



The University of Calgary

Department of Electrical and Computer Engineering

Online electronic library Reservation Project Report

Student:

Sergey Kiselev

Instructor: Dr. Behrouz H. Far

Fall 2005

Table of Contents

Online Electronic Library Reservation System.....	3
<i>Abstract</i>	3
Introduction.....	3
System specifications.....	3
1. Business case.....	3
2. System Description.....	4
3. System description.....	4
4. Assumptions.....	5
5. Requirements.....	5
6. Wish List.....	5
System Design Documents.....	5
1. System Architecture.....	5
2. Agent Description.....	6
2.1 'Main' agent.....	6
2.2 Scheduler Agent.....	7
2.3 Library Agent.....	7
2.4 Appointment agent.....	7
3. Agent Internal Architecture.....	7
4. Technology overview.....	8
4.1 Understanding Web Services.....	8
4.2 SOAP.....	8
4.3 Disco and UDDI.....	9
4.4 WSDL.....	9
4.5 Communication protocol: SOAP.....	10
5. Detailed design.....	10
5.1 Use Case: Main Agent.....	10
5.2 Use Case: Scheduler Agent.....	17
5.3 Use Case: Library Agent.....	20
5.4 Use Case: Appointment Agent.....	22
6. Data Specification.....	23
6.1 Database Structure.....	23
7. Inter-Agents Messages.....	25
7.1 Inter-Agent Messages for the Main Agent.....	25
7.2 Inter-Agent Messages for the Scheduler Agent.....	27
Conclusions.....	28
Abbreviation.....	28
References.....	28

Student: Sergey Kiselev (ID: 298498)

Online Electronic Library Reservation System

Abstract

Online electronic library reservation (OELR) system is a multi-agent system which allows users to make appointment to work with the desired online electronic library. This system can be widely used in libraries, universities or organizations which have public computing access. It offers efficient management to work with online electronic libraries.

Introduction

Some schools don't have access to some specific online electronic libraries. But students of these schools need information from some online electronic libraries to prepare their presentations. OELR system is initiated to solve these problems. With this system, customers (students, staff of schools) can become the member of this system, find information which library from another school of city has access with the desired online electronic library and make appointment to the some available computer in selected library with ability to work with desired online electronic library in the desired date and time. The appointment will last during some time which user will define in the process to make an appointment. It provides better public service with reduced staff involvement.

This report includes the following documents:

- System specifications
- Design document
- Detailed design document
- Data dictionary
- Inter-agents messages

Each document represents the some corresponded step of my design.

System specifications

1. Business case

The library industry is increasingly becoming industry that doesn't exist in the border only one library. Libraries from another school or public local library system begin to merge. It means that user which registers and uses one library can order some books from libraries of different school. In this case user doesn't know it (he/she doesn't care about it). Actually, this feature is hidden from user. From the second side, the each library has public computers that can be used by members of this library or members of this library system. In this case I have question. Why cannot user (for example: student of the SAIT)

work with facilities and services that libraries of UofC provide? Under facilities and services I mean available /challenge to work with online electronic libraries that the given library has access.

2. System Description

This system will help to solve management library problems with access to the online electronic libraries.

3. System description

- The proposed OELR System is a multi-agent system designed for main purpose:
 - to make appointment to work with one computer of some library, which access with the desired online electronic library. Each user of this system has user's profile package
- The user's profile package is composed of:
 - a user name
 - user surname
 - address
 - email
 - phone
 - status of student
 - name of school
 - preference of online electronic library name
 - preference of library location

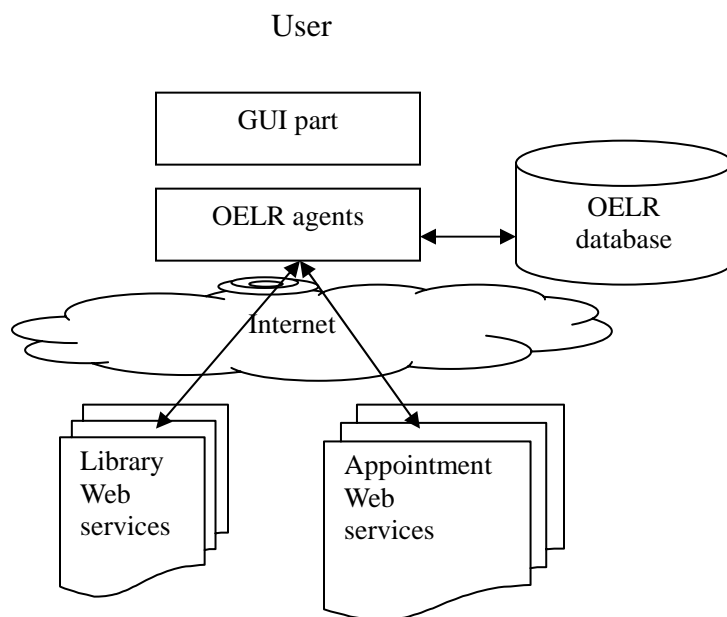


Fig 1. Common view of system description

4. Assumptions

- The OELR system maintains a user profile where each client has his or her own individual preferences, such as:
 - preference of online electronic library name
 - preference of library location
- The user profile includes the personal information, such as:
 - a user name
 - user surname
 - address
 - email
 - phone
 - status of student
 - name of school
- The user makes the appointment through a user interface
- There are already predefined Web services on the Internet which the OELR system makes use of. These web services have the same WSDL description meaning their input and output messages have a standard syntax.

5. Requirements

- The OELR system shall provide to make appointment with specific online electronic library
- The OELR system shall deal with another (different) libraries
- The OELR system shall make the appointment upon the approval by the user
- The OELR system shall send memo to the user about the appointment information, alerts user before the appointment and informed the user that hold of book is available. The period before which system shall remind the user is specified in the user profile

6. Wish List

- The Web services, that OELR system is used, is not designed in this project
- The Agents should use lookup services to locate Web services
- The OELR system shall allow the user choose another libraries from another cities

System Design Documents

1. System Architecture

- This system is multi-agent one between client and existing Web Services
- The Web services come from one provider or more providers and return quotations of their service area to the agents based on client input and user profile
- The agents try to sort the results in a way that user gets the nearest location of library that has access with the desired online electronic library

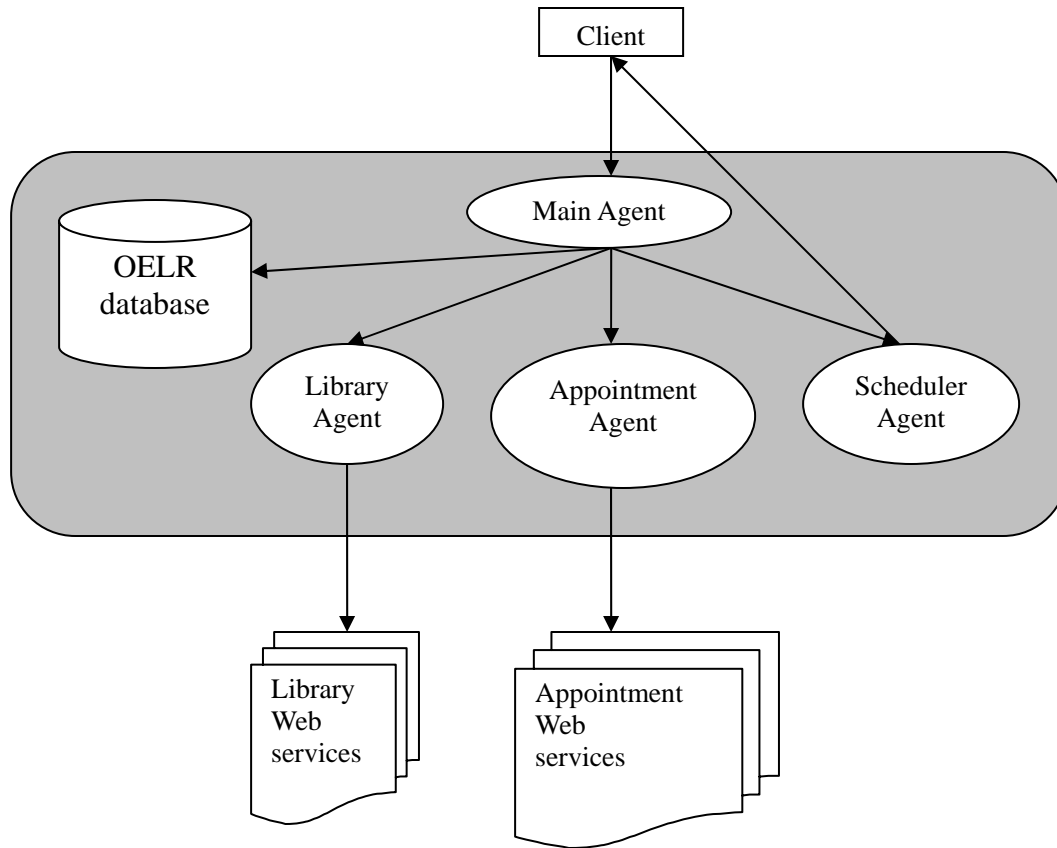


Fig2. System architecture

- This system is set up on the public computer in the library. Each student goes to the library to have opinion to work with the desired online electronic library. But the given library doesn't have access with it. So user launches the OELR system and submits request on through GUI part of EOLR system
- Any user that wants to use the OELR system services has to create his/her own user profile before hand
- The multi-agent system has to interact with Web Services to get the user a 'best' (the nearest) library location

2. Agent Description

2.1 Main agent

- The main agent intercepts the request by the user through GUI part of OELR system. In our architecture the Main agent handles the correspondence with all other agents and in this way we have a central point of delegation
- The Main agent requests to get list of all available online electronic libraries, to get list of all libraries which access with desired online electronic library, get list of all available computers which located in the selected library in some date, select computer and make appointment to work on this computer duration of

- appointment time with the selected online electronic library
- The Main agent appointment information to the Scheduler Agent.

2.2 Scheduler Agent

- The Scheduler Agent communicates with the Main Agent
- The Scheduler Agent gets the appointment information from the Main Agent
- The Scheduler Agent prepares email with appointment information and sends it to user
- If the given user email doesn't exist, the Scheduler Agent informs the Main agent

2.3 Library Agent

- This agent is responsible to provide list of all available libraries, that user can use with the desired online electronic library
- The Library Agent sorts the list of all available libraries
- The Library Agent highlights the libraries that are the nearest to user's school(user's library)

2.4 Appointment agent

- This agent gets request from user to provide a list of all available computers of selected library which have access with the selected online electronic library in some specified date
- This agent returns the list of all available computers
- After as user selected one computer for specified date and time, the agent makes appointment for user and sends to user confirmation number of appointment.

3. Agent Internal Architecture

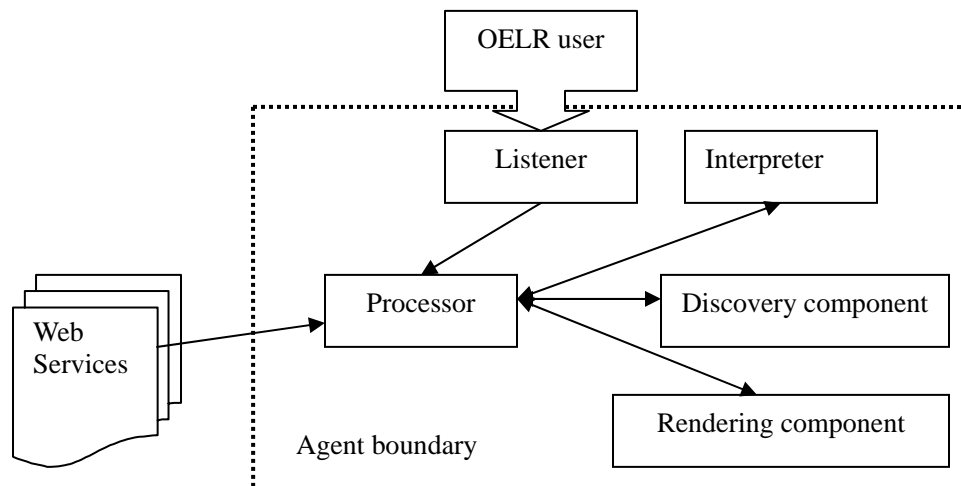


Fig. 3 Agent Internal Architecture

The Agent consists of the following parts: Listener component, Processor component, Interpreter component, Discovery component and Rendering component.

Listener component is listens a default port for any incoming requests from OELR application.

Interpreter component read and parse, validate the XML message.
We assume that all agents have agreed on a Document Type Definition (DTD).

Processor component receives an XML document as an input. It uses the Interpreter to parse document and call the necessary functions to run a process.

Discovery component provides the service discovery base-service (a superset of UUDI).

Rendering Agent is used by Processor to render data before sending it back to the calling function.

4. Technology overview

Web services are part of a natural progression:

- Object-oriented languages such as C++ and C# enable two objects within the same application to interact
- Protocols such as the Component Object Model (COM) enable two objects on the same computer, but in different applications, to interact
- Protocols such as the Distributed Component Object Model (DCOM) enable two objects on different computers, but in the same local network, to interact
- Web services enable two objects of different computers – even if they're only connected by the Internet – to interact

4.1 Understanding Web Services

The key to understand Web Services is to know the protocols that make them possible:

- Simple Object Access Protocol (SOAP)
- Disco and Universal Description, Discovery, and Integration (UDDI)
- Web Services Description Language (WSDL)

By default, all communication between Web services servers and their clients is through Extensible Markup Language (XML) messages transmitted over the Hypertext Transfer Protocol (HTTP). This has two benefits. First, because Web services messages are formatted as XML, they're reasonably easy for human beings to read and understand. Second, because those messages are transmitted over the pervasive HTTP, they can normally reach any machine on the Internet without being blocked by firewalls. [1]

4.2 SOAP

For Web services to manipulates objects through XML messages, there has to be a way to translate objects (as well as their methods and properties) into XML. SOAP is a way to encapsulate objects calls as XML sent via HTTP.

The SOAP has some obvious points:

- The SOAP message consist of an envelope and a body, each marked with a specific XML tag
- The particular message invokes a method named getLocation() from a specified uniform resource locator (URL)
- The method takes a single parameter, arg0, which is transmitted as XML element
- In the response message, the getLocationResponse element is the result of the call to the object on the server. It includes a string wrapped as an XML element [1]

4.3 Disco and UDDI

Before we can use a Web service, you need to know where to find the service. Handling such requests is the job of several protocols, including Disco and UDDI. These protocols enable you to communicate with a Web server to discover the detailed of the Web services that are available at that server.

Disco is a Microsoft standard for the creation of discovery documents. A Disco document is kept at a standard location on a Web services server and it contains paths and other information for retrieving useful information, such as the WSDL file that describes a service. This document is used for static discovery. That is, a potential user of your Web service must know location of the Disco document to use it.

UDDI is a method for finding services by referring to a central directory. These can be Web services, URLs for information, or any other online resources. UDDI registries are sites that contain information that is available via UDDI, you can search such a registry to find information about Web services. UDDI thus provides dynamic discovery, allowing you to find Web services even when you do not know their location in advance.

UDDI registries come in two forms: public and private. A public UDDI registry is available to all comers via the Internet and serves as a central repository of information about Web and other services for businesses. A private UDDI registry follows the same specifications as a public UDDI registry but is located on an intranet for the use of workers at one particular enterprise.

To add your own services to a UDDI registry, you must use the tools provided by the particular registry.

4.4 WSDL

Another prerequisite for using a Web service is knowledge of the SOAP message types that it can receive and send. You can obtain this knowledge by parsing WSDL files. WSDL is a standard by which a Web service can tell clients what messages it accepts and which results it will return.

WSDL files define everything about the public interface of a Web service, including the following:

- The data types that it can process

- The methods that it exposes
- The URLs through which those methods can be accessed

4.5 Communication protocol: SOAP

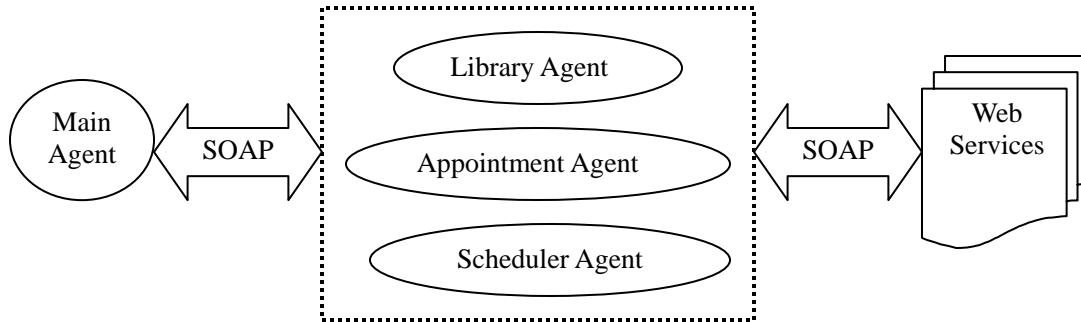


Fig.4 Communication protocol view

5. Detailed design

Use cases and use case definition for all the participating agents are documented further.

5.1 Use Case: Main Agent

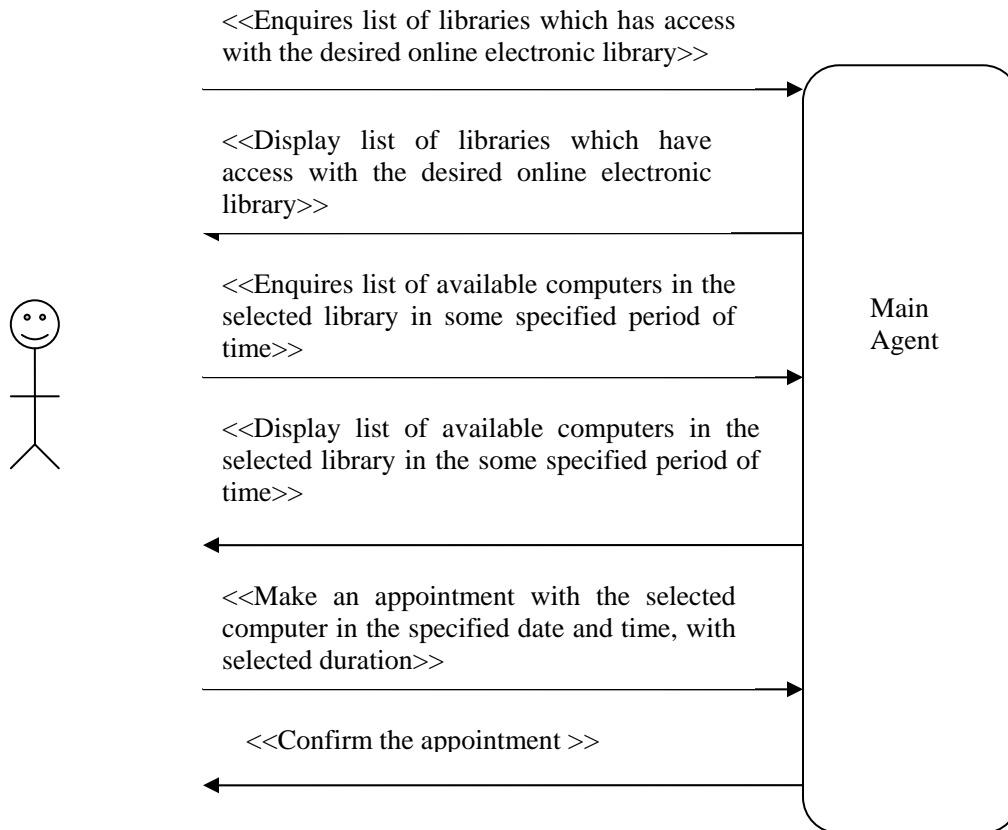


Fig.5 Use Case of Main Agent

5.1.1 Use Case Definition: The Main Agent

Brief Description:	The Actor uses this use case to make an appointment with the some computer which is in the library that has access with the desired online electronic library	
Precondition(s):	User profile is created before any service	
Post condition(s):	If all the rules are successfully met, than actor will be able to avail the facilities provided by the OELR system	
Process steps:		
1	Actor asks the list of libraries which have access with the desired online electronic library	
2	The Main Agent requires to get the list of all libraries which access with the desired online electronic library from the Library Agent	
3	The Library Agent returns the list of all libraries which have access with the desired online electronic library	
4	The Main Agent displays the sorted list of libraries. The Main Agent highlights the library which is the nearest one to the preferable library	
5	The actor selects the library from the list and asks to the Main Agent to get the list of all available computers and specifies the period of time (date and time)	
6	The Main Agent requests to the Appointment Agent to get the list of all available computers which are in the chosen library in the specified period of time	
7	The Appointment Agent returns the list of all available computers for specified period of time	
8	The Main Agent displays the list of available computers and highlights the computer with the nearest date and time to the desired one by actor	
9	The actor chooses the one computer and make appointment with chosen computer with date, time and duration of appointment	
10	The Main Agent sends this appointment information to the Appointment Agent	
11	The Appointment Agent makes the appointment for the actor and confirms the appointment. It comes back confirmation notification with number of the appointment (ID of the appointment) to the Main Agent	
12	The Main Agent writes appointment information into the OELR database and displays this information to the user browser.	
13	The Main Agent sends the appointment information to the Scheduler Agent to send it to user by email	
Exceptions		
1a	System can not find the library which has not access with the	Error message is generated. Use Case is not terminated. The

	desired online electronic library	system recommends to Actor to check the spelling the name of online electronic library.
6a	There are not available computers in the given period of time	Error message is generated. Use Case is not terminated. The system recommends to Actor to change the date and time of the desired appointment
13a	The Scheduler Agent gets an error notification that the given user doesn't exist	Use Case is terminated
Relationships:		
Initiating	Actor	
Collaborating	Scheduler Agent, Library Agent, Appointment Agent	
Other Diagrams:		
Data requirements:		
Data Required:	Data required for Main Agent: Online electronic library name Library preference Appointment information: <ul style="list-style-type: none"> • number of appointment • library location, which has access to the desired online electronic library • computer location and number of computer 	

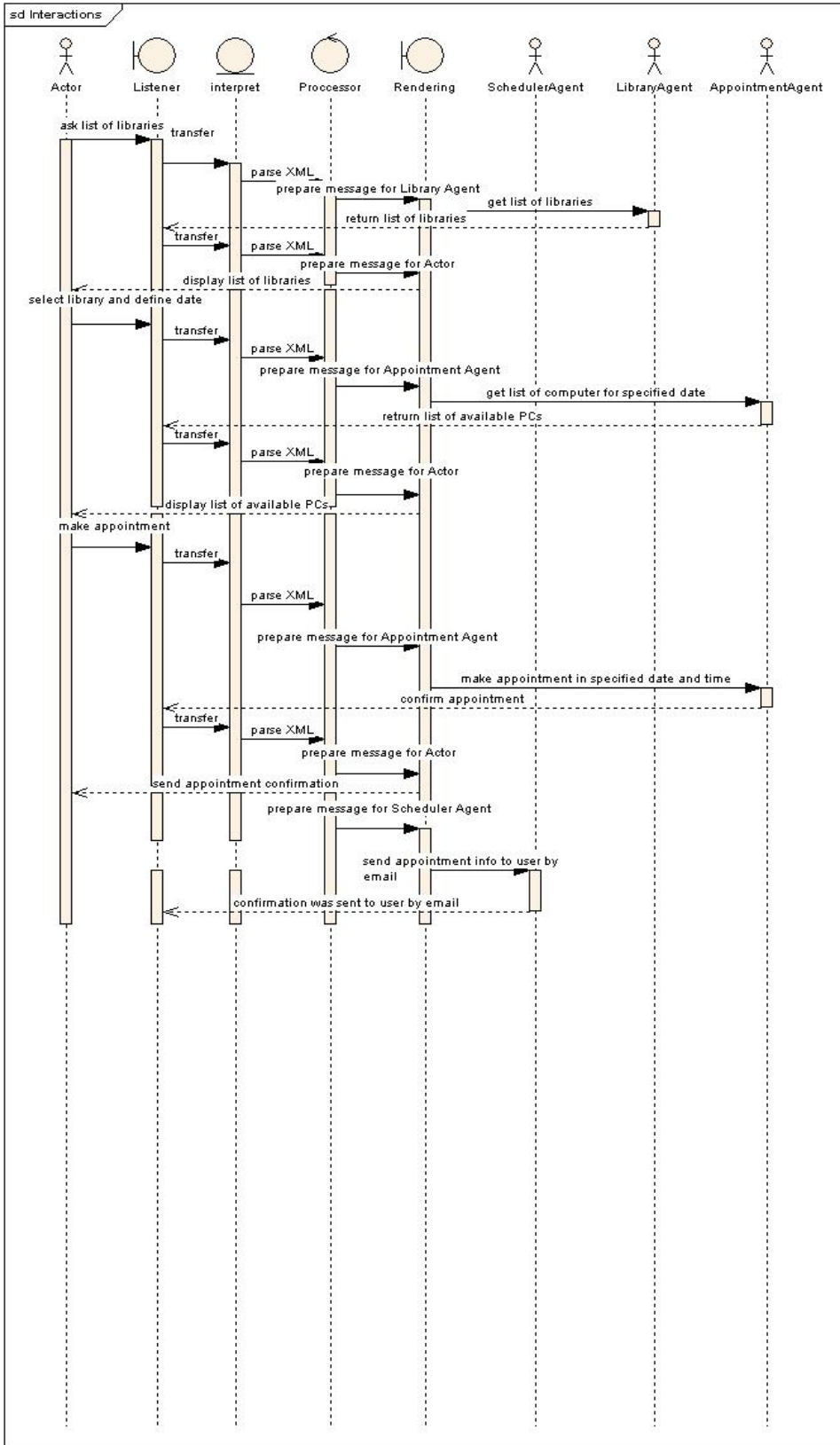


Fig.6 Sequence diagram of Main Agent (without command to write new appointment into the OEL database).

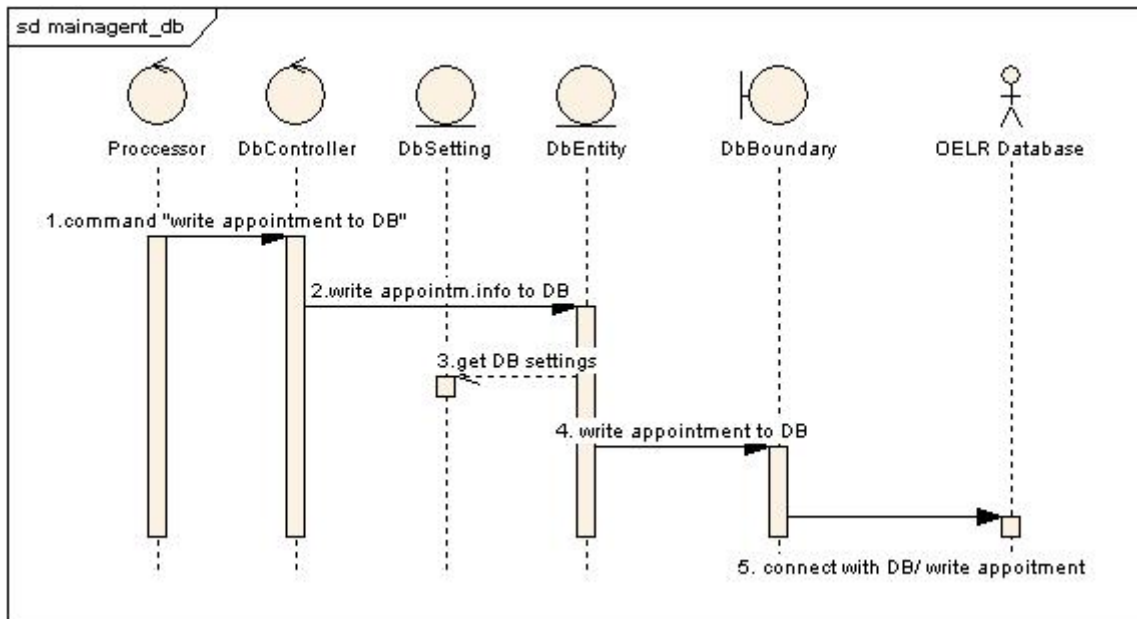


Fig.7 Sequence diagram of Main Agent. It describes that situation when Main Agent got the confirmation number of appointment from the Appointment Agent and the all necessary classes, such as: library, oel, computer and appointment were filled. This sequence diagram shows interconnection between the Processor (controller), DbContoller and depicts the whole process chain of writing the new created appointment into the OELR database.

The next diagram is the class diagram of the Main Agent. The Main Agent consists of the following classes:

- boundary classes: listener, render and DbBoundary. They works directly with different actors of system, such as the Actor(user) and other Agents of system. The listener listens the port and is ready to get XML (SOAP) message. Render prepares the XML (SOAP) response of the Main Agent to another agent, such as: Scheduler, Library and Appointment Agents. The DbBoundary is boundary class which connects with OELR Database.

Note: I think that the Main Agent should include the **listener** boundary class, because the the Main Agent will consist of from two separate software applications, such as: GUI part and Main agent. (I don't design of GUI part in this document, because GUI library is not specified for this application.) They will interact though SOAP message.

- Controller classes: processor and DbController. The processor gives commands which are necessary to provide the whole cycle of the Main Agent.

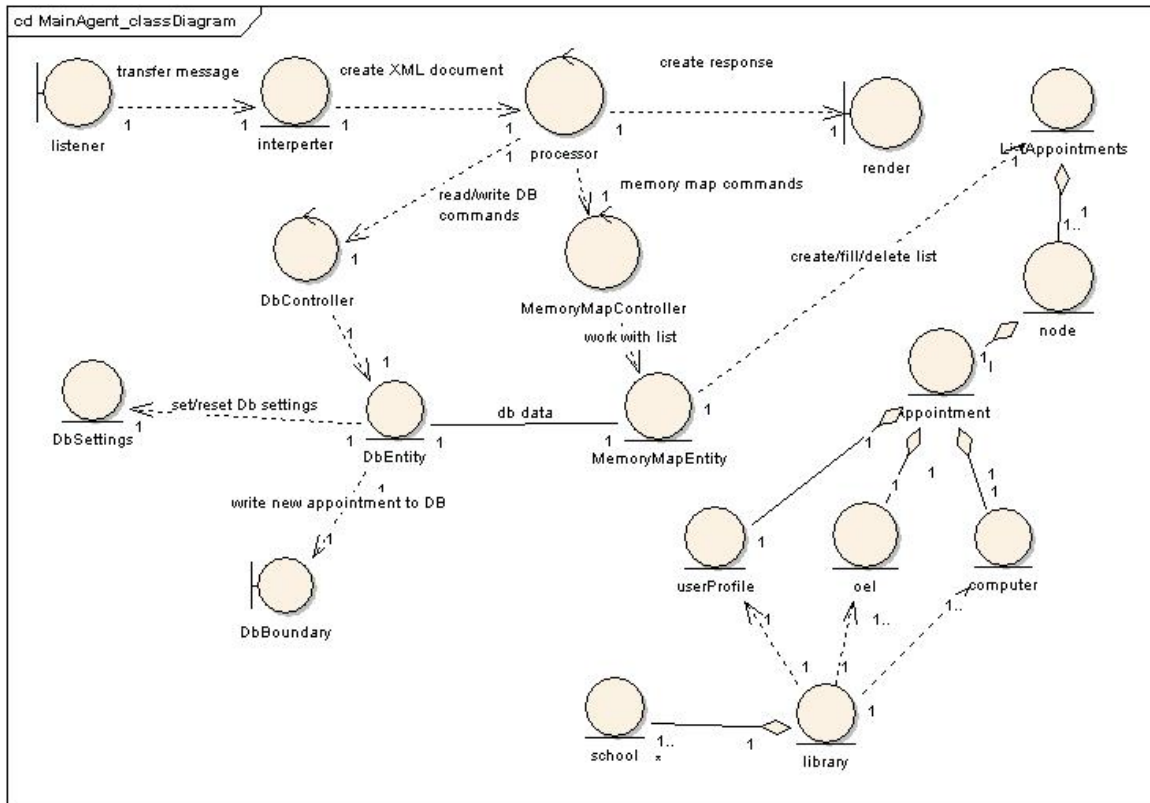


Fig. 8 The class diagram of Main Agent.

The second control class is the DbController. It gets commands from the **process** controller if it is necessary to connect with OELR database and write/read some information from database. Further DbController sends command to the DBEntity.

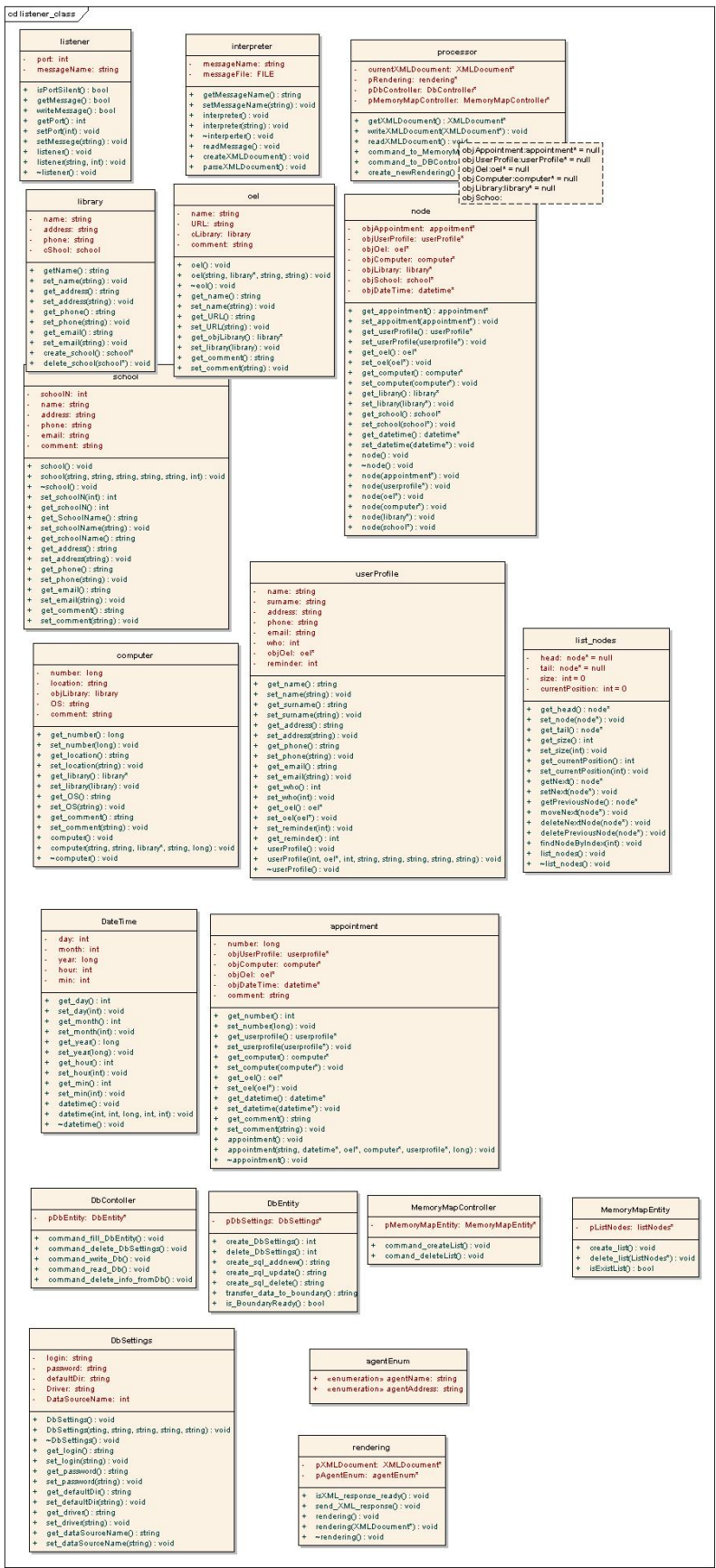
The third control class is MemoryMapController one. It is responsible to manage and control memory mapping operations. It sends commands to a MemoryMapEntity class which is responsible to work with list.

The Main Agent has the following Entity classes: DBEntity, DbSettings, school, library, userProfile, oel, computer, Appointment and ListAppointment classes.

The DBEntity class gets commands from DBController to connect with OELR database to write or read some information from the database. DBEntity class launches the DbSettings entity class, which knows all necessary access connection parameters, gets these connection parameters and is ready to work with database.

The classes: school, library, userProfile, oel, computer and Appointment entity classes need to keep specific information about school, library, user, oel and Appointment in the memory of computer. The detailed meaning of these classes will be understood from structure of Database (Data Specification).

If there is more than one instance of these classes, it will be created list. The current version of design doesn't explain the very useful features for the Main Agent, such as: to history of appointment for one use or get statistic report about appointment for some specific period of time. But real application will require these features, in these case I add ability to create list of appointment (ListAppointments Entity Class).



5.2 Use Case: Scheduler Agent

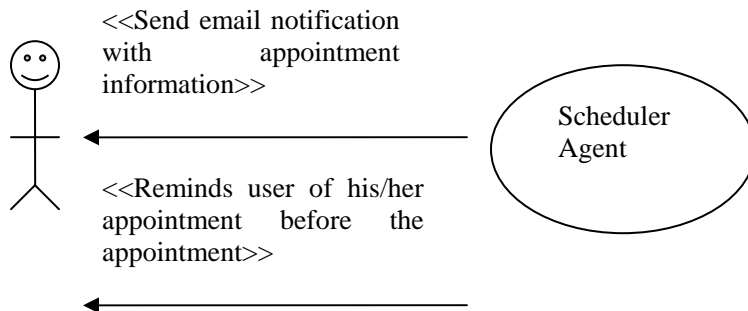


Fig.9 Use Case for Scheduler Agent

Brief Description:	The Actor uses this use case to handle conflicts and the Scheduler uses this use case to remind to user about appointment	
Precondition(s):	User profile is created before any service User made the appointment	
Post condition(s):	If all the rules are successfully met, than actor will be able to avail the facilities provided by the OELR system	
Process steps:		
1	The Scheduler Agent sends the email notification with the appointment information	
2	The Scheduler Agent reminds user of his /her appointment before the appointment, for example: before 3 hours before appointment	
Exceptions		
2a	System gets the error message that this email doesn't exist	Use Case is terminated.
Relationships:		
Initiating	The Scheduler Agent	
Collaborating	Actor	
Other Diagrams:		
Data requirements:		
Data Required:	Data required for Scheduler Agent: Appointment information	

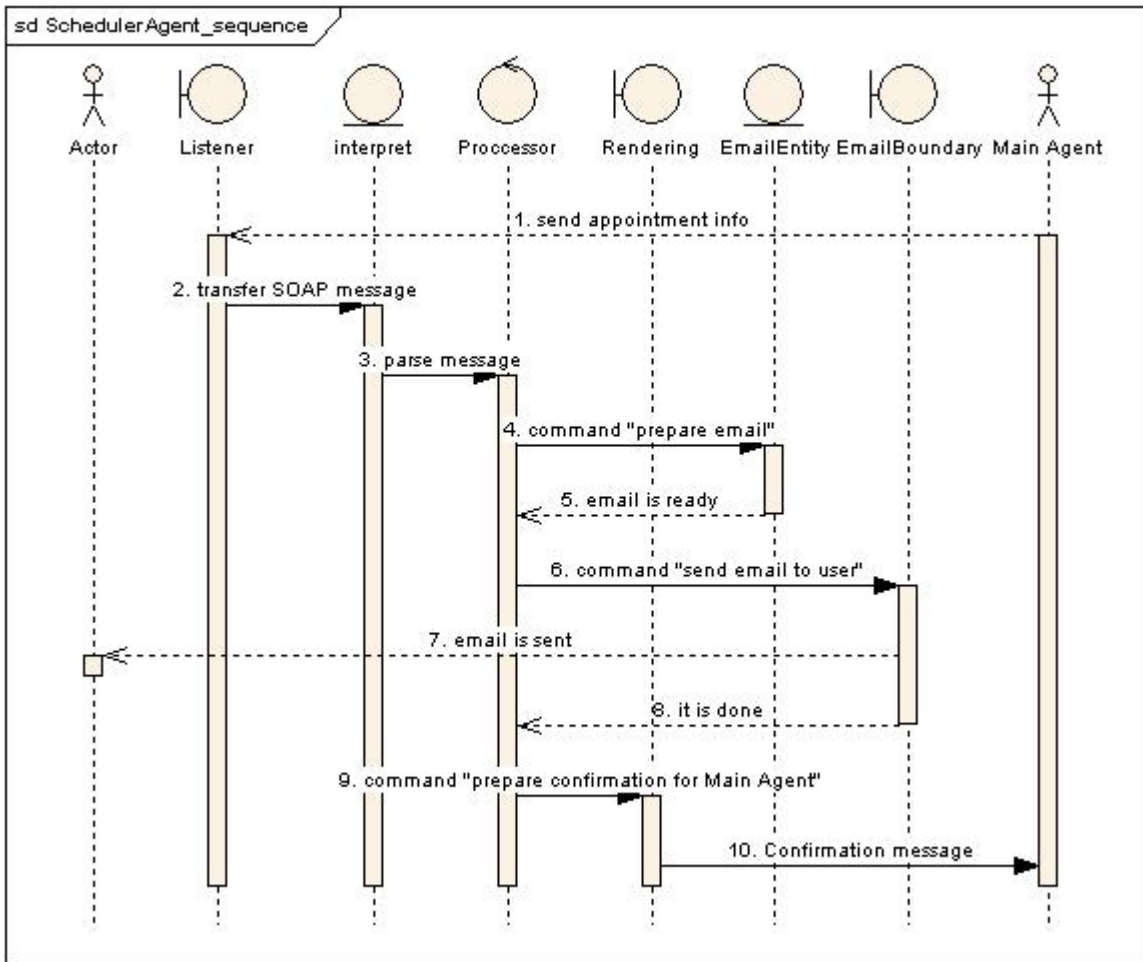


Fig. 10 The sequence diagram of the Scheduler Agent

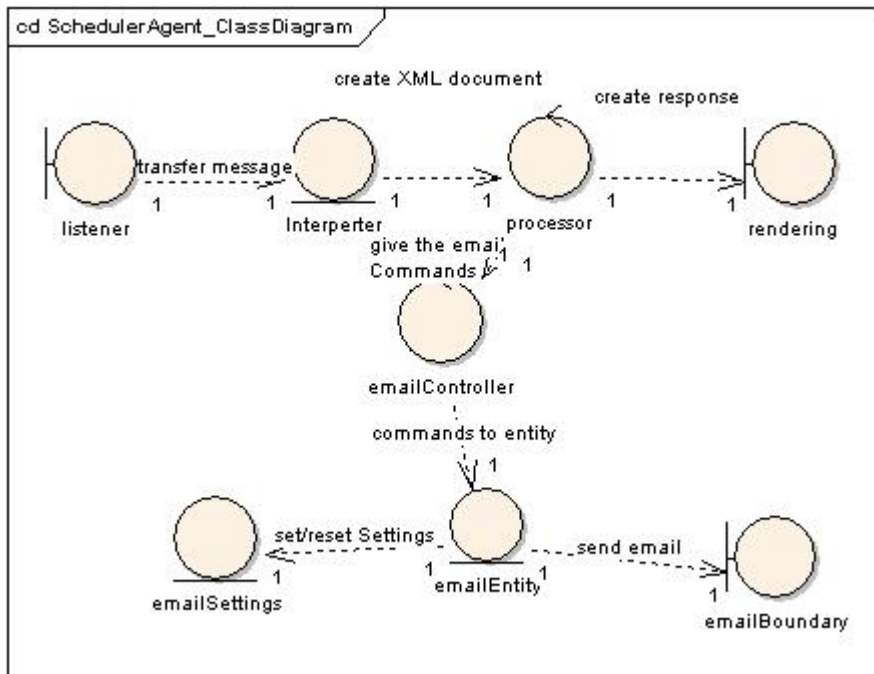
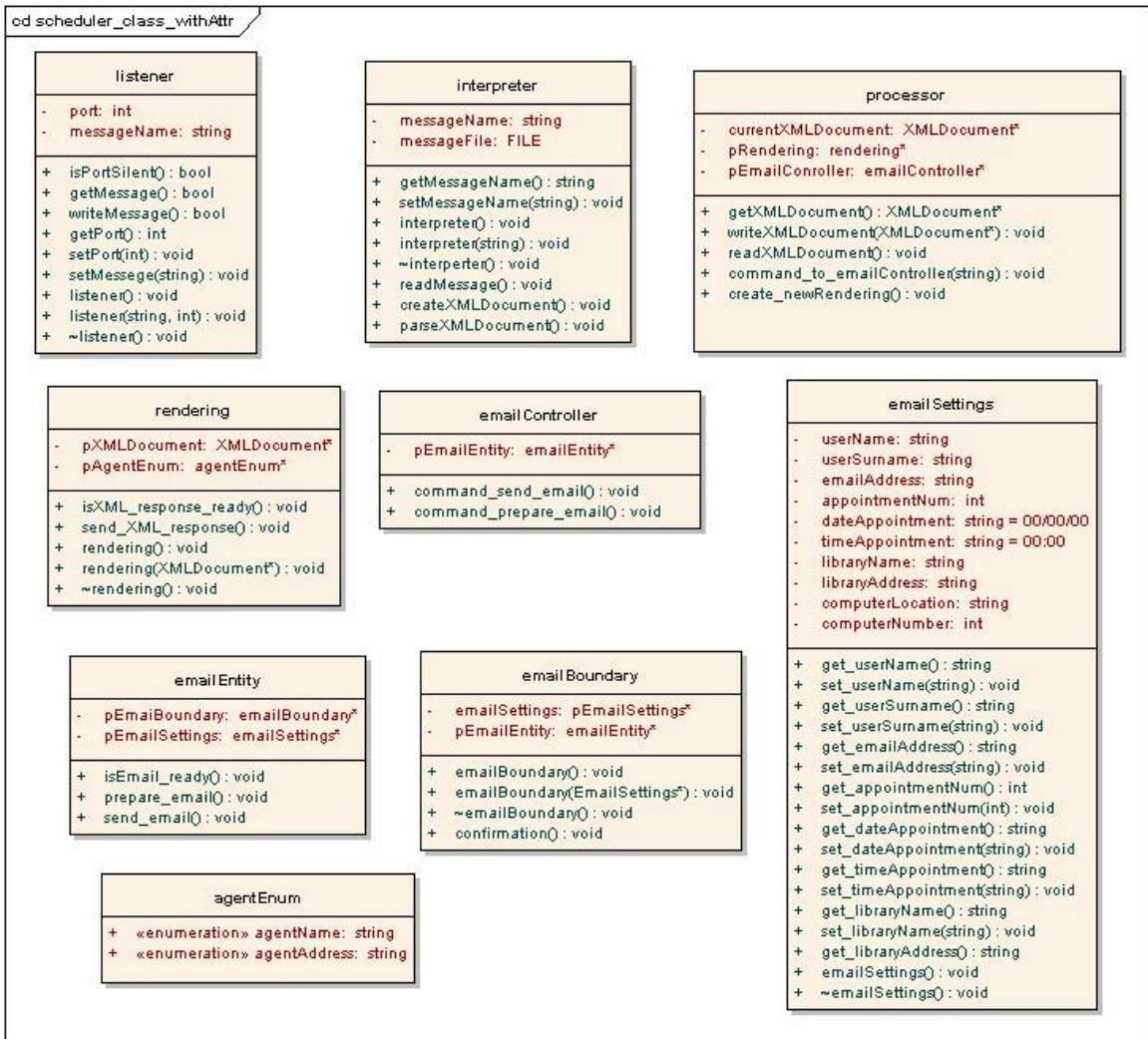


Fig.11 The class diagram of the Scheduler Agent

The Scheduler Agent consists of the following classes:

- Boundary classes: listener, rendering and emailBoundary. The listener listens the port and get SOAP message, transfer it to Interpreter (Entity class). Rendering class prepares SOAP response. EmailBoundary class is boundary class which sends the ready email with appointment information to user. There are two different variations. First one is when the email has the new-making appointment information and second version is when the email reminds the appointment information before the appointment according to the user preference (for example: before 3 hours).
- Controller class: the Processor and emailController classes. Processor class is the responsible to control the cycle of working the Scheduler Agent. According to the working cycle it gives necessary commands to other classes. The emailController class gives commands to emailEntity class to prepare email to user and send it to the user.
- Entity class: EmailEntity class. It gets command from the emailController class to prepare the email to the user with appointment information, which EmailEntity can get emailSettings class. The emailSettings entity is filled by emailEntity class after the the Scheduler agent gets message from the Main agent.
The EmailSettings has all necessary attributes: appointment number, user's name, user's surname, oel name, library location, computer number, computer location, date and time of appointment and duration of appointment.



5.3 Use Case: Library Agent

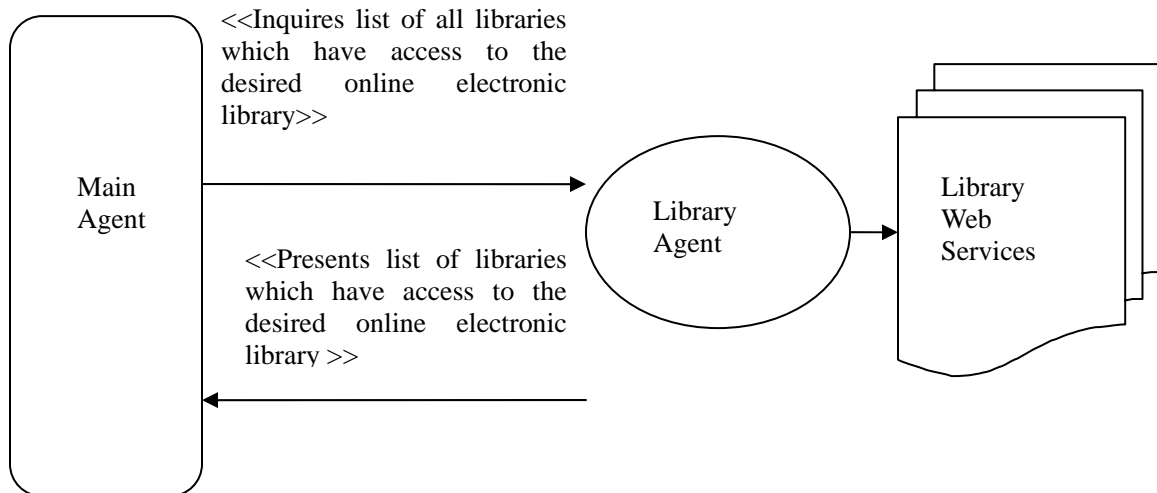


Fig. 12 Use Case for Library Agent

Brief Description:	The Main Agent uses this use case to get the list of libraries which get access to the desired online electronic library	
Precondition(s):	Actor asked the OEL system about the desired online electronic library	
Post condition(s):	User will get list of libraries which have access to desired online electronic library.	
Process steps:		
1	The Main Agent asks the Library Agent about list of all libraries which have access to the desired online electronic library	
2	The Library Agent returns the list of libraries which have access to the desired online electronic library	
Exceptions		
2a	The Library Agent exists the empty list	Error message is generated. Use Case is not terminated. The system recommends to check spelling the desired online electronic library
Relationships:		
Initiating	The Main Agent	
Collaborating	The Library Web Services	
Other Diagrams:		
Data requirements:		
Data Required:	Data required for Library Agent: Name of desired online electronic library	

5.4 Use Case: Appointment Agent

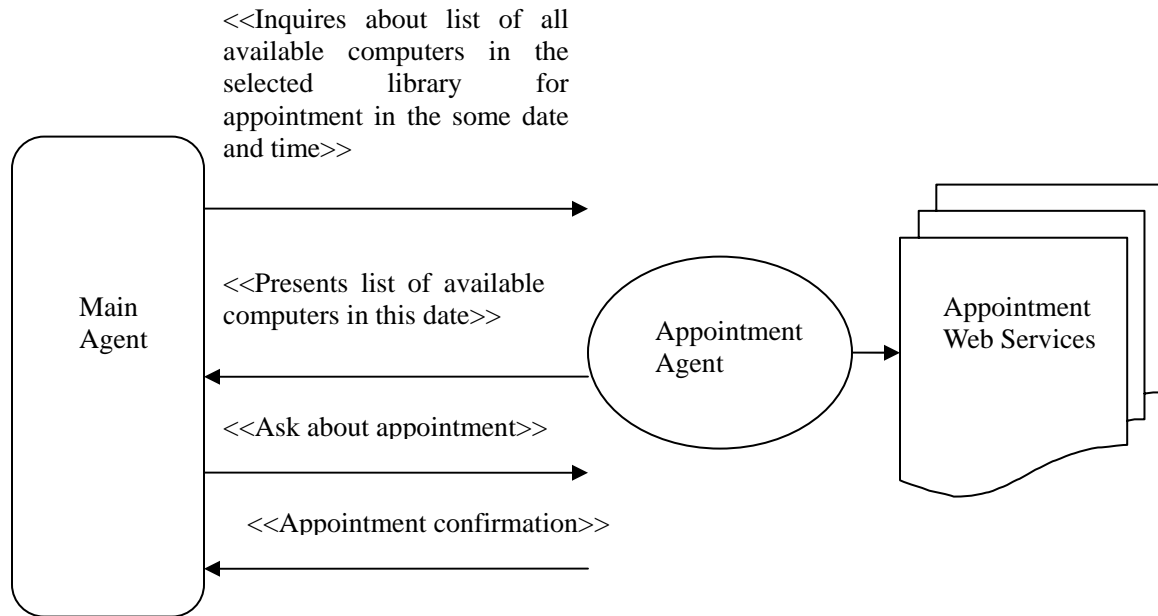


Fig.13 Use Case for the Appointment Agent

Brief Description:	The Main Agent uses this use case to make appointment	
Precondition(s):	Actor asked the OEL system to make appointment	
Post condition(s):		
Process steps:		
1	The Main Agent asks the Appointment Agent about list of all available computers in the selected library to make appointment. The Main Agent specifies the appointment date	
2	The Appointment Agent returns the list of all available computers in the selected library in this date	
3	The Main Agent makes appointment with selected date, time of appointment and appointment duration	
4	The Appointment Agent confirms the appointment. It sends the number of appointment, number and location of computer, date and time of appointment and appointment duration	
Exceptions		
2a	The Appointment Agent returns the empty list	Use Case is not terminated. The system recommends the user to change date/time of appointment
Relationships:		
Initiating	The Main Agent	
Collaborating	The Appointment Web Services	
Other Diagrams:		

Data requirements:	
Data Required:	Data required for Appointment Agent: Name of online electronic library

6. Data Specification

6.1 Database Structure

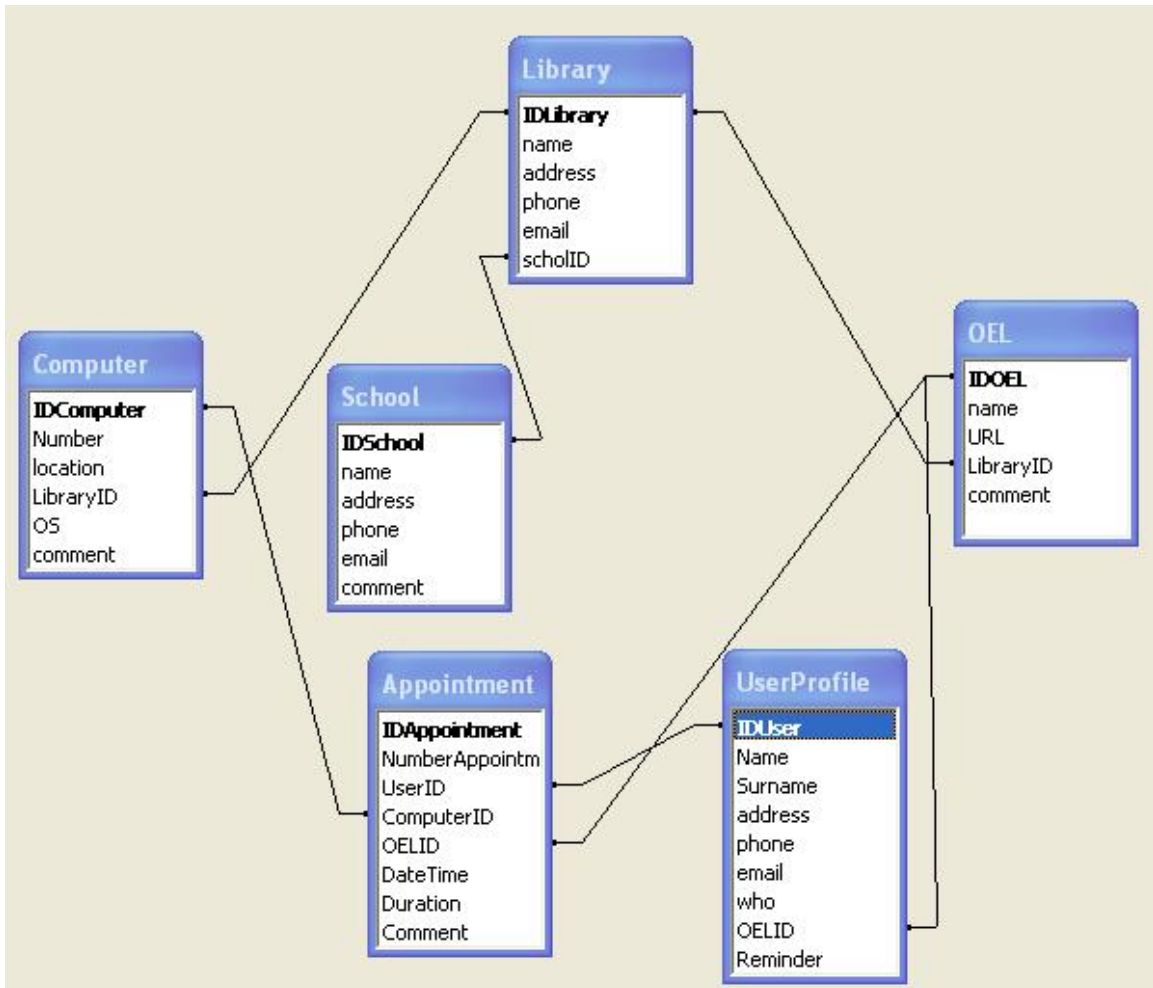


Fig. 14 Structure of OELR database

The given structure consists of from next tables:

- UserProfile. This table keeps user information, such as:

Field	Description	Type
IDUser	Primary key	AutoNumber, Increment, No duplicates
name	User's name	varchar(30)
surname	User's surname	varchar(30)
address	User's address	varchar(30)

phone	User's phone	Varchar(12)
email	User's email	Varchar(20)
who	0= Student 1= Staff 2= Other	integer
OELID	User's preference of OEL	Foreign key (for relation with OEL table)
Reminder	how many hours before appointment to remind user	integer

- School table. This table keeps school information

Field	Description	Type
IDSchool	Primary key	AutoNumber, Increment, No duplicates
name	School name	varchar(30)
address	School address	varchar(30)
phone	School phone	Varchar(12)
email	School email	Varchar(20)
Comment	Some additional information	varchar(30)

- Library table. This table keeps Library information

Field	Description	Type
IDLibrary	Primary key	AutoNumber, Increment, No duplicates
name	Library name	varchar(30)
address	Library address	varchar(30)
phone	Library phone	Varchar(12)
email	Library email	Varchar(20)
schoolID	Library relation with school	Foreign key (for relation with School table)

- OEL table. This table keeps OEL information

Field	Description	Type
IDOEL	Primary key	AutoNumber, Increment, No duplicates
name	OEL name	varchar(30)
URL	OEL URL	varchar(30)
email	Library email	Varchar(20)
Comment	Some additional information	varchar(30)

- Computer table. This table keeps Computer information

Field	Description	Type
-------	-------------	------

IDComputer	Primary key	AutoNumber, Increment, No duplicates
number	Computer number	long
location	Computer location	varchar(30)
libraryID	computer relation with library	Foreign key (for relation with library table)
OS	Computer OS	Varchar(12)
Comment	Some additional information	varchar(30)

- Appointment table. This table keeps current appointment information, such as:

Field	Description	Type
IDAppointment	Primary key	AutoNumber, Increment, No duplicates
NumberAppointment	Number of Confirmation Appointment	long
userID	Appointment has relation with UserProfile table	Foreign key (for relation with UserProfile table)
computerID	Appointment has relation with Computer table	Foreign key (for relation with Computer table)
OELID	Appointment has relation with OEL table	Foreign key (for relation with OEL table)
DateTime	Date and Time of Appointment	DateTime
comment	Place for some other information	Varchar(30)

7. Inter-Agents Messages

As discussed in the Design document, SOAP will be used as a protocol of communication between agents and between agents and Web services.

The input and output parameters of each function introduced below has an XML format.

These XML documents map the data structure defined in the Data Specification.

7.1 Inter-Agent Messages for the Main Agent

7.1.1 Get list of all available online libraries

- Input parameter

Parameter	Description
<pre><command> <userID>String</userID> <commandText>String</commandText> </command></pre>	Send command to get “list of all available OEL”

- Output parameters

Parameter	Description
<pre> <oels> <oel> <oelNumber>string</oelNumber> <oelName>string</oelName> <oelURL>string</oelURL> <libraryNumber>string</libraryNumber> <libraryName>string</libraryName> <libraryAddress>string</libraryAddress> </oel> </oels> </pre>	List of all available oels

7.1.2 Get list of all available computers for selected online electronic library (oel)

- Input parameter

Parameter	Description
<pre> <command> <userID>string</userID> <commandText>string</commandText> <libraryNumber>string</libraryNumber> <specifiedDate>string</specifiedDate > </command> </pre>	Send command to get “list of all available computers in the selected library for specified date”

- Output parameters

Parameter	Description
<pre> <computers> <computer> <userID>string</userID> <computerNumber>string</computerNumber> <libraryNumber>string</libraryNumber> <specifiedDate>string</specifiedDate> <Time>string</Time> </computer> </computers> </pre>	List of all available computers in the selected library in specified date

7.1.3 Make appointment

Input parameters: make appointment with specified data:

- Oel (online electronic library)
- Library (which access with the given oel)
- Specified date
- Specified time
- Specified duration of appointment

Parameter	Description
-----------	-------------

<pre> <command> <userID>string</userID> <commandText>string</commandText> <oelNumber>string</oelNumber> <libraryNumber>string</libraryNumber> <computerNumber>string</computerNumber> <specifiedDate>string</specifiedDate > <specifiedTime>string</specifiedTime> <specifiedDuration>string<specifiedDuration> </command> </pre>	Send command to make appointment
---	----------------------------------

Output parameters:

Parameter	Description
<pre> <command> <userID>string</userID> <commandText>string</commandText> <appointmentNumber>string</appointmentNumber> <libraryNumber>string</libraryNumber> <computerNumber>string</computerNumber> <specifiedDate>string</specifiedDate > <specifiedTime>string</specifiedTime> <specifiedDuration>string<specifiedDuration> </command> </pre>	Confirmation of appointment

7.2 Inter-Agent Messages for the Scheduler Agent

7.2.1 Send email with appointment information to the user

- Input parameters: appointment information

Parameter	Description
<pre> <appointment> <appointmentNumber>string</appointmentNumber> <userName>string</userName> <userSurname>string</userSurname> <userEmail>string</userEmail> <oelName>string</oelName> <libraryName>string</libraryName> <libraryAddress>string</libraryAddress> <computerNumber>string</computerNumber> <computerLocation>string</computerLocation> <Date>string</Date > <Time>string</Time> <Duration>string</Duration> </pre>	Notification with appointment information for user email

</appointment>	
----------------	--

Output parameters:

Parameter	Description
<confirmation> <appointmentNumber>string</appointmentNumber> </confirmation>	Confirmation response

Conclusions

Agent – based Software Engineering is new and rapidly development technology. It provides a lot of new ideas for development very sophisticated Web applications for different industries. So it is very useful.

Abbreviation

- OEL – online electronic library
- SOAP - Simple Object Access Protocol
- UDDI - Disco and Universal Description, Discovery, and Integration
- WSDL - Web Services Description Language
- XML - Extensible Markup Language
- HTTP - Hypertext Transfer Protocol
- URL - uniform resource locator

References

1. Dr. Behrouz H. Far, Agent-based software Engineering, handouts, University of Calgary, 2005
2. SENG 609.22, Agent-based Software Engineering, Sample Project: Travel Agency System (TAS), University of Calgary, Fall 2004
3. Developing XML Web Services and Server Components with Visual C# .NET and .NET Framework, Amit Kalami, Priti Kalami, 2003