**Hospital Management Sequence Diagram**

Michael Maye

Colorado State University Global

CSC470: Software Engineering

Dr. Shaher Daoud

07/23/2023

**Hospital Management Sequence Diagram**

Sequence Diagrams are essential tools for designing and analyzing a system, allowing developers to trace and visually verify system behavior over time. Analysis at this level provides insight and understanding of object roles and their interaction with others and the system during runtime operations. It gives developers and stakeholders a clear view of system operations and how they solve user problems. The following analysis will explore the submitted sequence diagram, representing a patient's appointment cycle within a Hospital Management System.

**Overview**

The Sequence Diagram to be analyzed represents the cycle of a patient making and attending an appointment at a hospital, specifically how the system handles this action. It begins with a patient requesting an appointment, then moves to the Doctor validating a date and time, then the patient attending the appointment and closing the action. Key actors in this sequence are the patient, an external entity, and the Appointment System, the Receptionist, and the Doctor, which are internal entities of the system. The diagram outlines messages between these entities, the activation, deactivation, and destruction of objects within the sequence, and notes that give further details on actions and system state.

**Actors**

The patient is an external entity that interacts with the system, represented by an Actor in the UML diagram. It is responsible for making an appointment with the Appointment System, checking the appointment details, visiting the hospital for the appointment, and finally checking out with the system. The Appointment System, Receptionist, and Doctor are each represented by objects within the diagram and are internal to the system. The Appointment System object is responsible for making and managing patient appointments, including receiving the appointment request, getting validation on doctor availability, creating the appointment record, setting a reminder for the patient, and finally completing patient checkout. The Receptionist is responsible for receiving a request for date time validation from the Appointment System, validating it with the Doctor, then closing the patient's appointment on checkout. The Doctor is responsible for confirming their availability for the appointment and treating the patient.

The Doctor and Receptionist are meant to perform further actions with other patients and will be deactivated but not destroyed. The Appointment System and Patient have completed their functions and are destroyed upon completion of checkout. If the appointment system is required again for another patient, the system must call a new instance, as a different patient should not have access to the appointments and records of the previous patient.

**Messages**

Arrows between entities represent messages sent between them. Synchronous messages, like 'requestAppointment(doctorId, dateTime)' indicate a direct call from the patient to the Appointment System, which will not proceed until a response is received. This visualizes the direction of information flow and dependency between objects. The patient (sender) expects a response before continuing with other actions.

This system also includes method calls as messages, as well as self-messages. The message 'returnConfirmation()' is a method call where the appointment system returns a confirmation to the patient that their appointment is successfully booked. The Appointment System calling createRecord() is a self-message method call instructing the Appointment System to create an appointment patient record for the patient during the initial booking phase.

Asynchronous messages, represented with dashed lines, do not impede the sender's execution of other actions. The 'sendReminder()' message is asynchronous, where the Appointment System can send these messages and continue with other operations required, as a return message is not required for other operations.

**Figure 1**

Sequence Class Diagram

*Hospital Management System Appointment Sequence:*

A screenshot of a computer screen

Description automatically generated

**Figure 2**

Sequence Class Pseudocode

// Actor: Patient

// Patient wants to book an appointment

Patient {

doctorId = getDoctorId()

datetime = getDatetime()

// Request appointment from the system

AppointmentSystem.requestAppointment(doctorId, datetime)

//Destroy object

destroy()

}

// Object: Appointment System

AppointmentSystem {

function requestAppointment(doctorId, datetime) {

// Validate availability with Receptionist

Receptionist.validateAvailability(doctorId, datetime)

// Create appointment record

createRecord()

// Return confirmation to patient

returnConfirmation()

}

function createRecord() {

// ... logic to create record ...

}

function triggerReminder() {

// ... logic to trigger reminder ...

// Send reminder to patient

sendReminder()

}

function sendReminder() {

// ... logic to send reminder ...

}

function checkAppointmentDetails() {

// ... logic to check appointment details ...

// Return details to patient

returnDetails()

}

function checkOut() {

// Verify patient checkout with Receptionist

Receptionist.verifyPatientCheckout()

// Close patient appointment

closePatientAppointment()

// Checkout complete for patient

checkoutComplete()

// Destroy object

destroy()

}

}

// Object: Receptionist

Receptionist {

function validateAvailability(doctorId, datetime) {

// Check Doctor's schedule

Doctor.checkSchedule(datetime)

// Confirm booking

confirmBooking()

}

function verifyPatientCheckout() {

// ... logic to verify patient checkout ...

// Close patient appointment

closePatientAppointment()

}

}

// Object: Doctor

Doctor {

function checkSchedule(datetime) {

// ... logic to check schedule ...

// Confirm availability

confirmAvailability()

}

function provideTreatment() {

// ... logic to provide treatment ...

}

}

**Conclusion**

The submitted sequence diagram properly demonstrates the use of key concepts such as actors, objects, messages and message types, control focus, self-messages, and object destruction. It provides a clear narrative of this system process, from a patient's intention to book an appointment to the final checkout, giving an overview of the system's functionality and control flow.

**References**

Bell, D. (2004, February 16). IBM developer. https://developer.ibm.com/articles/the-sequence-diagram/

Fakhroutdinov, K. (2013, January 13). UML sequence diagrams. UML Diagrams - overview, reference, and examples. https://www.uml-diagrams.org/sequence-diagrams.html

Sequence diagram syntax and features. PlantUML.com. (n.d.). <https://plantuml.com/sequence-diagram>

Unhelkar, B. (2020). 12. In Software engineering with UML. essay, CRC PRESS.