UML design of a system for coordinated care of mentally ill patients after discharge to home

Vasilios Christou* Tsirigoti 5 Ioannina GREECE bchristoul@gmail.com Maria Polizou Konitsis 8 Agios Dimitrios Athens GREECE mpolizou@yahoo.gr Vasilios Apostolopoulos Athens Remoundou 14 GREECE apostolopoulosv@yahoo.com Dr. Andriana Prentza Department of Digital Systems University of Piraeus Karaoli & Dimitriou St, 18534 Piraeus GREECE aprentza@unipi.gr

* Corresponding author

Abstract: In the present paper we present the design of a system for the coordinated care of mentally-ill patients after their discharge to home using UML (Unified Modeling Language). UML is a general purpose modeling language in the field of software engineering. The goal of the system is to set an individual plan for the patient that qualifies for such a plan and agrees to joining such a home care program and to form the team that will support and monitor the patient after discharge from the hospital (nurse, psychiatrist, social worker etc). The CASE (Computer Assisted Software Engineering) tool we used to model the problem is the Visual Paradigm Suite.

Keywords: UML, use case diagrams, class diagrams, sequence diagrams, object – oriented programming, Information System, homecare treatment group

1. Introduction

The purpose of the present paper is the study, analysis, and design of a system for the coordinated care of mentally-ill patients after their discharge to home. In particular, we describe and analyze the case studies and the main actors which produce a complete home care plan for a mentally ill patient. The actors compose the monitoring team and according the profile of the patient, the team synthesis may vary (new patients may enter the team and old patients may leave the team). Each plan is individually set for the specific patient and the same holds for the team that is formed.

As a first step in the analysis of the system requirements, we model the problem for better understanding the organization’s operation and needs. Modeling is important to estimate the feasibility of this project and the time and cost that is necessary for its completion. Also modeling helps developers to reduce production time and cost while they avoid developing errors. Using modeling, we locate possible risks and make a plan to solve them [1].

The modeling languages use methodologies to describe different solutions to the software development. One software development methodology has three basic elements (Symbolism, Procedure and CASE Tool) which are connected together as seen in Figure 1 [9]. In particular, symbolism is a modeling language which helps communication between software engineers and customers during the process of a procedure. This way we clarify the decisions that are not clear or cannot be described by the application code. Procedure defines the way in which we will use the symbolism and the CASE Tool during development. Also procedure is connected with the phases of the software’s life cycle assigning what has to be done in every phase. The CASE Tool validates the work and the support methodology to the computer.

![Figure 1: Methodology Triangle.](image-url)
The most common methodologies before UML where the Booch (created by Grandy Booch), the OMT (Object Modeling Technique created by James Rumbaugh) and OOSE (Object Oriented Software Engineering created by Ivar Jacobson) [1][4][9]. Booch methodology was very good in design and less at analysis, OMT methodology was very good at analysis and less at design. Finally OOSE methodology was very good at behavior analysis. Because these three methodologies started to converge between them and had mutual design techniques but different symbolisms, confusion was made. The development of UML language gave solution to the above problem unifying the symbolisms of Booch and OMT. The UML language has in-built components from other methodologies. In general UML is a hyper set of Booch and OMT methodologies as seen in Figure 2 [9].

![Figure 2: UML as hyper set.](image)

In Figure 3 we see the historical evolution of UML.

![Figure 3: Historical evolution of UML.](image)

UML is an industrial prototype and has been supported from the OMG (Object Management Group) organization which is responsible for its development and continuation. In this organization participate large computer manufacturing companies like IBM, Intel, Sun etc. software developing companies like Microsoft, Oracle etc and methodology creation companies like Rational [1][4][10].

The most significant advantage of UML is that is a modeling language usable by humans and machines. It can be used with any procedure/methodology of development, in any development stage, and is platform and programming language independent. It implements the object – oriented model and uses mostly diagrams to express the object – oriented analysis and software project design. UML simplifies the complex procedure of software design. All the above reasons led us to use UML as the modeling language [1][7][11].

2 Related Work

2.1 UML Methodologies

The most common life cycle models are the waterfall model, incremental model, evolutionary and spiral model.

The basic characteristics of the waterfall model are the serial sequence phases and the feedback between two neighborhood steps. It is based in creating specifications in every step which makes it easier to maintenance. Its major disadvantage is that the user participates only in the beginning of the procedure and then at the end he/she sees the product with delay. During the waterfall procedure it is difficult to check the process because the phase creation may not be always serial and complete [1][10].

At the incremental model we have multiple versions of the system and in each version we add more quality characteristics and functions than the previous one. This model’s advantage is that in each version we have a working system and we have better distribution of cost per time. The disadvantage is that requirements must not change during version changes and must be fixed from the design’s beginning [1][4][10].

The evolutionary model has also multiple versions of the system but with the major difference that the requirements change during system’s implementation. Also the user has continuous participation during these changes and can contribute to the implementation. The disadvantage is that the continuous system requirements change will increase the system’s implementation cost and time [1][10].

In spiral model, we divide the system into phases and in each phase end we do risk analysis. If this analysis has been a success, the system’s development continues or else the development
stops. This model is more suitable in large systems because has increased management cost.

2.2 System implementation using UML

System implementation using UML is done with the following ways [1]:
- Using class diagrams for implementation testing in the system classes
- Using component diagrams and communication diagrams for communication testing between component classes.
- Using sequence diagrams that resemble with the communication diagrams
- Using use case diagrams and activity diagrams that are for recognition that the system behaves as it was initially set in these diagrams
For our analysis, we use the use case diagrams and the sequence diagrams.

The use case diagram shows the functional relations between actors - external users and the system. It does not describe processes, actions or successive situations of the system. These are initially described in each use incident text and then with the activities and sequence diagrams. In particular the use case diagram constitutes the starting point for object - oriented development. An extension point is a report in an area within the use case in which action sequences can be imported from other use cases. Each extension point has a unique name in the use case and an area description in the use case behaviour [1][4][5][6][9][11].

A sequence diagram presents the interaction as presented in time sequences. In particular, presents the instances that participate in the interaction with their “life lines” and the impulses that they exchange placed in the time sequence. It does not present the relations between objects [1][4][6][9][11].

2.3 UML Tools

For the diagrams implementation a lot of commercial and free tools are available. The most common modeling tools are Visual Paradigm, Argo UML, Rational Rose, Power Designer and Eclipse UML 2 Tools [1][4]. For the diagrams implementation we used Visual paradigm because it has a user friendly environment and ease of use in methodologies development [3].

3 Research Analysis and Results

3.1 Problem Description

Our approach regards a real case that formed the Norwegian pilot in European research project Linkcare [2]. The description of the pilot follows. The patient has been diagnosed as manic – depressive and has been hospitalized for a long period [2][3]. During his hospitalization, his care team plans with the patient his rehabilitation in the society. The hospital's staff produces a personalized plan and makes the necessary calls to the community nurse that will be responsible for patient’s healthcare program, so that this program will be finalized according his needs.

The healthcare professionals agree on a meeting date. Before the responsibility team meeting for patient’s treatment, the patient meets his new coordinator (the municipality's nurse). These two together decide the access period and the access level for each participant in the plan and who can be included in the future care of the patient. The coordinator gains access to the system by collecting the required personal information from the participants. Besides the coordinator and the patient, they decide to include his psychiatrist from the hospital, a social worker from the municipality in which he lives, an officer from the local employment office and his best friend.

At the first meeting of patient’s responsibility team, the members are being trained in the system functionality which is used for the creation and maintenance of personalized patient monitoring plans. Security in handling sensitive data on the internet is crucial and a part of the first course emphasizes each participant's responsibility. At the same time they are informed for the creation of personalized plans, since not all the participants know the purpose and the procedure of creating such plans.

The first meeting establishes a framework for a future personalized plan for the patient. Some areas of life are included and some not. All his needs are based on his diagnosis, for the patient the residence is ok, but he needs a more flexible job according to his new health situation and financial help for a period. The family and the social network do not constitute test subject for the time being. In the plan actions, the deadlines and who is in charge are recorded.

The next meeting is being decided. The members begin to extend and do a more detailed plan, connected in this via encrypted Internet
access, adding new information, sending messages between them inside the application so that they are able to inform for the changes or the additions. All participate according to their most relative plan section to them. They build a solid support platform for the patient after his hospital discharge, based on the agreements and the detailed plans that are accessible for the participants.

At the beginning the patient’s responsibility team has meetings every third month. Since the patient's situation is being stabilized, they meet two times every year evaluating the plan, changing participants or subjects in which they focus their attention, according patient’s needs. While patient's situation is being improved, the patient begins controlling the biggest part of the plan, having the coordinator as the plan's executor. After five years, patient's situation is stable. He has a new suitable job and few problems living with his psychiatric diagnosis. The personalized plan is completed, since he continues to be followed up by his psychiatric nurse and psychiatrist at the hospital.

### 3.2 Results

The system implementation is described using two parts. In the first part we develop the Use case diagrams and in the second the Sequence diagrams [2][3].

#### 3.2.1 Actor's determination and Use Case Diagrams design

The main system actors are the hospital staff, the nurse-coordinator, the team that is responsible for the patient and the patient. Specifically the hospital staff is consisted from the group of employees that work in the psychiatric hospital and watch many and different incidents of patients that suffer from psychiatric problems. The nurse who works in the community undertakes incidents of patients that have been assigned to her by the hospital's staff. She has the main co-ordination of programs that are designed to support the patients after their hospitalization. The team that is responsible for the patient constitutes from a group of people that comes each time from different environment, but has a relation with the illness and patient's life. At early stages, the team that is responsible for the patient is composed from the community nurse, the hospital's psychiatrist, the community's social worker, the local employment office's employee and the patient's best friend. Then and depending on the patient's progress the members change attributes or withdraw. The patient has been diagnosed as maniac - depressive, he has been hospitalized for a long time and a personalized plan has been created for him after hospitalization [1][2][3].

The most significant use case diagrams follow in the next figures.

1. Nurse - Hospital staff

![Figure 4: Nurse – Hospital staff](image)

The hospital staff examines the patients’ incidents and produces a personalized plan for each case to support the patients after their hospitalization as seen in Figure 4. Then it promulgates the plan in a nurse from the local community to approve it or make any changes. If the nurse undertakes the incident, they arrange a meeting date, in order to study the patients’ action plan. If the plan does not need review the nurse finalizes it and begins the plan's action. At the first stage, the hospital staff with the nurse will create the patient's responsibility team. At the second stage, after nurse's meeting with the patient, the responsibility team will be affirmed. Then the nurse-coordinator creates the system's user accounts. These accounts will help users to communicate and record essential plan data. The nurse as main system actor is responsible for the team member's co-ordination and for meeting scheduling. If the patient's condition improves and the program's purpose has been achieved, the nurse completes and closes the action plan in the system [2][3].

2. Nurse - Patient

![Figure 5: Nurse-Patient](image)
As we previously reported, the patient actively participates in the meetings, initially with the nurse and then with the team as seen in Figure 5. He decides in agreement with the nurse for his team members [2][3].

3. Responsibility team members

In Figure 6 the initial structure of the responsibility team has as follows: Nurse, hospital staff, patient, psychiatrist, social worker, employment office employee and patient friend [2][3].

![Figure 6: Responsibility team members](image)

4. Responsibility team’s based actions.

We remind that all responsibility team members participate in the meetings and have access to the system. This procedure is clarified in the diagram of Figure 7.

![Figure 7: Responsibility team actions](image)

The hospital psychiatrist comes in the team meetings and continues his actions. These actions include patient observation for a long time, even afterwards the plan’s completion. Social worker’s main action is the participation in the meetings and patient’s observation with purpose his rehabilitation in the society. As secondary and optional actions we place patient's residence finding and his participation in social events. The officer of the employment local office makes meetings with the rest of the team and updates the system with the patient's progress. As secondary action, we place proper work finding, depending on the case and the patient's conditions. The patient's friend with the psychiatrist, the social worker and the employment office employee is an active member of the responsibility team and participates in the team meetings, closing - confirming the appointments via the system. As secondary action we place the patient's economic aid during treatment - plan initial stage and the participation in various events [2][3].

5. Responsibility team basic actions at the system.

When the responsibility team is educated at the system operation, it creates the patient's main life sections that will be included in the plan. Then we assign each sector's responsible actors and we set the completion deadlines for each of their actions. The application entry is remote, this requires each participant having internet connection and encrypted system access. Then the system users can import patient related information and his progress, to modify and exchange messages between them. This way they are recorded and the patient's personalised action plan is being updated. The whole procedure is been described in Figure 8 [2][3].

![Figure 8: Basic actions at the system](image)

3.2.2 Sequence diagrams creation based on principal use cases

1. Login Sequence diagram

The application operator opens his computer and the computer begins its operational system. Then
opens the application and the system asks the operator for the input data (user name and password). The user imports them and the system identifies him. If the provided data are correct displays the application's main screen and all the incoming messages from other users (the rest team members). The sequence diagram of Figure 9 describes this procedure [2][3].

![Figure 9: Login sequence diagram](image)

2. New project at the system

If the preceded actions (boot, user entry, user identification) have been completed patient search in the system follows in order to create a new action plan. The plan after its creation is stored in the application database and the application user returns in the program's initial screen [2][3]. The creation of a new project is described in the sequence diagram of Figure 10.

![Figure 10: New project](image)

3. Team creation

The application operator presses the button for new responsibility team creation. The command is been processed and sent in the application. The application opens the new team screen. The operator gives the required registration data, like the new team name, seeks and adds the users in the team. The application makes periodical checks, for data validity during their entry. After completion the operator stores the new team. The system processes the command and if all the provided data are correctly registered it stores the new team in its database. The user comes back in the initial application menu for any other system action. Figure 11 clarifies the procedure described above [2][3].

![Figure 11: Group creation](image)

4. Project completion

The nurse that watches and coordinates the patient's action plan, after many years of patient treatment and from the meeting decides that the patient was mentally cured and successfully rehabilitated in the society. The system processes the status change (given by the nurse), the database is being updated and the user sees the changes in his screen. The project completion procedure is been described in Figure 12 [2][3].

![Figure 12: Project completion](image)
4 Conclusions

The UML use for analysis and design of the system is very important, because through the use case diagrams and sequence diagrams becomes comprehensible from the application software engineer until the application end user, the operation of the system [2][3].

In particular, the software technology tools can be incorporated in a complex software environment, automate the program management activities and manage all the work products that are produced in the whole process. Therefore they help the engineers in the coding, testing and plan analysis. Most important is that the UML case tools can provide new ways of software technology that improve the software engineer's perception. This leads us to better decisions and to higher quality software.

References