A Model Driven Care Plan Modelling System

Abizer Shabbir Khambati

This thesis is submitted in fulfilment of the requirements for the degree of Master of Engineering in Software Engineering, completed at The University of Auckland.

February 2008

Abstract

People suffering from chronic illnesses often need to manage their health by themselves for large periods of time, they are required to take a proactive part in their treatment process, and there is a great potential for providing them as well as their health care professionals support in managing and achieving better health results.

There have been a multitude of software solutions that have been designed in the area of health management, some solutions focus on aiding the health care professional by providing them with ways to model complex health care information and allowing them to reuse this information for decision support and patient treatment. These solutions however have a low focus on transforming this health care information in a form which can be useful to the patient, and simply focus on aiding the health care professional without being concerned on how this information may be delivered to the patient.

Other health management solutions have been focussed on providing the patient with support in managing their health; these solutions have been commonly in the form of electronic wellness management software, which uses a combination of artificial intelligence and expert systems, to provide the patient with continuous management support. The problem with these solutions is that they have a weak connection with the health care professional, and such type of wellness management applications are generally very disease specific, and a generic solution which could be adaptable to a large variety of disease management support is lacking. Furthermore a major drawback for most of these solutions is that they tend to be device and platform specific which can be a barrier to the uptake of the solution.

In this thesis we present the development and prototyping of an integrated care plan modelling system which provides support for both the health care professional and the patient. The system developed will allow health care professionals to model health care information for patient treatment in the form of reusable health care plans, as well as allow these health care plans to be delivered to the patient on a generic wellness management or patient application which would give the patient support related to the health care plans designed for them. In addition to this the system also allows a GUI designer to model reusable device specific models of the patient application so that these can be used to generate the patient application for multiple device types.

Acknowledgements

I would like to express my heart felt thanks to my three supervisors, Dr. John Grundy, Dr. John Hosking, and Dr. Jim Warren, for their invaluable support in all aspects of my work. Their experience, knowledge, and commitment to the field as well as their students has meant that I could achieve a quality and satisfaction in my work and research that would have otherwise been not possible. I would also like to extend my thanks to Mr. Christian Hirsch for his help with the informal evaluation.

Lastly I would like to extend my thanks to my dear family without whose support and encouragement I would have found it difficult indeed to achieve anything.

Table of Contents

List of Figures

Chapter 1 - Introduction

1.1 Introduction

There are a multitude of ailments and illnesses that people across the globe face on a regular basis some of these ailments are of an acute nature and last for a short period of time, easily and quickly cured for example the common cold, sore throat, etc, some ailments however are much more longer lasting and chronic in nature, for example smoking, obesity, alcoholism, diabetes etc. Out of the two types of ailments, chronic ailments and diseases require the most management and care, the main reason for this is that these disease being long term require patients to self monitor their health for prolonged periods of time, over which they need to always remember their treatment, effectively record health measurements and data, follow a set health care plan, and effectively communicate with their health care professionals. Hence the research that is done under this thesis aims to develop an integrated approach through which patients suffering from chronic, long term ailments and the health care professionals that care for them are supported.

The remainder of the chapter discusses the motivations behind the research in more detail, also explaining the main goal behind the work along with an overview of the thesis chapters.

1.2 Motivation

When a patient suffers from a chronic illness, there are many management tasks that the patient is suppose to administer or carry out by themselves, for example they are expected to follow the health care plans that are prescribed to them by their health care professional, they are expected to do the health activities prescribed to them, they may be required to maintain a log or a diary recording their concerns, or health measurements they may be making, and they are needed to communicate with their health care professionals. The health care professionals in turn have to cope with structuring, modelling and reuse complex health care information, as well as being responsible for efficiently distributing relevant elements of this information to the patient.

There have been many software systems which have been designed with health care professionals in mind, allowing them to use the system to model complex health care information such as clinical guidelines and then reuse this information repeatedly for decision making for multiple patients suffering from similar ailments, however one of the key limitation of these systems include the fact that there is no attention paid into how this information could be transformed and delivered to the patient. Like wise there have been a multitude of software applications in the form of wellness management applications, designed to support patients in managing their health care, however these applications are generally very disease and device specific, for example a diabetes diary for a PDA is definitely useful if you are suffering from diabetes and you own a PDA, however in reality patients often suffer from various types of chronic illnesses simultaneously, and secondly a patient may not possess a PDA and most likely has a mobile phone. The other key limitation with these solutions is that such applications rely more on expert systems rather than health care professionals.

Therefore the main motivation for the work done in this thesis is to develop and prototype an integrated health care management system which would not only allow health care professionals to model health care information for the patient, but allow them to store and reuse this information for many patients as well as allow them to easily distribute this information in the form of a personalised wellness patient application in a device independent manner.

1.3 Goal

The goal of the research work done was as mentioned above to create a health care management system which would allow a health care professional to easily model health care information and then delivery it to the patient. This goal can be split up into five main parts; the first part of this overall goal includes the development of a visual language suite which would allow health care professionals to model health care plan templates for patients suffering from some type of ailment. These care plan templates could be reused for multiple patients suffering from similar ailments.

The second part of this goal was to develop a care plan instantiation tool, this tool would allow health care professionals to make the generic care plan template more specific to the patient before it is given to the patient, this would include incorporating into the template personalised and lifestyle information to make it a personal patient care plan instance.

The third part of the goal would be a wellness management patient application which would preferably run on a mobile device of some kind. The fourth part of the goal is to develop a visual modeller which can allow the wellness management patient application to be modelled by a designer, so that the designer can use the modeller to design definitions of how the patient application will look on different mobile devices.

The fifth and final part of the goal was to design a generator which would take in as input a modelled definition of the patient application suitable for a mobile device which the patient possesses and then generating the application in Flash.

1.4 Overview of Thesis

- Chapter 1: Introductions – Discusses the main motivations and goals of the thesis.
- Chapter 2: Background and related research – Covers the background research which was done as part of this research in the areas of health care modelling, wellness management, adaptive user interface design systems.
- Chapter 3: Overview of Approach – Gives a very overall description of the approach followed for the development of the care plan modelling system.

- Chapter 4: User Needs – Introduces and discusses in more detail the stakeholders of the care plan modelling system as well as their needs pertaining to the system.
- Chapter 5: The Care Plan Domain Model – Discusses the care plan domain model which forms the basis of the care plan modelling visual language.
- Chapter 6: Functional and Non-Functional Requirements – Discusses the functional and non-functional requirements of the care plan modelling system which arise from the user needs.
- Chapter 7: Design for the Care Plan Modelling System – Discusses the design of the components of the care plan modelling system.
- Chapter 8: Implementation of the Care Plan Modelling System – Discusses the implementation of the care plan modelling system.
- Chapter 9: Case Study and Prototype – Discusses a case study which demonstrates the prototype of the care plan modelling system in action.
- Chapter 10: Evaluation – Discusses the results from the cognitive walkthroughs as well as the informal evaluation done of the care plan modelling system.
- Chapter 11: Conclusions – Discusses the contributions of the thesis as well as the avenues for future work.

1.5 Summary

Patients suffering from chronic illnesses face a multitude of challenges in managing and maintaining their health care, these challenges include following prescribed treatment procedures, self monitoring their health, following a long term health care plan, communicating with their health care professional their concerns as well as changes in their health, and much more. The challenges faced however are not only by the patient, the health care professionals who care for these patients in turn are required to cope with large volumes of complex health care information, through which they are expected to sift, structure and reuse for multiple patients suffering from similar ailments.

There have been many approaches in the form of software systems to support both user groups, for the patient there have been a large array of wellness management applications which enable them to self monitor, manage, learn about, etc their chronic ailment, however some of the key limitations of this approach have been the fact that these solutions tend to be quite specific to the disease, and hence are not adaptive enough for a vast variety of chronic ailments, furthermore the solutions are reliant on particular types of technologies which the patient may or may not posses, lastly these application do not include the health care professional in the management process and are more reliant on the wellness management applications connecting to expert systems. For the health care professional as well there have been many software based solutions, which have been aimed at allowing them to model, structure and reuse health care information, however the focus here has been on providing health care professionals decision making support rather than allowing this information to be transformed in a way which can be delivered to the patient. Hence the goal of the research done here is to develop an integrated system, which would allow both user groups to be supported, this system would involve the a visual language for the development of health care plans, a tool to personalise the health care plans to the patient, a patient wellness application onto which the health care plans can be transferred, a visual language for modelling the patient application for different devices, and lastly a generator to automatically generate the patient application from these definitions in Flash.

The thesis is split up into eleven chapters, which includes the introduction chapter, chapter two on background and related research, chapter three on an overview of the approach, chapter four on the users of the system and their needs, chapter five on the care plan domain model, chapter six on the functional and non-functional requirements of the system, chapter seven on the design of the care plan modelling system, chapter eight on the implementation of the care plan modelling system, chapter nine on the case study with prototype of the system, chapter tem on evaluations of the system and approach, and finally chapter eleven on conclusions, contributions and future work pertaining to the research done.

Chapter 2 - Background and Related Research

2.1 Introduction

The purpose of this chapter is to introduce the background research which was done for the research project of this masters thesis. The focus of the chapter is on the related research done by other researchers on developing health care systems for planning modelling and managing patient health, with a variety of technological and theoretical work being discussed. Also discussed in this chapter are the current approaches of deploying the health care information to the patient, and hence allowing the patient to utilise the health care information modelled by health care professionals in a direct manner.

2.2 Health care modelling theories and systems

This section focuses on discussing the research done with regards to the theory of health care modelling and the systems built to allow such type of modelling.

2.2.1 GLIF – The guideline interchange format - (Boxwala, Peleg et al. 2004; OpenClinical 2007)

GLIF is a framework for capturing the treatment regimes which are embedded in clinical treatment guidelines in the form of flow diagram(Boxwala, Peleg et al. 2004; OpenClinical 2007). The syntax of the GLIF guideline modelling framework can be described as the set of decisions and actions that are needed to get a patient from one state to another. The latest GLIF framework has the following components:

1. A “Decision step” – This represents the different decisions that the health care professional has to make. GLIF has many different concrete decision classes extending the “Decision step” in order to cater for the different models and processes of different decisions (Boxwala, Peleg et al. 2004; OpenClinical 2007).

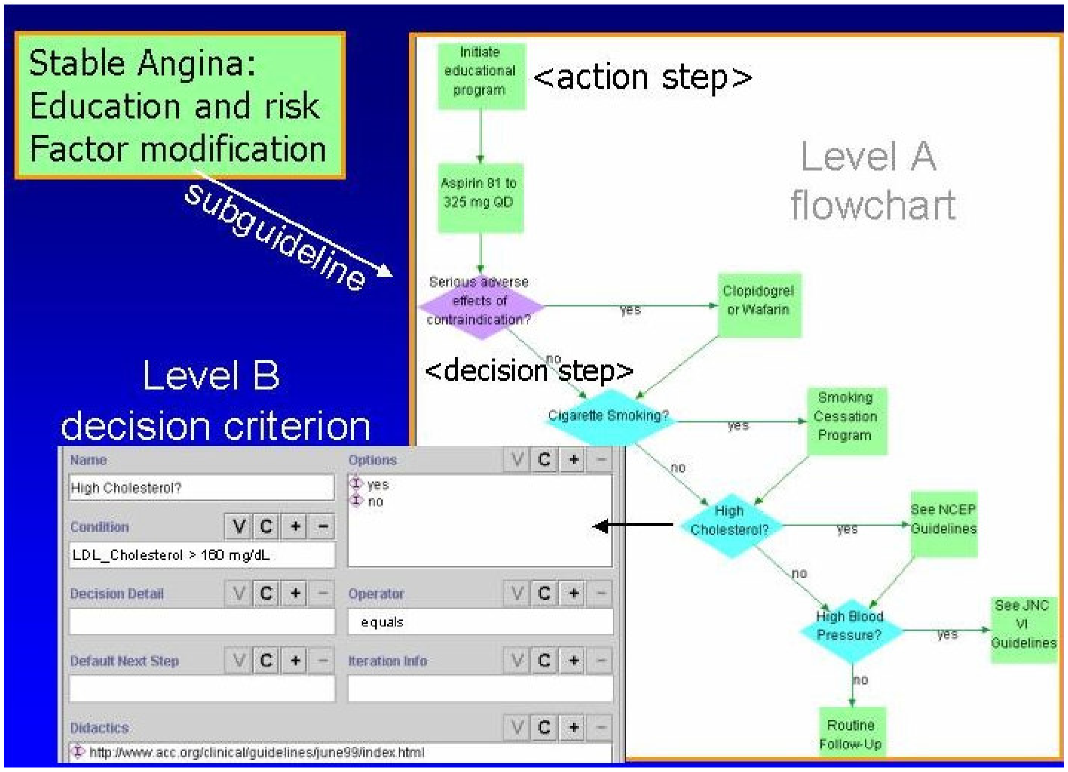
2. An “Action step” – This represents the actual actions or activities that the health care professional has to carry out, whether this action is some form of patient examination, or assessment, or it is something the health care professional needs to prescribe to the patient. Again just like the decision step, different concrete action classes representing different models of actions can be extended from the generic action step class (Boxwala, Peleg et al. 2004; OpenClinical 2007).

3. A “Branch step and Synchronisation step” – these two steps are used to model in a GLIF guideline different paths that may be taken in the treatment of a patient based on certain conditions, i.e. conditional flow to actions and decisions (Boxwala, Peleg et al. 2004; OpenClinical 2007).

4. A “Patient state step” – this step describes the actual condition of the patients health, a patient state step can be defined at various points in the flow of the clinical guideline modelled in GLIF, a guideline can be started with it to define what the patient state is likely to be before treatment commences, and various points in the middle to indicate how actions and decisions taken are expected to affect the patient health, and finally one at the end

of the guideline to describe the state of the patient after treatment finishes (Boxwala, Peleg et al. 2004; OpenClinical 2007).

The GLIF framework for modelling health care treatment guidelines was implemented in the Protégé knowledge management software (Boxwala, Peleg et al. 2004; OpenClinical 2007), and figure 1 shows the treatment guideline for stabilising angina, as implemented using the GLIF framework in Protégé.



Name: High Cholesterol?
Options: yes, no
Condition: LDL_Cholesterol > 160 mg/dL
Operator: equals
Didactics: http://www.acc.org/clinical/guidelines/june99/index.html

Figure 1: An example of the treatment guideline encoded in the GLIF format reproduced from (OpenClinical 2007).

2.2.2 PROforma – Executable clinical guidelines - (Fox, Johns et al. 1998; OpenClinical 2007)

Proforma similar to GLIF is a language and a framework in one which allows the modelling of and execution of medical and more specifically clinical knowledge (Fox, Johns et al. 1998; OpenClinical 2007). The object model of the Proforma approach can be seen in figure 2 below. It can be seen that all objects in the Proforma object model inherit from a generic task. A decision poses a question of the patient's health and requires the health care professional to make some sort of judgement, investigation, or assessment of the patient's health (Fox, Johns et al. 1998; OpenClinical 2007). Then there are actions, actions are actual health activities that the health care professional engages with the patient, for example giving the patient medicine (Fox, Johns et al. 1998; OpenClinical 2007). The other object in the proforma model is the enquiry object. The need for this object arises from the need to specify patient information and data in the clinical guideline which is needed by the health care professional as well as specifying where to find this information, for example patient health records (Fox, Johns et al. 1998; OpenClinical 2007). The final object in the proforma modelling language is a plan (or a care plan). A plan is a composite of other tasks, which includes other plans, decisions, actions, and enquiries (Fox, Johns et al. 1998; OpenClinical 2007). These four objects in the proforma model are essentially basic classes and can always be enhanced and extended by other more refined classes which specify finer elements of detail, for example having many different sub-classes of action defining the different action models that are possible (again similar to GLIF) (Fox, Johns et al. 1998; OpenClinical 2007).

The proforma framework also specifies the type of operations that can be performed on a model based on the object structure specified in the diagram below. Some of the main operations that can be performed using the proforma model of a clinical treatment guideline include triggers, problem solving and argumentation, scheduling and data acquisition (Fox, Johns et al. 1998; OpenClinical 2007).

“Triggers” allow the health care professional to receive reminders and alerts to medical knowledge which may be relevant while executing the clinical guideline modelled in proforma when the health care professional comes to a certain decision or state in the clinical guideline flow diagram modelled (Fox, Johns et al. 1998; OpenClinical 2007). “Problem solving and argumentation” focus on deductive logic to help the health care professional assess the pros and cons for treatment paths and regimes conveyed by the proforma modelled clinical guidelines, well as “scheduling” operations give the health care professional the ability to set specific schedules in time for actions that are part of the clinical guideline (Fox, Johns et al. 1998; OpenClinical 2007). Finally “data acquisition” operations allow the data collected during clinical practice (or even a consultation with a patient) to be interpreted by the model of the clinical guideline encoded in the Proforma logic (Fox, Johns et al. 1998; OpenClinical 2007).

Using the proforma object model along with the different operations that are possible with these objects, health care professionals can encode medical (and clinical) processes of assessing and treating patients. Similar to GLIF, Proforma is a language/framework, and hence there exists more than one implementation of the proforma logic, two common implementations of the Proforma framework include Arezzo a commercial product from InterMed Ltd London, and Tallis a Java based implementation (Fox, Johns et al. 1998; OpenClinical 2007).

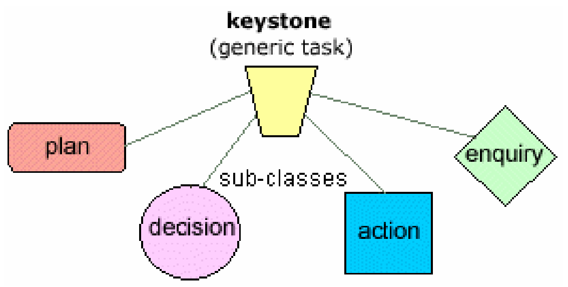


Figure 2: The object model for PROforma reproduced from (OpenClinical 2007).

2.2.3 Asbru – Task specific and intention based care plans - (Miksch, Shahar et al. 1997)

Asbru is a modelling suite which allows clinicians to model "intention based" (Miksch, Shahar et al. 1997) health care plans for treating patients, whilst providing the capability within the suite to execute the care plans for the health care professional. As with the PROforma and GLIF frameworks, Asbru is also based on a modelling language. The most important component in the Asbru object model is the plan (or care plan), each object firstly contains a body which defines the actions that are to be performed as part of the plan; these actions can be divided into a variety of sub plans. In addition to this each plan also includes a clear intention, a set of preferences, conditions and effects (Miksch, Shahar et al. 1997). The intention of a plan defines what the plan is suppose to achieve through the actions (and sub plans) that it consist of. The preferences of a plan are designed to specify the cases or metrics by which the suitability of the plan to the need at hand for treating a patient can be determined, for example a health care professional could define a preference which consisted of a heuristic to determine the relevance of a plan to treat a particular patient scenario, hence defining what condition would be a perfect fit for this plan, hence easing the selection of the plan (Miksch, Shahar et al. 1997).

Conditions of a plan are responsible for the transition of the plan from one state to another. A plan can exist in one of four states, active, completed, aborted and suspended and conditions define when a plan should move to either one of these states (Miksch, Shahar et al. 1997). A health care professional when defining a plan using the Asbru syntax may also decide to model parameters and arguments as part of the plan, for example a diabetes treatment plan may have measurable parameters which can be defined such as blood glucose sugar and weight (Miksch, Shahar et al. 1997). An effect defined under a plan defines the relationship between such types of measurable parameters, or the effect on these measurable parameters by the execution of the plan on the patient. For example executing the diabetes plan on the patient for 3 months may have the effect of reducing the patient's blood sugar, as well as their weight, and it may be that if their weight is decreased then there is a stronger chance that their blood sugar will decrease even further, the idea behind "effects" in the Asbru syntax is to describe such relationships (Miksch, Shahar et al. 1997).

Finally Asbru allows all care plans to be modelled along a concrete time line, allowing health care professionals to position the plans on a three dimensional time oriented plane, where the plans can be modelled to execute, concurrently (or in parallel) or sequentially, or cyclically (Miksch, Shahar et al. 1997).

2.3 Patient Wellness Management Applications

The above sections focus on the different types of frameworks and applications which allow the modelling of health care information, whether this health care information is in the form of clinical guidelines, or in the form of more patient oriented care plans. This section focuses on the patient wellness management applications designed to aid patients suffering from long term chronic illnesses managing their own health in an efficient way. The section discusses some of the typical applications and technologies currently prevalent in the domain.

2.3.1 Wireless Weight Wellness Monitor - (Parkka, Gils et al. 2000)

The motivation behind the wireless weight wellness monitor (WWWM) was to design and implement a wireless architecture to monitor and manage weight loss in overweight individuals. The WWWM consist of four main components, the first component is a set of measurement devices which the patient may use to measure some aspect of their health, in the case of weight management this may most commonly include weight and heart rate measure devices, these devices are assumed to be Bluetooth enabled (Parkka, Gils et al. 2000). The second component of the WWWM includes a mobile device of some kind (e.g. a PDA or a mobile phone), through this mobile device the patient can view their personal data, as well as data they have measured through the measurement devices, as well as expert advice from remote expert servers (Parkka, Gils et al. 2000). The third component of the WWWM consist of a home health server, this server stores a personal registry where the measurement data recorded by the patient is stored along with other personal information of the patient. The home health server also stores proxies for the measurement devices which are incapable to have a JVM (this is because the entire implementation of the software for the system is done in Java) (Parkka, Gils et al. 2000). The measurement devices, and the mobile devices, assumed to be Bluetooth enabled communicate with the home health server through the “Bluetooth ad-hoc networking” (Parkka, Gils et al. 2000) while at home, and the mobile devices such as the PDA can still communicate with the home health server through a HTML browser connecting to the web server on the home health server. The fourth and final component of the WWWM is a whole array of remote expert advice servers, the knowledge data bases that are present on these servers, for example the nutrition databases, and the diet databases are hosted as RMI services on the home health server so that a patient can view the data through their mobile device by connecting to their home health server (Parkka, Gils et al. 2000). The set up can be seen below in the diagram reproduced from the original literature.

Through this wellness monitor the patient can conduct health measurements (such as weight and heart rate) and monitor their own health, every time they enter in a measurement there may be an application running either on the application on their mobile device or on the home health server which may analyse their data and give them instant feedback on how well they are doing on their weight loss program, similarly the patient can receive expert advice at anytime from the remote health servers to their mobile device, hence the weight wellness monitor uses a two pronged approach to improve a patients health, firstly through quick feedback to give the patient experience on what strategies work and what don’t and secondly giving them expert advice where needed (Parkka, Gils et al. 2000).

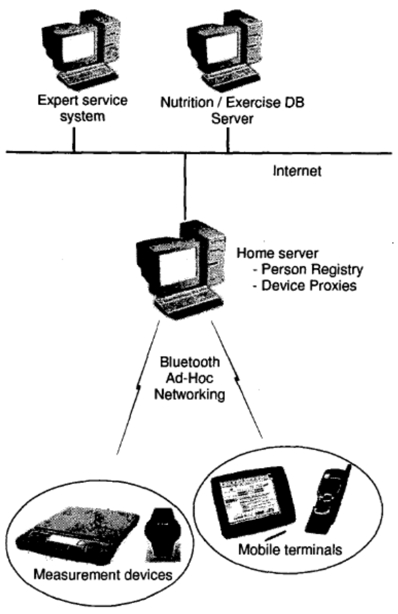


Figure 3: Diagram showing the architecture of the wireless weight wellness monitor, diagram reproduced from (Parkka, Gils et al. 2000).

2.3.2 Patient centred assessment and counselling mobile energy balance (PmEB) - (Tsai, Lee et al. 2007)

The PmEB is yet another mobile device based wellness application aimed at people suffering from eating disorders which lead to them being overweight or obese. The main logic behind the application is based on mathematical analysis of a patient's calories based on their diet, and their level of physical activity, working out a value known as the "calorie balance" which indicates to the user how far or over they are from their optimum calorie goal (Tsai, Lee et al. 2007). The mobile phone based application was implemented with a client server architecture. The client in the application is the mobile device application itself and the server is the health server with which all data is synchronised automatically every 24 hours (connectivity permitting). Through the mobile phone client a user can view their "current calorie balance", enter in new calorie values, search for and read information on different food groups and their calorie information, enter in the different types of physical activities they may have conducted and the level of physical exertion, as well as this the patient can receive regular reminders and alerts via SMS messaging, telling them when they should update their calorie information (Tsai, Lee et al. 2007). The entire process can be seen in figure 4 below, with the sample client application seen in figure 5 below it. The PmEB similar to the wireless weight wellness monitor, aims to educate the patient about the ideal diet for them by giving them instantaneous feedback on what they are eating the effect of it (in terms of calories) that it will have on their health, as well as providing the patient with helpful reminders, and making health care information such as calorie and health information on different food groups available to them through the mobile application, as well as making information available to them with regard to the physical exercises they can do in order to burn those calories (Tsai, Lee et al. 2007).

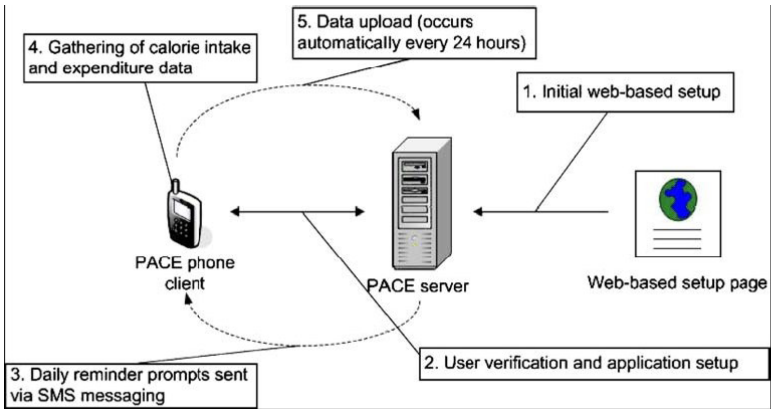

5. Data upload (occurs automatically every 24 hours)
4. Gathering of calorie intake and expenditure data
1. Initial web-based setup
PACE phone client
PACE server
Web-based setup page
2. User verification and application setup
3. Daily reminder prompts sent via SMS messaging

Figure 4: The process and architecture of the PmEB wellness application reproduced from (Tsai, Lee et al. 2007).

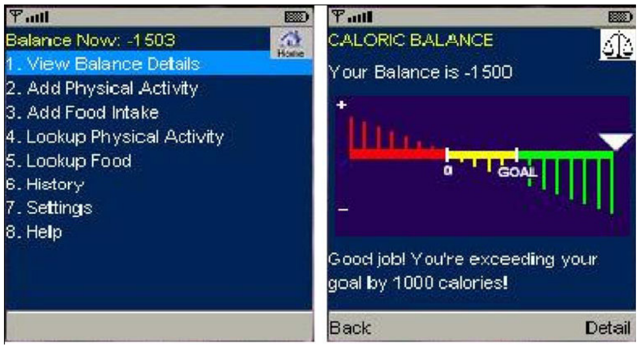
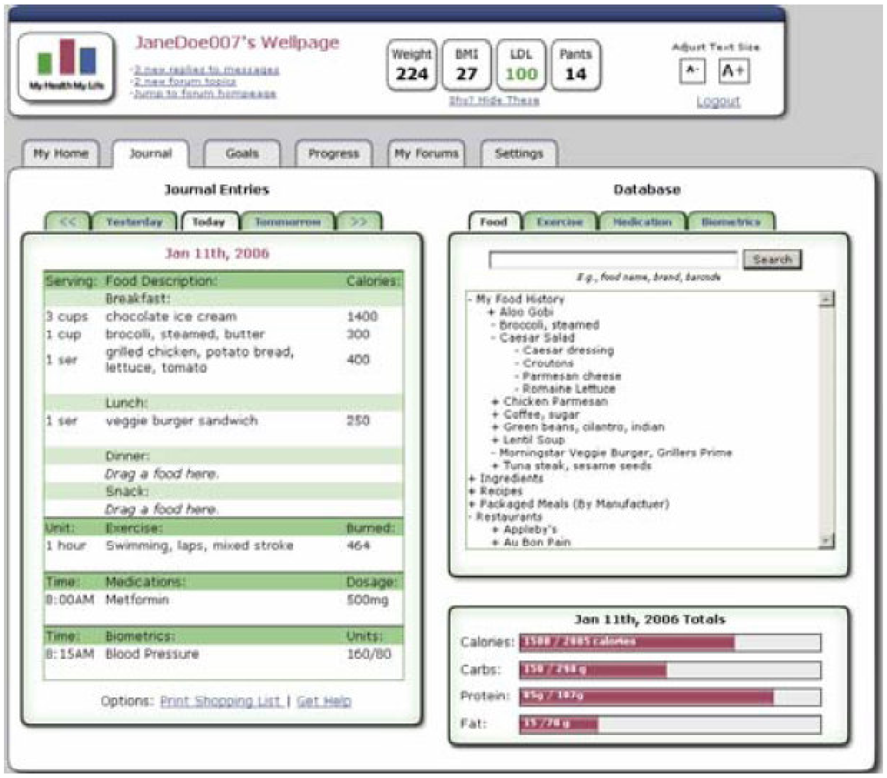


Figure 5: Screenshots of the mobile phone application used to record calorie information, and display the calorie balance of a patient, as well as giving them information on types of food and physical activity, reproduced from (Tsai, Lee et al. 2007).

2.3.3 A web based wellness application: My Health, My Life - (Nuschke, Holmes et al. 2006)

The My Health, My Life (MHML) wellness application is a web based application aimed at patients suffering from various long term illnesses such as diabetes, high blood cholesterol, obesity etc. It is meant to be in the form of a journal as well as a social networking website (Nuschke, Holmes et al. 2006). The patient starts of using the web application by first specifying a whole list of health goals that they want to pursue, for example in the screenshot shown below it can be seen that Jane has four health goals, firstly there is a weight goal, then a BMI (body mass index) goal, then a LDL cholesterol level goal, and finally a goal to reach a certain pant size (Nuschke, Holmes et al. 2006). Once the patient makes these goals, the web application personalises for the patient allowing the patient to record data related to their health goals like an electronic journal, furthermore Jane can always view her progress towards her goals in the form of bar graphs or other data representation charts, the web application may also have pre-programmed functions to calculate nutrition and other health information that Jane could utilise to monitor her own health (Nuschke, Holmes et al. 2006). In addition to all this Jane can share experiences and data with other patients and may be even health professionals through the forums option (see the tabs in line with the journal tab in figure 6 below). In addition to this the MHML web application has a series of built in food, exercise and medication database to provide information to the patient, using these the patient can personalise the web based journal to include favourite foods, recipes etc (Nuschke, Holmes et al. 2006). A prototype of the MHML was built with emphasis on a diabetic management/diary/journal. One of the disadvantages of this application was that it was not available to run on mobile devices which would have made for a very convenient attribute of this solution.



JaneDoe007's Wellpage

-2 new replies to messages
-2 new forum topics
-Jump to forum homepage

Journal Entries

Jan 11th, 2006

Serving:	Food Description:	Calories:
	Breakfast:	
3 cups	chocolate ice cream	1400
1 cup	brocolli, steamed, butter	300
1 ser	grilled chicken, potato bread, lettuce, tomato	400
	Lunch:	
1 ser	veggie burger sandwich	250
	Dinner:	
	Drag a food here.	
	Snack:	
	Drag a food here.	
Unit:	**Exercise:**	**Burned:**
1 hour	Swimming, laps, mixed stroke	464
Time:	**Medications:**	**Dosage:**
8:00AM	Metformin	500mg
Time:	**Biometrics:**	**Units:**
8:15AM	Blood Pressure	160/80

Options: Print Shopping List | Get Help

Database

E.g., food name, brand, barcode

- My Food History
  + Aloo Gobi
  - Broccoli, steamed
  - Caesar Salad
    - Caesar dressing
    - Croutons
    - Parmesan cheese
    - Romaine Lettuce
  + Chicken Parmesan
  + Coffee, sugar
  + Green beans, cilantro, indian
  + Lentil Soup
  - Morningstar Veggie Burger, Grillers Prime
  + Tuna steak, sesame seeds
+ Ingredients
+ Recipes
+ Packaged Meals (By Manufactuer)
- Restaurants
  + Appleby's
  + Au Bon Pain

Jan 11th, 2006 Totals

Calories: 1588 / 2885 calories
Carbs: 158 / 298 g
Protein: 85g / 187g
Fat: 35 / 78 g

Figure 6: Screenshot of the wellness page of a patient called Jane Doe reproduced from (Nuschke, Holmes et al. 2006)

2.4 Previous Research – A Personal Health Management Application

The personal health management application was a system which was designed and implemented by Abizer Khambati and Sumedh Kanade as part of an earlier research project (Kanade 2006; Khambati 2006). The aim of the research project was to develop a system where by a health care professional can design a health care plan for a patient, and deploy it to the patient onto a PDA application. The project was roughly split up into three main parts, the first part of the project involved the development of a model to represent health care plans for patients (this model was used as a basis for the care plan domain model expressed in chapter 5 of this thesis), the second part of the project involved in the implementation of a web based application through which the health care professional could model the health care plans for the patient, the final and third part of the system was a PDA application implemented using the .Net compact framework which allowed an instance of the designed health care plan to be loaded onto the PDA allowing the patient to use it.

The health care plan according to the care plan model was composed of three main components, firstly there were patient health goals and motivations, then there were health care activities, and finally there were other care plans. The website which was implemented in order to allow for the modelling of these health care plans provided an interface through which the health care professional could register their patients, including recording personal demographic information, and then model and tailor health care plans for the patient through the inbuilt wizards of the website. Once a care plan was prepared through execution of wizards, then it could be sent onto the PDA of the patient through the PDA application. Through this PDA application the patient can view their entire care plan including all the sub care plans, health care activities, and goals and motivations modelled for them, as well as this they can receive helpful reminders to do activities, and record data through the application, which later on can be synchronised with the health care professional website where the health care professional can view it and analyse it.

2.5 Discussion

There are two main types of applications (or systems) that are discussed in the literature described above, the first type is that of different health care modelling frameworks and systems, which can be used by the health care professional to model and reuse different types of health care information for treating and managing patients, there are numerous of these type of applications, and the three examples discussed above are just some of the important ones which are worth attention. GLIF and Proforma are both frameworks which have been designed for the modelling of clinical guidelines and procedures, hence transforming the relatively complex treatment information in the guidelines into a more formal reusable computer processable structure; both GLIF and Proforma have a slightly different object model; however the general idea is the same. Proforma adds a little more sophistication than GLIF by further developing the framework to put in customised operations which can be executed on the guidelines hence making the guidelines executable (Fox, Johns et al. 1998; OpenClinical 2007). Asbru on the other hand takes a slightly different approach to both Proforma as well as GLIF, in that where GLIF and Proforma focus on the modelling of entire treatment guidelines, Asbru focuses on small modules of treatment procedures called care plans, more specific to a particular patient rather than an entire general guideline (Miksch, Shahar et al. 1997).

The focus of all three modelling suites is on providing guidance to the health care professional, consider a health care professional and patient scenario, a patient by the name of Jane comes to the health care professional with some ailment, for example high blood cholesterol, the health care professional starts up one of the modelling/decision support suites for example a software system based on the Proforma concept, he/she types into the system the patients symptoms/ailments, and there are automatic operations that are in place which will recommend previously modelled health care procedures (or care plans in the case of Asbru) which will be relevant to the patient, the health care professional will then go through the procedures or care plans that are conveyed by the system executing and prescribing tasks to the patient based on the guidance given, with the prescribed information and tasks given to the patient on paper. The disadvantage of this approach for the patient is that it is difficult for the patient to carry around separate pieces of paper every where, and usually there are large volumes of information related to health advice given to the patient which overtime builds up, hence the patient would not be able to refer to their health information anywhere they like for example at work. The second disadvantage is that the patient misses out on features such as automatic reminders, easy note taking, easy data recording, data visualisation, instant feedback, regular updates of health information, etc which are usually features of wellness systems which carry health information.

The second type of applications which are discussed above include patient wellness applications, again here it is important to realise that there are numerous attempts at implementing patient wellness applications and the examples discussed above were chosen because they represent most of the typical functional characteristics of most wellness management applications for the patient. Taking the same scenario as above with Jane and her GP, if Jane were to be faced with being overweight, then Jane could in theory use the weight wellness monitor, to monitor her health, and to receive advice and learn about how to best manage here weight loss and to indeed maintain it through usage of the wellness monitor, if however she suffers from some other ailment such as diabetes along with her overweight then she will need to use two different wellness monitors, this problem can be solved with the *"My Health My Life"* application, which is a more generic wellness monitoring system which allows various types of health goals to be created and pursued. The advantages of Jane utilising the wellness application (which ever one she uses) is that she can get regular health reminders to monitor her health and perhaps to do health activities, in addition to this it makes it simpler for her to enter her data, and send it to her GP hence facilitating the patient to health care professional communication, furthermore it is much more easier for her to carry her health information with her (however this may not be possible with the *"My Health My Life"* application as it does not have anything that can run on a mobile device yet). One of the major disadvantages of these types of systems is that they often have a weak connection with the health care professional, in that most of the advice that is given to the patient is provided by generic expert systems, in addition to this most wellness management applications seem to be very disease and ailment specific.

Therefore what is needed is a system which combines the best of both worlds, in that it provides a way for health care professionals to easily model health care information, i.e. the form of guidelines or care plans which are specific to patients, and a generic wellness management application onto which the care plans for each patient can be loaded. Hence Jane's GP can design for her a high blood cholesterol management care plan, which when loaded onto the generic wellness management application converts it into a cholesterol wellness management application, giving the patient a full view of their care plan (s), as well as generating reminders and alerts for health activities that they need to do, allowing the recording of data prescribed to them to monitor for example blood glucose sugar, and also allowing them to make notes and enter in other comments and finally synchronise this information with the health care professional. The first attempt at implementing such an integrated system was conducted in 2006 by Abizer Khambati (author) and Sumedh Kanade, through the implementation of a personal health management application, which incorporated the two main components discussed above, in the form of a website for the health care professional which would allow them to model health care plans, for patients, through in built wizards, and then a PDA based application onto which the care plans could be loaded.

The proof of concept prototype that resulted from the project done was effective in demonstrating how the two aspects of health care, i.e. health care modelling for the health care professionals and wellness management for the patient can be combined into a generic system, however during the presentation of the work there were a number of key limitations which arose. For the care plan modelling website one of the main limitations included that there was no real visualisation for the care plans which were modelled and that modelling care plans through wizards may not be the most natural way to model such inherently complex type of information. For the generic wellness management application (or the patient application as it is called above) it was recognised that the application improved on existing wellness management applications by being mostly disease and ailment independent, however one criticism was that the wellness application was only able to run on PDAs (due to being implemented in .Net Compact), and hence if a patient owns anything other than a PDA, or indeed even if the patient owns a PDA with an operating system other than Windows CE the application will not run on it. This limitation is a significant one, as users may own a variety of different mobile devices and a viable solution for a wellness management application should be able to run easily on most devices.

These limitations formed the basis of the current work done for the completion of this thesis, regarding the development of a new integrated health care plan modelling system which builds upon the previous personal health management application, by replacing the wizard based modelling system for the health care plan with a visual language to model the care plans in a more natural and graphical manner, as well as implementing another visual modelling language, which will allow for the modelling of the wellness management application (or the patient application) for different mobile devices (accommodating for the look and feel for different devices) and lastly a generator to generate the wellness management application from these models in Flash which is runnable on most flash enabled mobile and computing devices.

2.6 Mobile Devices and Health Management

One important characteristic most wellness management applications share is that they are increasingly being designed for usage on mobile devices. This includes wellness management applications which are designed for PDAs, smart phones, mobile phones, and perhaps even a whole range of upcoming portable multimedia players. The following section describes the motivation behind the usage of mobile devices for health care management, and some of the technologies which were researched for the development or modelling of such applications.

2.6.1 Motivation for using Mobile Devices

As explained above most wellness management applications are designed for patients suffering from long term chronic illnesses, the main reason for this is that such patients need to monitor and manage their health on their own for large periods of time, and studies have shown that solutions such as wellness management applications which promote a more automated monitoring process through reminders and easy data entry are likely to improve noticeably the treatment outcomes of the therapy provided to patients suffering from major chronic illnesses such as diabetes and high blood cholesterol (Kollmann, Hayn et al. 2006).

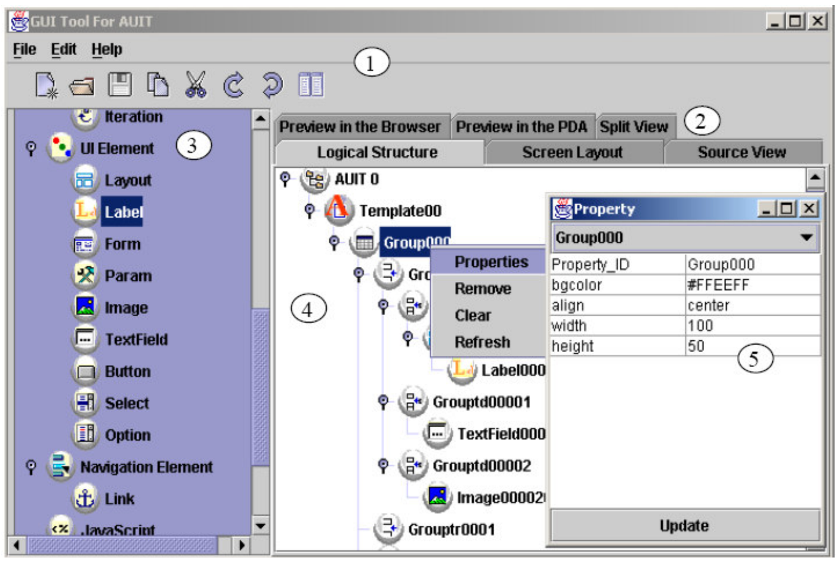
Taking this need into consideration, it is important to realise that any monitoring or management solution designed for the patient must be viable for the patient in most environments that they work in, for example a patient may in one day be present in three different environments, for example in the morning the patient is in a home environment, and in the middle of the day the patient is in a work environment, and in the evening a patient may be in a party/friendly/entertaining environment, hence any solution which is designed for the patient must be viable in all these environments. This means that the terminal (i.e. device or medium) that the wellness (or patient) application is hosted on must be convenient to carry around, easy to use, cheap enough to buy and also reliable. These characteristics make mobile devices an ideal choice for deployment of health care plans (or wellness management applications). In addition to this the usage of mobile devices allow patients to carry out management processes such as receiving reminders, and entering data far more discretely and unobtrusively than carrying a diary or using a PC. Furthermore communication and synchronisation of patient data to remote health servers is also easily achieved through the usage of for example WAP for mobile phones (Kollmann, Hayn et al. 2006).

2.6.2 Technologies for developing Mobile Devices

As mentioned above in the discussions section, for health care management applications based on mobile devices (or any computer based devices) to be successful it needs to be ensured that the solution is adaptable to different types of mobile devices as well as perhaps be adaptable to different platforms. One of the most important parts of the application that needs to be adapted includes the user interface of the application, the reason for this is that different devices have different characteristics, i.e. different screen sizes, different GUI capabilities etc, and hence an application designed for a PDA will not be suitable for a mobile phone, likewise a solution designed for a desktop will not be suitable to a PDA. For a mobile application, the GUI is not the only part of it which could be incompatible for different devices, if the application is a client server application then the GUI is the only concern, however for more fatter clients, platform incompatibilities may also arise, for example a fat client application designed for a windows operating system based PDA needs to adapt its GUI when it moves to another PDA which also has a windows operating system, however if the application needs to be transferred to a Palm OS PDA then not only the GUI needs adapting but the entire code for the application needs to be adapted. The following sections describe some of the solutions which are typical of the type of solutions that are out there for adaptive user interfaces, the research described below is by no means a systematic review of the literature available in this area, but simply a collection of approaches which were useful for the inception of the visual patient application modeller created as part of this research to allow the modelling of a device independent generic wellness management application (or as it is later referred to as a patient application).

2.6.2.1 An environment for developing adaptive multi-device user interfaces - (Grundy and Yang 2003)

This particular solution for multi device user interfaces involved the development of an integrated system through which device independent definitions of the graphical user interface of an application can be designed, in addition to this the system allows designers to specify how certain graphical user interface elements were appropriate (or in appropriate) to certain user roles and user tasks. The application works by allowing designers to model generic graphical user interfaces through three different view types, the first being a “logical view”, then a generic screen “layout view” and finally a “textual source view” (Grundy and Yang 2003). The logical view allows the designer to view a hierarchical tree of GUI components where by they can clearly see which GUI components are laid out in which other components and specify properties of these components through a dialog based menu (Grundy and Yang 2003). In contrast to this the designer may also choose to work in a screen layout view, this view being different from the majority of WYSIWYG editors, displays a generic “tiled” representation of the GUI components and their layout as can be seen in figure 8, rather than a device specific preview, the application however does provide for device and platform specific previews to be displayed of the generic GUI model, such as a preview of how the generic GUI will be represented in IE, or on a mobile phone or perhaps on a PDA (Grundy and Yang 2003). The third type of editing view is a textual view, which represents the generic GUI model in the form of an XML document, allowing the designer to hand code some of the finer details which maybe easier to specify directly through the source view, for example the Java beans that the interface will employ (Grundy and Yang 2003). The actual model of the GUI is stored in XML form and this model is utilized to automatically generate the implementation of the GUI. The implementation of the GUI is generated in the form of JSP pages which have been extended by a custom library called "AUIT (Adaptive User Interface Technology)" (Grundy and Yang 2003). These advanced JSP pages hold all the information about layout, GUI components and task and user role information related to the GUI, and hence when the JSP pages are accessed they will firstly investigate the characteristics of the type of device for which the mark-up is needed to be generated, i.e. items of information such as who the users are, what are the sort of tasks that they are expected to do, what is the size of the screen, what is the GUI capability, etc, and using this information as well as the information the JSP page itself contains modelled by the designer, the JSP can then produce the right markup for the device (Grundy and Yang 2003).



Property_ID	Group000
bgcolor	#FFEEFF
align	center
width	100
height	50

Figure 7: The application for modelling the adaptive GUI showing the logical view active reproduced from (Grundy and Yang 2003)

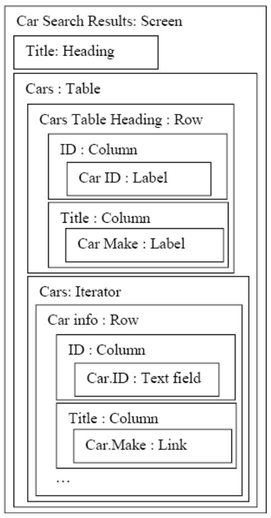


Figure 8: The tile layout showing the generic user interface reproduced from (Grundy and Yang 2003).

2.6.2.2 Adaptive User Interfaces through Mobile Agent Technology - (Mitrovic and Mena 2002)

Another approach to developing adaptive user interfaces was proposed through the utilization of mobile agent technology. Researchers at the University of Zaragosa, Maria de Luna (Nikola Mitrovic and Eduardo Mena) designed an approach through which user interfaces can be automatically adapted to different types of devices through the usage of mobile agents and standard graphical user interface definitions. The approach designed by the researchers allows generic user interface definitions to be encoded and stored in XUL (XML user interface language), which the mobile agent transforms into the GUI most suitable to the device for which it is responsible (Mitrovic and Mena 2002). The mobile agent was implemented utilising the Grasshopper libraries as well as the JXUL interpreter responsible for reading, interpreting and converting the XUL definitions into Swing user interface implementations (Mitrovic and Mena 2002).

One limitation of this which could be possible is that the mobile agent was implemented in Java hence it may not be viable for other devices which may not have java capability such as some WAP phones, or other devices which cannot for some reason host mobile agents. The researchers however accommodated for this limitation by designing the mobile agent in such a way that if device can connect to the internet, and support mobile agents (i.e. is Java enabled), then the mobile agent will go onto the device and render the application user interface in swing, however if the device cannot support mobile agents for example on a non-java enabled WAP mobile phone, then the mobile agent will not be downloaded onto the phone, instead the agent will act as a content provider and provide the WML mark-up for the user interface for the application to the phone, similarly it can also provide the HTML (Mitrovic and Mena 2002). For these transformations the mobile agent utilises XSL transformations as oppose to the JXUL middleware libraries for the swing interface. The functionality for structuring and laying out the GUI components for the user interface from the XUL definitions is coded entirely into the mobile agent, with the transformations handled by JXUL and XSL (Mitrovic and Mena 2002).

2.6.2.3 Dynamic User Interface generation for Web services - (He and Yen 2007)

This particular approach has been specifically designed for easily providing web services with an automatically generated user interface based upon models of the web service augmented with models of the interface preferences. The approach introduces a framework for the generation of user interfaces for web services for multiple types of devices from standard PC and desktop interfaces to mobile phones and PDAs. The structure of the framework has four main layers, each layer implemented as a web service of its own. These layers include (He and Yen 2007):

1. A “pre-interface processing service”,
2. A “user interface element processing service”,
3. An “interface structuring service” and finally
4. A “user interface structure generation service”.

The entry into the framework starts at the “pre-interface processing service”, this service takes into as its input one compulsory item, this is the WSDL definition of the web service, this describes to the “pre-interface processing service” the inputs that the web service needs from the user and hence it can generate the necessary GUI elements. This however may be in-sufficient to produce optimum user interface results hence the designer of the web service has an option to enter in other preference information into the service, this extra information may include one or more of the following (He and Yen 2007):

1. An “interface style” specification
2. An “interface option” specification
3. An “interface semantic” specification &
4. An “interface structure” specification

Through the first two options, i.e. the “interface style” and the “interface option” specification allows a designer to specify how the individual graphical elements that compose the entire GUI of the web service can be laid out, in addition to this the designer can use the semantic specifications to give the GUI elements extra meaning which may be useful to the generator in deciding how to present the GUI elements, lastly the designer may also want to specify the general structure of the GUI elements with respect to one another, most commonly this is in the form of a hierarchical structure (similar to the logical view in the first approach discussed in this section) (He and Yen 2007). This information still does not specify any user oriented specifications, such as the users preferences or the characteristics of the type of computer device that the user will be using hence, the “pre-interface processing service” also takes in a detailed profile of the user, which includes such type of information (He and Yen 2007). The function of the pre-interface processing service is simply to combine all the different specifications into one. This combine specification is then passed to the next service, i.e. the “user interface element processing service”, here the specification is scanned for the different types of GUI elements that are present and each element is converted into the language of choice (which is determined by the user profile, if the user is using mobile phones then WML, else desktops, then HTML, etc), in addition to this other preferences and characteristics specified for the elements in the “interface style” specification part are also applied here (He and Yen 2007). After all the individual GUI elements are generated the “interface structuring service”, decides the layout of the final GUI interface based on the device characteristics specified in the user profile part of the specification. The final service, i.e. the “interface structure generation service” actually generates and puts together the entire GUI interface for the web service based on the layout decided upon (He and Yen 2007).

2.6.2.4 Discussions – Lessons Learned

As mentioned above the solutions/approaches that are discussed above are by no means an exhaustive systematic review; however they more or less represent the typical types of strategies employed for adaptive user interfaces for different computing devices and there are some important lessons learned from them.

The most important lesson learned from the approaches described above includes the idea of a model driven solution to adapting user interfaces. From all the approaches discussed above it can be seen that all of them during some stage or the other encode generic definitions or models of the graphical user interface, specifying the basic anatomy of the GUI that is to be maintained across devices and platforms. In the case of the first approach shown this generic model is encoded through the many views that are possible through the system and the model is stored in XML, in the case of the second approach this model is encoded in XUL, and the final approach discussed the model of the GUI required is not explicitly created however it is implicitly recorded in the WSDL definition of the web service.

The second lesson that is learned is that there must always be a way in which the user preferences, including the characteristics of the intended device and platform can be either inputted or otherwise determined. In the first approach this information is automatically determined when the user accesses the AUIT extended JSP, in the second approach the logic for automatically detecting device and platform information is pre-coded into the mobile agent responsible for generating the GUI, and in the final approach this information is specified in the form of a user profile.

Finally, the user profile/information along with device characteristics, as well as the generic model of the GUI is used by custom generators either java based or XSLT rule based to generate the actual GUI or mark-up for the GUI. In addition to this there may be many other inputs of interface layout, style and structuring which may be applied by a designer in order to improve the design of the user interface. The general approach of systems for designing adaptive user interface applications is as shown in the figure below. The challenge presented hence is to apply the concepts learned from these solutions to designing a specialised approach specific to health care planning and wellness management through which a personalized patient application (or wellness management application) can be designed so that it can work on a variety of mobile and otherwise computerised devices that the patient may posses.

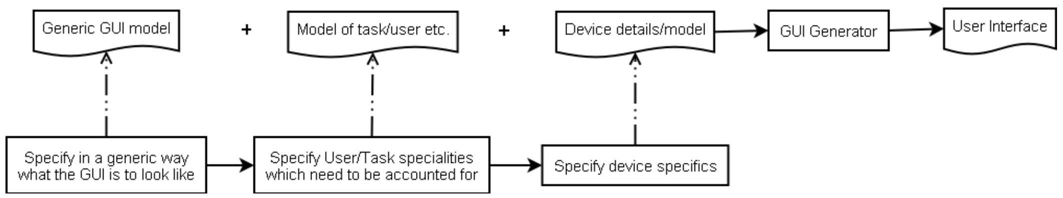

Generic GUI model
+
Model of task/user etc.
+
Device details/model
GUI Generator
User Interface
Specify in a generic way what the GUI is to look like
Specify User/Task specialities which need to be accounted for
Specify device specifics

Figure 9: The typical approach for applications that allow development of adaptive user interfaces.

2.7 Summary

The purpose of this chapter was to describe the main areas of background research that was done and the elements of discussion that arise from it. There are three main areas of background research done, the first category involved researching applications and frameworks designed for modelling and capturing health care information, the three examples of applications discussed include GLIF, a framework for capturing the logic of clinical guidelines, PROforma another framework focussing on capturing different clinical guidelines as well as operations which can be executed on the guidelines, lastly there was ASBRU, a system which focussed on capturing patient care strategies in the form of time oriented care plans. The second category of background research which was conducted included applications for managing patient health. These included wellness management applications, such as the wireless weight wellness application to manage and promote weight loss, the caloric management application again aimed at weight loss, and a generic web based wellness management application called My Health, My Life, aimed as a general solution for managing and monitoring a wide range of chronic diseases. Both types of applications have limitations of their own, the health care modelling suites are good at helping health care professionals model and use complex information, but do not support providing this information to patients in an efficient manner, well as the wellness management applications designed for patients, largely do not include a role for the health care professionals to participate in and are generally very disease specific.

Research done by Abizer Khambati and Sumedh Kanade, saw the development of a system which included a website which had a wizard based approach to designing health care plans and then providing these health care plans to patients on patient applications (or wellness management applications). This solution although combining both types of approaches of modelling health care information as well as distributing it to the patient in an appropriate manner suffered from limitations. These limitations included criticism that wizards were ineffective and an unnatural way of modelling complex health care information, as well as this the PDA based wellness management application was too device and platform specific.

Other background research done included the different approaches to developing adaptive user interfaces, and from the three solutions discussed it can be seen that the typical approach can be split up into four sections, firstly there is the modelling of a generic user interface model, then the specification of user preferences and profile, then details on the type of device it is to run on, and finally the generator which combines all this information and generates the user interface.

Chapter 3 - Overview of approach

3.1 Introduction

This chapter provides an overview of the overall approach of the research work done for this thesis. The main aim of the research is to explore a model-driven approach to designing and generating usable health care plan applications for patients suffering from chronic (long term) illnesses. This approach can be envisioned as a cyclic process, where in the first step a health care professional designs a generic health care plan template for the management of a particular ailment, next this template is instantiated for a particular patient making the template specific to the patient's needs, after which the health care professional can generate a software application by combining the instance of the care plan template designed for the patient, along with a set of generic application and GUI definitions that have been designed by an IT professional. The generated software application which now includes patient care plan can be loaded onto electronic devices used by the patient, such as mobile phones, PDAs, and desktop computers. The care plan application then provides continuous aid to the patient in the form of not only reminders to do health care activities, but providing items of education, and instructions important to maintaining their health.

As part of this research project a proof of concept system was designed that implemented all the steps of the cycle described above. The first part of the system was a domain specific visual language that allowed the implementation of the generic health care plan template. To support the instantiation of a care plan template for the patient a supporting tool was implemented which allowed the health care plan template to be personalized to the patient. Next another visual language was implemented that allows an IT professional to specify generic GUI definitions for the visualization of the care plan instance. Finally, a generator was written that combined the care plan visualization definitions along with the care plan instances to produce a software application for the patient.

The following sections provide a high-level view of each of these care plan system parts in the context of a scenario for a patient suffering from obesity. The different stages in the care plan modelling cycle are illustrated with reference to a fictional scenario of a patient suffering from obesity, the scenario being built on an actual guideline for treating obesity. The same scenario will be extended with more detail for use in later chapters.

3.2 Stages of Chronic Illness Management

Chronic illnesses are typically illnesses that last for a long period of time, from anything from a month to an entire lifetime. Chronic illnesses are sometimes a cause of bad lifestyle habits such as chronic smoking, eating disorders, alcoholism etc, and sometimes they are simply inherent of a patient's physiology, such as asthma, diabetes, and cholesterol problems. In general the treatment and management of a chronically ill person can be seen as five stages (Schimoda 2002):

1. "Pre-contemplation" – At this stage the patient is clearly facing the problems caused by the illness but does not yet know that they have the illness.
2. "Contemplation" – At this stage the individual knows that they have a problem and are thinking about getting help.
3. "Preparation" – At this stage the individual has decided a definite time frame in which to get help, and is preparing themselves mentally and physically for this change.
4. "Action" – In this stage the individual actually gets help and is actively working towards the treatment of their ailment.
5. "Maintenance" – This stage sometimes lasts an entire lifetime, where the patient maintains their health, sticking to the necessary health regimes, practices and lifestyle habits to keep their ailment in check.

It is important to recognize that the proof of concept health care plan modelling system that was built as part of this research only caters for the last two stages of the chronic health care management process, i.e. the action and maintenance stages.

3.3 A Chronic Illness Scenario

The fictional character in this scenario is John Doe who is a 35 year old software developer who works for a well known software company; he has a wife and two children and lives in a three bedroom apartment in the city, the medical facts in this scenario have been taken from the obesity management treatment guideline (National Institute Of Health 1998).

John suffers from class 1 obesity; this means that his body mass index (BMI) is over 30 (National Institute Of Health 1998). John's condition has many effects on his lifestyle, prime among them is low self esteem due to his appearance, in addition to this John also realizes that his condition makes it very difficult to spend time with his children, including such activities as participating in sports, and attending extracurricular activities with his children.

Due to these reasons and many more John has contemplated for a long time to do something about his condition, and finally has prepared himself to go see his GP to get the help that he needs.

3.4 Care Plan Modelling cycle overview

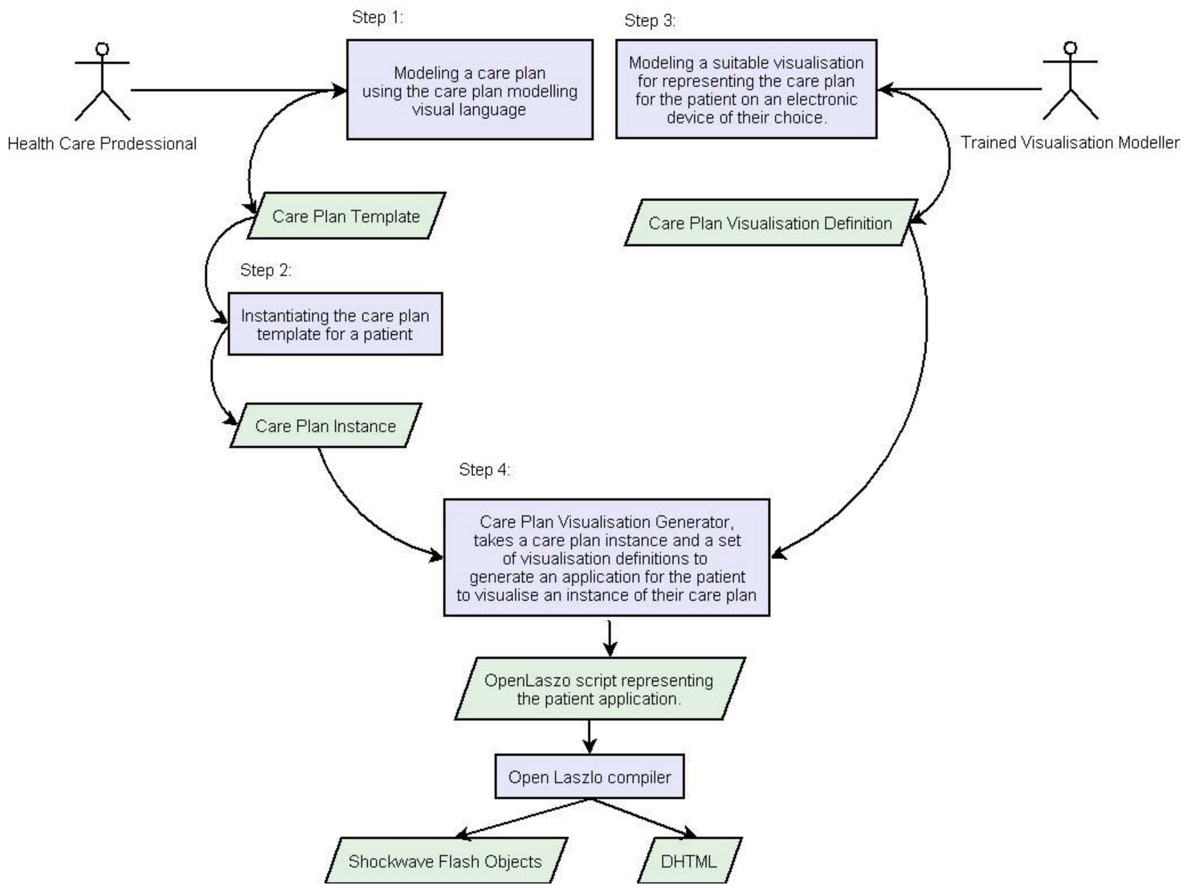
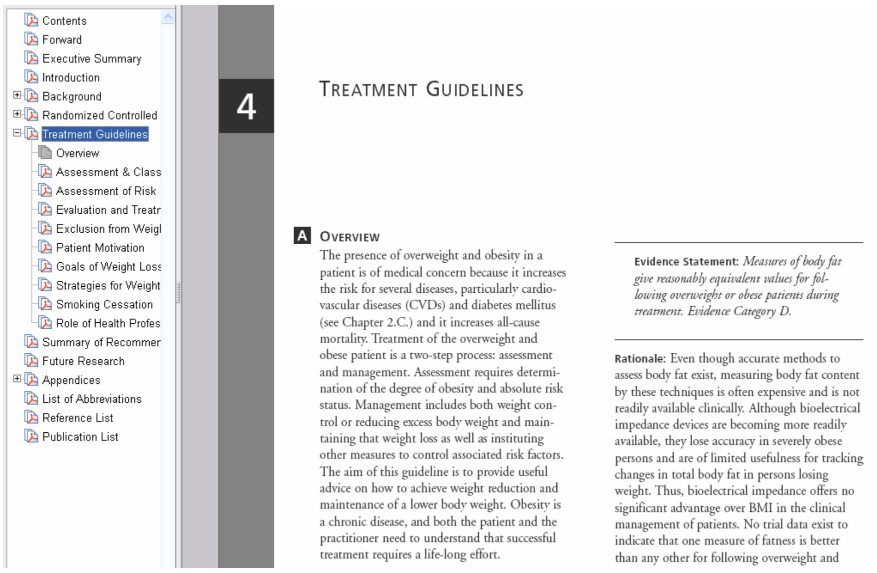

Step 1:
Modeling a care plan using the care plan modelling visual language
Step 3:
Modeling a suitable visualisation for representing the care plan for the patient on an electronic device of their choice.
Health Care Prodessional
Trained Visualisation Modeller
Care Plan Template
Care Plan Visualisation Definition
Step 2:
Instantiating the care plan template for a patient
Care Plan Instance
Step 4:
Care Plan Visualisation Generator, takes a care plan instance and a set of visualisation definitions to generate an application for the patient to visualise an instance of their care plan
OpenLaszo script representing the patient application.
Open Laszlo compiler
Shockwave Flash Objects
DHTML

Figure 10: The *"model driven engineering"* approach to care plan modelling.

The care plan modelling cycle show in figure 10 describes the stages that the health care professional goes through using the care plan modelling system, to model, instantiate and deploy a care plan for John, in order to get John's obesity problem under control. The subsequent sub-sections in this section look at each step in the cycle in more detail explaining how they were implemented.

3.4.1 Step 1 – Modelling a care plan for a patient

When John, first visits his health care professional, during their consultation the health care professional assess John's condition, identifying the main problem areas, and also identifying the action that needs to be taken to bring John's health back on line. In order to plan for the different lines of action, the health care professional may review the guideline for treating patient's suffering from obesity as shown in figure 11 below. After this the health care professional decides to design an action plan for John. This is the first stage in the cyclic process represented by the care plan modelling system.



4

TREATMENT GUIDELINES

A OVERVIEW

The presence of overweight and obesity in a patient is of medical concern because it increases the risk for several diseases, particularly cardiovascular diseases (CVDs) and diabetes mellitus (see Chapter 2.C.) and it increases all-cause mortality. Treatment of the overweight and obese patient is a two-step process: assessment and management. Assessment requires determination of the degree of obesity and absolute risk status. Management includes both weight control or reducing excess body weight and maintaining that weight loss as well as instituting other measures to control associated risk factors. The aim of this guideline is to provide useful advice on how to achieve weight reduction and maintenance of a lower body weight. Obesity is a chronic disease, and both the patient and the practitioner need to understand that successful treatment requires a life-long effort.

Evidence Statement: *Measures of body fat give reasonably equivalent values for following overweight or obese patients during treatment. Evidence Category D.*

Rationale: Even though accurate methods to assess body fat exist, measuring body fat content by these techniques is often expensive and is not readily available clinically. Although bioelectrical impedance devices are becoming more readily available, they lose accuracy in severely obese persons and are of limited usefulness for tracking changes in total body fat in persons losing weight. Thus, bioelectrical impedance offers no significant advantage over BMI in the clinical management of patients. No trial data exist to indicate that one measure of fatness is better than any other for following overweight and

Figure 11: Screenshot of the treatment guideline for obesity, a complete 262 page clinical guideline taken from (National Institute Of Health 1998).

The first part of the care plan modelling system is a domain specific care plan modelling language. The care plan modelling language was implemented in the Marama meta-tool software and represents a vocabulary that can be used to design care plans for patient's suffering from chronic illnesses, in a way that reflects guidelines to treat them. Here the health care professional can set up things like health care activities of different types, performance goals, instructional content, assessment regimes, review activities etc.

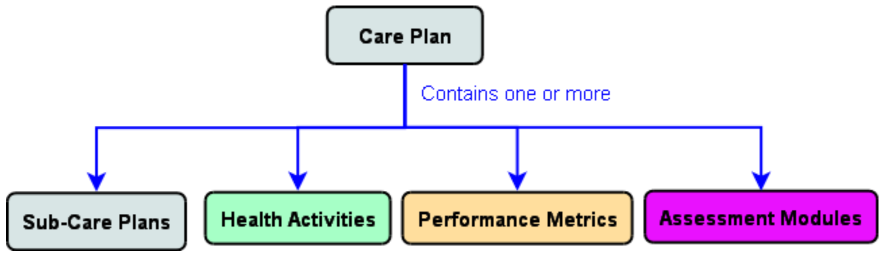


Figure 12: The general structure of the care plan, which contains other care plans, health care activities, performance metrics and assessment modules.

The resulting care plan, which in this case will be specific to obesity management is essentially a template for the management, monitoring and assessment of obesity, although the health care professional most likely has the patient in mind when creating/modelling the contents of the care plan, it can still be reused for other patient's with the same ailment, an example of an obesity management care plan can be seen in figure 13 where the obesity management care plan is composed of three sub care plans, an obesity treatment algorithm (an assessment module) and two performance metrics, the obesity management care plan shown in figure 13 was constructed by referring to the common treatment procedures outlined in the obesity management treatment guideline (National Institute Of Health 1998).

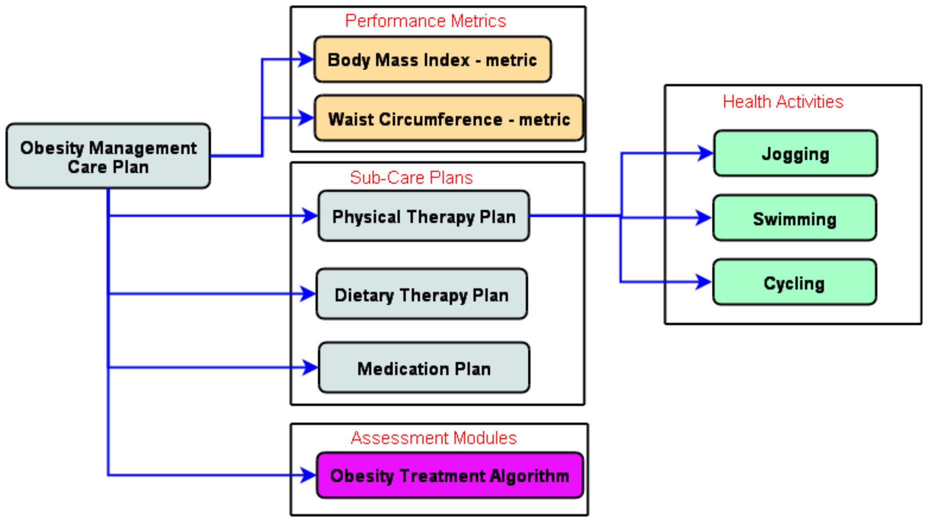


Figure 13: The sample of a care plan template graphically represented here, which pertains to an obesity management care plan, with three sub care plans, namely the physical therapy care plan, the dietary therapy care plan, and the medication care plan.

3.4.2 Step 2 – Instantiating a care plan template for a patient

The instantiation process for a care plan is essentially the stage where a generic health care plan template is made specific to the needs of the patient. The most common items of the care plan template that may be further specified for a patient are the timings and routines for the care plan, this includes specifying or modifying the necessary activity routines, activity instructions, and activity timetables to suit the patients own life and own routines, for example the general schedule and instructions for the jogging activity can be seen in figure 14 below. During this stage the health care professional may also feel that the care plan template needs some other element that is specialized for the needs of the patient, in which case the health care professional my jump back go to stage 1 and remodel some aspects of the care plan.

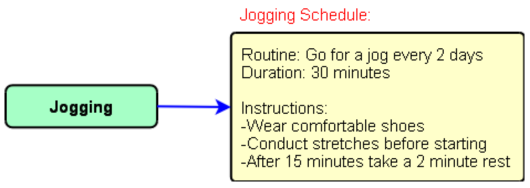
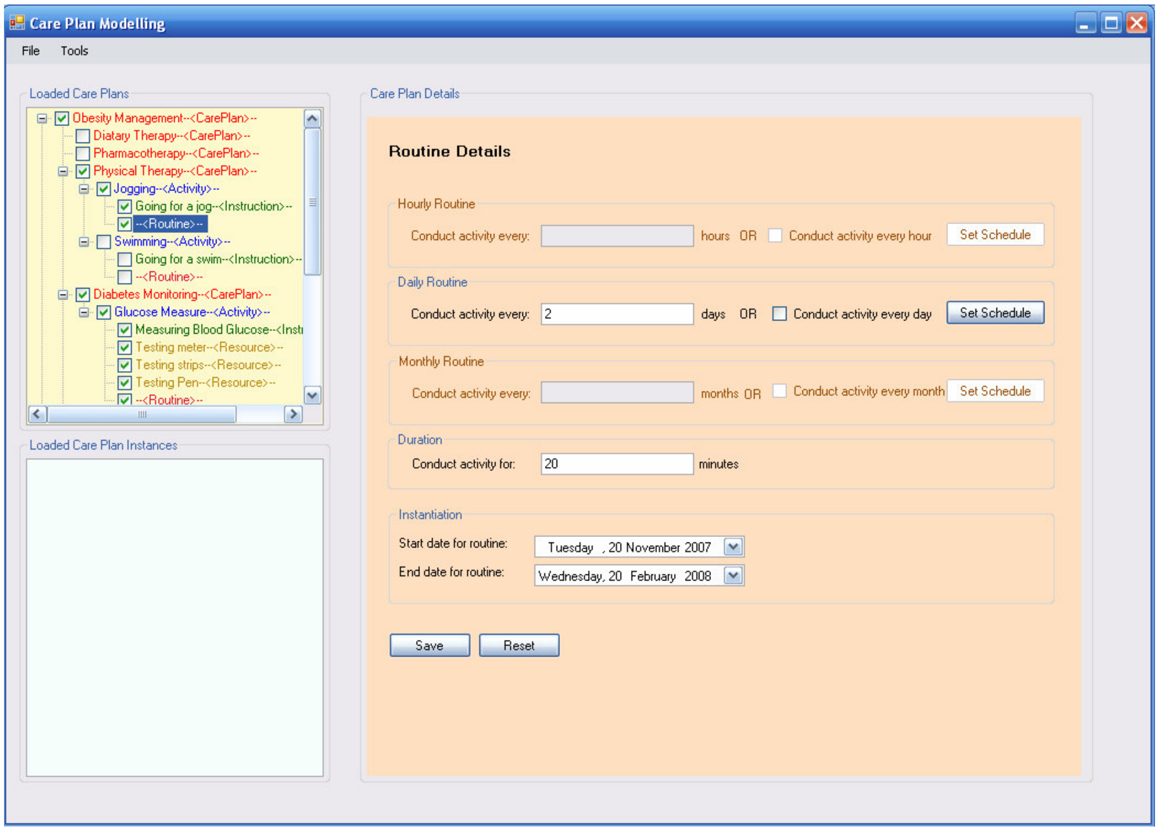

Jogging Schedule:
Jogging
Routine: Go for a jog every 2 days
Duration: 30 minutes
Instructions:
-Wear comfortable shoes
-Conduct stretches before starting
-After 15 minutes take a 2 minute rest

Figure 14: Sample health activity with a routine as well as a set of instructions.

In order to support the instantiation of a care plan template for a patient, a desktop application was designed to exhibit the functionality that was needed. Once the health care professional has modelled the health care plan template using the domain specific modelling language, then the care plan template can be loaded as an XML file into the desktop application, where the health care professional can view the care plan templates, can specify different routines, and timetables for the activities, and also modify items of the care plan template from the desktop application itself. Referring back to John, in the scenario above, after John's GP has modelled the care plan template for him (or used one he has modelled for another patient suffering from obesity), he loads the care plan template XML file into his desktop application, and then starts setting up the routines and timetables for the activities in the care plan to suit John's heavy work, and family responsibilities. Finally the GP sets a time frame for John's care plan of 3 months (the period over which the care plan will be active for John), the sample instantiation application with the jogging activity routine loaded can be seen in figure 15 below. The now instantiated care plan can be exported as an XML document.


Care Plan Modelling

File Tools

Loaded Care Plans

- ✓ Obesity Management--<CarePlan>--
  - ☐ Diatary Therapy--<CarePlan>--
  - ☐ Pharmacotherapy--<CarePlan>--
  - ✓ Physical Therapy--<CarePlan>--
    - ✓ Jogging--<Activity>--
      - ✓ Going for a jog--<Instruction>--
      - ✓ --<Routine>--
    - ☐ Swimming--<Activity>--
      - ☐ Going for a swim--<Instruction>--
      - ☐ --<Routine>--
  - ✓ Diabetes Monitoring--<CarePlan>--
    - ✓ Glucose Measure--<Activity>--
      - ✓ Measuring Blood Glucose--<Instı
      - ✓ Testing meter--<Resource>--
      - ✓ Testing strips--<Resource>--
      - ✓ Testing Pen--<Resource>--
      - ✓ --<Routine>--

Loaded Care Plan Instances

Care Plan Details

Routine Details

Hourly Routine
Conduct activity every: hours OR ☐ Conduct activity every hour [Set Schedule]

Daily Routine
Conduct activity every: 2 days OR ☐ Conduct activity every day [Set Schedule]

Monthly Routine
Conduct activity every: months OR ☐ Conduct activity every month [Set Schedule]

Duration
Conduct activity for: 20 minutes

Instantiation
Start date for routine: Tuesday , 20 November 2007
End date for routine: Wednesday, 20 February 2008

[Save] [Reset]

Figure 15: The instantiation application with the jogging activity routine loaded, showing the activity interval of jogging every 2 days, for 20 minute duration as well as the activity start and end date spanning 3 months.

3.4.3 Step 3 – Modelling an application to visualise a care plan instance for a patient

A care plan instance on its own as an XML file is of little use to the patient, therefore the next step is to visualize the instance of the care plan in a way that will be most useful to the patient. This means allowing the patient to see their care plan in all it's entirety, allowing them to see the different modules the care plan is made up of, including the health activities that have been prescribed for them, related instructions, performance goals and generally anything that is part of their care plan, including and not restricted to receiving health reminders where necessary.

In addition to visualizing the care plan, it is also important that the visualization is not dependent on only one device type, in that the care plan instance for a patient should be able to be visualized for most common type of electronic devices that are used, for example desktop computers, PDAs, even mobile phones. Although a web-oriented approach is often one which is suggested for bridging the different device gap, this may be inappropriate for mobile devices for which the most common architecture is partially connected, suggesting the need for an application that resembles more a desktop application with capabilities to synchronizing with remote servers.

One side effect of being able to run an application on different devices is that the graphical user interface needs to be adapted to the different sizes and quality of the display devices.

The approach taken for meeting the needs of the above mentioned patient application was to design another domain specific language (DSL) for the modelling of the visualization of a care plan instance for a patient. The DSL was once again implemented using the Marama meta-tool software. The target audience for this DSL was not the health care professional but rather technical minded IT personnel, who would consult with the health care professional on the structure of the care plan instance and perhaps consult with the patient on the most appropriate type of visualization for their care plan and then design different visualization definitions using the modelling language. Off course in practice a lot of the standard visualization definitions could be produced upfront and stored.

For example, John in the scenario described above primarily uses his desktop at work, and uses a smart-phone/PDA at work and in his personal life; therefore John decides to visualize his care plan on his smart-phone. Therefore the most appropriate visualization definition for him would be one which is meant for a PDA or even a mobile phone. His device may have certain peculiar features such as not being able to display windows in tabs, or that the screen is smaller than most PDAs and can't display long list, therefore the visualization definition can be changed to include these peculiarities.

3.4.4 Step 4 – Generating the patient application from a visualisation definition and a care plan instance

Once a care plan has been instantiated for a patient and an appropriate visualization definition has been chosen for their device type (this can be a standard one that has already been modelled by a trained IT personnel or one that has been slightly tweaked) then the next step is to actually generate the application by combining the care plan instance and the appropriate visualization definitions.

Through an evaluation of the various technologies out there for different devices it was decided that Open Laszlo would be used, programs written in this language can be compiled to either flash or DHTML. Therefore a generator was written that interpreted the visualization definitions to produce an application for the visualization of the care plan instance for a patient.

In the example scenario, once an appropriate visualization definition has been defined for John's smart-phone/PDA then the application can be fed into the generator which will produce the Open Laszlo code for the patient application based on the visualization definitions, this then compiled to flash becomes an application that is viable for running on most devices, which includes not only desktop applications but also PDAs, mobile phones and perhaps even devices such as IPODs and MP4 players that are flash capable, an example of the Flash application compiled from the OpenLaszlo code can be seen in figure 16 below .

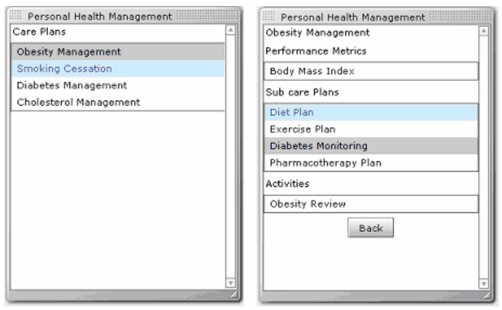


Figure 16: An example of the patient application interface in Flash as compiled from the Open Laszlo program code.

3.5 Summary

The care plan modelling system has been designed to support patients suffering from long term or chronic illnesses and the health care professionals that care for them. The treatment stages for treating a patient suffering from a chronic illness ranges includes pre-contemplation, contemplation, preparation, action and maintenance. The care plan modelling system is only concerned with the last two stages. The care plan modelling system consist of four main components, the first component is a visual care plan modelling language, through this component the health care professional can model the action component of the treatment for the patient in the form of care plans. A health care professional can model many care plans for many different types of conditions and make the care plans sufficiently generic so that they are applicable to many patients suffering from the same ailment. The second component in the care plan modelling system is the care plan instantiation tool, this tool can be used to load and fine tailor the care plan templates generated through the care plan modelling visual language, this step is to ensure that the generic care plan is made specific to the personal needs of the patient before it is deployed to the patient, hence forming a personalised care plan instance for the patient. The next step is to suitably represent this care plan instance to the patient. This is done in the way of specialised patient application which can run on an electronic device such as a patient's mobile phone or PDA. The third component of the care plan modelling system is the visual patient application modelling language, through this a visual GUI designer can model how a typical care plan instance can be represented, modelling the look and feel for the patient application for different types of devices such as mobile phones, PDAs, etc. The fourth and final component of the care plan modelling system is the patient application generator, the purpose of this component is to take in a model of the look and feel of a patient application and actually generate the patient application. Once the patient application is generated for the patient for their favourite device then the care plan instance tailored for them can be loaded onto the application allowing the patient to view and execute their care plan. At the end of the thesis a brief evaluation was conducted, this evaluation consisted of a cognitive dimensions work through of the two visual languages i.e. the visual care plan modelling language and the visual patient application modelling language, followed by a brief informal evaluation of the two languages by experts in the domain of visual language development.

Chapter 4 - User Needs

4.1 Introduction

The purpose of this chapter is to introduce the needs of the two main users of a care plan modelling system, the first and most obvious users of the care plan modelling system are the health care professional, and the second set of users being the patients that are cared for by the health care professionals. The care plan modelling system is primarily aimed at providing support for modelling health care plans for patients suffering from long term chronic illnesses. Each user group of the care plan modelling system has its own needs from the system, for the patient user group this need corresponds to the need to adhere to the care plans designed for them hence ensuring the best health outcomes. For the health care professionals these needs correspond to the need to efficiently model health care information and to store it in a digital and formal model syntax where by it can be processed in various ways through the computer, reused, and collaborated on by multiple health care professionals.

4.2 Users of the system

There are two central stakeholders in the chronic health care paradigm, and each stakeholder has their own needs regarding a health care modelling system. These two stakeholders include the patient and the health care professional. The patients are people who suffer from some long term ailment, where the ailment can be anything from a lifetime illness such as diabetes, or a bad lifestyle habit such as smoking. The health care professionals include professionals such as general practitioners (GP), physiotherapist, dieticians etc, these are people who play a central role in designing health care information and advice for the patient, helping patients manage their health and monitoring the patient's health. What follows is a detailed discussion of the needs of each of the stakeholders with regard to the care plan modelling system, these needs are discussed with regard to an example of chronic health care which includes firstly a patient, John suffering from obesity and the GP looking after him, this example is further extended and elaborated on in the following chapter in order to explain the care plan domain model.

4.3 Patient Needs

As explained above, the patients that the care plan modelling system is aimed at includes patients who suffer from long term chronic illnesses. Examples of these conditions include, but are not restricted to ailments such as diabetes, obesity, high blood cholesterol, smoking, overt alcoholism, etc. Patients suffering from these types of ailments usually require treatment, monitoring and support for managing their condition for most of their lifetime. There are various types of support that patients suffering from chronic illnesses may need from a prospective care plan modelling system, such as support for efficiently managing the vast volumes of health care information that they receive, support for learning about their condition, support for communicating with the health care professional etc (Pratt, Unruh et al. 2006). The most significant problem however for patients suffering from chronic illnesses is adhering to the health care plans that are designed for them, whether this disease is diabetes, smoking, HIV or any other lifetime ailment (Funnel, Anderson et al. 2000; Bartlett 2002), and it is this problem that the care plan modelling system focuses on. Other issues such as health information management for patients are still very important issues to consider, however due to a constraint in time for the research done; this is out of the scope of this project.

There are many reasons for which a patient may be non adherent to the health care plan prescribed to them by their GP (or any other health care professional) and what follows is the discussion of some of the most important aspects of patient adherence with respect to the needs of a patient from a care plan modelling system.

4.3.1 Patient adherence

Patient adherence or compliance to the health care plans that are prescribed for them is essential to improving and maintaining their health (Funnel, Anderson et al. 2000; Turner and Hecht 2001; Bartlett 2002). In order for John, in the our example, to reduce his weight and overcome his obesity and indeed to maintain the achieved results, he has to follow all the treatment prescribed to him, keeping up with all the health activities that make up his diet therapy, pharmacotherapy etc.

Non-compliance to treatment regimes can have different effects depending on the characteristic of the disease. It is easy to see why patients are much more compliant to treatment of diseases that are for the short-term, there could indeed be many reasons for this, however an obvious reason can be observed as the transient treatment period for the disease, in that the result of treating the disease can be seen soon and sometimes immediately, and also the patient has to put up with the treatment for a short period of time.

Maintaining compliance to treatment regimes for patients suffering from long-term, to lifetime lasting chronic illnesses is much more complicated. Examples of such conditions include diabetes, obesity, and of-course high blood cholesterol. Chronic ailments are not necessarily illnesses they could well be harmful habits such as smoking and overt alcoholism. What most chronic ailments have in common is that they require the patient to self-manage and self-monitor the health care plans that are prescribed to them for long periods of time, with intermittent consultations with the health care professional. It is during this period of self-management that most patients fall into non-compliance to the treatment regimes prescribed in their health care plans.

There are many reasons for the non-compliance of patients to their health care plans, however most researchers agree that the main factors can be summarised as follows (Bartlett 2002):

1. Forgetfulness of conducting the necessary health activities.
2. Having poor education on their condition and their treatment.
3. Having health care plans that are poorly suited to their lifestyle.
4. Side-effects to prescribed health activities.
5. Lack of patient adherence/compliance monitoring

4.3.1.1 Forgetfulness of conducting health activities

Research has shown that whenever patients are asked to rank the cause of their non-compliance to health care regimes, the highest rank is given to forgetting to conduct health activities on the right time, due to being busy (Bartlett 2002). In an investigation done of patients suffering HIV, a chronic illness that requires treatment and maintenance till death, 66% of patients interviewed for reasons of non-compliance to their treatment reported forgetfulness as the reason for their non-compliance (Bartlett 2002). Relating back to the obesity management example it can be envisioned how easily health care activities as simple as going for a jog or taking medicine on the right time can be forgotten by John, who may spend most of his time in meetings or in front of a computer, working, and by the time he remembers to go for that walk or take the medicine the time is no longer right.

In addition to this most research has suggested that the majority of patients welcome helpful reminders and alerts provided by a wide range of electronic devices as methods to combat this problem. In particular some of the solutions that have been used by some patients include innovative pillboxes, alarms, diaries, and pagers (Bartlett 2002).

4.3.1.2 Poor education on condition and treatment

Research has shown that the more knowledge a patient has about their condition and the treatment prescribed to them, the more compliant the patient is to their health care plan (Bartlett 2002). This same conclusion can be logically deduced from observation of the obesity management example. If John in the example is prescribed a dietary therapy plan, it is not possible to prescribe in the diet plan absolutely everything he is not allowed to eat or drink, instead the health care professional will focus on teaching and educating John, about the different food groups that could cause him harm, and how he can avoid them, whilst also educating him about the consequences of eating harmful foods. This way every time he feels like indulging himself in foods prohibited to him, not only will he know that they are prohibited, but it is hoped that the knowledge of the damage that the food product will do him will prevent him from eating it and hence remain compliant to his managed diet.

Similarly education on other health care plans such as the pharmacotherapy plan, for John could focus on teaching John what the medicine is doing for him, hence ensuring he knows how easily his health could deteriorate if he neglected taking the medicine, gradually developing in the mind of the patient the advantage of keeping up with their treatment in order to maintain their compliance.

There are many ways in which a patient can be educated about their condition; research suggests that the usage of multi-media in the education process can prove quite effective, for example instructions with the aid of pictures and perhaps even videos (Bartlett 2002). John could be educated about a healthy diet with the aid of instructional health videos, he could be explained how obesity affects his health, his family life, and his work life similarly with the usage of pictures, video and other multimedia material.

4.3.1.3 Poorly tailored health care plans

Patients often also claim that the reason they are non-compliant with health care plans is that the plans are poorly suited to their own lifestyle (Funnel, Anderson et al. 2000; Bartlett 2002). If John starts work from eight in the morning to five in the evening, then it is not suitable for him to have physical exercise scheduled in the morning, chances are he probably will not do it, it may be much better for him to either do these exercises on the weekends, or perhaps in the evening when he returns from work.

Besides interfering with a patient's work-life, poorly tailored health care plans may also be out of fit with a patient's personal life. Health care plans that prescribe health activities for patients on the weekend assuming that they are always free may find the patient non-compliant to these activities, as they may be spending their weekend doing things with their family, for example John may have a routine which sees him spend most of his weekend running around taking his children to music classes, and sports games, with his Sunday mornings spent in church, and most of his Sunday evenings spent with his wife entertaining friends. Research shows that if a health care plan expects a patient to drastically change their lifestyle in order to fit in the health care plan, then there is a high chance that patients will be non-compliant to the health care plan (Bartlett 2002).

Furthermore, tailoring a health care plan to a patient as a one of task is not greatly effective; the health care plan cannot reasonably assume that a patient's lifestyle and daily routines will remain static while the health care plan is executed (Bartlett 2002). A patient's lifestyle is always changing, and hence, research suggests that the tailoring of health care plans to a patient is an ongoing process, and should be done regularly (Bartlett 2002). Health care activities prescribed to John with a routine assuming that he works an 8-5 job may need to be reviewed if he changes his job and now mostly works as a consultant on an irregular timetable.

4.3.1.4 Side effects to prescribed health activities

As mentioned numerous times above, chronic illnesses are characteristically long-term ailments, where the periods of self-management between visits to the health care professional are generally very lengthy. Research shows that often health activities that are prescribed to the patient, give rise to side-effects, these side-effects may be something as simple as aches and pains, or more complicated symptoms indicating perhaps a change in course with respect to the prescribed health care plans (Bartlett 2002).

Due to these side effects a patient may many times neglect to do the health activities and hence become non-compliant with the health care plan (Bartlett 2002). In addition to this because of the long periods after which consultations are scheduled with the health care professional, the exact concerns and side-effects that cause the non-compliance may be forgotten and hence the poor adherence to the health care plan may continue (Turner and Hecht 2001). In our example, if John is prescribed an exercise plan with aerobic exercises, he may find that some of the aerobic exercises prescribed to him cause him to have aches and pains in his knees and ankles, or that the morning medication prescribed to him as part of his pharmacotherapy plan give him headaches, due to this he may sometimes choose to not do the exercises or miss the morning dose of his medication, which may in turn cause a deterioration in his health, and when he finally goes into the clinic for his review, he may have forgotten why he skipped some doses of the medication, or why indeed he could not keep up with the exercises, hence there is little constructive that the health care professional can do.

4.3.1.5 Patient adherence/compliance monitoring

Patient adherence monitoring as it stands today is still a "very imperfect science" (Turner and Hecht 2001), however this is not due to the effectiveness of the techniques but rather because of the nature of the problem, in that if a patient is determined to fool the health care professional about their adherence to the health care plans prescribed to them, then no matter the sophistication of the technology or processes used to measure patient compliance, the measure will always be inaccurate. Therefore most researchers believe that the most effective form of patient adherence is almost certainly interviewing patients as well as actual physiological assessment (Turner and Hecht 2001; Bartlett 2002).

Other attempts at measuring patient adherence, focus on counting, pills for example by the use of smart or electronic bottles, or monitoring the number of refills the patient gets from the pharmacy in order to get an idea on how regularly patients are actually using their medication (Turner and Hecht 2001). Although these methods are not completely in-effective, they can often overestimate patient adherence, in cases where the patient may be fooling the system (Turner and Hecht 2001). Therefore it is recommended that most patients opt for a mixture of approaches, for example, to measure John's adherence to the health care activities that are prescribed to him, the GP may ask him to keep a record of the times when he could not do activities, keep a record of the reasons for not doing the activities, as well as that, the GP may also prescribe certain self-measurement activities for him, such as fitness measure, blood glucose measure etc, results of which John can record and take to consultations with him, where he can take part in an informed discussion on how well he has been able to keep up with his health care plan.

4.4 Health care professional needs

There are limitless needs of a health care professional with regard to providing suitable health care to patients; some of these needs can be categorized into needs for health IT support, such as the need for administrative systems, that cater for issues such as efficient scheduling of health related appointments, patient health record management systems, patient data analysis systems and much more (Beyer, Kuhn et al. 2004). Other categories of needs of the health care professional fall into decision support systems, and yet other categories of health care professional needs can be formed from the need to manage and administer access to medical knowledge bases (Beyer, Kuhn et al. 2004). These needs are largely out of the scope of the care plan modelling system and will not be discussed here.

The main health care professional need that the care plan modelling system is going to cater for is the need for a formalized digital health care plan model. Analysis of existing solutions for formal models to capture and distribute health care information such as the Proforma and the GLIF approach, shows that there are many advantages of having a formalized digitally expressed health care plan model, such as reusability, ease of collaboration, assigning of responsibility, and consistency of approach (Fox, Johns et al. 1998; Boxwala, Peleg et al. 2004). The following section describes the need for health care professionals to have a digitalized health care plan model.

4.4.1 The Need for a Digitalized Formal Health Care Plan Model

There are many reasons why health care professionals may need a formal digitalised health care plan model what follows describes the need for a formalised care plan model, with further description on the structure of the care plan model, and how these needs can be satisfied by such a model described in the following chapter. The main needs for a formal digitalised care plan model can be summarised as follows:

1. The need to be able to easily process health care information, for example automatically validating, and categorising health information.
2. The need to easily reuse health care information.
3. The need to collaborate on developing and storing health care information

4.4.1.1 The Need to Easily Process Health Care Information

If health care plans for patients are converted into formal digital models, then this creates many possibilities for these models to be processed by computers in order to support a whole array of other convenient facilities, for example metrics could be developed which assess patients and automatically select the correct health care plan for patients, or alternatively treatment algorithms may be designed to help health care professionals make decisions on what care plans a patient should be put on, in addition to this having health care information in a formal digital format, allows the information itself to be formally validated, as well as providing health care professionals with helpful reminders and alerts at the point of care, examples of the types of possibilities for computer processing health care guidelines, and information can be seen in the Proforma approach where custom operations allow for triggers, scheduling, data retrieval etc to be executed on their formally modelled clinical guidelines (Fox, Johns et al. 1998). Furthermore having a health care plan as a formal digital model could also mean that the health care plan can be transferred to the patient in more ways than the simple pen and paper approach, i.e. through electronic devices such as mobile phones and PDAs.

4.4.1.2 The Need to Easily Reuse Health Care Information

When John in our example comes in to his GP to gain help in managing his obesity, the GP may design for him a health care plan containing health care regimes, advice and activities in order to help John achieve his ideal health result and get his obesity under control. The health care plan that John's GP designs for him may be a composite of the GP's own experience and knowledge as well as clinical guidelines which the GP may refer to while designing the care plan. If this information is represented in a formal digitalised health care plan model then it means that John's GP can reuse the care plan that is designed, for other patients suffering from obesity without the need to start designing the care plan from scratch. More specifically the advantage of a formal digitalised care plan model is that the GP can dissect the care plan through the computer and reuse small parts of it instead of the whole, for John's GP may decide to reuse the dietary therapy care plan for developing another care plan instead of using the entire obesity care plan, this idea was recognised by both ASBRU as well as GLIF, which allow an entire hierarchy of sub-plans, or "sub-guidelines" (in the case of GLIF) to be reused to construct other more bigger structures (Miksch, Shahar et al. 1997; Boxwala, Peleg et al. 2004). This form of reusability has further advantages of reducing the chances that important health advice is omitted from the care plans through monotonous redesign and implementation of the same health care information. More details on how this reusability can be achieved with a formal digitalised care plan model, along with discussion on how each part of this formal health care plan model can be reused is presented in the following chapter which discusses the formal care plan domain model.

4.4.1.3 The Need to Collaborate on Health Care Information

Along with being able to reuse health care information, it is also an important need to allow health care professionals to collaboratively work on modelling health care information. This need arises from the fact that a patient suffering from chronic illnesses usually has a whole array of health care professionals interacting with and helping the patient, hence it is only fitting that a health care plan can be contributed to by more than one health care professional so that the care plan can grow and encompass the entire treatment regime for the patient, instead of just one part of treatment from one health care professional. A digitalised formal care plan model can facilitate this need for health care professional collaboration by allowing the health care information to be efficiently structured and stored onto the computer, in databases where care plan ontologies can be maintained and freely communicated between health care professional who can extend the care plan models with their own knowledge and then reuse them. Further discussion on this need and how it can be achieved is presented in the following chapter with reference to a detailed obesity management scenario.

4.5 Summary

There are two main stakeholders in the care plan modelling system, the first is the health care professional, and the second is the patient. A health care professional can be anyone from a GP to a social worker, and plays a central role in the treatment of the patient. The patient stakeholder for the care plan modelling system is restricted to patients suffering from chronic long term illness. The needs for the patient stakeholder which the care plan modelling system focuses on includes the need for a patient to adhere to the health care plan designed and prescribed for them in order to ensure the best health results. The main causes of patient non-compliance to their health care plan were deemed to be; forgetfulness to conducting activities, patients having poor education of their health care, health care plans being poorly suited to the patients, side effects to prescribed health activities that go un-addressed and finally the need for patient adherence monitoring. For the health care professional these needs correspond to the need for having a digitalised care plan model, through which health care professionals can reuse health care plans, process health care plans through the computers to automate various decision making processes, and allow multiple health care professional to collaborate on the health care plan.

Chapter 5 - Care Plan Domain Model

5.1 Introduction

The purpose of this chapter is to introduce and discuss the formal care plan conceptual domain model which forms the basis of the visual care plan modelling language as well as being one of the needs of the health care professional as discussed in chapter 4 prior to this. The chapter firstly describes the advantages of the having a formal care plan modelling framework emphasising the needs of the health care professional discussed in the previous chapter, and then introduces a case study of a chronic illness management scenario in order to explain the structural components of the care plan domain model. Following this is the discussion explaining the conceptual difference in the care plan domain model between care plan templates and care plan instances, and how the care plan domain model can capture the health care professional's needs of reuse, collaboration and consistency.

5.2 Advantages of a formal Care Plan Modelling Framework

The advantages of having a generic care plan authoring framework are many, and most importantly these advantages are experienced by both user groups of the framework. The first user group of the framework is of course the one that encompasses the variety of health care professional that make up the health care industry. This includes not only general practitioners, but also any health care professional that helps in lifestyle rehabilitation, which includes professionals such as physiotherapist, dieticians, and also social workers who may help with programs such as help to quit smoking, and anti-alcoholism programs.

The second user group encompasses the patients themselves. More specifically this user group involves patients that are suffering from long term to permanent illnesses/ailments and need treatment and maintenance for the majority of their lives.

The health care professionals benefit from a generic health care plan model in many ways. Firstly, it translates a health care guideline, which is most often portrayed in a textual, descriptive form, into a digital computerized model, the advantage of this is that the care plans based on treatment guidelines essentially become reusable templates that can be applied in different ways to many patients suffering from similar ailments. Furthermore having a treatment guideline in an electronic format means that collaboration on improving and extending the care plan templates becomes easier, where by an entire community of multidisciplinary health care professionals can contribute to the health care plan.

The patient benefits from realization of a generic "digital" care plan authoring framework. The first advantage is that a treatment guideline which is usually only understandable by the health care experts can now be represented in a simpler generic way which makes it easier for non-health care personnel to comprehend health information. Secondly, because digital health care plans can be designed and reused, persistent usage of the system, could lead to a standard in care plans for patients, which in turn will mean that there will be consistency in the treatment of patient's suffering from similar ailments.

Lastly an advantage of having a computerized generic care plan authoring framework, that applies to both user groups, is that algorithms could be designed that automatically decide (perhaps piggy-backing on existing decision support systems) on the correct care plan for a patient by reading in the patient's symptoms, hence reducing the workload of the health care professional, and adding further consistency in the process of caring for patients.

5.3 Case studies of Chronic Illnesses

In order to get a good perspective of the types of situations that a care plan model must cater for, two separate chronic illnesses were chosen and investigated, the first being obesity, and the second being high blood cholesterol, and for each respective chronic illness a fictional scenario was built, that was based on the actual guidelines for treating, managing and evaluating the disease (National Institute Of Health 1998; Program 2002). The actual guidelines are too complex to express here, however the scenarios capture all the main points of the guidelines, presenting them in patient/doctor context.

Each scenario involves two fictional characters, the first being the patient that is suffering from the chronic illness and the second being the health care professional that is looking after them. For simplicity only one scenario, namely the obesity management scenario is explored in detail in this chapter.

5.3.1 Scenario for Obesity Management

Preamble: - Thus scenario centres on a fictional character John Doe, who suffers from obesity and his subsequent treatment regimes according to the clinical guidelines for obesity treatment (National Institute Of Health 1998).

Scenario: -

John Doe is a 35 year old software developer who works for a well known software company; he has a wife and two children and lives in a three bedroom apartment in the city.

John suffers from class 1 obesity; this means that his body mass index (BMI) is over 30 (National Institute Of Health 1998). John's condition has many effects on his lifestyle, prime among them is low self esteem due to his appearance, in addition to this John also realizes that his condition makes it very difficult to spend time with his children, including such activities as participating in sports, and attending extracurricular activities with his children. In addition to John is worried that his current obesity could lead to other serious illnesses.

Due to these reasons and many more John has decided to take action to improve his condition. He visits his general practitioner (GP), who in turn refers to the clinical guidelines for treating obesity and suggest to John the following strategies described in the guidelines.

The treatment strategies, performance goals, health activities and assessment regime that John, is prescribed to treat and manage his obesity by his GP are summarised below.

5.3.1.1 Strategies for treatment

The following sections describe and discuss the treatment strategies used to manage and treat John's condition.

5.3.1.1.1 Initial Assessment

During John's first visit to his GP, the GP fully assesses John's condition. This assessment includes an assessment of factors such as the fat content of the body, which is measured from the Body Mass Index (BMI) and also the diameter of the waist (National Institute Of Health 1998). In addition to this, John's GP also analyses the different forms of risks that his obesity is likely to have on him, this includes the development of other serious diseases such as diabetes, and heart problems (National Institute Of Health 1998). The process of assessment that follows during this stage is often represented in most guidelines as a decision based task flow. An example of this for the obesity management guideline used for this scenario can be seen in figure 17 below.

5.3.1.1.2 Treatment programs

After going through the assessment stage described above, John's GP realizes that there are a lot of problems that are contributing to John's obesity, furthermore there are many other serious risks posed by John's obesity such as a potential to develop diabetes, that need to be monitored. Therefore John's GP suggest a combination of therapies to treat and manage John's condition. These therapies include (National Institute Of Health 1998):

1. Dietary therapy
2. Physical activity
3. Behaviour therapy
4. Pharmacotherapy
5. Diabetes Monitoring

Each of the therapies/programs that are prescribed to John, contribute in some way to either treating John's obesity through weight loss, dieting, and the administration of medicine, or attempt at keeping an eye on risk factors such as monitoring John's diabetes. It is important to realize that although most of these modules may be prescribed by John's GP, some of them may be administered by different types of health care professionals, for example, for behavioural therapy, John may be prescribed to see a psychologist, who will specify the activities that John must do.

5.3.1.1.3 Performance standards

John's doctor says that the accepted performance standard for patients with class 1 obesity is to aim to get their BMI under 29.9 (National Institute Of Health 1998). All the programs to treat and manage John's conditions (as listed above), augment and contribute to this performance goal, but they in turn may also have their own performance goals (performance sub-goals). For example, Dietary therapy has the performance goal to reduce the total fat intake and also to reduce the total calories intake (more specifically John's doctor has prescribed that his diet should decrease his calorie intake by 500 -1000kcal/day) (National Institute Of Health 1998). Likewise the diabetes monitoring program may also have its own performance target of optimal blood sugar levels. Certain medicines part of the pharmacotherapy plan may have a performance goal of keeping the patient's blood pressure level to a certain optimal level.

5.3.1.1.4 Program activities

For each form of therapy that John's GP prescribed for him, he prescribed a whole range of health care activities that would help John achieve the performance goals/targets that are needed to treat his obesity. Some of the health activities that were prescribed to John are as follows:

Pharmacotherapy: Under this program the GP prescribes for John various medications to firstly control his blood glucose sugar and a few other weight loss drugs as an adjunct to the dietary plan.

Physical activity: Under this program the GP prescribes for John walking for 30 minutes at least 3 times a week (National Institute Of Health 1998).

Behavioural therapy: John's GP thinks that there are a few behavioural and attitude problems with John that could affect his adherence to the programs prescribed to him. Therefore the GP refers John to the resident psychologist who prescribes various stress management activities to John to conduct at specified times, in an effort to remove the un-health eating habits that are often a result of stress at work and home.

Dietary therapy: John is suggested a low calorie diet and given a few suggestions by the GP, also he is referred to an experienced dietician who creates a detailed meal schedule/plan for John.

Diabetes Monitoring: In the pharmacotherapy program John's GP has already prescribed glucose control drugs, under this program the GP prescribes to John a schedule for self measurement of his blood glucose sugar by using the testing pen and testing strips at home.

5.3.1.1.5 Data collection / assessment / & review

In order to have the best possible results the clinical guidelines for obesity suggest a close monitoring of the results of the various treatment programs. Therefore within each treatment program John's doctor also prescribes john activities of measurement, this includes activities such as measuring his blood glucose levels and recording it on a daily basis, or alternatively recording weight loss on a daily basis. John's GP also recommends him to take note of any concerns he has, or of any questions he may have.

John's GP also schedules in fortnightly consultations in order to asses John's progress according to the performance targets set out for him. During these consultations John's GP uses a treatment algorithm to asses John's progress the likes of which is shown in Figure 17 and that which was used in the initial assessment stage to determine the progress that John has made.

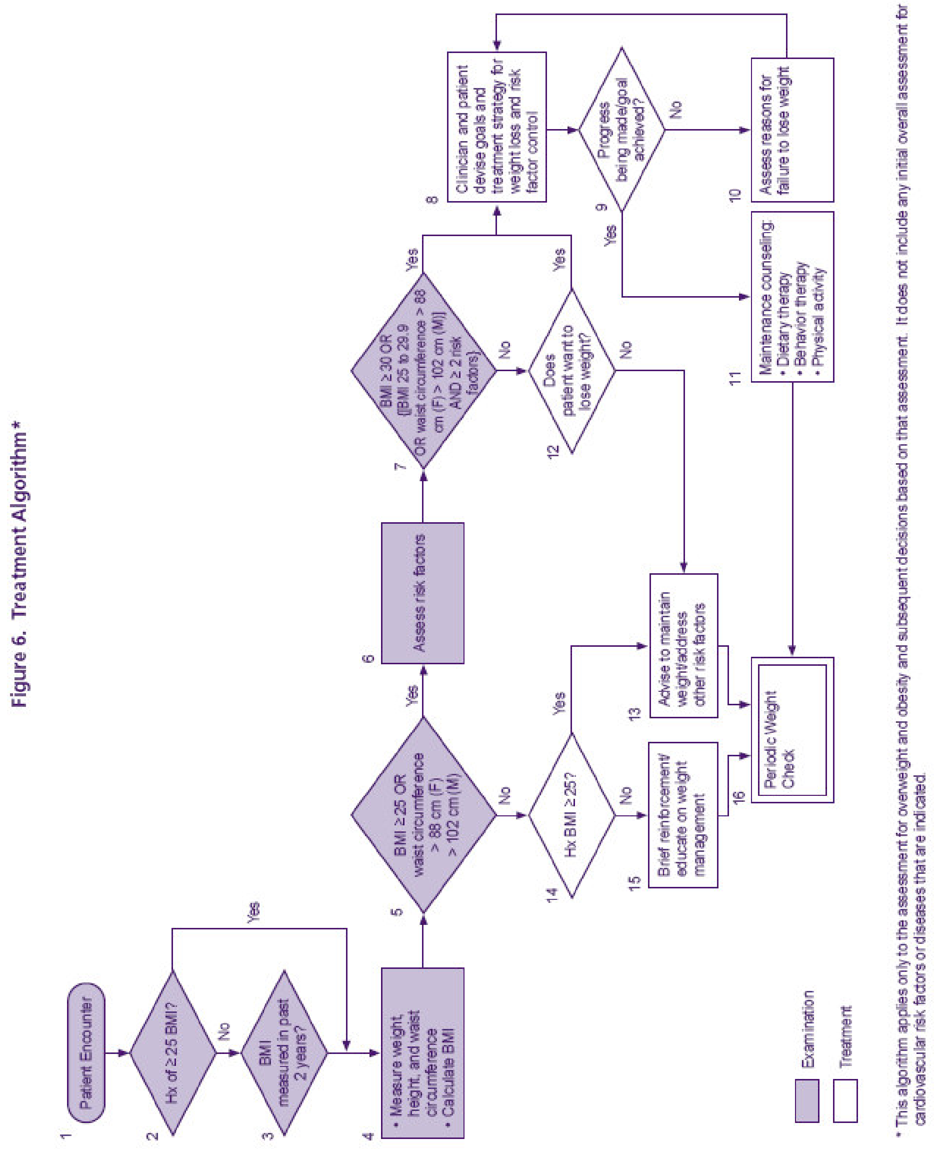


Figure 6. Treatment Algorithm*
1
Patient Encounter
2
Hx of ≥ 25 BMI?
No
3
BMI measured in past 2 years?
Yes
4
• Measure weight, height, and waist circumference
• Calculate BMI
5
BMI ≥ 25 OR waist circumference > 88 cm (F) > 102 cm (M)
Yes
6
Assess risk factors
7
BMI ≥ 30 OR ([BMI 25 to 29.9 OR waist circumference > 88 cm (F) > 102 cm (M)] AND ≥ 2 risk factors)
Yes
8
Clinician and patient devise goals and treatment strategy for weight loss and risk factor control
9
Progress being made/goal achieved?
No
10
Assess reasons for failure to lose weight
Yes
11
Maintenance counseling:
• Dietary therapy
• Behavior therapy
• Physical activity
No
12
Does patient want to lose weight?
Yes
No
13
Advise to maintain weight/address other risk factors
No
14
Hx BMI ≥ 25?
Yes
No
15
Brief reinforcement/ educate on weight management
16
Periodic Weight Check
▨ Examination
□ Treatment
* This algorithm applies only to the assessment for overweight and obesity and subsequent decisions based on that assessment. It does not include any initial overall assessment for cardiovascular risk factors or diseases that are indicated.

Figure 17: The treatment algorithm for obesity reproduced from the obesity treatment guideline reproduced from (National Institute Of Health 1998).

5.4 The Structure and Components of the Care Plan Model

The structure of the care plan modelling framework was first designed by Abizer Khambati and Sumedh Kanade in an earlier project (Kanade 2006; Khambati 2006) through the study of the guidelines that are typically used in the treatment and management of patient's suffering from chronic illnesses, such as the guideline for managing and treating obesity. This model was used and extended for the purpose of using it for the care plan modelling visual language. What follows is a discussion of the new extended care plan domain model presented here using the obesity management scenario, using the scenario to not only explain the structure of the care plan, but also to explain the rational behind the structure.

5.4.1 Care Plan Modules

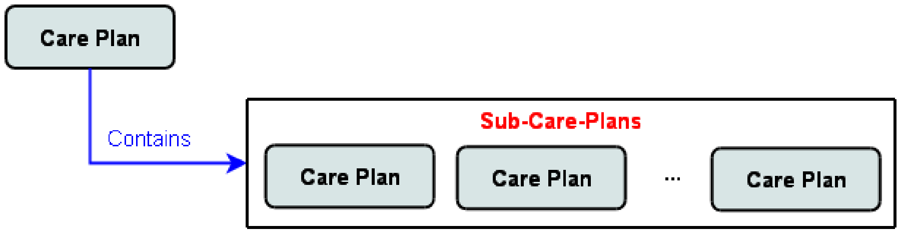


Figure 18: A care plan can contain sub-care-plans.

A care plan module can be thought of as a group or category of health care strategies, in the scenario expressed above it can be seen that the obesity management guideline can be thought of as one big care plan.

As explained above, most treatment regimes for chronic illnesses usually involve many different types of therapies, where different factors of a patient's physiology as well as lifestyle need to be monitored, controlled and or managed in order to see positive effects in the patient. In the obesity management scenario it can be seen that the GP has realized that in order to treat John's obesity there are many forms of therapy that are needed, such as pharmacotherapy, physical therapy, behavioural therapy, dietary therapy, and diabetes monitoring (National Institute Of Health 1998). Each of these forms of therapy, in themselves again, are a group of other strategies that are refined to that form of therapy, hence they are also essentially care plans on their own, for example the dietary therapy care plan is a group of meal planning strategies, the physical therapy care plan is a group of strategies for light aerobic exercises, the behavioural therapy care plan is a group of strategies for stress relief, and so on for the other types of therapeutic measures recommended to John for the management and treatment of his obesity.

Therefore the definition of a care plan can be further refined to being a group or category of health care strategies that may in turn be divided into other care plans, which themselves may be composed of multiple sub-care-plans as shown in figure 18. Hence the overall structure of a care plan can be visualized more as a tree like, or hierarchical structure.

5.4.2 Performance Metrics

As with any planned strategy to achieve something, there must always be performance goals that are defined before action is taken inline with the strategy. The aim of the performance goals is to keep track of how well the strategy is working. It is no different for the health care strategies encompassed by a care plan.

Looking at the obesity management scenario once again, it can be seen that in order to assess the seriousness of John's obesity his GP uses his body mass index (BMI) and his waist size as indicators, in John's case his BMI measure amounts to 31, and his waist size is 120cm. This classifies John's obesity as class 1; therefore in order to treat his obesity, the performance goal that John's GP sets him involves getting a BMI of 29 or lower and reducing his waist size to less than 110cm (National Institute Of Health 1998).

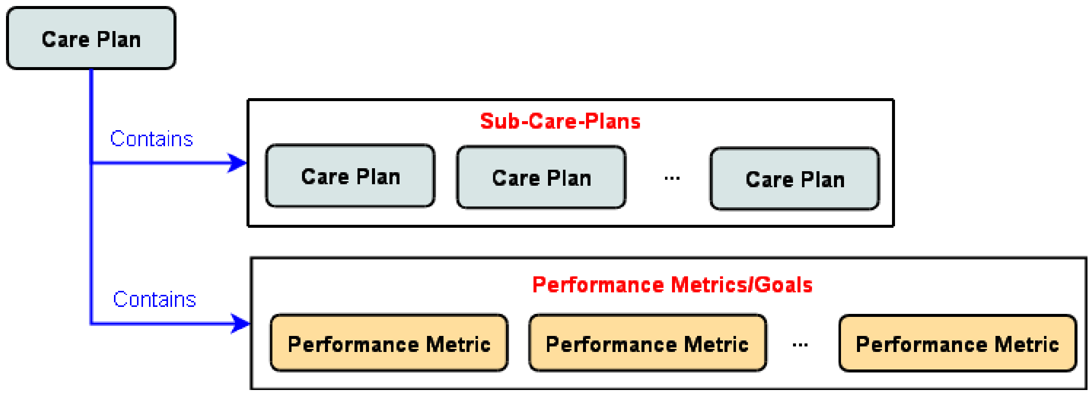


Figure 19: Care plan contains sub-care-plans and performance metrics.

Therefore the tree like structure of a care plan can now be extended to contain another component/node, and this is a performance metric as shown in figure 19 above, putting it in the context of the obesity management scenario, the obesity management care plan will have two performance metrics, the first being the BMI and the second being the waist size in cm.

Similarly each of the sub-care plans under the obesity management plan, such as dietary therapy and diabetes monitoring, whilst contributing to the overall performance goals/metrics of the obesity management plan, can also have performance goals of their own. For example, a performance goal/metric of the dietary therapy plan could be to reduce the calorie intake per meal to 1200 as shown in figure 20 below.

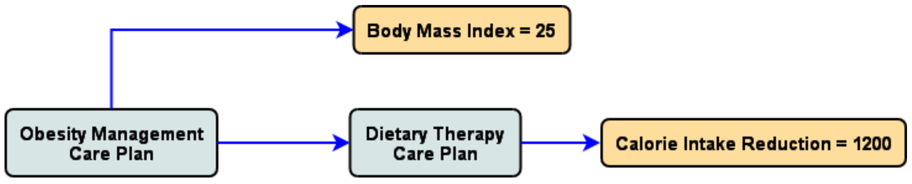

Body Mass Index = 25
Obesity Management Care Plan
Dietary Therapy Care Plan
Calorie Intake Reduction = 1200

Figure 20: Example of care plans containing performance metric, the obesity management care plan containing the body mass index performance metric and the dietary therapy care plan containing the calorie intake reduction metric.

5.4.3 Health Activities

The core of any strategy is the actual action that makes up the strategy. In a health care strategy, this action can be seen as health activities. In the obesity management scenario above, it can be seen that John's physical therapy may be made up of light aerobic exercise, such as jogging before breakfast, well as his pharmacotherapy may involve him taking one type of medicine before lunch and one at the end of the day before dinner.

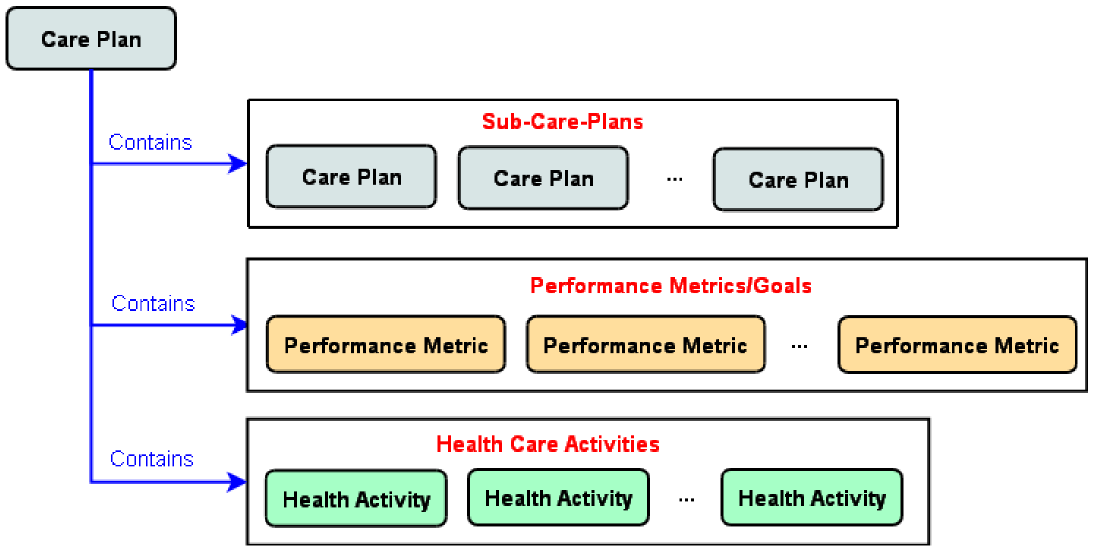


Figure 21: Care plan contains sub care plans, performance metrics, and health care activities.

Similarly the diabetes monitoring care plan for John may require him to measure his blood sugar using the self testing unit, every week before breakfast. Furthermore John may need to have a consultation with his GP every three months, in order to review his obesity, for which he needs to come in after an all night fast, and before breakfast as shown in figure 22 below.

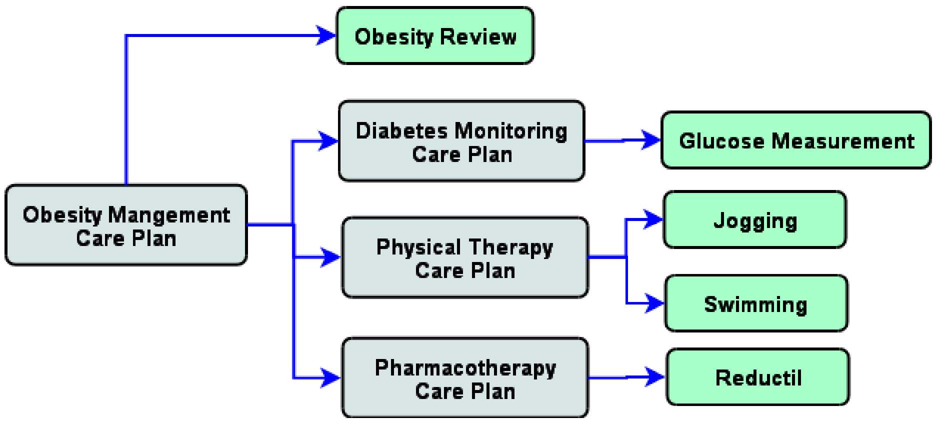

Obesity Review
Diabetes Monitoring Care Plan
Glucose Measurement
Obesity Mangement Care Plan
Physical Therapy Care Plan
Jogging
Swimming
Pharmacotherapy Care Plan
Reductil

Figure 22: Diagram showing the obesity management care plan with 3 sub-care plans, namely the diabetes monitoring, physical therapy, and the pharmacotherapy care plan. Each care plan is made up of health activities (the green boxes).

These actions that John is required to take as part of the different health care strategies correspond to the health activities that each care plan is made up of as shown in figure 21 above. However upon close inspection of the different health activities that have been prescribed for John, it can be seen that the activities can be further generalized or grouped into different types as shown in figure 23 below.

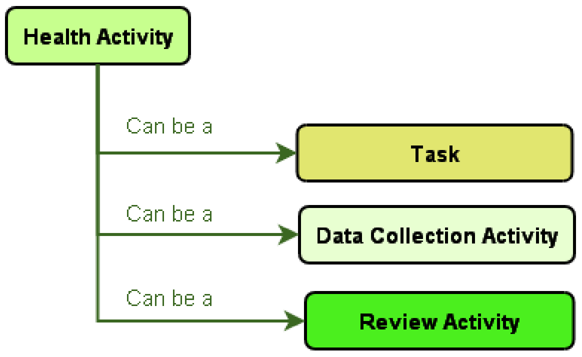


Figure 23: The different types of activities that can be deduced by analysing the obesity scenario.

It can be seen from figure 24 that activities such as jogging, or medicine taking are essentially simple tasks, which the patient usually conducts themselves, in comparison to activities such as measuring the blood glucose sugar, or perhaps measuring the resting heart rate before jogging, even though these activities are still usually done by the patient themselves, in contrast to the simple tasks, these activities can rather be thought of as data collection activities, because they require the patient to measure or self-assess some parameter and record the result, in the glucose measurement activity, this result will be the blood glucose sugar, in mmol/L, well as for the heart rate measurement activity this result will be the number of heartbeats per minute.

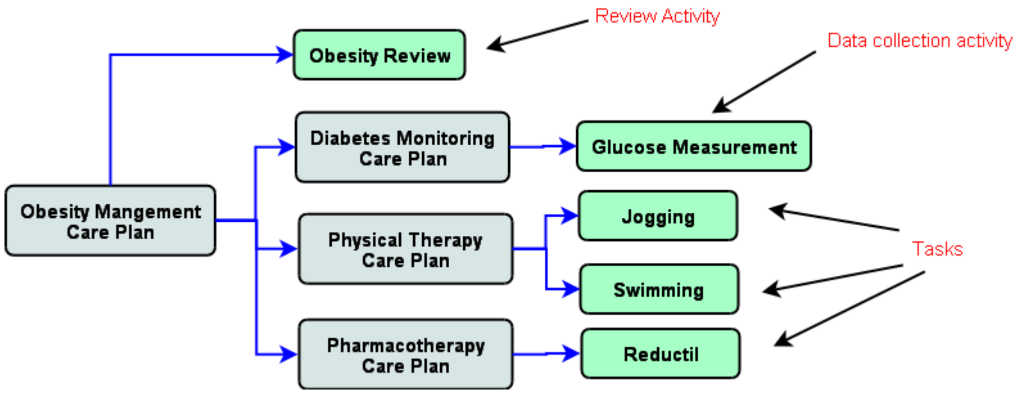

Review Activity
Data collection activity
Obesity Review
Diabetes Monitoring Care Plan
Glucose Measurement
Obesity Mangement Care Plan
Physical Therapy Care Plan
Jogging
Swimming
Tasks
Pharmacotherapy Care Plan
Reductil

Figure 24: The diagram above shows examples of the different types of activities that a care plan can be composed of, from normal tasks such as jogging and swimming, to a data collection activity such as the glucose measurement activity, and lastly review activities such as the obesity review.

In addition to having tasks and data collection activities it can be seen from the scenario that John has been ordered a review every three months. This can be extracted as another type of activity, it is different from the afore mentioned activities in that it is usually conducted by two or more people, i.e. the patient, and one or more of their health care professionals. This however, is not the only difference between a task or a data collection activity and a review activity. It can be seen that usually tasks and data collection activities, are composed of a single activity, however review activities can be seen rather as composites of many other review activities. In the example obesity management scenario it can be observed that the obesity review activity may in turn be composed of a diabetes review, a cholesterol review and a weight loss review. Another separate example can be seen if a patient coming in for an eye check-up review activity, which in turn is made up of a review of their reading capacity, and a review of the eye's physiology itself.

5.4.3.1 Components of an Activity

So far it has been deduced from the analysis of the obesity management scenario that a care plan which encompasses a group or category of health care strategies can be composed of performance metrics/goals, other care plans as well as health care activities that form the action for the care plan. Each health care activity in turn can also be seen as a composite of other components as shown in figure 25 below.

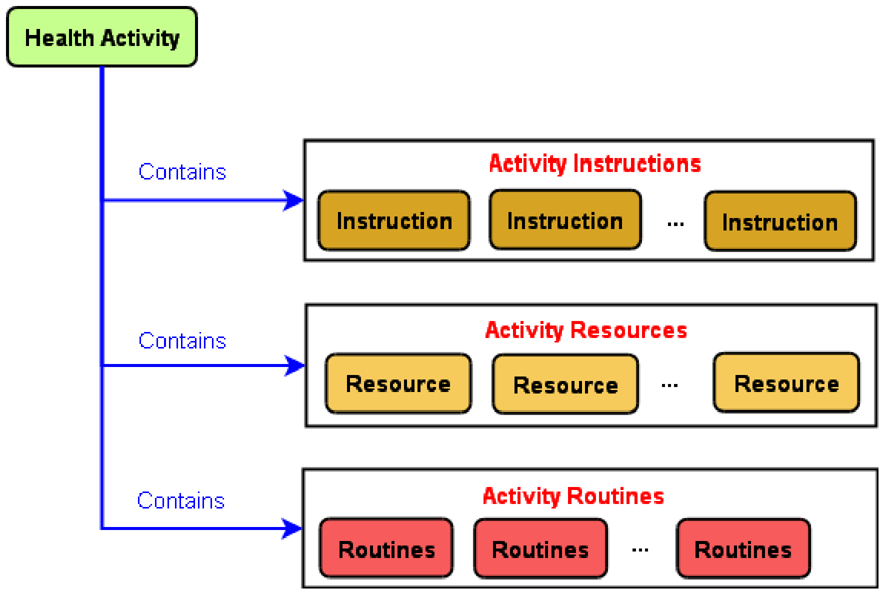


Figure 25: A health care activity is made up of activity instructions, resources needed to do the activity, and routines dictating when to do the activity.

5.4.3.1.1 Instructions

Observing the obesity management scenario above, it can be seen that when John's GP prescribes a certain activity for the patient, there are always instructions that are provided on how to conduct that activity. An example of this can be seen in the diabetes monitoring care plan, for the glucose measurement activity, here John is required to use the self-test kit for blood glucose to measure the glucose levels in his blood. John may never have used such a kit before, and may need guiding steps to help him complete the activity properly, such as (see figure 26):

"1. Make sure to change the pricking needle after every 10 uses
2. Choose a needle setting
3. Charge the pen by rotating the head of the pen, until you here a click
4. Angle the pen, pointing it at the finger
5. Prick the finger by depressing the button on the testing pen
6. …"

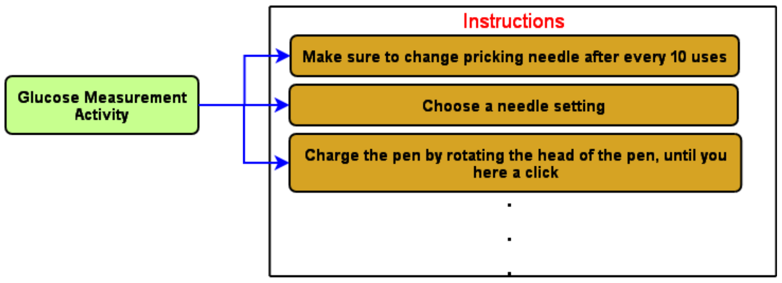

Instructions
Glucose Measurement Activity
Make sure to change pricking needle after every 10 uses
Choose a needle setting
Charge the pen by rotating the head of the pen, until you here a click

Figure 26: Diagram showing a glucose measurement activity with instructions.

Furthermore even if there are certain activities that the patient is used to conducting, if the procedure required for the activity is complicated the patient may need to refer to instructions for doing the activity every time. Therefore the structure of a health care activity can be extended to include instructions. There is no barrier as to the media type of the instructions, with the fast growth rate of technology, most electronic devices such as PDAs, mobiles and certainly desktop computers support a range of multimedia, therefore instructions can be in the form of simple textual messages, or can be more complicated flash videos, or alternatively can be in the form of a web annotation. Due to timing constraints in the prototyping stage however, activity instructions have been limited to text based.

5.4.3.1.2 Resources

Another component that can be added to the composite of a health activity is a resource component. Any health activity whether it be a simple jogging activity part of a physical therapy care plan, or a glucose measurement activity, before the activity can be done by the patient, they require certain materials, for example before a patient can go running, they need to ensure that they have a pair of running shoes. Similarly before John, in the scenario above can test his blood glucose levels, he needs to have a self-testing meter, a testing pen, and one or more fresh "*gluco-card*" measuring strips as shown in figure 27 below.

The above describes resources that are of material type, however if the obesity management scenario is considered once again, it can be seen, that under the obesity management care plan, there is an obesity review health activity. The obesity review, may be maid up of a diabetes review, a weight loss review, a cholesterol review, etc for these different types of review not only will the health care professional need material resources, such as the availability of an examination room, and perhaps the necessary equipment, but the health care professional will most probably also need the services of a nurse, or "*med- lab*" staff.

Therefore the structure of the health care activity can once again be enhanced to include necessary resources to conduct the activity, whether they are material resources, or service resources.

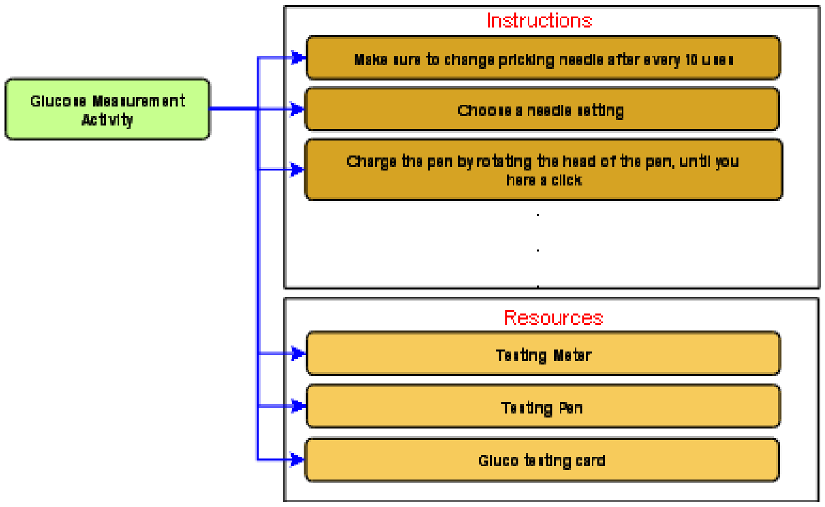

Glucose Measurement Activity
Instructions
Make sure to change pricking needle after every 10 uses
Choose a needle setting
Charge the pen by rotating the head of the pen, until you here a click
Resources
Testing Meter
Testing Pen
Gluco testing card

Figure 27: Glucose measurement activity with instructions and resource needs specified.

5.4.3.1.3 Routines

In every clinical guideline, whenever any action (or health care activity) is suggested, there is always a suitable time frame/ routine that it is associated with it. It is important to realize that this routine is a very generic guideline indicating to the health care professional how often or when some action is to be taken, and it is not an affirmative timetable for the patient.

The first example that can be seen of this is in the routine that is suggested for the obesity review activity in the obesity management care plan. It is suggested that John should have his obesity reviewed *"every 3 months"*, similarly the guideline suggest that as part of the exercise plan, John should jog or do some other light aerobic exercise *"every 2-3 times a week for 30 minutes"*, or *"bicycling every day for 35 minutes"*.
These generic routines tell the health care professional as well as the patient when to do the recommended activities, however they are not an exact timetable for the patient, telling them what date and time exactly to do the activities on. The complete structure of the glucose measurement activity can be observed in figure 28 below.

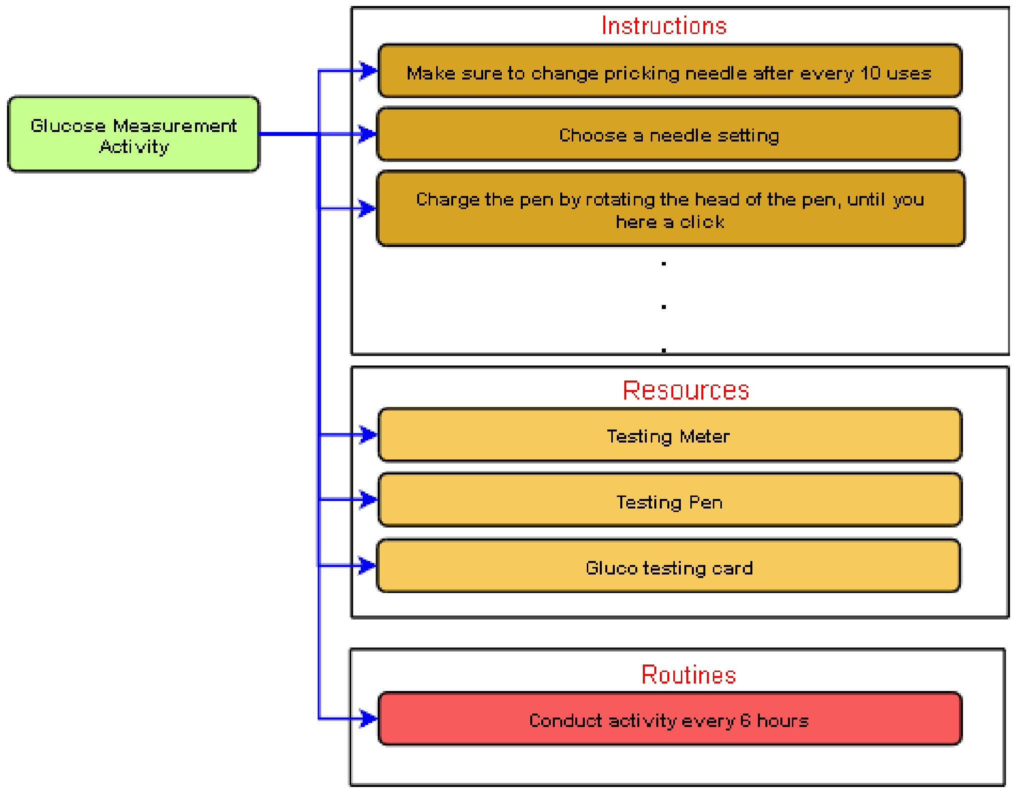

Instructions
Make sure to change pricking needle after every 10 uses
Choose a needle setting
Charge the pen by rotating the head of the pen, until you here a click
Glucose Measurement Activity
Resources
Testing Meter
Testing Pen
Gluco testing card
Routines
Conduct activity every 6 hours

Figure 28: Glucose measurement activity with instructions, resource needs, and routines.

5.4.4 Assessment Modules

One of the most important parts of a clinical guideline for the treatment of any illness is in fact the assessment, and or evaluation of the patient. Assessment of the patient can occur at various times in the treatment process, it can start with an initial assessment in the first instance when the patient comes in, and then it can follow in regular instalments, as a form of keeping check of the patient and maintaining positive results. A similar strategy can be seen in the obesity management scenario, where when John comes in for his first check-up regarding his obesity, his GP assesses him completely to determine the seriousness of his condition and then sets up three monthly review sessions in order to keep check of any progress made.

Therefore in order to truly represent the guideline for treating a patient, a care plan must be extended to include assessment modules as shown in figure 29. Just like the health care activity component of the care plan, an assessment component is also a composite of other components. In contrast to other components however, which are focused at capturing the health care strategies used to improve/treat the patients condition, the assessment module rather is focused at capturing the decisional task-flow that determines the health care strategies that need to be used.

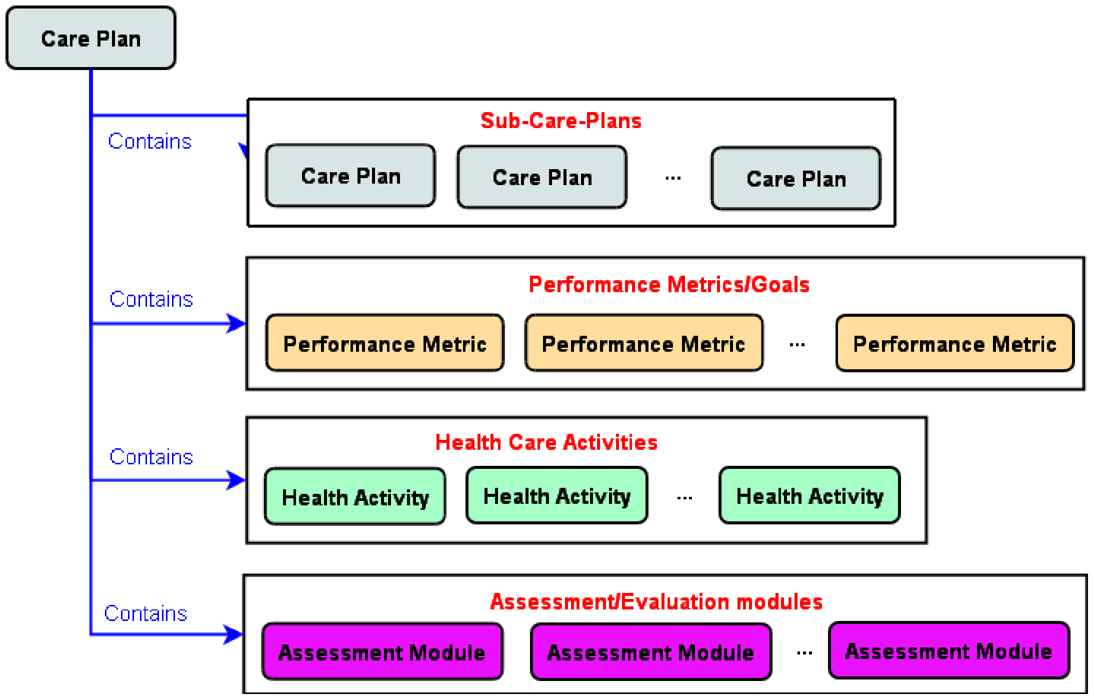


Figure 29: A care plan contains sub-care-plans, performance metrics, health activities and assessment modules.

The decisional task-flow guiding the assessment of patient's is often expressed in their respective clinical guidelines, in both forms, textual as well as diagrammatic, it is often also referred to as a *"treatment algorithm"* (National Institute Of Health 1998). By observing the treatment algorithm for assessing obesity provided in figure 17, in the obesity management scenario above, the components that make up an assessment module can be deduced.

5.4.4.1 Assessment Components

Analysing the treatment algorithm presented in the obesity management treatment guideline and shown in figure 17, it can be seen that the entire decisional task flow that makes up the algorithm, is composed of three main components as shown in figure 30 as well as figure 31(National Institute Of Health 1998):

1. Decision – this can be viewed as rather a condition or a question that the health care professional has to answer with regard to the patient that branches to some action if true and another if false. An example of this is *"Has BMI been measured in the past two years"*, or *"Hx>=25 BMI"*.
2. Examination – This is an actual act of assessment that the health care professional conducts for the patient, for example, *"Measure weight, height and waist circumference"*, or *"Assess risk factors"*.
3. Treatment – The final component that can be extracted from the treatment algorithm, is a recommendation for treatment, for example, *"Provide brief education on weight management"*, or *"Provide maintenance counselling: Dietary therapy, physical therapy …"*.

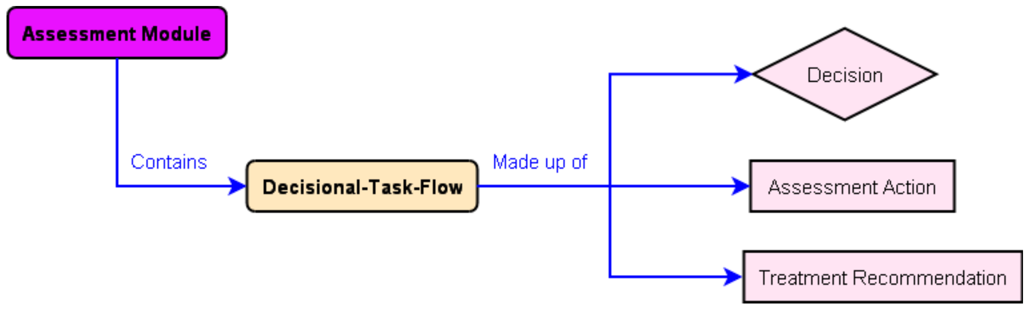


Figure 30: An assessment module is made up of a decisional task flow guiding the health care professional on how to assess a patient; it is made up of decisions, assessment actions, and treatment recommendations

5.4.4.2 Relationships between Assessment Components

The relationships between the other components of a care plan are relatively easy to explain and rather intuitive, for example, it is easy to see that the relationship between most components in a care plan is the *"contains"* relationship, i.e. a care plan *"contains"* other care plans, a care plan *"contains"* performance metrics, a care plan *"contains"* health activities, health activities *"contains"* instructions, resources and routines.

The relationships between assessment components are however, different, and reflect the flow between the decisions (or conditions), assessment actions, and the treatment recommendations.

In general the relationship between assessments components can be thought of as the *"leads to"* relationship, from the treatment algorithm it can be seen that this relationship fits all the relationships in the diagram. Therefore the relationships between the assessment components can be viewed as the following:

1. A decision/question/condition → *Conditionally Leads to*
   → Other decisions/questions/conditions
   → Assessment actions
   → Treatment recommendations

2. An assessment action → *Leads to*
   → Other decisions/questions/conditions
   → Assessment actions
   → Treatment recommendations

3. An treatment recommendation → *Leads to*
   → Other decisions/questions/conditions
   → Assessment actions
   → Treatment recommendations

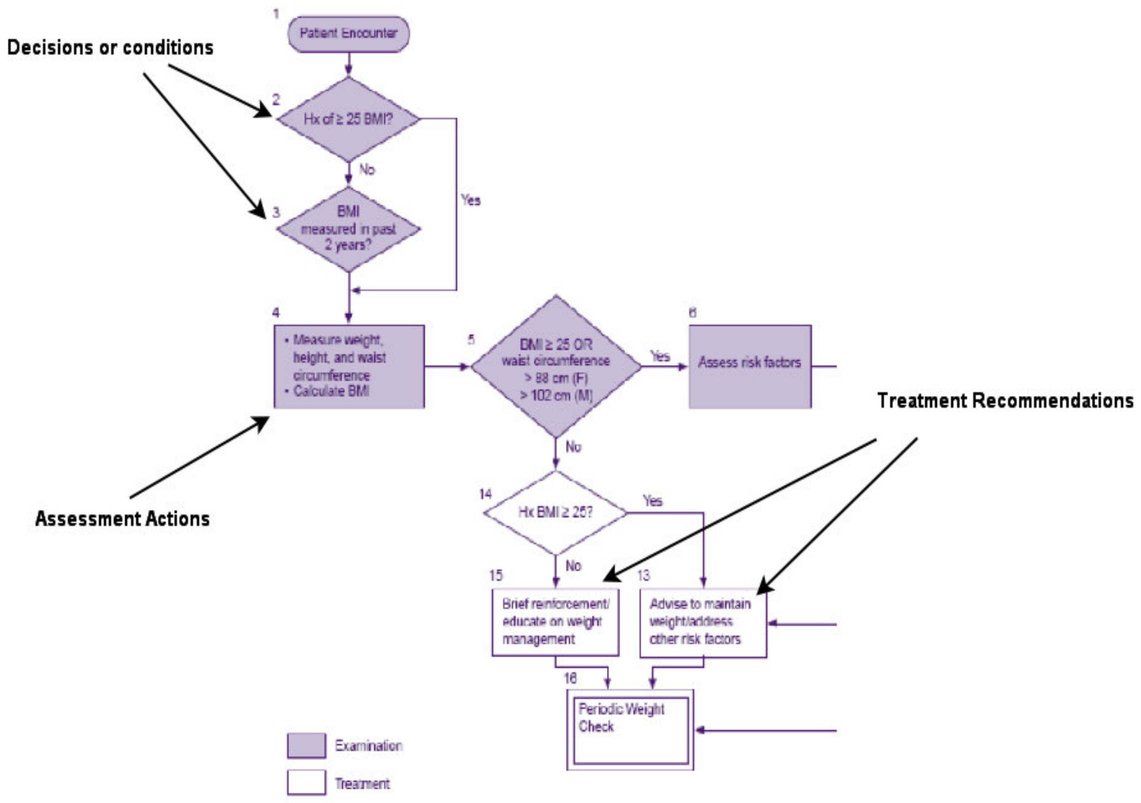

Decisions or conditions
1 Patient Encounter
2 Hx of ≥ 25 BMI?
No
Yes
3 BMI measured in past 2 years?
4 • Measure weight, height, and waist circumference • Calculate BMI
5 BMI ≥ 25 OR waist circumference > 88 cm (F) > 102 cm (M)
Yes
6 Assess risk factors
Treatment Recommendations
Assessment Actions
No
14 Hx BMI ≥ 25?
Yes
No
15 Brief reinforcement/ educate on weight management
13 Advise to maintain weight/address other risk factors
16 Periodic Weight Check
Examination
Treatment

Figure 31: A section of the treatment algorithm taken from the obesity management guideline (National Institute Of Health 1998), with the different assessment components identified, as decisions or conditions, assessment actions, and treatment recommendations.

5.4.5 Care Plan Templates, Care Plan Instances, and Reusability

5.4.5.1 Care Plan Templates and Care Plan Instances

A care plan once designed for a patient exists in one of two stages, the first stage is the template stage, and the second stage is the instance stage. Looking at the obesity management scenario once again, John's GP uses his understanding of the obesity management clinical guideline, to design an obesity management care plan, this care plan may have been designed for the first time, during John's visit, or it may be that the GP had designed an obesity management care plan previously for another patient suffering similarly and is now making use of this for John. Regardless, this care plan can be thought of as a template for the treatment and management of obesity.

It may be that when the GP designs the care plan template, that he decides to include all possible strategies for the management of obesity in the care plan, to make the template complete, similarly it can also be possible, that the GP decides to only model the care plan with a few selected treatment strategies that he finds relevant to the patient he is currently treating.

The result of this is that the care plan template that the GP may use for a patient, may have aspects in it that are either not needed for the patient, or are unsuitable to the patient in someway. Therefore before a patient can reasonably use the care plan template an instance of the care plan template must be created that is specific, and fine tailored to the patient's needs, the care plan instantiation process can be observed in figure 32 below.

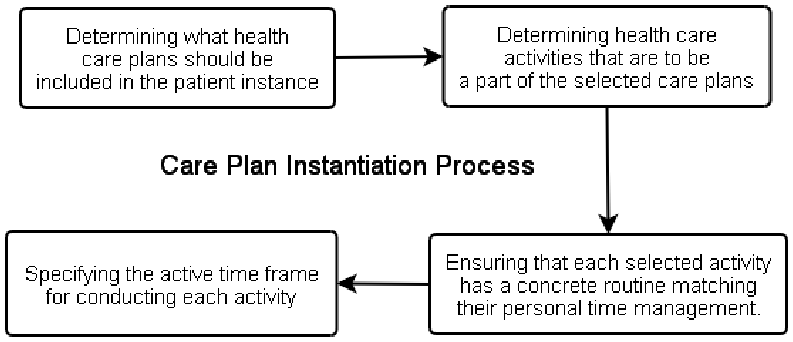

Determining what health care plans should be included in the patient instance
Determining health care activities that are to be a part of the selected care plans
Care Plan Instantiation Process
Ensuring that each selected activity has a concrete routine matching their personal time management.
Specifying the active time frame for conducting each activity

Figure 32: Diagram showing the process for converting a care plan template into a patient specific care plan instance.

The levels of fine tailoring that are permissible within the structure of a care plan are as follows:

1. Tailoring exactly what sub care plans are needed and which sub care plans can be excluded from the patient's care plan instance. For example, It may be that the obesity care plan template that John's GP is planning to use for him has, among other sub-care plans, the diabetes monitoring sub-care plan, but perhaps it is unnecessary for John to be monitoring his diabetes, hence the obesity care plan instance for John, will have the diabetes monitoring care plan removed from it.

2. Tailoring exactly which health care activities part of care plan's action are necessary for the patient. It may be that in the physical activity sub-care plan in the obesity management care plan template, has certain activities that will cause John to strain his back, the GP knowing that John has an active back problem, may remove these activities from the physical activity care plan's instance for John.

3. Tailoring the routines for the health care activities to match each patient's unique time management. As described above each health care activity has a suggested routine attached to it, however the routine may not be acceptable or too general to the patient, and may require some change before the patient can practically do the activity. For example, the physical activity care plan template, may have a jogging activity, with a suggested routine to do the activity every 2 days, for 30 minutes at eight in the morning, for John, eight in the morning may not be convenient, because he has to drop his children to school, and then head to work, therefore his instance of the physical activity care plan will have the jogging activity routine, change its time of conduction to perhaps an earlier time of seven in the morning, or perhaps later on in the day.

4. Tailoring the time frame for different actions. Different patients need to conduct the same health care activities for different amounts of time, one patient may only need to take some of the pharmacotherapy drugs, that make up the health activities of the pharmacotherapy plan for 3 months, but it is possible that John, may need to continue taking the drugs for over 6 months, therefore the care plan instance of John, for the pharmacotherapy care plan will have the time frame for the drug taking activities to be 6 months.

After tailoring all these aspects of the care plan template to match the needs for the patient the care plan template has in essence been transformed into an instance of the template that is now specific to the personal requirements of the patient for whom it is required.

5.4.5.2 Capturing Reusability, Collaboration, and Consistency

As described in the section above, on the advantages of having a formal care plan modelling framework, the main motivations involve enhancing the reusability, collaboration and consistency of health information that is applied to patient's suffering from chronic illnesses. This section looks at each of the three motivations and discusses how the care plan modelling framework can achieve this through its structure.

5.4.5.2.1 Reusability

Research shows that adverse events in health care are relatively frequent especially so in cases where the patient is expected to monitor themselves, research shows that preventability of these effects is relatively easy and the main cause of these problems are omissions in advice and important aspects of clinical guidelines (Beyer, Kuhn et al. 2004).

There may indeed be many reasons for omission related errors, however most more often than not the omissions are a result of simple human error caused due to dealing with health care information that is inherently high in complexity (Beyer, Kuhn et al. 2004). The preventability of omission related errors is high; many solutions such as reminders, alerts, checklist etc have been suggested. Other solutions that have been suggested have focused on reducing the complexity of health care information, suggesting solutions such as "structured data entry" (Beyer, Kuhn et al. 2004). An added advantage of approaches that focus on structuring data is that the data becomes more accessible for reuse.

Therefore the aim of the care plan modelling framework was to capture a generic structure into which information from clinical treatment guidelines could be moulded, hence making the resulting care plan models reusable for many patients suffering from similar ailments, therefore reducing the repetitive effort of extracting and utilizing complex information from clinical guidelines, hence reducing the chances of errors.

The different ways in which components in a care plan model can be reused include:

1. **Reusing entire care plans for multiple patients**. For example John's GP may have several patients that are suffering from obesity, and once a care plan for obesity has been designed, the majority of it can be readily utilized to support other patients suffering from obesity. Furthermore it can be seen in the scenario, that the obesity management care plan is composed of many other sub-care-plans, for example physical activity care plan, or the diabetes monitoring care plan, it could be that John's GP has other patients that suffer from illnesses other than obesity, for example high blood cholesterol, in such cases it may still be necessary for these patients to monitor their diabetes, to increase their physical activity, therefore the GP could in effect even reuse these sub-care plans to produce new care plans for his other patients

2. **Reusing health care activities**. On a finer level of granularity in the care plan model, even action components, such as health care activities can be reused to compose other care plans. For example when the GP, is building a care plan for exercise for an overweight patient, he can use the activities in the physical activity care plan designed for John, and then add new activities to form a completely new and updated care plan module.

3. **Reusing assessment modules**. A care plan may have as part of it one or more assessment modules, for example, the obesity management care plan, may have two assessment modules, the first being a weight assessment module, the second being a diabetes assessment module, it is easy to see how each of the assessment modules could be used in other related care plans, for example the diabetes assessment module

could be used in a diabetes care plan, likewise the weight loss assessment module could be used for not only in a diabetes care plan, but also in a care plan focused at high blood cholesterol patients.

5.4.5.2.2 Collaboration and Consistency

On studying the obesity management scenario, it can be observed, that in order to treat John's obesity, many different approaches have been applied at once, for example, physical therapy, behavioural therapy, dietary therapy, pharmacotherapy, etc. It may be that all forms of treatment are prescribed by John's GP himself, however it is also possible that other health care professionals of other disciplines are brought in to help John, for example, a dietician may design John's dietary therapy plan, well as a psychologist may design John's behavioural therapy.

One of the advantages of having treatment guidelines in a structured form, besides that of reduced complexity and reusability, is that it introduces a degree of free form transformability, what is meant by this is that theoretically the dietary therapy care plan from the "super" obesity management care plan could be transferred over to the dietician to extend with their own knowledge, and likewise the behavioural therapy sub-care-plan could be transferred over to a psychologist to extend with specialist stress relief activities. Therefore it can be seen that the care plan modelling framework can in effect aid collaboration between multi-disciplinary health care professionals, an extended advantage of this being richer more effective treatment care plans.

Furthermore an overall advantage of structuring, reuse, and collaboration of health care information in such a formal model based manner, is that it ensures a certain degree of consistency of treatment across not only patients suffering from similar illnesses, but also across different health care professionals treating the same ailment.

5.5 Summary

This chapter discusses the care plan domain model which will be used in the visual care plan modelling language, and which is one of the central needs of the health care professional caring for patients (see previous chapter for more). The main advantages of having a formal care plan domain model are being able to reuse health care information, making it easier to collaborate on the health care information, and also introducing a consistency in the way health care is administered.

According to the care plan domain model discussed in this chapter, a care plan is a composite of health performance goals, and metrics, health activities, assessment modules and other sub health care plans. Performance metrics or goals in a care plan describe the optimum, or ideal health states that the patient needs to aspire to attain, and typically a health care plan may have multiple performance metrics, in the example discussed above for John's obesity management care plan, two performance metrics relevant may include the body mass index and the waist circumference.

Health care activities are the actual action component of a care plan, which specifies the tasks that a patient needs to do in order to attain their health goals specified by their performance metrics. Health care activities similar to care plans are composite objects which are composed of three main components including instructions, resources and routines. Instructions that are part of a health care activity tell the patient how a health care activity should be conducted, for example doing warm-up exercises before going for a jog, resources tell the patient what they need before they can start the activity, and routines that are part of an activity tell the patient how often and for how long the activity needs to be conducted. There are three main types of health care activities part of the care plan domain model, the first is a simple task, such as jogging, or going for a swim, the second is a review activity, which requires a health care professional and a patient to participate in the activity at once, the third and final type of activity is a data collection activity, this activity requires a patient to measure some aspect of their health, for example the glucose measurement activity part of a diabetes care plan, which would require a patient to measure and record their blood glucose sugar.

Health care plans also consist of assessment modules. The purpose of assessment modules is to allow the health care professional to encode as part of the care plan a decisional task flow which describes an assessment algorithm guiding the health care professional on how a patient on a care plan should be assessed. The components of the assessment module (or assessment task flow), include an a conditional clause which allows the encoding of questions that need to be answered about the patient's health, the second component is an assessment component, which includes assessment and actual examinations that are to be performed on the patient, and the final component in an assessment module is a treatment recommendation, guiding action that the patient and the health care professional need to take in order to treat the patient. The components in the assessment module are related to each other either by a task flow sequential relationship, or a conditional relationship.

The structure and syntax of the care plan domain model mean that each part of the care plan can be reused to design other care plans, this includes the reuse of entire care plans, or the individual reuse of components such as performance metrics, health activities, or assessment modules. A care plan can exist in either one state a patient independent and disease specific template, or a patient specific and disease specific care plan instance.

Chapter 6 - Functional and Non-Functional Requirements

6.1 Introduction

The purpose of this chapter is to discuss the detailed functional and non-functional requirements for the care plan modelling system as dictated by the key stakeholder needs discussed in chapter 4.

6.2 The Overall Approach

Chapter 5 discusses the formal care plan model which can be used to capture the treatment regimes for patients as prescribed by their health care professional. The focus of this chapter is to discuss the requirements of the care plan modelling system which can be used to model health care plans, tailor them for patients and then deploy them onto electronic devices such as mobile phones for patients to use. The overall approach of the care plan modelling system can be observed in the cyclic process shown in figure 33 below.

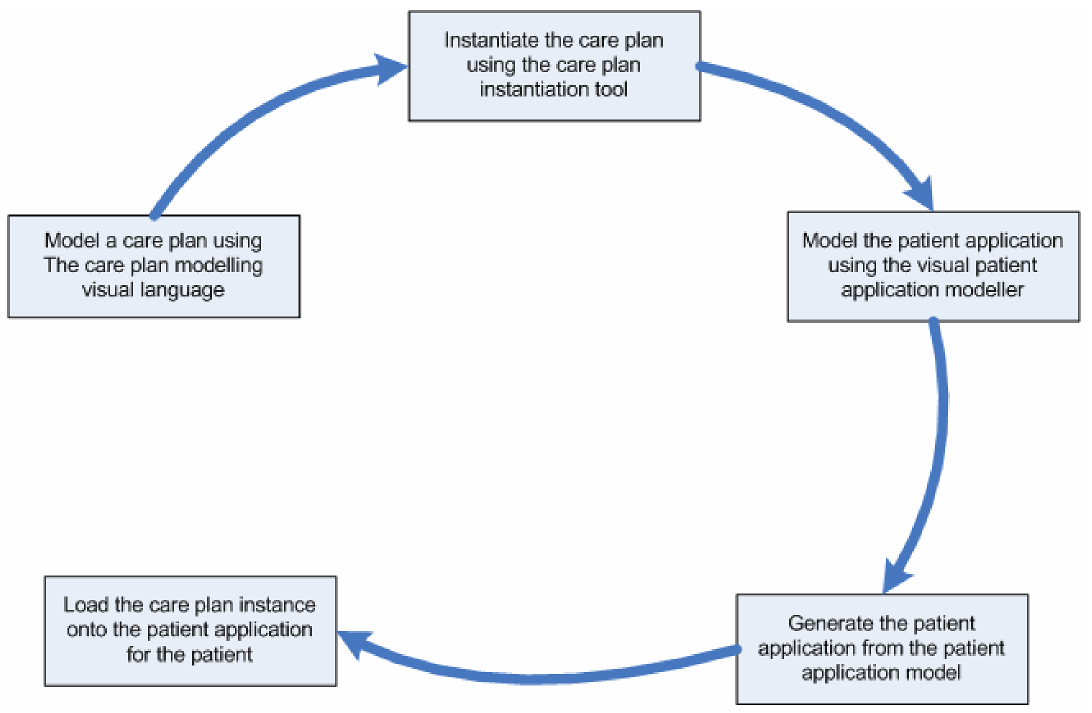

Instantiate the care plan using the care plan instantiation tool
Model the patient application using the visual patient application modeller
Generate the patient application from the patient application model
Load the care plan instance onto the patient application for the patient
Model a care plan using The care plan modelling visual language

Figure 33: Diagram showing the cyclic process behind the care plan modelling system.

From the above diagram it can be seen that the cycle in the care plan modelling system begins with the health care professional modelling a care plan for a patient from their knowledge of treatment guidelines, then this care plan is made more specific to the patient through the care plan instantiation tool, then in order to deploy this care plan to the patient so that they can use it, a designer is asked to model a patient application for the patient depending on the electronic device that they posses (i.e. desktop, PDA, mobile phone etc.), resulting in a definition of how the patient application is to look representing the care plan instance for the patient on their electronic device. After this model of the patient application is designed then the patient application is generated, and the care plan instance tailored for the patient can be loaded onto the generated application and given to the patient to use.

From this cycle it can be noted that the care plan modelling system is made up of:

1. Visual language to model health care plans according to the care plan structure discussed in chapter 5 above.
2. Care Plan Instantiation tool to fine tailor the care plan templates designed from the visual language above to match the patient more closely.
3. Patient application which is suppose to represent the care plan instance to the patient allowing them to interact with their care plan and receive the support they need, as specified in the user needs chapter 4.
4. Visual patient application modeller, this tool would allow a designer to specify how a care plan instance should be graphically represented to the patient in the form of a patient application. Furthermore this visual toolkit includes a generator which generates the patient application based on models outputted by the visual patient application modeller.

The following sections discuss the functional requirements of each part of the care plan modelling system whilst relating the functional requirements to the stakeholder needs expressed in chapter 4 earlier.

6.3 Functional Requirements

The following section documents the functional requirements of the care plan modelling system in the form of four main use cases, each use case corresponding to a part of the care plan modelling system, the first being the visual care plan modelling language, followed by the care plan instantiation tool, and the patient application, and finally the visual patient application modeller. For each use case the purpose of the use case, the input, output, stakeholder, the scenario and a detailed description in the context of the scenario is provided.

6.3.1 Use cases for the care plan modelling visual language

6.3.1.1 Purpose

The purpose of this use case is to portray the type of functionality that is exposed through the visual care plan modelling language, to allow health care professionals to use their knowledge on treating patients and convert this knowledge into reusable formal health care plans.

6.3.1.2 Inputs

The input into this use case is the actual knowledge which the health care professional possesses or extracts from clinical treatment guidelines in order to model health care plans for patients suffering from some illness.

6.3.1.3 Outputs

The output of this use case is the actual care plan which is modelled by the health care professional using the visual care plan modelling language. This care plan is represented in XML form.

6.3.1.4 Stakeholders

The main stakeholder for this use case is the health care professional, where the term health care professional can extend to a variety of health professionals such as general practitioners, physiotherapist, dietitians, fitness experts etc.

6.3.1.5 Scenario

A GP by the name of Dr. Sally has a patient suffering from obesity, i.e. John, for the purpose of this use case, Sally needs to build an obesity management care plan for John, in the form of a care plan template which can be used again for other patients suffering from obesity.

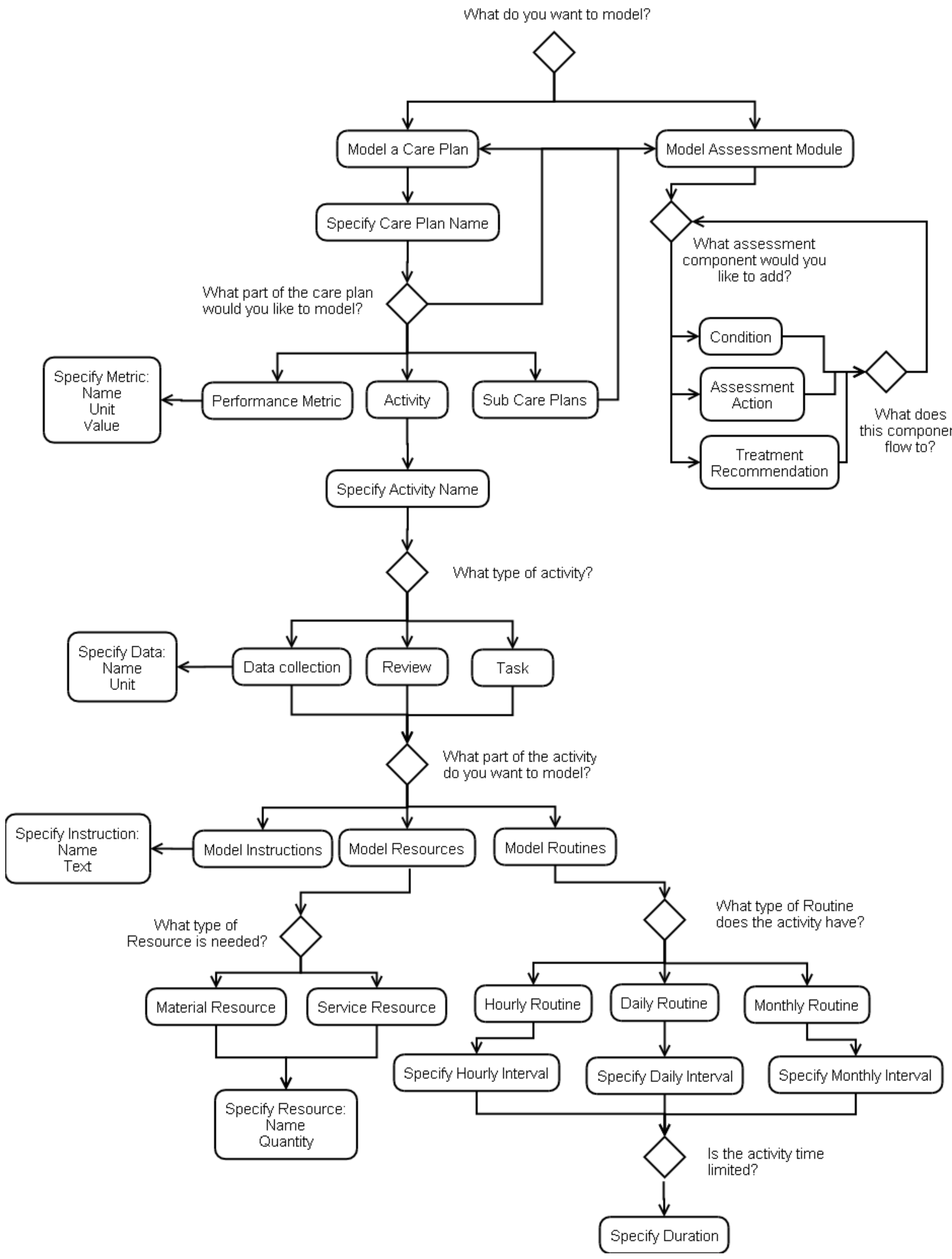


Figure 34: An activity diagram showing the use case for the visual care plan modelling language.

6.3.1.6 Description

Figure 34 above shows the use case for modelling health care plans using the visual care plan modelling language. Using the visual care plan modelling language Sally can choose what she wants to model first, she can either start modelling the obesity management care plan, or start to model an assessment module (or a treatment algorithm). It is assumed that Sally starts to model the obesity management care plan. After naming the care plan “obesity management” she has a choice of how she wants to extend this care plan, she can add more sub care plans, such as dietary therapy, physical therapy, diabetes monitoring etc, or she can add health care activities, or performance metrics, or once again assessment algorithms. Sally starts of by modelling two performance metrics in order to provide John with a clear set of goals to follow, she models a body mass index metric and a waist circumference metric, for each she needs to provide the units for the metric values, the name of the metric value and the metric value it self.

After this Sally decides to extend the obesity management care plan by adding sub care plans such as physical therapy and diabetes monitoring care plan, Sally then extends the physical therapy care plan to contain tasks such as jogging and swimming, whilst she extends the diabetes monitoring care plan with data collection activities such as the glucose measurement activity. For the diabetes monitoring activity Sally needs to provide the name and unit of the data that John needs to collect, i.e. “blood glucose” and “mmol/L”. For each health care activity she can also choose to provide a list of instructions, as well as resources that John will need, for example he needs a testing meter, testing strips, and a prick pen for measuring his blood glucose levels, and he has instructions to discard the needle of the prick pen after every 10 uses, in the case of the obesity review activity the resource needed may be a service resource instead of a material one for example need of nurses, with the instruction to John to fast before the consultation. For each of the instructions sally needs to provide a name and the actual instructions, and for the resources she needs to provide the name and the quantity required, for example two testing strips.

Sally also adds an obesity review activity underneath the “super” obesity management care plan. For each of the health care activities Sally also needs to specify routines, which tell John when and how often he should do the activity, she sets a monthly routine for the obesity review activity setting an interval of 3 months, and a duration of 120 minutes, i.e. John has to come in for a review every 3 months for 2 hours, similarly she sets a daily routine for the jogging and swimming activity with an interval of 2 days, and a duration of 30 minutes, saying that he must try to jog and swim every 2 days, for 30 minutes.   After all this has been modelled Sally may then finally choose to model an assessment algorithm which she can use to review John’s obesity during the review sessions, this is in the form of an assessment module. She models conditions, assessment actions, and treatment recommendations, based on the treatment algorithm she refers to from the treatment guideline for obesity as shown in chapter 5.

6.3.2 Use cases for the care plan instantiation tool

6.3.2.1 Purpose

The purpose of this use case is to specify how the care plan instantiation tool can be used to tailor a health care plan for the patient. As expressed in chapter 4, this is an important part of ensuring patient adherence to the health care plan, as a patient is more likely to follow a health care plan that closely maps their personal needs and lifestyle.

6.3.2.2 *Inputs*

There are two main inputs that are required into this use case the first input is the care plan template which the health care professional has designed (from the previous use case, using the visual care plan modelling language), this template is required in XML format, the next input that is required is information about the patient's daily routines, lifestyle and preferences, these are assumed to be collected by the health care professional during periods of discussion prior to the execution of this use case.

6.3.2.3 *Outputs*

The main output of this use case is the care plan instance, this care plan instance is essentially the same as the care plan template inputted into the use case (or the care plan instantiation tool), however, it has been modified, and catered to the patient's preferences. This care plan instance outputted will be in XML.

6.3.2.4 *Stakeholders*

There are two main stakeholders for this use case, firstly there is the patient who benefits from the care plan instantiation tool through having the care plan template personalized for them through the application, the second stakeholder is the health care professional who is the direct user of the system using the system to personalize the care plan template for the patient.

6.3.2.5 *Scenario*

For this scenario again we use the same two characters as the previous scenario, i.e. Sally the general practitioner and John her patient who suffers from obesity. For this scenario it is assumed that Sally has already designed an obesity management care plan for another patient, as described in the use case above, and she decides to reuse this care plan for John, retrieving the saved obesity management care plan from a repository.

6.3.2.6 *Description*

The use case for the instantiation process can be seen in figure 35 below, Sally starts of the instantiation process by retrieving an obesity management care plan template she had previously designed through use case from a repository. This care plan template although intended for all patients suffering from obesity is however largely generic and is not fine tailored to John's personal needs. In order to tailor this generic care plan template to suit John better, Sally needs to do two things, firstly she needs to go through the care plan template and determine aspects of it that are appropriate for John, or in appropriate for him, and indeed elements that are needed for John and are missing from the generic template. Furthermore Sally needs to set up detailed schedules for the health care activities part of the template.

Sally starts the instantiation process by loading the obesity management care plan template into the care plan instantiation tool. Through this tool sally can view all the details of the obesity management care plan, as well as being able to edit all parts of the care plan excepting the assessment modules (which are not meant to be instantiated for the patient). On examining the obesity management template through the tool, Sally can observe that the template has three care plans a behavioural therapy care plan. A physical therapy care plan, and a dietary therapy care plan. Sally may feel that John is in no need of behavioural therapy and hence may deselect this part from the care plan template, furthermore from the above use case it can be noted that the physical therapy plan has a swimming and jogging task, however John says he can swim, and hence Sally deselects the swimming activity from the physical therapy plan leaving only the jogging task. Like wise Sally may add a diabetes monitoring plan as a sub plan in the obesity management plan through the tool itself, although this can be done through the visual language in the use case above.

Once Sally has ensured that the obesity management plan contains only the parts that are relevant to John, she can now start setting detailed schedules for the health care activities that are a part of the care plan. Depending on the type of routine modelled for the activity in the first use case, Sally needs to set the schedules for them. The jogging activity for example has a daily routine, with an interval of 2 and a duration of 30 minutes, in that do the activity every 2 days, for 30 minutes, for this type of activity routine, sally needs to set the preferred times in the day when John should do these activities, with the preferred time matching John's personal life routines, for John this may be at 7 pm in the evening after work, or 7 am in the morning before work. Similarly for the obesity review activity, which is to be done every 3 months for 2 hours, Sally can set a preferred day for this review, and on that day a preferred time for the review. If the health care activity has an hourly routine, for example John may need to take a medication every 6 hours, then Sally does not need to set a detailed schedule, because an hour is already on a very low time granularity. The health care activities that are prescribed to John are not to be conducted indefinitely, therefore Sally sets for each activity an activation start date and end date, for example John may start taking a medication at the beginning of April, and stop at the end of May. Once Sally completes these tasks of personalisation then the obesity management plan template is now personalised to John and ready for deployment.

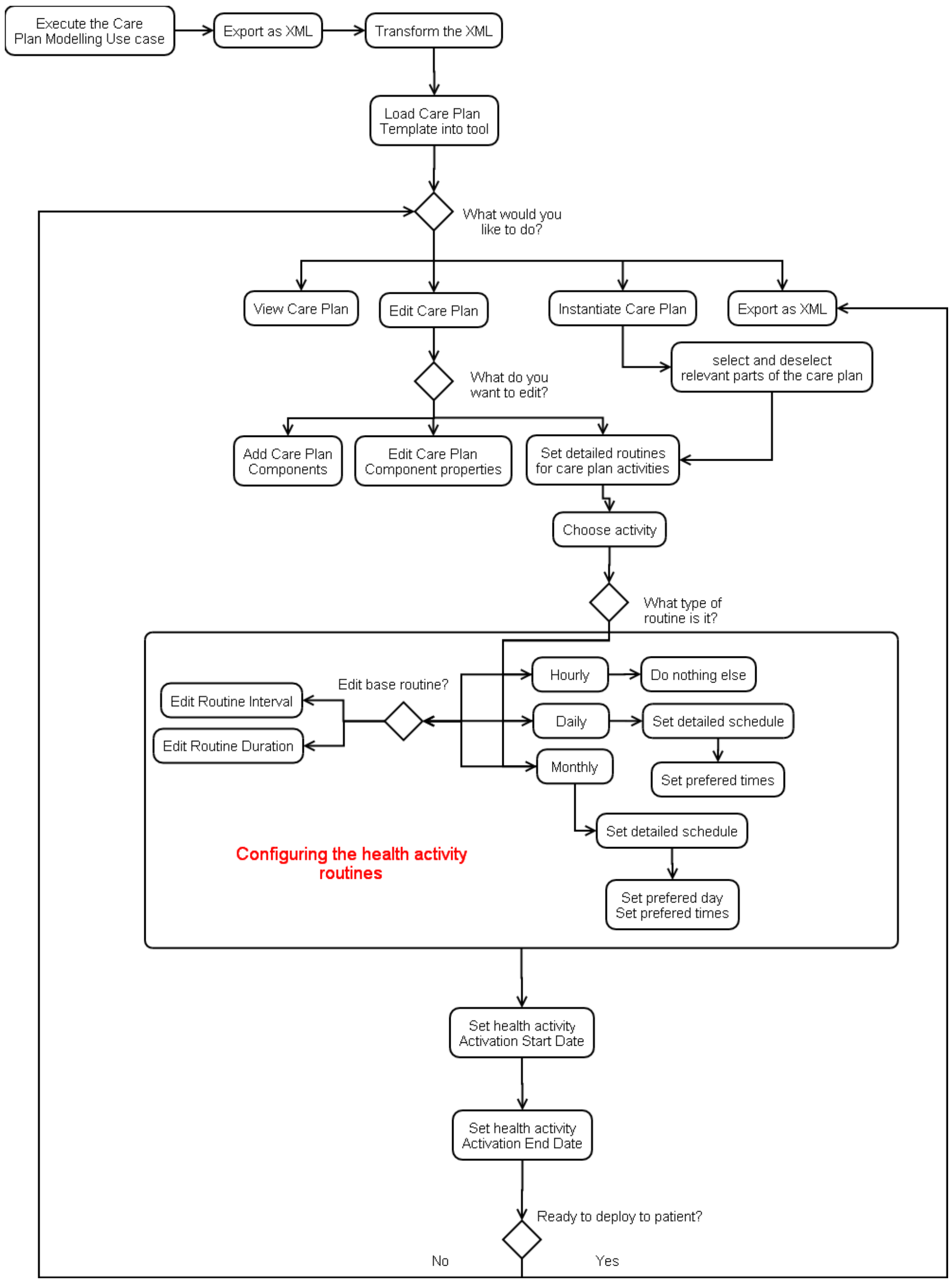

Execute the Care Plan Modelling Use case
Export as XML
Transform the XML
Load Care Plan Template into tool
What would you like to do?
View Care Plan
Edit Care Plan
Instantiate Care Plan
Export as XML
select and deselect relevant parts of the care plan
What do you want to edit?
Add Care Plan Components
Edit Care Plan Component properties
Set detailed routines for care plan activities
Choose activity
What type of routine is it?
Edit base routine?
Edit Routine Interval
Edit Routine Duration
Hourly
Do nothing else
Daily
Set detailed schedule
Monthly
Set prefered times
Set detailed schedule
Configuring the health activity routines
Set prefered day Set prefered times
Set health activity Activation Start Date
Set health activity Activation End Date
Ready to deploy to patient?
No
Yes

Figure 35: Activity diagram showing the use case for the care plan instantiation tool.

6.3.3 Use cases for the patient application

6.3.3.1 Purpose

The purpose of the patient application is to support the patient directly in the execution of the care plan personalised for them in the previous use case. Through this application the patient can get helpful reminders for appointments they need to do to combat the forgetfulness problem, enter in data, comments, and mark the status of the activities that they have to do, in order to allow health care professionals to monitor the patient's adherence to their health care plan, as well as to help the patient communicate efficiently with the health care professional during consultations.

6.3.3.2 Inputs

The main input into this use case is a patient tailored care plan instance which can be loaded into the patient application in XML form.

6.3.3.3 Outputs

There are no direct outputs of this use case however the patient data that is entered, as well as patient comments for activity appointments etc is sent to the health care professional when the patient synchronises their executed care plan with the remote health servers.

6.3.3.4 Stakeholders

The main stakeholder for this use case is the patient, through the patient application the patient can interact with their care plan, being able to view their care plan, enter in health activity related data, as well as receive reminders and health alerts for their health related activities.

6.3.3.5 Scenario

Once again we use the same characters as for the above use case, i.e. John the patient suffering from obesity and Sally his general practitioner. In this use case John has been given a personalized obesity management care plan instance in the form of an XML document and loads it onto a patient application to use.

6.3.3.6 Description

Once Sally has created for John a personalised care plan instance of the obesity management plan, then this plan is ready for John to use. John loads this care plan instance onto the patient application on his PDA, the use case diagram in figure 36 shows the functionality of the patient application. Once the plan is loaded, John can view his entire care plan, including the multiple sub care plans that he is on, for example the physical therapy plan, the diabetes plan etc, in addition to this John can view all his health care activities, their instructions, resource needs etc, John can also view his performance metrics, for example Body Mass Index. John can also receive reminders from patient application for the conduction of health care activities, for example John may get a reminder to go for a jog, the patient application will then navigate to the activity page, where John can decide to view the resources needed for him to do the activity, and the instructions related to the activity. John may also receive a reminder for a data collection activity for glucose measure as part of his diabetes monitoring plan. Since John is required to collect data, the appointment screen for that activity will automatically have the fields for the data, and John can enter in the data and save it. If the reminder is received for a review activity, for example the obesity review, the conduction is exactly the same as the simple task. Regardless of the type of activity appointment, once the activity has been completed by John, he can mark the activity completed, he can also add comments to the appointment, if John cannot go for a jog he can mark the appointment as not attempted, if he goes for a jog and can't complete it, then he can mark it in complete, and specify reasons in the comments area. After conducting the care plan for a period of time John can send his updated care plan with his data to the remote health servers where Sally may look at it and give him feedback possibly modifying his plan.

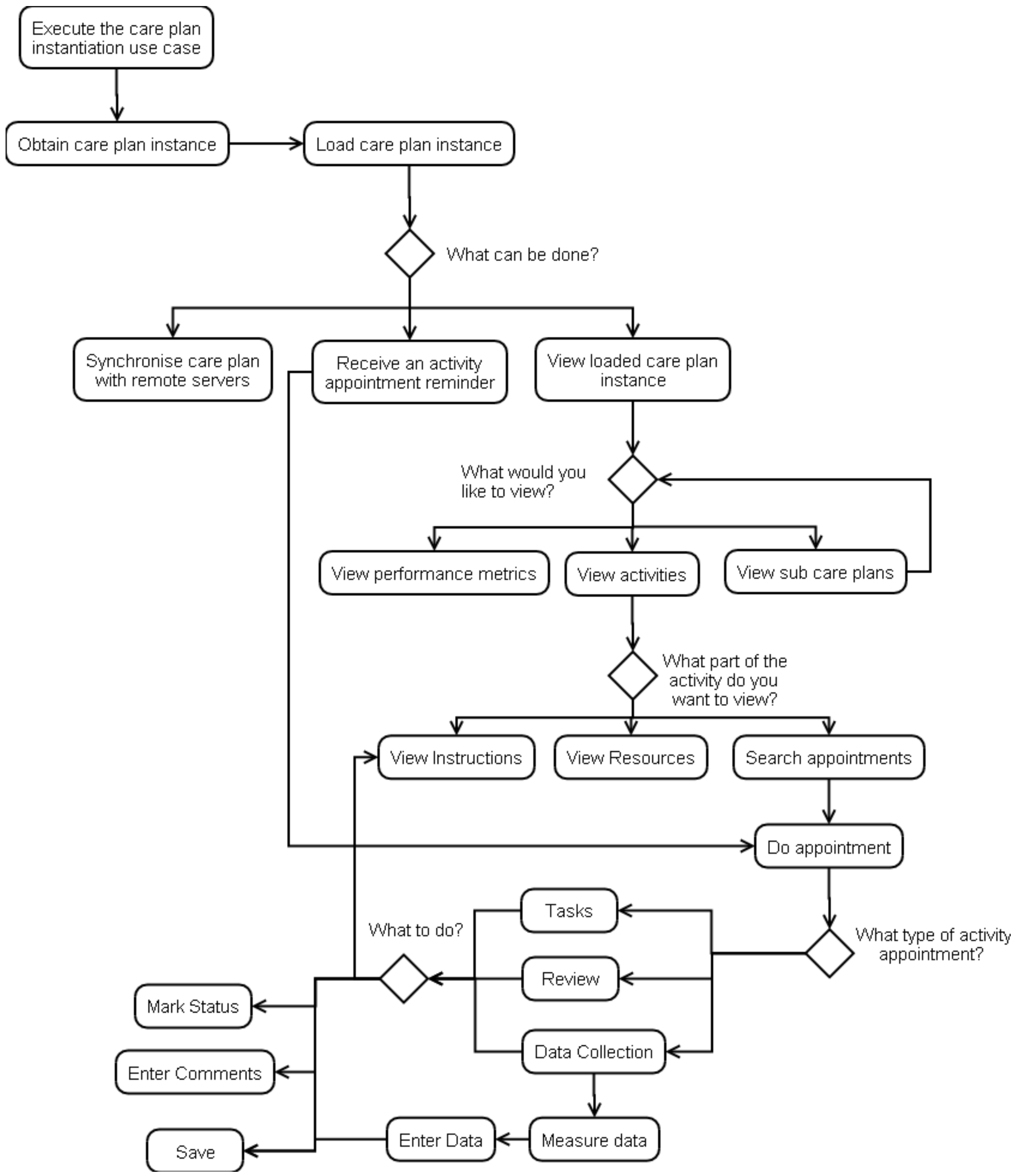

Execute the care plan instantiation use case
Obtain care plan instance
Load care plan instance
What can be done?
Synchronise care plan with remote servers
Receive an activity appointment reminder
View loaded care plan instance
What would you like to view?
View performance metrics
View activities
View sub care plans
What part of the activity do you want to view?
View Instructions
View Resources
Search appointments
Do appointment
What type of activity appointment?
Tasks
Review
Data Collection
What to do?
Mark Status
Enter Comments
Save
Enter Data
Measure data

Figure 36: Activity diagram showing the use cases for the patient application.

6.3.4 Use cases for the patient application visualization language

6.3.4.1 Purpose

The purpose of the visual patient application modeller is to allow a GUI designer to design a patient application for the patient as a high level definition, and to do this for different electronic devices such as desktop computers, mobile phones, PDAs etc., with the generator accompanying this visual language using the models to generate the patient applications in Flash. The main reason for this visual language is to target the non-functional requirement for the patient, which dictates that a patient may have many types of electronic and computing devices, and hence a patient application should be available to run on most of them (see section below on non-functional requirements).

6.3.4.2 Inputs

There are no inputs required here except the GUI design knowledge of a designer, as well as knowledge of the device for which the patient application is being designed, i.e. the characteristics of the device such as screen size, type etc. Also needed is the knowledge of the structure of the care plan instance that will be loaded into the patient application designed.

6.3.4.3 Outputs

The output of the visual patient application modeller is a definition of the patient application; this definition can then be fed into the patient application generator which will in turn generate the code for the patient application in OpenLaszlo, which is then compiled to Flash. The patient application definition is device specific (the type of device that the designer is aiming to model the patient application for) and is in XML format.

6.3.4.4 Stakeholders

There are two stakeholders for this use case; one stakeholder is indirect and one a direct stakeholder. The patient is the indirect stakeholder, benefiting from the visual patient application modeller through being able to run the patient application on whatever mobile device they posses. The direct stakeholder for the visual patient application modeller is the graphical user interface designer whose responsibility it is to design the patient application for the different types of mobile devices (PDA, smart phone, mobile phone etc.) that the patient may posses.

6.3.4.5 Scenario

The scenario for this use case includes three people, the first is Sally the general practitioner, the second is John, Sally's patient suffering from obesity, and then there is Tom the graphical user interface designer whose job it is to build a user interface for the patient application.

6.3.4.6 Description

In the above use case it was mentioned that John has a PDA, and the patient application use case was defined with for a PDA application, however it is not necessary that John has a PDA, he may instead posses a mobile phone, or a smart phone hence the patient application may need to be re-implemented for other types of devices. It is to this need that the patient application modeller has been designed. Tom the graphical user interface designer in our example can use the patient application modeller to design a model of the patient application specific for different types of devices, and through this model the patient application can be generated in Flash, the use case for the patient application modeller can be observed in figure 37 below.

In order for Tom to start designing the patient application he needs two items of information, these include firstly the schema of the care plan instance document, this tells Tom how the care plan instance data is structured. The second item of information required by Tom includes knowing about the characteristics of the mobile device that John possesses, i.e. screen size, GUI capabilities, etc. Once the Tom knows these two items of information, he can decide how the care plan instance data is to be represented graphically in the patient application suited for a particular type of device. Now Tom needs to start modelling the patient application. For the purpose of the example it is assumed that John has a smart phone instead of a PDA.

As shown in figure 37 Tom needs to model two aspects of the patient application, the first is the data structure that the patient application uses, this comes from the care plan instance XML document schema, next Tom needs to model the GUI for the patient application, this needs to be done with the characteristics of the mobile device in mind. Finally Tom needs to map the GUI of the patient application with the care plan instance data model elements.

It is assumed that Tom begins with modelling the care plan instance data model, this he does using knowledge given to him about the care plan instance XML schema. Using the patient application modeller Tom can recreate the necessary parts of the schema within the patient application model, through the usage of objects called data references. Data references are the equivalent of the element object in an XML schema; they can contain child elements, which may be other elements or attributes. Once Tom has built the care plan instance XML data model next he needs to model the GUI to represent the care plan instance data he has modelled.

Tom can start building the GUI using a variety of GUI components part of the patient application modeller, he needs to use the information on the characteristics of the mobile device that John possesses and use to decide on the properties of the GUI components he decides to model. Depending on the type of GUI components some GUI components may be composed of other GUI components, Tom may need to use this feature to split up the screens for the patient application into smaller screens or composite GUI components so that the GUI will fit onto John's smart phone. While modelling GUI components, Tom will also need to specify the type of these GUI components, for example labels in a GUI are inherently of display type as they will probably only display care plan instance data mapped to them, well as textfields on the other hand are input components and will take in input to update against the care plan data mapped to them. Like wise Tom may include some GUI components in the model which are of navigate type and navigate to other GUI components, or alternatively are of update type and update the data from input components to the underlying data mapped to them.

Once Tom has modelled the GUI components of the patient application, then next he needs to map the relevant care plan instance data model elements to the GUI components that are suppose to represent them, once Tom has completed this data to GUI mapping then the resulting model can be used to generate the patient application. It is important to realise that Tom may not be present as an active stakeholder in the care plan modelling system, it is very likely that Tom may be asked to construct one of models for patient applications for a variety of mobile devices, and that these models are stored in persistence and reused for many patients.

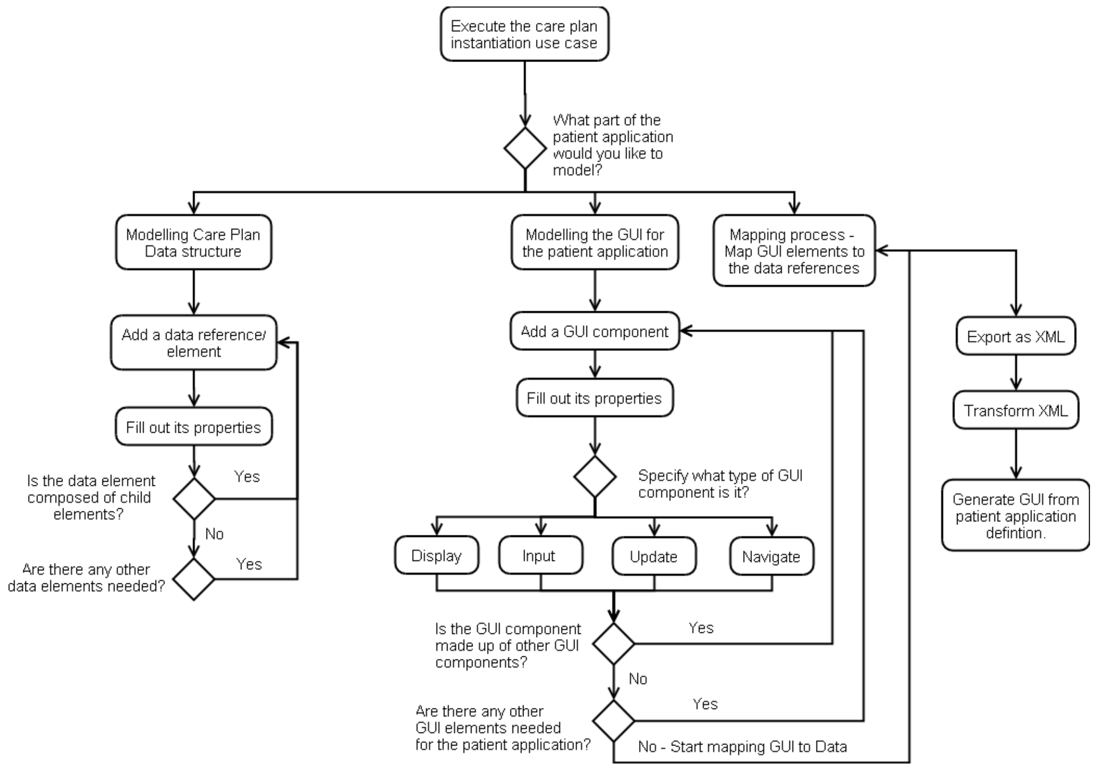

Execute the care plan instantiation use case
What part of the patient application would you like to model?
Modelling Care Plan Data structure
Modelling the GUI for the patient application
Mapping process - Map GUI elements to the data references
Add a data reference/ element
Fill out its properties
Is the data element composed of child elements?
Yes
No
Are there any other data elements needed?
Yes
Add a GUI component
Fill out its properties
Specify what type of GUI component is it?
Display
Input
Update
Navigate
Is the GUI component made up of other GUI components?
Yes
No
Are there any other GUI elements needed for the patient application?
Yes
No - Start mapping GUI to Data
Export as XML
Transform XML
Generate GUI from patient application defintion.

Figure 37: Activity diagram showing the use case for the visual patient application modeller.

6.4 Non Functional Requirements

Along with the functional requirements of the care plan modelling system discussed above there are also a whole range of non-functional requirements for the system. What follows is a discussion on what the non-functional requirements are for each part of the care plan modelling system.

6.4.1 Visual care plan modelling language

There are indeed many non-functional requirements for the visual care plan modelling language, however the following were considered to be the most important:

1. Usability
2. Scalability

6.4.1.1 Usability of the visual care plan modelling language

The usability of the visual care plan modelling language is one of the most important non-functional requirements of the care plan modelling system; the reason for this is that it allows the health care professional to get most out of the system and the language. The usability requirement for the visual language can be roughly divided into two main aspects the first aspect is ease of use and convenience and the second aspect is flexibility of visualisation.

Health care professionals are very busy people (in particular general practitioners) hence for them to use the visual care plan modelling language the language has to be easy to use and package in a great deal of convenience so that they can achieve the modelling of care plans with ease. One of the main ways in which this ease of use can be achieved in a visual language is through careful consideration of the visual notation that is used for the language, in that the visual metaphor used in the graphical notation of the visual language needs to be a close map to the care plan modelling paradigm. This in turn makes the language intuitive and hence easier to learn and use. Other ways of improving the usability of the language, is to include facilities by which the health care professional can automate some of the functions and modelling tasks that they regularly do, for example automatically adding routine components to all activity components etc. (see the following chapter for more details on the design choices made for the development of the visual care plan modelling language).

The other part of usability is to provide health care professionals the flexibility to visualise the same care plans in different ways, the main reason for this requirement is that the health care professional may be many different types of professional, from a GP, to a dietician, to a physiotherapist or a social worker. Each professional may have a different way to grasp the health care plan knowledge and hence may prefer different types of visualisations for the same care plan (for example visualising the same care plan at different levels of detail and granularity).

6.4.1.2 Scalability of the visual care plan modelling language

There are many things that scalability can mean when talking about the visual language, the first and most obvious meaning that can be extracted is related to how easily the same visual language can cope with the modelling of increasingly larger and more complex care plans. Care plans by their very nature are complex, they usually are derived from a mixture of knowledge sources spanning from the knowledge stored in the health care professionals mind to the mining of the knowledge embodied in the two hundred page long clinical treatment guidelines, hence it is a desirable requirement for the visual language to be viable for small care plans as well as to model more complex scenarios inherent of larger care plans. This type of non-functional requirement can be fulfilled in two different ways, the first being in the design of the syntax or grammar of the language so that it is extensive enough to handle the full complexity of expression expected from a decent sized care plan (see the care plan domain model in the previous chapter), the second way to achieve this type of scalability is to again consider efficient graphical notations and visual metaphors to augment the syntax of the visual language.

The second meaning of the scalability that can be seen for the care plan modelling visual language or for any visual language for that matter is how easy it is to extend the visual language to add greater functionality.

6.4.2 Patient Application

The non-functional requirements of the patient application for the care plan modelling system are with regard to the patient application which is intended to provide them support during their treatment and or management of their ailments; in particular these requirements are related to the mobile device usage for hosting the patient application. The non-functional requirements for the patient application can be roughly divided into three different criteria:

1. Security of the patient information
2. Hardware and technology availability
3. Internet connectivity and bandwidth

6.4.2.1 Security of patient information in the patient application

The patient application is expected to hold all the information related to the patient care plan, as well as the information that is collected about the patient in the form of adherence monitoring and prescribed data collection activities. If a patient is to use the patient application they need to have confidence that the information that they enter into the patient application is completely secure and only accessible to authorised personnel, furthermore the patient application also needs to ensure that the transfer of information from the patient application hosted on the patient's mobile device to the remote health servers is done in a secure manner. Despite the measures that can be taken to secure the patient information contained in the patient application on the patient's mobile device, there are still possibilities that the information can be viewed by undesirable people, and this risk is exposed simply due to human error, the patient may leave their mobile device lying around or lose it.

6.4.2.2 Hardware and technology availability

How successful the idea of the patient application is largely dependent on how readily the patient can utilise the patient application, and whether a patient can utilise the application depends on what type of mobile device the patient application has been designed for, this means that if the patient application is designed for PDAs then it will not function on a mobile phone and if it has been designed for a desktop computer then it most certainly will not work on a PDA. The main reason for this is that the patient application due to being a fat client (see next chapter for more detail) is prone to being platform and device dependent, which means that if the patient does not posses the mobile device the patient application is intended for then they will not be able to use the application. Therefore one of the most important non-functional needs of the patient application is to be able to run on most mobile devices whether they are mobile phones, PDAs, smart phones, desktops or tablets. In fact one of the main motivations of the visual patient application modeller, is to allow the designers of the patient application to easily define the look, feel and layout of the patient application for different types of mobile devices with the patient application being generated from these look and feel definitions in Flash, a format able to run on most mobile devices (see next to chapters for more details).

6.4.2.3 Internet Connectivity and Bandwidth

It is mentioned above that the patient application collects patient data in the form of prescribed data collection activities, and adherence monitoring information, and one of the requirements that is a side effect of this collection is that the data has to be synchronised to the remote health servers so that the health care professional can view and process this data. There in lies the challenge, as most mobile devices cannot guarantee perfect connectivity to the internet all the time, and if there is connectivity to the internet, the bandwidth cannot be guaranteed, one of the non functional requirements of the patient application is to allow the patient to have local storage on the mobile device the patient application is running on, at least until the patient is in a zone where there is connectivity so that the data collected as part of the application can be securely synchronised with the remote server (for more details of the architecture that is used to satisfy this non-functional requirement see the following chapter).

6.4.3 Visual patient application modeller

Similar to the visual care plan modelling language, the visual patient application modeller also has two main non-functional needs, i.e. usability and scalability.

6.4.3.1 Usability of the visual patient application modeller

The purpose of the visual patient application modeller is to allow a designer to model the graphical user interface of the patient application and to relate this interface with the data structure of the care plan instance which it is suppose to interact with. Hence usability for the visual patient application modeller corresponds to having an interface for modelling graphical user interfaces as easily as possible, with the most important aspect of usability being providing the designer with a good preview of the GUI that they are designing. The main reason for this requirement is to reduce the mental load of the designer, who without a WYSIWYG preview based GUI designer may have to do a substantial amount of work designing the GUI on paper before using the visual patient application modeller (although for the prototype designed this was not possible, see the next chapter for the design decisions made). Other usability needs are once again similar to the visual care plan modelling language, in that there should be facilities where by the designer can package common functionalities into single actions (for example macros) in order to make modelling simpler and quicker.

6.4.3.2 Scalability of the visual patient application modeller

In order for the visual patient application modeller to be scalable it would need to be able to cope with the modelling of graphical user interfaces with different levels of complexity, one level of complexity includes the variety of GUI elements that are possible to model into the visual patient application modeller, the greater the variety the more complex and different types of GUIs can be built. In addition to this, it can be seen from the use case of the visual patient application modeller shown in figure 37 above that the only type of components that are allowed are of input, update, display, and navigate type. In a more complex graphical user interface there may be a need to design GUI components with other types of interactions with the care plan instance, and hence it is necessary that in order to make the visual language scalable to modelling larger and more complex screens, the modelling of complex interactions besides the four indicated in the use case for the proof of concept prototype would be needed.

6.5 Summary

This chapter discussed the functional and non-functional requirements of the care plan modelling system. The care plan modelling system has two main stakeholders, the first being the health care professional and the second being the patient. The care plan modelling system is composed of four main parts, the first being the visual care plan modelling language, followed by the care plan instantiation tool, the third being the patient application and the fourth being the visual patient application modeller.

The visual care plan modelling language allows health care professionals to model complex health care plans composed of different types of health care activities, performance metrics (goals), assessment modules, and other sub-care plans. Once a care plan is designed through the visual patient application, it can be loaded into the care plan instantiation tool, where it can be fine tailored to the needs of the patient it is to be deployed to. The fine tailoring process involves selecting and deselecting the parts of the care plan which are relevant to the patient, and creating detailed schedules for health activities that are part of the care plan so that these schedules are in sync with the lifestyle and routines for the patient. Once the care plan has been tailored for the patient it can be loaded into a patient application. The patient application provides support to the patient by allowing the patient to readily view their care plan, receive reminders for health activities, enter in data prescribed to them to collect, enter in adherence information, as well as comments and concerns about their health. The patient application is often device specific, which tends to be a major drawback, therefore the care plan modelling system was extended with a visual patient application modeller, through which a designer can model the look and feel of the patient application for different mobile devices and then the patient application can be automatically generated from these definitions.

Each part of the care plan modelling system also has non-functional requirements. The two visual languages, i.e. the visual care plan modelling language as well as the visual patient application modeller both have the non-functional requirements of usability and scalability. For the care plan modelling language usability corresponds to ease of modelling, and convenience, augmented by the availability of different visualisations of a care plan for different health care professionals. Similarly for the visual patient application modeller usability corresponds to ease of modelling and providing the user with a good preview of the patient application being designed. For both visual languages scalability corresponds to being able to deal with not only simple models but increasingly complex models (whether they are care plans or patient application look and feels). For the patient application there are three main non-functional needs the first is the need for securely transferring patient data to and from the remote health servers, the second is related to hardware and technology availability, where by the patients are limited by the hardware and technology that they posses (the motivation for the visual patient application modeller), and lastly the limitations of the internet and bandwidth which most mobile devices have.

Chapter 7 - Design for the care plan modelling system

7.1 Introduction

The purpose of this chapter is to provide the design for the system components of the care plan modelling system. In particular the design of the care plan modelling visual language, the care plan instantiation tool, the patient application architecture, and finally the visual patient application modeller are discussed here.

The care plan modelling system is composed of two visual languages, the first being the visual care plan modelling language and the second being the visual patient application modeller, in order to describe the design for these visual languages, two main aspects about the language are discussed, the first being the meta-model describing the components, relationships and syntax of the language, and the second being the graphical notation that is employed to represent the visual languages.

In addition to this the design of the care plan instantiation tool is also discussed, including design aspects such as the difference between a care plan template and a care plan instance, the concretisation of care plan activity routines to generate the care plan instances. Further more the structure of the care plan instance as formally emitted by the care plan instantiation tool is discussed.

Also included in this chapter are the architectures that can be employed for the patient application, including the final choice of architecture for the patient application and the reasoning behind the choice.

7.2 Advantages of Domain Specific Visual Languages

There are two components in the care plan modelling system which are implemented in the way of domain specific visual languages, the first is the care plan modelling visual language (which allows a health care professional to create detailed health care plan templates), the second component is the visual modeller for modelling of the patient application graphical user interface. Other forms of user interfaces that could be used to allow the modelling of care plans as well as the modelling of the patient application include a textual wizard based approach, where multiple wizards could be developed that would take a user through the modelling of a care plan or patient application. These solutions have up to date been widely used, however there are some flaws that are inherent of such solutions, the first being that a user may always like to have some form of visualisation in order to give feedback on how they are progressing or to give the idea of the big picture while they are modelling, if wizards include a visualisation to give feedback, then surely it would be far easier and a more natural approach to make changes through the visualisation rather than make them through the wizard and see the changes reflected in the visualisation.

Another reason why a visual language may be preferred to a wizard based approach is that wizards may make it a little harder for users to modify their models once modelled through the wizard, no doubt it would still be possible through custom menus, and windows, however, once again visual languages propose a much more natural method of modifiability, the component in the model can visually be located, chosen and modified, with the user getting an immediate picture of the change with respect to the whole model.

Lastly wizards also have a steeper learning gradient than using a visual modelling language. In a wizard a designer may take much longer to learn the different components, and steps to modelling than a visual language. With visual languages, if an appropriate visual metaphor is employed then the learning gradient can be substantially reduced.

7.3 Care Plan Modelling Language

The care plan modelling language is based on the care plan model that is discussed in chapter 5. Based on this discussion a UML model describing the care plan and the components that make up a care plan was constructed. This section focuses on explaining the UML model and the components and relationships of the care plan modelling language that are exhibited as part of this model.

7.3.1 The UML model of a Care Plan

As discussed in chapter 5 the care plan is a composite of many different health care components. The four main types of components that a care plan can be made up of are performance metrics, health care activities, assessment modules and off course other health care plans.

From the UML diagram below it can be seen that a care plan component has only one property, a name, and the relationship that it shares with any other component in the diagram is a contains relationship (expressed as a composite below), i.e. *"a care plan contains other care plans"* or *"A care plan contains health care activities"* or *"A care plan contains assessment modules"*. For example, an obesity management care plan contains the dietary therapy care plan, and the physical therapy care plan, with the physical therapy care plan containing the jogging activity and the swimming activity.

A health care activity like the care plan has only one property a name, however a health care activity can be of many types, i.e. a health care activity can be a simple task, for example jogging, or it can be a review activity such as diabetes review, or it can be a data collection activity such as measuring blood glucose sugar, which in turn has two properties in addition to the standard activity, i.e. a data name (e.g. Blood Glucose Sugar) and a data unit (e.g. mmol/l).

Just like the care plan an activity is also a composite of other components that describe the activity. In the care plan model presented below, the health care activity is a composite of three main components, the instruction on how to do the activity, the resources needed to do the activity, and lastly a general routine on when to do the activity and for how long. Once again, just like the care plan the health care activity has a "*contains*" relationship between these components.

Lastly a care plan is composed of assessment modules; an assessment module is essentially a task flow that aids the health care professional in assessing the patient's condition and progress. An example can be seen in figure 17 in chapter 5 from the obesity management treatment guideline. An assessment module task flow is made up of three main components, the first being an assessment action that describes actual tests or assessments that the health care professional should do, then there are conditions, which represent the questions that the health care professional needs to ask when assessing the patient, and lastly there are treatment recommendations for the patient.

In contrast with the other components in the care plan model such as health care activities and care plans themselves, the relationships between the components of an assessment module is not a containment relationship. From the UML model of the care plan it can be seen that there are two different types of relationships that can exist between components of an assessment module. The first such relationship is a simple *task flow* relationship. This relationship helps the health care professional model the order in which decisions and assessment tasks should be conducted. The second relationship is a *yes/no* conditional branching relationship that exist between conditions (or questions) and other assessment components. This gives the health care professional the ability to model different routes of action depending on different conditional health factors. Hence using the *task-flow* and *yes/no* relationships the health care professional can model an assessment algorithm as part of the care plan.

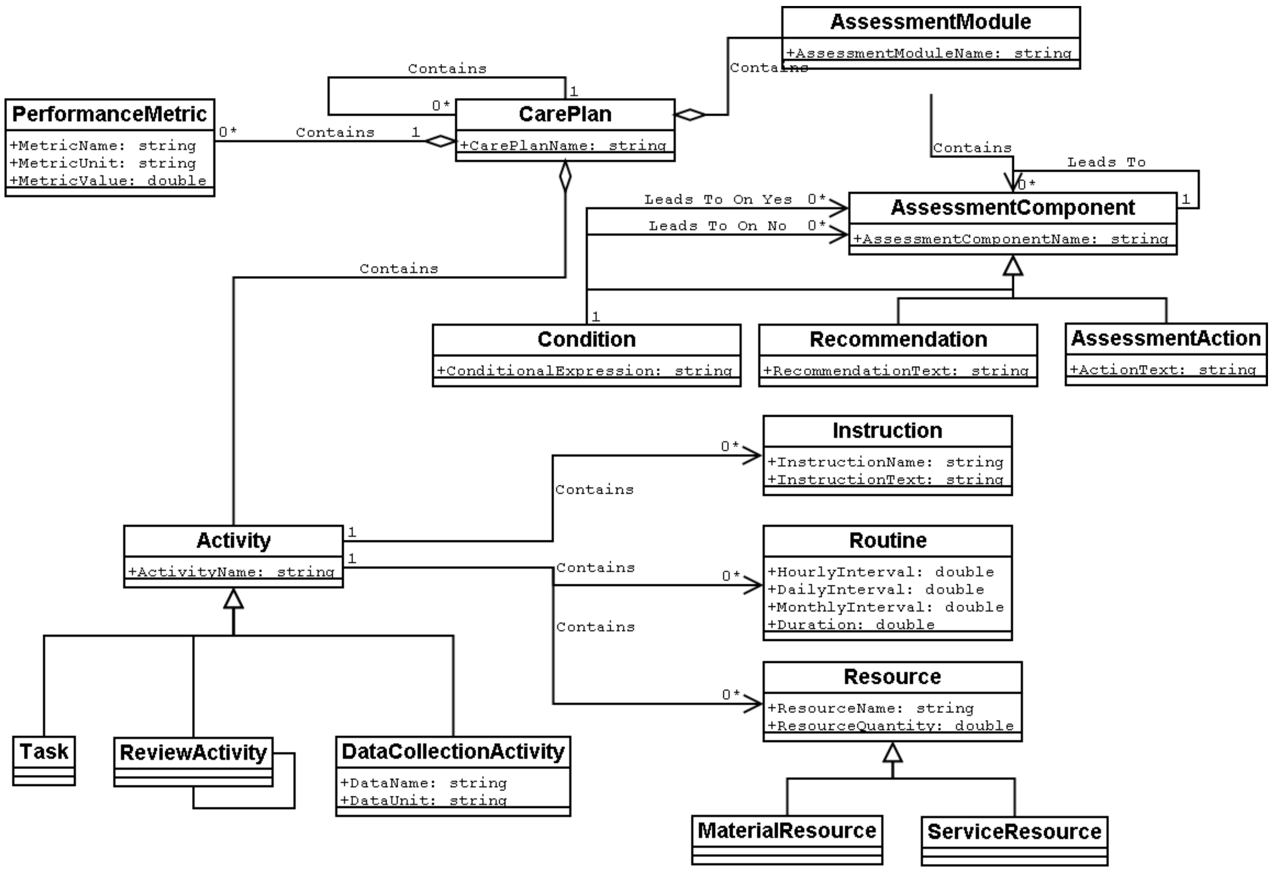


Figure 38: The UML model of the care plan.

7.3.1.1 *Interesting design issues – Capturing Recurring Events in the Care Plan Model*

The rational behind the design and structuring of the care plan components is largely explained in chapter 5 through the use of clinical guideline based scenarios (*i.e. the obesity management scenario*), however one interesting design issue that arose other than this includes how the visual language allowed the health care professional to capture recurring events using the generic routine component shown in the UML model above.

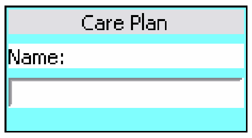
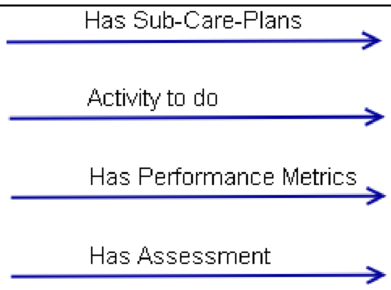
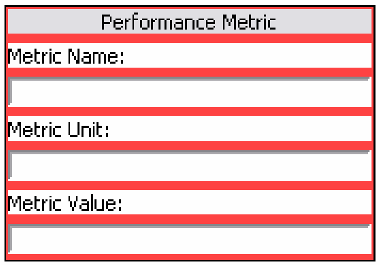
It can be seen from the UML model that each health care activity is related to one or more general routines. As explained in chapter 5 the general routines are not really specific timetables, specifying when in a day, what exact time, etc, these details are to be specified during the instantiation period by the health care professional after discussing the patient's daily routines. Instead the purpose of the general routines was to specify simple rules on how often a health care activity should recur. Therefore health care professionals should be able to use the general routine component to specify rules like; *"Do an activity every 2 days for 30 minutes"* or *"Come for a 2 hour review on a 3 monthly basis"* or *"Take your medication every 6 hours"*. Note that here the health care professional does not specify things like what time in the day to do the activity, or how often in a day to do it; this information is specified during the care plan instantiation process, as it depends on the patients personal needs and daily routines.

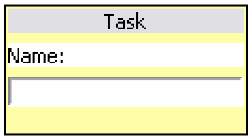
In order to allow the health care professional to capture these generic rules, there were four properties introduced to the general routine component, the first is the hourly interval, the second is the daily interval, the third is the monthly interval, and finally the duration for conducting the activity. This way the health care professional can decide that a health care activity is to be conducted on a monthly basis such as health reviews and hence will set the monthly interval to 3, and then specifies that this review will be for 120 minutes. What time exactly this review will take place after the 3 months, is specified during the instantiation stage. Like wise the health care professional can specify daily rules and hourly rules. Using these rules the health care professional can still specify templates with a general sense of when health care activities are recommended to be done without putting in information that will be in appropriate for any one patient. The design of how exactly a detailed patient schedule is wrapped around these general routines is explained in the section describing the design of the care plan instantiation tool.

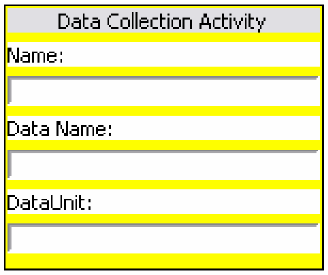
7.3.2 The Notation for the care plan modelling visual language

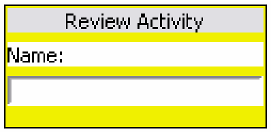
The notation of a visual language is inherently one of its most important aspects as it directly affects its usability. The notation of the care plan modelling language was designed to be a simple block and arrow flow diagram, where each component is represented as a rectangular block with, the properties of the component contained inside the block. Furthermore arrows were used to convey relationships between components, with the expressions on top of the arrows indicating the nature of the relationships. In addition to this, in order to provide easy recognition between different types of components, colour coding was used, i.e. each care plan was coloured a shade of neon, health care activities coloured shades of yellow, activity instructions coloured shades of blue and so on as shown in the table below. A similar approach has been used by the Protégé system, which uses block and arrow diagrams to visualize information and knowledge structures such as clinical guidelines (Stanford Center for Biomedical Informatics Research 2007). In fact the GLIF clinical guideline framework was implemented in Protégé as shown in figure 39 below (OpenClinical 2007).

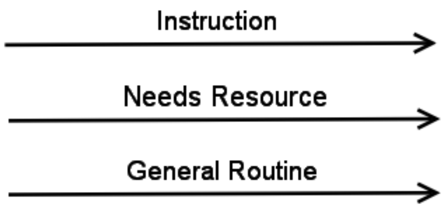
Table 1: Showing visual notation for the components and relationships of the care plan model.

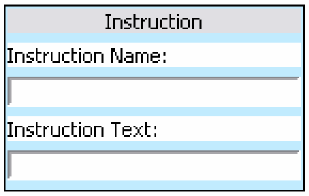
Components	Relationships
	 Has Sub-Care-Plans Activity to do Has Performance Metrics Has Assessment
 Performance Metric Metric Name: Metric Unit: Metric Value:	

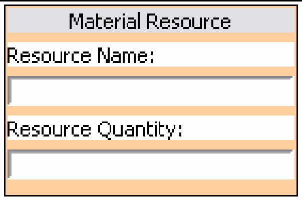

Task
Name:

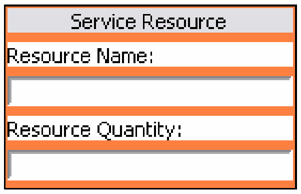

Data Collection Activity
Name:
Data Name:
DataUnit:

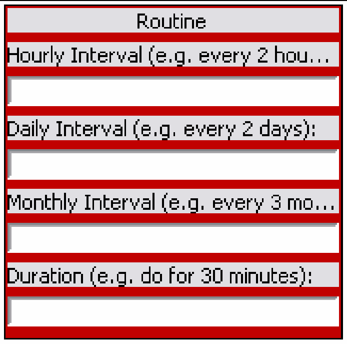

Review Activity
Name:

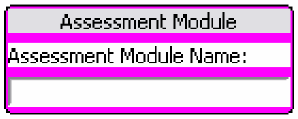
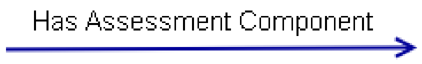
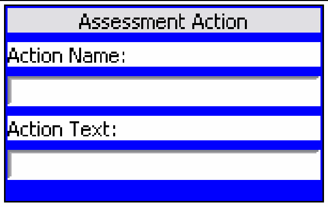
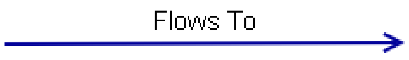
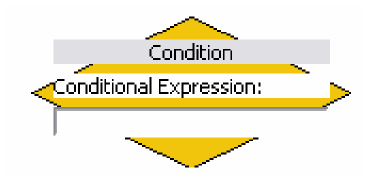
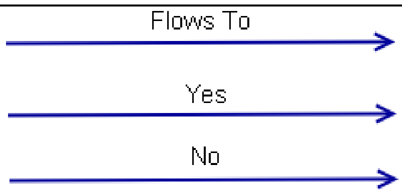
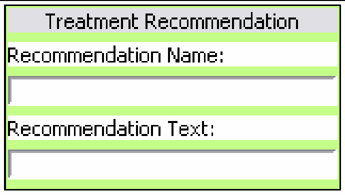
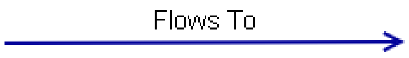

Instruction
Needs Resource
General Routine


Instruction
Instruction Name:
Instruction Text:


Material Resource
Resource Name:
Resource Quantity:

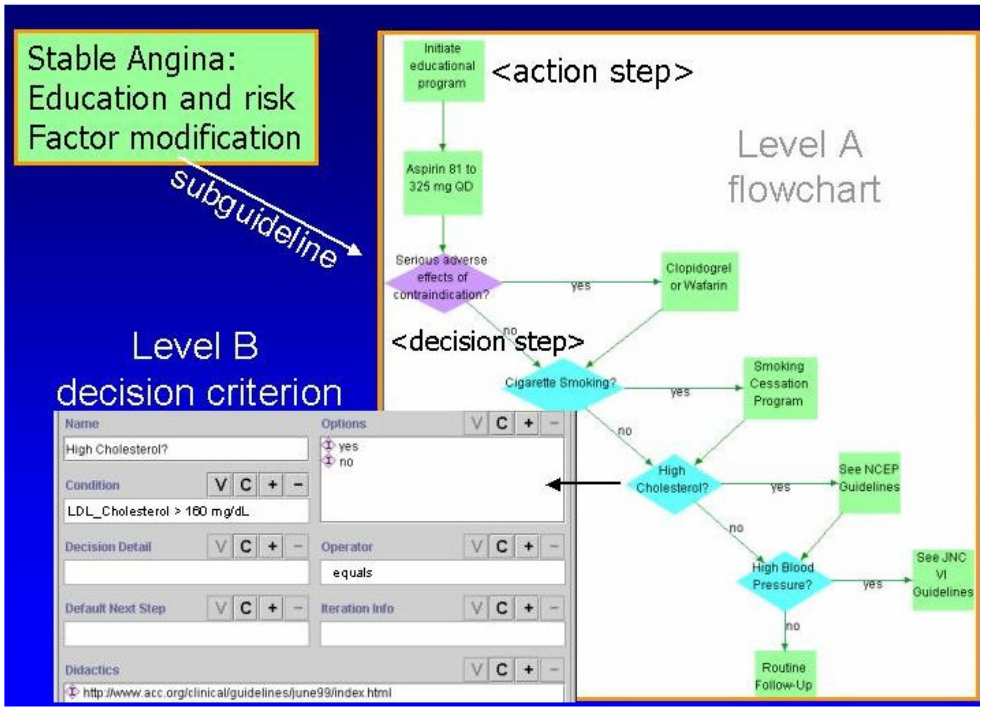

Service Resource
Resource Name:
Resource Quantity:


Routine
Hourly Interval (e.g. every 2 hou...
Daily Interval (e.g. every 2 days):
Monthly Interval (e.g. every 3 mo...
Duration (e.g. do for 30 minutes):

	
	
	
 Treatment Recommendation Recommendation Name: Recommendation Text:	

Besides the block and arrow diagram technique, many other techniques exist to visualize information structures such as clinical guidelines and health care plans; a vastly different type of visualization for care plans was designed by Asbru View (Open Clinical 2007), figure 40 contains a sample jaundice treatment plan provided with the Asbru View download. Asbru uses a time scale and a three dimensional format of lanes, to describe the decisions and treatment scenarios of the health care plan, with the positioning of the blocks on each lane along the axis indicating the temporal relationships between each part of the care plan.

Other approaches at visualizing health care information can be observed in systems such as Lifelines, which uses a continuous time scale to visualize the health care information such as patient history, i.e. the symptoms of the patient, health care activities done by the patient, other events, medication plans etc as shown in figure 41 (University of Maryland 2007). These types of visualizations may not necessarily be useful when modelling the health care plans described in this research, however they may be of particular importance when the health care professional wants to view the big picture of the treatment of the patient over a period of time, therefore even though the simple block and arrow approach may be good enough to model and visualize care plans when they are first created, if the care plan modelling visual language is to be truly useful it should incorporate other visualizations to show the care plan for example, during execution, with respect to other patient historic information, etc. However due to the timing constraints imposed on the research done, more emphasis was placed on the development of the syntax and structure of the care plan model on which the care plan modelling visual language was based rather than perfecting the visualization for the end user. Therefore it is recognised that the notation presented in this section is by no means perfect and may be subject to improvements with the addition of multiple visualizations for health care plans, in the many different contexts that the care plans may need to be analysed in.



Name: High Cholesterol?
Options: yes, no
Condition: LDL_Cholesterol > 160 mg/dL
Decision Detail:
Operator: equals
Default Next Step:
Iteration Info:
Didactics: http://www.acc.org/clinical/guidelines/june99/index.html

Figure 39: Example of a GLIF clinical guideline coded in Protégé reproduced from (OpenClinical 2007)

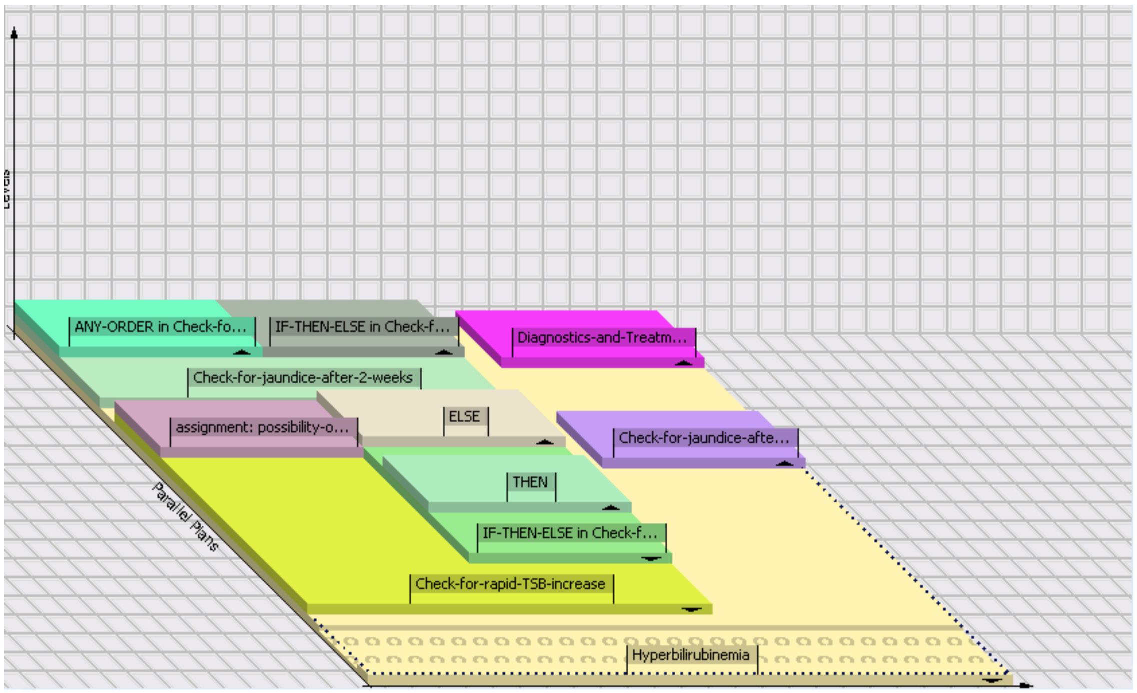
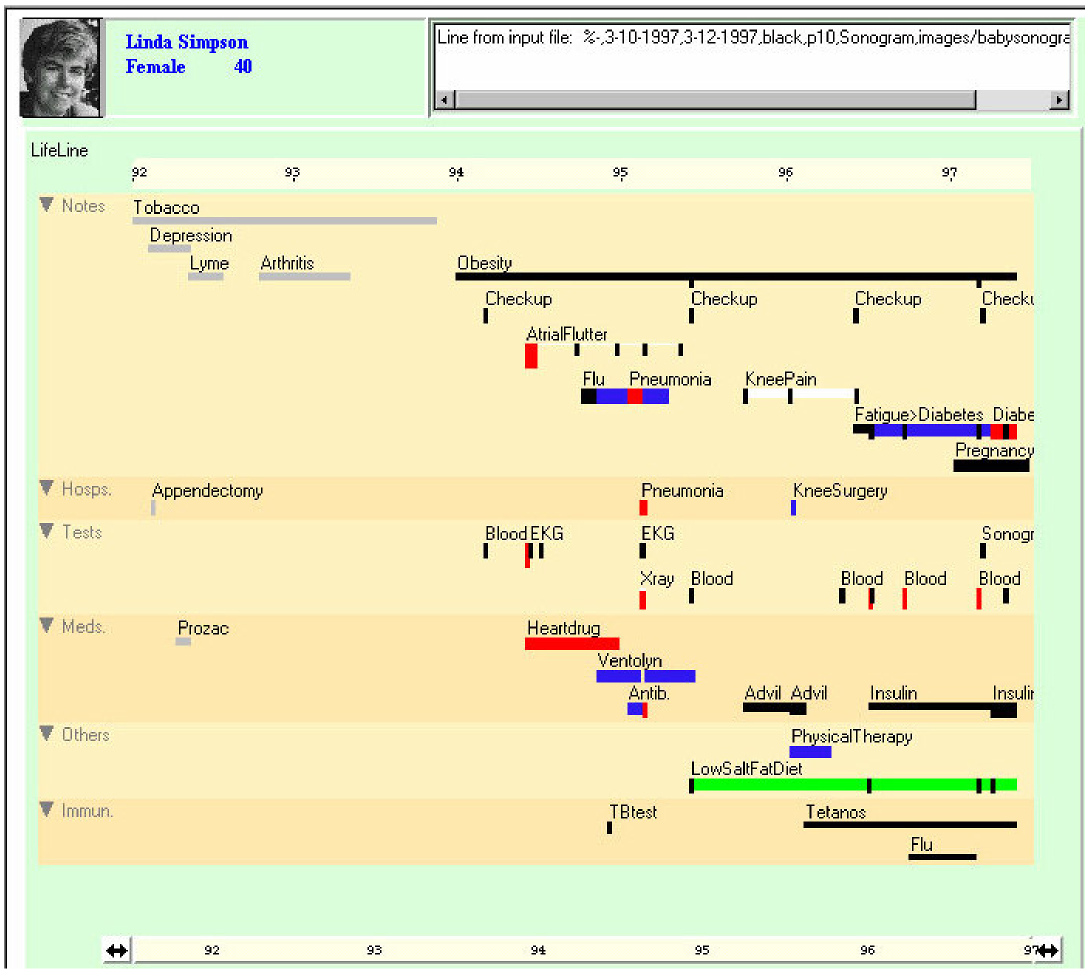

Levels
ANY-ORDER in Check-fo...
IF-THEN-ELSE in Check-f...
Diagnostics-and-Treatm...
Check-for-jaundice-after-2-weeks
assignment: possibility-o...
ELSE
Check-for-jaundice-afte...
THEN
IF-THEN-ELSE in Check-f...
Check-for-rapid-TSB-increase
Hyperbilirubinemia
Parallel Plans

Figure 40: Asbru View care plan for Jaundice treatment reproduced from the software program by (Huber, Votruba et al. 2007)



Line from input file: %-,3-10-1997,3-12-1997,black,p10,Sonogram,images/babysonogra

Figure 41: Example of LifeLines, which presents a visualization of a patient health record, treatment history and other patient health care plan information reproduced from (University of Maryland 2007).

7.4 Care Plan Instantiation Tool

As mentioned in chapter 5 and chapter 6, the care plan instantiation process ensures the care plan template that is generated through the CPMVL is fine tailored to the patient. This tailoring process can essentially be split up into two main categories, the first being editing the care plan template components (this means adding any extra needed components such as health care activities and removing any unnecessary components), the second being ensuring that the general routines that are programmed for the patient's health activities are made into a specific schedules/timetables from which appointments for the health care activities can be generated. Since adding, removing and editing health care components can be done within the care plan modelling visual language itself, the focus of the instantiation tool was on providing the functionality to specify detailed schedules for health care activities from their respective general routines.

7.4.1.1 The difference between a care plan template and a care plan instance

In order for a patient to use a care plan actively, the care plan needs to be instantiated for them. This instantiation is where a largely general care plan template is fine tailored to a patient. As explained above this tailoring process can be divided into two, the first part involves adding or removing relevant care plan components from the care plan. The second part is to instantiate the actual action part of the care plan i.e. the health care activities, this part involves taking the general routines for the health care activities that were specified through the care plan modelling stage, and specify a more detailed schedule or timetabling rules, which allow the generation of medical alerts and reminders for the patient in order to conduct the health care activity appointments.

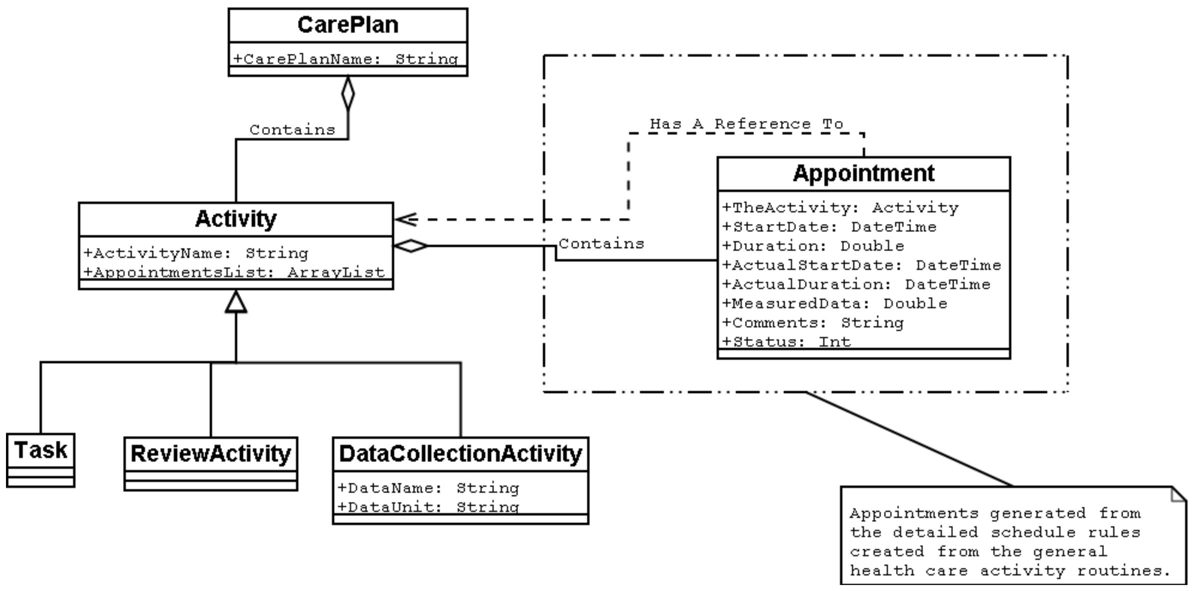


Figure 42: The care plan instance object oriented analysis.

In order to understand the structure of the care plan instance it is helpful to look at the object oriented model of a care plan instance. The object oriented model of the care plan instance is almost exactly the same as the care plan UML model shown in figure 38, with the addition of an appointment object. The purpose of the appointment object shown in figure 42, is to capture the real time data related to the conduction of a health care activity, as shown in the UML diagram above, it includes the start time of the appointment (based on which, health care activity reminders and alerts could be thrown by the patient application), the duration on how long the activity should be done, the actual start date (i.e. when the patient actually started the activity, perhaps after snoozing it the first time), the actual duration (i.e. how long the patient actually took to do the health care activity), the measured data that the patient recorded (in the case of the appointment being for a data collection activity), comments describing general concerns and notes (perhaps used to record side-effects after conducting the health care activity), the status of the activity appointment, i.e. has been attempted, not attempted, or attempted and not completed, and finally the appointment contains a reference to the activity that it is concerned with.

Once a care plan has been instantiated each health care activity that is part of the care plan will have a list of the appointments regarding the instances where the patient is supposed to do the activities.

7.4.1.2 Converting General Health Care Activity Routines to Detailed Schedules

The object model of a generalised routine is shown below, and as expressed above, it allows health care professionals to specify general rules for when the health care activities that the routine is associated with are supposed to be done. The routine object has four properties, an hourly interval, a daily interval, a monthly interval, and duration.

Using these properties the health care professional, can specify for health care activities:

1. Hourly rules: For example take medicine every 2 hours, or conduct muscle stretch exercises for 10 minutes every 6 hours.

2. Daily rules: for example go for a jog every 3 days, for 30 minutes.

3. Monthly rules: For example come for obesity and diabetes review every three months, for 120 minutes.

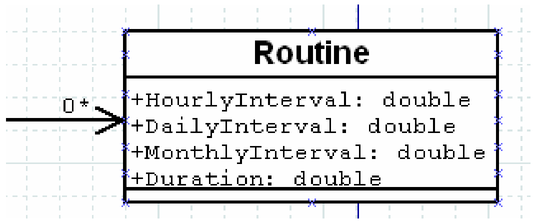


Figure 43: The object model of the general routine object.

In order for the hourly, daily, and monthly rules to be translated to more specific patient timetables, they need more information. The first item of information that is required is an activation start and end date. This property specifies that when the patient is supposed to start conducting the health care activity according to the specified routine, and when the patient should terminate the conduction of the activity. For example, a certain medicine prescribed to the patient may be specified to be taken every 6 hours, every day, however the patient may only need this medicine for 3 months, therefore the health care professional can specify that the patient should start doing this activity on for example 1st of January (activation start date) and stop taking the medicine on 1st of April (activation end date), figure 44 below shows the routine object with the extra properties for activation start and end date added.

The next step in detailing the general routine, is to convert the hourly, daily and monthly rules (expressed through intervals), into more detailed routines. In order to do this, the decorator design pattern was applied to the general routine object, in that the general routine was decorated by specific routine objects that added specific timetabling information to the routine, making the routine concrete for the instantiation process. The following decorators were added to the routine object:

1. Hourly routine: Once an hourly interval is specified, there is no real need to specify any more detail. The reason for this is that the hour is already one of the smallest units of time, smaller than this would be the minute, and it is unlikely that the health care professional may want to specify the exact minutes that activity must be done in an hour.

2. Daily routine: Once a daily interval has been specified, the health care professional may want to specify, how many times in a day, and at what exact times, the heath care activity may need to be performed, for example,

take a medicine, every 3 days, at 8 am in the morning, and at 8 pm in the evening, or go jogging every day, at 7am in the morning for 30 minutes. Therefore the daily routine decorator of the general routine, has a list of preferred times, that the activity needs to be done in that day.

3. Monthly routine: Once a monthly interval has been specified the health care professional may want to set a preferred day on which to do the health care activity, and the number of times, and the exact times at which the activity must be done on that day. For example, a health care professional may specify, that a the patient must come in for a review every 3 months, he may specify that the preferred day, is the first Tuesday, 3 months from now, and that the review will take place at 10am in the morning. Therefore the monthly routine decorator has the properties, of preferred day, and list of preferred times.

Using these three decorator objects, the health care professional can cater the general routine that is part of the care plan template, into something that is more concrete and finely tailored to the patient's daily routines and timetables. Once the routine is made specific, then it can be used to generate the health care activity appointments for the patient as part of the care plan instance. The process by which a specific routine can be used to generate health care activity appointments is explained below.

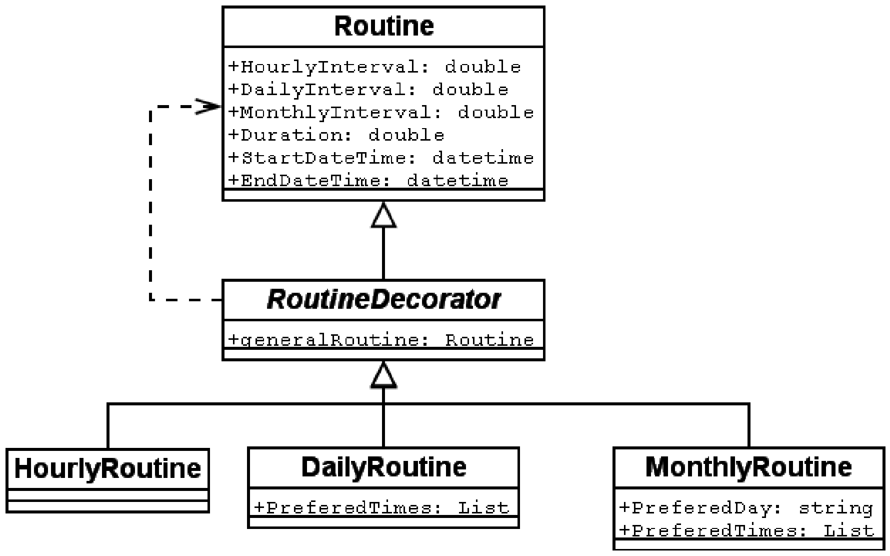


Figure 44: Object model of a routine instance.

7.4.1.3 The Instantiation process for Generating Appointments from Concrete Routines

Once a general routine is made concrete by decorating it with the necessary information, then it can be used by the instantiation process, to generate appointments as part of the care plan instance, the appointments in turn being used to generate the medical reminders and alerts on the patient's mobile device application. Figure 45 below shows how the concrete routine is used to generate the appointments for the care plan instance. The first step in this process is to determine which type of routine is specified by the routine object, is it an hourly routine, or a daily routine, or is it a monthly routine? If it is an hourly routine then the dates for the appointments can be determined by repeatedly adding the hourly interval to the activation start date until the activation end date is reached. For daily routines the daily interval is added to the activation start date to get the day on which appointments need to be set, and then appointments are set on that day, at the preferred times, this is again repeated until the end date is reached. In a similar fashion, for monthly routines, the monthly interval is added to the activation start date, to determine the month of the appointment, and then the preferred day is searched for in that month, and appointments are set for the preferred times on that day, this is repeated until the activation end date is reached in order to generate all the monthly appointments.

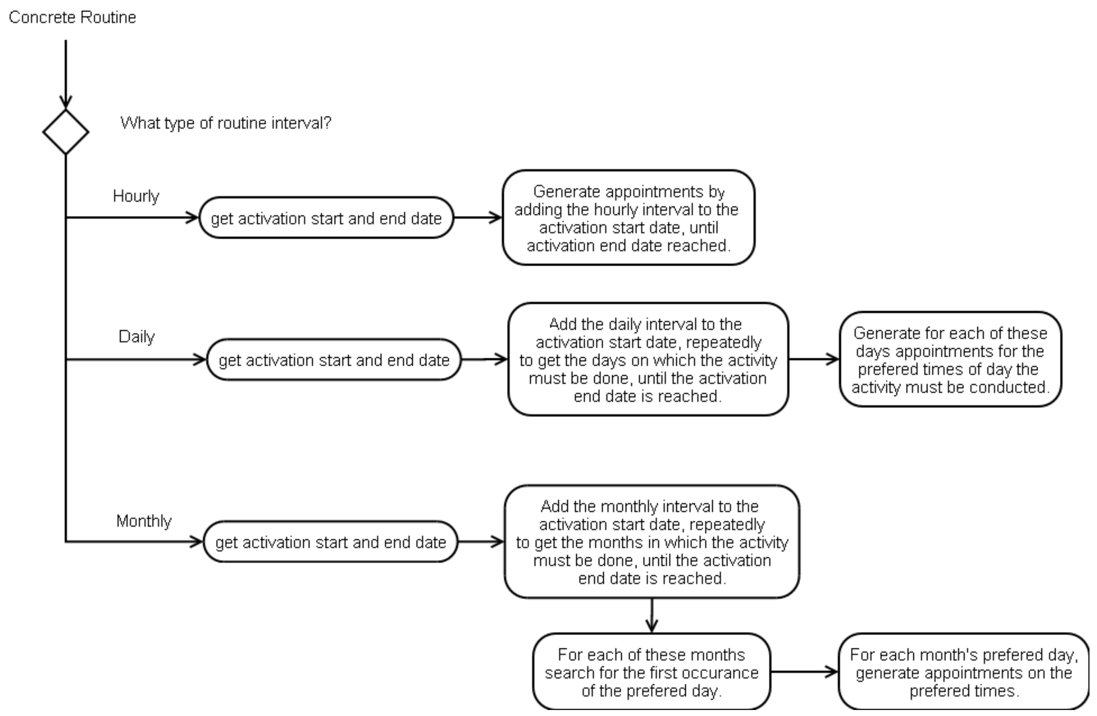

Concrete Routine
What type of routine interval?
Hourly
get activation start and end date
Generate appointments by adding the hourly interval to the activation start date, until activation end date reached.
Daily
get activation start and end date
Add the daily interval to the activation start date, repeatedly to get the days on which the activity must be done, until the activation end date is reached.
Generate for each of these days appointments for the prefered times of day the activity must be conducted.
Monthly
get activation start and end date
Add the monthly interval to the activation start date, repeatedly to get the months in which the activity must be done, until the activation end date is reached.
For each of these months search for the first occurance of the prefered day.
For each month's prefered day, generate appointments on the prefered times.

Figure 45: The instantiation process for hourly, daily, and monthly routines.

Once this instantiation process is completed, the care plan template has been converted to a care plan instance and is now ready for execution on the patient's mobile application, providing useful health care information, along with the capability for medical alerts, and reminders for health activity appointments.

7.5 Formal Representation of Care Plan Instances

As described above a care plan can exist in one of two states, the first being a care plan template, and the second being a more patient specific care plan instance. The care plan modelling visual language and the care plan instantiation tool, provide interfaces to design, model, create and instantiate care plans, however once this is done, the care plan needs to be appropriately represented in order to facilitate further processing and use. For the purpose of the care plan modelling system it was deemed suitable to represent the care plan template as well as the care plan instance in the form of an XML document, with a fixed schema. The general data flow in the care plan modelling system can be seen in figure 46, it can be noted that the care plan templates generated from the care plan modelling visual language, the care plan instances emitted by the care plan instantiation tool as well as the patient application models are stored in XML format.

There are many reasons for which XML is the ideal choice for care plan data representation, the first is that XML is human readable, therefore, a health care professional can open a care plan template without the visual language editor and still understand its content. The second advantage is that it does not depend on any platform, or software application, and is a standard that is widely accepted. These two advantages combined mean that care plan data can easily be transferred and communicated from one health care professional to another, or to patients in a very ordinary and easy manner, by email for example. Lastly a major advantage of XML is that it can be easily transformed into other formats or visual representations through either custom software applications, or simply through the use of XSL transformation style sheets. This means that in the future third party providers may come up with multiple representations of the same care plan data (templates and instances) to show or emphasise different aspects of the data from the same care plan template or instance.

The schema for the care plan instance XML file as emitted by the care plan instantiation tool is shown below in figure 47. One thing to note is that it mimics almost exactly the meta-model of the VCPML shown in figure 38, with two exceptions, the first is that assessment modules have been excluded from the care plan instance data, as the assessment modules are simply to give the health care professional a visual guide to assessing the patient and hence not intended for deployment to the patient. The second point of difference is the fact that activities have appointment elements as child elements, being proof that the care plan template has been instantiated for the patient (see above the difference between the care plan instance and template).

Hence the care plan instance XML document can be described as being composed of care plan nodes, each care plan node containing sub care plan nodes, metrics node, and activities node. The sub-care plan nodes contain sub care plans, the metrics node contains performance metrics, and the activities node contains activities. Like wise the activity node contains name and type attributes, along with instruction nodes, resource need nodes and appointment nodes.

The XML schema of the care plan template is not discussed here, as the XML for the care plan template is automatically generated by the care plan modelling visual language editor of Marama, it is transformed once, before usage in the care plan instantiation tool and after transformation its structure is similar to the care plan instance, and hence fore the sake of clearance it has been omitted from this section. The care plan instance XML schema is far more interesting and necessary to discuss as the structure of the care plan instance is directly used in the patient application modelling process.

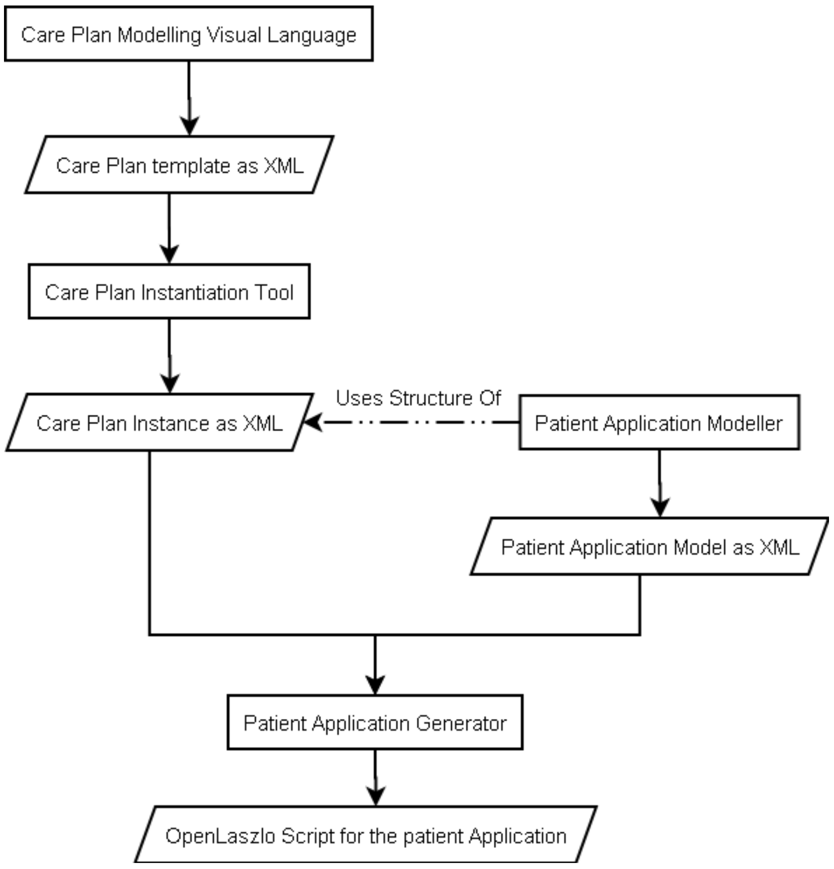

Care Plan Modelling Visual Language
Care Plan template as XML
Care Plan Instantiation Tool
Care Plan Instance as XML
Uses Structure Of
Patient Application Modeller
Patient Application Model as XML
Patient Application Generator
OpenLaszlo Script for the patient Application

Figure 46: The data flow in the care plan modelling system through the usage of XML documents

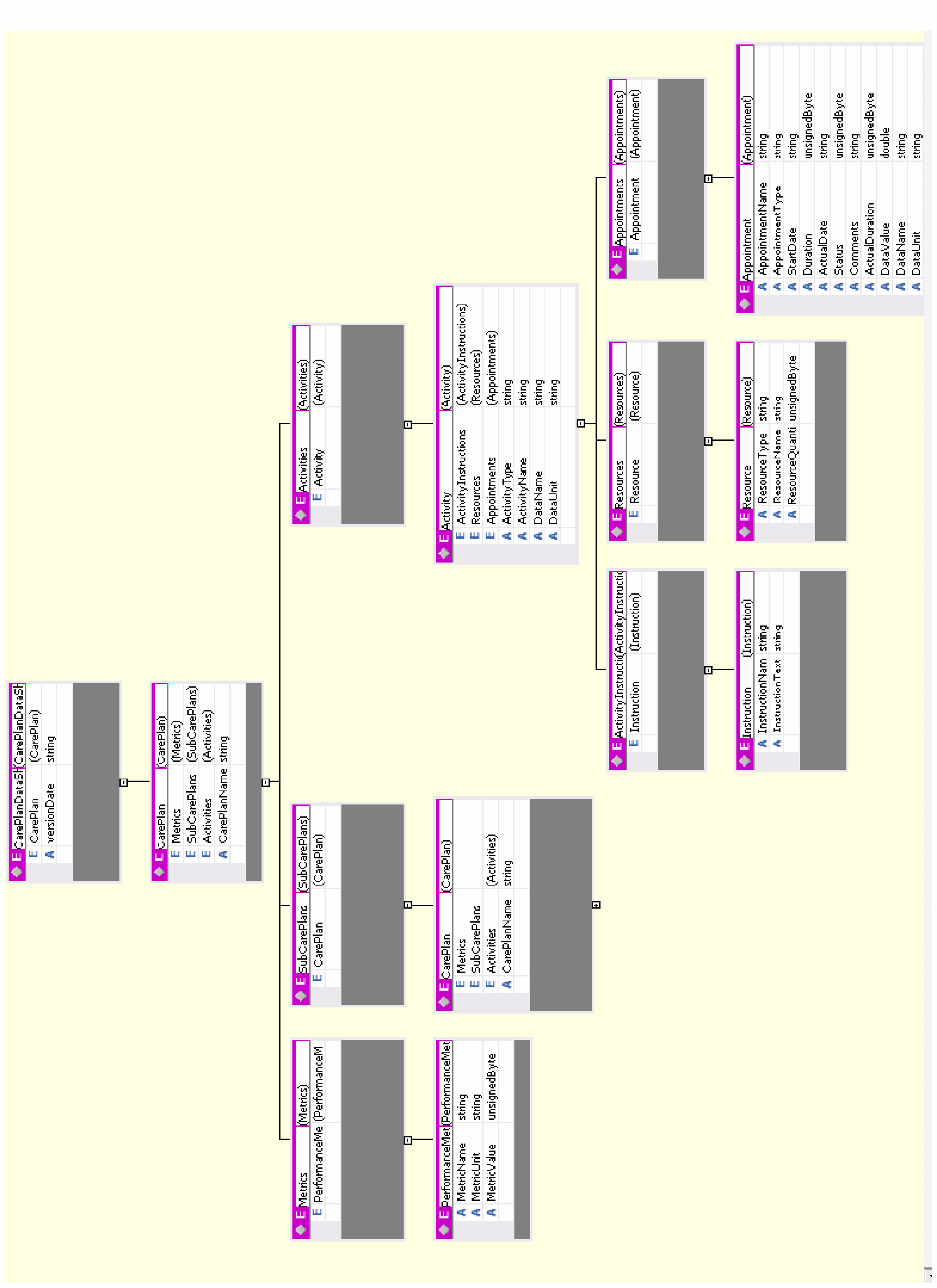


Figure 47: The combined XML schema for the care plan instance.

7.6 Patient application design

7.6.1 Patient Application Architecture

The functionality of the patient application is largely discussed in earlier chapters; the purpose of this section is to discuss the recommended architecture of the patient's mobile health application. As always the architecture of any application, is driven by the non-functional quality attributes of the application. For the purpose of the mobile patient application, three main quality attributes were used as a driver, the first one was connectivity of the mobile application to the internet, this quality attribute driver was used to choose the appropriate architecture for the patient mobile application, the other two quality attributes that were considered included, the ease of development and the portability between different mobile devices, and platforms. These two quality attributes were not used to influence the architecture of the mobile application, but instead they form the motivation for the visual patient application modeller described in the following sections of the chapter. Other architectural drivers such as security are recognised to be important to consider while designing the mobile device application, however due to timing constraints imposed on the research, they were out of the scope of the development work done.

As mentioned above, one of the main architectural drivers for the patient mobile application was connectivity, there are three main states of connectivity that most mobile applications are designed for, the first is never connected, i.e. the mobile device is "never connected" to the internet, the second is "always connected", i.e. perfect connection to the internet all the time, the third is "partially connected", i.e. some times connected, followed by possibly large periods of disconnectivity (Lee, Schneider et al. 2004).

7.6.1.1 A Never Connected Architecture

Examples of never connected architectures can be typically seen in the form of fat clients. Never connected fat clients are generally three tiered applications, where the first tier is the presentation or user interface layer, the second layer is the business logic layer and the third layer is the database or persistency layer, all three layers residing on the client side, which in this case is a mobile device (Lee, Schneider et al. 2004).

The obvious advantage of such an architecture for the patient mobile application, is that there is a "perceived high performance", as the application does not suffer from poor latency due to communication over a low bandwidth to a server (Lee, Schneider et al. 2004). However one of the advantages of having a personal mobile patient application, is that the patient can communicate their health related data and concerns to the health care professional, and hence the patient application needs to have some connection to a remote server in order to facilitate this information exchange.

7.6.1.2 Always Connected Architecture

An always connected architecture is an architecture that assumes that the user has continuous and guaranteed connection to the internet. Applications of this type are usually client server applications where the client is completely thin with no application code, i.e. all application code and persistency are on server. However it may also be that the application has a partially fat client, which has some layers of application code, such as presentation and business layer on the client, and the data and persistency are delegated to a remote server, with the client continuously requesting and updating data, with the processing of the data taking place at the client side itself (Lee, Schneider et al. 2004). The advantage of this over a completely thin client is that it may act to improve the perception of performance, because all the processing is done on the client itself, it reduces the latency of the application (Lee, Schneider et al. 2004).

If a user can be sure that the connectivity to the internet that they receive on their mobile device is guaranteed at all times, with reasonable bandwidth to keep latency at a minimum, then these two architectures would be perfect for the patient application, because it would mean that the health care professional, after instantiating the care plan can host it on a remote server where by the patient can connect to it through a thin or partially fat client, and any input they enter in would be immediately sent to the remote server and updated against the patients records. It would also make it possible for any form of feedback to be delivered to the patients almost instantaneously.

Despite the advantages, of such an architecture, realistically such guaranteed connectivity and bandwidth is very rare, and hence most users, will find that they are not connected to the internet for substantial periods of time, after being connected for a certain period of time. Hence it is necessary to find some mid-way between an always connected and a never connected architecture to satisfy the non-functional design requirements for the patient mobile application.

7.6.1.3 A Partially Connected Architecture

A partially connected architecture as the name suggests, assumes, that there may be large periods of time when the user is not connected to the internet, and designs a way to make these periods transparent to the user. The client in the partially connected architecture has the same structure as that of the never connected architecture, i.e. it is a fat client that is completely self-sufficient, i.e. it has a presentation layer, a business layer, a data access layer, and a persistence store (e.g. a local database) (Lee, Schneider et al. 2004). On the server side the number of layers can vary from having a minimum of a remote data base, and a data base server, to also including separate application and presentation servers to aid the client in periods of connectivity (Lee, Schneider et al. 2004).

With such an architectural pattern the patient application would be a completely fat client with local persistence and would communicate with the remote server to obtain care plan instances, this care plan instance is then somehow stored in the local persistence, and when the patient enters data through their personal mobile device application, this data is then stored in the local persistence (i.e. by updating the locally stored care plan instance), and when the patient is connected to the internet, this update is sent to the remote server which in turn updates its records of the patient's care plan instance, and may send some feedback to the patient. This strategy is commonly known as a "store and forward strategy" and meets the patient mobile applications non-quality requirements by allowing connectivity to remote health servers, as well as accommodating for when the user is not connected to the internet (Lee, Schneider et al. 2004). The partially connected architecture can be observed in figure 48 below, how this architecture can be implemented using the OpenLaszlo language is discussed in the next chapter.

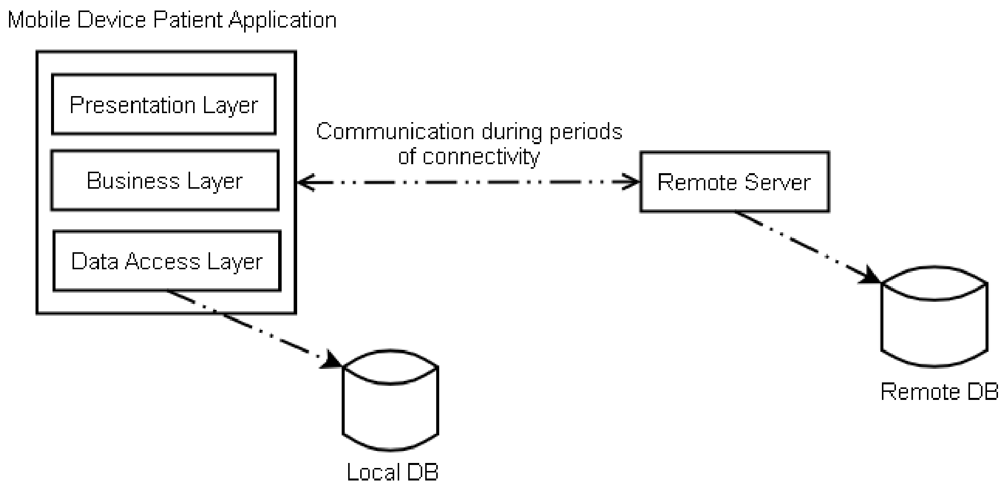


Figure 48: The general architecture of the patient application, as a partially connected mobile device application with a local DB which connects to the remote server to download and update data during periods of connectivity recreated from (Lee, Schneider et al. 2004).

7.7 Patient Application Modeller

The purpose of the patient application modeller was to enable a technical minded graphical user interface designer to model how a care plan instance is to be represented to the patient in the form of a patient application. What follows in this section presents the design of the patient application modeller, first presented is the meta-model of the visual language designed to allow patient application modelling, after which the graphical notation of the visual language is presented and discussed.

7.7.1 Visual Modeller Meta-Model

The purpose of the visual patient application modelling language is to design the graphical user interface for the application through which the patient can view and interact with their care plan instance. The goal of the patient application modeller was to allow a GUI designer to associate the properties of the care plan instance to graphical user interface elements to expose the entire care plan and its properties to the patient for interaction via the patient application. The meta-model diagram of the visual patient application modeller (VPAM) is shown below in figure 49.

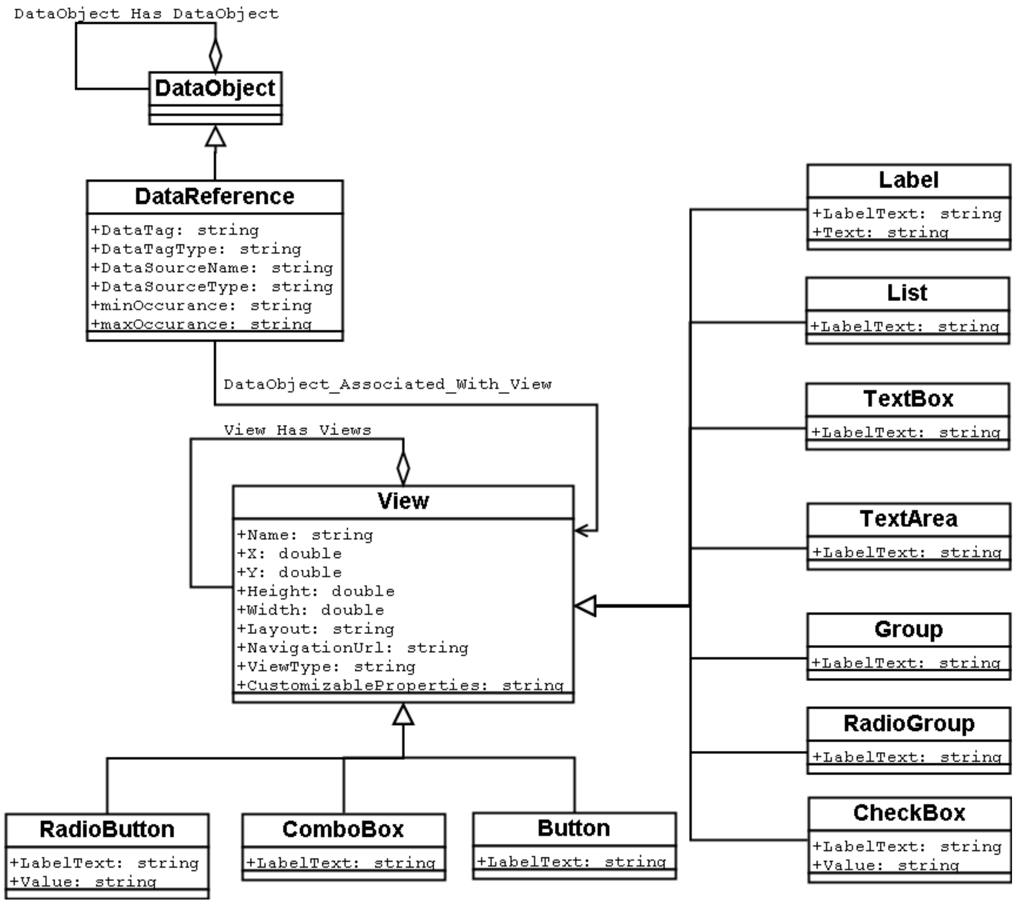


Figure 49: The meta model of the visual patient application modeller.

7.7.1.1 Representing Care Plan Properties through the Visual Modeller

The first step in mapping care plan properties to GUI elements is to represent the care plan structure through the visual patient application modeller. In order to do this, two things need to be known, the first being what the actual properties of a care plan are, and what the sub-components of the care plan are and what in turn their properties are. This knowledge is described in the meta-model of the care plan modelling visual language as shown in figure 38. The second item of information that needs to be known is the physical format and structure in which the care plan, and its sub-components are stored, in this case all care plans and care plan instances are stored in XML format, with a fixed structure as described by the XML schemas in section 7.5 above.

From section 7.5 it can also be seen that the care plan is represented as a hierarchical structure where all components are represented as XML elements, their properties represented as XML attributes and their sub-components represented as XML child elements. Hence if a designer needs to model the XML structure of the care plan template or instance, there needs to be some way to model this structure in a similar way to which XML schema editors such as Altova and Visual studio allow (see the Care Plan Instance schema in figure 47 above), yet specific to the needs of the visual patient application modeller.

In order to provide this modelling functionality to the visual patient application modeller, a data reference object was added to the meta-model of the visual language. The data reference object and its properties can be observed in the meta-model of the visual language shown in figure 49, and include a data tag to represent the name of either XML elements or XML attributes, a data tag type which captures information on what type of data reference this is, i.e. is it an XML element, or an XML attribute or something else, next there is a data source name, this is the actual name of the file in which the care plan data is stored, following this there is the data source type which in this case is an XML file, and finally there is min and max occurrence to capture how often child components appear in parent components.

From the meta-model it can be seen that the data reference object extends from a generic data object. The reason for this is that in the future it could be that the care plans are no longer stored in XML format, they are now, instead stored in either a data base table, or in another in-house standard of a third party, therefore the patient application meta-model could then be extended with another type of data reference object. Lastly it can be seen from the meta-model shown in figure 49, that a data object can be composed of many other data objects, the design purpose of this relationship was to allow a designer to capture parent child relationships between XML elements.

7.7.1.2 GUI Elements to Map Care Plan Properties With

Once the designer of the patient application has modelled the structure of the care plan data using the data reference object discussed above then the next step is to associate each item of data in the care plan data structure to graphical user elements that will be used to represent the data in the patient application. For this the designer needs to have access to a range of GUI elements. For most programming languages the graphical user interface elements that are provided as part of their API include items such as buttons, textboxes, text areas, checkboxes etc, therefore the GUI elements that were made available for the patient application modeller include some of the most common GUI elements that are used in graphical user interface development. These GUI elements include labels, lists, textboxes, textareas, groups, radio-groups, radio-buttons, checkboxes, comboboxes and views. Besides these components many more gui elements could be added to the visual modelling language, such as calendars, date spinners, etc, however due to the fact that the entire care plan modelling system is really a proof of concept prototype, the array of gui elements were limited to those shown in the visual language meta-model in figure 49.

7.7.1.2.1 The concept of the View

One interesting design that can be observed about the GUI elements in the visual patient application modeller from figure 49 is that all the GUI components inherit their general properties from the view component. The reason for this design decision is because the application that is intended to be generated through the patient application model is targeted at the OpenLaszlo development language, programs written in which are compilable to Flash and DHTML (OpenLaszlo 2007). In OpenLaszlo the View is the top-most object in the language, with all other GUI elements (and any other objects in general) in OpenLaszlo inheriting from it (Laszlo Systems Inc 2007), therefore it was deemed a suitable design to mimic this object structure in the patient application modeller.

From the meta-model shown in figure 49, it can be seen that a view has general properties such as a name to give it identification, x and y coordinates to determine where exactly in the parent view the current view will be located, a height and width of the view, a layout of the view i.e. either a "simple vertical layout" or a "simple horizontal layout" (to be used if exact location is not provided) (Laszlo Systems Inc 2007), a navigation URL, to specify the page which needs to be loaded if the view is of navigate type and clickable (see further on for discussion on different types of GUI components), a view type property to specify what type of view the current view is, and finally customisable properties, to capture information on which properties of the view should be allowed to be customisable by the user of the application. All of these properties are inherited by the GUI components that inherit from the view.

Mimicking the OpenLaszlo object structure in the visual patient application modeller (VPAM), at first seems like a restrictive design, because it implies that the visual patient application modeller can only be used to model and describe OpenLaszlo applications, this however is not true, firstly the GUI components included in the VPAM, such as lists, textboxes, textareas, buttons etc are not specific to OpenLaszlo, they exist in all application development frameworks and languages, including .Net compact and J2ME. Secondly consider the design of inheriting all GUI components from the view, it is true that the view object is recognised only by OpenLaszlo; however the concept of a view is not unknown to other application development frameworks. In that a view is a simple container, which can be composed of a variety of GUI components including other views, according to this definition, a view is easily comparable to a panel, a form, a window, or even a customised user control (all of which in one way or another group graphical user interface functionality), in most frameworks and languages such as .Net compact framework, and the J2ME framework.

Therefore although using the OpenLaszlo object structure makes it slightly easier to design the application generator to generate OpenLaszlo code, other generators could also be written to generate application code in other languages from the same VPAM models.

7.7.1.2.2 Modelling GUI component layout

As discussed above, views can have a layout property. For the purpose of prototyping the visual patient application modeller and the associated OpenLaszlo generator, the different types of layouts were restricted to just two, a simple vertical layout, or a simple horizontal layout. It is recognised that there are many possible layouts that the VPAM could be extended to handle, for example, grid layout, box layout, and other dynamic layouts such as flow layout, however this was out of the scope of development work due to severe timing constraints. The way in which views and sub-views could be used along with the simple layout scheme to create a GUI can be seen in figures 50 and 51.

Along with specifying the layout of the GUI components in a view, a designer should also be able to specify explicitly which GUI components will be contained in which other GUI components, for example specifying which textboxes, and buttons will make up a view, or which radio buttons will make a radio group, etc. For this purpose the *view-contains-view* relationship was introduced to the VPAM meta-model, and through this the designer can specify the way in which the GUI components are composed together. Examples of this containment can again be seen in figure 50 and 51.

One interesting pattern that can be observed with regard to the layout of components in figures 50 and 51, is that most GUI components are accompanied by some form of describing label (exception being a view), in the case of textboxes these labels appear on left side of the components (usually), with text areas these describing labels appear on top of the textareas, with buttons, the label is directly overlaid on top of the component, like wise combo boxes, radio buttons, and check boxes also have their own describing label. Therefore when a designer decides to model a text box, they need to model a describing label, and a textbox, and then tie the two together through a view with a simple horizontal layout, as shown in figure 51. This process can get quite repetitive; therefore the VPAM automates this process for GUI components by adding into each GUI component the property of label text. The OpenLaszlo generator on reading the label text of a GUI component in a VPAM model automatically places the label at the right place in the GUI depending on the primary GUI component for which the describing label is intended for. This idea is not new however and can be seen also in XForms, where the input tag in an XForm translates over to a text field, and one of the parameters for the input tag is a describing label to be put along side the text field, where exactly the label is put depends on the interpreter of the XForms script (W3CSchools 2007).

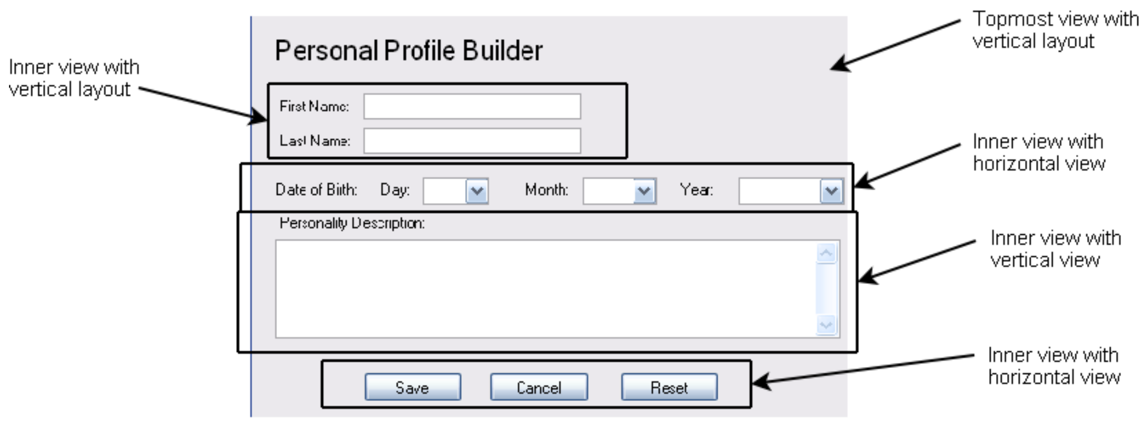

Personal Profile Builder
First Name:
Last Name:
Date of Bith:
Day:
Month:
Year:
Personality Description:
Save
Cancel
Reset
Inner view with vertical layout
Topmost view with vertical layout
Inner view with horizontal view
Inner view with vertical view
Inner view with horizontal view

Figure 50: Form showing general layout of views.

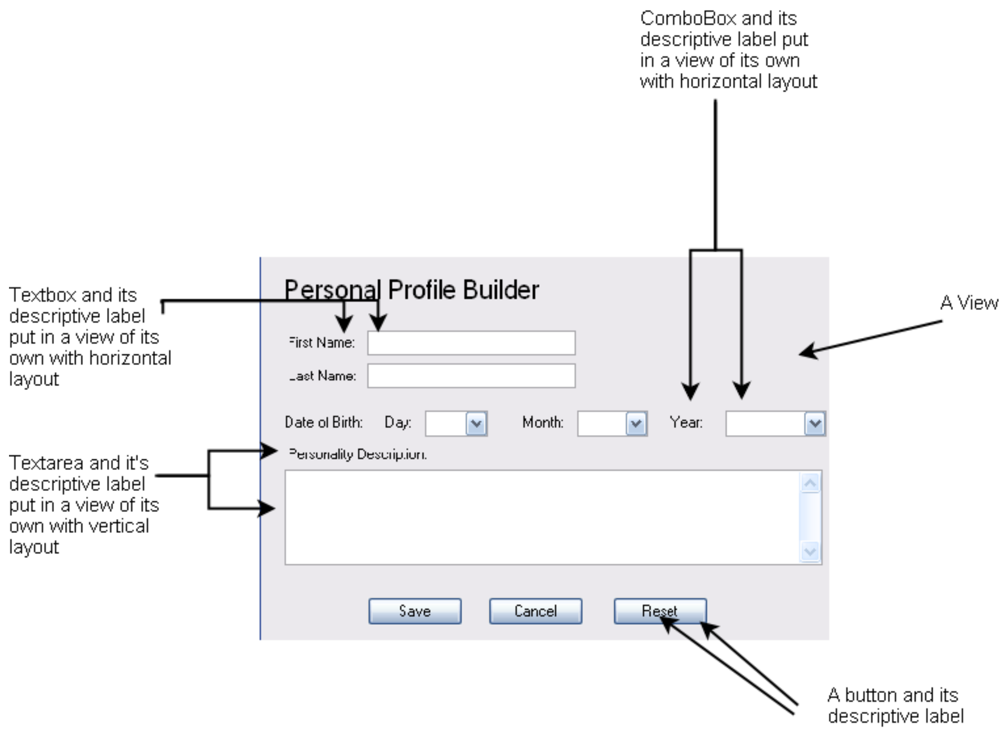

ComboBox and its descriptive label put in a view of its own with horizontal layout
Textbox and its descriptive label put in a view of its own with horizontal layout
Personal Profile Builder
A View
First Name:
Last Name:
Date of Birth: Day: Month: Year:
Personality Description:
Textarea and it's descriptive label put in a view of its own with vertical layout
Save
Cancel
Reset
A button and its descriptive label

Figure 51: Diagram showing a sample view "containing" GUI components and their descriptive labels.

7.7.1.2.3 Different types of GUI components

Along with modelling the underlying care plan instance data and modelling the graphical user interface for the patient application, the VPAM also needs to allow a designer to model the different types of interactions with the underlying care plan instance that are possible through the designed GUI. Ordinarily this would require an entire event modelling system, which would allow a designer to model complex events that arise from interacting with the GUI of the patient application, however for the VPAM this was considered to be unnecessary. Through considerations of the functionality that was required for the patient application it was discovered that there are four main types of interactions that were needed in order to model most of the functionality of the patient application. These interactions included:

1. GUI components to ***Display*** Care Plan Instance data
2. GUI components that ***Navigate*** to other views and screens when clicked
3. GUI components that take in ***Input*** that is to be updated with the care plan instance
4. GUI components that actually ***Update*** the entered input on the view or screen to the underlying care plan instance.

The type of a GUI component is captured with the property ViewType as shown in the meta-model shown in figure 49.

No doubt that there are other functionalities that the patient application has for which the events and interactions cannot be modelled through the VPAM and hence cannot be auto generated through the patient application generator, such as code and functionality for throwing reminders for health care activities, or the code and functionality for synchronising patient data. Therefore the generated application will have most of its code generated; however some of the complex functionality would still need to be hand programmed.

Examples of GUI components for each different type of interaction can be seen below, with a detailed context specific example shown in the next chapter.

Display type components:

- Labels
- Simple list
- Etc. (Any component which does not have any interaction besides simply viewing)

Navigate type Components

- Lists
- Buttons
- Etc (Any components that on clicking navigates to another screen or view)

Input type components:

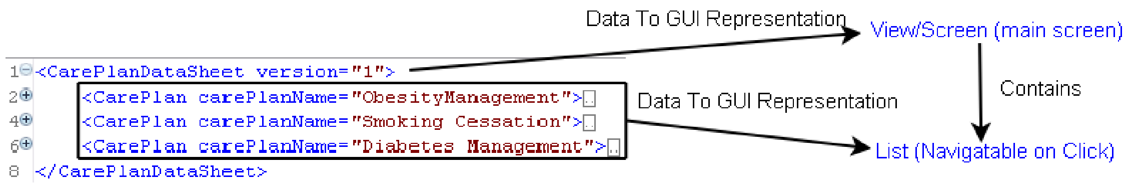
- Textboxes
- Textareas
- Comboboxes
- Radiobuttons
- Checkboxes
- Etc.

Update type components:

- Buttons
- Etc (or any component which is suppose to write inputted data on the page to the underlying care plan instance).

7.7.1.3 Mapping Care Plan Instance Data to its GUI Representation

Once the care plan instance data and its GUI have been modelled, then the next step would be to map the care plan data and its properties to its respective graphical user interface representation. For this purpose the "data object associated with view" relationship was introduced in the VPAM meta-model. Through this relationship, a designer can specify how each part of the care plan instance should be graphically represented to the patient in the patient application, and by specifying the different interaction types of the GUI elements (as described above), the designer can also specify how the patient can use the GUI element to interact with the underlying care plan instance property/data that it represents. The idea of mapping XML care plan instance data to their respective GUI elements can be observed in figure 52 below, where the idea can be used to model the main screen for the patient application, which represents the care plan data sheet of the patient, this view contains a list that contains all the care plans that are child elements of the care plan data sheet node, the list being of navigate type, i.e. clicking on a list item would navigate you to another page, probably a page showing more detail about the care plan clicked on. In a similar fashion the remainder of the care plan instance elements in the XML schema shown in section 7.5 can be mapped to their respective GUI components, chapter 8 shows a more in depth example using the implemented patient application modeller.



```
<CarePlanDataSheet version="1">
    <CarePlan carePlanName="ObesityManagement">
    <CarePlan carePlanName="Smoking Cessation">
    <CarePlan carePlanName="Diabetes Management">
</CarePlanDataSheet>
```

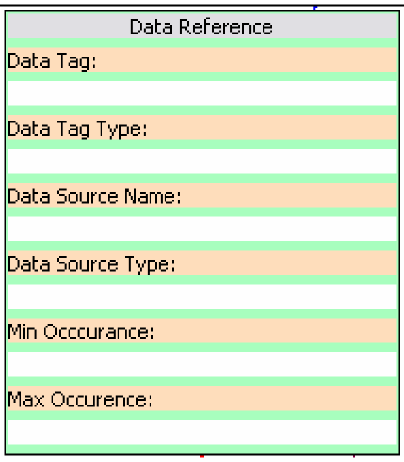
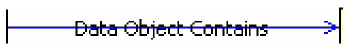
Data To GUI Representation
View/Screen (main screen)
Contains
Data To GUI Representation
List (Navigatable on Click)

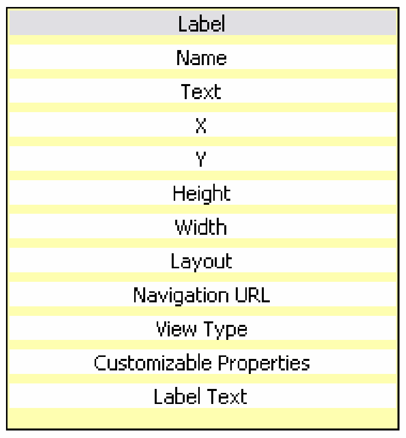
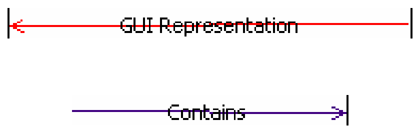
Figure 52: Diagram showing the conceptual mapping of care plan instance data in XML to GUI elements.

7.7.2 Graphical Notation for the VPAM

The graphical notation of the VPAM was very similar to the visual care plan modelling language, as the VPAM was also built in the Marama visual language development tool set, i.e. all modelling elements of the VPAM being represented by colour coded boxes, and relationships represented as lines. There are no doubt many more suitable visual metaphors to represent the modelling elements of the VPAM, particularly for representing the different GUI elements, metaphors similar in style perhaps to the Visual Studio Windows Form designer, however once again due to a great constraint of time more research was not invested in developing these more advanced visual metaphors, instead a simple box and line representation was used as shown in the table below.

Table 2: Table showing the graphical notation used in the visual patient application modeller for the data reference object, the GUI elements and the relationships between these two components.

Components	Relationships
 Data Reference Data Tag: Data Tag Type: Data Source Name: Data Source Type: Min Occcurance: Max Occurence:	

 Label Name Text X Y Height Width Layout Navigation URL View Type Customizable Properties Label Text	

The table above shows two main shapes, the first one is the shape for the data reference object, this object is used to model the care plan instance XML structure as described in the above sections, the second shape is an example of a GUI elements representation in the VPAM, the example in this case is a label, however all GUI elements have been colour coded light yellow as shown above, and all data references colour coded to light green.

7.8 Summary

This chapter has discussed the design of the 4 main components of the care plan modelling system, which include the visual care plan modelling language, the care plan instantiation tool, the patient application and the visual patient application model. Two of the components of the care plan modelling system, namely the care plan modelling language and the patient application modeller were implemented in the form of visual languages, the main advantage of which is that users get immediate feedback on any modelling work they do, it is a more natural way to do modelling and lastly it has a much lower learning curve than other textual wizard based approaches.

For both the care plan modelling visual language as well as the patient application modeller, the meta-models have been discussed in detail, along with the graphical notation that was employed. The graphical notation that is suggested for both visual languages include a simple colour coded box and line scheme, it is recognised that better visual metaphors do exist and indeed would make the visual languages much richer, however due to a timing constraint these metaphors were not investigated and developed.

The design for the instantiation tool focussed on extending the object model of the care plan template to include appointments, and also to extend the object model of routines to decorate them with more specific hourly, daily and monthly routines. The resulting care plan instances that are emitted from the tool were designed to be represented as XML documents with a fixed schema that largely mimics the object model of the care plan instance.

The main design issues discussed above with regard to the patient application include the architecture of the patient application. The architecture chosen for the patient application includes a client server partially connected architecture.

In this architecture the client is a fat client with local persistence and contacts the server intermittently for either processing, updating or retrieving of care plan data.

Chapter 8 - Implementation of the Care Plan Modelling System

8.1 Introduction

The purpose of this chapter is to explain the implementation of the care plan modelling system. As described in earlier chapters the care plan modelling system is made up of 4 main components, the first being the visual care plan modelling language, the second being the care plan instantiation tool, the third is the visual patient application modeller, and finally the accompanying patient application generator. The system consists of two visual languages these are the visual patient application modeller, and the visual care plan modelling language. Each of these languages was implemented in the Marama meta-tool set, with the care plan instantiation tool being implemented as a desktop application in Visual Studio 2005. Accompanying the visual patient application modeller, a visual patient application generator was implemented, which would enable the generation of patient applications in OpenLaszlo from the visual patient application models. Before the patient application generator for OpenLaszlo was implemented, the patient application was hand implemented in order to be used as a reference application to design and implement the OpenLaszlo generator for the patient application.

8.2 Implementing Visual Languages

This section describes the way in which the two visual languages part of the care plan modelling system, i.e. the visual care plan modelling language (VCPML) and the visual patient application modeller (VPAM) were implemented using the Marama visual language development environment.

8.2.1 Choice of Visual Language Development Environments

There are indeed many visual domain specific language development environments but for the purpose of developing the two domain specific languages (the VCPML and VPAM) part of the care plan modelling system, the Marama visual language development tool set was used. Marama is a specialised eclipse based tool set for modelling domain specific visual languages as well as the code generation associated with these languages from start to end. Once the language is designed it can be deployed as another instance of eclipse, where by Marama generates a custom editor with the domain specific visual language designed loaded in the editor ready to go (Liu 2007). The main reason for using Marama is that it possessed relatively advanced features for developing domain specific visual languages, even when compared to commercial products such as visual studio's DSM tools, and MetaEdit+ meta case tools, in addition to this it was designed in house to the University of Auckland, and hence the learning curve was less steeper as the designers of the system were easily contactable, reducing the time for development.

There were many alternatives to using Marama, the first one being Microsoft's Visual Studio 2005's Domain Specific Modelling (DSM) Tools, however at the time the development work was being conducted it was felt that Marama had more advanced features than VS DSM tools, mainly in the area of shape design for the visual aspect of the domain specific visual language more information on Microsoft DSM tools can be obtained from (Microsoft 2007). Other alternatives include MetaEdit+, which is a very advanced domain specific visual language development environment, however due to the fact that this is a commercial product a license could not be purchased at the time to experiment and further evaluate it, more information about MetaEdit+ can be found at (MetaCase 2007).

8.2.2 Process of Implementing a Visual Language in Marama

The process of implementing a domain specific visual language in Marama can be divided into four main stages (Liu 2007):

1. Model the domain for the visual language using the meta-modeller in Marama
2. Model the shapes for the domain elements in the Marama shape designer
3. Map the shapes to their domain model elements in the Marama view designer
4. Deploy the visual language as another eclipse instance.

As discussed above two components of the care plan modelling system are composed of domain specific visual languages the first being the care plan modelling visual language, the second being the visual patient application modeller, aimed at designing the patient application to host care plan instances for the patient to interact with. This section describes the implementation of these two languages in Marama. For each stage in the implementation process described above, the implementation of both languages is discussed instead of dealing with them separately, the main reason for this is simply to avoid a great deal of repetition as the implementation of the two languages only differ in their actual design (i.e. their actual meta models and shapes), however the implementation process is exactly the same.

8.2.2.1 Modelling the domain for the visual language

In chapter 7, it can be seen that the domain for the VCPML and VPAM, are expressed as UML models, transferring these into Marama domain models is a very simple process as Marama's meta-modeller is based on the UML concept, in that classes are modelled by "Entities" and relationships are modelled by "Associations" (Liu 2007). Figure 53 below shows a sample visual language project created in Marama, related to modelling which trees give which type of fruit, in this it can be seen that the tree and the fruit are entities with attributes of TreeName and FruitName respectively, i.e. the UML equivalent of classes, and the apple and orange are entities as well which inherit from the fruit entity. It can also be seen that the fruit has an "aggregation" relationship with the tree, and this relationship is represented as an association in the meta-model diagram shown in figure 53, called "treeHasFruits".

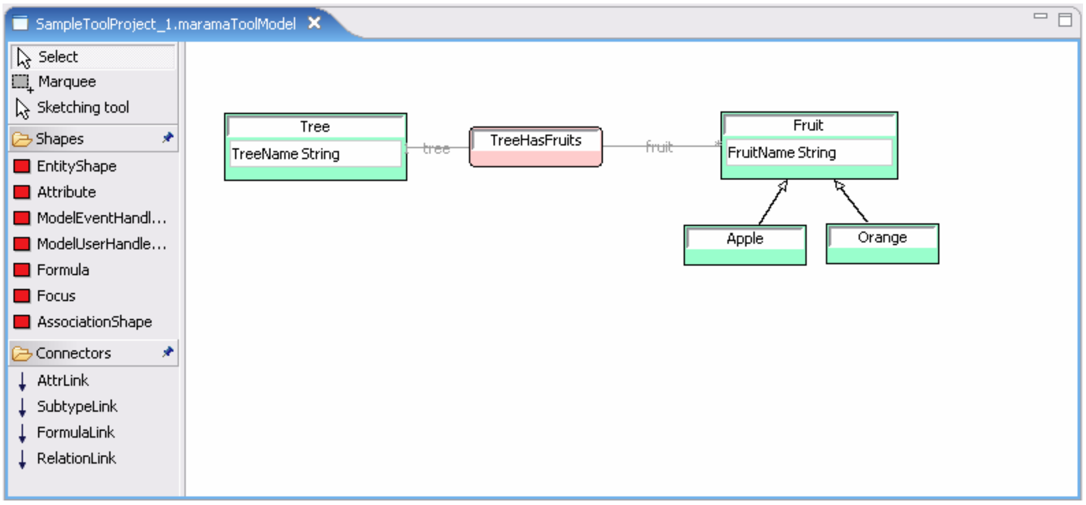


Figure 53: The Meta-Model builder view in Marama with a sample Fruit and Tree example.

In a similar fashion the UML models of the two domain specific visual languages, i.e. the VCPML and VPAM can be converted into Marama domain models as shown in figures 54 and 55 respectively.

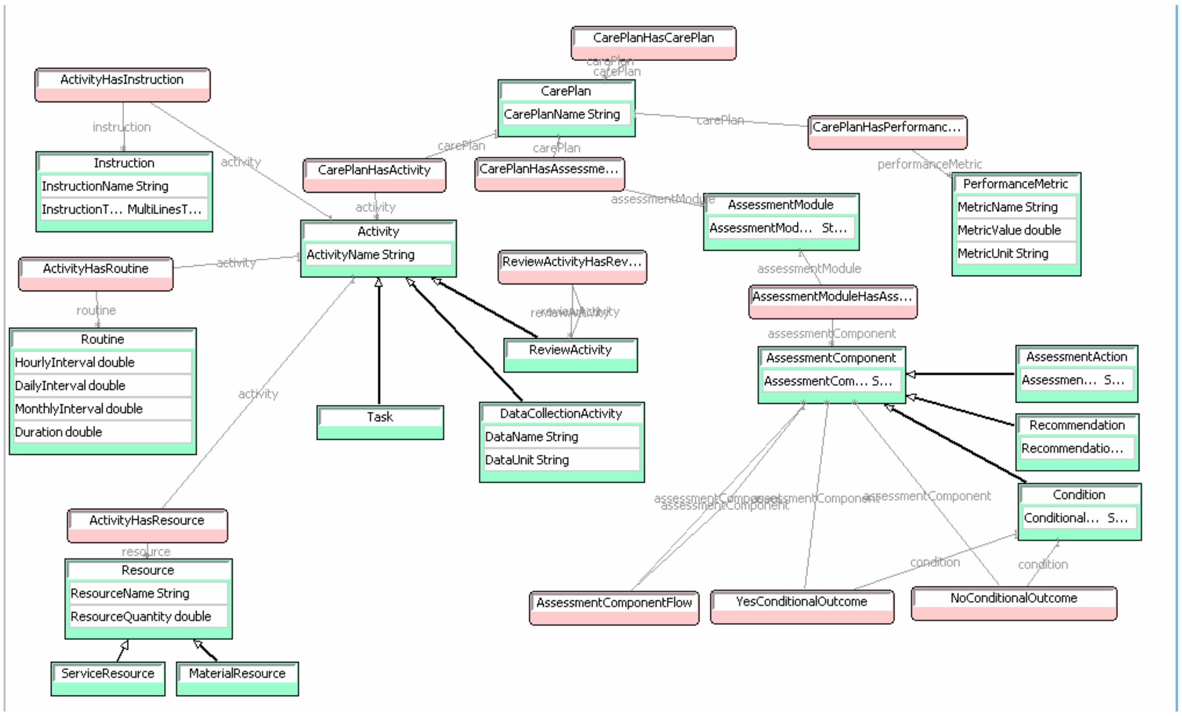


Figure 54: The domain model for the VCPML as built in Marama from the UML model in chapter 6.

It can be seen from the meta-model of the care plan modelling visual language shown in figure 54 that the care plan has four main relationships that pertain to the components which care plans are composed of, i.e. a care plan has other care plans, performance metrics, health activities, and assessment modules. Similarly health activities are in turn composed of other components such as instructions, routines, and resources, where an activity can be a simple task, a data collection activity or a review activity. Assessment modules in turn are decisional task flow, which are made up of assessment components directed by sequential flow relationships, where assessment components can be conditional components, assessment actions, and treatment recommendations. For more information on the care plan meta-model see chapter 5.

The meta-model of the visual patient application modeller shown in figure 55 below is composed of two main types of components, the first being data objects, and the second being GUI objects. The data objects pertain to data references that can be used to represent the structure of XML documents such as the care plan instance, well as the GUI components pertain to Views and other GUI components that inherit from Views which represent the graphical user interface elements which are to be mapped to the XML data structure. For more information on the meta-model and design of the visual patient application modeller see previous chapter.

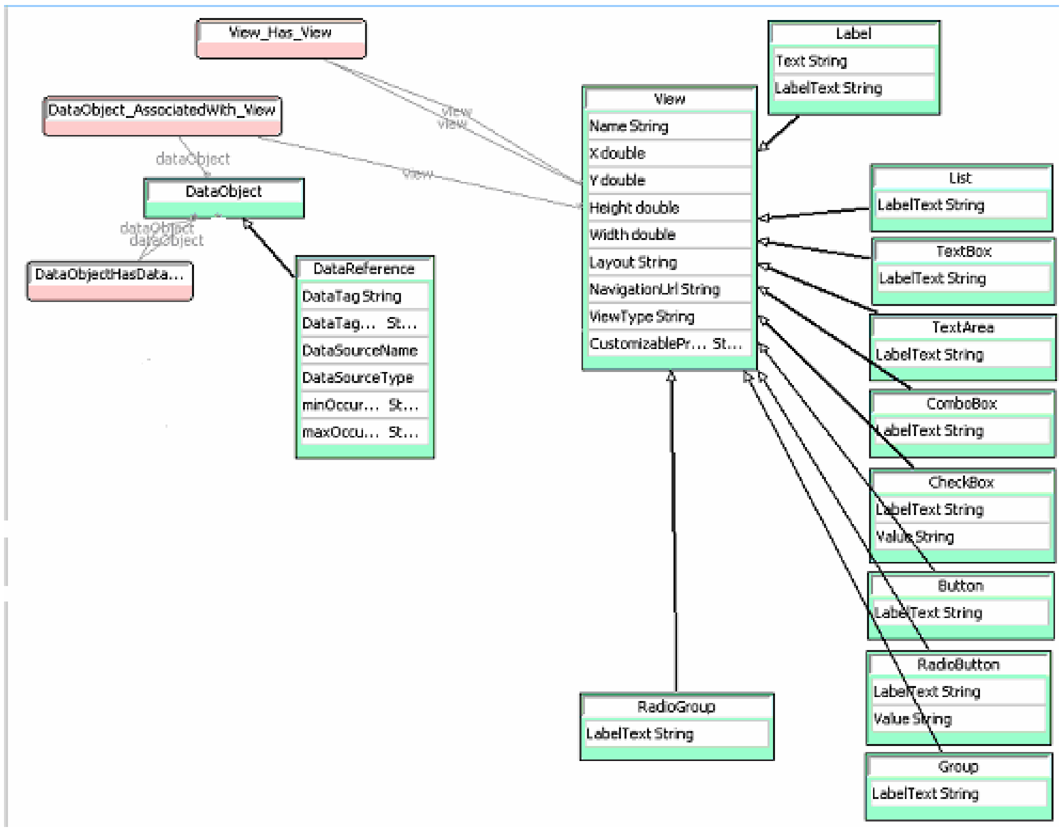


Figure 55: The VPAM domain model as built in Marama, based on the VPAM UML model shown in chapter 6.

8.2.2.2 Modelling the shapes for the visual languages

The shapes for the visual language can be modelled using the Marama shape designer. Using the shape designer, individual shapes for each domain element can be modelled, some of the shapes that are currently inbuilt into Marama include shapes such as rectangles, ellipses, rhombus, etc, along with items such as textboxes, labels, etc that can be added to these shapes (Liu 2007). In addition to this the connectors to represent the relationships between different domain elements can also be modelled here. Using this functionality the graphical notations that are shown in the previous chapter for each of the visual languages, i.e. the visual care plan modelling language and the visual patient application modeller were implemented, a screenshot of the shape designer can be seen in figure 56 below. In this figure, the modelling of a shape to represent the care plan element in the visual care plan modelling language is shown.

To avoid cluttering, the modelling of each of the shapes in the two visual languages is not shown here, however it does not differ greatly from the likes of figure 56 shown below. Figure 56 shows the modelling of the shape to represent care plan components, it can be seen that the component has a label representing what type of a component it is, another label representing the property of the component (i.e. Name), and finally a text box to represent the value of the property entered for the care plan component (i.e. the name value of the care plan).

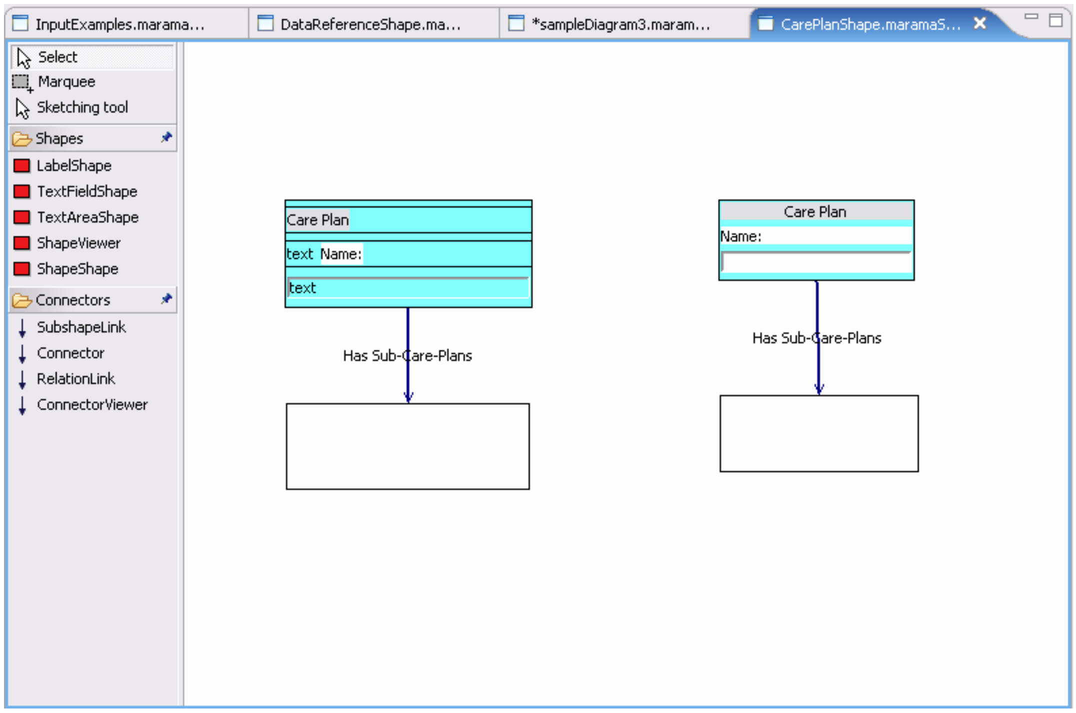


Figure 56: A screenshot of the Marama shape builder showing the shape to represent the care plan element in the VCPML.

8.2.2.3 Mapping domain elements to their respective shapes

Once the domain model has been modelled and the shapes created then the last step is to map the domain elements with actual visual shapes, this is done through the Marama view mapping interface (Liu 2007). The process involves mapping of domain components and their properties and domain relationships and their properties with their respective shapes and connectors. An example screenshot of the mapping process for the visual care plan modelling language is shown in figure 57. The screenshots for the mapping process of the visual patient application modeller are not shown here as the process is largely the same as shown in figure 57. In figure 57 the green rectangles represent the domain component mapping and the purple rectangles represent the domain relationship mapping. The cornered rectangles are the actual domain elements, and the rounded rectangles are the actual shapes and connectors. There fore it can be seen from figure 57 for example that the care plan meta-model component specified in figure 54 can be mapped to the care plan shape specified in figure 56, with the mapping represented by the yellow rectangles, through which even the properties of the care plan meta-model component can be mapped to the graphical elements of the care plan shape (e.g. mapping the name property to the name textfield), similarly other meta-model components are mapped to their respective shapes in both the view mapping of the visual care plan modelling language as well as the visual patient application modeller.

Once this mapping process is completed for both the visual languages then the visual languages are ready for use. To start modelling using any one of the languages, all that needs to be done is to start a new diagram, choosing the visual language definition file that is generated from the afore mentioned mapping process. A detailed example of modelling using both the visual languages is shown in the next chapter, which describes an example of the entire care plan modelling system in action.

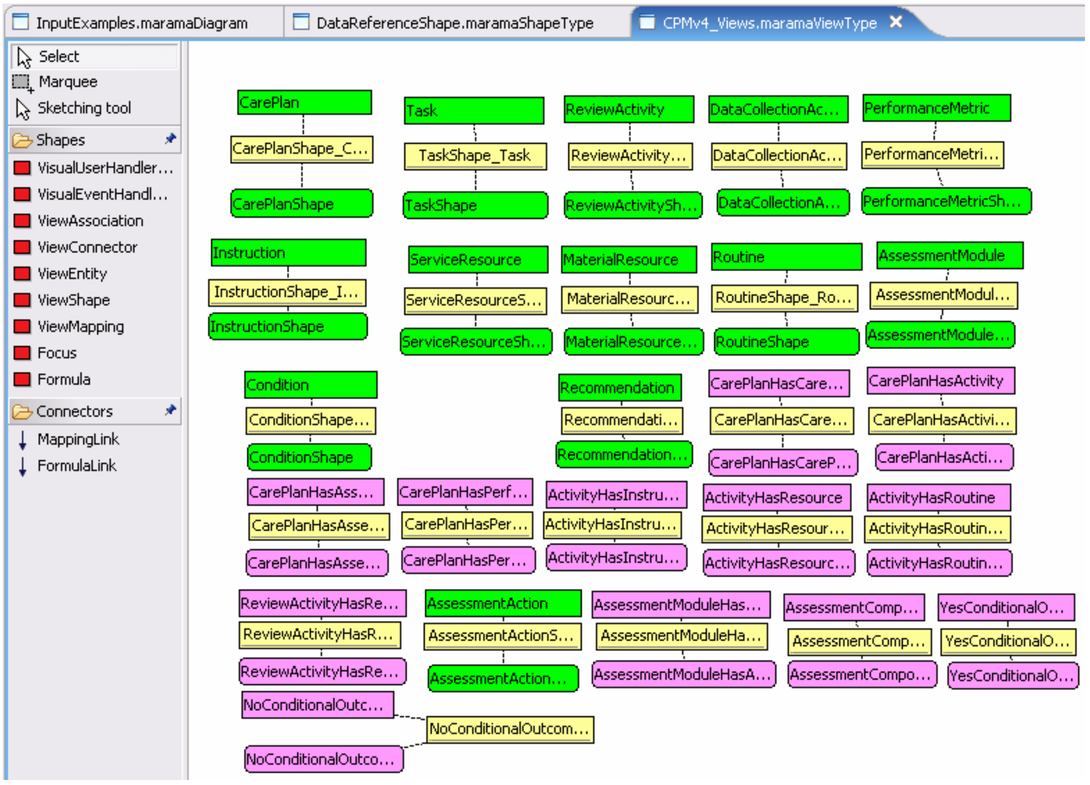
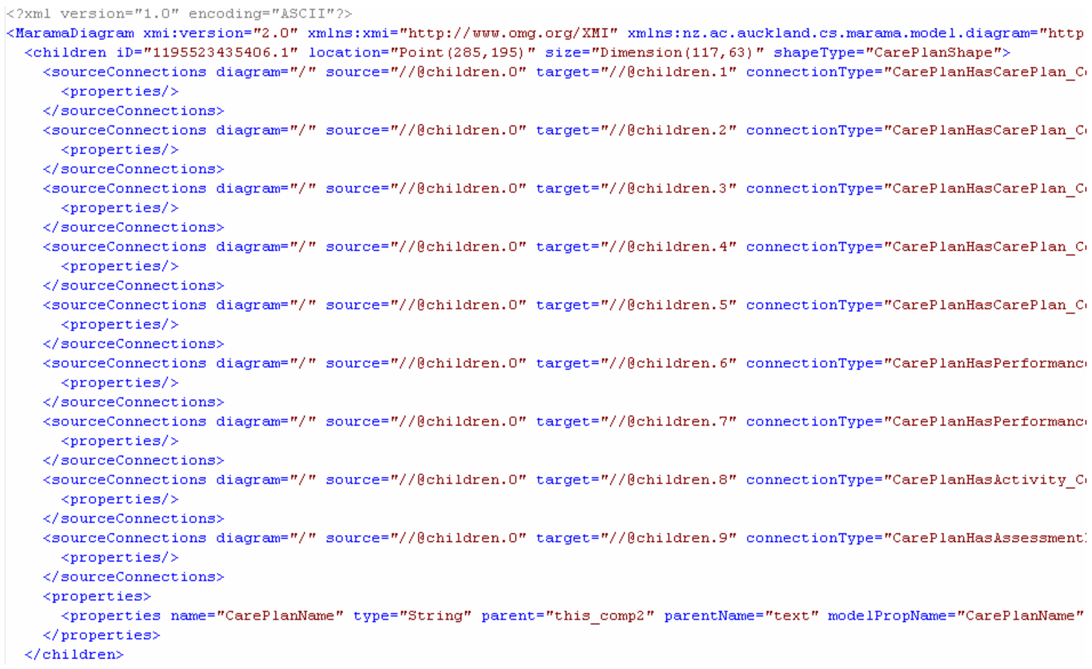


Figure 57: Screenshot of the view mapping for the visual care plan modelling language.

8.2.2.4 Transforming visual diagrams into readable XML documents

Once the visual languages are implemented as explained above, then Marama generates visual editors for each of the languages allowing modelling using the implemented language syntax and graphical notation. Each diagram whether it be a care plan model, or a patient application model is emitted as an XML document as shown in figure 58 below, which shows a sample care plan model diagram as an XML document. This XML document however is quite verbose and difficult to understand and hence process. Therefore in order to make the diagrams generated from the Marama visual languages, more readable and processable by program code, such as the care plan instantiation tool, and the patient application generator, an XSL style sheet was implemented in order to transform the raw diagram.

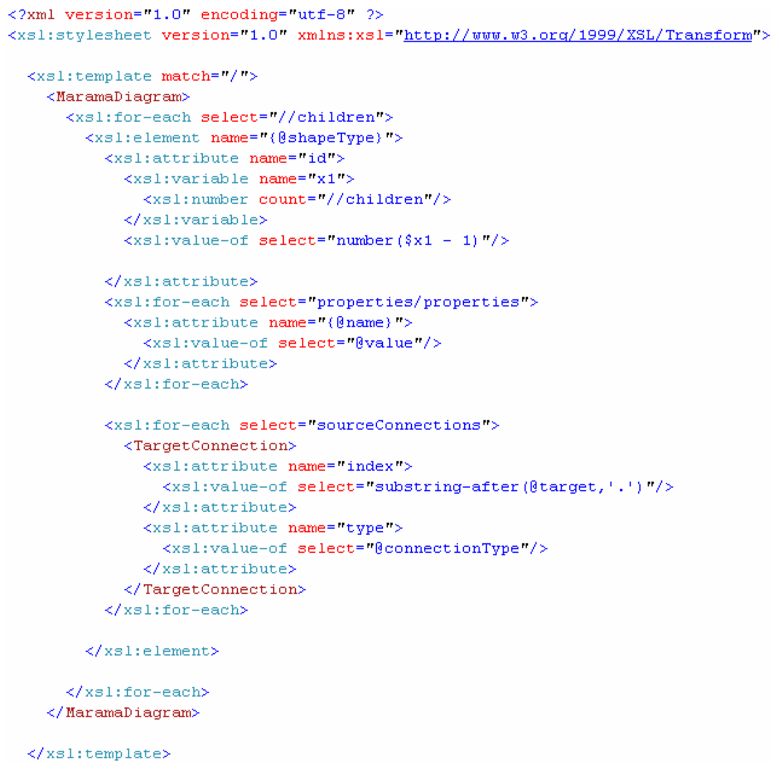
The XSL style sheet implemented to transform the Marama based visual language diagrams is shown in figure 59 below, it acts to cut out all the unnecessary XML information that the diagram contains, replacing generic tag information from the diagram XML file, such as calling each component “children”, with the actual names of the components specific to the visual language context. The Marama care plan model XML file is shown in figure 58 below with the sample transformed diagram file for a care plan model shown in figure 60, in a similar way patient application model diagrams are also transformed.



```
<?xml version="1.0" encoding="ASCII"?>
<MaramaDiagram xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:nz.ac.auckland.cs.marama.model.diagram="http
  <children iD="1195523435406.1" location="Point(285,195)" size="Dimension(117,63)" shapeType="CarePlanShape">
    <sourceConnections diagram="/" source="//@children.0" target="//@children.1" connectionType="CarePlanHasCarePlan_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.2" connectionType="CarePlanHasCarePlan_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.3" connectionType="CarePlanHasCarePlan_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.4" connectionType="CarePlanHasCarePlan_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.5" connectionType="CarePlanHasCarePlan_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.6" connectionType="CarePlanHasPerformanc
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.7" connectionType="CarePlanHasPerformanc
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.8" connectionType="CarePlanHasActivity_C
      <properties/>
    </sourceConnections>
    <sourceConnections diagram="/" source="//@children.0" target="//@children.9" connectionType="CarePlanHasAssessment
      <properties/>
    </sourceConnections>
    <properties>
      <properties name="CarePlanName" type="String" parent="this_comp2" parentName="text" modelPropName="CarePlanName"
    </properties>
  </children>
```

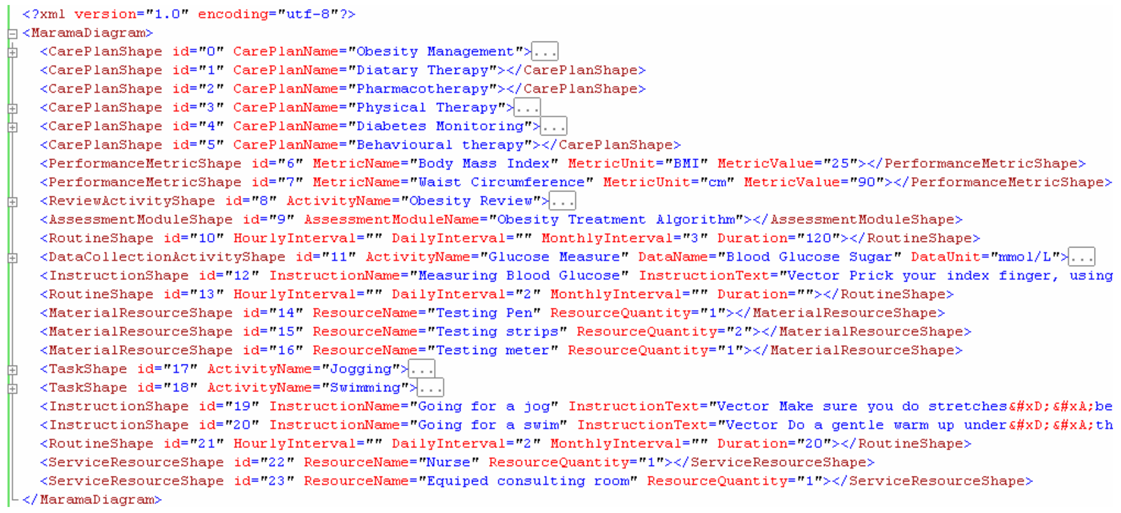
Figure 58: The XML generated by the Marama from a diagram modelled in a Marama visual language editor.

It can be seen from the sample Marama care plan diagram file that all components in the diagram are stored in nodes called "children", with the connections that the components have stored in "sourceConnections" nodes, and the properties of each component stored in "properties" node. The XSL transformation style sheet searches for these tags and transforms them into domain specific tags. It firstly checks for nodes called children and replaces these with the actual type of the shape/component that the children node represents, for example CarePlanShape, or PerformanceMetricShape etc. Then it searches for the properties tag for that node and converts all the properties in that node into XML attributes of the component, for example name for the care plan component. Then finally it searches for all the source connections that the components may have and resolves these and stores them in the component element under the "TargetConnections" node. The final transformed XML for a care plan modelling diagram can be seen in figure 60, with the XSLT sheet shown in figure 59 below.



```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <MaramaDiagram>
      <xsl:for-each select="//children">
        <xsl:element name="{@shapeType}">
          <xsl:attribute name="id">
            <xsl:variable name="x1">
              <xsl:number count="//children"/>
            </xsl:variable>
            <xsl:value-of select="number($x1 - 1)"/>

          </xsl:attribute>
          <xsl:for-each select="properties/properties">
            <xsl:attribute name="{@name}">
              <xsl:value-of select="@value"/>
            </xsl:attribute>
          </xsl:for-each>

          <xsl:for-each select="sourceConnections">
            <TargetConnection>
              <xsl:attribute name="index">
                <xsl:value-of select="substring-after(@target,'.')"/>
              </xsl:attribute>
              <xsl:attribute name="type">
                <xsl:value-of select="@connectionType"/>
              </xsl:attribute>
            </TargetConnection>
          </xsl:for-each>

        </xsl:element>

      </xsl:for-each>
    </MaramaDiagram>

  </xsl:template>
```

Figure 59: XSL style sheets used to transform Marama visual language diagrams.



```
<?xml version="1.0" encoding="utf-8"?>
<MaramaDiagram>
  <CarePlanShape id="0" CarePlanName="Obesity Management">...
  <CarePlanShape id="1" CarePlanName="Diatary Therapy"></CarePlanShape>
  <CarePlanShape id="2" CarePlanName="Pharmacotherapy"></CarePlanShape>
  <CarePlanShape id="3" CarePlanName="Physical Therapy">...
  <CarePlanShape id="4" CarePlanName="Diabetes Monitoring">...
  <CarePlanShape id="5" CarePlanName="Behavioural therapy"></CarePlanShape>
  <PerformanceMetricShape id="6" MetricName="Body Mass Index" MetricUnit="BMI" MetricValue="25"></PerformanceMetricShape>
  <PerformanceMetricShape id="7" MetricName="Waist Circumference" MetricUnit="cm" MetricValue="90"></PerformanceMetricShape>
  <ReviewActivityShape id="8" ActivityName="Obesity Review">...
  <AssessmentModuleShape id="9" AssessmentModuleName="Obesity Treatment Algorithm"></AssessmentModuleShape>
  <RoutineShape id="10" HourlyInterval="" DailyInterval="" MonthlyInterval="3" Duration="120"></RoutineShape>
  <DataCollectionActivityShape id="11" ActivityName="Glucose Measure" DataName="Blood Glucose Sugar" DataUnit="mmol/L">...
  <InstructionShape id="12" InstructionName="Measuring Blood Glucose" InstructionText="Vector Prick your index finger, using
  <RoutineShape id="13" HourlyInterval="" DailyInterval="2" MonthlyInterval="" Duration=""></RoutineShape>
  <MaterialResourceShape id="14" ResourceName="Testing Pen" ResourceQuantity="1"></MaterialResourceShape>
  <MaterialResourceShape id="15" ResourceName="Testing strips" ResourceQuantity="2"></MaterialResourceShape>
  <MaterialResourceShape id="16" ResourceName="Testing meter" ResourceQuantity="1"></MaterialResourceShape>
  <TaskShape id="17" ActivityName="Jogging">...
  <TaskShape id="18" ActivityName="Swimming">...
  <InstructionShape id="19" InstructionName="Going for a jog" InstructionText="Vector Make sure you do stretches&#xD;&#xA;be
  <InstructionShape id="20" InstructionName="Going for a swim" InstructionText="Vector Do a gentle warm up under&#xD;&#xA;th
  <RoutineShape id="21" HourlyInterval="" DailyInterval="2" MonthlyInterval="" Duration="20"></RoutineShape>
  <ServiceResourceShape id="22" ResourceName="Nurse" ResourceQuantity="1"></ServiceResourceShape>
  <ServiceResourceShape id="23" ResourceName="Equiped consulting room" ResourceQuantity="1"></ServiceResourceShape>
</MaramaDiagram>
```

Figure 60: A sample of a care plan visual language model after it is transformed by the XSL style sheet shown in figure 7.

8.3 Implementing the Care Plan Instantiation Tool (CPIT)

As described earlier the purpose of the care plan instantiation tool is to allow the health care professional to load a care plan template and fine tailor it to the patient by selecting and deselecting parts of the care plan specific for the patient and further more setting detailed timetables for the patient to conduct health care activities. The design of the care plan instantiation tool is discussed in chapter 7; the actual modules of implementation are separated into the actual domain code functionality and the graphical user interface functionality as shown in figure 61 below.

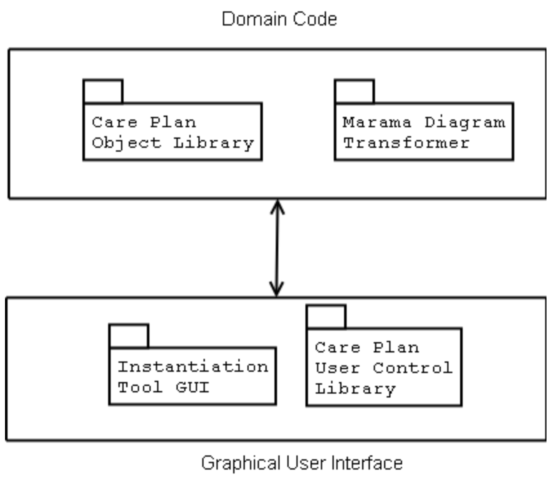


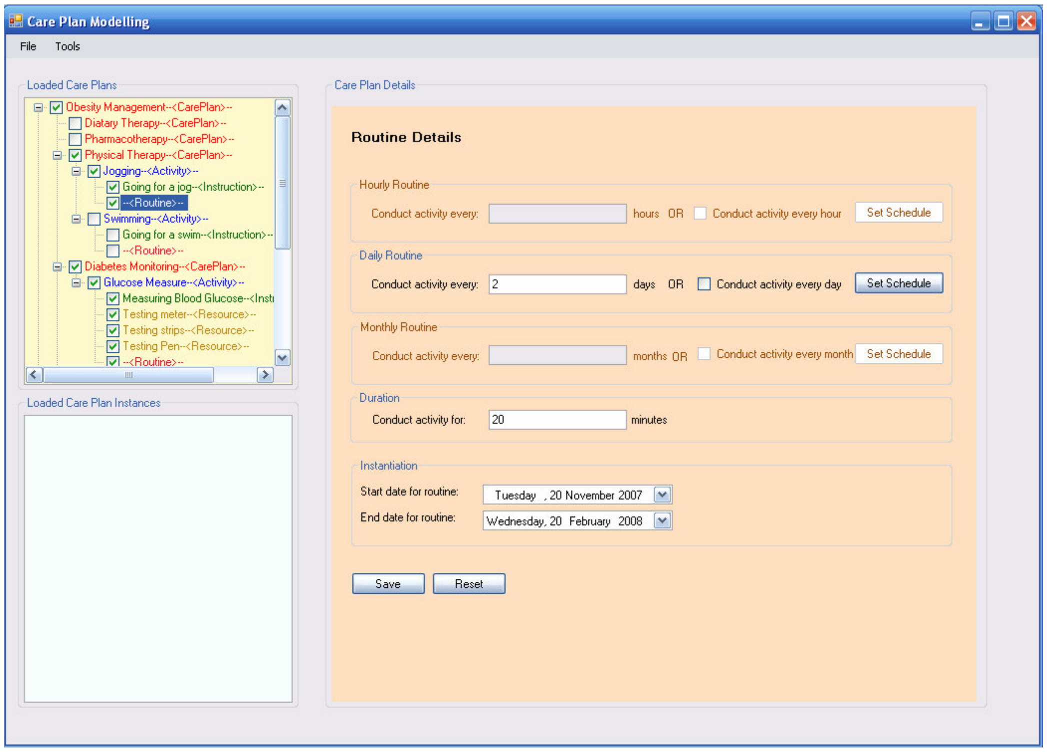
Figure 61: Modules implemented as part of the care plan instantiation tool.

The domain code implemented for the care plan instantiation tool can be roughly split up into two packages, the first being the entire care plan object library, and the second being the Marama diagram transformer. The care plan object library consist of the classes for all the care plan components expressed in the care plan OOA model shown in chapter 6, as well as the classes for the appointment object and the routine decorators that are additions for the care plan instance OOA model, also discussed in chapter 6. The main reason for implementing this library, was to ease the processing of a care plan model diagram when it is passed to the care plan instantiation tool, in that all the processing can be dealt with elegantly by the objects of the classes that correspond to each of the care plan components in the care plan model diagrams.

The second module of domain functionality is related to writing a set of utility classes that would facilitate the transformation and loading of the care plan model diagrams that are Marama based into the care plan instantiation tool. The functionality was implemented through the usage of the XSL style sheets to transform the Marama XML files into more readable XML data files, then this easy to process XML care plan data is transformed into a series of objects from the care plan object library and loaded into the GUI of the care plan instantiation tool, for the user of the application to interact with.

The implementation of the graphical user interface of the care plan instantiation tool was similarly split up into two parts. The first involved the development of the library of user controls which would allow a user to modify the different components part of the care plan model, such as care plans themselves, activities, performance metrics, routines etc through the care plan instantiation tool. A user control was implemented for each component in the domain model of the care plan. The second part of the GUI involved the actual desktop interface which used the user controls to provide the functionality of interacting with the care plan model. The entire GUI interface can be seen in figure 62 below, with example of its usage discussed in the following chapter.

It can be seen that the GUI for the care plan instantiation tool is roughly divided into three main segments, the first segment in the light yellow panel describes the current care plan that is loaded into the tool, the light orange panel displays the user controls for the right care plan component when clicked on in the yellow panel so that the properties of the care plan components can be easily edited, and finally the third light green component on the bottom left describes the care plan instance that is derived from the care plan model in the panel above.



Obesity Management--<CarePlan>--
Diatary Therapy--<CarePlan>--
Pharmacotherapy--<CarePlan>--
Physical Therapy--<CarePlan>--
Jogging--<Activity>--
Going for a jog--<Instruction>--
--<Routine>--
Swimming--<Activity>--
Going for a swim--<Instruction>--
--<Routine>--
Diabetes Monitoring--<CarePlan>--
Glucose Measure--<Activity>--
Measuring Blood Glucose--<Inst
Testing meter--<Resource>--
Testing strips--<Resource>--
Testing Pen--<Resource>--
--<Routine>--

Routine Details

Hourly Routine
Conduct activity every: hours OR Conduct activity every hour

Daily Routine
Conduct activity every: 2 days OR Conduct activity every day

Monthly Routine
Conduct activity every: months OR Conduct activity every month

Duration
Conduct activity for: 20 minutes

Instantiation
Start date for routine: Tuesday , 20 November 2007
End date for routine: Wednesday, 20 February 2008

Figure 62: The care plan instantiation application screenshot.

8.4 *Implementing a patient prototype*

In order to fully understand the language and the platform in which the patient application will be generated, the patient application was first hand coded as a prototype, so that it can be used as a reference to implement the patient application generator. Before a platform and a language were chosen for the patient application, a brief technology evaluation was conducted to firstly establish the different technologies that were available to develop mobile device applications and also to determine which technology will be the most suitable for implementing the patient application in. What follows is the process of implementation and the resulting prototype of the patient application.

8.4.1 Technology Evaluation

There are indeed many platforms, languages and frameworks that are out there that could be used to implement the patient application, however for the purpose of the research done here only four of the popular technologies were reviewed and considered as a choice for developing the patient application. These technologies include MUPE, a mobile phone developing technology from Nokia, the .Net Compact framework for PDA and smart-phone type applications, XFORMS for device independent HTML based applications and OpenLaszlo for developing flash applications.

8.4.1.1 MUPE (Multi-User-Programming-Environment) - (MUPE 2007)

MUPE is a set of development tools/libraries released by Nokia to allow easy development of multi-user client server applications for mobile phones (MUPE 2007). Currently Mupe is used to develop mobile games that can run on most Java enabled mobile phones. There are two main parts of a MUPE application the first being a Server to be implemented in J2SE, the other being the client implemented in an in-house script developed by Nokia called MUPE-Script, which is XML based (MUPE 2007). The developed MUPE application can run on any mobile with a MUPE client installed, which is in the form of a single jar file. The advantage of this technology is that it is easy to install and run on almost any mobile that is java enabled, by simply downloading the MUPEClient.jar, with the GUI being immediately adaptable to the different displays of different mobile devices, the other advantage is that MUPE provides great support for interfacing between supporting technology, including APIs to facilitate secure communication between client and servers (MUPE 2007). The disadvantages of this technology are that firstly it is only applicable for developing applications for mobile devices, other than this it is also a disadvantage that MUPE has a strictly always connected service oriented architecture, which is unsuitable for developing the patient application, as the patient may face many periods when they are not connected to the internet, lastly the low familiarity with this development technology would mean a much higher development time.

8.4.1.2 .Net Compact Framework

The .Net compact framework was introduced by Microsoft, in order to allow development of applications to run on PDAs and smart-phones with the Windows CE based operating system. The .Net compact framework provides GUI components of the same look and feel as any other Windows desktop based application; however the functionality of the GUI components is limited. The .Net compact framework is integrated into the Visual studio development environment, and comes with its own lite database and server, specialised to run on PDAs and smartphones called SQL server CE. The advantages of using the .Net compact framework for developing the patient application is that firstly, there is a great familiarity, in both the development language as well as the development environment, in that the development language and environment is the same as developing a normal desktop application for Windows, where designing the GUI for a PDA application is as simple as dragging and dropping GUI elements onto the canvas. Other advantages include, a good support for having persistence on the mobile device through the lite database, along with support for synchronising data, therefore it is easy to implement applications with a partially connected architecture. The disadvantages of using this technology include, the fact that the applications will only run on Windows based operating systems, i.e. if the PDA has a Palm OS then the application will not run on it, the application will also not freely run on mobile phones, other than the smart phones with the Windows CE OS, and there are many mobile phones with operating systems other than Windows.

8.4.1.3 XFORMS - (W3CSchools 2007)

XForms is said to be the future replacement of HTML, and allows a flexible separation of GUI (i.e. the presentation layer) from the business logic and data access layer (W3CSchools 2007). XForms, provides the facility to construct the GUI of an application by defining its general properties, i.e. an input field, or a submit control. XForms does not require for a developer to specify what exact elements need to be used, i.e. textboxes, buttons etc, this decision is made automatically by the XForms renderer based on the device that the application is running on (W3CSchools 2007). In addition to this XForms provide the coding capability to define domain code in the form of actions and functions, with the possibility of even employing other scripts to add further functionality, with all data in XForms application being stored in XML (W3CSchools 2007). The obvious and main advantage of XForms is that it is completely device and platform independent, i.e. if you had the appropriate renderer an application written once could be run on multiple devices. The major disadvantage however, is that the technology is relatively new and currently browsers such as IE, and Firefox do not provide very good support for them. There are however third party rendering software/browsers, such as FormFaces, and XSmiles, both of which can render XForms on desktops with XSmiles having the capability to render XForms on PDAs as well, however the rendering is still very poor, and hence XForms was not used to develop the patient application (XML4Pharma 2007).

8.4.1.4 OpenLaszlo - (OpenLaszlo 2007)

OpenLaszlo is an upcoming development language, which allows developers to create rich flash applications to run on multiple flash enabled devices. The OpenLaszlo language, is an XML based language, with the capability of embedding JavaScript code to develop enhanced functions, with all data transporting and processing done as XML. Currently the OpenLaszlo development suite comes with two compilers one which compiles OpenLaszlo script to FLASH and another which compiles it to DHTML (OpenLaszlo 2007). One of the major advantages of using OpenLaszlo is that the Flash applications generated, run consistently on a wide range of devices and platforms that are flash enabled, including Windows, PocketPC, Mac OS, Linux, Solaris, etc without much change at all. Therefore once developed, a mobile device application can even run on a desktop (if need be), or an application designed for a flash enabled mobile phone, will also run on a PDA with the lite flash player installed.

One significant disadvantage of OpenLaszlo programs when compiled to flash, is that a flash application due to security reasons executes in a sandbox of security, which prevents any Flash application to write to the local disk, this can be particularly problematic, if the application is to have a partially connected architecture, as this requires some form of at least temporary persistence, where data can be securely stored when there is no connectivity. Through some research it was discovered that this problem can be easily overcome, either by the use of action script or through the use of specialised shared objects part of the Adobe Flash architecture, which allows controlled local disk writing (see the following sections on the patient application prototype architecture for more details) (Adobe 2007).

Once this disadvantage was dealt with, OpenLaszlo seemed like the ideal choice in comparison with the other technologies that were considered for the development of the patient application.

8.4.2 Patient application prototype

Once OpenLaszlo was chosen as the language for generating the patient application, then the patient application was implemented by hand in the OpenLaszlo language. The main reason for this step was to gain a better understanding of the language, which was needed in order to implement an application generator for OpenLaszlo, and also to provide a reference application, which could be consulted when implementing the finer aspects of the OpenLaszlo application generator. What follows in this section, is a more specific portrayal of the partially connected architecture of the patient application specific to the OpenLaszlo language, as well as the actual prototype that was developed.

8.4.2.1 Patient application architecture

The patient application architecture specific to the implementation in OpenLaszlo language is shown in figure 63. Despite the similarity to the partially connected architecture expressed in chapter 6, there are still some differences that are important to mark, the first among these differences is that the OpenLaszlo application is coded as one big file, and not as three separate layers, presentation, business and data access, the main reason for this lack of separation is due to a restriction of time. The other important difference that needs to be noted is that during periods of non-connectivity the care plan instance XML file is stored not in a database as shown in the partially connected architecture in chapter 6, but instead is stored as a Flash shared object of XML type, the main reason for this as mentioned above is that a flash application executes in a sand box of security, and hence writing to the local disk is not possible, however the Flash architecture provides shared objects which allow information to be securely written on the users computer, and retrieved when the application restarts.

Shared Objects are the Flash equivalent of cookies; however they possess a lot of advantages over cookies. The first such advantage is that unlike cookies shared objects do not have an expiry date, and can be kept onto the users computer till explicitly deleted, furthermore a cookie has a size restriction of 4000 bytes well as a shared object can be 100 kilobytes in size (TechRepublic 2007). In addition to this unlike cookies, shared objects in flash can have many different types, such as XML, string, Boolean, Object and even complex types such as class instances (TechRepublic 2007). Due to these advantages, the fact that Flash applications cannot directly write to the local file system was not considered to be a great drawback.

It is important to realise that due to a severe constraint in time, the logic to synchronise care plan instances with the remote server, as well as the actual code for writing data to the local disk through shared objects was not implemented in the patient application prototype presented here, as the sole purpose of this prototype was to increase the understanding of OpenLaszlo so that the patient application generator could be implemented. Furthermore an earlier research project done by the author saw the implementation of this use case, and hence it is known that it can be done with relative ease.

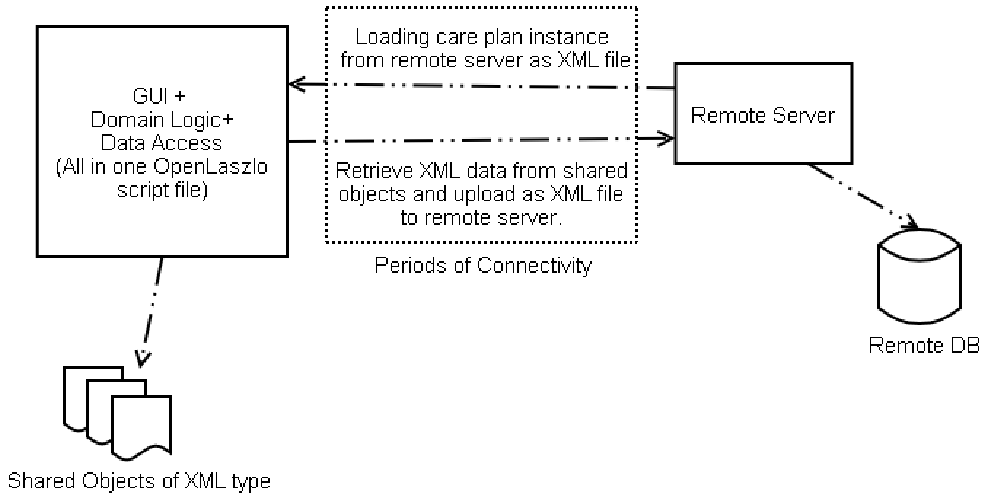
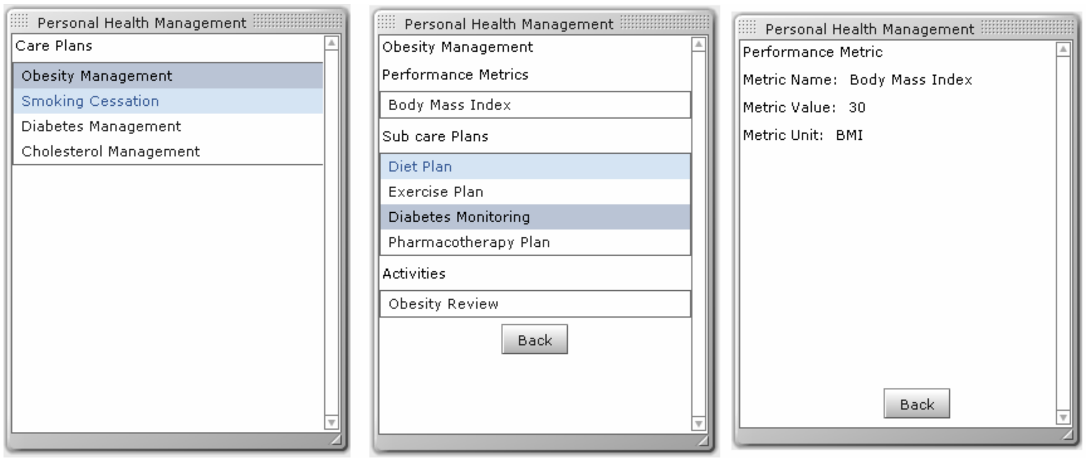

GUI +
Domain Logic+
Data Access
(All in one OpenLaszlo script file)
Loading care plan instance from remote server as XML file
Retrieve XML data from shared objects and upload as XML file to remote server.
Periods of Connectivity
Remote Server
Remote DB
Shared Objects of XML type

Figure 63: The architecture of the patient application prototype as implemented in the OpenLaszlo language.

8.4.2.2 The Prototype

This section show cases the patient application reference prototype developed in OpenLaszlo. The main screenshots of the application are shown in the figures below, starting from the left in figure 64, the first screen shows the patient all the care plan instances that they are on, in this case the patient is on four other care plans, an obesity management care plan, a smoking cessation care plan, a diabetes management care plan and a cholesterol management care plan. The next screen from the left in figure 64 shows the obesity management care plan, once the plan is selected in the prior screen. This screen shows all the performance metrics that the care plan has, i.e. the body mass index metric, the sub-care plans that the patient is on, and finally the activities that the patient is on. Clicking on the BMI metric on the care plan page takes the patient to the screen which shows more information about the metric, in this case it is the value and the unit. The patient can also click on one of the sub-care plans under the obesity management care plan, for example the diabetes monitoring care plan, this takes them to the page representing that care plan, i.e. the first screen from the left in figure 65. It can be seen on this screen that the plan has only one activity, i.e. the glucose measurement activity, clicking on this activity, takes the patient to the next screen from the left in figure 65, i.e. showing the details of the glucose measurement activity. This page shows the name of the activity, the type of an activity (i.e. task, review, or data collection), it shows the instructions that are available for the activity, and finally the resources that are needed for conducting the activity.

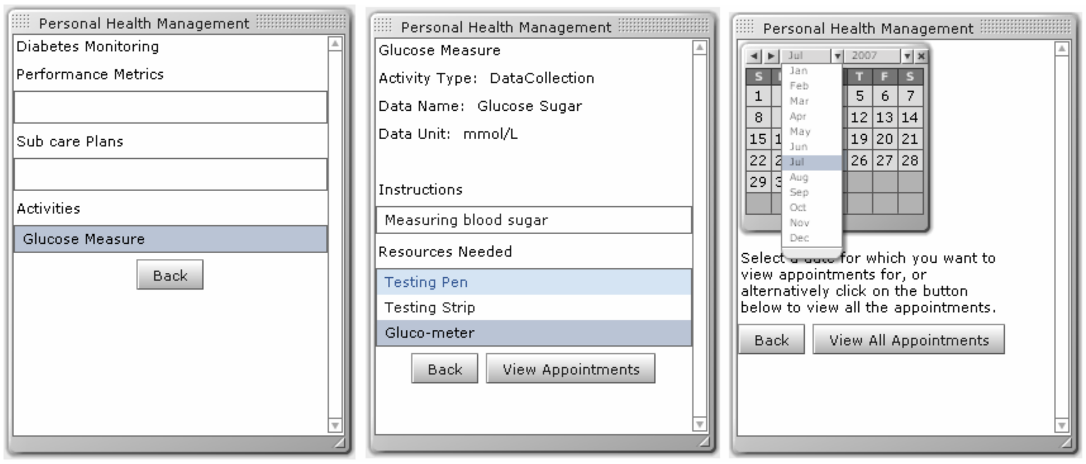


Metric Name: Body Mass Index
Metric Value: 30
Metric Unit: BMI

Figure 64: From the left the first screen is the main screen describing all the care plans the patient is on, then the Obesity management care plan main page, then the performance metric for BMI.

There are two ways in which a patient can see the appointment page of an activity the first way is to receive a reminder to conduct an activity, with the reminder immediately navigating the patient to the appointment screen, for the purpose of the prototype built here the reminder facility was not implemented. The second way to reach the appointment screen for the activity, is to search the appointments manually, by simply pressing the view appointments button on the second screen from the left on figure 65, the user is taken to the last screen from the left on figure 65, where they can either view all their appointments as shown in the first screen from the left in figure 66, or they can select the appointments they want to view according to date by using the calendar.

Once they click on the appointment they want, they are ready to do the appointment and fill in the information related to the activity that the patient application demands. The second screen from the left in figure 66 shows the appointment screen for the glucose measurement activity, it can be seen that the activity is of data collection type, and that it demands to the patient to enter in their glucose measure once they have measured it, and then it requires the patient to mark the appointment as either completed, not completed or not attempted (default), and then the appointment can be saved by clicking on the save button, whereby the underlying care plan instance is updated. The final screen from the left is the instruction screen that can be seen from the appointment screen guiding the patient how to measure their blood glucose sugar.



Personal Health Management

Diabetes Monitoring

Performance Metrics

Sub care Plans

Activities

Glucose Measure

Back

Personal Health Management

Glucose Measure

Activity Type: DataCollection

Data Name: Glucose Sugar

Data Unit: mmol/L

Instructions

Measuring blood sugar

Resources Needed

Testing Pen

Testing Strip

Gluco-meter

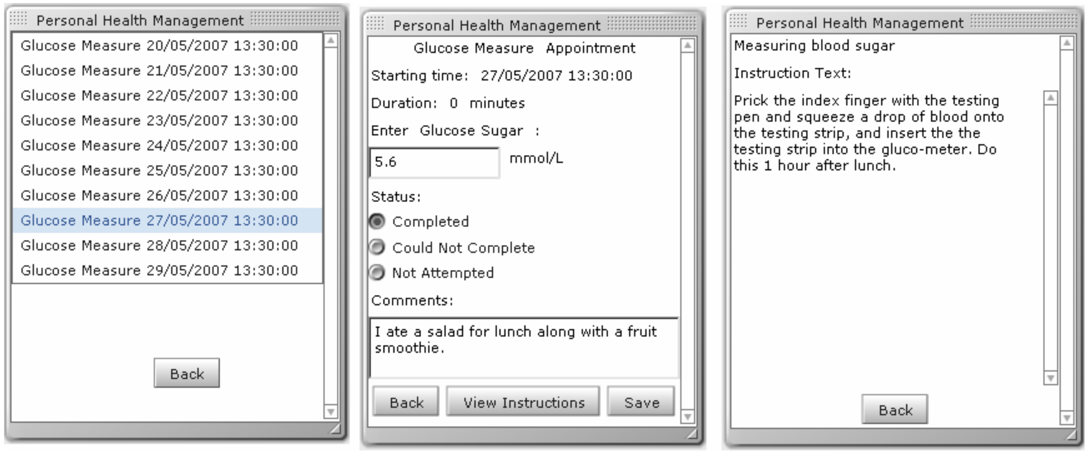
Back View Appointments

Personal Health Management

Select a date for which you want to view appointments for, or alternatively click on the button below to view all the appointments.

Back View All Appointments

Figure 65: From the left the first screen is the Diabetes monitoring care plan, the glucose measurement activity and finally the screen for selecting appointments for the activity.



Personal Health Management

Glucose Measure 20/05/2007 13:30:00
Glucose Measure 21/05/2007 13:30:00
Glucose Measure 22/05/2007 13:30:00
Glucose Measure 23/05/2007 13:30:00
Glucose Measure 24/05/2007 13:30:00
Glucose Measure 25/05/2007 13:30:00
Glucose Measure 26/05/2007 13:30:00
Glucose Measure 27/05/2007 13:30:00
Glucose Measure 28/05/2007 13:30:00
Glucose Measure 29/05/2007 13:30:00

Back

Personal Health Management

Glucose Measure Appointment

Starting time: 27/05/2007 13:30:00

Duration: 0 minutes

Enter Glucose Sugar :

5.6 mmol/L

Status:

[✓ Completed]

Could Not Complete

Not Attempted

Comments:

I ate a salad for lunch along with a fruit smoothie.

Back View Instructions Save

Personal Health Management

Measuring blood sugar

Instruction Text:

Prick the index finger with the testing pen and squeeze a drop of blood onto the testing strip, and insert the the testing strip into the gluco-meter. Do this 1 hour after lunch.

Back

Figure 66: From the left the first screen is for all the appointments, then the glucose appointment screen for one of the appointments, and then the instructions page for the appointment.

8.5 Implementing the patient application generator

Once a good understanding of OpenLaszlo was gained through development of the prototype patient application, then the next step was to implement the patient application generator which would automatically generate the patient application from the VPAM models that are generated. This section describes the anatomy of the patient application generator.

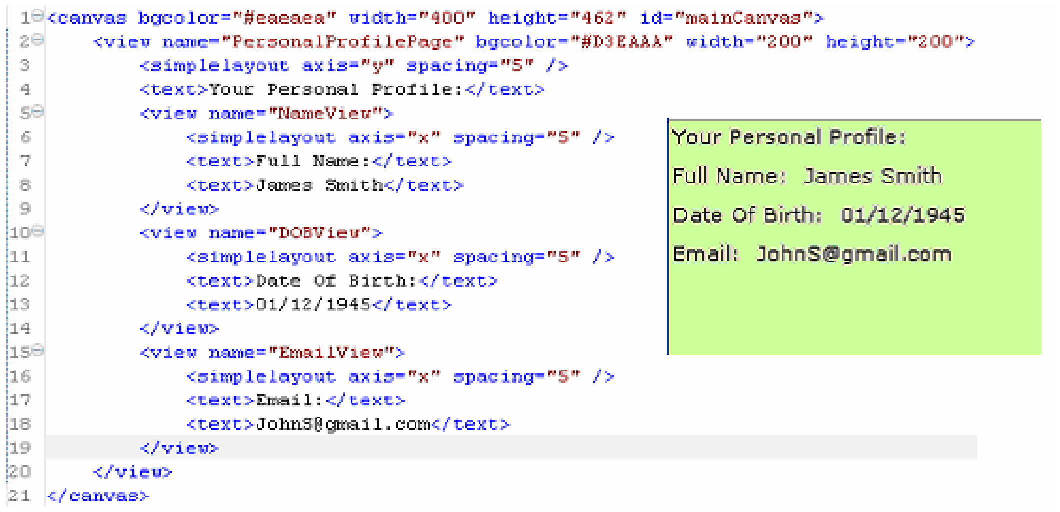
8.5.1 Important aspects to consider in the implementation of the patient application generator

Through developing the reference patient application in OpenLaszlo (exhibited in the section above), a lot of important aspects about the structure of the language and the structure of the programs written in the language were learned, which were vital in the implementation of the patient application generator. In order to implement the patient application there were three aspects that were most important for consideration:

1. The way in which graphical user interface elements can be coded in OpenLaszlo.
2. The way in which actual functionality can be coded in OpenLaszlo
3. The way in which access to XML data (in this case the care plan instance data) is handled in OpenLaszlo.

8.5.1.1 Coding graphical user interface elements in OpenLaszlo

OpenLaszlo is an XML based programming language, hence all the GUI code written in the language is actually written in XML as shown in figure 67 below. Figure 67 below shows the OpenLaszlo code for a sample personal profile page dummy. It can be seen that the code looks identical to any other structured XML file, where all the GUI elements are represented as XML nodes, such as views, and text labels, and layout specifications, where the attributes of each GUI elements are specified as actual XML attributes.



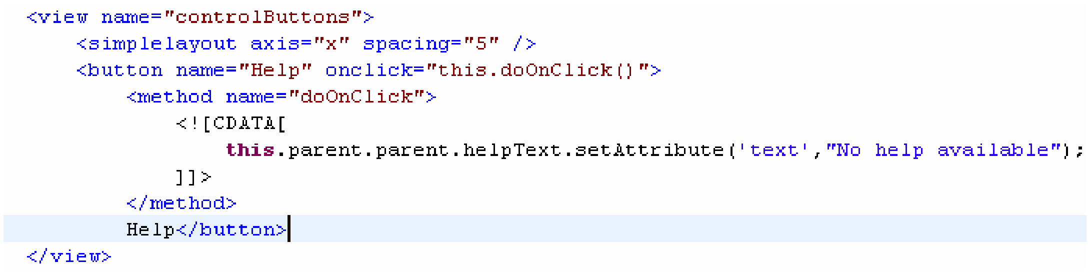
```
<canvas bgcolor="#eaeaea" width="400" height="462" id="mainCanvas">
    <view name="PersonalProfilePage" bgcolor="#D3EAAA" width="200" height="200">
        <simplelayout axis="y" spacing="5" />
        <text>Your Personal Profile:</text>
        <view name="NameView">
            <simplelayout axis="x" spacing="5" />
            <text>Full Name:</text>
            <text>James Smith</text>
        </view>
        <view name="DOBView">
            <simplelayout axis="x" spacing="5" />
            <text>Date Of Birth:</text>
            <text>01/12/1945</text>
        </view>
        <view name="EmailView">
            <simplelayout axis="x" spacing="5" />
            <text>Email:</text>
            <text>JohnS@gmail.com</text>
        </view>
    </view>
</canvas>
```

Your Personal Profile:
Full Name: James Smith
Date Of Birth: 01/12/1945
Email: JohnS@gmail.com

Figure 67: Code for a sample OpenLaszlo program, exhibiting the XML nature of the GUI code in the OpenLaszlo language as well as the screen generated from the code.

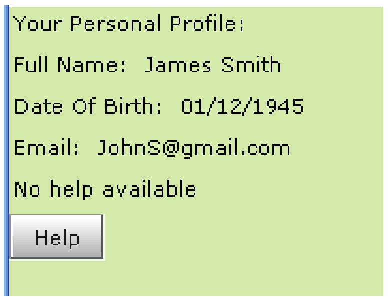
8.5.1.2 Coding actual functionality in OpenLaszlo

As with any language, there is always a need to have a facility to describe detailed code, such as coding events based on interactions with GUI elements such as buttons. Detailed functionality code in OpenLaszlo can be written as snippets of Java Script code embedded into the normally XML OpenLaszlo code in the form of CDATA sections of OpenLaszlo methods as shown in figure 68 below. The code shown in figure 68 below shows a help button added to the page shown in the section above, when the button is clicked it shows a message saying “no help is available”. It can be seen that the functionality of this event is coded in Java Script and embedded in a method as a CDATA section, the program page compiled from this code can be seen in figure 69.



```
<view name="controlButtons">
    <simplelayout axis="x" spacing="5" />
    <button name="Help" onclick="this.doOnClick()">
        <method name="doOnClick">
            <![CDATA[
                this.parent.parent.helpText.setAttribute('text',"No help available");
            ]]>
        </method>
        Help</button>
</view>
```

Figure 68: Sample code for the functionality of a help button in OpenLaszlo.

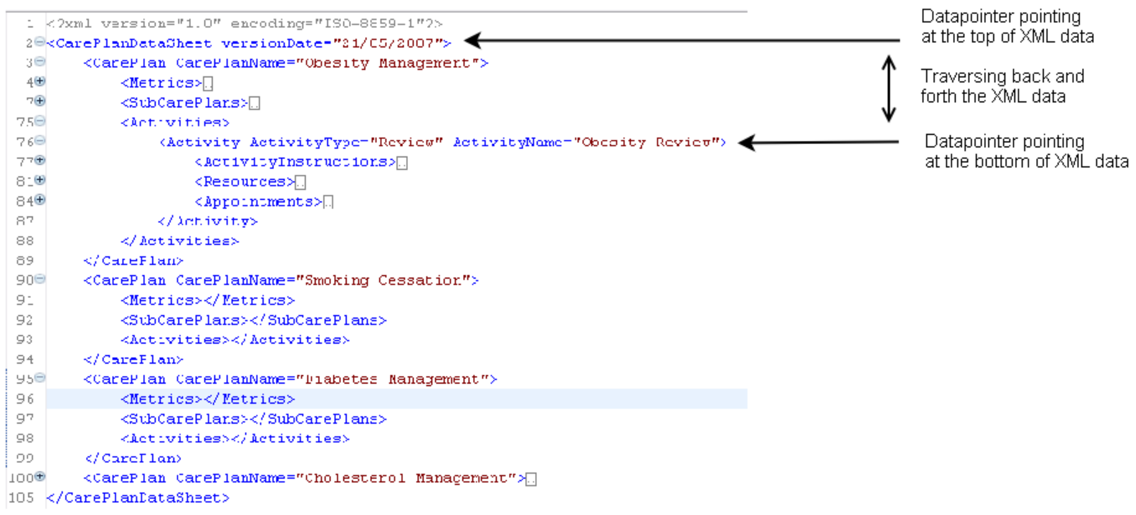


Your Personal Profile:
Full Name: James Smith
Date Of Birth: 01/12/1945
Email: JohnS@gmail.com
No help available

Figure 69: The compiled program for the sample help button code.

8.5.1.3 Coding data access code in OpenLaszlo

All data in OpenLaszlo is represented as XML, the care plan instance data itself is stored in an XML file. OpenLaszlo gives a method of easily traversing and accessing this XML based data through the use of "*datapointers*". An OpenLaszlo program can have multiple datapointers for any one XML file pointing at different locations (or otherwise different elements) in the XML file (Laszlo Systems Inc 2007). Furthermore a datapointer can traverse along the XML file in both directions, i.e. a datapointer pointing at the top most element (for example at the care plan data sheet node), and traversing to elements in the forward direction, as well as being able to traverse backwards to elements already visited. In addition to this the datapointer can allow the data to be viewed, new elements and attributes to be added and existing XML elements and attributes to be updated.


Datapointer pointing at the top of XML data
Traversing back and forth the XML data
Datapointer pointing at the bottom of XML data

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CarePlanDataSheet versionDate="31/05/2007">
    <CarePlan CarePlanName="Obesity Management">
        <Metrics>
        <SubCarePlans>
        <Activities>
            <Activity ActivityType="Review" ActivityName="Obesity Review">
                <ActivityInstructions>
                <Resources>
                <Appointments>
            </Activity>
        </Activities>
    </CarePlan>
    <CarePlan CarePlanName="Smoking Cessation">
        <Metrics></Metrics>
        <SubCarePlans></SubCarePlans>
        <Activities></Activities>
    </CarePlan>
    <CarePlan CarePlanName="Diabetes Management">
        <Metrics></Metrics>
        <SubCarePlans></SubCarePlans>
        <Activities></Activities>
    </CarePlan>
    <CarePlan CarePlanName="Cholesterol Management">
</CarePlanDataSheet>
```

Figure 70: Example care plan data with data pointers pointing at different parts of the XML data.

8.5.2 The patient application generator algorithm

Looking at the structure of the OpenLaszlo code and its language syntax, it can be seen that in order to generate the patient application in OpenLaszlo from a visual patient application definition, there are three items of code that need to be generated:

1. Code for the GUI components and their respective layouts that are specified in the VPAM definition of the patient application.
2. Code to provide the GUI components with the correct data pointers so that they can interact with the correct data in the care plan instance that has been specified in the mapping between the GUI and the data in the VPAM definition.
3. Code to actually specify the different interactions that the GUI components can actually do with the data once they have the correct data pointer.

What follows in the diagram below is the general algorithm which was implemented to generate the code for the patient application from the visual patient application definitions, detailed snippets of code generated are not shown here however examples of actual models of patient application screens and the code generated is expressed in the next chapter, describing an example usage of all the components of the care plan modelling system.

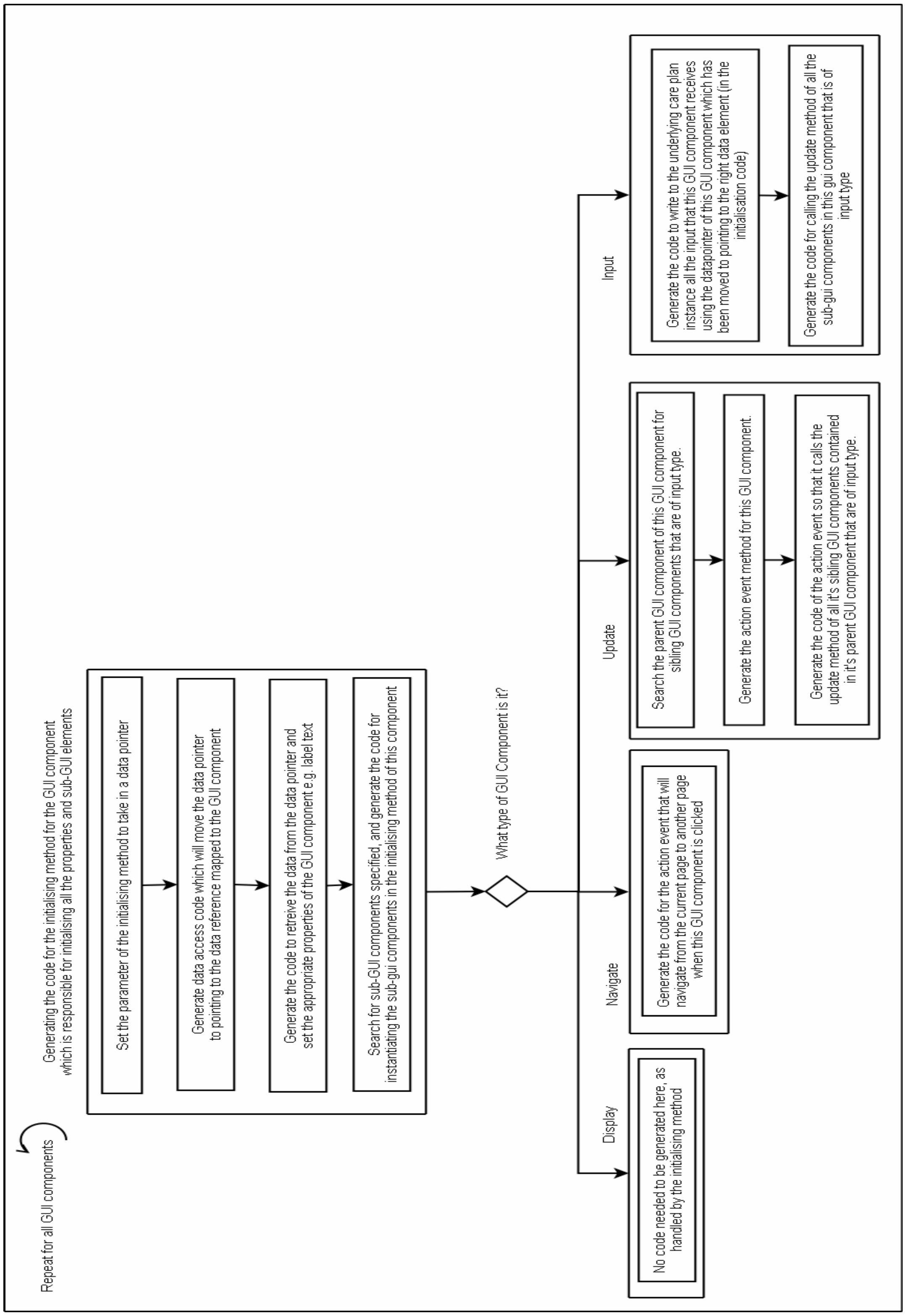

Repeat for all GUI components
Generating the code for the initialising method for the GUI component which is responsible for initialising all the properties and sub-GUI elements
Set the parameter of the initialising method to take in a data pointer
Generate data access code which will move the data pointer to pointing to the data reference mapped to the GUI component
Generate the code to retreive the data from the data pointer and set the appropriate properties of the GUI component e.g. label text
Search for sub-GUI components specified, and generate the code for instantiating the sub-gui components in the initialising method of this component
What type of GUI Component is it?
Display
No code needed to be generated here, as handled by the initialising method
Navigate
Generate the code for the action event that will navigate from the current page to another page when this GUI component is clicked
Update
Search the parent GUI component of this GUI component for sibling GUI components that are of input type.
Generate the action event method for this GUI component.
Generate the code of the action event so that it calls the update method of all it's sibling GUI components contained in it's parent GUI component that are of input type.
Input
Generate the code to write to the underlying care plan instance all the input that this GUI component receives using the datapointer of this GUI component which has been moved to pointing to the right data element (in the initialisation code)
Generate the code for calling the update method of all the sub-gui components in this gui component that is of input type

Figure 71: The general algorithm for the patient application generator.

The general algorithm that was implemented for the OpenLaszlo patient application generator is shown in figure 71 above, this algorithm begins by taking in an XML file which has a model of one or more parts of the patient application built using the VPAM. After taking in this, the generator begins a loop which goes through all the graphical components that are specified in the VPAM model. For each component the patient application generator first generates a reusable class representing the GUI component, and in this class it adds a method for initialising the GUI component (a sort of a constructor for the OpenLaszlo class), this initialising method takes in as a parameter a data pointer. The patient application generator then proceeds to generate the code for this initialising method by firstly ensuring the data pointer in the method parameter is pointing to the correct data value, this is done by checking the VPAM model to determine which data tag the GUI component is mapped to and then generating the code to move the data pointer until this data tag is found. After this the patient application generator generates the code to retrieve the data that is pointed to by the data pointer, and checks whether any properties of the GUI component needs to be set to this data, for example the text property of a display only label, and generates the code to do this. The final bit of code to generate for the initialising method involves the generator searching for the GUI components that are contained in the current GUI component in the VPAM model, and generating the code for instantiating them in the initialising method.

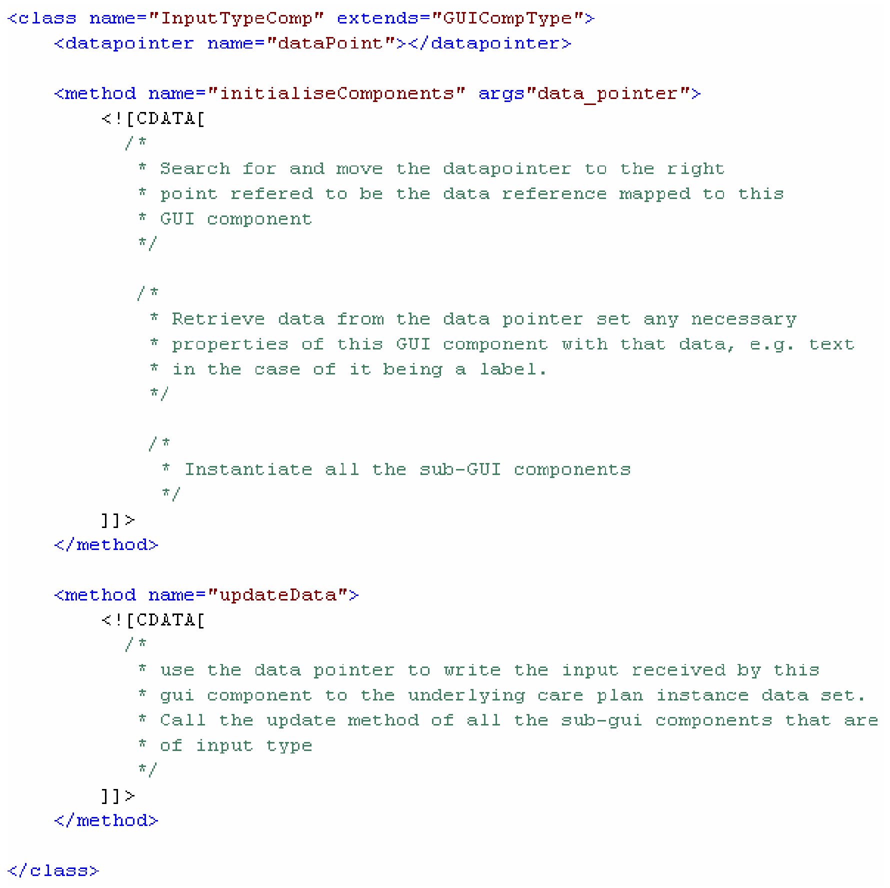
After the generator has generated the code for the initialising method, then next it begins to check the type of the GUI component. The reason for this is so that the code for the basic interactions with the data, such as input, update, navigates, and display can be generated. If the GUI component is of input type, then the generator generates an update method in the GUI component class. The generator generates the code for this method so that the method uses the data pointer (which has been moved to the correct point in the initialising method) to write to the underlying care plan instance all the input that this GUI component directly receives. After this the generator once again checks the sub-gui components contained in this GUI component and if any of these sub-components are of input type, the generator generates the code for calling their own update methods within its own update method.

GUI components that are of update type are typically buttons such as save buttons, or other components that actually take the input from an input type component and update it to the underlying care plan instance. Therefore if the generator finds that a GUI component is of update type, the first thing the generator needs to determine are which input GUI components that it needs to update the input from to the underlying care plan instance. For the purpose of the patient application generator implemented here, an update component will only update the input from input GUI components that are directly contained in its parent view or parent GUI component, i.e. its siblings. Therefore the generator checks which of the current GUI component's siblings are of input type, and then generates an action event method for the current GUI component which calls the update methods of its sibling GUI components that are of input type.

If the GUI component is of navigate type, then generator will read the navigation URL of the current GUI component and then generate the action event method for the current GUI component, generating the code for instantiating and initialising the GUI component (or view essentially) which will be navigated to.

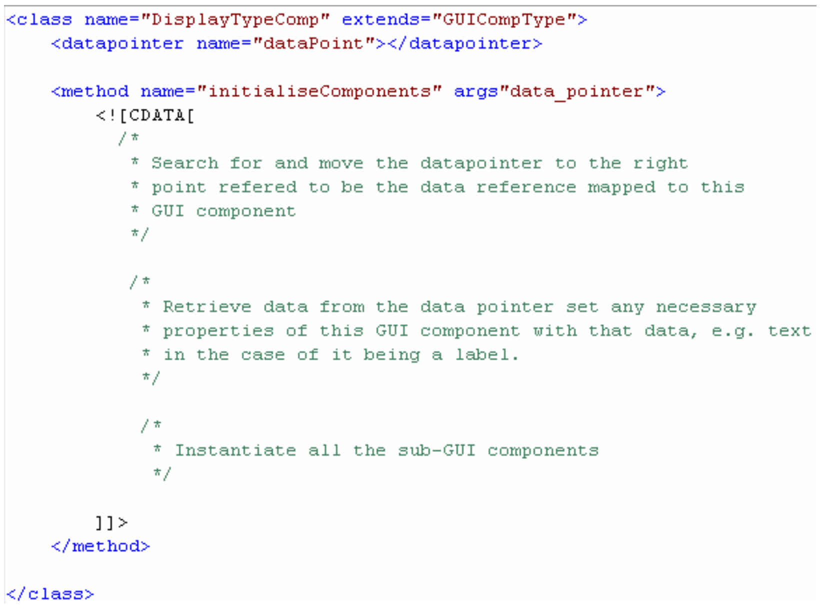
Lastly if the GUI component is simply a display type component, then there is no need for any further code to be generated, as the code needed to display the data pointed to by the data pointer of the GUI component, is already generated as part of the component's initialising method, which retrieves the data and sets the appropriate properties of the component with the retrieved data.

The above algorithm is then repeated until all the GUI components in the VPAM model file have been processed and the code generated for them. The general templates for the code that is generated for each type of GUI components with the method headers is shown below from figures 72 to 75, the specific code in each of the templates is omitted as it is unnecessary for the understanding of the algorithm of the generator, and instead there are comments that explain where this code will go in the generated code and what it will do.



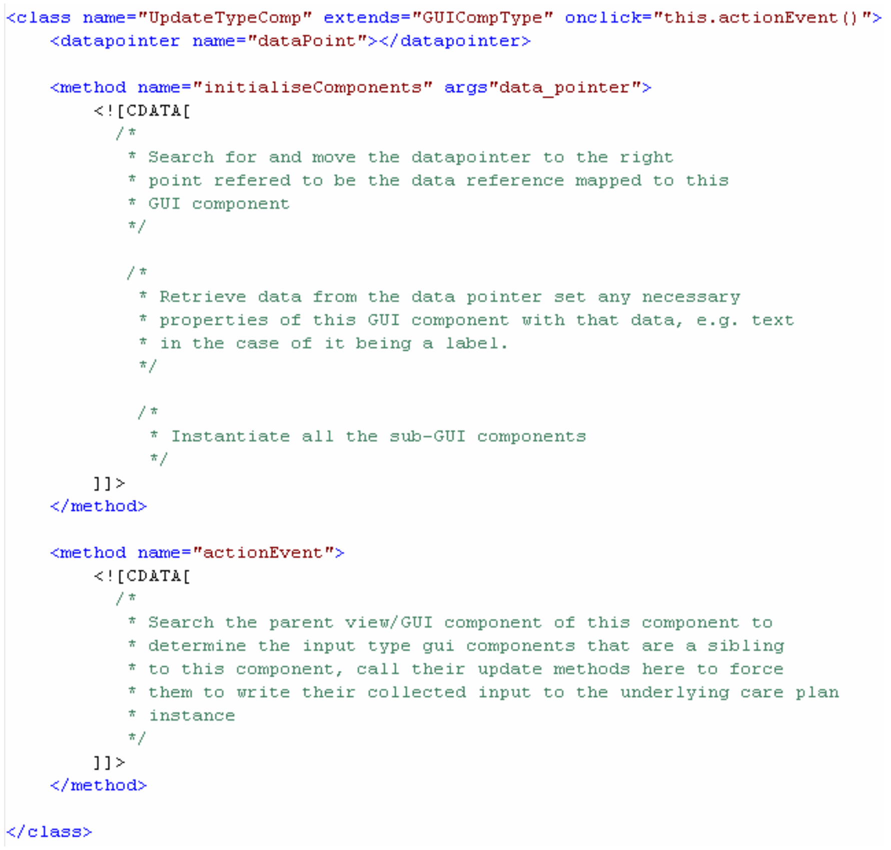
```
<class name="InputTypeComp" extends="GUICompType">
    <datapointer name="dataPoint"></datapointer>

    <method name="initialiseComponents" args"data_pointer">
        <![CDATA[
          /*
           * Search for and move the datapointer to the right
           * point refered to be the data reference mapped to this
           * GUI component
           */

           /*
            * Retrieve data from the data pointer set any necessary
            * properties of this GUI component with that data, e.g. text
            * in the case of it being a label.
            */

            /*
             * Instantiate all the sub-GUI components
             */
        ]]>
    </method>

    <method name="updateData">
        <![CDATA[
          /*
           * use the data pointer to write the input received by this
           * gui component to the underlying care plan instance data set.
           * Call the update method of all the sub-gui components that are
           * of input type
           */
        ]]>
    </method>

</class>
```

Figure 72: Template for the generated code for an input type gui component



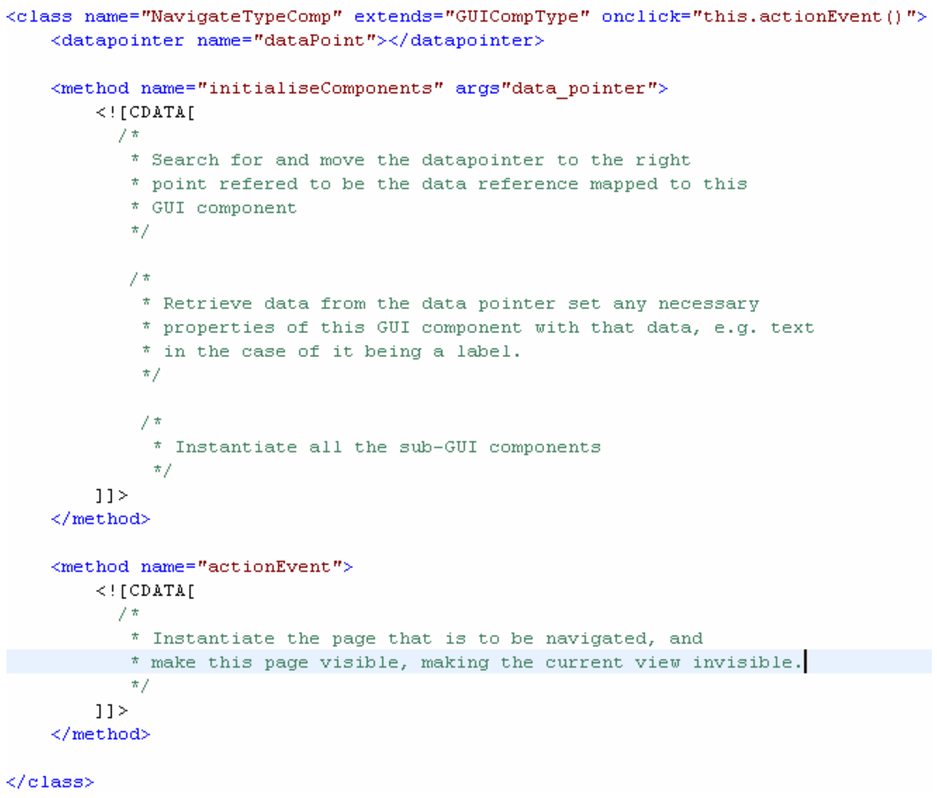
```
<class name="DisplayTypeComp" extends="GUICompType">
    <datapointer name="dataPoint"></datapointer>

    <method name="initialiseComponents" args"data_pointer">
        <![CDATA[
          /*
           * Search for and move the datapointer to the right
           * point refered to be the data reference mapped to this
           * GUI component
           */

           /*
            * Retrieve data from the data pointer set any necessary
            * properties of this GUI component with that data, e.g. text
            * in the case of it being a label.
            */

            /*
             * Instantiate all the sub-GUI components
             */

        ]]>
    </method>

</class>
```

Figure 73: The template for the code generated for the GUI component of display type.



```
<class name="UpdateTypeComp" extends="GUICompType" onclick="this.actionEvent()">
    <datapointer name="dataPoint"></datapointer>

    <method name="initialiseComponents" args"data_pointer">
        <![CDATA[
          /*
           * Search for and move the datapointer to the right
           * point refered to be the data reference mapped to this
           * GUI component
           */

           /*
            * Retrieve data from the data pointer set any necessary
            * properties of this GUI component with that data, e.g. text
            * in the case of it being a label.
            */

            /*
             * Instantiate all the sub-GUI components
             */
        ]]>
    </method>

    <method name="actionEvent">
        <![CDATA[
          /*
           * Search the parent view/GUI component of this component to
           * determine the input type gui components that are a sibling
           * to this component, call their update methods here to force
           * them to write their collected input to the underlying care plan
           * instance
           */
        ]]>
    </method>

</class>
```

Figure 74: The template for the generated code for an update type GUI component.



```
<class name="NavigateTypeComp" extends="GUICompType" onclick="this.actionEvent()">
    <datapointer name="dataPoint"></datapointer>

    <method name="initialiseComponents" args"data_pointer">
        <![CDATA[
          /*
           * Search for and move the datapointer to the right
           * point refered to be the data reference mapped to this
           * GUI component
           */

           /*
            * Retrieve data from the data pointer set any necessary
            * properties of this GUI component with that data, e.g. text
            * in the case of it being a label.
            */

            /*
             * Instantiate all the sub-GUI components
             */
        ]]>
    </method>

    <method name="actionEvent">
        <![CDATA[
          /*
           * Instantiate the page that is to be navigated, and
           * make this page visible, making the current view invisible.
           */
        ]]>
    </method>

</class>
```

Figure 75: The template for the code generated for a navigate type GUI component.

8.6 Summary

The implementation of the care plan modelling language can be split up into four main parts, the first being the implementation of the visual care plan modelling language, the second the implementation of the care plan instantiation tool, the third being the implementation of the visual patient application modeller, and finally the implementation of the patient application generator. In addition to this the patient application was also implemented in OpenLaszlo in order to provide a reference application for the implementation of the patient application generator.

The implementation of the visual care plan modelling language and the visual patient application modeller were done in the Marama visual language development suite. The development process of building both visual languages consisted of firstly specifying the meta-model of the language using the Marama-meta-model definer inbuilt in the environment, then specifying the graphical notation of the language by specifying the shapes that will represent each meta-model element, and finally mapping the shapes of the language with the actual meta-model elements.

The care plan instantiation tool was implemented as a desktop application, in order to allow health care professionals to fine tailor the health care plans to patients before deploying it to them. The domain code for the application can be split up into two parts, firstly the class libraries for the care plan object model, which encompassed all the functionality for representing care plans, and instantiating them. The second part of the domain code represented the Marama diagram transformer to convert the visual language diagrams for the care plan model into objects to load into the tool GUI. Similarly the GUI of the tool was divided into two parts, between the user controls to allow editing of the care plan components and the actual tool GUI built to allow the visualisation of the care plan.

In order to learn the OpenLaszlo language for the implementation of the patient application generator, a sample reference patient application was built in OpenLaszlo representing most of the functionality of the patient application. Before OpenLaszlo was decided upon a technology evaluation was conducted for alternative technologies to use to implement the patient application, among technologies considered, included MUPE from Nokia, the .Net compact framework from Microsoft, XForms, and OpenLaszlo. Out of all these languages it was decided that OpenLaszlo would be the best choice because programs written in it are easily compilable to flash, which runs on most mobile and electronic devices these days.

The final part of the implementation included the implementation of the patient application generator so that it would consume the VPAM models and produce from them the patient application in OpenLaszlo. The general algorithm designed and implemented for the generator involved a mechanism through which the generator can read and generate code for the graphical user interface, data access, and basic interactions of the patient application screens based on the VPAM definition of the screens.

Chapter 9 - Case study with Prototype

9.1 Introduction

The purpose of this chapter is to show a sample use of the care plan modelling system prototype, with an example patient and health care professional scenario discussed in chapter 5. Firstly a brief recap of the obesity management scenario in chapter 5 is discussed here, following which the chapter describes how a care plan for John in the scenario can be designed using the visual care plan modelling language, and then how this care plan can be personalised to John using the care plan instantiation tool. Finally examples of how the patient application to host a care plan instance for John on a mobile device can be designed using the visual patient application modeller are discussed.

9.2 Example Scenario of Prototype

In order to gain a better understanding of the care plan modelling system as a whole it is important to consider an example scenario where the care plan modelling system can be employed from start to end to design and deploy a care plan to a patient suffering from a long term ailment. For the purpose of the example shown here the scenario will concern a patient John Doe suffering from obesity, same as the scenario of chapter 5.

9.2.1 Scenario Preamble

The scenario that will be used for exhibiting the functionality of the care plan modelling system is the same as that which was used in chapter 5, a brief preamble of it is as follows:

“John Doe, a 35 year old software engineer, working for a reputable software engineering firm, he has been married for 8 years and has 2 children one 6 years old, and another 7. John suffers from class 1 obesity; this not only affects his self esteem, but also affects his ability to take part in family activities, with his wife and children. In addition to this John, has a history of diabetes in his family, and he is increasingly worried that his obesity and associated unhealthy eating habits may lead him to a more serious health crisis. As a result of this John decides to see his GP in order to improve and take control of his health.”

The following sections show how the care plan modelling system can be used by a health care professional to design, instantiate, and deploy the obesity management care plan to John. For more details on the scenario used for the example presented here see chapter 5.

9.2.2 Designing a Care Plan for John

The GP starts of in the VCPML editor, to create the basic care plan structure for John’s obesity management care plan, the diagram that the GP builds can be seen below in figure 76, it can be seen that the obesity management care plan is the top-most level care plan and that it has five sub-care plans, i.e. the dietary therapy, pharmacotherapy, physical therapy, diabetes monitoring and behavioural therapy care plan.

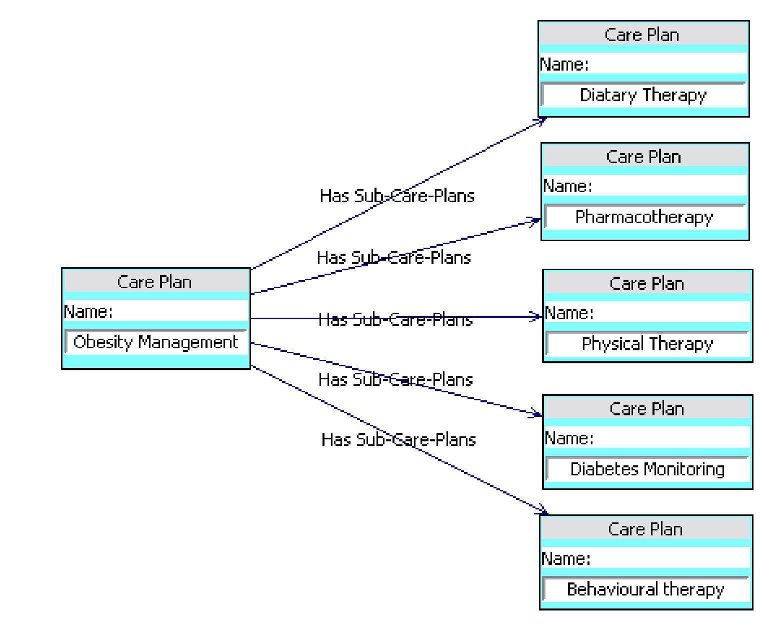


Figure 76: Modelling the obesity management care plan with its sub-care plans in the VCPML.

Once the GP has defined the sub care plans that are part of the obesity management care plan, next the GP can grow the care plan with more information, the GP can now model the performance indicators or performance metrics for John, so that he can judge his progress, in figure 77 below it can be seen that he GP has modelled two performance metrics for John, the first being the body mass index (BMI), of 30, i.e. John, needs to reduce his weight so that his body mass index falls down to 25, the second performance metric is regarding John's waist circumference, in this case it is recommended that John gets his waist size down to at least 90 cm.

After adding in the performance metrics, the GP decides to schedule in a review for John's obesity, and schedules this review on a 3 monthly basis, the GP then also specifies that while he conducts this review activity with the patient, he will also need a consulting room, and the help of a nurse. Figure 77, shows how this review activity, its routine, and its resource needs have been modelled. Furthermore the GP can also associate an assessment module to the care plan, which in this case is an obesity assessment treatment algorithm, which the GP possibly uses during the review sessions to assess the patient's performance; the modelling of this obesity treatment algorithm under the assessment module is discussed later on in this section.

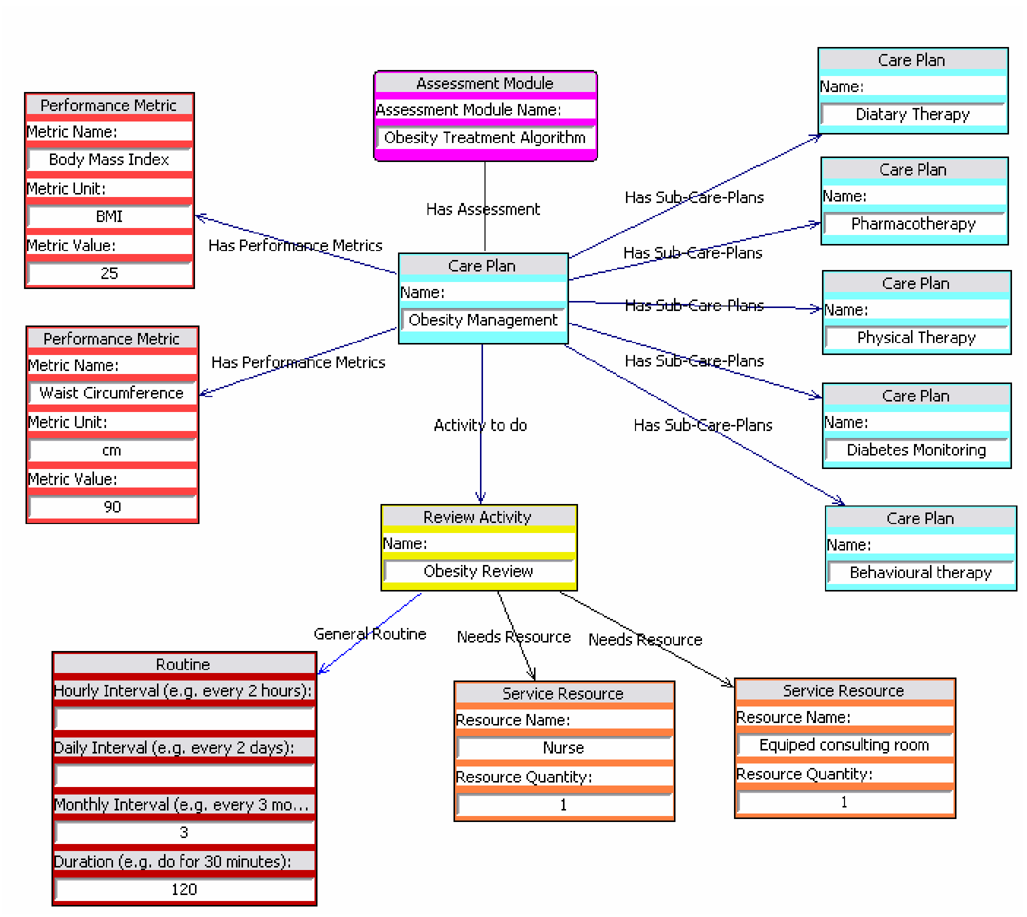


Figure 77: A more detailed model of the care plan, sub care plans, performance metrics, assessment modules, and review activities.

Once the GP has defined the performance metrics and the review sessions for the care plan, it is now time to extend the sub-care plans in order to add health care activities to help John achieve his performance goals. It is important to realise that the actual obesity treatment guideline has multiple activities under each of the care plans shown above, however for the purpose of the example shown here only two of the care plans are extended, i.e. the diabetes monitoring care plan and the physical therapy care plan.

Figure 78 shows a glucose measurement activity that is designed for John, it can be seen that the activity has a routine to conduct every 2 days, and also has instructions on how the activity needs to be conducted. Furthermore it can also be seen from figure 78 that in order to do the activity John will need certain material resources, such as a testing pen, testing strips and a testing meter. It can also be seen that John is required to record his blood glucose sugar once it is measured in the mmol/L data unit, also specified as part of the data collection activity.

In addition to modelling data collection activities, and review activities, a GP can also model ordinary tasks for the patient to conduct, such as taking of medicine, or doing some exercise, figure 79 below shows the model of two exercise tasks, namely jogging and swimming, each with it's own set of instructions, and both sharing the general routine of conducting the activity, every two days for 20 minutes these activities are grouped under the physical therapy care plan.

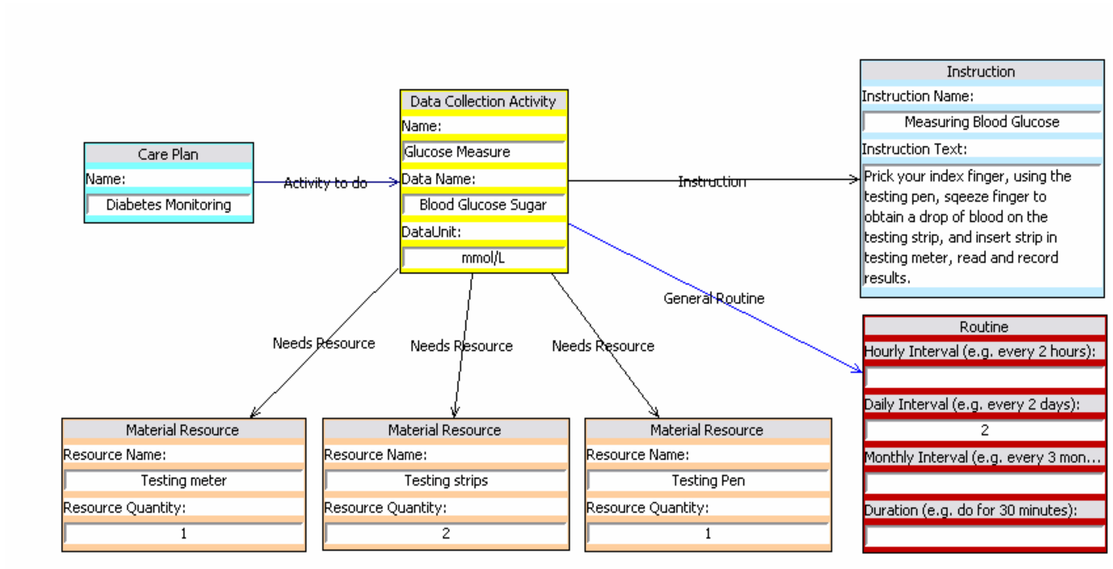


Figure 78: Diagram showing the model of a data collection activity under the diabetes monitoring care plan for the collection of blood glucose sugar.

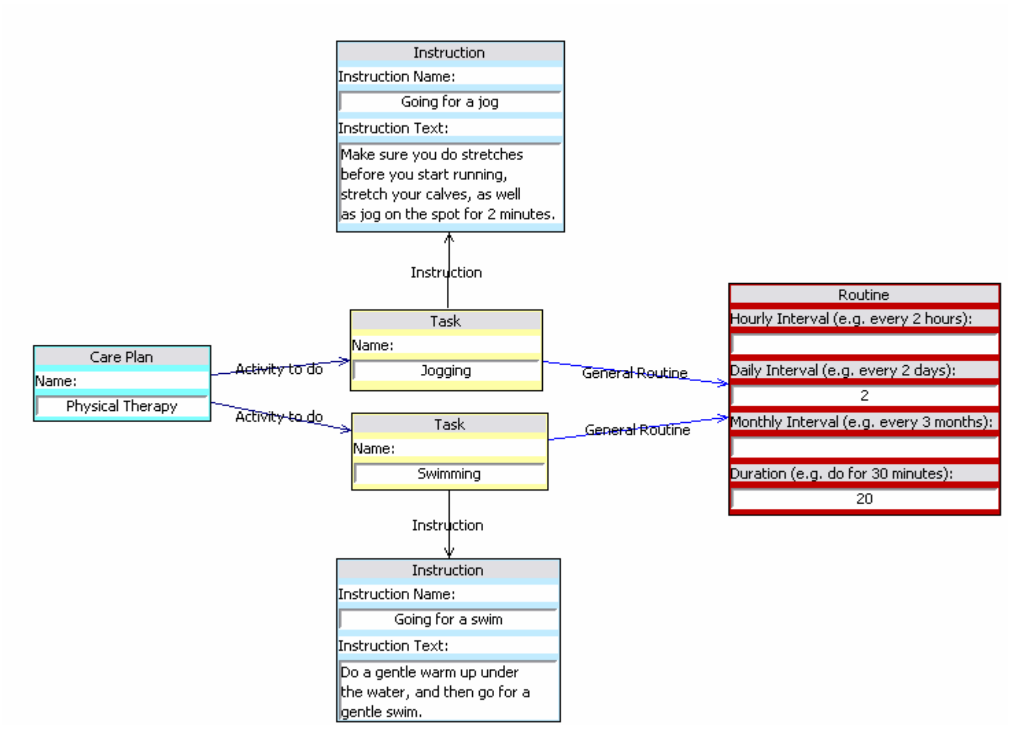

Instruction
Instruction Name:
Going for a jog
Instruction Text:
Make sure you do stretches before you start running, stretch your calves, as well as jog on the spot for 2 minutes.
Instruction
Care Plan
Name:
Physical Therapy
Activity to do
Activity to do
Task
Name:
Jogging
Task
Name:
Swimming
General Routine
General Routine
Routine
Hourly Interval (e.g. every 2 hours):
Daily Interval (e.g. every 2 days):
2
Monthly Interval (e.g. every 3 months):
Duration (e.g. do for 30 minutes):
20
Instruction
Instruction
Instruction Name:
Going for a swim
Instruction Text:
Do a gentle warm up under the water, and then go for a gentle swim.

Figure 79: The model of two exercising tasks under the physical therapy care plan.

In a similar fashion the to the above activity models, the GP can model all the activities under the remaining care plans, specifying for each activity, instructions on conducting the activity, resources needed to do the activity, and general routines on when to do the activity.

In figure 77 above it can be seen that the GP has specified that in order to assess patients on the obesity care plan the obesity treatment algorithm must be used. Using the VCPML editor, the GP can model this treatment algorithm as shown in figure 80 below.

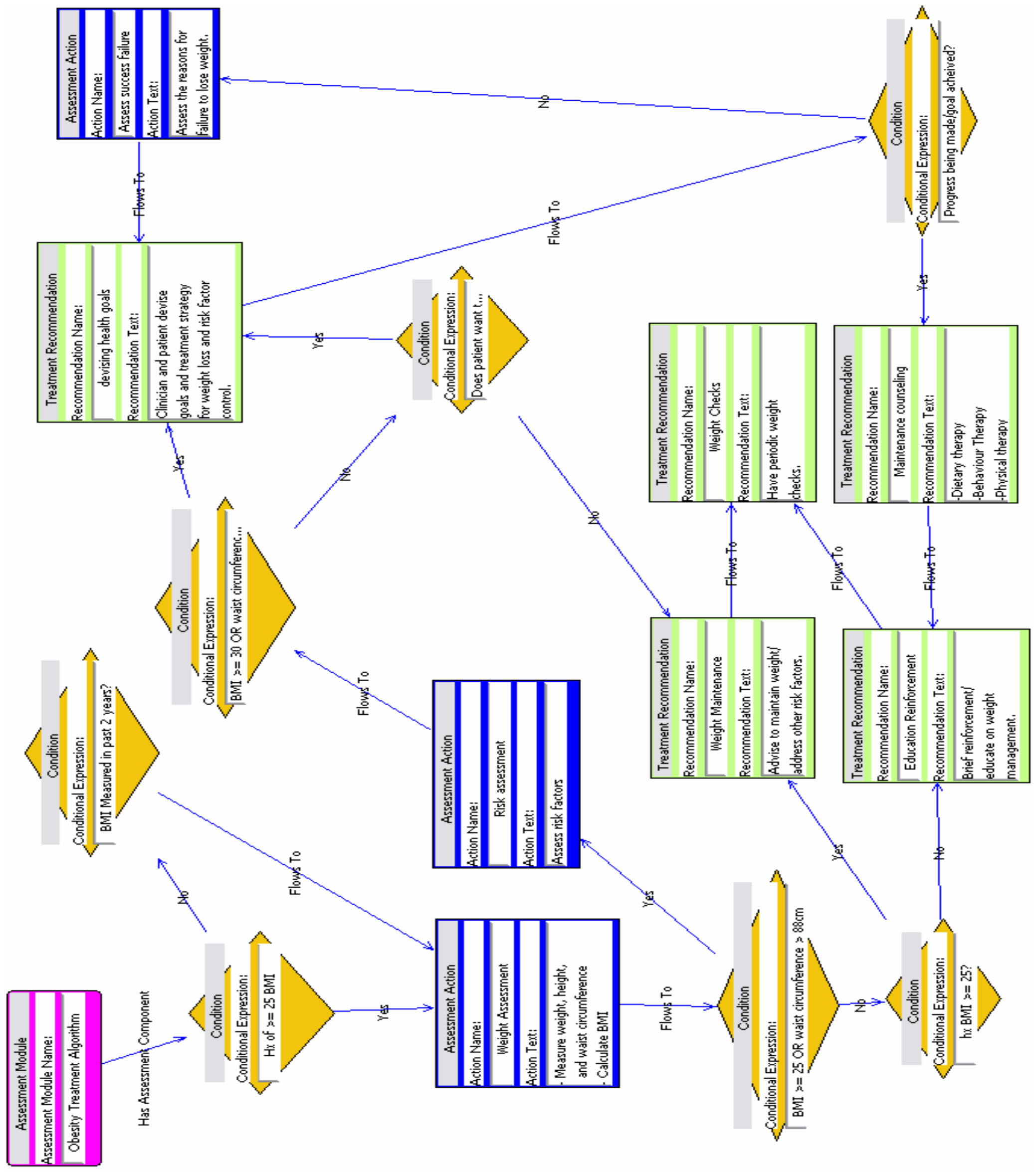


Figure 80: The treatment algorithm for obesity management modelled using the VCPML originally taken from the treatment algorithm in the obesity treatment guideline, also shown in chapter 5.

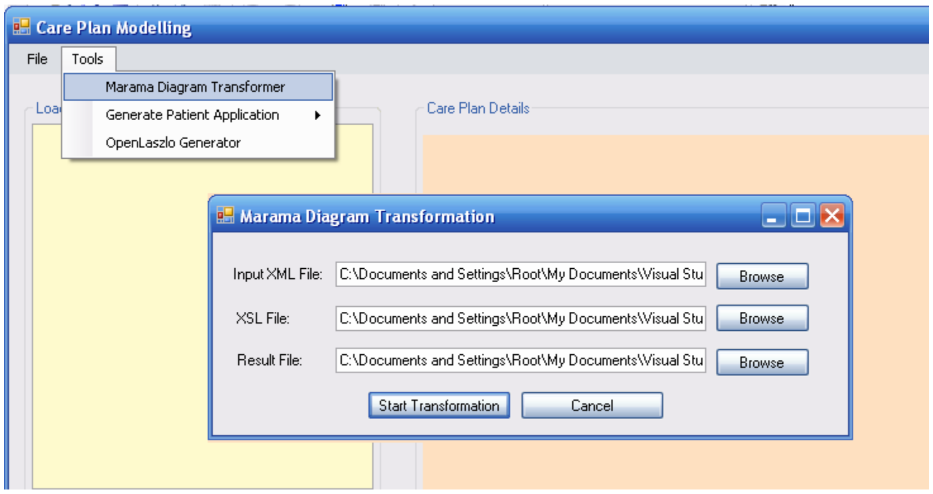
The model of this assessment module can then be used by the GP in order to decide upon the assessment actions, and treatment recommendations that need to be adhered to while assessing the patient's condition as well as the patient's progress and performance. It is important to realize that in contrast with the other parts of the care plan that are deployed onto the patient application for the patient to view, the assessment module is solely for the GP to view, as an aid for assessing the patient. It is also important to notice that the assessment module coded in figure 80 above is exactly coded from the treatment algorithm for obesity management shown in diagrammatic form in chapter 5.

Once the GP has designed the entire obesity management care plan as explained above, adding in the performance metrics, sub-care plans, activities, and assessment modules for the patient, then the next step is to instantiate and deploy the care plan to the patient on their patient application.

9.2.3 Instantiating the care plan for John

When a patient comes in to the health care professional for a consultation, the health care professional can either choose from a database of already existing care plans, or model a new one from scratch, regardless there may always be some aspects of the care plan template that needs to be changed or fine tailored to the patient before the care plan can be deployed to the patient on their patient application, at the very least the health care professional needs to specify a detailed timetable for each of the activities that the care plan is composed of so that appointment reminders can be generated for the patient.

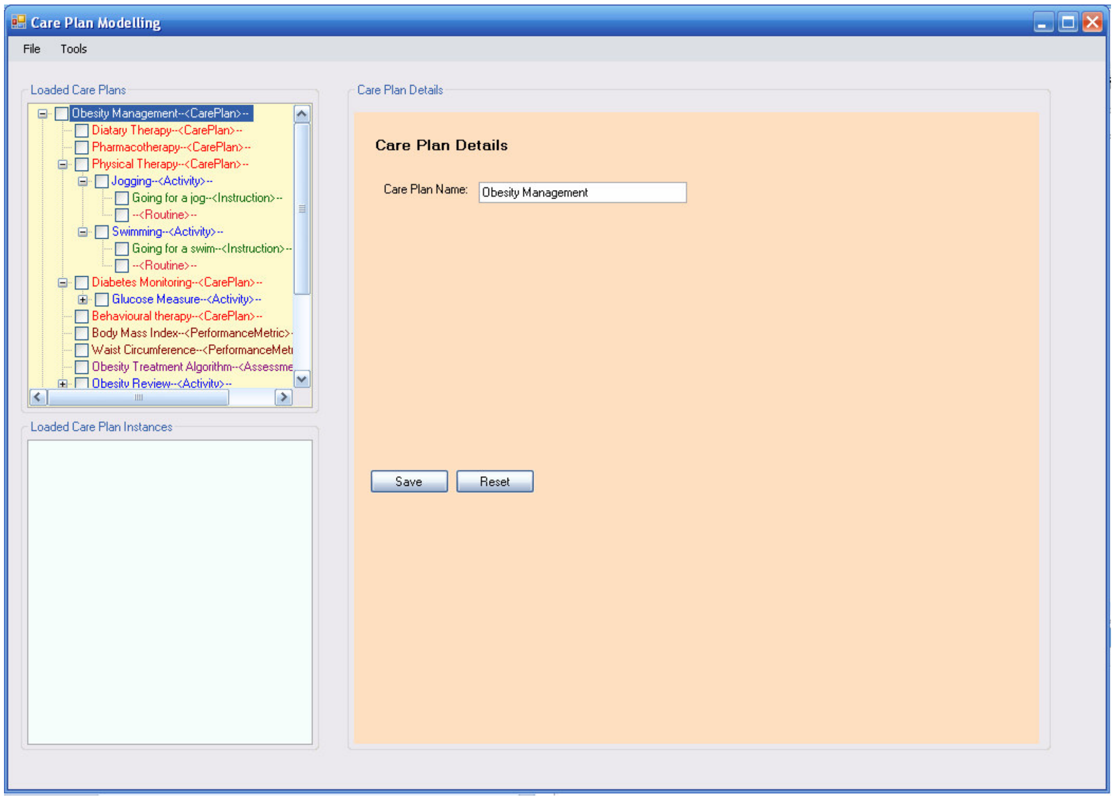
Once the care plan has been modelled or retrieved from a database of care plans, it is a raw XML file, before the care plan can be loaded into the care plan instantiation tool it needs to be transformed using the XSL style sheet described in chapter 7. The health care professional can do this by choosing the right XSL style sheet and input XML care plan and then transforming the care plan document into a more readable XML sheet (as shown in figure 81 below).



Input XML File: C:\Documents and Settings\Root\My Documents\Visual Stu
XSL File: C:\Documents and Settings\Root\My Documents\Visual Stu
Result File: C:\Documents and Settings\Root\My Documents\Visual Stu

Figure 81: Transforming the raw care plan model into a readable and loadable XML document.

Once the care plan template model is transformed it can then be loaded into the care plan instantiation tool as shown in figure 82 below, through this interface the health care professional can view the entire care plan model in a tree like structure as shown in figure 82 below in the yellow panel. In addition to this the health care professional can select any one part of the care plan, such as sub care plans, activities, activity instructions etc, and edit its properties.



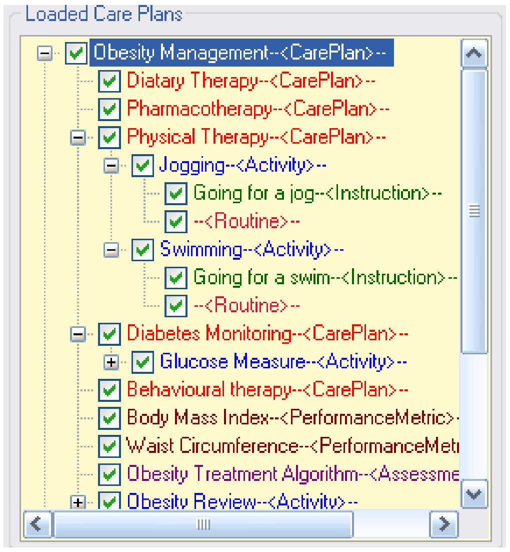
```
[-] [ ] Obesity Management--<CarePlan>--
      [ ] Diatary Therapy--<CarePlan>--
      [ ] Pharmacotherapy--<CarePlan>--
  [-] [ ] Physical Therapy--<CarePlan>--
      [-] [ ] Jogging--<Activity>--
            [ ] Going for a jog--<Instruction>--
            [ ] --<Routine>--
      [-] [ ] Swimming--<Activity>--
            [ ] Going for a swim--<Instruction>--
            [ ] --<Routine>--
  [-] [ ] Diabetes Monitoring--<CarePlan>--
      [+] [ ] Glucose Measure--<Activity>--
      [ ] Behavioural therapy--<CarePlan>--
      [ ] Body Mass Index--<PerformanceMetric>-
      [ ] Waist Circumference--<PerformanceMetı
      [ ] Obesity Treatment Algorithm--<Assessme
  [+] [ ] Obesity Review--<Activity>--
```

Care Plan Details

Care Plan Name: Obesity Management

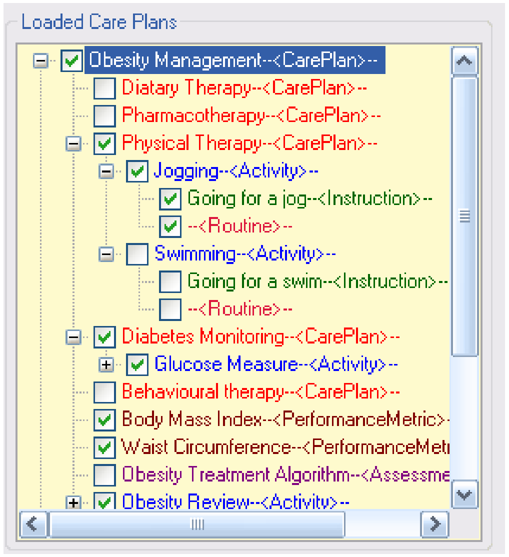
Figure 82: A care plan template loaded into the care plan instantiation tool ready for the instantiation process.

Once the care plan has been loaded into the care plan instantiation tool, the health care professional can now start trimming and changing the care plan to fine tailor it to the patient, note that this can also be done in the visual language itself, however this is provided as an alternative. It can be seen from figure 83 below that the health care professional can instantiate the entire care plan with no changes, by simply ticking the checkbox for the top most care plan, or alternatively as shown in figure 84, the health care professional can deselect entire care plans, or parts of care plans such as activities that are irrelevant for the current patient in their care plan instance. In the case of instantiating a care plan for John, we know that the original care plan model has no activities for the dietary therapy, pharmacotherapy, and behavioural therapy care plans, therefore the GP deselects these, in addition to this the GP also deselects the swimming activity in the physical therapy care plan because John can't swim, finally the GP also deselects the assessment module from the care plan template, as this is only for the health care professional to view.



```
Loaded Care Plans
⊟ ☑ Obesity Management--<CarePlan>--
    ☑ Diatary Therapy--<CarePlan>--
    ☑ Pharmacotherapy--<CarePlan>--
  ⊟ ☑ Physical Therapy--<CarePlan>--
    ⊟ ☑ Jogging--<Activity>--
        ☑ Going for a jog--<Instruction>--
        ☑ --<Routine>--
    ⊟ ☑ Swimming--<Activity>--
        ☑ Going for a swim--<Instruction>--
        ☑ --<Routine>--
  ⊟ ☑ Diabetes Monitoring--<CarePlan>--
    ⊞ ☑ Glucose Measure--<Activity>--
    ☑ Behavioural therapy--<CarePlan>--
    ☑ Body Mass Index--<PerformanceMetric>-
    ☑ Waist Circumference--<PerformanceMetr
    ☑ Obesity Treatment Algorithm--<Assessme
  ⊞ ☑ Obesity Review--<Activity>--
```

Figure 83: Selecting all parts of the care plan to instantiate.

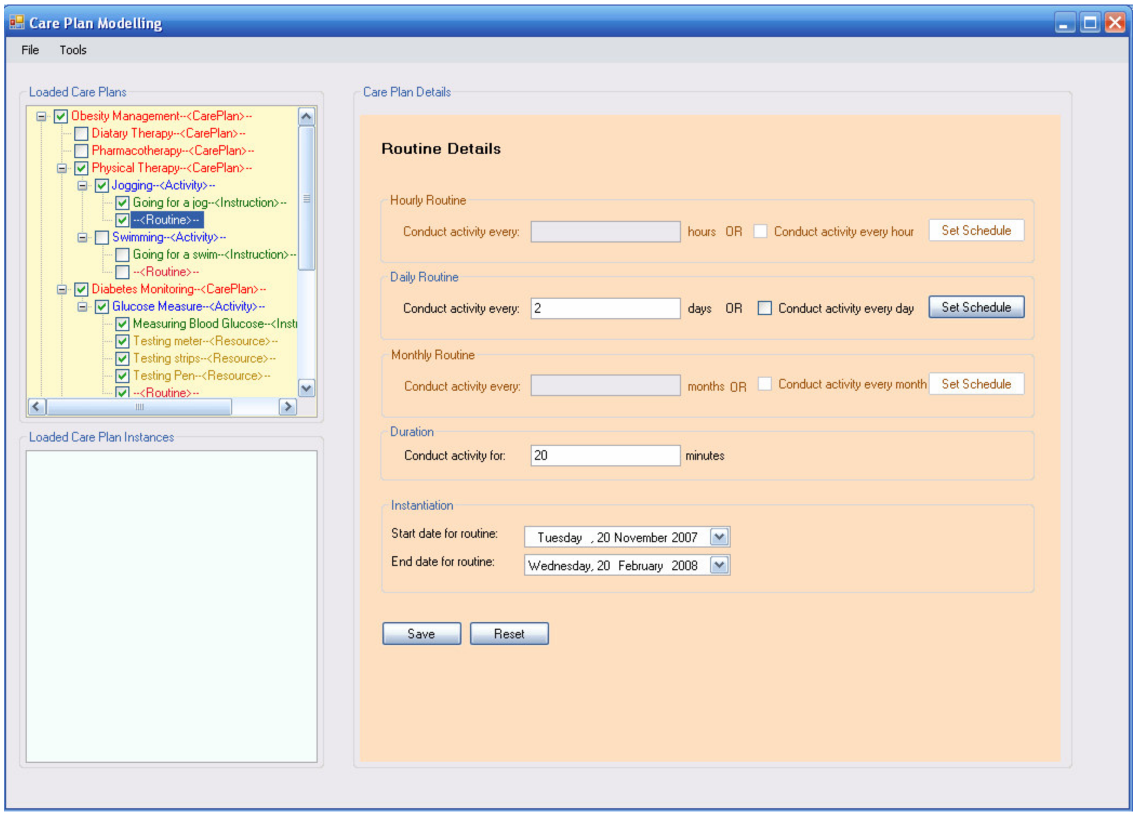


```
Loaded Care Plans
⊟ ☑ Obesity Management--<CarePlan>--
    ☐ Diatary Therapy--<CarePlan>--
    ☐ Pharmacotherapy--<CarePlan>--
  ⊟ ☑ Physical Therapy--<CarePlan>--
    ⊟ ☑ Jogging--<Activity>--
        ☑ Going for a jog--<Instruction>--
        ☑ --<Routine>--
    ⊟ ☐ Swimming--<Activity>--
        ☐ Going for a swim--<Instruction>--
        ☐ --<Routine>--
  ⊟ ☑ Diabetes Monitoring--<CarePlan>--
    ⊞ ☑ Glucose Measure--<Activity>--
    ☐ Behavioural therapy--<CarePlan>--
    ☑ Body Mass Index--<PerformanceMetric>-
    ☑ Waist Circumference--<PerformanceMetr
    ☐ Obesity Treatment Algorithm--<Assessme
  ⊞ ☑ Obesity Review--<Activity>--
```

Figure 84: deselecting the parts of the care plan that are irrelevant for the patient in their care plan instance.

After all the trimming is done John's obesity management care plan has two sub care plans, the first being the physical therapy care plan with the jogging activity, and the next being the diabetes monitoring care plan with the glucose measurement activity. In addition to this the main obesity management care plan also has the obesity review activity.

The next step for the health care professional is to take the three activities (jogging, glucose measurement, and obesity review), and specify detailed timetables for them according to the general routines that have been modelled for them through the visual care plan modelling language. This can be simply done by clicking on the routines for the activities under the activity node in the yellow panel. Figure 85 shows the general routine that was modelled for the jogging activity. The jogging activity is a daily routine, and for this the health care professional can firstly set an activation start and end date, i.e. when John should start doing the activity from, and when he should no longer do the activity. After this the health care professional can specify a detailed schedule by pressing the set schedule button to set the times at which the activity needs to be done as shown in figures 86 and 87. The routine for the glucose management activity is also a daily routine and the detailed schedule for it can be set in a similar manner as shown in figure 88. The obesity review activity in contrast to the other two activities has a monthly routine, as shown in figure 89, with the activity being done every three months, and to set the detailed schedule for this activity the health care professional selects a preferred time as well as a preferred day that the activity will be done 3 months from now as shown in figure 90.



Obesity Management--<CarePlan>--
Diatary Therapy--<CarePlan>--
Pharmacotherapy--<CarePlan>--
Physical Therapy--<CarePlan>--
Jogging--<Activity>--
Going for a jog--<Instruction>--
--<Routine>--
Swimming--<Activity>--
Going for a swim--<Instruction>--
--<Routine>--
Diabetes Monitoring--<CarePlan>--
Glucose Measure--<Activity>--
Measuring Blood Glucose--<Instr
Testing meter--<Resource>--
Testing strips--<Resource>--
Testing Pen--<Resource>--
--<Routine>--

Routine Details

Hourly Routine
Conduct activity every: hours OR ☐ Conduct activity every hour Set Schedule

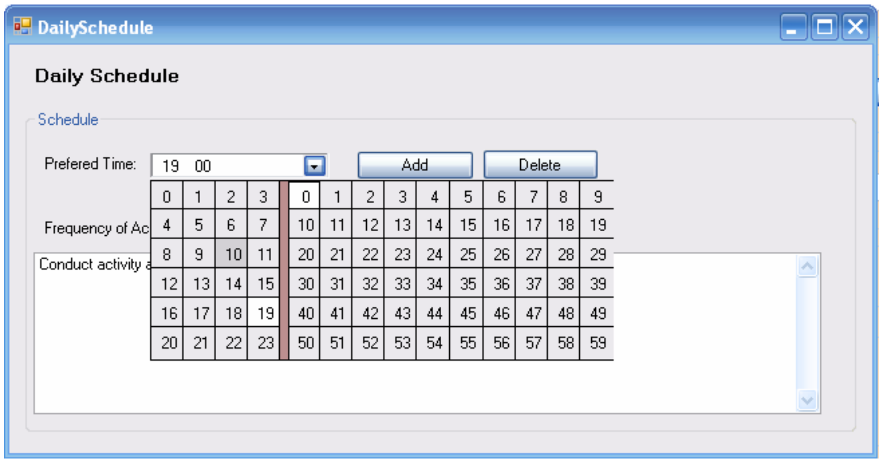
Daily Routine
Conduct activity every: 2 days OR ☐ Conduct activity every day Set Schedule

Monthly Routine
Conduct activity every: months OR ☐ Conduct activity every month Set Schedule

Duration
Conduct activity for: 20 minutes

Instantiation
Start date for routine: Tuesday , 20 November 2007
End date for routine: Wednesday, 20 February 2008

Figure 85: Setting the timetable for the jogging activity.

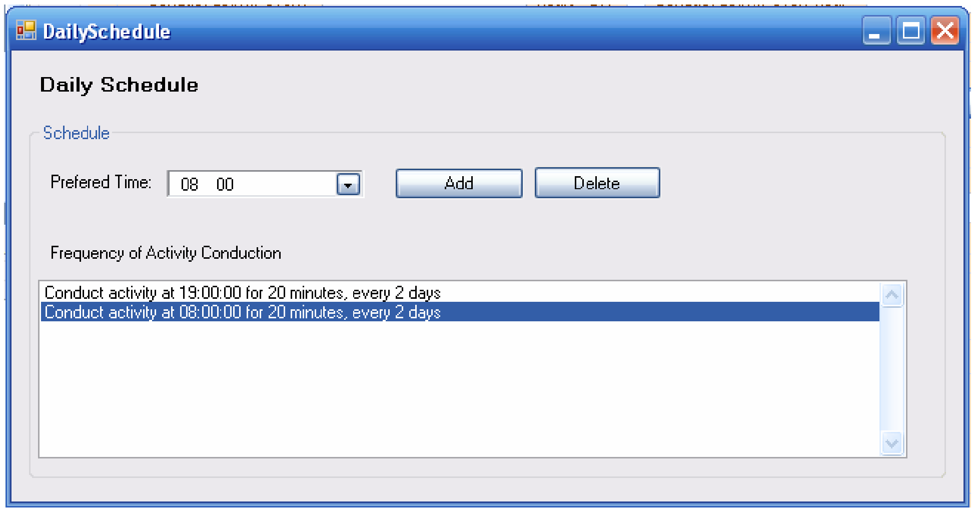


Daily Schedule

Schedule
Prefered Time: 19 00
Frequency of Ac
Conduct activity a

0	1	2	3	0	1	2	3	4	5	6	7	8	9
4	5	6	7	10	11	12	13	14	15	16	17	18	19
8	9	10	11	20	21	22	23	24	25	26	27	28	29
12	13	14	15	30	31	32	33	34	35	36	37	38	39
16	17	18	19	40	41	42	43	44	45	46	47	48	49
20	21	22	23	50	51	52	53	54	55	56	57	58	59

Figure 86: Setting the detailed schedule for the daily routine of the jogging activity.

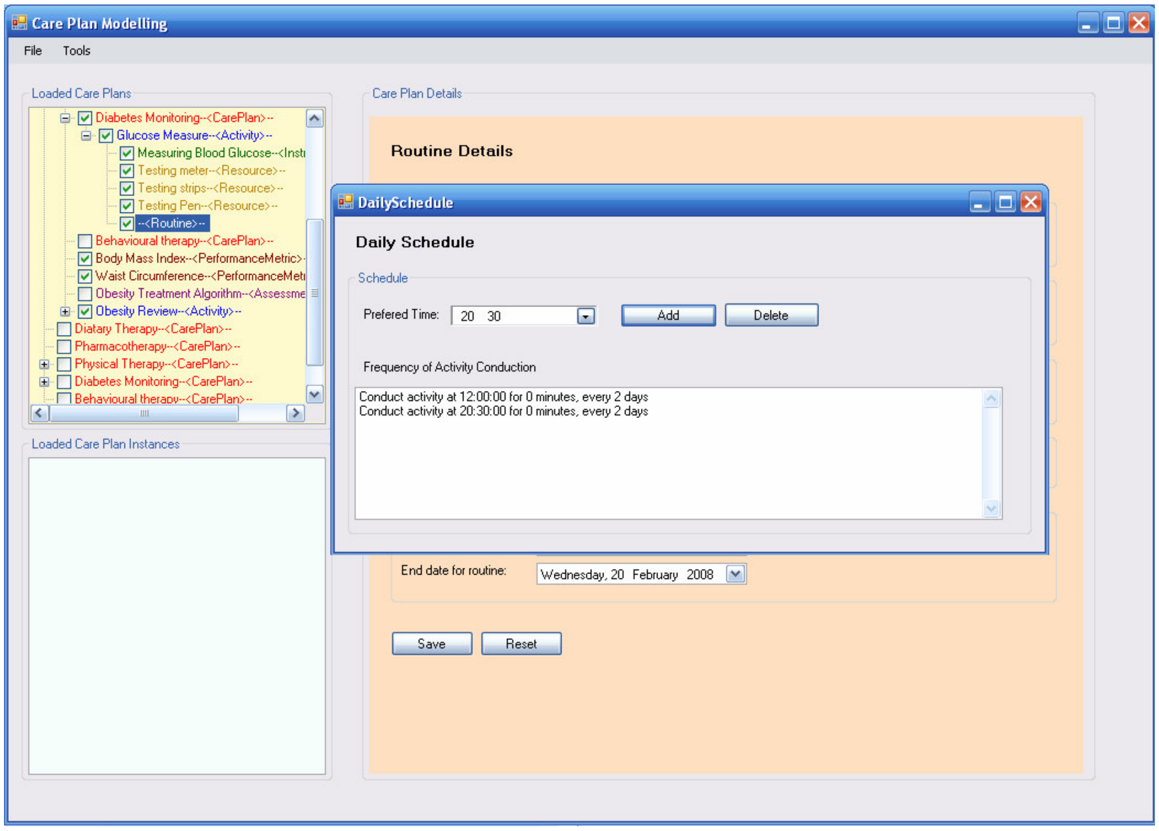


Prefered Time: 08 00

Frequency of Activity Conduction

Conduct activity at 19:00:00 for 20 minutes, every 2 days
Conduct activity at 08:00:00 for 20 minutes, every 2 days

Figure 87: Detailed schedule of the jogging activity.



Diabetes Monitoring--<CarePlan>--
Glucose Measure--<Activity>--
Measuring Blood Glucose--<Insti
Testing meter--<Resource>--
Testing strips--<Resource>--
Testing Pen--<Resource>--
--<Routine>--
Behavioural therapy--<CarePlan>--
Body Mass Index--<PerformanceMetric>-
Waist Circumference--<PerformanceMetı
Obesity Treatment Algorithm--<Assessme
Obesity Review--<Activity>--
Diatary Therapy--<CarePlan>--
Pharmacotherapy--<CarePlan>--
Physical Therapy--<CarePlan>--
Diabetes Monitoring--<CarePlan>--
Behavioural therapy--<CarePlan>--

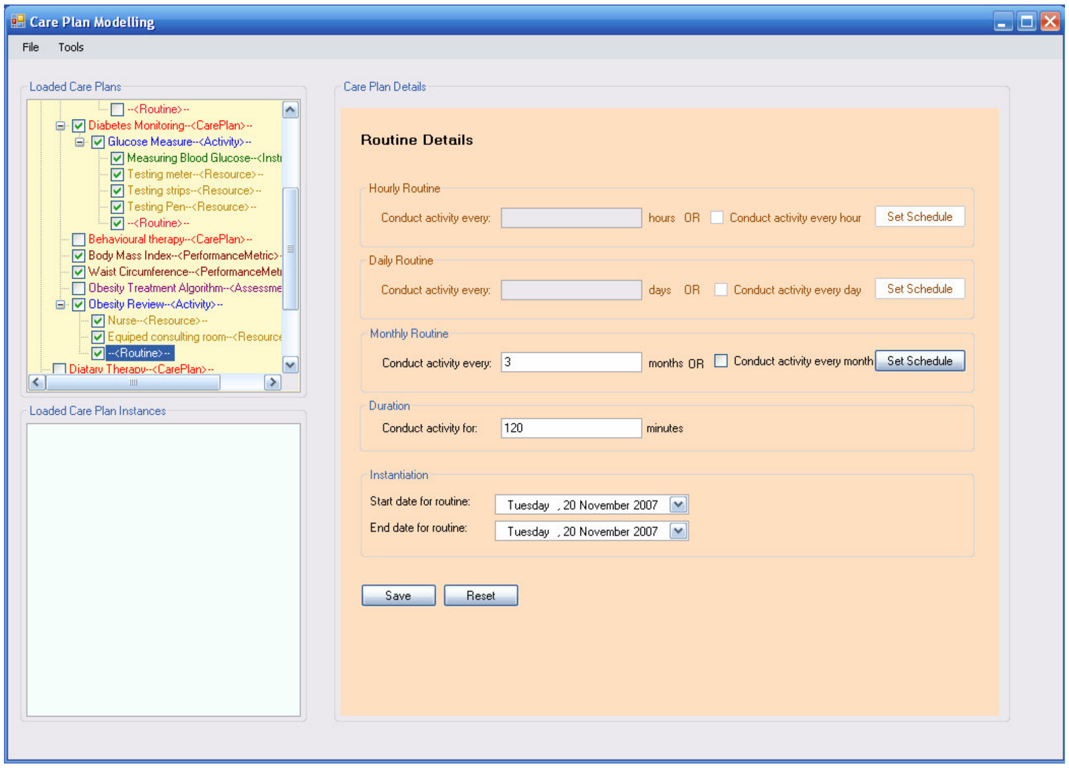
Prefered Time: 20 30

Frequency of Activity Conduction

Conduct activity at 12:00:00 for 0 minutes, every 2 days
Conduct activity at 20:30:00 for 0 minutes, every 2 days

End date for routine: Wednesday, 20 February 2008

Figure 88: The screen showing the detailed schedule for the glucose measurement activity.


Care Plan Modelling

File Tools

Loaded Care Plans

- --<Routine>--
- ⊟ ✓ Diabetes Monitoring--<CarePlan>--
  - ⊟ ✓ Glucose Measure--<Activity>--
    - ✓ Measuring Blood Glucose--<Instr
    - ✓ Testing meter--<Resource>--
    - ✓ Testing strips--<Resource>--
    - ✓ Testing Pen--<Resource>--
    - ✓ --<Routine>--
- Behavioural therapy--<CarePlan>--
- ✓ Body Mass Index--<PerformanceMetric>-
- ✓ Waist Circumference--<PerformanceMetr
- Obesity Treatment Algorithm--<Assessme
- ⊟ ✓ Obesity Review--<Activity>--
  - ✓ Nurse--<Resource>--
  - ✓ Equiped consulting room--<Resource
  - ✓ --<Routine>--
- Diatary Therapy--<CarePlan>--

Loaded Care Plan Instances

Care Plan Details

Routine Details

Hourly Routine
Conduct activity every: [ ] hours OR ☐ Conduct activity every hour [Set Schedule]

Daily Routine
Conduct activity every: [ ] days OR ☐ Conduct activity every day [Set Schedule]

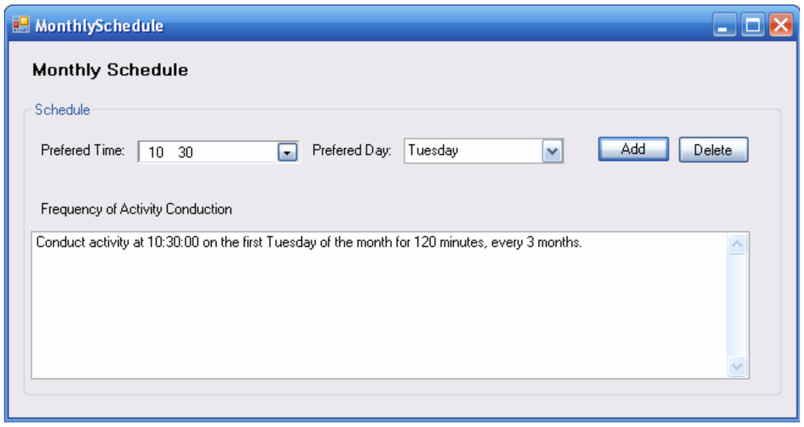
Monthly Routine
Conduct activity every: 3 months OR ☐ Conduct activity every month [Set Schedule]

Duration
Conduct activity for: 120 minutes

Instantiation
Start date for routine: Tuesday , 20 November 2007
End date for routine: Tuesday , 20 November 2007

[Save] [Reset]

Figure 89: The screen showing the overall routine for the obesity review activity.


MonthlySchedule

Monthly Schedule

Schedule
Prefered Time: 10 30 Prefered Day: Tuesday [Add] [Delete]

Frequency of Activity Conduction

Conduct activity at 10:30:00 on the first Tuesday of the month for 120 minutes, every 3 months.

Figure 90: The dialog for setting the detailed schedule for the monthly activity, which in this case is the obesity review activity.

Once detailed schedules have been set for the general routines for activities, then the care plan can be instantiated as shown below, in figure 91 below. It can be noted from the figure below that the instantiated version of the care plan has only the parts of the care plan that the health care professional had specifically selected for John, therefore there is no dietary therapy, pharmacotherapy, or behavioural therapy as part of John's care plan instance, neither is their a swimming activity under John's instantiated physical therapy care plan, as he can't swim. Therefore the care plan instance is now ready for transfer to a patient application for John to take away and execute. The care plan instance can be exported as an XML file, as shown below in figure 92, the schema of this XML file is shown in the earlier chapter in the section describing the care plan instance representation.

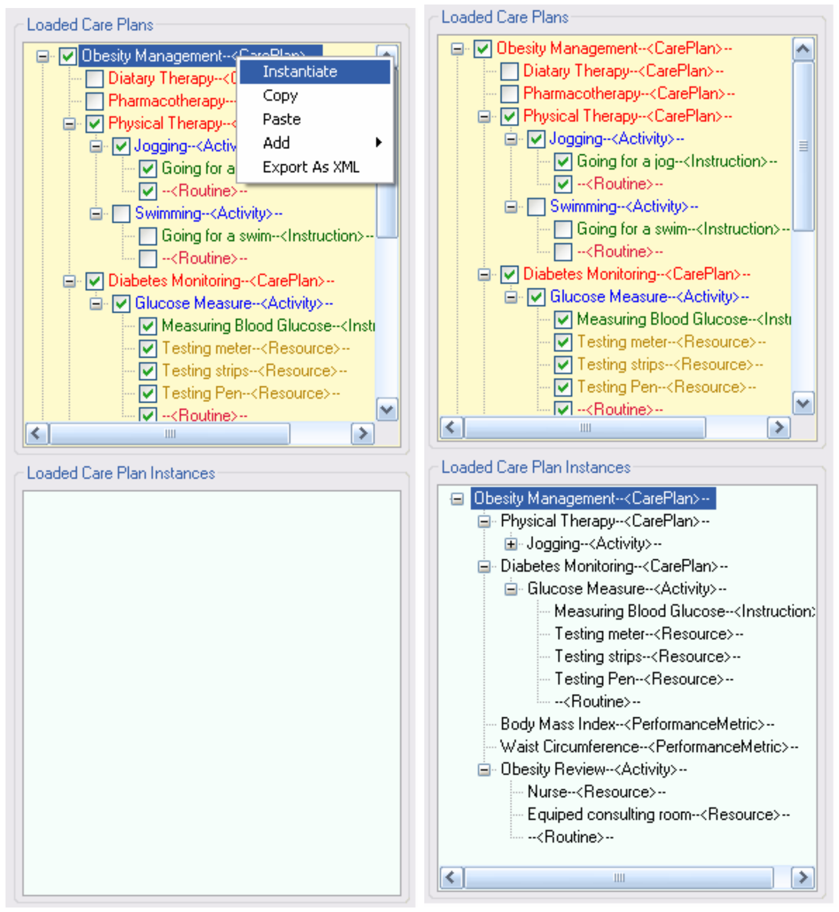
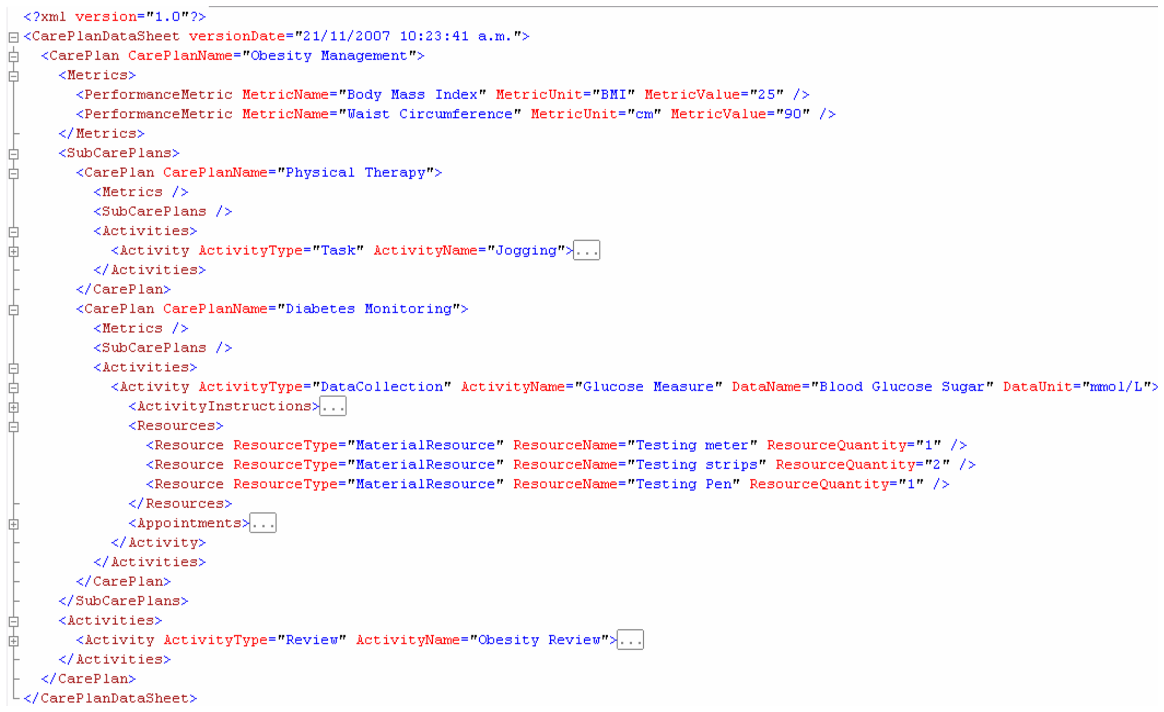

Loaded Care Plans
Obesity Management--<CarePlan>
Instantiate
Copy
Paste
Add
Export As XML
Diatary Therapy--<CarePlan>--
Pharmacotherapy--<CarePlan>--
Physical Therapy--<CarePlan>--
Jogging--<Activity>--
Going for a jog--<Instruction>--
--<Routine>--
Swimming--<Activity>--
Going for a swim--<Instruction>--
--<Routine>--
Diabetes Monitoring--<CarePlan>--
Glucose Measure--<Activity>--
Measuring Blood Glucose--<Instı
Testing meter--<Resource>--
Testing strips--<Resource>--
Testing Pen--<Resource>--
--<Routine>--
Loaded Care Plan Instances
Obesity Management--<CarePlan>--
Physical Therapy--<CarePlan>--
Jogging--<Activity>--
Diabetes Monitoring--<CarePlan>--
Glucose Measure--<Activity>--
Measuring Blood Glucose--<Instruction>
Testing meter--<Resource>--
Testing strips--<Resource>--
Testing Pen--<Resource>--
--<Routine>--
Body Mass Index--<PerformanceMetric>--
Waist Circumference--<PerformanceMetric>--
Obesity Review--<Activity>--
Nurse--<Resource>--
Equiped consulting room--<Resource>--
--<Routine>--

Figure 91: Showing the instantiation command for the instantiating a care plan.



```
<?xml version="1.0"?>
<CarePlanDataSheet versionDate="21/11/2007 10:23:41 a.m.">
  <CarePlan CarePlanName="Obesity Management">
    <Metrics>
      <PerformanceMetric MetricName="Body Mass Index" MetricUnit="BMI" MetricValue="25" />
      <PerformanceMetric MetricName="Waist Circumference" MetricUnit="cm" MetricValue="90" />
    </Metrics>
    <SubCarePlans>
      <CarePlan CarePlanName="Physical Therapy">
        <Metrics />
        <SubCarePlans />
        <Activities>
          <Activity ActivityType="Task" ActivityName="Jogging">...
        </Activities>
      </CarePlan>
      <CarePlan CarePlanName="Diabetes Monitoring">
        <Metrics />
        <SubCarePlans />
        <Activities>
          <Activity ActivityType="DataCollection" ActivityName="Glucose Measure" DataName="Blood Glucose Sugar" DataUnit="mmol/L">
            <ActivityInstructions>...
            <Resources>
              <Resource ResourceType="MaterialResource" ResourceName="Testing meter" ResourceQuantity="1" />
              <Resource ResourceType="MaterialResource" ResourceName="Testing strips" ResourceQuantity="2" />
              <Resource ResourceType="MaterialResource" ResourceName="Testing Pen" ResourceQuantity="1" />
            </Resources>
            <Appointments>...
          </Activity>
        </Activities>
      </CarePlan>
    </SubCarePlans>
    <Activities>
      <Activity ActivityType="Review" ActivityName="Obesity Review">...
    </Activities>
  </CarePlan>
</CarePlanDataSheet>
```

Figure 92: The exported care plan instance after instantiation is shown here as an XML file.

9.2.4 Modelling and generating the patient application for John

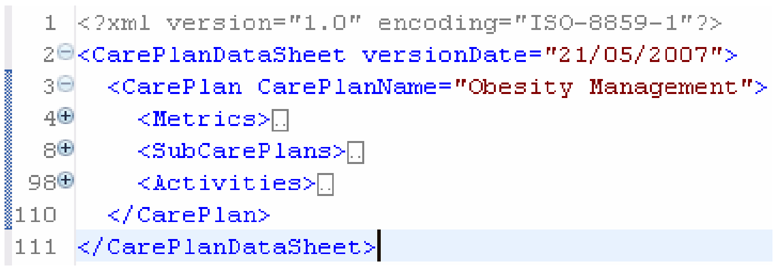
Once the care plan instance that is to be deployed to the patient is ready, the next step is to design the patient application that will graphically represent the care plan instance to the patient, and allow the patient to interact with the care plan. This designing is ideally done by a user who is familiar with GUI design principles, and in real life usage the health care professional probably has an entire database of different types of patient application definitions for different device types that they can use to generate the patient application. The following section exhibits the design and generation of some of the main screens for the patient application, in order to show clearly the functionality of the visual patient application modeller in action.

For the purpose of demonstrating the visual patient application modeller this section will demonstrate the modelling of three of the patient application screens:

1. The main screen of the patient application
2. Screen to represent individual care plans
3. Screen to represent the appointment for a data collection activity

9.2.4.1 Modelling the main screen for the patient application

The first step in modelling a screen for the patient application is to decide what care plan instance data that will be exposed through that part of the patient application. In this case the main screen is simply going to show a list of care plans that the patient is on. The hand programmed screen can be seen as the first from the left in the diagrams for the patient application that was built as a reference in chapter 7. In that screen it can be seen that the patient on that care plan is on many care plans, however John's care plan instance only contains one super care plan, i.e. the obesity management care plan, and this in turn contains three other care plans. The reduced data structure of John's care plan instance is shown in figure 93 below.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CarePlanDataSheet versionDate="21/05/2007">
  <CarePlan CarePlanName="Obesity Management">
    <Metrics>...
    <SubCarePlans>...
    <Activities>...
  </CarePlan>
</CarePlanDataSheet>
```

Figure 93: Snippet of code showing the structure of the care plan element, which in this case is the obesity management care plan that John is prescribed.

In order to model the main screen of the patient application, the first thing that needs to be modelled is the structure of the data that the main-screen will interact with, next this data structure needs to be associated with GUI elements that will represent this data. The diagram below shows a model of the main screen for the patient application built using the VPAM language, it can be seen that the data that is modelled as part of the model, includes a specification that literally reads as: *a care plan data sheet is an element which has one or more child elements called care plan, and each care plan has an attribute which is the name of the care plan.*

Once this data structure is established, then the care plan data sheet element can be represented by a view (interchangeable with a screen), called *MainScreen* it can be seen from the diagram this view is a display only screen, with a height of 150 and a width of 100, as well as having a simple vertical layout of any components that are contained within it. It can be seen from the diagram below, that the view contains a list, this list represents the names of all the care plans that are part of the care plan data sheet. It can also be seen that the list is of navigate type, with the URL specifying that if a click is made on a list item in the list representing the care plans, then the page that needs to be navigated to is the care plan view page, which will in turn load the care plan clicked on. It is important to note that the code for navigating to the care plan view, will only work if the care plan view has been modelled as well through the VPAM. The modelling of the page fore the care plan view is the next one demonstrated in the following sections.

Once the model of the main screen for the patient application is created using the VPAM, then the model can be extracted as an XML file and transformed using the same XSL style sheet that was used to transform the care plan diagrams, the transformed model of the patient application main screen can be seen in figure 95 below. It can be seen in this XML file that all the information pertaining to the diagram of the main screen model has been preserved, in that there are three data references, and two GUI elements, the mapping between the components can be seen in the file as target connection nodes. This file is now ready to pass to the patient application generator to generate the main screen of the patient application.

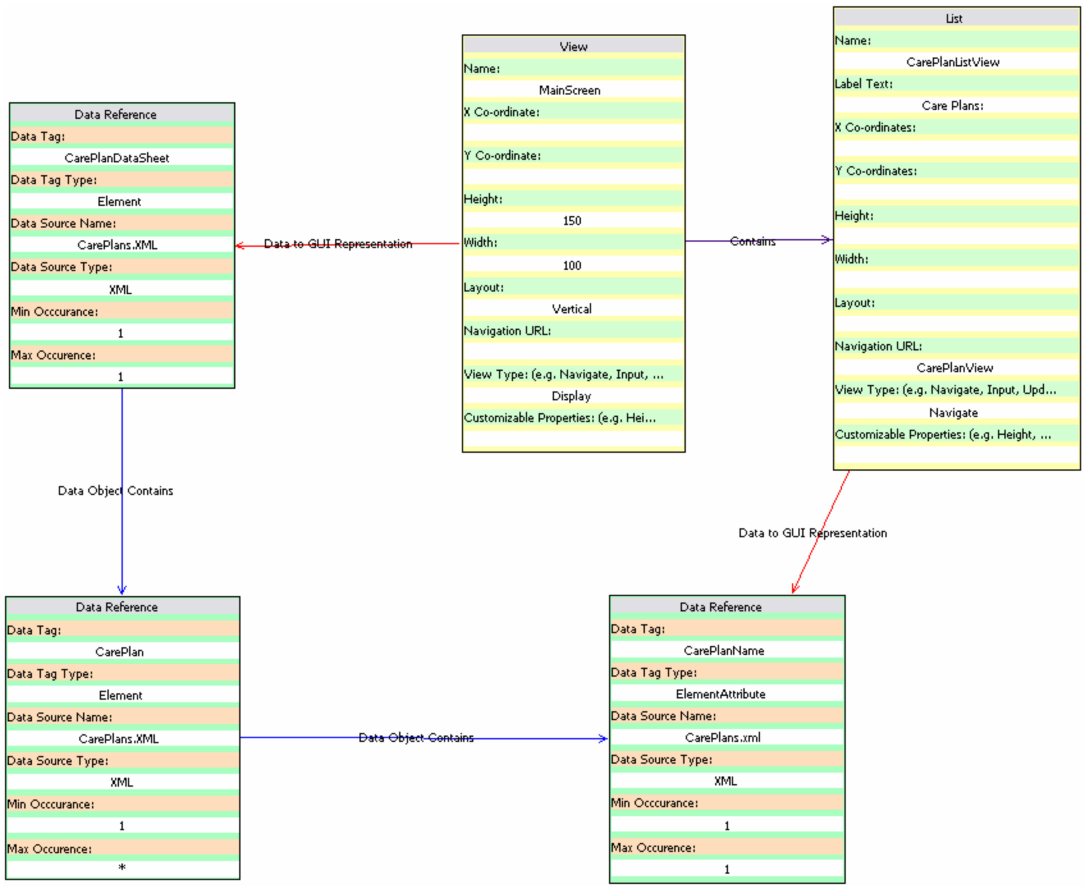


Figure 94: The VPAM model for the main screen for the patient application.

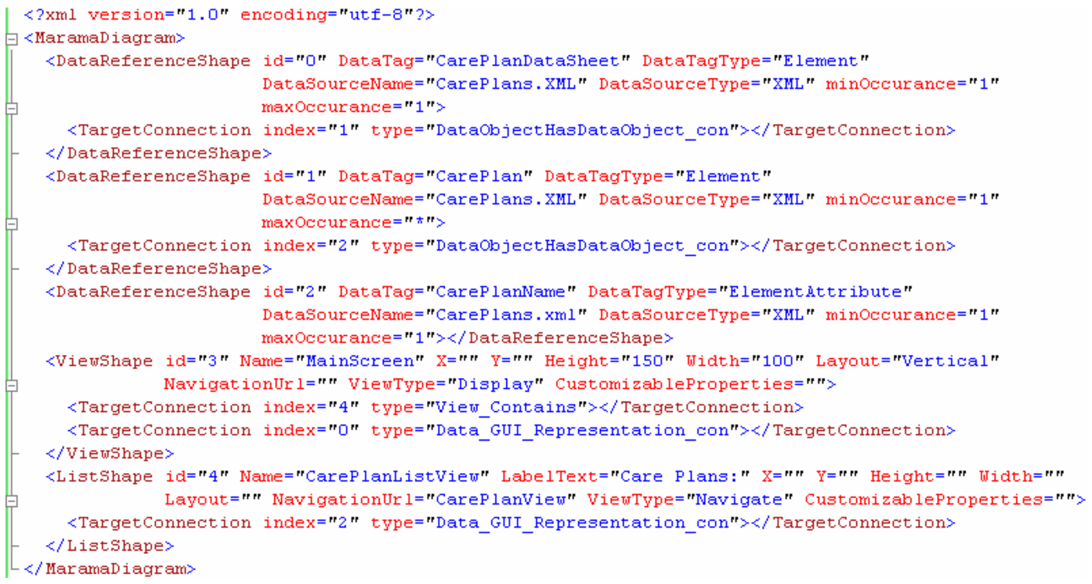

```
<?xml version="1.0" encoding="utf-8"?>
<MaramaDiagram>
  <DataReferenceShape id="0" DataTag="CarePlanDataSheet" DataTagType="Element"
                      DataSourceName="CarePlans.XML" DataSourceType="XML" minOccurance="1"
                      maxOccurance="1">
    <TargetConnection index="1" type="DataObjectHasDataObject_con"></TargetConnection>
  </DataReferenceShape>
  <DataReferenceShape id="1" DataTag="CarePlan" DataTagType="Element"
                      DataSourceName="CarePlans.XML" DataSourceType="XML" minOccurance="1"
                      maxOccurance="*">
    <TargetConnection index="2" type="DataObjectHasDataObject_con"></TargetConnection>
  </DataReferenceShape>
  <DataReferenceShape id="2" DataTag="CarePlanName" DataTagType="ElementAttribute"
                      DataSourceName="CarePlans.xml" DataSourceType="XML" minOccurance="1"
                      maxOccurance="1"></DataReferenceShape>
  <ViewShape id="3" Name="MainScreen" X="" Y="" Height="150" Width="100" Layout="Vertical"
            NavigationUrl="" ViewType="Display" CustomizableProperties="">
    <TargetConnection index="4" type="View_Contains"></TargetConnection>
    <TargetConnection index="0" type="Data_GUI_Representation_con"></TargetConnection>
  </ViewShape>
  <ListShape id="4" Name="CarePlanListView" LabelText="Care Plans:" X="" Y="" Height="" Width=""
            Layout="" NavigationUrl="CarePlanView" ViewType="Navigate" CustomizableProperties="">
    <TargetConnection index="2" type="Data_GUI_Representation_con"></TargetConnection>
  </ListShape>
</MaramaDiagram>
```

Figure 95: The VPAM model of the main screen of the patient application as a transformed XML document.

Since the care plan modelling system was essentially a proof of concept prototype, the patient application generation utility was not given a GUI of its own, instead the patient application generator was designed to be an extension of the care plan instantiation tool, whereby it can be accessed through the tools menu in the desktop application. Once the VPAM definition of the patient application main screen is passed through the patient application generator, there are three reusable units of OpenLaszlo code that are generated, the first item of code generated can be seen in figure 95 below, it shows the code for the overall main screen view that is to represent a list of care plans that the patient is on. It can be seen in the code that view has one initialising method, which sets the data pointer that the view will use to access it's data, then setting in the main window in which this view will show (also taken in as a parameter), and finally it instantiates the sub-GUI component contained in the view, which in this case is the care plan list, by calling it's initialising method.

The other two items of code generated include the code related to the functionality of the actual list of care plans contained in the main screen view. The code needed for this was generated as two reusable classes, the first class involved the class for an actual reusable list item, which had an initialising component to set the data pointer that the list item would need and also the view that would be navigated to when the list item is clicked on. In addition to this the generator also generated the action event method "doOnClick" which is called when the list item is clicked, the code for which takes care of the navigation to the page/view which represents the actual care plan.

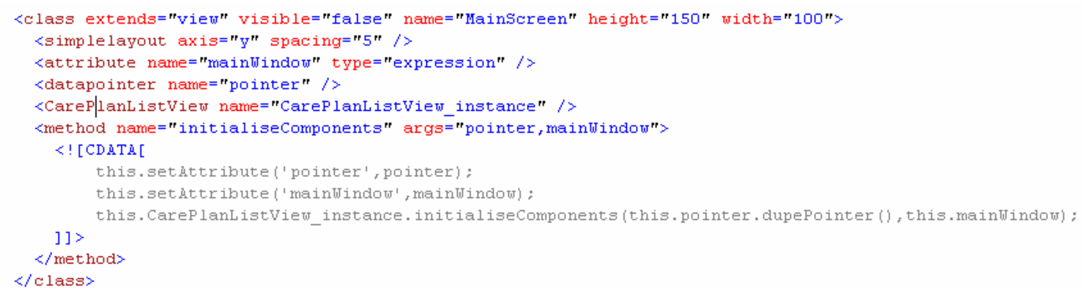
The second reusable class that was generated for implementing the list functionality included the class for the list itself. The list class as shown in figure 98 has an initialising method, which sets the data pointer, and also has the code which initialises the list items in the list, by moving the data pointer to the correct care plans and then calling the initialising method of each list item in the list sending in the data pointer pointing to the care plan that the list item is to represent as the parameter to the initialise components method of the list item.

Since the patient application generator is just a proof of concept implementation, the entire patient application was not generated using the generator, hence in order to test the sample reusable GUI code for the patient application main screen it was necessary to copy this code into the hand coded reference application to test the actual GUI that was portrayed by the generated code. The main screen expressed by the generated code can be seen in figure 99 below. It can be seen that it has the only care plan that the patient is on loaded, i.e. the obesity management care plan.

Some important assumptions made in the generation of the patient application main screen that are important to note include:

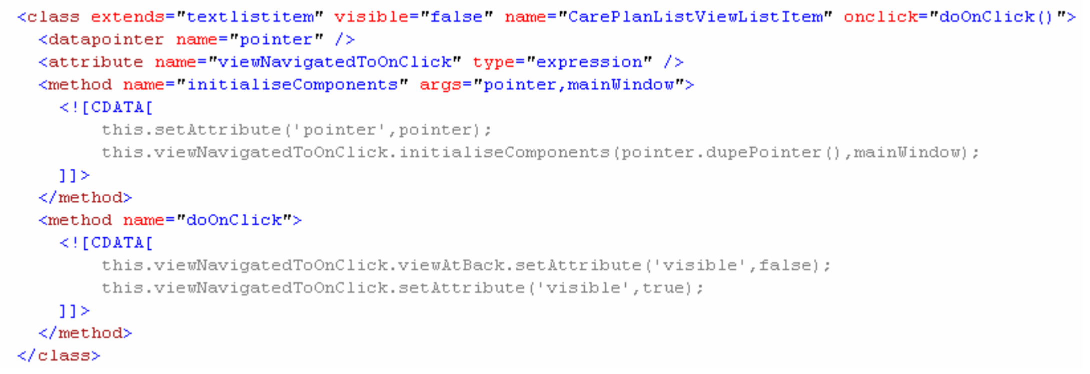
1. The data pointer that is passed to the main screen view in figure 96 has already been initialised to point to the right care plan instance data sheet that was emitted by the care plan instantiation tool.

2. The data pointer that is passed to the main screen view, has been moved so that it is pointing to the care plan data sheet node which is mapped to the main screen view in the VPAM model, hence the view as well as the list can move the data pointer in the forward direction to access the data that was mapped to it in the VPAM model, as is done in the initialising method of the list class in figure 98, where the data pointer is copied, and then moved to each care plan element in the care plan data sheet and then given to the list item that represents that particular care plan for use.



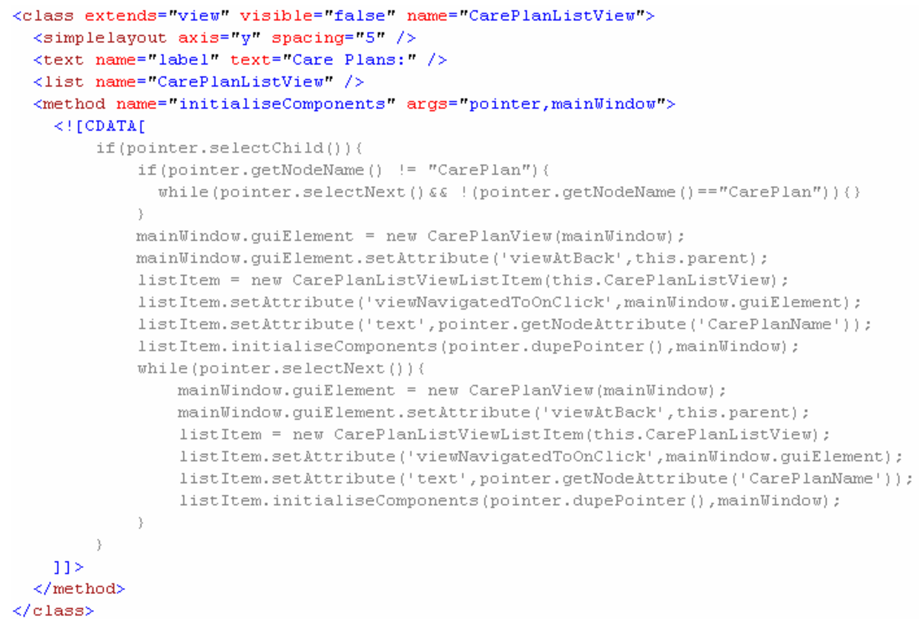
```
<class extends="view" visible="false" name="MainScreen" height="150" width="100">
  <simplelayout axis="y" spacing="5" />
  <attribute name="mainWindow" type="expression" />
  <datapointer name="pointer" />
  <CarePlanListView name="CarePlanListView_instance" />
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.setAttribute('mainWindow',mainWindow);
        this.CarePlanListView_instance.initialiseComponents(this.pointer.dupePointer(),this.mainWindow);
    ]]>
  </method>
</class>
```

Figure 96: Shows the reusable class that is generated for the main screen view of the patient application



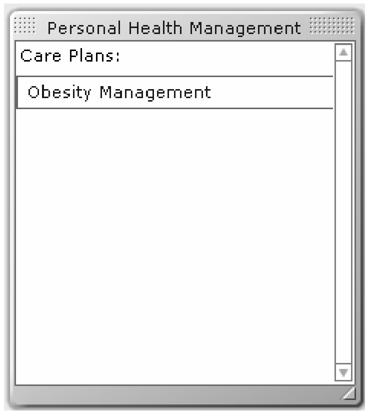
```
<class extends="textlistitem" visible="false" name="CarePlanListViewListItem" onclick="doOnClick()">
  <datapointer name="pointer" />
  <attribute name="viewNavigatedToOnClick" type="expression" />
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.viewNavigatedToOnClick.initialiseComponents(pointer.dupePointer(),mainWindow);
    ]]>
  </method>
  <method name="doOnClick">
    <![CDATA[
        this.viewNavigatedToOnClick.viewAtBack.setAttribute('visible',false);
        this.viewNavigatedToOnClick.setAttribute('visible',true);
    ]]>
  </method>
</class>
```

Figure 97: Shows the reusable class that is generated for the text list item that will be loaded into the list for representing the care plans that the patient is on.



```
<class extends="view" visible="false" name="CarePlanListView">
  <simplelayout axis="y" spacing="5" />
  <text name="label" text="Care Plans:" />
  <list name="CarePlanListView" />
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        if(pointer.selectChild()){
            if(pointer.getNodeName()  != "CarePlan"){
              while(pointer.selectNext() && !(pointer.getNodeName()=="CarePlan")){}
            }
            mainWindow.guiElement = new CarePlanView(mainWindow);
            mainWindow.guiElement.setAttribute('viewAtBack',this.parent);
            listItem = new CarePlanListViewListItem(this.CarePlanListView);
            listItem.setAttribute('viewNavigatedToOnClick',mainWindow.guiElement);
            listItem.setAttribute('text',pointer.getNodeAttribute('CarePlanName'));
            listItem.initialiseComponents(pointer.dupePointer(),mainWindow);
            while(pointer.selectNext()){
                mainWindow.guiElement = new CarePlanView(mainWindow);
                mainWindow.guiElement.setAttribute('viewAtBack',this.parent);
                listItem = new CarePlanListViewListItem(this.CarePlanListView);
                listItem.setAttribute('viewNavigatedToOnClick',mainWindow.guiElement);
                listItem.setAttribute('text',pointer.getNodeAttribute('CarePlanName'));
                listItem.initialiseComponents(pointer.dupePointer(),mainWindow);
            }
        }
    ]]>
  </method>
</class>
```

Figure 98: Shows the reusable class for the list that is to represent the care plans that the patient is on.



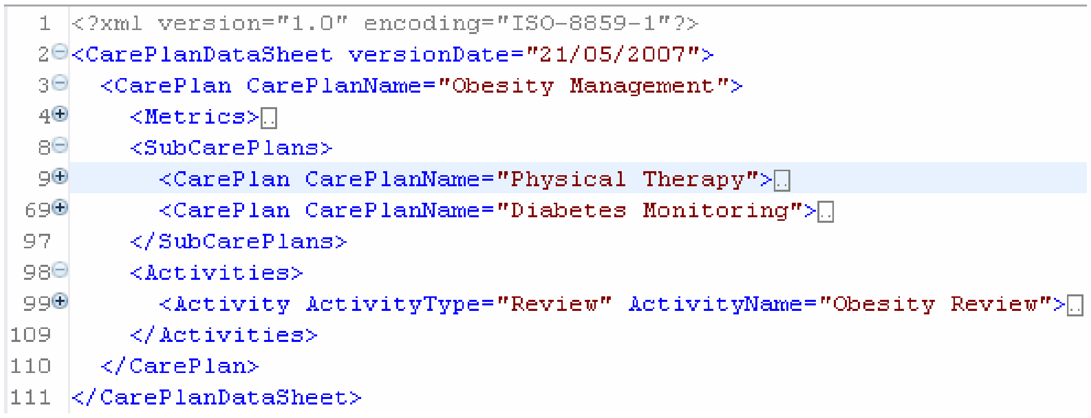
Care Plans:

Obesity Management

Figure 99: The generated main screen for the patient application containing one care plan loaded.

9.2.4.2 Modelling the screen for individual care plans

As expressed above the first thing that needs to be known when modelling the care plan view screen is the data structure of the care plan instance that needs to be represented by the GUI, this structure can be observed in the XML schema for the care plan instance data expressed in chapter 7, as well as a snippet of the actual data in reduced form can be seen in figure 100 below. From this structure it can be seen that the content that needs to be represented on each care plan's page includes, its name, the performance metrics that are part of the care plan, the sub-care plans that make up the care plan, and finally the actual activities that are involved in the care plan.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CarePlanDataSheet versionDate="21/05/2007">
  <CarePlan CarePlanName="Obesity Management">
    <Metrics>
    <SubCarePlans>
      <CarePlan CarePlanName="Physical Therapy">
      <CarePlan CarePlanName="Diabetes Monitoring">
    </SubCarePlans>
    <Activities>
      <Activity ActivityType="Review" ActivityName="Obesity Review">
    </Activities>
  </CarePlan>
</CarePlanDataSheet>
```

Figure 100: A snippet of the care plan instance that John is put on in XML form, showing the structure of the care plan element.

According to this each individual care plan page will not differ greatly from the main screen of the patient application expressed above, in that instead of having one list like the main screen, each page representing a particular care plan will consist of firstly a label indicating the name of the care plan that the user is viewing, then a list of performance metrics, a list of sub-care plans and finally a list of activities. Once again each of these lists will be of navigate type, meaning that clicking on a list item in the list of performance metrics will navigate to a view or a page to show more details of the performance metric, and likewise for the sub-care plan list, and the activities list. As mentioned above the actual pages to show this information, i.e. the information for each individual performance metric, activity etc, needs to be modelled in VPAM, so that the code for these screens can also be generated. The screen to represent individual care plans may also contain a "back" button to allow the patient to view previous screens with ease however to maintain the simplicity of the example this GUI component has been left out of the VPAM model of the care plan view screen.

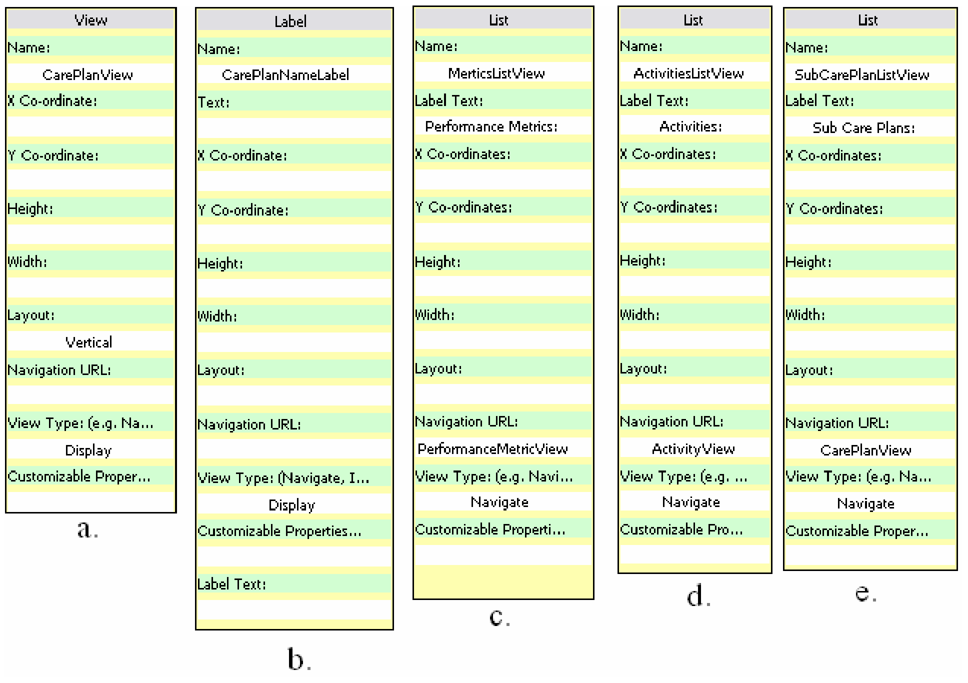
The VPAM model of the screen/view to represent individual care plans can be seen in figure 102 below, here it can be seen that the graphical shapes representing the components of the diagram have been reduced in size as compared to the model diagram expressed above, hence hiding the properties of the components. The main reason for this is that the diagrams are quite large and hence become difficult to portray here without reduction. The expanded GUI components however can be seen on their own in figure 101 below.

It can be seen from the diagram, that there are five GUI components that make up the entire GUI, firstly there is the *CarePlanView* which is the page which shows each individual care plan, it can be seen that this page is mapped to the CarePlan XML element. It can also be seem that the care plan view contains a label called CarePlanNameLabel which is in turn mapped to the care plan name XML attribute of the care plan element. Finally the care plan view contains within it three list, the metric list, the sub care plan list and the activities list, each of which are of navigate type, navigating to the correct pages to represent more information on each respective item. It can be noticed in the VPAM model that each list is mapped to an XML element attribute that will be represented as the text of the list item, in the case of the metrics list, this mapping maps to the MetricsName element attribute, which is what each list item in the metrics list will be represented by. Similarly the sub care plan list is mapped to the care plan name element attribute, and the activities list is mapped to the activity name element attribute.

When the VPAM model was passed into the patient application generator eight reusable classes were generated, the first one being a reusable class for the care plan view, next a reusable class for the care plan label view, and then two classes each for each of the list GUI components, in that one class is actually the customisable list item class for the list, and the second class is the actual reusable list class for that list. Once again to test the generated code, the code was integrated into the reference application which was hand coded, and tested with real data loaded onto the page, the generated GUI screen for the care plan view can be seen in figure 103 below. It can be seen here that the care plan represented is the obesity management care plan. All the reusable classes generated are not shown here as the code generated is quite similar to the main screen code generated above.

The assumptions made with the generation of this screen are the same as above in that:

1. The data pointer that is provided to the care plan view, should at least be pointing to the correct care plan node in the XML file representing the care plan instance.
2. The code of the initialising method of the GUI components will move the data pointer forward from its original point, to point to the right node, based upon all the data references that are mapped to the GUI component.


View
Name:
CarePlanView
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Vertical
Navigation URL:
View Type: (e.g. Na...
Display
Customizable Proper...

Label
Name:
CarePlanNameLabel
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, I...
Display
Customizable Properties...
Label Text:

List
Name:
MerticsListView
Label Text:
Performance Metrics:
X Co-ordinates:
Y Co-ordinates:
Height:
Width:
Layout:
Navigation URL:
PerformanceMetricView
View Type: (e.g. Navi...
Navigate
Customizable Properti...

List
Name:
ActivitiesListView
Label Text:
Activities:
X Co-ordinates:
Y Co-ordinates:
Height:
Width:
Layout:
Navigation URL:
ActivityView
View Type: (e.g. ...
Navigate
Customizable Pro...

List
Name:
SubCarePlanListView
Label Text:
Sub Care Plans:
X Co-ordinates:
Y Co-ordinates:
Height:
Width:
Layout:
Navigation URL:
CarePlanView
View Type: (e.g. Na...
Navigate
Customizable Proper...

Figure 101: Expanded GUI components of the VPAM model in figure 102 below.

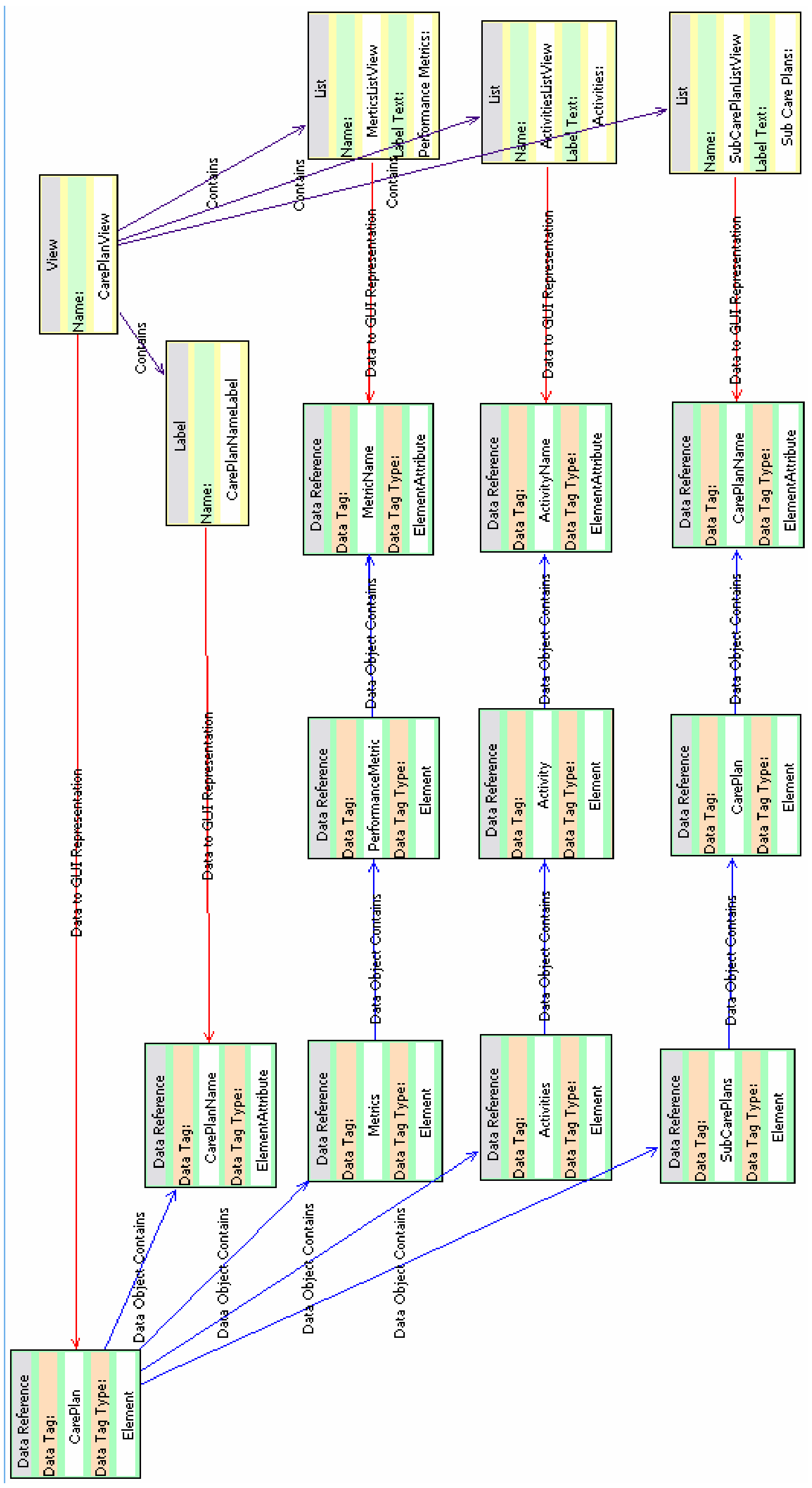
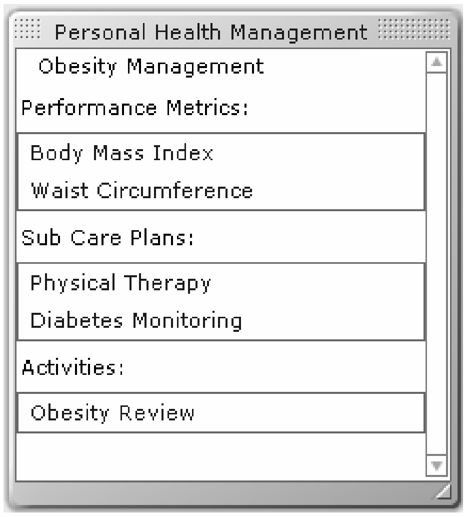


Figure 102: The VPAM model of the screen to represent individual care plans in the patient application.


Personal Health Management

Obesity Management

Performance Metrics:

Body Mass Index

Waist Circumference

Sub Care Plans:

Physical Therapy

Diabetes Monitoring

Activities:

Obesity Review

Figure 103: The care plan view screen generated from the VPAM model in figure 102, with the obesity management care plan data loaded into the screen.

9.2.4.3 Modelling the screens for a data collection appointment

So far the examples for the patient application modeller discussed above focus on modelling relatively simple screens that contain GUI components that are of display type (i.e. just displaying some care plan instance data) e.g. labels, and components of navigate type, such as list that navigate to other pages when their respective list items are clicked on. This section details the modelling of a slightly more complex screen which involves GUI components that actually take in input from the user and update the underlying care plan instance.

The part of the patient application that will be modelled and discussed here is the screen that is presented to the patient for a data collection activity appointment. A sample of the screen for a glucose measurement appointment can be seen in figure 104 below, here it can be seen that the patient is required to measure and enter in their glucose sugar in mmol/L. In addition to this the patient can also mark the status of the appointment through the status radio buttons, as well as enter in comments or concerns they had while doing this activity. Once the input is entered the patient can save the data entered on the page by pressing the save button (this saves all input, the data, status, and comments). Furthermore the page also has buttons to navigate back a page, and to navigate to the instructions page to read instructions on how to do the activity. For the purpose of the modelling example shown here, the back and instructions buttons will not be modelled here, and the buttons will be restricted to the save button (the two buttons however could be modelled in the system as navigate type buttons, but have been omitted for maintaining simplicity).

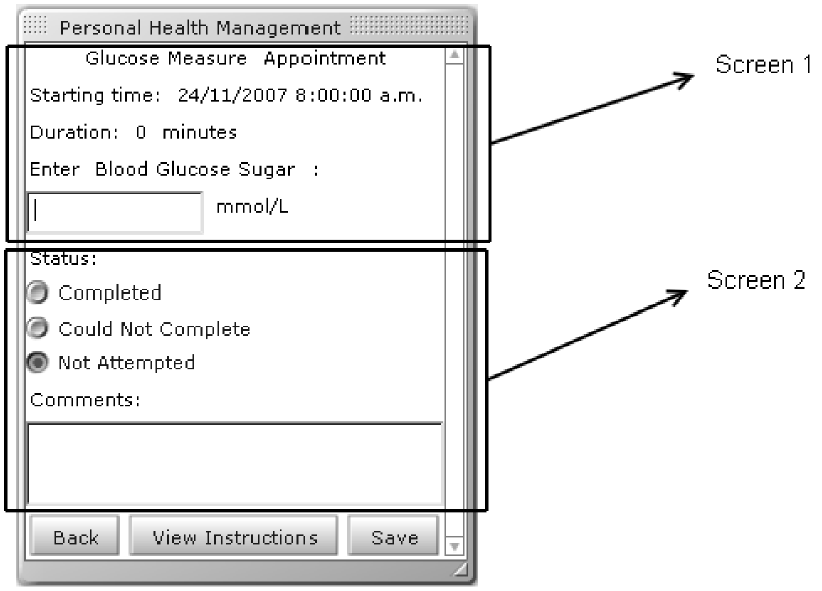
So far the screens that have been modelled in the example above, and as part of the reference patient application prototype have a size that is approximately equivalent to most decent sized PDA screens, however for the purpose of this example it is assumed that the John, instead of having a normal PDA, has a smart phone, which has a screen size quite a bit smaller than most normal PDAs, and hence the screen for the data collection activity appointment cannot be as big as shown in figure 104 below, therefore, the data collection appointment screen needs to be split up into two smaller screens that will fit onto John's smart phone screen (or alternatively a PDA with a smaller than usual screen). The proposed split up of the screen is marked out in figure 104 below.

According to this split-up, the first screen will have the following:

1. A view to represent the appointment name – probably a label
2. A view to represent the start date of the appointment – again a label
3. A view to prompt the user to enter the prescribed data – expressed using labels
4. A view to allow entering of the data prescribed – a text box and a label for the data unit.
5. Save button to allow data to be saved on the page
6. Next button to allow the next screen to be navigated to

According to this split-up, the second screen will have the following:

1. A view to represent the appointment name – as a label
2. A view to represent the group of status radio buttons – using a radio group with three radio buttons
3. A view to represent the comments input field – using a text area
4. A save button to save the data on the page
5. A back button to go to the previous page



Glucose Measure Appointment

Starting time: 24/11/2007 8:00:00 a.m.

Duration: 0 minutes

Enter Blood Glucose Sugar :

mmol/L

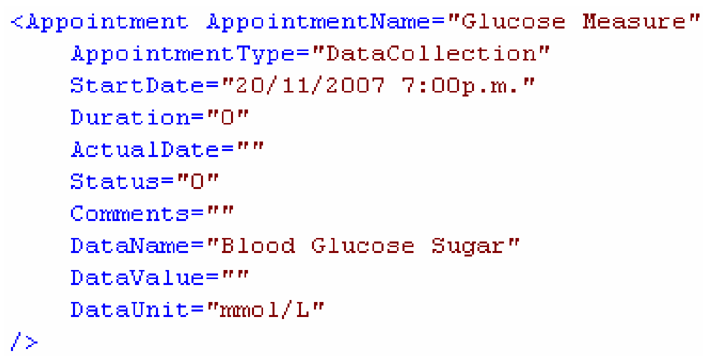
Status:

Completed [✓ Not Attempted] Could Not Complete

Comments:

Figure 104: The appointment screen for an activity which is of data collection type, and the markings show the way in which the screen can be split up into two smaller screens to fit a smaller display.

To keep the example simple the next, back and instruction viewing buttons will not be modelled as part of the two screens that will be modelled using the VPAM. Furthermore the structure of the appointment data that the two appointment screens will be interacting with can be observed in figure 105, which shows the appointment element with all its attributes specific to the appointment.



```xml
<Appointment AppointmentName="Glucose Measure"
    AppointmentType="DataCollection"
    StartDate="20/11/2007 7:00p.m."
    Duration="0"
    ActualDate=""
    Status="0"
    Comments=""
    DataName="Blood Glucose Sugar"
    DataValue=""
    DataUnit="mmol/L"
/>
```

Figure 105: An XML snippet of the care plan instance showing the structure of the appointment element.

9.2.4.3.1 Modelling the first data collection appointment screen

The model of the first data collection appointment screen can be seen in figure 106 below, it can be noted that just like the VPAM model of the care plan view screen shown above, the components in the diagram have been reduced to hide their properties, the main reason for this being making the diagrams more readable. The expanded GUI components in the diagram with the properties showing can be seen in figures 107 and 109.

It can be seen that the appointment screen maps to individual appointment elements, with appointment name, start date and duration being mapped to labels to represent these values. Furthermore the name of the data that the patient is to collect i.e. the *DataName* is mapped to a label that adds the data name to a prompt of the form *"Enter DataName:"*. Following this prompt there is a data entry view, this view has a horizontal layout, and contains two components, firstly it contains a textbox component mapped to the data value attribute of the appointment element, horizontally positioned next to this there is a label mapped to the data unit of the data value to be entered, the layout expected can be seen in figure 104 of the hand coded screen. An interesting thing that can be noticed about the data entry view containing the textbox, and the label, is that the type of the view is input type. The main reason for this is that view has contained within it another component which is of input type, i.e. the textbox taking in the user input for the data to be collected, hence making the entire view of input type. The patient application generator in turn will generate an update method for this view, in which the update method of all the input components contained in the view will be called, (in this case just the update method of the textbox component).

Finally at the bottom of the page there is a save button which is of update type and will write the input from all the input components on the page, to the underlying care plans instance, in this case the data entered in the textbox component of the data entry view will be written to the data value attribute of the appointment element in the underlying care plan instance.

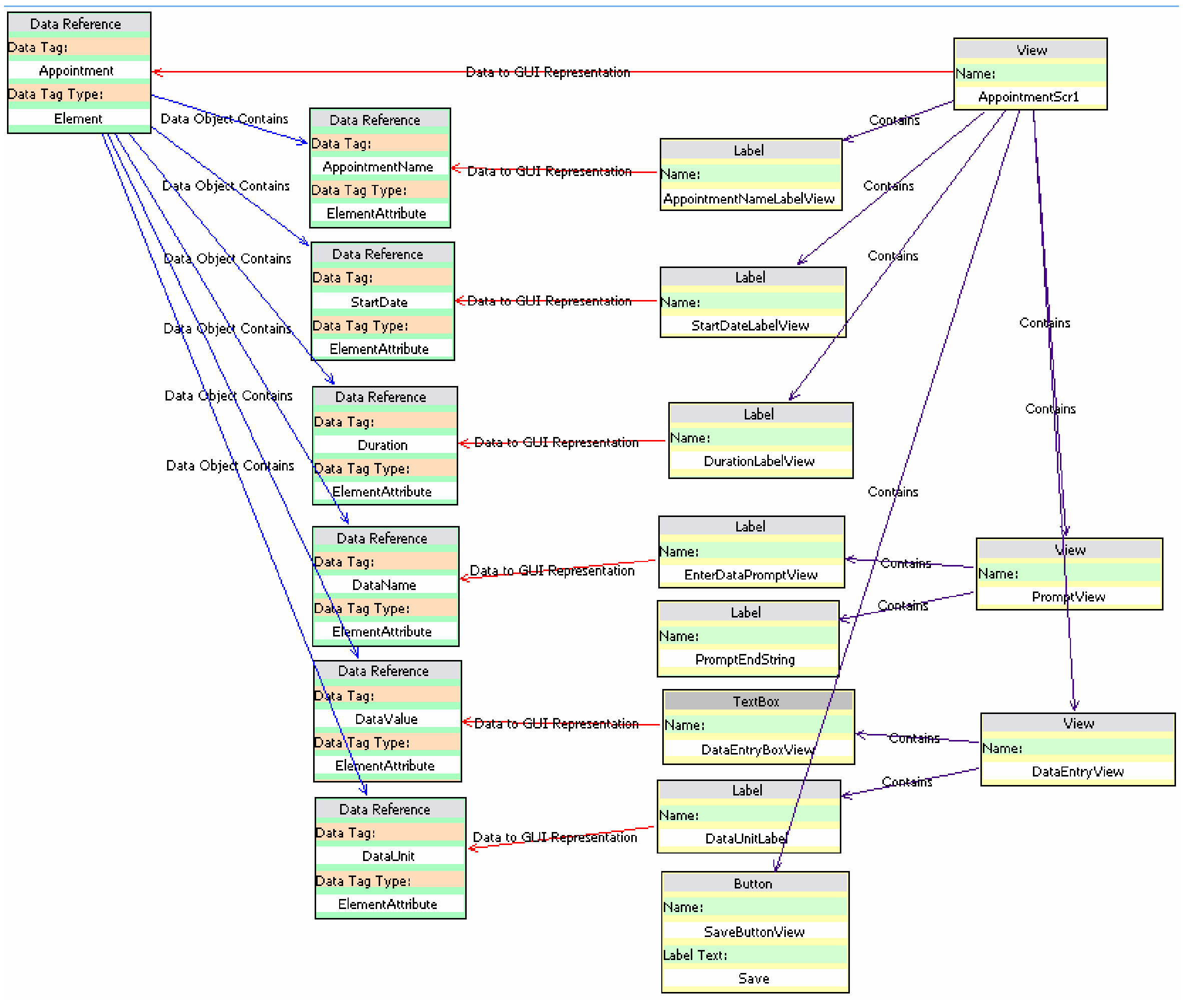
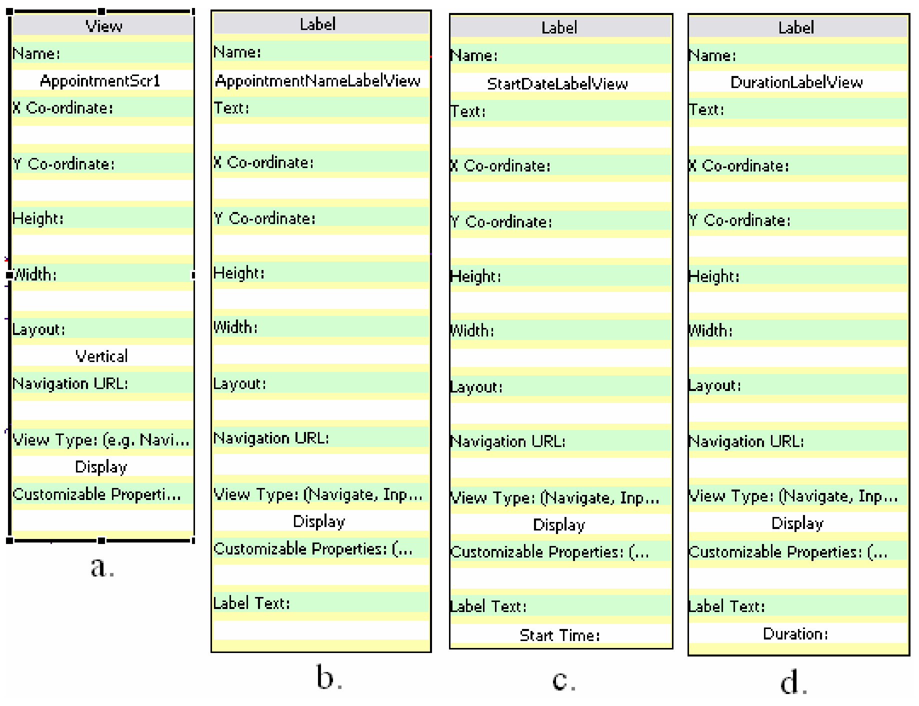


Figure 106: The VPAM model of the first split up appointment screen for a data collection activity.



View
Name:
AppointmentScr1
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Vertical
Navigation URL:
View Type: (e.g. Navi...
Display
Customizable Properti...

a.

Label
Name:
AppointmentNameLabelView
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:

b.

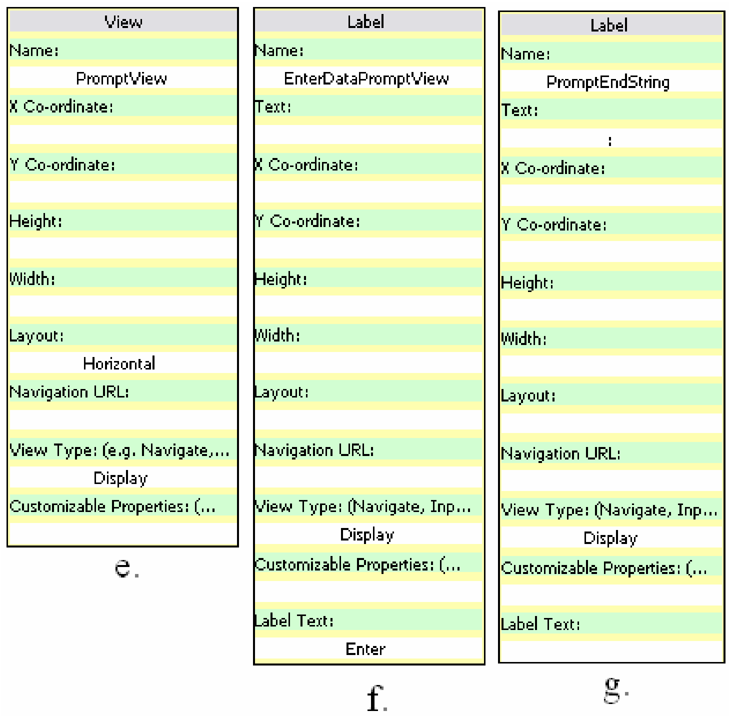
Label
Name:
StartDateLabelView
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:
Start Time:

c.

Label
Name:
DurationLabelView
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:
Duration:

d.

Figure 107: Expanded GUI components from the VPAM model shown in figure 106.



View
Name:
PromptView
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Horizontal
Navigation URL:
View Type: (e.g. Navigate,...
Display
Customizable Properties: (...

e.

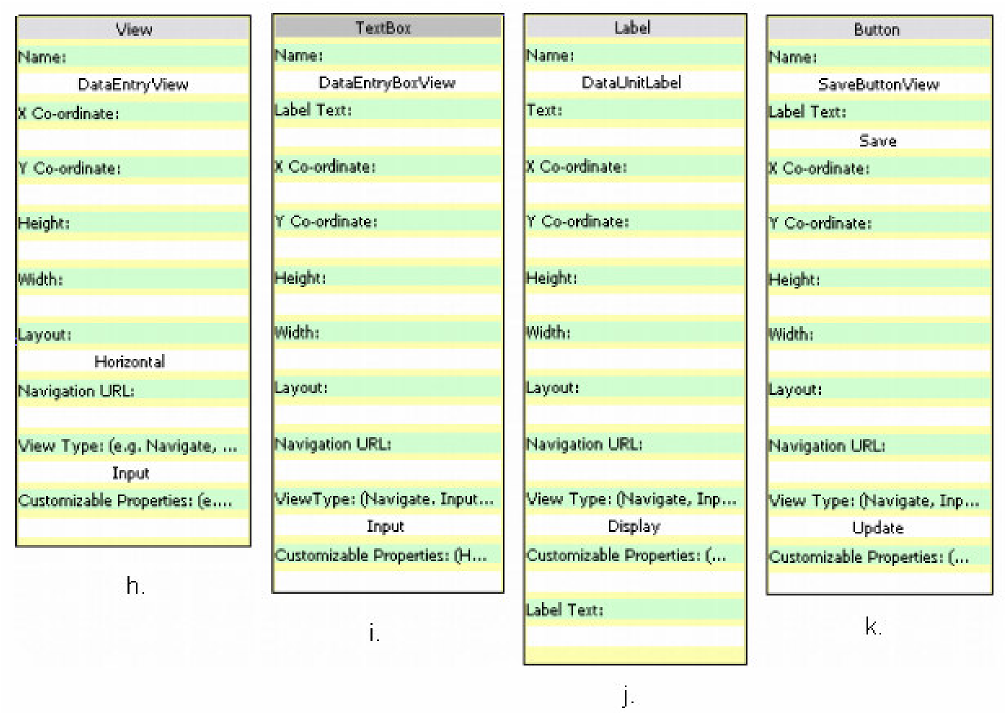
Label
Name:
EnterDataPromptView
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:
Enter

f.

Label
Name:
PromptEndString
Text:
:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:

g.

Figure 108: Expanded GUI components from the VPAM model shown in figure 106.


View
Name:
DataEntryView
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Horizontal
Navigation URL:
View Type: (e.g. Navigate, ...
Input
Customizable Properties: (e....

h.

TextBox
Name:
DataEntryBoxView
Label Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
ViewType: (Navigate, Input...
Input
Customizable Properties: (H...

i.

Label
Name:
DataUnitLabel
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Display
Customizable Properties: (...
Label Text:

j.

Button
Name:
SaveButtonView
Label Text:
Save
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp...
Update
Customizable Properties: (...

k.

Figure 109: Expanded GUI components from the VPAM model shown in figure 106.

9.2.4.3.2 Modelling the second data collection appointment screen

The VPAM model of the second screen can be seen in figure 110 below, as explained above, the purpose of the second screen was to take in patient notes and comments regarding the appointment conduction, as well as to allow the patient to mark whether or not the they have completed or could not complete the activity appointment. Therefore the VPAM model of the second screen contains three main GUI components, the first being a set of radio buttons, as can be seen in figure 110 there are three such radio buttons, mapping to the appointment status attribute, the radio buttons are grouped/contained in the status radio group. The expanded components for the radio group as well as the radio buttons can be seen in figure 111 c, d and figure 112 e, f. It can be seen that since the radio buttons are expected to take in input from the user regarding the status of the appointment, their type is of input, it can also be seen that since the radio button group contains input components, it's type is also of input. Furthermore from the expanded components in figures 111 and 112 it can be seen that each of these radio buttons have values, of 0, 1, and 2. When the page is saved, depending on the status chosen these values will be written to the appointment's status attributes, as specified by the VPAM mapping of the radio buttons, to the status element attribute.

The second GUI component on the screen is a text area that is mapped to the comments attribute of the appointment element. The purpose of this component is to gather in comments or concerns from the patient once they have neared the completion of the appointment. As explained above because the component takes in input from the user, it is of input type as can be seen in figure 112 g, and when the page is saved, this input will be written to the underlying comments attribute in the appointment element of the underlying care plan instance.

The third GUI component on the screen is the save button, the purpose of which is, when pressed, to write the input entered by the input type components on the page, i.e. the status radio group and its radio buttons and the comments text area to the underlying attributes and elements of the care plan instance.

The generated screens and code can be seen in the following section, which explains all the main parts of the code generated as well as the GUI screens that correspond to this code.

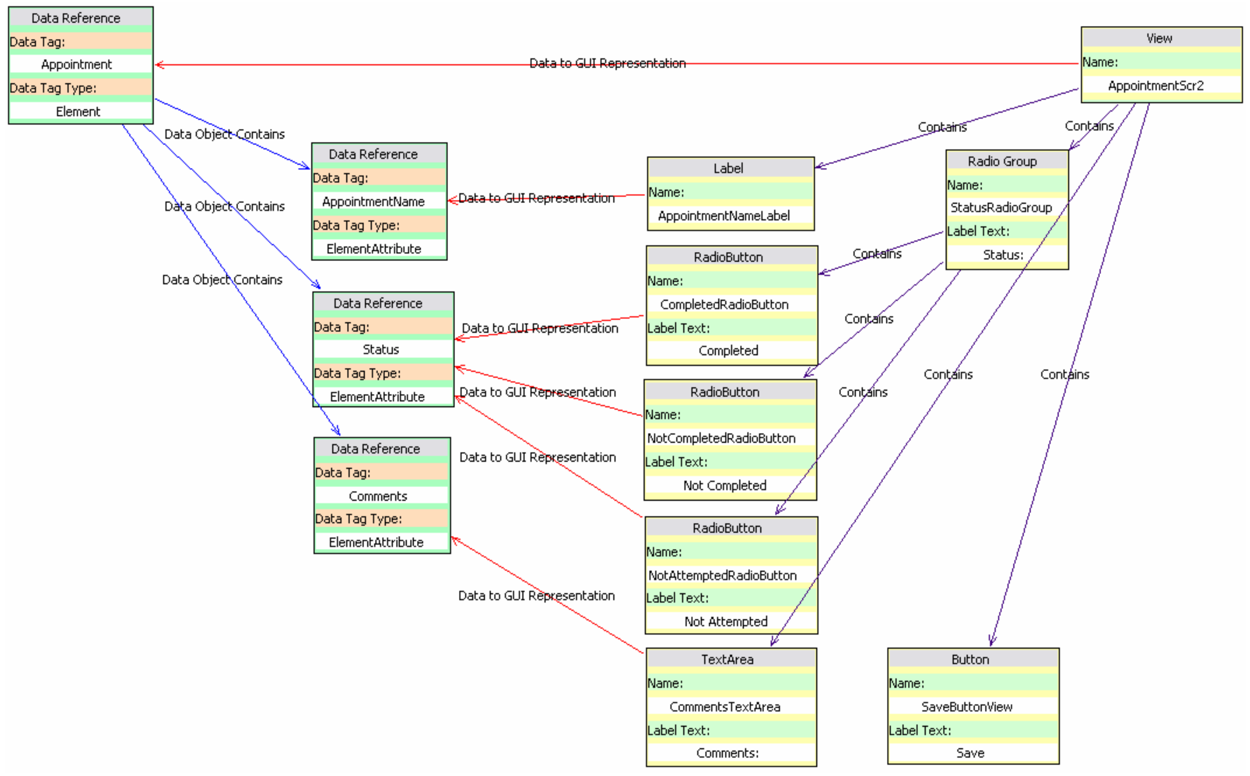
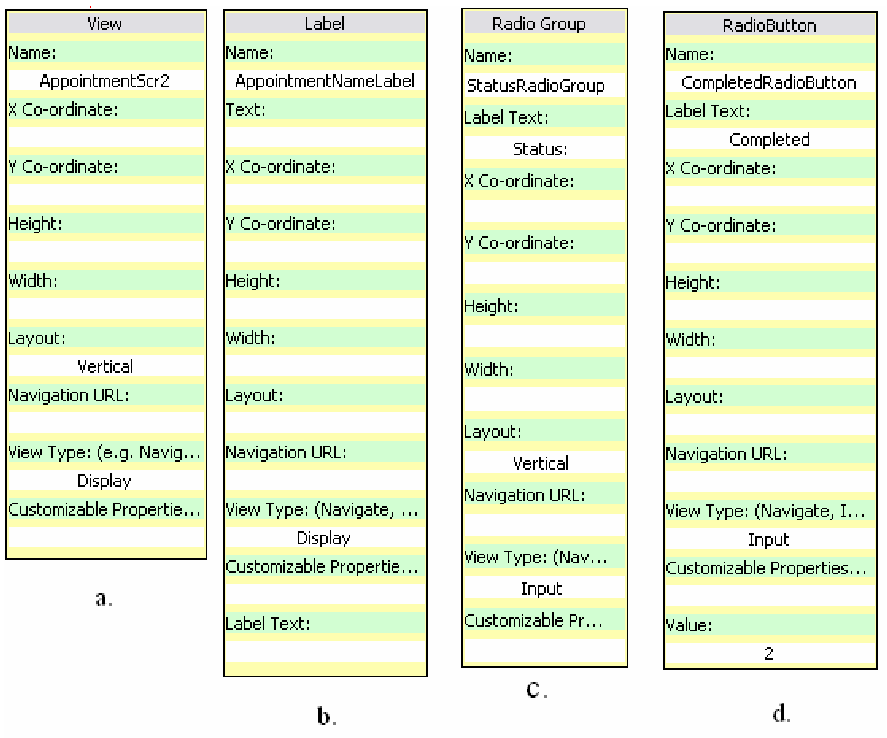


Figure 110: The VPAM model for the second appointment screen for a data collection activity.



View
Name:
AppointmentScr2
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Vertical
Navigation URL:
View Type: (e.g. Navig...
Display
Customizable Propertie...

a.

Label
Name:
AppointmentNameLabel
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, ...
Display
Customizable Propertie...
Label Text:

b.

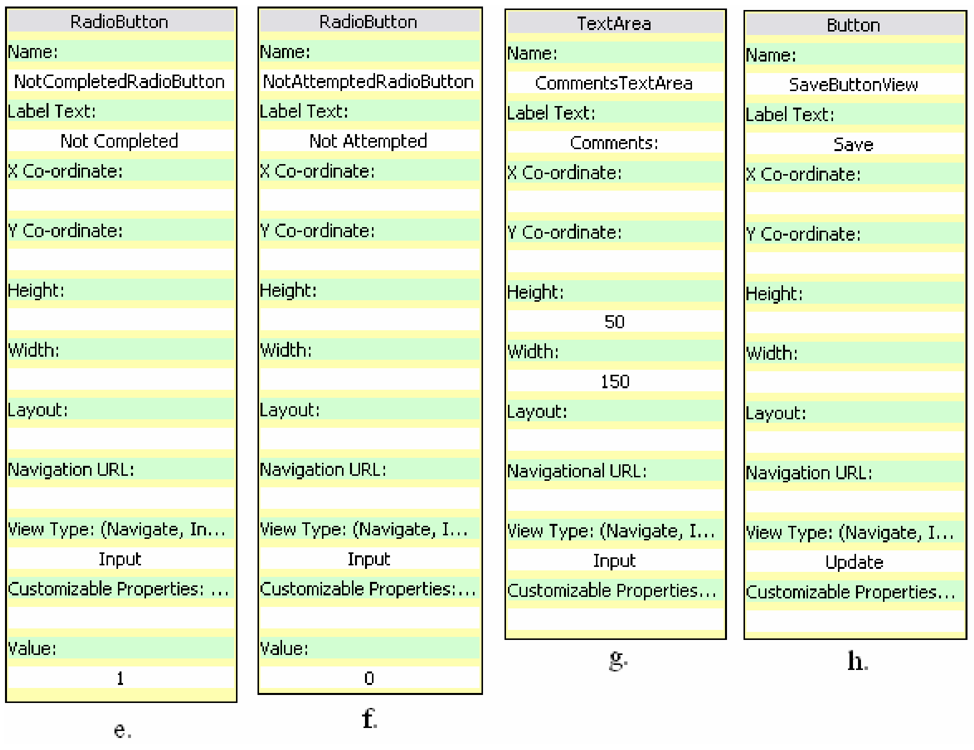
Radio Group
Name:
StatusRadioGroup
Label Text:
Status:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Vertical
Navigation URL:
View Type: (Nav...
Input
Customizable Pr...

c.

RadioButton
Name:
CompletedRadioButton
Label Text:
Completed
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, I...
Input
Customizable Properties...
Value:
2

d.

Figure 111: Expanded GUI components for the VPAM model in figure 110.



RadioButton
Name:
NotCompletedRadioButton
Label Text:
Not Completed
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, In...
Input
Customizable Properties: ...
Value:
1

e.

RadioButton
Name:
NotAttemptedRadioButton
Label Text:
Not Attempted
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, I...
Input
Customizable Properties:...
Value:
0

f.

TextArea
Name:
CommentsTextArea
Label Text:
Comments:
X Co-ordinate:
Y Co-ordinate:
Height:
50
Width:
150
Layout:
Navigational URL:
View Type: (Navigate, I...
Input
Customizable Properties...

g.

Button
Name:
SaveButtonView
Label Text:
Save
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, I...
Update
Customizable Properties...

h.

Figure 112: Expanded GUI components for the VPAM model in figure 110.

9.2.4.3.3 Generated screens and code

The two screens generated from the VPAM models explained above, by the patient application generator, can be seen in figure 113 b and c, with the sample of the original hand coded screen seen in figure 113 a. It can be seen that the screens contain the same components which were coded in the original screen, with the exception of the back, next, and view instructions buttons (shown with dotted lines on the actual screens in figure 113).

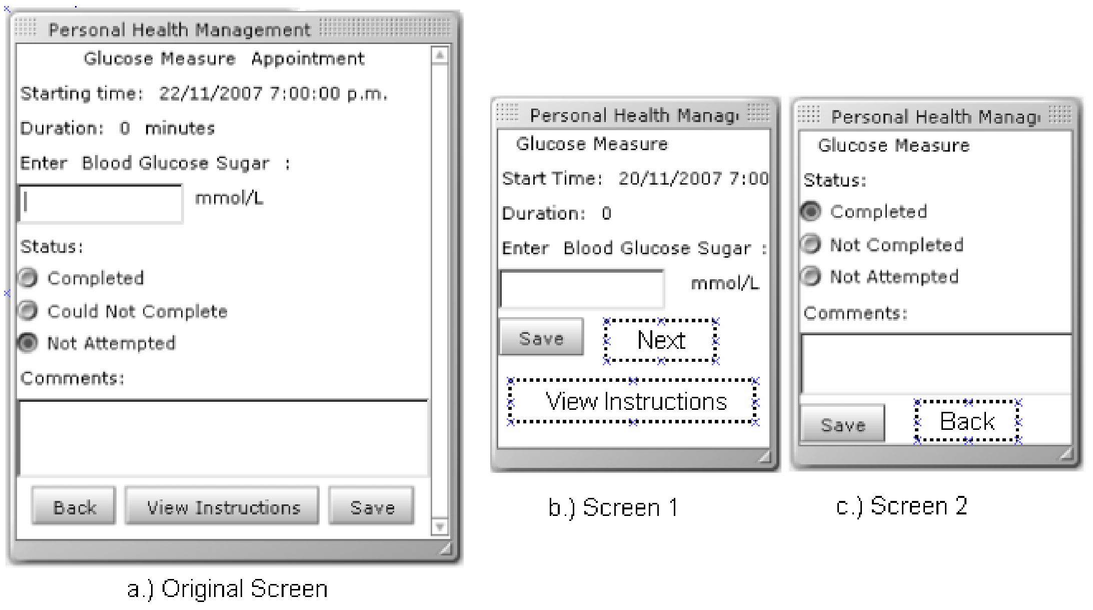


Figure 113: Diagram showing the sample original hand coded data collection activity appointment screen, along with the smaller split up two screens that are generated from two VPAM models in figures 106 and 110.

For screen 1 above, there were eleven classes that were generated, each class either describing some GUI component such as a button or a label, or describing an aggregation of GUI components, such as the class for the data entry view which aggregates two GUI components, the textbox for taking in the data from the user, and a label to describe the data unit next to the textbox. Similarly there were five classes generated for the second screen describing the GUI components on the screen.

A large portion of the code (classes) generated for the two screens were classes for customized labels, these are already explained in the above examples as demonstrations of GUI components that are of display type, as well as this there are examples of list components which are of navigate type, hence all the code that is generated for the two screens is not shown and explained here, however important aspects such as the code for input type components such as text boxes, text areas etc and the code for update type components such as save buttons, is discussed here.

The two screens shown above in figure 113 together have three different input type GUI components, first is the textbox in screen 1 which is meant to take in the glucose measurement from the user, the second is a text area on screen 2, which is meant to take in the comments and concerns of the patient once they have completed the activity, and lastly there is the set of radio buttons, which the patient can use to mark the status of the appointment once they have gone through it. For the purpose of demonstrating the code generated for input type GUI components, figures 114, 115, and 116 shows the code generated for the input type components of the comments text area in screen 2 and the radio group and radio buttons for status in screen 2. The code for the text box in screen 1 is omitted as it is very similar to the text area code generated for the comments input of screen 2.

```
<class extends="view" visible="true" name="CommentsTextArea">
  <simplelayout axis="y" spacing="5" />
  <attribute name="mainWindow" type="expression" />
  <datapointer name="pointer" />
  <text name="label" text="Comments:" />
  <edittext name="CommentsTextArea"  height="50" width="100" multiline="true" />
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.setAttribute('mainWindow',mainWindow);
        this.CommentsTextArea.setAttribute('text',pointer.getNodeAttribute('Comments'));
    ]]>
  </method>
  <method name="updateData">
    <![CDATA[
        this.pointer.setNodeAttribute('Comments',this.CommentsTextArea.getAttribute('text'));
    ]]>
  </method>
</class>
```

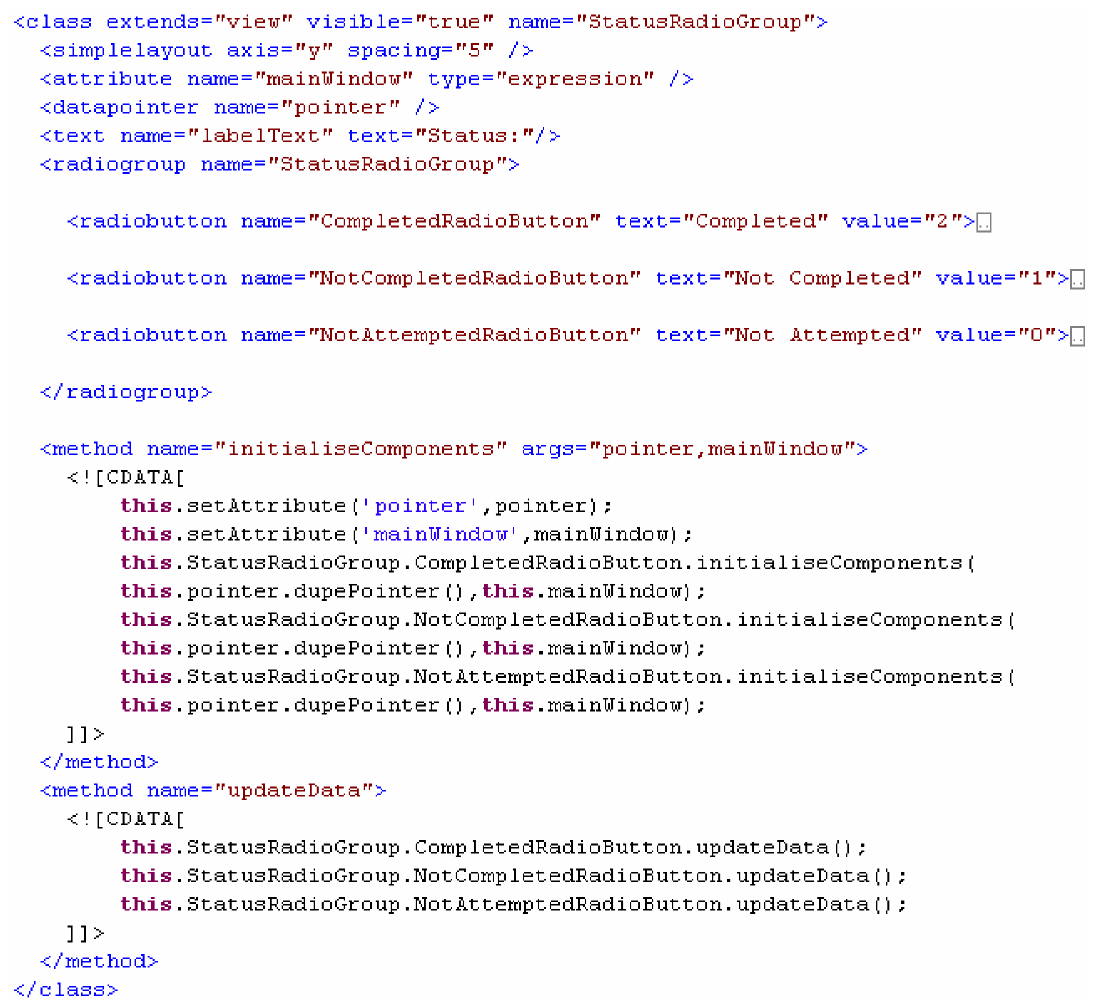
Figure 114: The OpenLaszlo code that was generated for the comments text area that is contained in the second appointment screen.

Figure 114 shows the code generated for the text area in screen 2 for entering in comments the patient may have after they have completed the appointment. It can be seen that just like the other GUI component classes this class also has an initialise components method, which sets the data pointer belonging to the text area with the one that is entered, and then loads the text area with the data from the comments attribute from the appointment element (which may be empty, if the appointment is a new one i.e. not done yet). It can also be noted that since the component is of input type there is an update data method that is generated; this in turn takes the data which is entered in through the text area, and uses its data pointer to update the underlying comments attribute of the appointment element, in the underlying care plan instance.

In a similar fashion to the comments text area of screen 2, the code for the status radio group and its contained radio buttons was generated as shown in figures 115 and 116. It can be seen in the code of the status radio group in figure 115 that it contains the initialisation of the three radio buttons contained within the group, i.e. one each for the completed, not completed and not attempted status. It can also be seen that the class for the status radio button contains an initialise components method, which initialises the data pointer of the class, along with the three radio buttons.

In essence the code for the three radio buttons is also included in the status radio group class shown in figure 115, however due to space constraints of the diagram, the code is not shown in figure 115, however the code for one of the individual radio buttons, namely the completed status radio button can be seen in figure 116, the code for the remainder of the status radio buttons is largely the same, and each radio button simply varies in its value.

It was discussed in the VPAM model of the second appointment screen, that each of the status radio buttons were of input type, similar to the comments text area, and that the radio group which contained the radio buttons, was in turn also of input type, hence it can be seen that both the class for the status radio group, and the code for the individual radio buttons, each has a method called update data. In the case of the individual radio buttons this method writes the value attribute of the radio button to the underlying status attribute of the appointment element of the underlying care plan instance providing that the button is in the selected state. In the case of the status radio group, this update data method calls the update data method of all the radio buttons that are contained within it. The update data method of the individual radio buttons will in turn check whether it is selected, and if it is, it will write its value attribute to the status attribute of the appointment element of the underlying care plan instance.



```
<class extends="view" visible="true" name="StatusRadioGroup">
  <simplelayout axis="y" spacing="5" />
  <attribute name="mainWindow" type="expression" />
  <datapointer name="pointer" />
  <text name="labelText" text="Status:"/>
  <radiogroup name="StatusRadioGroup">

    <radiobutton name="CompletedRadioButton" text="Completed" value="2">

    <radiobutton name="NotCompletedRadioButton" text="Not Completed" value="1">

    <radiobutton name="NotAttemptedRadioButton" text="Not Attempted" value="0">

  </radiogroup>

  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.setAttribute('mainWindow',mainWindow);
        this.StatusRadioGroup.CompletedRadioButton.initialiseComponents(
        this.pointer.dupePointer(),this.mainWindow);
        this.StatusRadioGroup.NotCompletedRadioButton.initialiseComponents(
        this.pointer.dupePointer(),this.mainWindow);
        this.StatusRadioGroup.NotAttemptedRadioButton.initialiseComponents(
        this.pointer.dupePointer(),this.mainWindow);
    ]]>
  </method>
  <method name="updateData">
    <![CDATA[
        this.StatusRadioGroup.CompletedRadioButton.updateData();
        this.StatusRadioGroup.NotCompletedRadioButton.updateData();
        this.StatusRadioGroup.NotAttemptedRadioButton.updateData();
    ]]>
  </method>
</class>
```

Figure 115: The OpenLaszlo code for the status radio group class that was generated for the second appointment screen.

```
<radiobutton name="CompletedRadioButton" text="Completed" value="2">
  <attribute name="mainWindow" type="expression" />
  <datapointer name="pointer" />
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.setAttribute('mainWindow',mainWindow);
    ]]>
  </method>
  <method name="updateData">
    <![CDATA[
      if(this.selected == true){
          this.pointer.setNodeAttribute('Status',this.CompletedRadioButton.getAttribute('value'));
      }
    ]]>
  </method>
</radiobutton>
```

Figure 116: The code for the completed status radio button, shown here as a sample of the code for all the radio buttons that are part of the status radio group, and essentially only differ in their value.

It can also be seen in the screens in figure 113 that both screens have save buttons, the purpose of them being to trigger the update methods of the input type components on the page such as the data text box, comments textarea, and status radio groups etc. so that the user entered data can be written to the underlying care plan instance. The generated code for the save button on screen 2 can be seen in figure 117 below, with the code for the save button on screen 1 being largely similar. It can be seen that the save button in addition to the initialising method, also has an action event method called *doOnClick* which calls within its body the update data methods of all the input type GUI components which are in its parent view, i.e. its siblings, i.e. on the same page as the button, in this case it means that the update data methods for the status radio group and the comments text area are called forcing the data entered through these components to be written to the underlying care plan instance.

```
<class extends="view" visible="true" name="SaveButtonView">
  <simplelayout axis="y" spacing="5" />
  <attribute name="mainWindow" type="expression" />
  <datapointer name="pointer" />
  <button name="SaveButtonView" onclick="classroot.doOnClick()" text="Save"/>
  <method name="initialiseComponents" args="pointer,mainWindow">
    <![CDATA[
        this.setAttribute('pointer',pointer);
        this.setAttribute(mainWindow);
    ]]>
  </method>
  <method name="doOnClick">
    <![CDATA[
        this.parent.StatusRadioGroup_instance.updateData();
        this.parent.CommentsTextArea_instance.updateData();
    ]]>
  </method>
</class>
```

Figure 117: The OpenLaszlo code for the class for the save button in the second appointment screen.

The same assumptions that were made for the modelling for the other screens before this were made for the modelling of the two appointment screens, in that the data pointer that is provided to both the appointment screens is pointing to the right appointment, and from their the data pointer is moved forward by the various GUI components to point it to the correct data element or attribute that is specified by the data reference mapped to the GUI component in the VPAM model.

9.2.4.4 Modelling the other screens for the patient application

In a similar way to modelling the screens shown in the examples above, the other screens for the patient application such as the screen to represent performance metrics, or the screen to represent activities, or resources etc can be modelled. The VPAM modelling facility due to being a prototype is of-course limited and hence the generated screens for the patient application may still need some hand coding, and some screens may not be able to be modelled at all, for example the screen for selecting appointments by date, with the calendar item. However the implementation of the VPAM is solely for the proof of concept of the model driven care plan modelling system, and hence due to timing constraints a more in depth implementation was not possible.

It is envisaged that in the future a health care professional may be able to maintain an entire database of ready made patient application definitions for different mobile and electronic devices which they can easily use to generate the patient application for whatever mobile device the patient has, and if a definition for the patient's mobile device does not exist, a health care professional can order one, where by a designer may use an advanced version of the visual patient application modeller (VPAM) to model the definition and send it to the health care professional. In addition to this multiple generators may be coded in order to generate the patient application in many other formats besides OpenLaszlo and Flash.

9.3 Summary

The example used in the demonstrating of the care plan modelling system, involves a patient, John, suffering from obesity. When John comes in to his GP to get help, the GP decides to design for him a care plan, the GP uses the visual care plan modelling language. The care plan that the GP designs for John is an obesity management care plan, the first thing the GP sets up is the performance metrics that can be used to judge John's progress, in the example these two metrics are the body mass index and the waist circumference, in addition to this there is also a 3 monthly review set up for John. In order to help John control the risks associated with his obesity and reduce his obesity the GP models two other sub-care plans under John's obesity management care plan, namely the diabetes monitoring and physical therapy care plans, under each there are activities of different types with instructions, resource needs, and general routines specified, for example the glucose measurement activity, swimming activity, jogging activity etc.

Once the main care plan has been designed the GP encodes under this care plan a treatment algorithm, in the form of an assessment module, which is essentially a decisional task flow diagram guiding the GP on how to assess John the next time he comes in. Once this is done it is now time to deploy the care plan to John. To do this the GP first loads the care plan into the care plan instantiation tool, the purpose of this is to fine tailor the care plan to the patient's specific needs. Once the care plan is loaded in this tool, the GP realises that some parts of it are not relevant to John, for example John cannot swim, hence the GP deselects this activity from John's physical therapy care plan, in a similar way the remainder of the care plan (s) is trimmed to suite John. After doing this the GP looks at the activities in each care plan, and sets detailed timetables around the general routines, so that John knows when exactly activities need to be done. Once this is done, the patient specific care plan instance is extracted as an XML file ready to be loaded into the patient application.

The next step is to model the patient application using the VPAM, so that the patient application can be generated using the patient application generator. For the purpose of the example shown here, only a part of the patient application modelling is shown, namely the modelling of the patient application main screen, the screen for representing individual care plans, and the two screens needed to represent activity appointments. Once the VPAM models for the screens are generated they can then be used to generate the screens for the patient application in OpenLaszlo which is later compiled to Flash. Due to the fact that the VPAM is a proof of concept prototype the generated screens may need hand coding for representation of complex events is not possible through the VPAM, as well as this some screens may not be able to be modelled through the VPAM due to the lack of an extensive toolbox of GUI components, hence screens with complicated GUI components such as calendars and date spinners are not supported in the modelling suite.

Chapter 10 - Evaluation

10.1 Introduction

The purpose of this chapter is to present three elements of evaluation which were conducted for the care plan modelling system. The three parts of evaluation done include a cognitive dimensions evaluation for the visual care plan modelling language, followed by another cognitive dimension analysis for the visual patient application modeller, and finally an informal evaluation for the care plan modelling system done by experts in the field of visual language and modelling suite design.

10.2 Cognitive dimensioning explained

The cognitive dimensioning approach is one which was first developed by Green and Petre in 1996. The approach includes a methodology through which visual environments such as visual programming and modelling languages can be assessed. The approach sets up key questions that a developer needs to ask about the design of their visual language while designing and assessing it. These questions pertain to a set of “dimensions” of visual language assessment and are as follows (these dimensions and questions are reproduced from the lecture notes from (Hosking 2007)):

1. **Abstraction gradient**: What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?
2. **Closeness of mapping**: What ‘programming games’ need to be learned?
3. **Consistency**: When some of the language has been learnt, how much of the rest can be inferred?
4. **Diffuseness**: How many symbols or graphic entities are required to express a meaning?
5. **Error proneness**: Does the design of the notation include careless mistakes?
6. **Hard mental operations**: Are there places where the user needs to resort to fingers or pencilled annotation to keep track of what’s happening?
7. **Hidden dependencies**: Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?
8. **Premature commitment**: Do programmers have to make decisions before they have the information they need?
9. **Role expressiveness**: Can the reader see how each component of a program relates to the whole?
10. **Secondary notation**: Can programmers use layout, colour or other cues to convey extra meaning, above and beyond the official semantics of the language?
11. **Viscosity**: How much effort is required to perform a single change?
12. **Visibility**: Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side-by-side at will? If the code is dispersed is it at least possible to know in what order to read it?

These dimensions and questions were used to walk through the two visual languages that were designed, i.e. the visual care plan modelling language and the visual patient application modeller in order to conduct an initial evaluation of the two modelling languages. It is important to recognise that these dimensions do not act independently and there are always tradeoffs, where by improving one aspect may deteriorate another.

10.3 Cognitive dimensioning of the visual care plan modelling language

10.3.1 Abstraction Gradient

What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?

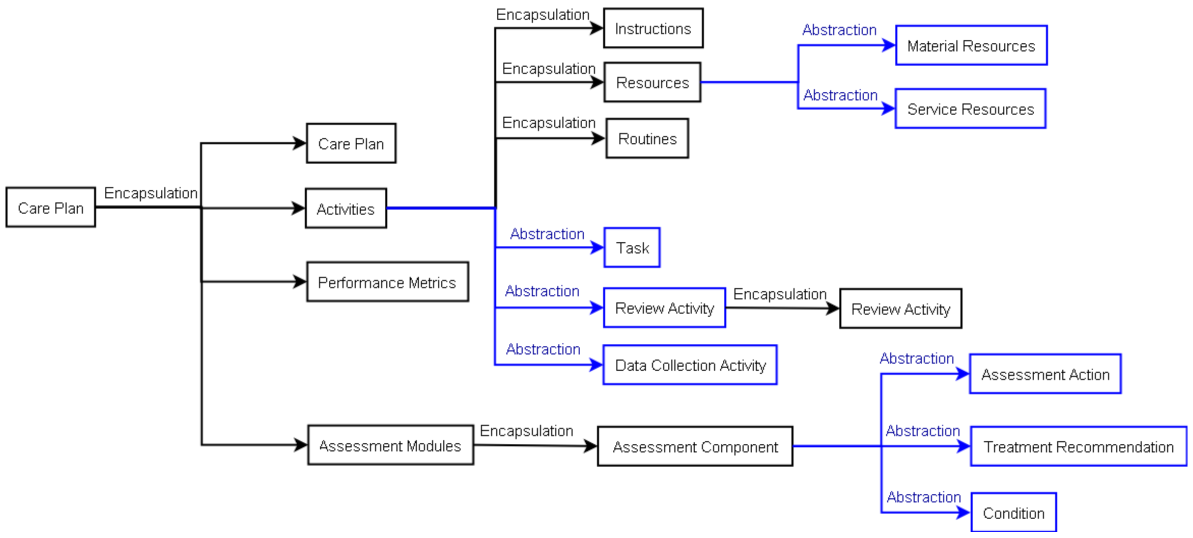


Figure 118: Abstraction gradient of the care plan model.

In the above diagram the maximum level of abstraction is the care plan. The care plan encapsulates all other components/ fragments of the care plan model, which includes other care plans, activities that are part of the care plan, performance metrics of the care plan, and finally the assessment modules guiding the assessment of the patient on the care plan. An example of this can be observed in the concrete obesity management scenario discussed in chapter 5 where the obesity management care plan has sub care plans such as dietary therapy, and physical therapy, as well as having health care activities such as jogging and swimming. In addition to this the obesity management care plan also includes performance metrics such as the waist circumference of 90 cm and the body mass index (BMI) of 29. Furthermore the obesity management care plan has an assessment module (or a treatment algorithm), guiding the health care professional on how to assess the patient's health and progress.

It can be seen from the above diagram that an activity encapsulates other components such as instructions describing how the activity is to be done, resources describing what are the materials or services that are needed to the activity, and also routines which describe how often and for how long an activity needs to be done. An activity however is an abstraction with three concrete types, i.e. an activity can be a simple task for example go for a jog, or a review activity for example an obesity review which in turn may encapsulate other review activities such as the diabetes and cholesterol review, and finally an activity can be a data collection activity, such as the glucose measurement activity part of the diabetes monitoring care plan.
A care plan also encapsulates assessment modules. The purpose of the assessment module is to allow a health care professional to encode a treatment algorithm in order to clearly specify the way in which the patient's health is to be

assessed; hence an assessment module is really a decisional task flow. The assessment module hence encapsulates multiple assessment components, an assessment component being one of three types, the first is the assessment action, the second is the treatment recommendation, and the third is the conditional statement. Using these components and specifying the conditional and sequential flow relationships between them the health care professional can encode a treatment algorithm similar to shown in chapter 5 above.

10.3.2 Closeness of Mapping

What 'programming games' need to be learned?

The main programming game that needs to be learned is the actual syntax of the care plan modelling language, this learning process includes learning the encapsulation relationships between components, learning the different concrete representations of abstract components, and learning any other relationships explicit or hidden that may be present or need specifying during the modelling process. For example it is necessary for a health care professional to know what the different components of a care plan are, what the different types of activities are, what are the different types of relationships between assessment components in an assessment module etc.

In addition to this the health care professional needs to come to terms with the idea of modelling care plan templates rather than care plan instances. As care plan templates disease specific and relevant to any patient suffering from that ailment, and a care plan instances are disease specific, however have been fine tailored to the patient's personal needs. For example while modelling a health care activity the health care professional should specify a generic routine which describes how often and for how long an activity must be done without specifying details on the exact timetable or days and hours that the activity will be done as this is something which is more specific to a patient as it depends on the patients already existing daily timetable.

In addition to this the health care professional has to come to terms with the fact that an effective template is one that is complete, i.e. includes the care plans and activities for treating all aspects of the condition/disease, and not parts that are relevant to only a particular group of patients, because once the template is created it is to represent the complete management plan for all patients suffering from the condition, and that selective specialisation and personalization of the template is done only during the instantiation period, where literally a template is made *"relevant to a patient"*.

10.3.3 Consistency

When some of the language has been learnt, how much of the rest can be inferred?

The main learning curve centres around two aspects, the first is getting use to the object-oriented care plan hierarchy, i.e. as mentioned above, getting use to the components of the care plan and the relationships between them, in the context of the visual language.

Secondly it is the idea of modelling a complete care plan template, rather than focusing on only a few routes of treatment, it is important to focus on all routes of treatment possible for the management of the condition, and deferring the selection of a route (i.e. the personalization of the care plan) to the instantiation of the template for a particular patient.

After these two aspects have been learned, then a user can infer many different ways of reusing templates to create new templates, by utilizing their object-oriented understanding of the care plan components, they can reuse anything from individual activities, to entire care plans, to form larger, and more expressive care plan templates.

10.3.4 Diffuseness

How many symbols or graphic entities are required to express meaning?

As mentioned in the design of the care plan modelling system in chapter seven prior to this, the graphical notation that is used for the care plan modelling language is a simple box and line notation, in that all the entities in the care plan modelling language are represented by colour coded rectangular boxes and the relationships between them is represented in the form of lines. Hence as can be seen from the diagram in figure 118 above the care plan modelling language has 12 entities and hence there are 12 symbols in the visual language, each being a colour coded rectangle.

10.3.5 Error-proneness

Does the design of the notation induce “careless mistakes”?

There are no major types of errors which can occur while using the visual care plan modelling language, the only type of error which may occur is a user trying to add an invalid component to another component, for example adding a care plan to an activity does not make sense, as a care plan usually contains activities not the other way round. This type of mismatch is not checked for in the language, as it is relatively simple and once a health care professional gets to know the syntax of the language, these types of mistakes may rarely happen. Other small errors which could occur is forgetting to add certain necessary components to the care plan, hence leaving the care plan incomplete, for example a health care activity should contain at least one routine, however having an activity without a routine will not cause an error, the routine can as easily be added through the care plan instantiation tool wizard before the actual instantiation.

10.3.6 Hard mental operations

Are there places where the user needs to resort to fingers or pencilled annotation to keep track of what’s happening?

The hard mental operations that the health care professional may face while modelling a care plan using the visual care plan modelling language are related to the programming games that need to be learned as expressed above, in that before a health care professional starts to model a care plan they need to determine how broad the care plan is going to be, i.e. will the care plan template cater for a specialised group of patients suffering from that ailment or will the care plan have enough options to cover all type of people who may suffer from the condition. In addition to this the health care information which is put into health care plans is relatively complex in nature and hence it may be a hard mental operation for the health care professional to recall this information all at once to model an entire care plan, the reason for this is that most usually health care professionals recall this information on a case by case basis for each individual patient, this hard mental operation may be eased by the fact that the health care professional is not expected to model the entire care plan all at once, but instead slowly evolve the care plan through time (see the section below on premature commitment).

Other hard mental operations may involve remembering the syntax of the care plan modelling language, i.e. remembering what components are composed of what other components, for example remembering that a health activity should have at least one instruction, or a routine etc. Due to a constraint in time it was not possible to invest a great deal of time in the design of the graphical notation of the language; however it is important to realise that a more intuitive graphical notation (using a better metaphor) over the same language can easily solve this problem, in addition to this the language can be augmented with various design critics which would analyse the care plan in real time as it is built and inform the health care professional on things they need to correct in the health care plan, however this also was out of the scope of the research done.

10.3.7 Hidden dependencies

Is dependency overtly indicates in both directions? Is the indication perceptual or only symbolic?

The dependencies between each component in the care plan modelling language/ meta model can be observed in the table below, it can be seen that most of the relationships between the components is that of containment (or encapsulation as can be observed in the above abstraction/encapsulation gradient), for example care plans containing other care plans, activities, performance metrics and assessment modules, in a similar manner activities are also composites of other components such as instructions, resources and routines. All of the containment relationships are explicitly represented by the graphical connectors between the entities in the visual language description of the care plan, hence it is sufficiently easy to determine which components belong or are contained in which other components. One type of relationship which may be implicit is realising that if a care plan is contained in a parent care plan, the child care plan inherits properties such as performance metrics of the parent care plan, in a similar way a review activity is also a composite of other review activities and hence an implicit relationship/dependency is present between the resources, routines and instructions of the parent activity and the child activity, in the child activity also inherits the same components. These relationships were not expressed in a concrete explicit graphical form as they were considered to be relatively simple and can be easily deduced through knowledge of the care plan syntax. Other notable relationships include the relationships between the assessment components, these can be of two types, the first being a simple sequential flow, and the second being the conditional flow, once again these relationships form the basis of the assessment decisional task flow and are expressed explicitly in the graphical notation as labelled connectors.

Table 3: Shows all the relationships and dependencies between the components of the care plan model.

Dependencies			
Component	**Name of dependency/relationship**	**Component depended upon or part of relationship**	**Explanation**
Care Plan	Contains	Care Plans Performance Metrics Activities Assessment Modules	A care plan contains (or is composed of) other care plans, performance metric, health activities, and assessment modules
	Depends on	Parent Care Plans	A care plan inherits the performance metrics of its parent care plan.
Performance Metric			
Activity	Is an abstraction of	Task Review Activity Data collection activity	An activity can be either a task, a review activity or a data collection activity.
	Contains	Instructions Routines Resources	An activity is composed of instructions, routines, and resources.
Review Activity	Contains	Review Activities	A review activity can be a composite of other activities.
Resource	Is an abstraction of	Service Resource Material Resource	A resource can be in the form of a service needed or a material need.
Instruction			
Routine			
Assessment Module	Contains		An assessment module is composed of a set of assessment components.
Assessment Component	Is an abstraction of	Condition Assessment Action Treatment Recommendation	An assessment component can be either a condition, an assessment action or a treatment recommendation.
	Sequentially flows to	Assessment Components	Each assessment component can flow to (sequentially) another assessment component (similar to a task flow diagram)
Condition	Conditionally flow to	Assessment Components	Each conditional component can flow to different assessment components depending on whether the condition is true or false.

10.3.8 Premature Commitment

Do programmers have to make decisions before they have the information they need?

As mentioned above one of the hard mental operations while using the visual care plan modelling language is the health care professional recalling large amounts of information in order to model complete health care plans, this taxing mental operation can be overcome in the visual care plan modelling language because care plans can be built gradually using the visual language, it is possible using the language to start building the care plan to cater for a select few groups of patients suffering from an ailment, and then gradually evolve the care plan to add in treatment procedures and regimes to cater for a wider patient group, hence this allows the health care professional to make decisions when they have the information they need or when they have to (may be when they start getting patients for whom the current care plan is not really good enough, they can start extending it to cater for the new type of speciality that has arisen).

10.3.9 Progressive Evaluation

Can a partially-complete program be executed to obtain feedback on "How am I doing"?

There is no real way that a care plan can be executed, however a care plan can always be instantiated and deployed to a patient, this instantiation and deployment can occur at any time in the care plan modelling process whether the care plan is of a large size or a small one as long as the care plan is syntactically complete according to the care plan meta-model. In addition to this due to the very nature of the modelling medium (i.e. a visual language) there is a continual visual feedback on the structure of the care plan as it is being built and modified.

10.3.10 Role Expressiveness

Can the reader see how each component of a program relates to the whole?

As expressed in the section above describing the dependencies within a care plan model, the relationships between the components in the model are expressed through graphical connectors labelled with the descriptors of the relationship, hence looking at a component it is possible to trace the relationships that the component shares with other components. However some relationships in the care plan model are hidden, for example when a care plan contains another care plan, this containment relationship is graphically visible however the graphical notation used is not rich enough to convey the added meaning to this relationship which describes that the performance metrics of the parent care plan also belong to the child care plan. The main reason for this is due to a timing constraint which did not allow the development of more sophisticated graphical notation. A similar lacking is seen when the containment relationship between review activities and sub review activities is specified. Other than these hidden dependencies, the relationships between the components in a care plan model are explicitly represented by graphical connectors.

10.3.11 Secondary notation

Can programmers use layout, colour, or other cues to convey extra meaning, above and beyond the official semantics of the language?

At this stage the care plan modelling language does not accommodate for a secondary notation to be used along side the main language syntax, however it is recognised that this may be an important feature, as health care professionals may want to mark and highlight important aspects of the care plan, or alternatively document their decisions for the design decisions taken while building the care plan, however this type of notation is out of the scope of the current work done.

10.3.12 Viscosity

How much effort is required to perform a single change?

The system being designed has a very low viscosity; any change that needs to be made to a care plan model is simply made by selecting a component in the visual language and editing its properties in the properties window.

Furthermore the fact that each component in the care plan is stand alone means that the user does not need to worry about reflection of changes. For example consider a care plan with one activity and a performance metric, in the event of the care plan being deleted, the activity and performance metric still remain. This does not prove a problem, because during the care plan instantiation process, only whole care plans will be instantiated, and leaving the activity and performance metric by themselves would be ok, because they could easily then be attached to another care plan.

One aspect of the care plan modelling language that may increase the viscosity is that each change made to a care plan may propagate a large number of manual rearrangements of the visual nodes in the diagram, however this is a direct product of the visual notation used and not the syntax.

10.3.13Visibility

Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side by side at will? If the code is dispersed, is it at least possible to know in which order to read it?

Due to the graphical notation that is in use for the care plan modelling visual language (i.e. the box and line notation) smaller care plans are easily visible in the canvas, however as the size and complexity of the care plan grows it becomes increasingly difficult to express the care plan in a single canvas, hence large parts of the care plan often flow off the page and need to be scrolled to, however the visual language still allows (under most reasonable circumstances) for components of a care plan model to be compared and viewed together with other components, once again a more intuitive and compact graphical notation can overcome this problem easily.

10.3.14Tradeoffs

What are the tradeoffs that may be there for improving on any one dimension for the visual language?

As with most visual languages the aspect that has the greatest effect on all the dimensions talked about above is the visual or graphical notation itself, improving the visual metaphor for the visual language means an improved graphical notation (or symbols), which in turn will act to improve most of the dimensions expressed above in one way or another. It will improve the visibility of the visual language, which in turn will make it easier for different parts of the care plan to be compared with each other, and hence this in turn will make it easier for care plans to be progressively evaluated. Another positive trade off that can be made includes adding of a secondary notation to the visual language, this in turn can enhance the value of the care plans themselves, as a secondary notation could be in the form of a notes or highlights explaining the design decisions in modelling the care plan and hence could not only improve the understandability of the care plan but also aid other health care professionals to extend the care plan.

10.4 Cognitive dimensions analysis of the visual patient application modeller

10.4.1 Abstraction Gradient

What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?

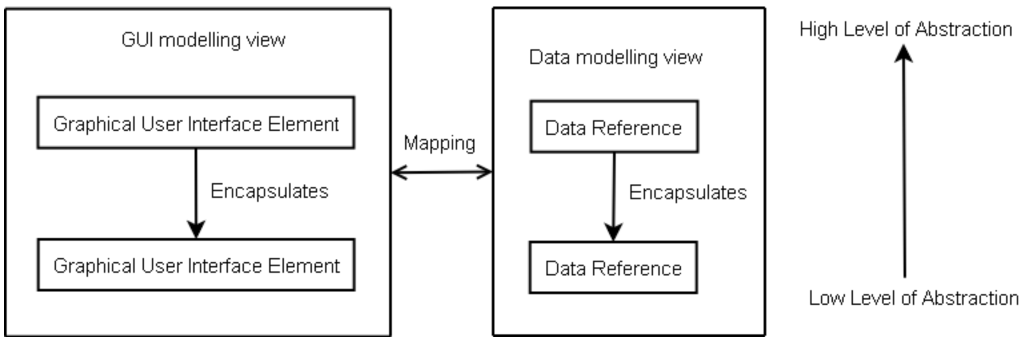


Figure 119: Diagram showing the level of abstraction for the visual patient application modeller.

The abstraction gradient for the visual patient application modeller can be seen in the figure 118 above. The VPAM consist of two different types of views, the first view is focussed on capturing information on the graphical user interface elements that make up the patient application, and the second view is concerned with modelling the care plan instance data that the patient will interact with through the GUI, and finally map the GUI elements to the data. It can be seen from the figure above that each view has just one level of abstraction, for the GUI modelling view, this abstractions amounts to one GUI element encapsulating other GUI elements, similarly for the data modelling view, one data reference can be composed of multiple other data references.

For the GUI modelling view this encapsulation relationship corresponds to a physical containment of one GUI element in another, for example, a view or a screen representing individual care plans that the patient is on, is expected to contain other GUI elements such as lists to represent the performance metrics that the patient is on, buttons to save and navigate to other pages etc.

For the data modelling view this one level encapsulation corresponds to yet again a physical containment relationship which can be used to model the structure of an XML care plan data sheet. As explained in the earlier chapters a data reference is essentially an XML node, it may be an XML element or an XML attribute, and using the containment relationship, the structure of XML elements for the care plan instance data can be modelled, i.e. specifying which elements are contained in which other parent elements and what their attributes are.

10.4.2 Closeness of Mapping

What “programming games” need to be learned?

Similar to the programming games that need to be learned about the visual programming language, the main programming game that needs to be learned about the visual patient application modeller includes learning the syntax of the modelling language. This learning process can be roughly split up into three main parts, firstly the user of the visual patient application modeller needs to learn how to model the GUI elements of the patient application, in particular learning what the look and feel of the different GUI elements are, what different interactions that are possible with the GUI elements and how the GUI elements can be laid out with respect to each other. Unfortunately with the current version of the visual patient application modeller this learning curve may be steep due to the visual notation in use not being very intuitive; however this learning curve can be substantially reduced once the graphical notation for the visual language is upgraded to a WYSIWYG interface (similar for example to the visual studio drag and drop GUI panel).

The second point of learning for the designer of the patient application is to learn how to model the data structure of a typical care plan instance XML file through the modeller using the data reference object. This involves the designer firstly learning what the structure of the care plan instance is and then learning how to specify XML elements and attributes, and how to structure the elements and attributes to model the structure of the care plan instance XML document.

The third point of learning for the designer is to get use to the idea of mapping GUI elements to XML elements of the modelled care plan instance XML structure, this can be quite challenging to a designer at the start because they need to combine the interactions possible with the GUI elements with the mapping of the GUI elements to the XML structure of the care plan instance in order to specify how the user can interact with a typical care plan instance. It is envisioned that this problem gradually subsides as the designer gets more experienced with using the system and with the syntax of the visual language, however the learning curve can once again be reduced with a more intuitive graphical notation for the visual language however due to time limitations this was out of the scope of the work done.

10.4.3 Consistency

When some of the language has been learnt, how much of the rest can be inferred?

From the above description it can be seen that there are three main things that a patient application designer needs to learn in order to use the VPAM effectively, i.e. the modelling of the GUI elements, the modelling of the data references and finally the mapping of the GUI elements to the data references. Once these three aspects have been learned then the designer can invent new ways in which to layout GUI components in the patient application, to expose different types of data in the patients care plan instance, through the patient application.

10.4.4 Diffuseness

How many symbols or graphic entities are required to express meaning?

As shown in the diagram above there are two main types of views for modelling the VPAM, the first is the GUI modelling view and the next is the data modelling view, therefore symbols or graphic entities are needed for representing the GUI components and the data reference components. In addition to this graphic entities or representations are also required for the relationships between GUI components and data references, as well as the containment relationships for both GUI components as well as data references. As explained in the design chapter (chapter 6), the graphical entities or symbols used for both types of components are colour coded boxes, with the relationships represented as appropriately directed lines.

10.4.5 Error-Proneness

Does the design of the notation induce “careless mistakes”?

One of the main flaws of the VPAM, is that it does not invest a lot of effort in preventing user error, user error can happen in any three aspects/stages of modelling the patient application, either when modelling the GUI of the patient application, or when modelling the care plan instance data using the data reference components, or when the user/designer is mapping the GUI to the data references.

While modelling the GUI for the patient application, the user may find it a bit tedious to specify each individual property of the GUI components, for example width, height, x and y coordinates etc, and due to this tedium make errors in the values. The main solution to this is to have better graphical symbols for the components, and make it so that the user can directly change the symbol to change the properties of the GUI component, for example to adjust the height and width of a GUI component graphically.

Other errors that are possible during the modelling of the GUI include, making mistakes in specifying the containment relationship. As can be seen in chapter 7 the containment relationship is specified through connecting GUI components by a line, the main form of error that can result from this is when a designer may try to specify an invalid containment relationship, for example a button containing a screen or a view. This type of error is likely to be caused by accident as GUI designers most likely have a good sense of how a graphical interface is to be put together.

10.4.6 Hard mental operations

Are there places where the user needs to resort to fingers or pencilled annotation to keep track of what is happening?

The hardest mental operation for the designer using the VPAM, involves visualising in their mind the graphical user interface that they are modelling using the VPAM. Due to timing constraint imposed on the implementation of the VPAM prototype, a richer graphical notation for the visual language was not possible. Therefore, it is likely that the designer has to use a pen and paper at least initially to plan the interface, or set of screens for the application. This planning may be most commonly in the form of a paper prototype, which would include rough sketches of all the views/screens that make up the application interface, along with annotations of what each component is for. It is important to realise however that a richer graphical notation for the VPAM, would allow a designer to perhaps bypass a paper prototype, for example if the GUI component in the VPAM could be modelled like the GUI modeller in visual studio, then planning an application screen through the VPAM would be as simple as dragging and dropping the actual GUI elements on the canvas and arranging the layout. However since the graphical notation of the VPAM does not give a designer a preview of the GUI being designed it is very probable that a designer may resort to paper prototyping the GUI before beginning to use the VPAM.

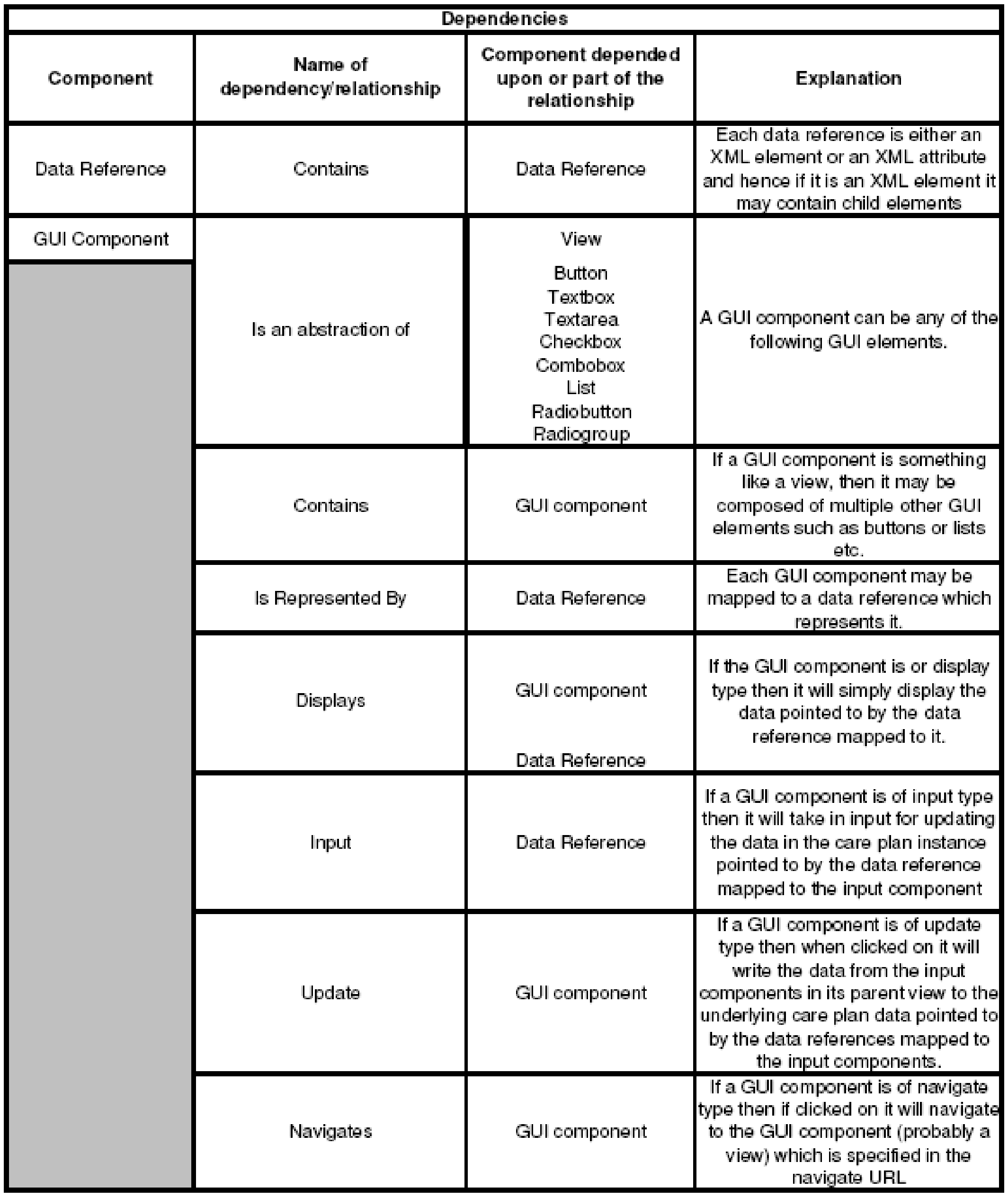
10.4.7 Hidden dependencies

Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?

The table shown in figure 120 below shows all the components and their respective dependencies on other components in the VPAM syntax. Out of all the relationships shown in the table in figure 120 below, there are two types of explicit relationships which are portrayed by the graphical notation of the VPAM. These two relationships include the containment relationship and the data to GUI representation relationship. The containment relationship can exist between either two data references or two GUI components. When it exist between two data references it signifies that an XML element has one or more child elements (this is because data references represent XML elements), and when the relationship exists between two GUI components, it means that one GUI component is composed of other GUI components, for example a view being composed of various buttons, lists, textboxes etc. The data to GUI representation relationship exist between a GUI component and a data reference, this relationship specifies which GUI component is to represent which data of the care plan instance, where the data itself is in XML form. Both of these relationships are explicitly represented in the graphical notation of the VPAM as directed, colour coded lines with labels connecting up the components concerned in the relationships.

What interactions that are possible with each GUI element, and indeed how the GUI element interacts with the data pointed to by the data reference that is mapped to it is largely dependent on the type of the GUI element, these types include the display, input, update and navigate type. These types make the GUI components dependent on the other components in the VPAM diagram; however they are hidden in that they are not expressed graphically, because the data to GUI representation relationship although describing that a GUI component will interact or represent some data does not really express in what type this interaction will be, for example a GUI component of display type will only display the data, well as a GUI component of input type will collect data to update the data which is mapped to it, well as a GUI component of navigate type, does not interact with the data at all, instead it just navigates to another GUI element when clicked on. This information although easily read off the properties window of the GUI component are not explicitly graphically represented in VPAM diagrams. Furthermore if a GUI component is of update type, it is never mapped to a data reference, it's function is to update the GUI components of input type that are in its parent view, this relationship is also hidden, there is no direct and obvious way that the designer can recognise this just by looking at a VPAM diagram for the first time, and it is one of the "programming games" that the designer must learn.

As mentioned above an improved graphical notation can solve all of these problems, however more sophisticated graphical notation could not be developed due to the time constraints imposed, instead time was spent in designing the actual syntax and meta-model which is the heart of the VPAM, with the improvements in the graphical notation being slotted as future work (see chapter 10 for more details).



Dependencies			
Component	**Name of dependency/relationship**	**Component depended upon or part of the relationship**	**Explanation**
Data Reference	Contains	Data Reference	Each data reference is either an XML element or an XML attribute and hence if it is an XML element it may contain child elements
GUI Component	Is an abstraction of	View Button Textbox Textarea Checkbox Combobox List Radiobutton Radiogroup	A GUI component can be any of the following GUI elements.
	Contains	GUI component	If a GUI component is something like a view, then it may be composed of multiple other GUI elements such as buttons or lists etc.
	Is Represented By	Data Reference	Each GUI component may be mapped to a data reference which represents it.
	Displays	GUI component Data Reference	If the GUI component is or display type then it will simply display the data pointed to by the data reference mapped to it.
	Input	Data Reference	If a GUI component is of input type then it will take in input for updating the data in the care plan instance pointed to by the data reference mapped to the input component
	Update	GUI component	If a GUI component is of update type then when clicked on it will write the data from the input components in its parent view to the underlying care plan data pointed to by the data references mapped to the input components.
	Navigates	GUI component	If a GUI component is of navigate type then if clicked on it will navigate to the GUI component (probably a view) which is specified in the navigate URL

Figure 120: The dependencies related to the VPAM modelling syntax.

10.4.8 Premature commitment

Do programmers have to make decisions before they have the information they need?

Due to the relatively primitive graphical notation in use for the VPAM, it is very well possible, that designers may need to be forced to plan the screens they are modelling, much before they start modelling using the VPAM, else they may be forced to make a premature commitment that due to the poor graphical notation, they would not detect.

10.4.9 Progressive Evaluation

Can a partially complete program be executed to obtain feedback on “How I am doing”?

As mentioned above the graphical notation used does not allow a designer to get a preview of the screens they are modelling while they are modelling, therefore in this direct sense there is no progressive evaluation possible through the usage of the VPAM modelling suite, however, if a designer wants to know how the screen will look, they can always pass the partially modelled VPAM diagram into the generator, and compile the screen to see the final outcome so far, although it is recognised that the code to preview cycle with this approach is long and is a substantial drawback for the VPAM. In the future if the VPAM is upgraded to have a more WYSIWYG modelling editor, then these problems will be easily solved, however such development was considered to be out of scope of the proof of concept prototype of the care plan modelling system and hence not implemented.

10.4.10 Role Expressiveness

Can the reader see how each component of a program relates to the whole?

In most circumstances the relationships between each component in the VPAM diagram can be observed, for example looking at a GUI component it is possible to trace which other GUI components are contained within it, or to determine what type of data the GUI component is suppose to represent, or what the structure of the data that the GUI is to interact with is. However as explained in the earlier sections there are hidden dependencies between the components of a VPAM model which mean that it may be unclear at times how a component relates to another component and indeed whether there is a relationship. Examples of such ambiguity mainly occur when specifying the interactions of the GUI component with relation to other GUI components and the data that is mapped to them. For example in a VPAM model with a view containing two textboxes and a save button, where the textboxes are of input type and the save button of update type, it may be unclear by simply looking at the save button component, what its relationship is with the textboxes and its parent view.

10.4.11 Secondary notation

Can programmers use layout, colour, or other cues to convey extra meaning, above and beyond the official semantics of the language?

At this stage the VPAM does not have any provisions for a secondary notation, all the properties of each component in a VPAM model, whether it be a data reference or a GUI element are entered and modified through a properties window. A secondary notation however which would be most welcomed in future improvements of the VPAM may be a way to attach documentation and notes to the GUI components while modelling, this aspect combined with an improved WYSIWYG editor could remove the need for paper prototyping before modelling, and considerably reduce the hard mental operation of designing the Gui screens using the VPAM.

10.4.12 Viscosity

How much effort is required to perform a single change?

Different types of changes require a different amount of effort, for example adding new GUI components to a VPAM model is very simple, as is adding new GUI elements to a composite GUI element, for example adding another button to a view, however one aspect of this task which may require some effort (i.e. the ripple of the change) is ensuring that the new component is laid out in the right place in the composite GUI element, the reason for this is that the order that GUI elements are laid out in their parent view or GUI element depends upon the order in which the containment relationship between the parent and the child is specified, so the new component added to a composite GUI element will always appear at the end of the GUI component, (e.g. the last element to appear in the view), to fix this all the containment relationships would need to be reordered. This shortcoming is again a shortcoming of the editing environment, and not the syntax of the language, i.e. the problem can be easily overcome with a WYSIWYG editor.

On the other hand if there is a view with two textboxes and a save button, then adding a new textbox to the view may involve adjusting the layout of the textbox so that it appears immediately after the two textboxes and not after the save button, however besides this there is no real effort that needs to be put into this task, as the save button whose relationship to the GUI components from which it saves data is implicit (i.e. it will save data from any input component in its parent view) the addition of the textbox does not really need any other modifications to the other GUI components in the view.

10.4.13 Visibility

Is every part of the code simultaneously visible (assuming a large enough display), or is it at least possible to compare any two parts side by side at will? If the code is dispersed, is it at least possible to know in which order to read it?

At this stage the visibility of the VPAM diagrams is average, in that different parts of the diagram can be compared side by side, however even small parts of the diagram take up a large amount of canvas area which means that on an average screen size the VPAM model of even a small/simple GUI flows over many screens, which may make it difficult for a user to comprehend the diagram quickly.

10.4.14 Tradeoffs

What are the tradeoffs that may be there for improving on any one dimension for the visual language?

There are many improvements that can be made to the VPAM, hence there are many possibilities of tradeoffs. The most obvious improvement, and arguably the most immediate needed for the VPAM is to improve the graphical notation of the visual language. The most probable improvement that can be made is to switch to a WYSIWYG editor for the language, where components such as GUI elements are expressed exactly as they are and properties such as location, layout and size can be set in a more natural manner, as well as this it is important for certain relationships which are hidden (such as the interaction specification between GUI components and other GUI components, and data references) to be made explicit. A trade-off for this is that it may slightly increase the viscosity of the language, as the more explicit the relationships are the more rippled changes need to be made by a simple modification. However this trade-off may be an acceptable one, as improving the graphical notation, will substantially reduce the hard mental operations of visualising the designed GUI through the VPAM, and also allow designers to progressively evaluate their design more easily. In addition to improving the graphical notation of the visual language, if an improvement is made to implement a secondary notation as explained above then it is possible to also reduce the need for a designer to pre design a GUI before using the VPAM, because the VPAM will, through the secondary notation allow a designer to create rough designs full with notes and documentation much like a paper prototype, which however can be used to generate the actual screens.

10.5 Informal Evaluation

10.5.1 Purpose and Audience

The purpose of the informal evaluation was to get an initial feedback on the work that is done in the development of the care plan modelling system. In general the evaluation survey was split up into three parts:

1. Evaluation of the care plan modelling visual language
2. Evaluation of the visual patient application modeller
3. Evaluation of overall approach

The first two sections of evaluation focused on providing users with modelling scenarios to analyse, in the case of the visual care plan modelling language these scenarios were with regard to modelling care plans using the language, and with the visual patient application modeller these scenarios corresponded to modelling screens for parts of the patient application. The users were asked to analyse and answer questions on their understanding of the models created in the language. The last section focussed on asking questions regarding the usefulness of the overall approach.

The informal evaluation was aimed at an audience which consisted of experienced visual language and software developers in order to get a perspective on both the quality of the two visual languages which were developed as well as the overall approach of the care plan modelling language. The focus of the evaluation questions were more towards how easy it was to understand the visual modelling language, and how the general quality of the visual language can be graded. There was however no time to conduct a practical evaluation of the care plan modelling language in action.

It is recognised that in order to improve the care plan modelling language, there is a need for real assessment with actual end-users however this was not possible for two main reasons, the first is that the care plan modelling language although implemented substantially towards completion could not be completed to a stage which can be displayed to end users (due to it being only a proof of concept prototype), and secondly there was also in sufficient time to conduct ethics approval for a formal evaluation to be conducted with end users.

The following sections present a brief summary of some of the important comments made by the expert software developers who participated in the survey (there were 3 software developers in total partaking in the survey), the survey itself can be found in Appendix A of this document.

10.5.2 Evaluation of prior knowledge in the domain

The first part of the informal evaluation included questions aimed at establishing what the participant's prior knowledge was in the domain of health care plan modelling, health care software in general and the concept of care plans. In general when participants were asked to define what a care plan was, all were able to define it as a strategy or group of strategies/steps etc to "manage" and "treat" the health of a patient suffering from some illness. In addition to this when asked to comment on what a care plan should be composed of all participants recognised that a care plan should contain some form of aim (or health goal) specifying what the illness or condition that is being treated or managed, tasks, activities or interventions in order to treat and manage the patient health, and ways of assessing and knowing when particular interventions or activities are effective and when they should be administered. The general result from these questions was in line with the structure which was adopted for the care plan domain model, which consisted of a hierarchy of performance goals, health activities and assessment modules. Furthermore the participants were also questioned whether they had heard or had prior knowledge of any health care modelling software which would allow modelling of health care information in a similar way to the care plan modelling language. In general most participants were not aware of the literature in this domain and hence could not quote of many such systems however one system which was quoted included Protégé, which although not directly a health care modelling suite, is knowledge management software which can be used to model health care knowledge ontologies, another health care system quoted included Predict, which is an evidence based medical support system.

10.5.3 Evaluation of the understandability of the care plan modelling visual language

This section of the informal evaluation was a scenario based survey in which the participants were provided with models at various degrees of detail of care plans modelled in the visual care plan modelling language in the form of screenshots and participants were asked to study the models and answer questions related to the model, in order to grade and evaluate how easy it is to understand care plans modelled in the visual care plan modelling language. As explained in various other chapters prior to this the visual care plan modelling language allows the modelling of two main models, the first is care plan models, and the second is assessment modules (which can be modelled as part of care plans or separately), therefore the evaluation of the understandability of the visual language is a combination of the ease of understanding of the two model types.

In general all participants were able to answer all the questions in this section successfully, showing that the care plan models as well as the assessment module models can be easily read and information can be easily extracted from it. After answering the questions when the participants were asked to grade the ease of understanding of the models made using the visual language from 1 being very hard to understand to 10 being very easy to understand, on average the ease of understanding of the care plans was given an 8 out of 10, the main comment made as the reason behind this rating was due to their being hidden dependencies, this comment was in line with the cognitive dimensions analysis of the visual language which discovered some hidden dependencies in the language.

Similarly for the assessment module model diagrams after studying the diagrams for assessment algorithms encoded in the visual care plan modelling language, and answering questions about it, it was found that all participants could answer all the questions and when asked to grade the ease of understanding of the models similar to the care plan models the average rating given was 6.5, the main comment being that the flow of components was sometimes hard to read or follow.

10.5.4 Evaluation of understandability of the visual patient application modeller

Similar to evaluating the ease of understanding of the visual care plan modelling language, this section of the survey aimed to get feedback on the ease of understanding of the visual patient application modeller, providing the participants with screenshots of actual models built using the visual patient application modeller and asking them to study the models and answer questions based on them. All participants could answer all questions in this section; however all found that the graphical notation was not adequate to the modelling task at hand, in that the graphical notation was quite “abstract”. When asked to grade how easy it was to grasp and understand the models of parts of the patient application encoded using the visual language from 1 being very in comprehensible and hard to understand to 10 being very easy to understand on average the rating given was 5.75. This rating was in line with the cognitive dimensions analysis done of the visual language, which suggested that the graphical notation used (box and line to represent all meta-model elements) would cause a great deal of hard mental operations while using as well as understanding the models, and a more WYSIWYG interface is necessary. In addition to this when users were asked to rate the overall concept of the modelling the patient application in the current version of the visual patient application modeller from1 being not very useful to 10 being very useful, on average the rating given was 6, and the main comment was that the notation was too close to the meta-model, and abstract, the need for better icons, graphical notations and more fitting visual metaphors was expressed, it can be seen once again that this result was in line with the cognitive dimensions analysis done of the visual language.

10.5.5 Evaluation of the overall concept of the Care Plan Modelling System

The aim of this section of the evaluation was to gain feedback on the views of the participants with regard to the overall approach behind the care plan modelling system, based on other systems that they had encountered and their overall experience in the software development industry. When participants were asked whether they had encountered similar model driven health care systems, most participants said that they had not come across an integrated system which combined the modelling of health care plans with the delivery of the same health care plans, with all participants agreeing that the underlying concept was a good one.

In addition to this when the participants were questioned about the advantages of having such a care plan modelling system to the health care professional they responded by saying it allowed “collaborative” and “proactive” health care modelling where patients and health care professionals can design health care strategies for the patient together, participants also suggested that the care plan modelling system would allow reuse of health care plans and would be useful in transferring some responsibility of care to the patient themselves.

These responses were well in line with the original motivations of developing the care plan modelling system. The participants were also questioned on how this sort of system would be advantageous to the patient, responses to these questions included the fact that it "passes responsibility" to the patient, by converting a care plan which is usually static, into something which is "operational", "executable" so that it can directly relate to the patient, keeping the patient connected to the care plan, giving helpful reminders, collecting data etc.

In addition to this when questioned about the advantages and disadvantages of using mobile devices as a deployment medium for operational care plans, the general response was that it is definitely a good idea, in that the care plans become portable, however, a great deal of care needs to be taken to ensure that whatever technology is used is fully tested and works without hitches. When participants were questioned about what they considered needed more attention or change, all participants suggested that the visual metaphors and the graphical notation used in the two visual languages i.e. the visual care plan modelling language as well as the visual patient application modeller, need quite a bit of work. Finally when the participants were asked to grade the overall approach of the care plan modelling system from 1 being not very useful, without potential, to 10 very useful, definitely has potential the rating was given as 7.3.

10.6 Summary

This chapter presents three items of evaluation conducted for the care plan modelling system developed as part of the completion of this Masters thesis, these three components of evaluation include a cognitive dimensions of the visual care plan modelling language, a cognitive dimensions analysis for the visual patient application modeller, and an informal evaluation of the two visual languages, as well as the overall concept of the care plan modelling system by an expert panel of software developers. The overall result which can be presented from the evaluation which was conducted includes the consensus that the overall approach of the care plan modelling system is a good one (7.3/10) as it provides ways in which to formally and collaboratively build health care plans, making the care plans operational so that they can be transferred to the patient, and that the approach provides a way in which responsibility of the health care can be delegated to the patient in a way which allows the patient to keep direct contact with the care plan. The main point of improvement that is suggested from the evaluation is to rethink and improve the visual and graphical metaphors/notation that is used for the two visual languages that are a part of the care plan modelling system in order to make the visual language easier to understand and easy to use.

Chapter 11 - Conclusions

11.1 Introduction

The purpose of this chapter is to discuss the contributions made by the Masters project, as well as discussing avenues for future work. The main contributions discussed in the chapter include the contributions of those made by the visual care plan modelling language, the visual patient application modeller and the overall model driven approach of the care plan modelling system. The future work for extending the research work done as part of this thesis includes future work for improving and extending the care plan modelling language, the visual patient application modeller, the overall approach and future evaluations to investigate where further improvement or changes in direction for the research done is needed.

11.2 Contributions from Research

The main contribution made by the research done for the completion of this Masters thesis includes the following:

1. The development of a high level health care plan modelling visual language
2. The development of a modelling language to design device specific patient applications to host personalised health care plans.
3. A model driven approach to health care plan modelling and distribution.

11.2.1 High level Care Plan Modelling Visual Language

The visual care plan modelling language is a very integral part of the overall approach of the care plan modelling system as it allows health care professionals to capture their knowledge on treating patients into formal care plans which can later be reused and collaborated on by multiple health care professionals. In addition to this the care plan modelling language also includes a care plan instantiation tool which allows health care professionals to fine tailor the modelled health care plans to particular patients before deploying the care plans to the patient, hence ensuring that each care plan is personalised to the patient for which it is being used, which in turn has been shown to improve patient adherence to their health care plan.

The syntax of the care plan modelling language unlike other health care modelling languages, can handle both modelling of information for decision support, such as treatment algorithms which health care professionals can use to decide on treatment procedures for the patient, and modelling actual actions for the patient to conduct in the form of executable care plans which can then be delivered to the patient through the generic patient application. In addition to this the visual care plan modelling language provides a far easier and perhaps a more natural way to model complex health care information through the visual interface, rather than using the typical wizard based approaches to modelling health care information.

11.2.2 Visual Patient Application Modeller

Solutions which aim at providing wellness management applications for the patient often suffer from being very device and platform dependent. This can be a serious barrier in the uptake of any wellness management solution, in that a application designed to run on a PDA will not run on a mobile phone, or a smart phone. In addition to this an application that runs on a Windows CE OS will not run on a Palm OS, and so on.

The visual patient application modeller was created to allow a GUI designer to design the look and feel of the patient application for different device types and then generating the patient applications based on these definitions in Flash. Using the modeller the designer can decide to split up screens in the patient application so that it can fit on smaller displays, or alternatively change the layout of the GUI components to suit a type of device, or even use particular types of GUI elements which would be most convenient on the device of choice. Through this approach models of the patient application can be generated for multiple common devices and then stored to be used to generate the patient application for the different devices when needed.

11.2.3 Model driven approach to Care Plan Modelling

As mentioned in the background research chapter, there are two main types of health care management applications which have currently been developed and are in use, these include the frameworks and applications which allow the modelling of health care information for decision support for the health care professional, and then there are applications in the form of digital wellness management applications which aim to provide the patients support for managing their health care. The health care information modelling applications for the health care professional, although allowing the health care professional to model treatment guidelines to reuse and provide decision support lacked the facility to transform this information into a patient specific context and then deliver it to them. Well as the various wellness management applications although providing patients with support, had a weak connection with the health care professional, with the health care professional having little input. In addition to this these wellness management applications were largely disease, device and platform specific.

Hence to combat this short coming we designed a model driven approach to health care planning, which combined both aspects of health care management, i.e. an integrated system which allows health care professionals to model not only decision support information such as treatment algorithms but also patient specific health care information in the form of formal care plan templates and instances. Furthermore the system allows the modelled health care information to be delivered to the patient in the form of a disease independent generic patient application. In addition to this the system also includes a patient application modeller, through which a GUI designer can model the look and feel of the patient application for different devices, and use these models to actually generate the patient applications in Flash.

The approach of an integrated model driven care plan modelling system is perhaps the most important contribution of the research work done for this Masters thesis. The care plan modelling system introduces an approach to not only formally model health care information, but to fine tailor it to patients and to deliver it to patients for active use in a device independent way, and we feel that this integrated approach in the health care plan modelling domain is unique.

11.3 Future Work

There is a large array of future work avenues which arise from the work done as part of this thesis work, both work to improve the current care plan modelling system which is designed as well as work which arises from the concepts and approach of the care plan modelling system. This section lists some of the more important avenues of future work that can be carried out regarding the care plan modelling system, the possible future work is split up into three main categories the first is the visual care plan modelling language, the second is the visual patient application modeller, the third is future work regarding the overall approach of the care plan modelling system.

Future work recommended for the visual care plan modelling language:

1. Improving the visual/graphical notation for the visual care plan modelling language, to make it more convenient, and to use a visual metaphor which may be closer map to the domain and more intuitive to the health care professional to use.

2. Adding a secondary notation for the visual care plan modelling language, so that health care professionals can add the design decisions and notes regarding their modelling choices in building the health care plan, including being able to things like highlight items of the care plan and mark out things of importance, hence enhancing the understandability of the health care plan, making it easy to reuse and collaborate on.

3. The syntax of the care plan modelling language could be extended to add a design critiquing tool, which would allow health care professionals to encode best practices for modelling health care plans, and which would in turn give the health care professionals real time reminders and alerts of these best practices and ensure that collaboration and extensions of the care plan happen to a fixed set of design practices.

4. Improving the care plan instantiation tool, to include more personalisation features for the patients, this includes investigating and implementing more ways in which health care plans can be personalised to the patient.

Future work recommended for the visual patient application modeller:

1. Improve the visual/graphical notation for the visual patient application modeller, so that it is easier to use, replace the box and line diagramming notation for modelling GUIs with a WYSIWYG visual editor so that GUIs for the patient application can be easily modelled.

2. Improve the patient application generator to enhance it with automatic GUI layout algorithms so that once a patient application definition (from the visual patient application modeller) is passed to it the designer of the patient application can simply specify the type of device the patient application is meant for and the algorithms will automatically correct the layout of the GUIs to match the screen availability of the device, such algorithms are readily available and systems demonstrating them have been built and hence this is not regarded as a very taxing task however it was out of the scope of this work.

3. A design critiquing tool similar to that recommended for the visual care plan modelling language could be implemented for the visual care plan modeller as well, hence the designer of the patient application can record the best practices for designing applications for the patient.

4. Extend the patient application generator so that it produces patient applications which allow the patient to personalise the GUI of the application according to their own personal preferences, for example allow patients

to choose how they want the list of care plans on their application main screen to be represented, for example they may choose a combobox representation instead of a listbox representation

5. Implement patient application generators for languages other than OpenLaszlo, for example to generate code from the patient application for a .Net compact PDA application.

Future work recommended for the overall approach of the care plan modelling system:

1. Extend the current approach of the care plan modelling language to include ways in which health care data that is collected from the patient through the patient application and the patient care plan instance can be packaged, formatted and integrated into other legacy medical systems such as Medtech which are in the mainstream use by health care professional stakeholders.

Future evaluations recommended:

1. Evaluate the visual care plan modelling language and meta-model with actual health care professionals to investigate ways in which the meta model (and hence the language) as well as the visual representation can be improved or made more suitable for actual use.

2. Trial the visual language with different types of health care professionals and hence determine whether the health care plan modelling language is really suitable for a wide variety of health care professionals or just a select few (i.e. GPs).

3. Evaluate the visual patient application designer, to evaluate how useful the patient application modeller is designing the patient application, investigating how the modelling suite and the language itself can be improved.

4. Investigate with patients how useful the idea of carrying a personal health care management application is. Investigate with the patients how they feel about the whole approach of the care plan modelling system (i.e. is it necessary?).

5. Investigate with health care professionals with the overall usefulness of the whole approach, evaluating the viability of the whole approach in real practice.

11.4 Summary

There are three main contributions made by the care plan modelling system designed and prototyped as part of the research work done for the completion of this Masters thesis, this includes the design of a care plan modelling visual language which can be used for the design and developing of high level health care plans for the management and treatment of patients suffering from long term illnesses, which can later be reused and collaborated on by multiple health care professionals. The second contributions is the visual patient application modeller which can be used to design and model device specific definitions of the patient application hence making it possible to design and generate the patient application for the patient automatically for different devices that the patient may possess. The third and final contribution made by the research work is the concept of a model driven approach to care plan modelling, which combines the aspects of modelling care plan information formally for reuse and collaboration and then packaging and deploying this information to the patient in device independent way. The future work recommended to carry on the research work and improve it can be split up into four parts, the first part is concerning the visual care plan modelling language, concerning firstly improving the visual metaphors for the language to make the language more usable, along with adding secondary notations to the language to allow more flexibility in recording design decisions and finally adding a way to provide the health care professional with ways of recording and using design critics to allow modelling of care plans according to a set of best practices. The second sector of future work recommended was for the visual patient application modeller, the two most important recommendations include improving the visual metaphor (or graphical notation) of the visual language to make it a WYSIWYG editor, the second recommendation being to incorporate automatic layout algorithms according to different screen types in the patient application generator. Besides this, similar to the care plan modelling language, it is also recommended that the patient application modeller implement a design critiquing tool as well as make avenues for patients to self specify their personal preferences on the look and feel of the patient application GUI through the generated application itself. The third sector of future work includes work related to the overall model driven approach that is behind the care plan modelling system, and under this it is recommended that the care plan modelling system be extended to incorporate methods of collecting and packaging data from the patient application (and patient care plan instance) to make it available to third party health care management systems that are currently in legacy use for example Medtech. The last sector of future work recommended includes different avenues of evaluation that should be done, including evaluating the usability and usefulness of the two languages (i.e. the care plan modelling language, and the visual patient application modeller), in addition to this it is also recommended that the overall approach and its usefulness be evaluated.

References

Adobe. (2007). "Macromedia Flash Support Center: Using ActionScript, Using local shared objects in Macromedia Flash MX." Retrieved September 8th, 2007, from http://www.adobe.com/support/flash/action_scripts/local_shared_object/.

Bartlett, J. A. (2002). "Addressing the challenges of adherence." Journal of Acquired Immune Deficiency Syndromes **29**: S2-S10.

Beyer, M., K. A. Kuhn, et al. (2004). Towards a Flexible, ProcessOriented IT Architecture for an Integrated Healthcare Network. 2004 ACM Symposium on Applied Computing. Nicosia, Cyprus.

Boxwala, A. A., M. Peleg, et al. (2004). "GLIF3: A representation formatfor sharable computer-interpretable clinical practice guidelines." Journal of Biomedical Informatics **37**: 147-161.

Fox, J., N. Johns, et al. (1998). "Disseminating medical knowledge: the PROforma approach." Artificial intelligence in Medicine **14**: 157-181.

Funnel, M. M., R. M. Anderson, et al. (2000). "The problem with compliance in diabetes." JAMA **284**: 1709.

Grundy, J. and B. Yang (2003). An environment for developing adaptive, multi-device user interfaces. Fourth Australasian User Interface Conference. Adelaide, Australia, Australian Computer Society, Inc.

He, J. and I.-L. Yen (2007). Adaptive User Interface Generation for Web Services. IEEE International Conference on e-Business Engineering, IEEE Computer Society.

Hosking, J. (2007). Softeng 450 Lecture 3: Visual Languages/Notations**:** 1-22.

Huber, B., P. Votruba, et al. (2007). AsbruView.

Kanade, S. (2006). A Personal Health Management Application. Proceedings of 2006 Year 4 Research Projects, Auckland, New Zealand.

Khambati, A. (2006). A Personal Health Management Application. Proceedings of 2006 Year 4 Research Projects, Auckland, New Zealand.

Kollmann, A., D. Hayn, et al. (2006). "Mobile phones as user interfaces in the management of chronic diseases." e & i Elektrotechnik und Informationstechnik **123**(4): 121-123.

Laszlo Systems Inc. (2007). "Software Engineer's Guide to Developing OpenLaszlo Applications." Retrieved March 20th, 2007, from http://labs.openlaszlo.org/wafflecone-nightly/docs/developers/.

Lee, V., H. Schneider, et al. (2004). Mobile Applications: Architecture, Design and Development, Pearson Education.

Liu, K. (2007). "Marama Meta-Tools User Manual." 2007, from http://www.cs.auckland.ac.nz/Nikau/marama/tutorial/MaramaMetaTools-UserManual.html.

MetaCase. (2007). "Domain Specific Modelling with MetaEdit+." 2007, from http://www.metacase.com/.

Microsoft. (2007). "Domain-Specific Language Tools." 2007, from http://msdn2.microsoft.com/en-us/library/bb126235(VS.80).aspx.

Miksch, S., Y. Shahar, et al. (1997). Asbru: A task specific, intention based, and time oriented language for representing skeletal plans. 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97). Milton Keynes, UK.

Mitrovic, N. and E. Mena (2002). Adaptive User Interface for Mobile Devices. Interactive Systems. Design, Specification, and Verification. 9th International Workshop. Rostock (Germany).

MUPE. (2007). "LearnMupe." Retrieved June 7th, 2007, from http://mupe.nrln.net/wiki/index.php/LearnMUPE.

National Institute Of Health, N. H., Blood and Lung Institute (1998). Clinical Guidelines On the Identification, Evaluation, And Treatment Of Overweight And Obesity In Adults**:** 228.

Nuschke, P., T. Holmes, et al. (2006). My Health, My Life: A Web-Based Health Monitoring Application. Human Factors in Computer Systems. Montreal, Quebec, Canada, ACM.

Open Clinical. (2007). "AsbruView." Retrieved May 12th, 2007, from http://www.openclinical.com/dld_asbruview.html.

OpenClinical. (2007, 19th September, 2007). "Open Clinical: Kowledge management for medical care - Guideline modelling methods and technologies - GLIF." 2007, from http://www.openclinical.org/gmm_glif.html.

OpenClinical. (2007, 8th January, 2007). "Open Clinical: Kowledge management for medical care - Guideline modelling methods and technologies - Proforma." 2007, from http://www.openclinical.org/gmm_proforma.html.

OpenLaszlo. (2007). "OpenLaszlo: the premier open source platform for rich internet applications." Retrieved March 20th, 2007, from http://www.openlaszlo.org/.

Parkka, J., M. V. Gils, et al. (2000). A Wireless Wellness Monitor For Personal Weight Management. Information Technology Applications in Biomedicine.

Pratt, W., K. Unruh, et al. (2006). "Personal Health Infomation Mangement." Communications of the ACM **49**(1): 51-55.

Program, N. C. E. (2002). Detection, Evaluation, and Treatment of High Blood Cholesterol in Adults (Adult Treatment Panel III).

Schimoda, T. (2002). An Interactive Software-Agent Smoking Cessation Program. 36th Hawaii International Conference on System Sciences. Hawaii.

Stanford Center for Biomedical Informatics Research. (2007). "Protege." Retrieved August 20th, 2007, from http://protege.stanford.edu/.

TechRepublic. (2007). "Creating powerful Flash applications with shared objects." Retrieved September 8th, 2007, from http://articles.techrepublic.com.com/5100-22-5061241.html.

Tsai, C. c., G. Lee, et al. (2007). "Usability and Feasability of PmEB: A mobile phone applicaiton for monitoring real time caloric balance." Mobile Network Applications **12**: 173-184.

Turner, B. J. and F. M. Hecht (2001). "Improving on a coin toss to predict patient adherence to medications." Annals of Internal Medicine **134**(10): 1004-1006.

University of Maryland. (2007). "LifeLines for Visualising Patient Records." 2007, from http://www.cs.umd.edu/hcil/lifelines/.

W3CSchools. (2007). "XForms Tutorial." Retrieved May 2nd, 2007, from http://www.w3schools.com/xforms/default.asp.

XML4Pharma. (2007). "XForms Browsers for PCs, mobile phones and PDAs." Retrieved August 15th, 2007, from http://www.xml4pharma.com/ODMinEDC/browsers.html.

Appendix A – Informal Evaluation Script

A1.0 Background

The care plan modelling system was developed to allow health care professionals to model health care plans for patient's suffering from chronic illnesses and then deploy the health care plan onto multiple different types of mobile devices as flash applications. The care plan modelling system is made up of four main components, the first is a visual language designed to allow the modelling of health care plans, the second component is an instantiation tool through which the care plans designed through the visual care plan modelling language can be fine tuned/tailored to the patient. The third component of the system is a visual patient application modeller, through which a designer can model how the care plan will graphically look to the patient on different types of mobile devices in a patient application hosting the care plan. The fourth component is a patient application generator which generates a patient application according to the patient application models, with the generated application being compiled to flash. The general cycle of care plan design and deployment that the care plan modelling system embodies is shown in the diagram below.

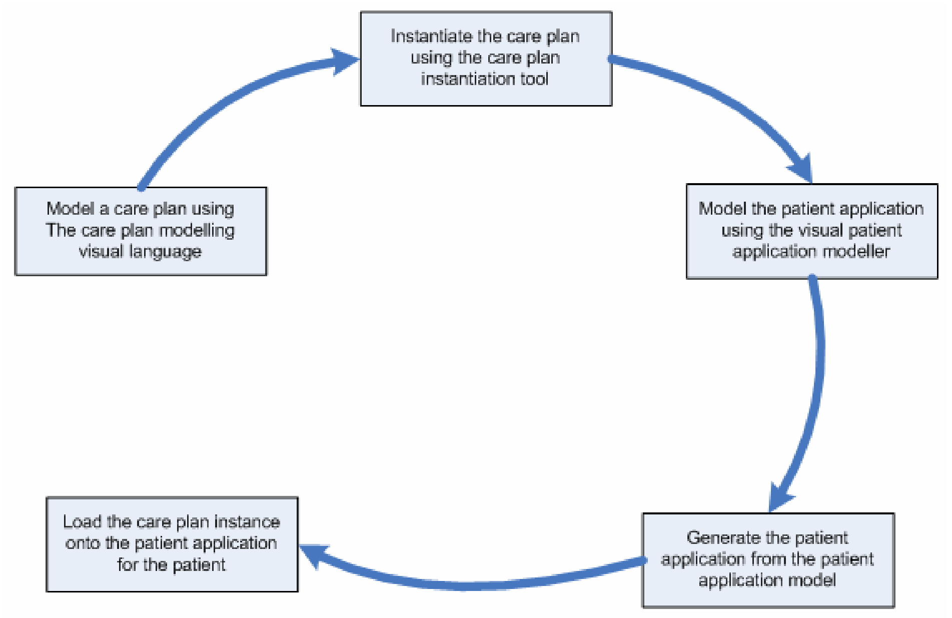

Instantiate the care plan using the care plan instantiation tool
Model a care plan using The care plan modelling visual language
Model the patient application using the visual patient application modeller
Load the care plan instance onto the patient application for the patient
Generate the patient application from the patient application model

Figure 121: The cycle embodied by the care plan modelling system, i.e. modelling care plans, tailoring care plans specifically to patients, modelling the patient application, generating the patient application, and loading and executing the care plan.

A1.2 Purpose of Informal Evaluation

The purpose of this informal evaluation is to get an initial feedback on the work that is done in the development of the care plan modelling system. In general the evaluation survey is split up into three parts:

4. Evaluation of the care plan modelling visual language
5. Evaluation of the visual patient application modeller
6. Evaluation of overall approach

The informal evaluation of the visual care plan modelling language and the visual patient application modeller, is ideally meant to be done by people who are experienced in the development of domain specific visual languages, and the purpose of the evaluation is to get a general idea on the "quality of the languages". It is recognised that actual evaluation with actual end-users is still a vital part of designing an efficient domain specific language, this however is out of the scope of the project.

The informal evaluation of the overall approach will aim, at gathering again from a panel of experienced developers, a general opinion, on the effectiveness of the model driven approach of designing, modelling and deploying health care plans. Other motivations behind evaluating the overall approach include getting opinions on the effectiveness of ideas such as personalised health care plans, and mobile application based health care plans.

A1.3 Evaluating the Visual Care Plan Modelling Language (VCPML)

A1.3.1 Background

The care plan modelling visual language was designed to allow health care professionals to design and reuse health care plans for patient's suffering from chronic illnesses. Typically these health care plans include treatment and management strategies such as diabetes management, obesity management, smoking cessation etc. There are many formal structures that have been designed to capture the structure of the care plan, however the one used for the purpose of the visual care plan modelling language can be seen in figure 122 below. It can be seen that the health care plan is generally made up of performance goals/metrics, activities to achieve these performance goals, assessment modules describing how patients on specific care plans should be assessed, and finally other sub-care plans.

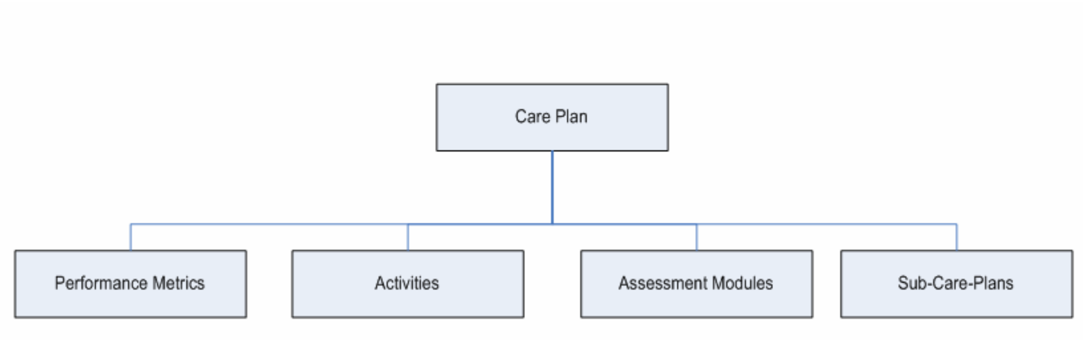


Figure 122: The structure of a care plan.

A1.3.2 Prior knowledge of care plan modelling

The purpose of this section is to evaluate the level of previous knowledge of health care plan modelling that participants in this evaluation may have.

1. How would you define a health care plan?
2. Have you heard of or used other visual care plan modelling languages? If yes:
    a. Try and state them
    b. What was their idea of a care plan?
    c. Did the care plan modelling visual languages/systems have any way to deliver the prepared health information to the patient? If so what were they?
3. What would you feel a health care plan should be composed of? Try to state the parts that you feel a health care plan should definitely have.

A1.3.3 Ease of understanding

The purpose of this section of evaluation is to get opinions on how well already modelled care plan diagrams can be understood.

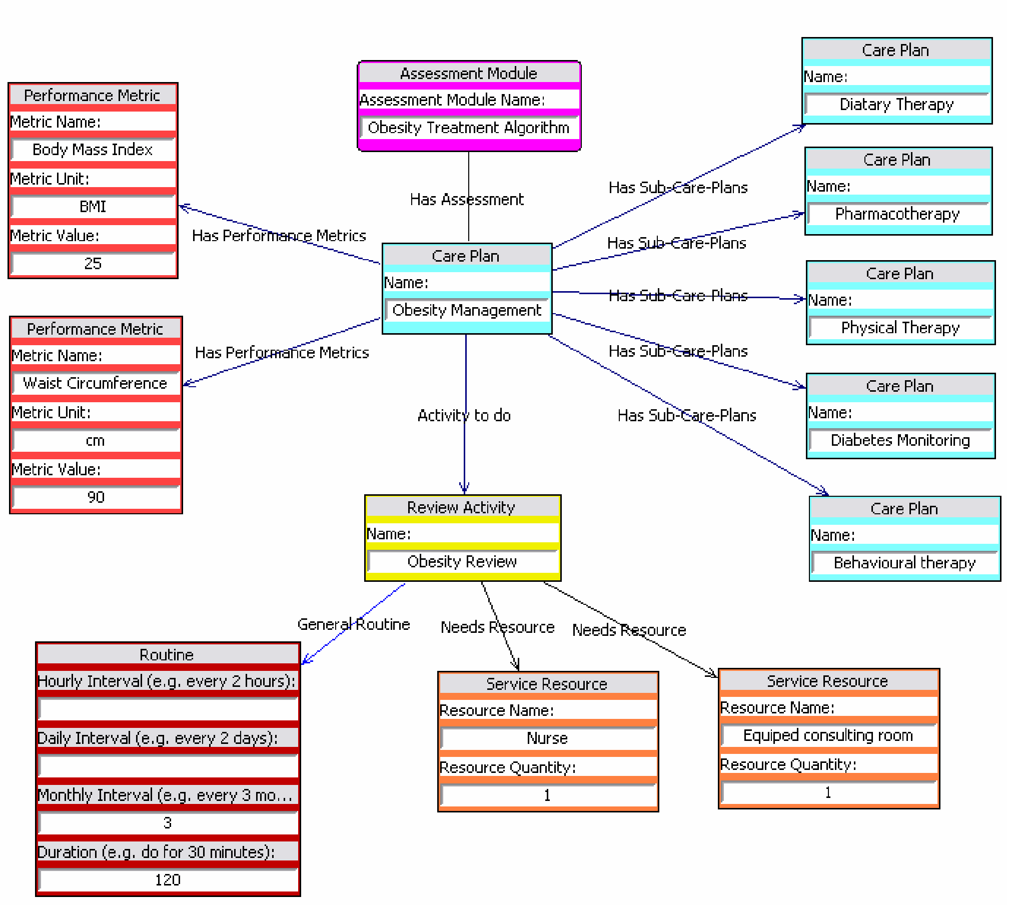


Figure 123: A sample obesity management care plan.

The following questions/tasks are based on the diagram above.

1. To what type of patient would the care plan shown in figure 123 be useful to?

2. What are the different parts of treatment (or sub care plans) prescribed as part of the main obesity management care plan?

3. What are the performance goals that the patient ideally should attain as part of the above care plan?

4. The main obesity care plan above has one activity,

   a. Name this activity?
   b. What are the resources that are needed to do this activity?
   c. How often is this activity to be done and for how long?

5. Can you describe the entire care plan in plain words?

6. Can you rate the overall understandability of the care plan above between 1→10, 1 being the lowest (in comprehensible) to 10 (being very clear easy to understand)?

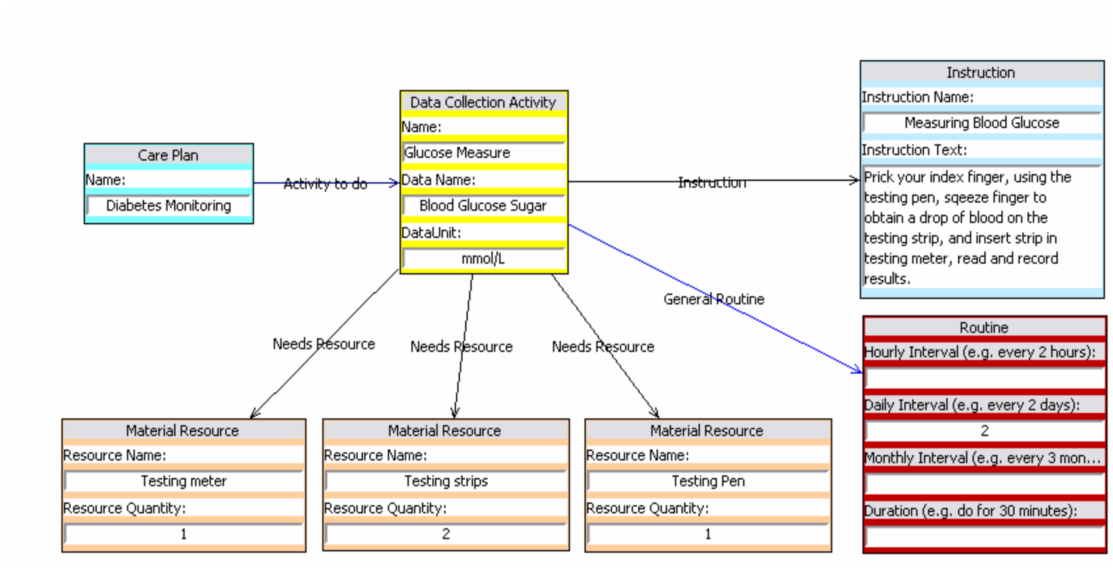


Figure 124: A sample diabetes monitoring care plan.

1. Figure 124 shows an activity prescribed to a patient under the diabetes monitoring care plan prescribing the collection of some health data:
    a. Can you understand exactly what the patient is asked to collect as part of this activity?
    b. Can you list the resources that a patient will need before doing this activity?
    c. Can describe the procedure of how the activity can be conducted?
    d. Can you describe how often the activity needs to be conducted?
    e. Grade how easy it was to understand the care plan shown in figure 124 from 1 (being the lowest, completely incomprehensible) to 10 (being very easy to understand, clear).

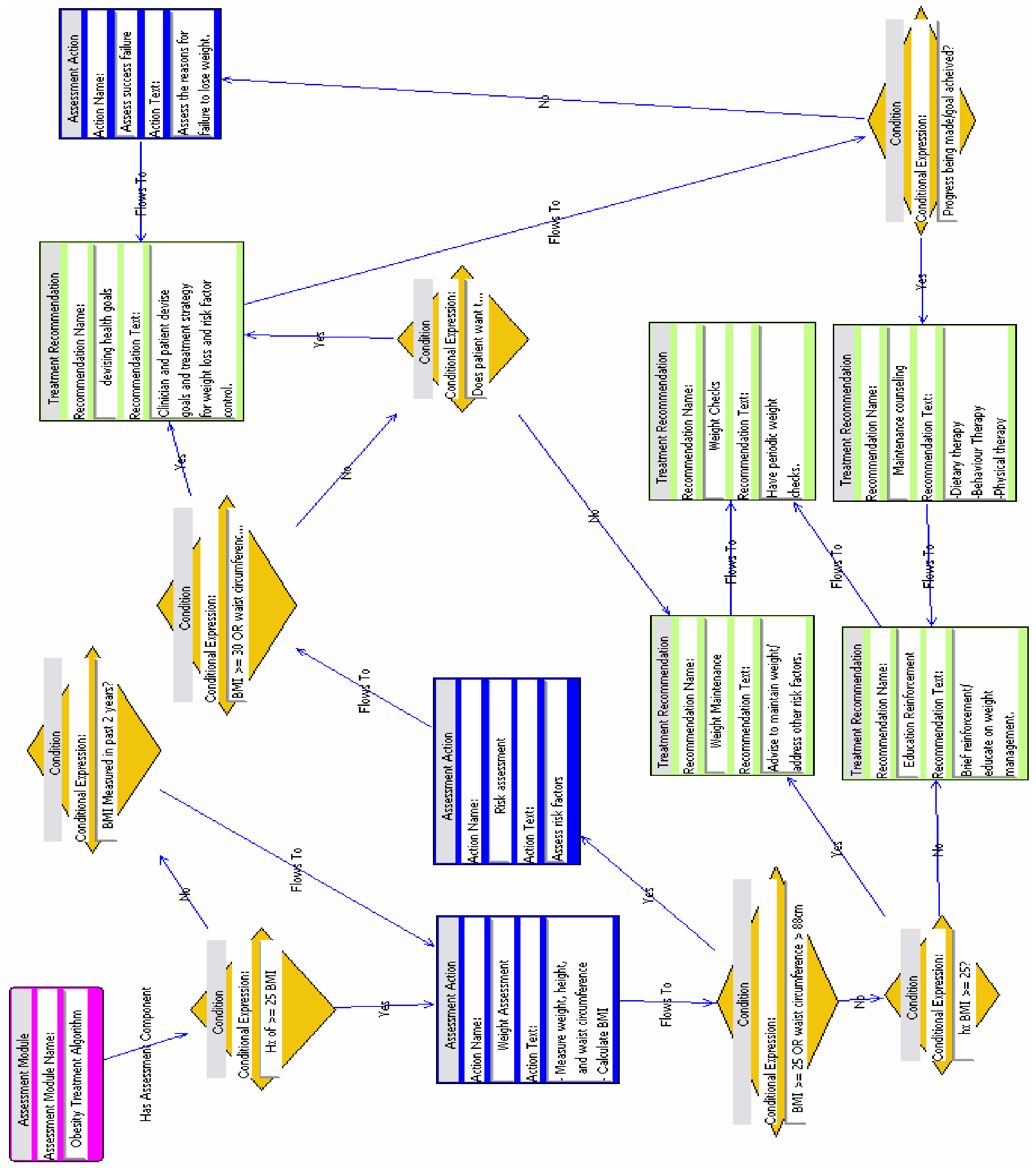


Figure 125: The assessment algorithm for assessing a patient with obesity

Each health care plan can also be composed of one or more assessment modules, which guides the health care professional on how a patient on a particular health care plan can be assessed, as well as what the appropriate treatment recommendations are, based on their health condition. The following questions are to gather opinion on the easy of understanding of a treatment algorithm modelled using the VCPML for assessing patients suffering from chronic illnesses.

Starting from the top of the assessment algorithm,

1. What assessment actions are recommended if the patient has a body mass index (BMI) of greater than 25?

2. What treatment recommendation is given if the patient has a BMI of greater than 30?

3. If the patient achieves some of their weight loss goals, what treatment recommendation is given? (Hint: Look at the condition near the bottom right of the diagram)

4. Grade the ease of understanding of the assessment algorithm from 1 (being most incomprehensible) to 10 (being very easy to understand, clear).

A1.4 Evaluating the Visual Patient Application Modeller (VPAM)

Once the a health care plan has been designed and personalised to the patient, then the next step is to model how the care plan instance will appear graphically to the patient on their patient application modeller, the purpose of this part of the evaluation is to get feedback on the visual patient application modeller through which the modelling of the patient application can be done.

A1.4.1 Background – Using the VPAM

The purpose behind the visual patient application modeller, is to provide a way in which designers can define how an instance of a care plan intended to be deployed to a patient, will graphically look to the patient in the form of a mobile device application, and since there are many mobile devices that are out there each probably with different screen sizes GUI capabilities, etc, the VPAM could be used to develop multiple patient application definitions viable for different types of mobile devices. The VPAM modelling suite relies on the concept of mapping the structure of the XML care plan instance to GUI elements that will represent the XML care plan instance data.

The VPAM language/syntax is essentially made up of two different types of components, the first type is called a data reference component, the purpose of this component is to allow a designer to model the XML data structure of the care plan instance data. The second type of component is the GUI component this component includes a subset of GUI elements, such as labels, views, textboxes, comboboxes etc. The VPAM allows a designer to map GUI elements to data references in order to specify how the XML care plan data is to be graphically represented by the patient application. Depending on the type of interaction that the GUI is meant to have with the data that it is mapped to, each GUI element may be of one of the four types:

1. Display type – Components that simply display data that is mapped to them, e.g. labels, static list etc.
2. Navigate type – Components that when clicked on navigate to another page, e.g. list (list items), buttons, etc.
3. Input type – Components that take in input from the user in order to update the data which is mapped to them, e.g. textboxes, textareas, radio buttons etc.
4. Update type – Components that when clicked on take the data from input components and write it to the underlying care plan instance data, e.g. save buttons.

Once this definition is modelled then it can be used to generate this screen in Flash. In a similar fashion properties of the care plan instance data can be mapped to their respective GUI representations allowing the generation of a patient application.

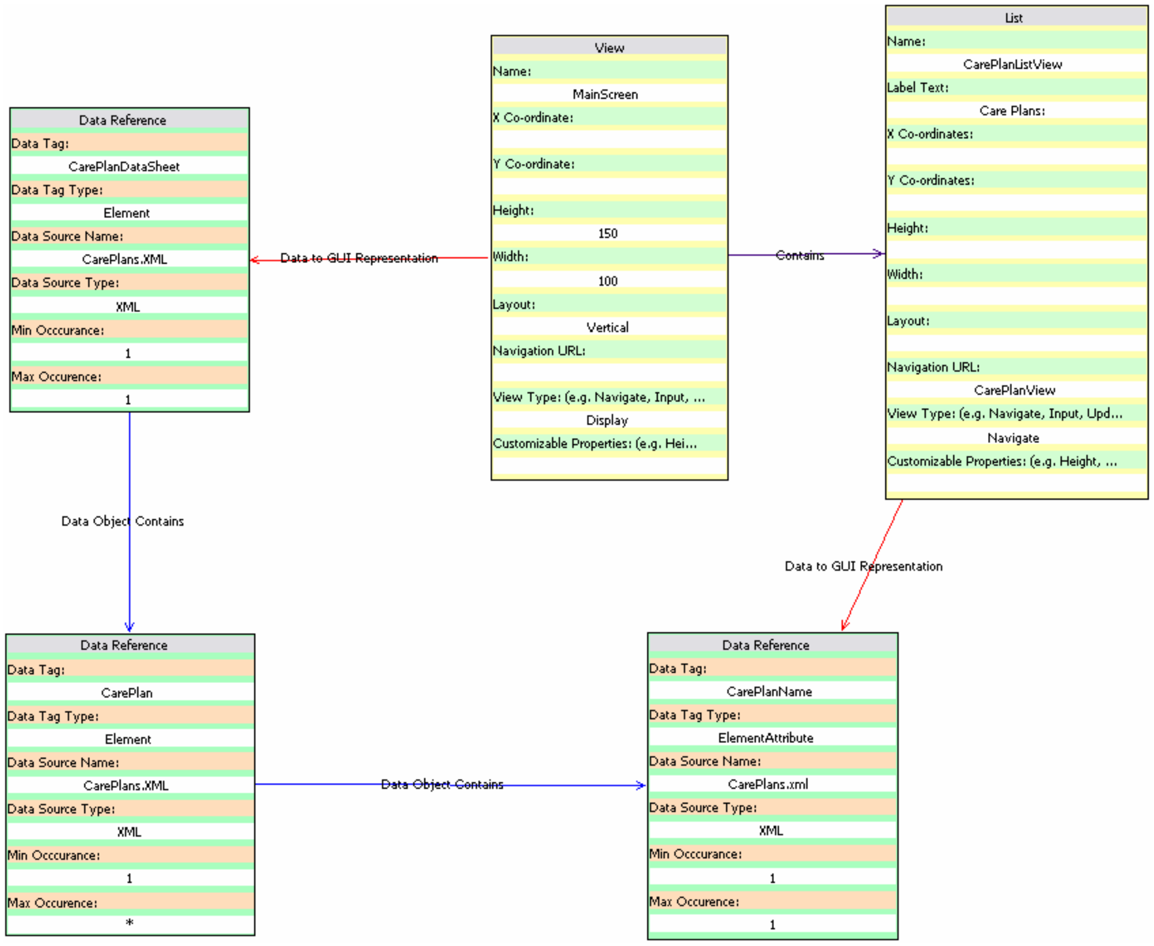
A1.4.2 Ease of Understanding

The purpose of this section is to gather opinion on the ease of understanding of the language and syntax behind the visual patient application modeller, as well as to get feedback on the concept of modelling reusable application definitions.

A1.4.2.1 Questions/Tasks

A1.4.2.1.1 Question set 1

This set of questions concerns the building of a model for the main screen for the patient application. The purpose of the main screen in the patient application is to show the patient all the care plans that they are on. Figure 126 below shows the VPAM model of the main screen of the patient application, with figure 127 showing care plan data sheet for a patient, it can be seen from this that the patient is on 4 care plans, the main screen will simply show this. The questions in this section are regarding the ease of understanding of the VPAM model for the main screen of the patient application. Please answer the questions following figures 126 and 127.



Data Reference
Data Tag: CarePlanDataSheet
Data Tag Type: Element
Data Source Name: CarePlans.XML
Data Source Type: XML
Min Occcurance: 1
Max Occurence: 1

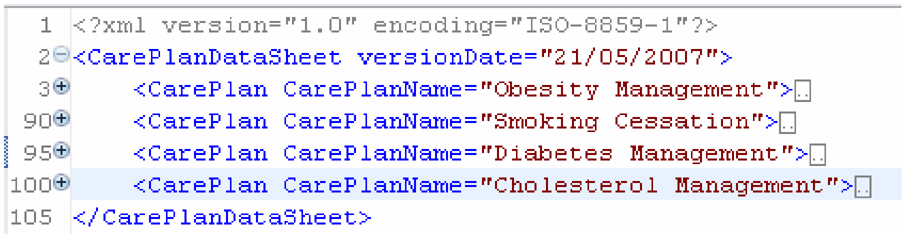
View
Name: MainScreen
X Co-ordinate:
Y Co-ordinate:
Height: 150
Width: 100
Layout: Vertical
Navigation URL:
View Type: (e.g. Navigate, Input, ... Display
Customizable Properties: (e.g. Hei...

List
Name: CarePlanListView
Label Text: Care Plans:
X Co-ordinates:
Y Co-ordinates:
Height:
Width:
Layout:
Navigation URL: CarePlanView
View Type: (e.g. Navigate, Input, Upd... Navigate
Customizable Properties: (e.g. Height, ...

Data Reference
Data Tag: CarePlan
Data Tag Type: Element
Data Source Name: CarePlans.XML
Data Source Type: XML
Min Occcurance: 1
Max Occurence: *

Data Reference
Data Tag: CarePlanName
Data Tag Type: ElementAttribute
Data Source Name: CarePlans.xml
Data Source Type: XML
Min Occcurance: 1
Max Occurence: 1

Figure 126: The VPAM model of the main screen for the patient application.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CarePlanDataSheet versionDate="21/05/2007">
    <CarePlan CarePlanName="Obesity Management">
    <CarePlan CarePlanName="Smoking Cessation">
    <CarePlan CarePlanName="Diabetes Management">
    <CarePlan CarePlanName="Cholesterol Management">
</CarePlanDataSheet>
```

Figure 127: The reduced structure of the care plan data sheet.

1. Try and describe in words the data structure that is portrayed by the data references in figure 126.

2. Does the data structure efficiently describe the structure that can be observed in the reduced XML document in figure 127?

3. Reading the model can you tell the name of the GUI component that will represent the entire care plan data sheet? (Hint: which GUI component is mapped to the Care Plan Data Sheet?)

4. Can you detect a GUI component that is a composite of other GUI components (i.e. is made up of)?

5. How are the many care plans that make up a care plan data sheet represented in the GUI that is modelled in the VPAM model in figure 126? (Hint: What is the GUI component that is used to represent the care plans of a care plan data sheet to the patient?)
    a. What is the type of this GUI component? (i.e. input, navigate, update, display etc.)
    b. What action should occur if a care plan represented by this component is clicked on?

6. Can you draw the screen that will be generated by the VPAM model in figure 126 with the data in figure 127?

7. What do you find hardest to understand about the VPAM model diagram in figure 126?

8. What are some of the reasons for the VPAM model being hard to understand?

    a. Do you find the concept hard to understand? If so what parts of it are most difficult to grasp?
    b. Dou you find the graphical notation in appropriate to describe such models? If so which parts are most problematic?

9. Rate how easy it was to understand the VPAM model in figure 126, from 1 (being quite difficult to understand) to 10 (being very hard to understand)?

A1.4.2.1.2 Question set 2

The next set of questions is regarding the modelling of another screen in the visual patient application modeller. As described above a care plan is usually composed of one or more activities, each of these activities are composed of appointments specifying when the activity is to be done. Some of these activity appointments require the patient to measure and record some aspect of their health, for example, measuring and recording the blood glucose sugar every time before lunch. Figure 128 below shows the VPAM model for an appointment screen for an activity to collect some data from the patient.

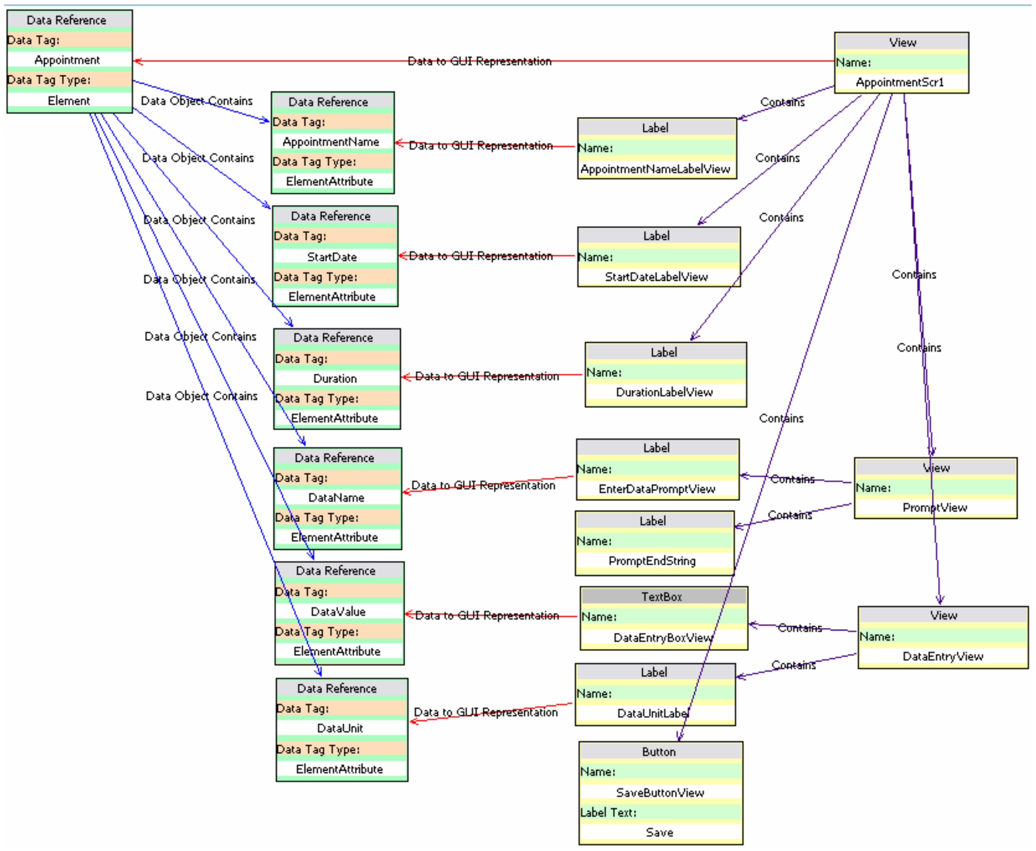


Figure 128: The VPAM model of a screen to represent activity appointments to the patient.

Due to readability concerns, the graphical components in the VPAM model diagram above have been reduced so that all the properties are not visible, however, these components, expanded and with their properties showing can be seen in figures 129 and 130.

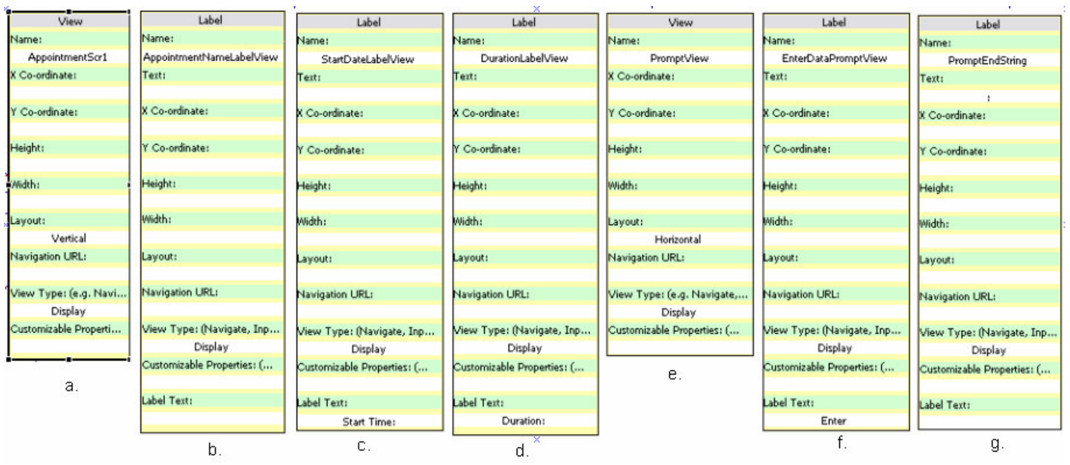
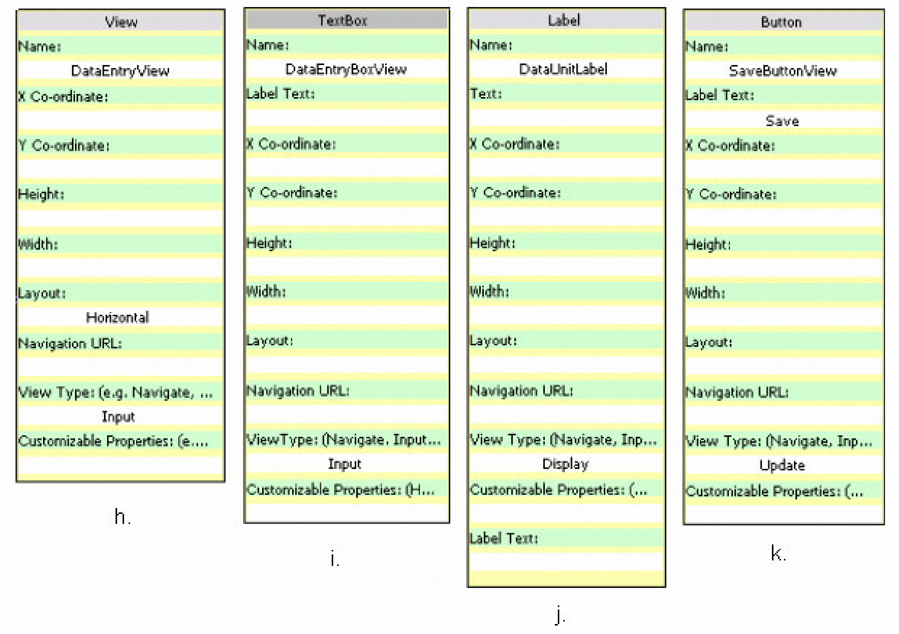


Figure 129: Expanded GUI components of the VPAM model in figure 128.



h. View
Name: DataEntryView
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout: Horizontal
Navigation URL:
View Type: (e.g. Navigate, ... Input
Customizable Properties: (e....

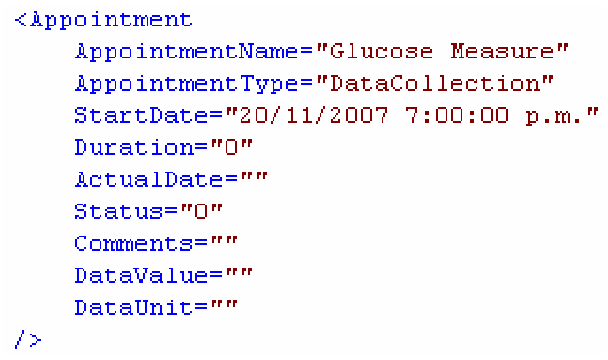
i. TextBox
Name: DataEntryBoxView
Label Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
ViewType: (Navigate, Input... Input
Customizable Properties: (H...

j. Label
Name: DataUnitLabel
Text:
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp... Display
Customizable Properties: (...
Label Text:

k. Button
Name: SaveButtonView
Label Text: Save
X Co-ordinate:
Y Co-ordinate:
Height:
Width:
Layout:
Navigation URL:
View Type: (Navigate, Inp... Update
Customizable Properties: (...

Figure 130: Expanded GUI components of the VPAM model in figure 128

As explained above the concept behind the visual patient application modeller is the mapping of the care plan instance data (which is represented as XML) to the GUI elements that will represent them. The appointment node as stored in the care plan instance XML document can be seen with its attributes in figure 131 below. The VPAM model in figure 6 is a model of how the appointment node and its attributes will be portrayed as a patient application screen. Please study the VPAM model in figure 128 and answer the questions following figure 131.



```
<Appointment
    AppointmentName="Glucose Measure"
    AppointmentType="DataCollection"
    StartDate="20/11/2007 7:00:00 p.m."
    Duration="0"
    ActualDate=""
    Status="0"
    Comments=""
    DataValue=""
    DataUnit=""
/>
```

Figure 131: The snippet of a care plan instance XML document showing the appointment XML node that is contained in an activity.

1. How are the attributes of the appointment such as name, type, start date, and duration represented on the appointment screen?
   a. What type of GUI element is used to represent it?

   b. Write down the label which would occur next to the start date? (Hint: check the label text in figure 129 c above)

2. Study the VPAM model in figure 128 and find the data entry view, then answer the following questions:
   a. What is the layout of the data entry view? (Hint: check the expanded component in figure 130 j.)

b. What are the GUI components that are contained inside the data entry view?

c. The XML element of the appointment node in figure 131 has an attribute called data value, this stores the prescribed data that the user enters through the screen. How does the VPAM model specify this? (Hint: In other words how is the patient supposed to enter in this data, which GUI element is mapped to this data values attribute in the VPAM model?)

3. What GUI component has been included in the model to allow the patient to save the entered data?

4. What is the type of this GUI component (i.e. Input, Update, Display, and Navigate)?

5. After studying the VPAM model in figure 128, can you envision what the screen will look like? Can you sketch the screen with the sample data from figure 131?

6. Rate how easy it was for you to understand the VPAM model in figure 128, from 1 (being very easy to understand, straight forward) to 10 (very difficult to understand, confusing). (Hint: Try to think on how easy it was for you to answer the questions above.)

7. Rate the graphical notation used by the VPAM model in figure 126 and figure 128, from 1 (being difficult to understand, does not relate too well) to 10 (perfect, make language very easy to understand, relates closely to the underlying concept of the language).

8. Rate the concept of the modelling applications using the VPAM language from 1 (being not a very viable concept, may not be very useful) to 10 (being definitely a useful concept).

A1.5 Evaluating the Overall Concept for the Care Plan Modelling System

The purpose of this section of evaluation is to gather feedback on the overall approach and concept behind the "model driven" care plan modelling system. Please answer the following questions on the overall approach of the care plan modelling system.

1. Have you encountered other systems which have followed an approach similar to that of the care plan modelling system discussed here (and shown in figure 121)? If so please state them.

2. What advantages do you think this system has for health care professionals modelling health care information?

3. What advantages do you think this system has for the patients who receive the modelled health care information?

4. Do you think that this is a good approach for modelling and deploying health care information to patients?

5. Do you think that deploying health care information for patients on mobile devices is a good idea? Do you envision any drawbacks?

6. What are the changes if any, would you recommend to the approach of the care plan modelling system? Do you think there is anything missing?

7. Grade the usefulness of the overall approach from 1 (being the very useful, definitely has potential) to 10 (not very useful at all, has no real future of being developed into something useful).