Chapter 4

Software System Architecture on Treatment Plan in Healthcare System

This chapter identifies Treatflow Management System which aims at automating a treatment process. TFMS is an instantiation of a Treatflow. Here we propose a framework to implement Treatflow Management System. A requirement analysis for such a system considering users viz. (doctors, patients and staffs) and their roles in treatment process has been discussed. The analysis is presented with the help Usecase and Activity diagrams. These diagrams lead to a first-cut of a system architecture and the software modules that the system should have. Then we provide higher level design diagrams for each module. In order to be brief and at the same time to bring clarity on design decisions, we have made use of Class and Sequence diagrams.

4.1 Introduction

The proposed Treatflow Management System aims at automating treatment process and is different from other existing healthcare systems which focus on patient registration, bill collections, maintaining the history of medication undergone by patients especially EPR: electronic patient record management. We feel, such a system will be helpful for both doctors and patients in managing treatment process. On registration of a patient, a doctor can make use of the proposed system.

1This work has been published in proceeding of International Conference on Managing Next Generation Software Application, Coimbatore, pages 871-879, 2008
to synthesize a treatment process for the patient by choosing treatment modules which the doctor thinks useful for treating the patient. The patient views the treatment records and updates his/her status. Using the system, the doctor can give health advice to the patient. The proposed system aims at addressing these requirements.

In previous chapters we have provided a theoretical basis to specify a treatment process and here we propose a framework to implement it. First, we present requirement analysis for such a system considering users viz. doctors, patients and staffs and their roles in the treatment process. The analysis is presented using Use case and Activity diagrams. These diagrams lead to first cut of a system architecture and the software modules that the system can have to meet the requirements. Then we provide a higher level design diagrams for each module. In order to be brief and at the same time to bring clarity on design decisions, we have made use of Class and Sequence diagrams.

In the following section we present a survey on state of the art of healthcare management systems. This section is followed by Requirement analysis of the Treatflow Management System (TFMS) is discussed in 4.3. The proposed system architecture is discussed in section 4.4. And then, a design overview of each major architectural component is presented in section 4.5. The behavior of a prototype system is explained in section 4.6. Finally, the chapter ends with a summary.

4.2 Related Work

A software architecture for workflow management system model comprises of major components of workflow management system and their relationship amongst the components [38]. In some cases architectural description is basically divided
into two parts i.e. start up time architecture during which the workflow is constructed and build up time architecture during which the combined negotiation is conducted [10].

In case of healthcare, an architecture that integrates workflow management and context aware actions for personalized healthcare services has been proposed [3]. It supports interaction among hospital personnel and patients and stores patients’ health status for future reference. The proposed architecture follows web service composition techniques to compose healthcare services for a patient and implements collaborative autonomous agent concept to deliver context-based personalized healthcare advisory to patients. The architecture deploys agents like guideline manager, supplier, context manager, monitoring devices, context record manager and activity execution agent. The agents like guideline manager, supplier, context manager, monitoring devices are treated as web services that can be triggered by invoking the other agents viz. context record manager and activity execution agent. These agents can be located on Internet thus making it a distributed architecture. Mainly, the architecture is meant for managing work schedules of healthcare staff and issuing health advisory to patients. It logically decouples workflow management and action management but is able to provide orchestrated services by deploying collaborative agents.

Flexible workflow management system in the area of patient care delivery was designed and implemented by using high level components. In this case OBJCHART was used as a coordination and communication manager. Similarly CLP(R) was used as a logic based deductive system for reasoning purposes and $W_a f^e$ wss used for graphical user interface and front end capabilities [58].
4.3 Requirement Analysis

As stated before, Treatflow Management System (TFMS) mainly focuses on treatment process management. A treatment process is initiated by a patient on the first visit to a doctor for consultation on some ailments. In the process of treatment actors (users) for example, administrative staff for patient registration play certain roles. Here, we have narrowed our scope by considering three types of users viz. doctor, patient and staff, and perform the analysis. Usually we find three types of users in a small or medium sized typical healthcare organization. A similar approach can be extended for other actors in correspondence to their roles in treatment process.

The purpose of this system is to specify and verify a Treatflow as well as to execute, monitor and handle exceptions while treating a patient. This system is to be used as an aid by different actors to participate in the treatment process and moreover the proposed system should have proper means for enabling interaction as well as making decisions. Next we analyze the requirements in perspective of each type of users.

- Patient Perspective

Patient as a user can use the services provided by the system. A patient wishing to use system services is first to be authenticated. A patient can perform two types of services used for retrieval of treatment related information and providing feedback on effectiveness of treatment process. Treatment related information can be provided to patient on request basis (received from the patient) and by pushing the information to the patient. Thus the system provides both pull/push modes for information retrieval. Pushing of information to a patient is performed by a doctor or a paramedical staff in charge of
treating the patient. This can be implemented by making use of email/short message services. Pulling information is primarily performed by a patient for clarifications and to acquire necessary information on treatment process. For example, a patient may refer to nutrition advice during intervention of a particular drug. Treatment related information regarding current line of treatment, care, caution, prescription of drug etc must be available to patient on demand. As told earlier, the other objective for patient interaction is to provide feedback. For example, on intervention of a drug, the treating doctor may like to know its effectiveness and for that the patient is required to inform about his/her health status. Similarly, a drug may react and a patient may develop exigencies that requires doctor’s intervention. In order to deal with situation, the patient should have means to feed the system on his/her health conditions and inform the concerned doctor and staff. The use case diagram given in Figure.4.1 has three major use cases viz. Authentication, Retrieve Information and Update Status. The first use case is
meant to ensure security and privacy for a patient so that only the patient is eligible to access/update his/her health information. The second use case Retrieve Information uses two sub-use cases viz. Pull Info and Push Info. These sub-use cases receive information from database. This database contains information on patient’s health status as well as treatment process. The database is instantiated at the beginning of a treatment process for a patient. And it is updated by the patient, the treating doctor and the staffs. While patient updates health status, the treating doctor update treatment process. The third major use case is Update Status and it has two sub-use cases. These are Update Health Status and Update Health Exigencies. These sub-use cases are meant for a patient to update health status and to inform on health exigencies. Figure 4.2(a) presents flow of activities pertaining to Authentication use case as shown in Figure 4.1. In this case patient as a user has to enter his/her user-id and passwd which is to be validated by the system in order to maintain security. Lest an incorrect user should re-enter user-id and passwd if he/she has to continue further. Similarly, activities for patient Retrieve Information use case are given in activity diagram as shown in Figure 4.2(b). After authentication, patient can query his/her health related information. The information is displayed if it is available, otherwise a message will be displayed informing about the unavailability of information. Activity diagram for patient Update Status use case is given in Figure 4.2(c). In order to give feedback, patient has to update information regarding either health status or health exigencies. In both the cases, if information already exists and there is a need to change the information, then in those cases modification of information related to health status as well as health exigencies can be done. And if the information already exists and that information is not necessary, then that information can be removed.
Figure 4.2: Activity Diagrams for Patient
Otherwise new information can be added.

- **Doctor Perspective**

As an authenticated user, doctor can use the services viz. Authentication, Manage_Treatment and Handle_Exigencies provided by the system. Doctor has to make a treatment after going through patient’s health status. Doctor can create a treatment with the help of Treatplan. Treatplan is available in database which can help not only in specifying but also in verifying a treatment. A treatment plan may have several treatments for patient having different ailments. During the treatment process, doctor can view the details about the treatment. Depending on the progress of treatment and based on doctor’s past experience, he/she has to decide whether to continue with the same treatment plan or to make some change in the current line of treatment or altogether go for an alternative treatment. In certain cases exigencies may happen which have to be taken care of by doctor. Doctor can also update patient’s health status in the database. The use case diagram given in Figure.4.3 has three major use cases namely Authentication, Manage_Treatment and Handle_Exigencies. Manage_Treatment use case which has two sub-use cases viz Make_Treatment and Update_Treatment. Make_Treatment has two sub-use cases viz Create_Treatment and Validate_Treatment. Doctor can make a treatment The Create_Treatment use case is meant for making a treatment. Doctor can make use of Treatflow which is already stored in the database, for creating a treatment. Therefore this Make_Treatment use case which can extend to Treatflow. After making a treatment, the treatment plan can be validated with the use of Validate_Treatment use case. Similarly Update_Treatment use case has three sub-use cases viz. View_Treatment, Modify_Treatment and Delete_Treatment. Doctor as an actor, can view all the details of treat-
Figure 4.3: Use case Diagram for Doctor

ment with the help of View_Treatment use case. After going through patient’s health status and treatment detail, doctor has to decide whether to continue in the current line of treatment or to make changes in the treatment process. Apart from viewing, doctor can change a treatment plan with the help of Modify_Treatment use case or can remove treatment plan with Delete_Treatment use case. The last use case is the Manage_Exigencies which is used to view the exigency messages sent by patient or staff in case of emergency. This use case not only uses Handle_exigency use case for handling exceptions but also uses Update_Exigency use case for updating health related exigencies. Authentication use case is to provide security so that only authorized doctor and staff can access the system. Therefore in case of Authentication, activity diagrams for doctor and staff are same as patient activity diagram in Figure. 4.2(a). According to Figure.4.4(a) doctor has to first make a treatment which has to be validated. Doctor can save the treatment in case it is valid otherwise he/she has to go for an alternate
(a) Activity Diagram for Doctor Manage Treatment Use Case (b) Activity Diagram for Doctor Handle Exigencies Use Case

Figure 4.4: Activity Diagrams for Doctor

treatment. For an already existing treatment if a modification is needed, then doctor can modify the treatment. But if the ongoing treatment is not satisfactory, then the doctor can remove the treatment and can go for an alternate treatment. Similarly list of activities for handling exigencies are shown in activity diagram in Figure 4.4(b). Doctor has to first view the exigency messages sent by patients and take action for each and every message. Finally doctor has to update the health related exigencies in the database.

• Staff Perspective

Services to be provided by the system to staff are as follows. System should first authenticate the staff as a user of the system. Staff has to do registration for patient by entering all the detailed information about the patient. After registration, staff has to allocate user-id and passwd to the patient. Staff is responsible for recording symptoms as well as assigning doctor to patient during the first phase of Administration. He/she is also responsible
for updating patient’s health status. Figure 4.5 has two use cases viz Authe-
number, mail-id etc. But in case of existing patient, if there is a need to modify information about the patient, then staff can modify the information. Otherwise if wanted, staff can also remove the existing patient from the database. The list of activities for Recording use case are listed in activity diagram as shown in Figure 4.6(b). If the patient is visited for the first time, then the staff records all symptoms of patients and assigns a doctor to the patient. But in case of already registered patient, if patient’s feedback is to be included, then the staff can update patient’s health status in the database.

4.4 System Architecture

System architecture and its components are discussed in this section. While considering different end users functionalities, system is basically divided into two parts: Treatflow\_Library\_Make, which is prepared off-line and Treatflow\_Consultation, which is considered to be on-line. System architecture for TFMS is shown in Figure 4.7. The system architecture comprises of modules which are represented as
A module is composed of sub-modules and these sub-modules are executed by users. Each module of the system architecture is represented in a rectangular box. There are three layers in the box as shown in Figure 4.8. First layer shows module name, second layer represents names of sub-modules (if any) and whereas third layer shows the users of the module.

Figure 4.8: Representation of Module

- **Treatflow.Library.Make**
  As we have already discussed in the previous chapter, the order of execution of treatment activities is termed as Treatflow. Treatflow is specified
off-line by doctor and is stored in library for further use. Specification and verification of a Treatflow is implemented by the module Treatflow. Therefore, Treatflow.Library.Make a part of system architecture is responsible for specification and verification of Treatflow. It can be implemented by using Specify and Verify sub-modules.

- **Treatflow.Consultation**
  This part of system architecture is composed of modules namely Authentication, Administration, Treatplan, Treat-Management, Handle-Exigencies, Role-Based-Enquiry. They can be described as follows.

  - **Authentication**
    A user of any category namely doctor, patient or staff can access the services of the system by using his/her user-id and passwd. The functionality of Authentication module is to provide and maintain security and privacy for the Treatflow Management System (TFMS). A user can change his/her passwd using Authentication module.

  - **Administration**
    The Administration module is used by staff for registration and for recording patient’s health parameters. Therefore, this Administration module has two sub-modules namely Registration and Recording. With the help of Registration module staff can enter the detailed information of patient viz name, address, phone number etc. Similarly with the help of Recording sub-module, staff can record patient’s symptoms and health parameters viz temperature, blood pressure etc. Staff is also responsible for assigning a doctor to the patient. The above administrative functionalities can be implemented by using the Administration module.
- **Treatplan**
  The functionality of Treatplan module is to help a doctor in making a treatment plan for a patient depending on the symptoms. Treatment plan thus made can also be validated. This making up of Treatplan and validation of Treatplan can be implemented by Treatplan module.

- **Treat_Management**
  With the help of Treat_Management module, a patient can view certain parts of treatment activities like drugs, dosage, effects, caution etc. In order to give feedback, patient has to update his/her current health status. Similarly doctor can also view the progress of treatment. Therefore as an end user, doctor can view the treatment details of patient and can modify / upgrade the treatment as well as patient health status. The Treat_Management provides doctor, staff and patient a facility to view treatment detail. Using Treat_Select sub-module, doctor can initiate a treatment process. And using Update_Treatment sub-module doctor can update existing treatment plan.

- **Handle_Exigencies**
  During treatment, in cases of emergency like sideeffects or allergy which may erupt at any point of time, a patient has to inform the drug side-effects. Doctor has to take action for these type of exigencies and has to handle such exigencies which is possible with the help of Handle_Exigencies module. Also the flow of treatment may not proceed further in case Postc (post-condition) is not satisfied. For example, a patient has not taken medicine as per the treatment activity Take_Drug. In this type of exigency patient may get a message and he/she has to take action in order to handle exigencies.

- **Role_Based_Enquiry**
Authorized users like doctor, patient and staff who have special permission, can access allowable treatment related information for their purpose.

- **Treatment Repository**

While looking for a repository in Treatflow Management System for storing treatment related activities and associated data well proved relational database technology RDBMS have chosen for this purpose. Information model is a fundamental concept to study in healthcare workflow repository. The information model determines how information is stored and retrieved from a repository and how well the integrity constraint can be maintained. Healthcare industry is very sensitive and health related information keeps changing and so does treatment procedures. Therefore there is a need to update information by extending the existing information model to accommodate the changes.

### 4.5 Module Design

We sketch a broad picture on design of each module with the help of class diagrams. While these class diagrams depict the structural view of the system architecture, the behavioral aspects of it are analyzed in the following section with the help of interaction diagrams. Design of each module of the system architecture is discussed below.

- **Treatflow**

  Class diagram for Specifying a Treatflow has member functions namely Draw\_tflow, Add\_tflow, Modify\_tflow and Delete\_tflow as shown in Figure 4.9. A Treatflow (denoted as tflow) can be specified with the help of these operators. In order to start a Treatflow, Draw\_tflow member function can
Figure 4.9: Class Diagram for Treatflow

be invoked. Similarly in order to connect treatment activities Add_tflow is used. After specifying a Treatflow it invokes another class Treatflow_Verify for verifying the Treatflow. With the help of Verify_tflow member function, the Treatflow can be verified. Once a Treatflow is correct, then by invoking Save_tflow as member function the Treatflow can be stored in Treatment_Repository.

- **Authentication**

  As already shown in system architecture in Figure 4.7 In order to authenticate, users of any category namely doctor, patient and staff can access the system with their user-id and passwd. This is possible with the help of Ask_toAuthenticate as the member function. This user-id and passwd must be validated in order to maintain the system secure which is done by the member function Execute_query available in Treatment_Repository. After validation a valid user can directly enter into the system and there is no need to inform to the user which can be done by the Hide function. Similarly in case of an invalid user, system has to display message which is possible by Show member function. Validate member function automatically invoked
by the Ask_to_authenticate member function and therefore all these member functions are appeared in class Authentication as shown in Figure 4.10.

- **Administration**

  In Administration, staff can do registration and record the patient’s health parameters with the help of registration and recording sub-module respectively as discussed in Figure 4.7. During registration, staff has to allocate user-id and enter detail information of patient. This can be implemented by invoking the member function Allocate_user-id and Enter_patient_detail respectively. This information has to be stored into Treatment_Repository. Staff has to record patient’s symptoms, assign doctor to patient, for which member functions like Record_symptom and Assign_doctor can be implemented as shown in Figure 4.11.

- **Treatplan**

  As we have already discussed, doctor has to prepare a treatment plan by invoking Create_treat member function. This treatment plan can be validated and stored in the treatment repository with the help of Validate_treat.
and Save_treat respectively as shown in Figure 4.12.

- **Treat_Management**
  
  Treat_Management module comprises of Update_Status, Update_Treat and View_Treat sub-modules as discussed in Figure 4.7. While managing treatment, doctor can update patient’s health status by adding, changing or removing patient’s health parameter by implementing Update_Status class. Before updating treatment, doctor has to view patient’s treatment details and patient’s health status. Update_Treatment class includes member functions like View_pt_treatment, View_pt_detail and Update_treatment as shown in Figure 4.13. To support these functionalities we need Treatment_Repository.

- **Handle_Exigencies**
  
  In case of emergency patient as well as doctor can update health related exigencies and take necessary actions (if required) to handle these exigencies.
The doctor and patient can view the messages related to exigencies and take action to handle those exigencies. So, in order to satisfy these functionalities, View_message and Take_action are the member functions that must be implemented. This is possible only with the help of Treatment_repository as shown in Figure.4.14..

- **Role_Based_Enquiry**

  As an authenticated user, namely doctor, patient or staff can query about treatment related information and get the result based on their roles. For example, as discussed in previous chapter some treatment related attributes must be hidden from patient as well as staff. To achieve these functionalities, member functions like Take_query and Display_result must be implemented which is given in Figure. 4.15.

### 4.6 System Behavioral Analysis

Here we discuss the behavioral analysis of class diagram with respect to each requirement. In this case analysis is done with the help of interaction diagram.
Based on each user requirements how the modules interact among themselves is discussed below.

- **Authentication** Considering patient, doctor and staff as users, authentication functionality provided by the system is shown in interaction diagram in Figure 4.16. As we have already discussed, to maintain the system secure, a user has to authenticate with his/her passwd. In this case, in order to authenticate, user has to enter user-id and passwd with the help of member function Ask_to_authenticate as shown in Figure 4.10. This user-id and passwd has to be validated by the member function Validate_treat which is declared in Validate class as shown in Figure 4.9. After executing query, it sends back the information regarding validity. In case of an invalid user a message is displayed with the help of Display function. But in case of a valid user it hides the message with the help of Hide function and directly enters into the system.

- **Handle_Exigencies**: In order to handle exigencies the way the doctor as well as patient interact with the system is given in Figure 4.17. During treat-
Figure 4.14: Class Diagram for Handle_Exigencies

Exigencies, in case of exigency patient as well as doctor can update health related exigencies with the help of Update_exigencies member function which is given in Handle_Exigencies class in Figure 4.14. While doing modification, user can select any of the option for example Add, Change or Remove which invoke Add_exg, Modify_exg or Delete_exg member functions declared in class Treatment_Repository as in Figure.4.14. In order to handle such exigencies doctor can view messages stored in repository. Once the message is accessed from the repository with the help of Get_message function, then doctor has to take action by invoking the member function Inform_action. This action has to be recorded in the repository with the help of Record_action. This process continues until there are no exigencies messages available in the repository.

- **Make_Treatment**. Based on symptoms, doctor has to prepare a treatment plan for patient. The treatment plan is prepared with the help of Create_treat member function which is to be validated by validate_treat function as declared in Figure. 4.12. After validation doctor can store the treatment plan in the Treatment_Repository using Store_treat function as
• **Role Based Enquiry** Staff, doctor and patient as users want to access some treatment related information stored in Treatment Repository. Interaction diagram for role based enquiry is given in Figure 4.19. Depending on their requirements, they can write a query which is implemented by Take query member function declared in Role Based Enquiry class as in Figure 4.15. This query has to be executed by the member function Execute_RE_query available in Treatment Repository class.

• **Recording** During patient’s first visit staff has to record patient’s symptoms into the repository. This can be implemented by using the Record_symp
member function declared in Recording class given in Figure 4.20. While recording, in order to decide whether to add new symptoms or change or remove already existing symptoms, the functions like Add_symp, Modify_symp or Delete_symp must be implemented. After recording staff can assign doctor to patient based on his/her symptoms which are stored in repository with the help of member function Save_doctor_pt.db. This interaction of Recording_Symptom and Treatment_Repository is given in interaction diagram shown in Figure 4.20.

- **Registration** When a patient enters into a hospital for treatment for the first time, staff must do registration for the patient.
tration he/she should allocate record for that patient by implementing the
member function Allocate_user-id which is available in Registration class.
This initiates a patient record in the database by implementing the func-
tion Init_pt$db in Treatment_Repository. The detail information of patient
e.g name,address,phone no can be stored by implementing Enter_pt_detail
member function. This helps in maintaining patient database with the help
of Populate_pt$db function available in Treatment_Repository. During reg-
istration interaction of staff as a user is given in diagram Figure. 4.21.

- **Treatflow** For specifying a Treatflow, a doctor has to first draw a Treatflow
using Treatflow primitives. This can be done by implementing Draw_tflow
Figure 4.21: Interaction Diagram for Registration

Figure 4.22: Interaction Diagram for Treatflow

member function available in Treatflow_Specify class as in Figure 4.9. After drawing a Treatflow, doctor can add, modify or delete a Treatflow with the help of Add_tflow, Modify_tflow, Delete_tflow function respectively until the specification is over. Once a Treatflow is specified, that can be verified by implementing Verify_tflow member function available in Treatflow_Verify class. After confirmation about the correctness of Treatflow from Verify_Treatflow, doctor can send a message for storing it into the Treatment_Repository by implementing the function Store_tflow. Interaction between Treatflow_Specify, Treatflow_Verify and Treatment_Repository
• **Update Status:** Users of any type can update the patient’s health status by invoking Update_status member function and for this type of updating status, the interaction diagram is given in Figure 4.23. In this case patient, doctor or staff as a valid user can add, change or remove health status of patient by implementing the member function Add_status, Modify_status, Delete_status respectively which are available in Treatment_Repository class.

• **Update Treatment** Doctor as a user can update a treatment for a patient. Interaction diagram for updating treatment is given in Figure 4.24. Doctor has to first go through the details of patient’s treatment. In order to view the treatment details it has to first invoke View_pt_treat function of Update_Treatment class Then to get information from the repository Get_pt_treat function must be invoked. After receiving information and before updating the treatment plan, doctor must view the current health status of the particular patient with the help of View_pt_status. After receiving information doctor can update the treatment with the function Update_treat. This helps in modification i.e. either adding, changing or removing a treat-
Figure 4.24: Interaction Diagram for Update Treatment

ment in the Treatment Repository class using member functions namely Add_treat, Modify_treat or Delete_treat. In this way how different types of users interact among themselves as well as with the system is discussed in detail with the help of interaction diagrams.

4.7 Summary

Treatflow Management System aims at automating a treatment process which is an instantiation of a Treatflow. Here we proposed a software system architecture to implement Treatflow Management System. A requirement analysis for such a system considering users doctors, patients and staffs and their roles in treatment process has been discussed. The analysis is presented as Use case and Activity diagrams. These diagrams lead to a first-cut of a system architecture and the software modules that the system should have. Then we provide higher level design diagrams for each module. In order to be brief and at the same time to bring clarity on design decisions, we have made use of Class and Interaction diagrams.

Healthcare workflow is to be flexible for modification or even replace-
ments. Usually such a workflow has to have long life as a treatment happens in reality. Again, treatment for each patient is unique, so instances of ongoing treatments are to be properly stored so that during treatments these can be accessed readily. Treatment associated information about patient, doctor and healthcare staff are to be stored. As discussed, all these demand a scalable repository to cope up with demand of storing information on treatments of teeming population.