri nets and their application to schedulability analysis of real-time system specifications. *IEEE Trans. Softw. Eng.*, 1995.

[118] F. G. B. van den Berg, A. K. I. Remke, and B. R. H. M. Haverkort. A domain specific language for performance evaluation of medical imaging systems. In *Proceedings of the 5th Workshop on Medical Cyber-Physical Systems, MCPS 2014, Berlin, Germany*, pages 80–93. Schloss Dagstuhl, 2014.

[119] F.G.B. van den Berg, A.K.I. Remke, and Haverkort. A domain specific language for performance evaluation of medical imaging systems. 2014. Proceedings of the 5th Workshop on Medical Cyber-Physical Systems, MCPS 2014.

[120] G. Van Den Broek, F. Cavallo, and C. Wehrmann. *AALIANCE Ambient Assisted Living Roadmap*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2010.

[121] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, 2000.

[122] David M. Weiss. Family-oriented abstraction specification and translation : the fast process. In *Proceedings of the 11th AnualConference on Computer Assurence (COMPASS), page 14-22*, 1996.

[123] M. Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer-Verlag New York, Inc., 2012.

[124] Oded Male Yasmina Abdeddaîm. Job-shop scheduling using timed automata? In *13th International Conference, Computer Aided Verification (CAV'2001)*, 2001.

[125] Sabrina Zefouni. Aide à la conception de workflows personnalisés : application à la prise en charge à domicile. *Thèse de doctorat de l'université de Toulouse*, 2012.

[126] Rahul Mangharam Zhihao Jiang, Miroslav Pajic. Cyber-physical modeling of implementable crdiac medical devices. *Real time and embded system Lab (mLAB)*, 2011.