COMP 6231 Technical Report

submitted in partial fulfillment
of the requirements for the
Assignment 3

by
Manfred Ramón Díaz Cabrera
Student ID: 40014085

Montréal, Québec
07/19/2016

OVERVIEW

The assignment demands the adaptation of the Staff Management System solution develop for Assignment 1 to Java JAX WS web service APIs.

GOALS

As per assignment specification, the following items delimit the scope of this work to:

1. Extract the Java client-server implementation by removing the CORBA specific code from your Assignment 2.
2. Properly annotate your Java implementation to adapt it as a web service.
3. Build the endpoint files using the wsgen command before publishing the service.
4. Import the wsdl files using the wsimport command.

SPECIFICATIONS

System requirements has been elicited from the assignment document and represent using *Behavior Driven Development*[1] notation, a simple representation for depicting users' stories and non functional requirements in a concrete way. These sections establish a common understanding for development, testing and deployment purposes.

Non Functional Requirements	
NFR1: Manage Records Transparently **As** a Client, **I want** Clinic Manager manage the records in their clinic only **So I** have control over the accessed information	**NFR2: System Auditing** **As** a Client **I want to** log all actions users perform **So I can** plan security accordingly
NFR3: Platform should be Java **As** a Client, **I want** the system implement in Java **So I can** have multi-platform capabilities	**NFR4: Java RMI for C/S Communication** **As** a Client, **I want** the communication between client and server uses Java RMI **So I can** simpler RMI invocation
NFR5: IP Stack for S/S Communication **As** a Client, **I want** the communication between servers uses low level TCP/UDP **So I can** decouple the servers	**NFR6: Deployment for three locations** **As a** Client, **I want** the system deployed in three locations **So I can** have a performance through resource sharing
NFR7: High Concurrency Degree **As** a Client **I want** the system designed to maximize concurrency **So I can** have a higher performance while executing the operations	**NFR8: Clinic Manager Identification** **As** a Client **I want** Clinic manager identified with their location plus a unique four-digits number **So I can** keep track of the operations performed

Functional Requirements	
FR1: Create Doctor Record **As a** Client Manager **I want to** create a doctor related record	**FR2: Create Nurse Record** **As a** Client Manager **I want to** create a nurse related record

[1] Dan North, Introducing BDD, Online: https://dannorth.net/introducing-bdd/, Accesed: July 2016.

So I can keep track of doctor's staff status	**So I can** keep track of nurses staff status
FR3: Get Record Count **As a** Client Manager **I want to** know all doctors or nurses records in all locations **So I can** keep track of the staff status	**FR4: Edit Record** **As a** Client Manager **I want to** edit a record field **So I can** keep the staff information updated
FR5: Doctors Record Id → [FR1] **As a** Client **I want** the doctor's records identified with the "DR" prefix plus a five digits number **So I can** properly identify each record	**FR6: Nurse Record Id → [FR2]** **As a** Client **I want** the nurses records identified with the "NR" prefix plus a five digits number **So I can** properly identify each record
FR7: Transfer Record **As a** Client **I want to** be able to transfer a record **So I can** keep track of my staff movements	

SOLUTION

Domain Model

In order to fulfill the functional requirements, a domain model was designed in Assignment 1 and 2 reflecting the major concepts in the discussion domain for the assignment as depicted in **Figure 1**. No major changes have been introduced in the domain model to fulfil the new requirements.

Architecture

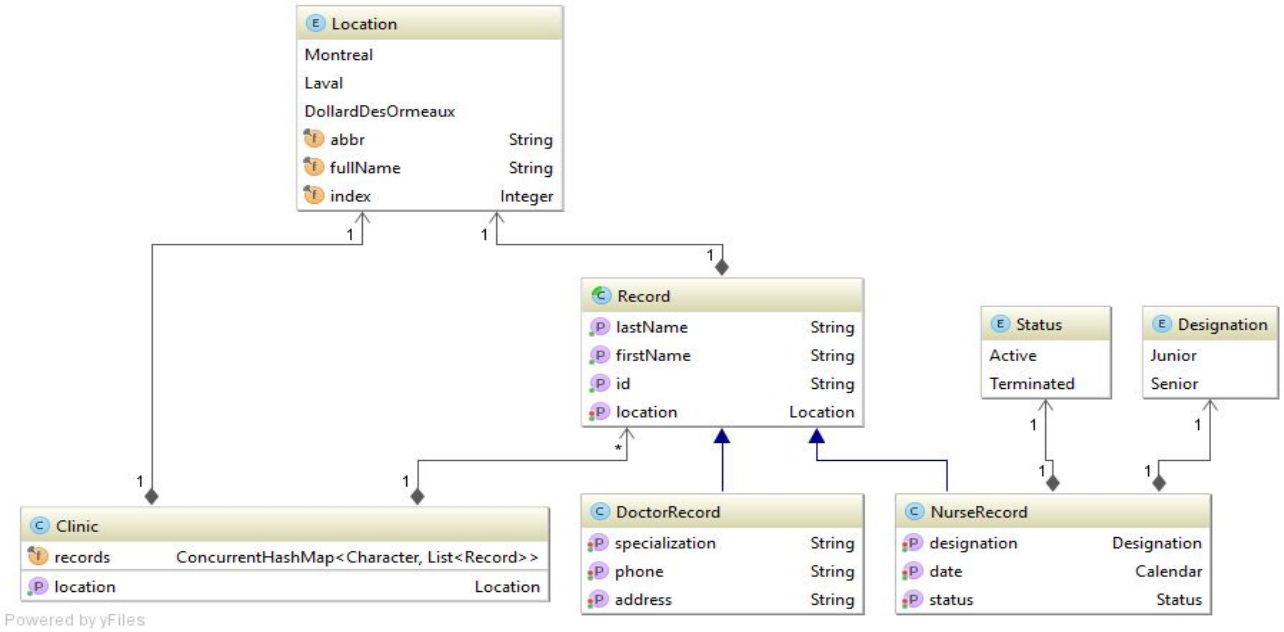
Using as a starting point the solution presented for Assignment 2, some changes have been carried out in order to accommodate the new requirements in the previous architecture.

In the server side of the solution a new model has been presented in order reuse the existing implementation of a new IPC technologies. Since Assignment 2, server side features were abstracted into a *ClinicServer* concept from which *ClinicServerRMI* and *ClinicServerCORBA* inherited and now *ClinicServerWeb* also inherits, maintaining the translation features (reply/response messages

conversion) to the JAX WS web service interfaces. Client side solution has followed the same pattern, a new endpoint manager has been added -*ClientWebEndpoint*- in order to provide the abstraction for querying the new web interface.

Given all this, a new component has been added -*web*- which now encapsulates the necessary implementation for providing web services using JAX-WS.

Figure 1. Domain Model



Client-Server Communication

As per explicit requirements, Java JAX-WS Web Services APIs was used for client-server communication. Java JAX-WS Web Services APIs offers several options for generating interprocess communication code. One of this features is the on-the-fly generation of the WSDL and the client side proxy required for client-server communication.

Table 1 Component and Responsibilities

Component	Brief Responsibility Description
shared	Contains all cross-cutting models, concerns (security, base logging, exception handling) and representations (interfaces, messaging).
server	Encapsulates all server related features: clinic server implementation, UDP multicast server-server communication.
client	Encloses the client UI, user Access Transparency Layer (Session class), both CORBA and RMI can be selected as IPC technologies.

rmi	Implements RMI communication logic from Assignment 1
corba	Contains CORBA server side implementation.
web	Implement JAX-WS interoperation layer.

Figure 2 depicts the model for the web service migration according to the Assigment 3 requirements. *ClinicServerWeb* class has been marked with the appropriate annotations provided by Java JAX-WS APIs. Exposition of the interface for consumption has been carried out by the ClinicServerWebEndpoint using the provided *java.xml.ws.Service* class.

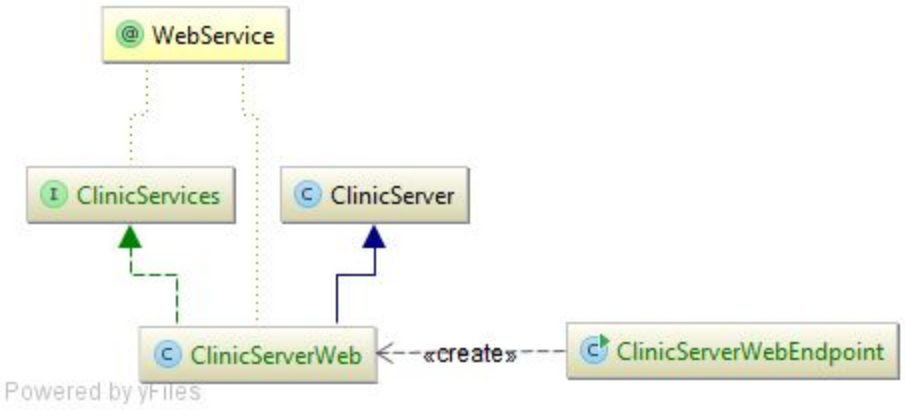


Figure 2 Web Service Implementation

Server-Server Communication

No major changes were introduced in the server-server communication layer which was fully reused for the purpose of this assignment and remains with the model depicted in **Figure 3**.

Development environment

For developing purposes several tools have been used to facilitate the whole SDLC. The tools listing can be found in **Table 3** that depicts the tools, versions and the purpose inside the development process.

Testing

The testing suite has been kept unchanged, validating this way the consistency of the migration to wbe services interface.

Figure 3 Multicast Communication Model

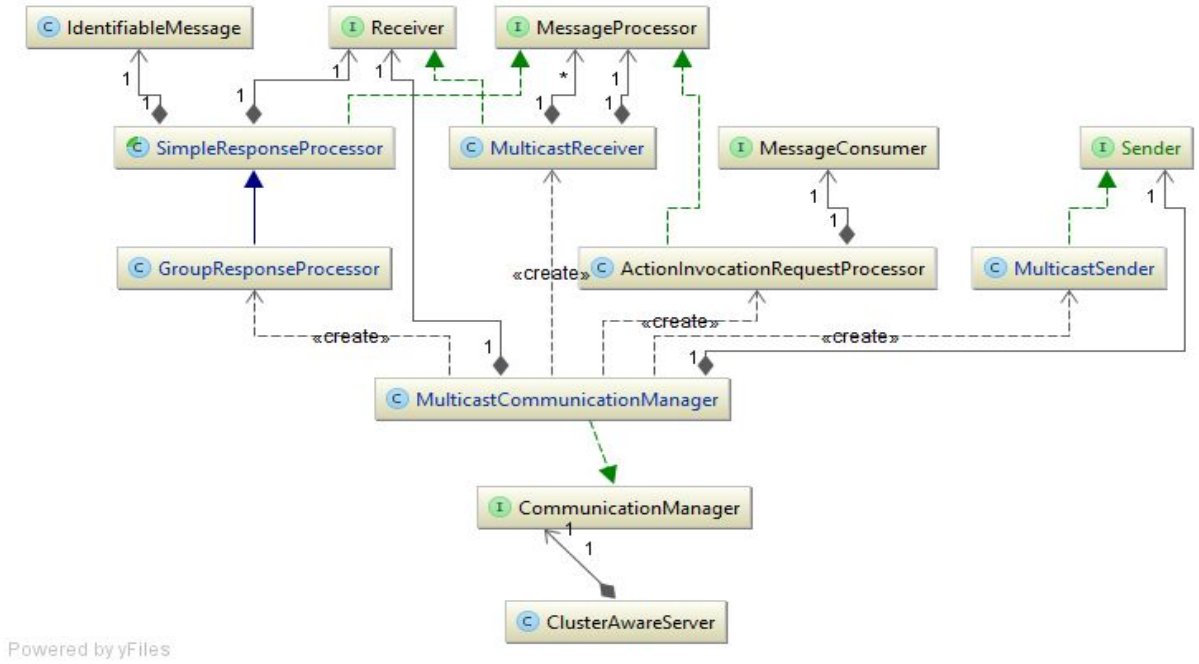


Table 3 Tools Listing

Tool	Version	Purpose
IntelliJ Idea	15.04	Integrated Development Environment
Gradle	2.13	Build System
junit	4.12	Test system

CONCLUSIONS

A few extensibility points have been left in the solution for future implementations and enhancements. Along with other considerations, like improving security (proper session management and reuse) and logging capabilities, these are the main place were some optimizations can be carried on to improve several aspects of the system like scalability and failure handling.