

UNIVERSITY OF
CALGARY

A Project Report for SENG 609.22
Agent Based Software Engineering
Course Instructor: Dr. Behrouz H. Far

Project Report

“Job Finding Agency System”

- JFAS -

Dec. 2002-ENEL-University of Calgary-

Table of Contents

1. INTRODUCTION	9
2. SYSTEM SPECIFICATIONS	9
2.1. Business Case	9
2.2. System Description.....	9
2.3. Assumptions	11
2.4. Requirements:.....	11
2.5. Wish List	13
3. SYSTEM DESIGN DOCUMENT	13
3.1. System Architecture	13
3.2. Role Identification	15
3.3. Agents Description	15
3.4. Agents Internal Architecture.....	17
3.5. Technology Overview.....	19
3.6. Communication Protocol.....	20
4. DETAILED DESIGN DOCUMENT	22
4.1 Actors Usecase Diagrams.....	22
4.2 Collaboration Diagram	27
4.3 Sequence Diagram.....	28
4.4 Activity Diagram.....	29
4.5 System Complete Use Case Diagram.....	30
4.6 System Class Diagram.....	31
4.7 System Use cases specification.....	32
5. DATA SPECIFICATION	35
5.1. ER Diagram.....	35
5.2. Typical Data Definition.....	36
6. INTERNAL AGENT MESSAGES	37
7. CONCLUSION	40
8. REFERENCES	41
9. ACKNOWLEDGEMENT	41

□ **Abstract**

With the increasing importance of complex software systems in the software industry, the need for using agent technologies to develop large scale commercial and industrial software systems is growing rapidly. As web applications become increasingly integrated in business strategies for individual, small and large companies, the need to build reliable and adaptive systems grows in importance. In this project we propose a Job Finding Agency System framework JFAS, which will assist the job seeker in the process of searching for a matching opportunities to his or her skills, and sending the application consisting of the pre-built resume and a created custom cover letter.

The over all system incorporates the three enabling technologies: component-based development, security, and internet standard. The system characteristics include the software agent based technology consisting mainly of three components: cooperation, autonomy, and learning

□ **Preface: Project Proposal**

Problem statement and motivation

The process of searching for a job is an unavoidable, frustrating and time-consuming process. Today people are selected for interviews based on their application, this one composed of a cover letter and a resume. Internet is becoming one of the most used tools to find out the available jobs in the market. Millions of people are accessing thousands of databases in order to find out those that match their qualifications. The available search engines provide the user with a window interface with: keyword search, category search, location search, period and so on. When using the available search engines usually for each keyword or category search the applicant will be faced with the process of matching what is available with his/her qualifications, so he /she has to prepare a cover letter template with his/her details, the available job poster details, and a customized content depending on the order of to the job posted in order to get the employer attraction. So a manual matching and style is needed between the qualifications and the requirements in order to prepare a job application consisting of a cover letter and a resume. This process is basic; however it is a time consuming it can extend from days to months of matching process.

The proposed approach

We want to automate the task of hunting for a job on the Internet. Our approach is to build a software agent that will get inputs from the applicant who enters key criteria (period/date, location, and category) and a list of his/her search keywords, the list will be of a length 1 to n. The agent will be using an iterative process by communicating individually the quadruple inputs (period/ date, location, category, keyword) to the available search engines and it will get the available posted jobs for that research criteria and automatically generates a job application consisting of a cover letter and attached to it the applicant resume. Each time the agency will open an empty cover letter template put in the header: the applicant information, the date, and pull up from the posted job the information of the employer (company name, correspondent, address, telephone, etc), with a standard text content in it. The automatic generated cover letter will be joined to the pre- built applicant resume, and send both as an attachment files to the job poster¹.

Justification of the use of software agent:

- We want this piece of software to function continuously and autonomously in any environment. This requirement matches exactly with the definition of an agent given by [Shoham 97]
- We like the software to have the following characteristics:

¹ See figure 1, and figure 2

- Knowledge ability: has the capability to cooperate between the applicant (user) and the search engine(processor): information to be received from both sources
- Learning: be able to have the ability to evolve by including new categories, keywords and main qualifications introduced by the different users.
- Autonomy: The ability to initiate the communication with different search engines, and to be set for any category or type of user with no distinction.
- Communication: ability to communicate between the user (applicant), search engine, the employer (when sending the application) and between separate agents.
- Mobility: be able to be executable in any web environment.

Fig.1. The proposed system

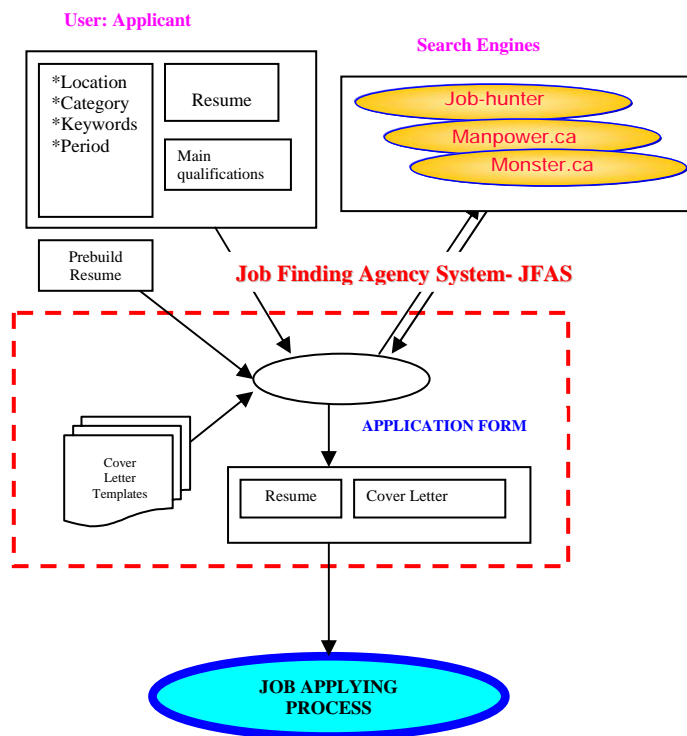
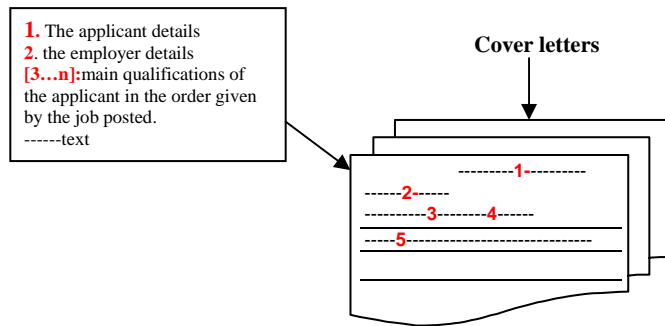


Fig.2. The Cover Letter Generation Process

Methodology:

Our approach is to provide a system that most relevantly has the usability, reliability, efficiency and maintainability characteristics [5]. We carry out three main activities: Formulation, analysis and Design. In the *formulation* activity we identified the goals and objectives of the system and we established the scope. The *Analysis* activity consists of establishing the technical requirements of the system and we identified the content items that will incorporate requirements for the design. The *design* activity consists on using the Unified Modeling Language (UML) as a graphical language for visualization, specifying, constructing, and documenting the artifacts of our system. UML is a very expressive language, addressing all the views needed to develop and then deploy such systems.

We were willing to carry on two other steps for complete system software these are *Implementation* and *testing* but due to time constrains we could not reach this stage.

The activities we carry in this project are summarized in the following steps:

- System Specifications
- Design Document
- Detailed Design Document
- Data Dictionary
- Inter-Agents Messages.

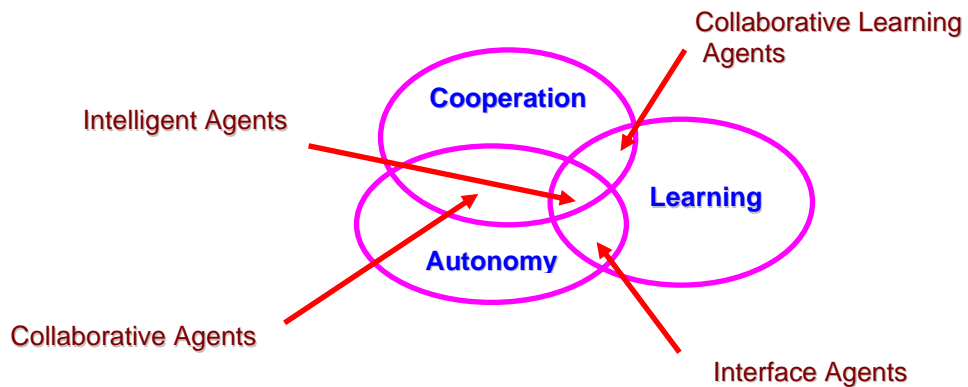
1. Introduction

The process of searching for a job is an unavoidable, frustrating and time-consuming process. Today people are selected for interviews based on their application, this one composed of a cover letter and a resume. Internet is becoming one of the most used tools to find out the available jobs in the market. Millions of people are accessing thousands of databases in order to find out those that match their qualifications. The available search engines provide the user with a window interface with: keyword search, category search, location search, period and so on. When using the available search engines usually for each keyword or category search the applicant will be faced with the process of matching what is available with his/her qualifications, so he /she has to prepare a cover letter template with his/her details, the available job poster details, and a customized content depending on the order of to the job posted in order to get the employer attraction. So a manual matching and style is needed between the qualifications and the requirements in order to prepare a job application consisting of a cover letter and a resume. This process is basic; however it is a time consuming it can extend from days to months of matching process.

We want to automate the task of hunting for a job on the Internet. Our approach is based on the use of an Intelligent Agent System. The notion of an Intelligent agent (IA) is one of the most important concepts to emerge in IT system in the 1990s. IAs are viewed as “Software components and/or hardware which are capable of acting exactly to accomplish tasks on behalf of its user and learn as they react and /or interact with the external environment [1]”. The partial view of an agent typology is represented in Fig.3. There are three main components: cooperation, autonomy, and learning [2]. Agent technology looks set to radically alter not only the way in which computers are interacted but the way complex systems are conceptualized and built as well. The choice of agents as solution technology was motivated by the following observations:

- The domain involves an inherent distribution of data,
- The integrity of the existing web services and the autonomy of its update of changes must be maintained.
- Interactions are fairly sophisticated, including information sharing, and coordination.
- The job search is an iterative process and needs a reasoning capability for matching the available positions and creating a custom application.

When taken together, this set of requirements leaves agents as the strongest solution candidate.

Fig.3. Partial View of an Agent Typology

Our approach for this project is to build a software agent that will get inputs from the applicant who enters key criteria (period/date, location, and category) and a list of his/her search keywords, the list will be of a length 1 to n. The agent will be using an iterative process by communicating individually the quadruple inputs (period/ date, location, category, keyword) to the available search engines and it will get the available posted jobs for that research criteria and automatically generates a job application consisting of a cover letter and attached to it the applicant resume. Each time the agency will open an empty cover letter template put in the header: the applicant information, the date, and pull up from the posted job the information of the employer (company name, correspondent, address, telephone, etc), with a standard text content in it. The automatic generated cover letter will be joined to the pre- built applicant resume, and send both as an attachment files to the job poster.

The remaining of this report describes the work undertaken in this project to develop the agency that will assist the job seeker. The system specifications are presented in section two, section three describes the system design. Section four, five, and six give the detailed design and section seven concludes the project report.

2. System Specifications

2.1 Business Case

Job-hunting is a time consuming process, with many steps involved.

1. Connect to a site that post available job.
2. Search for jobs that match qualifications and interests.
3. Check to make sure Resume is okay for particular job.
4. Write a cover letter for the job advertisement.
5. Write an email to apply for the job.
6. Send the email with the resume and cover letter attached to the contact that is supervised hiring.
7. Report to the user on activities taken on his/her behalf.

This process must be repeated for every job. Each step requires the personal attention of the job seeker.

2.2 System Description

An IA framework called Job Finding Agency System (JFAS) is proposed to enhance the traditional process of finding a job. The system characteristics include the following [3]:

- *Intelligence*: The agent automatically generates the cover letter, and sends the user application. It also automatically adapts to changes in its environment.
- *Autonomy*: An agent is able to take the initiative and send the application and customs the cover letter.
- *Cooperation*: An agent does not blindly obey commands but makes suggestions to modify the cover letter template format.

The proposed Job Finding Agency System (JFAS) is a multi-agent system designed to maintain the seeker update with the available job opportunities in his or her area, and prepare the application consisting of resume and cover letter to the employer.

The JFAS deals with multiple search engines posted in the web to get from them the data bases of the available opportunities.

The system has to prepare the application form consisting of resume and cover letter, and send it to the employer and let him/her know about the replay. Fig.4 shows the system's described.

Fig.4. System Description

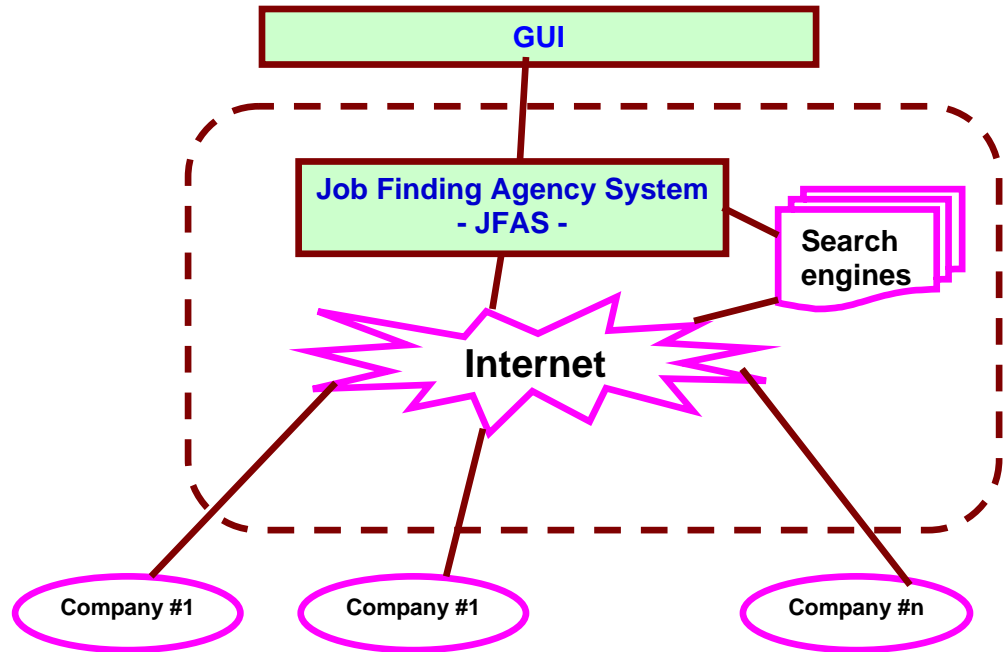
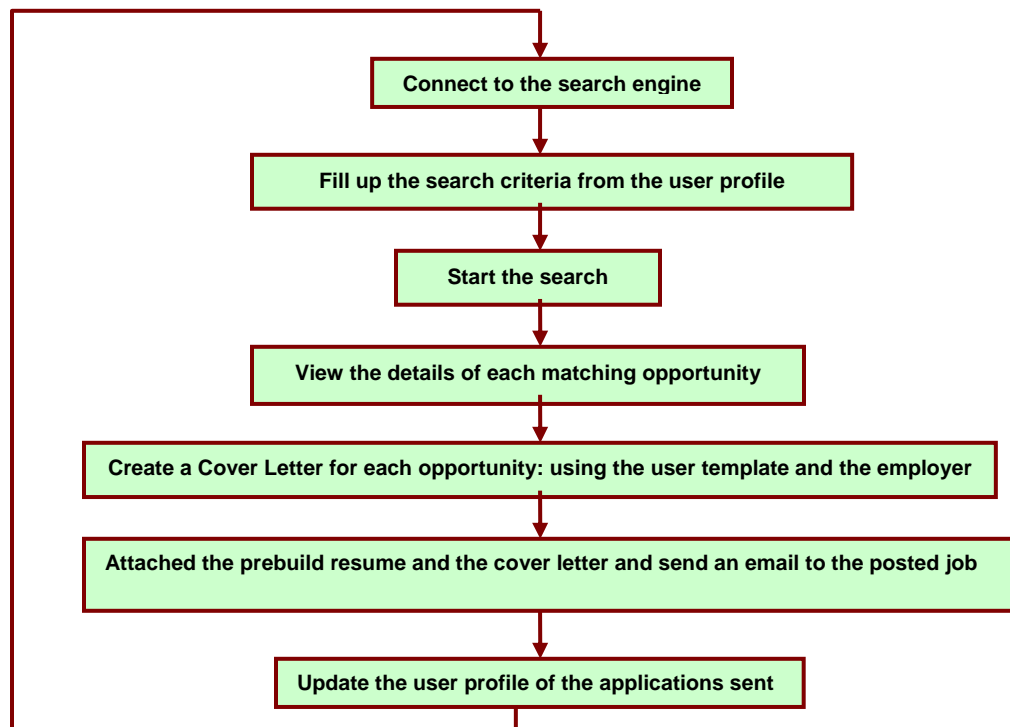


Fig. 5. Job Finding Process: An Iterative Approach

2.3 Assumptions

- The system maintains a user profile with his or her own individual preferences such as:
 - ◇ Location
 - ◇ Category
 - ◇ Job type (part time, full time, contract, permanent)
 - ◇ Keyword search
 - ◇ Period when the system should search for the job
 - ◇ Period of the process of sending the application
 - ◇ Salary. etc..
- The resume is pre-build by the user.
- Cover letters templates in the system data base.
- An executable program that open an empty cover letter, with a standard paragraphs and incorporates custom user information to complete it.
- Search engines known by the system and updated in the system data base.

2.4 Requirements

Some of the problems that may arise during the requirements engineering process are a result of failing to make a clear separation between these different levels of

description. We make this separation by using the term *user requirements* to mean the high level abstract requirements, these are statements in a natural language plus diagrams of what services the system is expected to provide and the constraints under which it must operate. And system *requirements* to mean the details description of what the system should do, these set out the system services and constraints in details. As well as these two levels of detail, a more detailed description: a *software design specification* is produced to bridge the requirements engineering and design activities. It is an abstract description of the system software design which is a basis for more detailed design and implementation. This specification adds further details to the system requirements specification [4]

Tab.1: User Requirements Definition

- The JFAS shall be used for all different categories and job type and be able to future changes in any job requirement or specification
- The JFAS shall be able to have different Cover letter templates
- The JFAS shall prepared the application from the pre-build resume, create a custom cover letter for each posted job, and send an email with the application attached.
- The JFAS shall deal with the search process for all the available search engines. The JFAS shall provide the search engine with the users search criteria and preferences.
The JFAS shall collect the detailed or brief matching opportunity from the search engine to prepare the custom cover letter and email to send.
- The JFAS shall alert the user of the allocation sent and the acknowledgment.
- The JFAS shall allow the permitted user to create and update his/her profile. The JFAS shall give the user permission to change his/her access password to the system.

Tab2 : System Requirements Specification

- The steps of job finding process that the JFAS should follow are described in Fig. 5
- The system should be multiagent based and provides an infrastructure specifying the communication In terms of interaction protocol, communication language and transport protocol..
- The JFAS shall have the following characteristics: Usability, Functionality, reliability, Efficiency, and Maintainability.
- The JFAS shall incorporate three important enabling technologies: components development, security and Internet standard.
- The JFAS shall communicate with more than one provider.

These requirements may not be complete and consistence. The reason for this is partly because of the inherent system software complexity and partly because different view points have inconsistent needs.

2.5 Wish List

- 3.1 The JFAS sets multiple profile for a single user
- 3.2 The JFAS updates the resume.
- 3.3 The JFAS ranks user skills to fits the available opportunities.
- 3.4 The JFAS carries out the application send process only to the user's preferences destination.

3. System Design Document

3.1 System Architecture

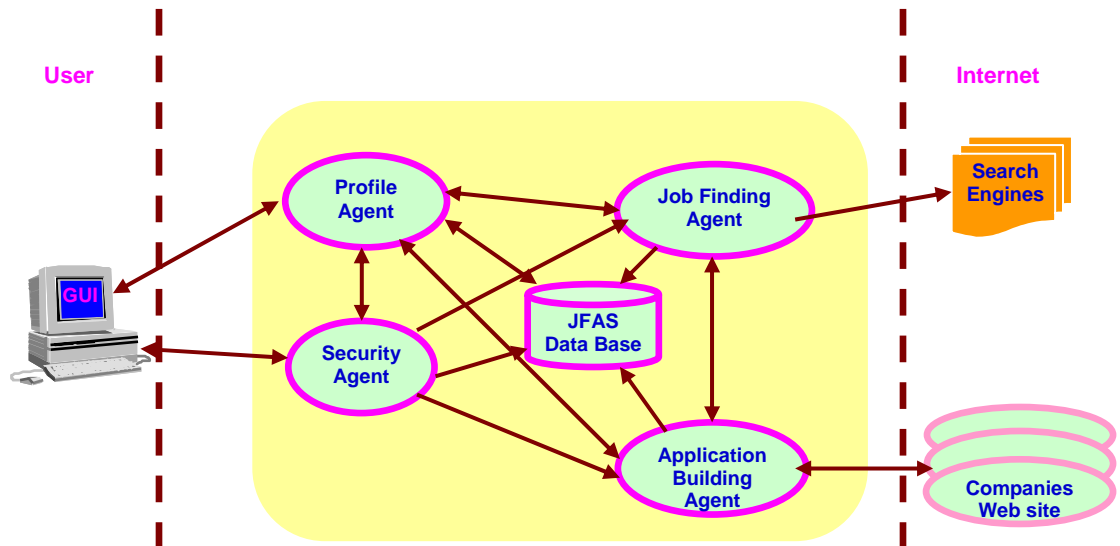
- The way this system supposes to work is that there will be a layer of multi agent system between the client (browser) and already companies existing Web sites and already existing search engines known by the system.
- The search engines can be from different providers, and they shall return their services to the system based on the user input and profile.
- The agents then sort the results and create and send an application for each matching opportunity.

The view of the detailed or brief job of interest will be provided from the web search engines based on the user inputs and profile.

The system then sorts the results in the way the user wants and prepares the application and sends it the employer autonomously.

We proposed a horizontal layering architecture (Fig 7)

Fig.6. System Architecture



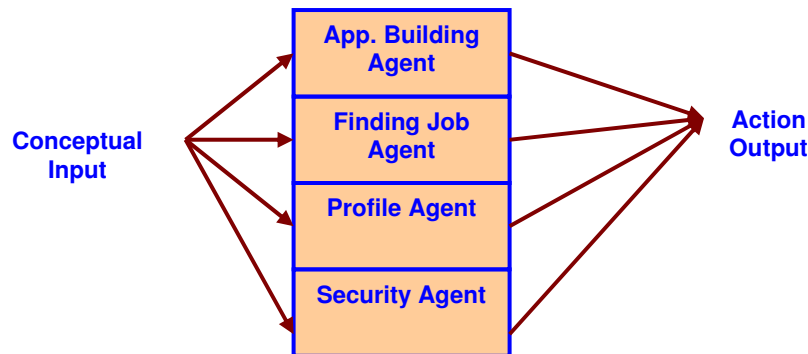
In this system we have the user login in to the browser interface. The first screen will ask for the user name and password, than a main screen to allow the user to inter:

- Keys search:
 - Keywords Search
 - Category Search
 - Location Search
 - Period Search
 - Job Type
- Pre-built Resume
- Selection of the Cover Letter Template
- Preferences:

Company's name, job locations, cover letter templates, search engines information, and are all stored in the JFAS data base.
- Any user who want to use the JFAS has to create his' her own profile before hand, the profile consists of:
 - A copy of an updated resume
 - Selection of the cover letters preferences.
 - Setting his/her preference.
 - Setting the search criteria.

The multi agent system has to interact with the search engines to get the user available job opportunities and with the pre-build resume and the template of the cover letter selected in the user profile the system send the application to the posted job available.

Fig 7: JFAS- Horizontal layered architecture.



The agents are directly connected to the inputs and action output. Where each layer is an agent producing suggestions as what action to perform [7]. The great advantage of horizontal layered architecture is its conceptual simplicity. However because the layers are competing with one another to generate action suggestions. There is a danger that the overall behavior of the system will not be coherent. In order to ensure that horizontal layered architectures are consistent. We are including a mediator function which makes decision about which layer has “control” of the system at any given time. In our JFAS this function is given to the profile Agent.

3.2 Role Identification

The roles that that are required from the system are:

Personal assistance:

- The system provides the user with assistance for a job in the web.
- The system keeps an update profile of the user.
- The system sends the application to the job poster.
- The system does the job search using the web search engines.

Job Application Creation:

- The system creates a custom cover letter for each matching available position.
- The system joins the pre-build resume and the created cover letter to form an application

Web Services handling:

- The system handles he communication and interaction with the web search engines
- The system keeps an update of the different search engines of different providers.

3.1 Agents Description

Security Agent

- The Security Agent gets permission from the provider to give or deny the user access to the system.
- The security Agent lets the user log in and log off the system.
- The Security Agent updates the provider with the new password the user may use.
- The Security Agent gives the user permission to change the password.
- The security agent verifies user's name and password.

Profile agent

- The Profile Agent keeps the user profile consisting of:
 - User's personal details (name, address, telephone number, email).
 - User's main skills.
 - User's search criteria
 - User's pre build Resume
 - User's preferred cover letter template.
 - User's preferences for work: location, period and companies.
 - The history of the applications sent with the time and the destination.
 - Personal user name and password to access to the system
- The profile Agent allows the user to update his/her profile.
- The profile Agent provides the Security Agent of the new access password the user may change after a certain period of time for security purpose. This one to be communicated to the provider.
- The Profile agent provides the user with a set of cover letters templates for choice
- The Profile Agent gets the permission from the security agent to set a profile for the user.
- In our architecture the Profile Agent handles the correspondence with all the other agents and in this way we have a central point of delegation.

Job Finding Agent

- The Job Finding Agent obtains the user key search criteria from the Profile Agent.
- The Job Finding Agent obtains a list of matching jobs to the user criteria from the search engines.
- The Job Finding Agent obtains the search period time from the user via the Profile Agent.
- The Job Finding Agent provides the search criteria to the search engine and runs the search.
- The Job Finding Agent communicates the matching job found information to the application Building Agent.
- The Job Finding Agent updates the list of the search engines information and availabilities.
- The Job Finding Agent gets permission from the security engine to access to the search engine data bases

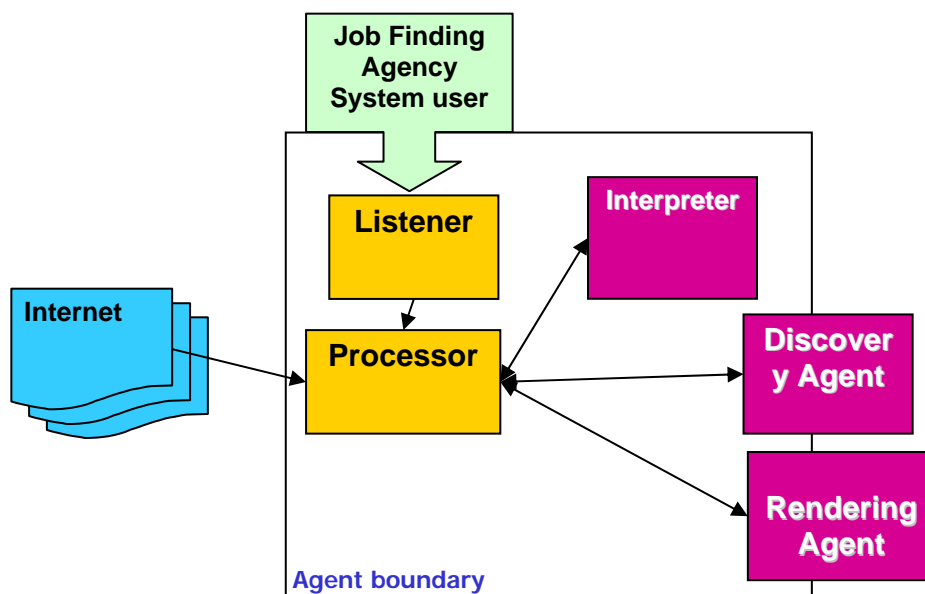
Application Building Agent

- The Application Building Agent receives user's pre build resume via the Profile Agent.
- The Application Building Agent receives the cover letter template to be used from the Profile Agent.
- The Application Building Agent gets permission to send the application from the Profile Agent.
- The Application Building Agent gets employer details from the Job Finding Agent.
- The Application Building Agent generates the cover letter.
- The Application Building Agent generates the email and attached to it the resume and the generated cover letter.
- The Application Building Agent sends the application.
- The Application Building Agent receives the application receive acknowledgment.
- The Application Building Agent updates the profile list of the employers who receive the user application

3.2 Agents Internal Architecture

Agent Internal architecture supports developing agents from reusable components at the sub system level. Subsystems can be added to an agent, this is a dynamic description, event and ontology registration process.

Fig. 8. Agents Internal Architecture



- **Listener:** The *Listener* component listens to a port for any incoming requests from the JFAS application.
- **Interpreter**
- The *Interpreter* parses and interprets the XML messages. We assume that all agents have agreed on a Document Type Definition (DTD).
- **Processor:**
- The *Processor* receives an XML document as an input. It uses the Interpreter to parse the document, and calls the appropriate function to run a process.
- **Discovery Agent:**
- The *Discovery Agent* provides the service discovery base-service (a superset of UDDI). May be implemented as an external service.
- **Rendering Agent:**
- The Rendering Agent is optional; it can be used by the Processor to render data before sending it back to the calling function. May be implemented as an external service.

3.5 Technology Overview

We did look at the three important enabling technologies for the implementation of the system but due to the time constraints we have not really explore them. These three technologies are: components-based, security and internet standard.

For the *component-based development* technologies have evolved in large part because of the explosive growth of Web-based systems and applications, and here three major infrastructure standards are available: CORBA, COM/DCOM, and JavaBeans. These standards (accompanied by pre-built, tools, and techniques) provide an infrastructure that enables us to deploy third party and custom developed components.

The *security* aspect deals with the fact that the system will reside on the network, it will be open to unauthorized access. A variety of security measures are provided by the network infrastructure, encryption techniques, firewalls, and other measures. The FIPA(Foundation for Intelligent physical Agent) organisation provides a standard Policies and Domains specification [8].

Approaches to Service Discovery

- Two categories of information must be discovered before a service is used:
 - Logical information
 - Technical information
- The logical information provides a description of the nature of the service, such as business information.
- The technical information describes the interface parameters of the logic that drives the service. Technical information, such as the exact protocol that the service uses, the inputs and outputs, and the encoding of messages, form a crucial part in machine-to-machine conversation.

- A service must be visible to be discovered. In much the same way businesses list themselves in directories, such as the yellow pages, and mount signs on their entrances describing the nature of their business, Web services must provide short descriptions (on the Web) of themselves.
- Roughly, there are three major ways by which Web services can be described and, consequently, discovered:
 - Universal Description, Discovery, and Integration (UDDI) and Web Service Description Language (WSDL), the yellow pages approach
 - World Web Wide Consortium's (W3C) Resource Description Framework (RDF), the sign mounting approach
 - JINI discovery, the Java code discovery approach.

1. UDDI and WSDL

- Universal Description, Discovery, and Integration (UDDI) with Web Service Description Language (WSDL) are analogous to the yellow pages approach. UDDI is a set of specifications for describing services and registries that maintain these descriptions. WSDL is where the technical information of a service is described. This approach depends on worldwide standardization between businesses and the limitations of directories. Not all businesses are listed in the yellow pages and depending on such semi-centralized approach provide the potential for monopolizing the directories resulting in charges for businesses and clients. However, the yellow pages approach has proved to be a success and, we believe, will continue to be so in the future.
- For the scope of this project, we consider that Web Services are pre-defined and they all use a standard WSDL description document. We assume that each JFAS agent will use a Discovery Agent to locate search engines from different providers

2. RDF

- The World Web Wide Consortium's (W3C) Resource Description Framework (RDF) is analogous to the sign mounting approach.
- RDF provides the specifications and mechanisms to provide meta-data (data describing data), and thus the ability to describe both logical and technical information of services.
- Although RDF can be used as the framework for describing services within UDDI registries, RDF does not require such registries. RDF meta-data describing a given service will be stored within the service itself.
- The next-generation search engines will be used to walk through these descriptions and discover ample information describing different aspects of the service. Such an approach will be another huge success.

3. JINI

- In the JINI approach to Web service description and discovery, each JINI service has a proxy object. These are objects that the client uses in order to interact with

the service. These proxies are maintained by a special service, called the lookup service. A JINI service advertises itself to other services by publishing its proxy with the lookup service. Since lookup is another service, services need to find and download its proxy so that they can use it. JINI solves this problem by building communities of JINI services. One way to assemble communities is by having these services (including lookup) periodically send messages over the network allowing them to be aware of each other. When a JINI service is discovered, its proxy is downloaded to the client, which need only know how to interact with the proxy object.

- JINI proxies are Java objects and, therefore, require a Java Virtual Machine (JVM) running on the client for desired service to be usable by the client. In fact, the JINI clients must be written in Java and, even though a service may be written in any other language, a Java Surrogate is needed for JINI clients to be able to talk to these services.

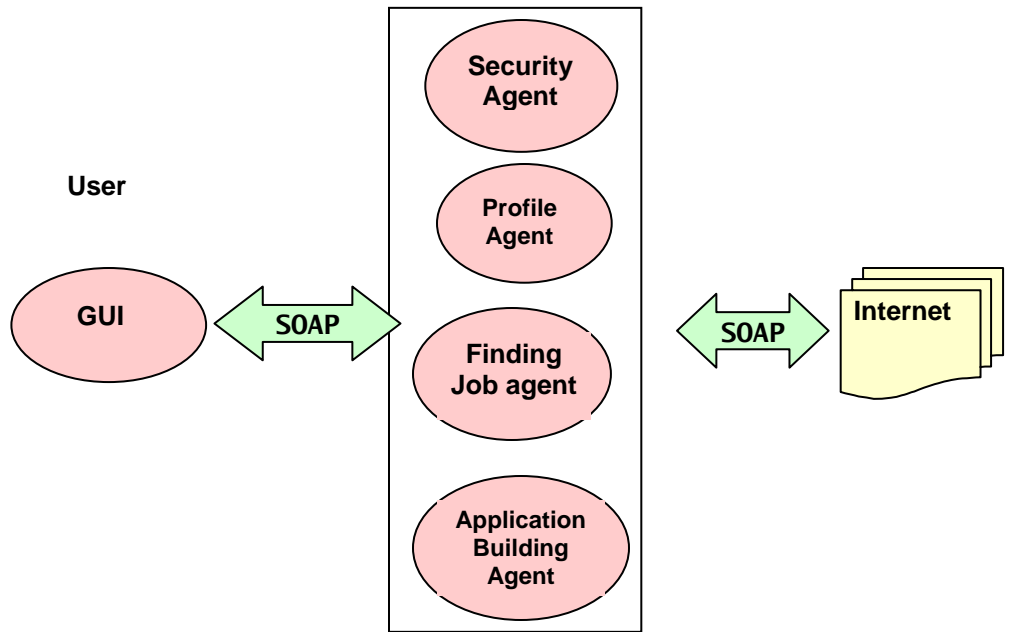
3.6 Communication Protocol: SOAP

Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it
- A set of encoding rules for expressing instances of application-defined data types
- A convention for representing remote procedure calls and responses.

SOAP is proposed to encode an HTTP header and an XML message so that the JFAS agents can call and pass information to each other and communicate with external search engines, and companies we site.

Fig. 9. Communication protocol: SOAP



4. Detailed Design Document

In modeling real systems typically we can view the static part of a system using one of the four *structural diagrams* :Class diagram, Object Diagram, Component diagram, or Deployment diagram. And we can use five *behavioral diagrams* to view the dynamic parts of a system: Use case diagram, Sequence diagram, Collaboration diagram, Statechart diagram, or Activity diagram.[9]

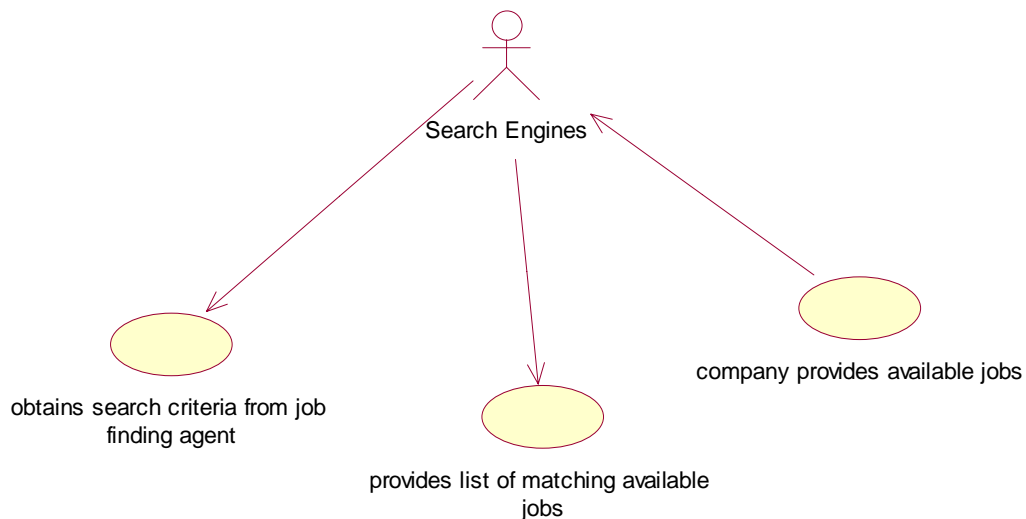
In this project we applied use cases to capture the intended behavior of the system we are developing, without having to specify how that behavior is implemented. We build the use case diagram for each actor.

We identify the following actors:

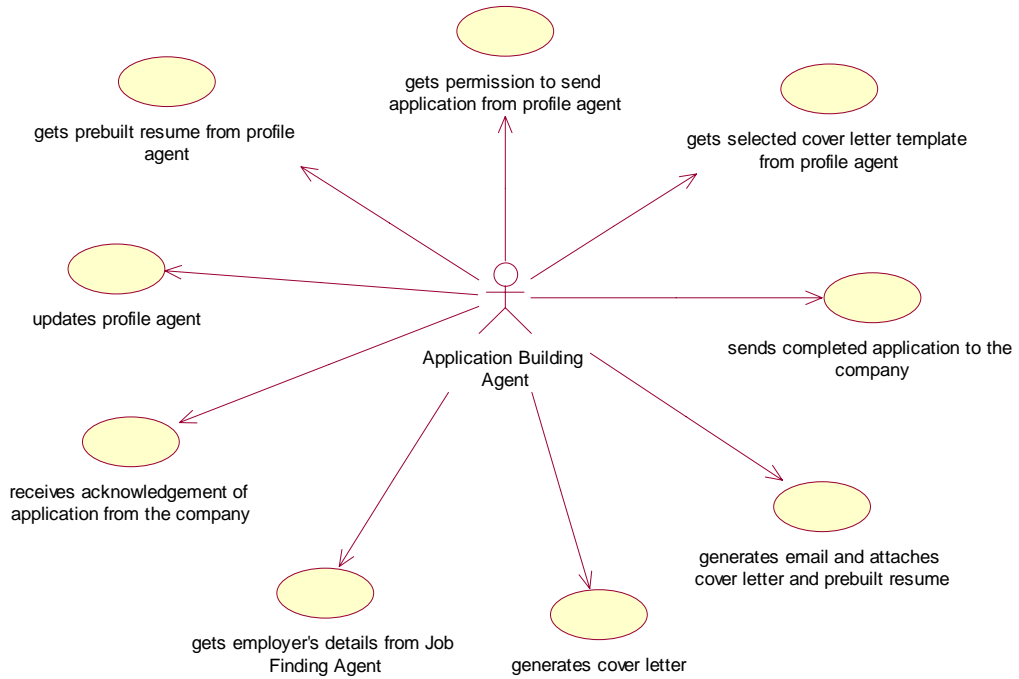
1. User/Job Seeker
2. Search engine
3. Application Building Agent
4. Company
5. Job Finding Agent
6. Profile Agent
7. Provider Agent
8. Security Agent

4.1 Actors use case diagrams.

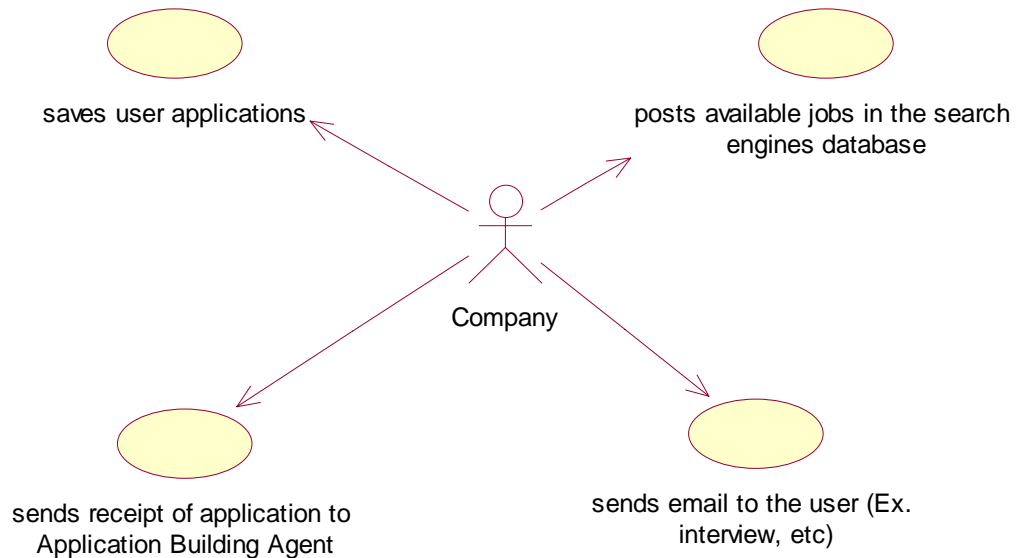
Use case diagram for Search Engines



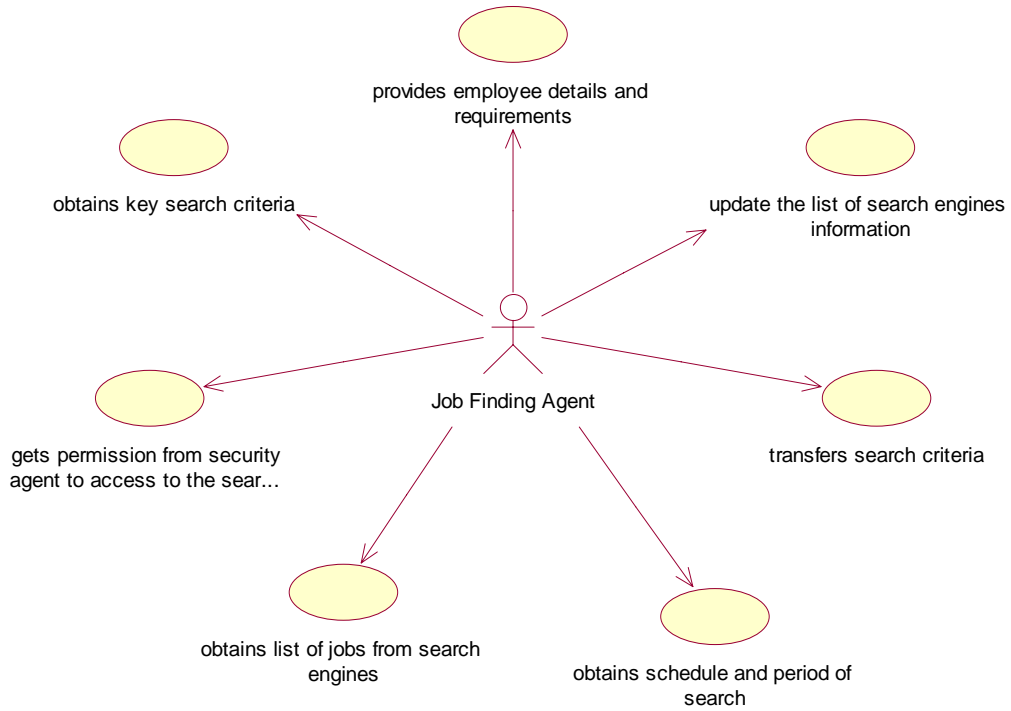
Use case diagram for Application Building Agent



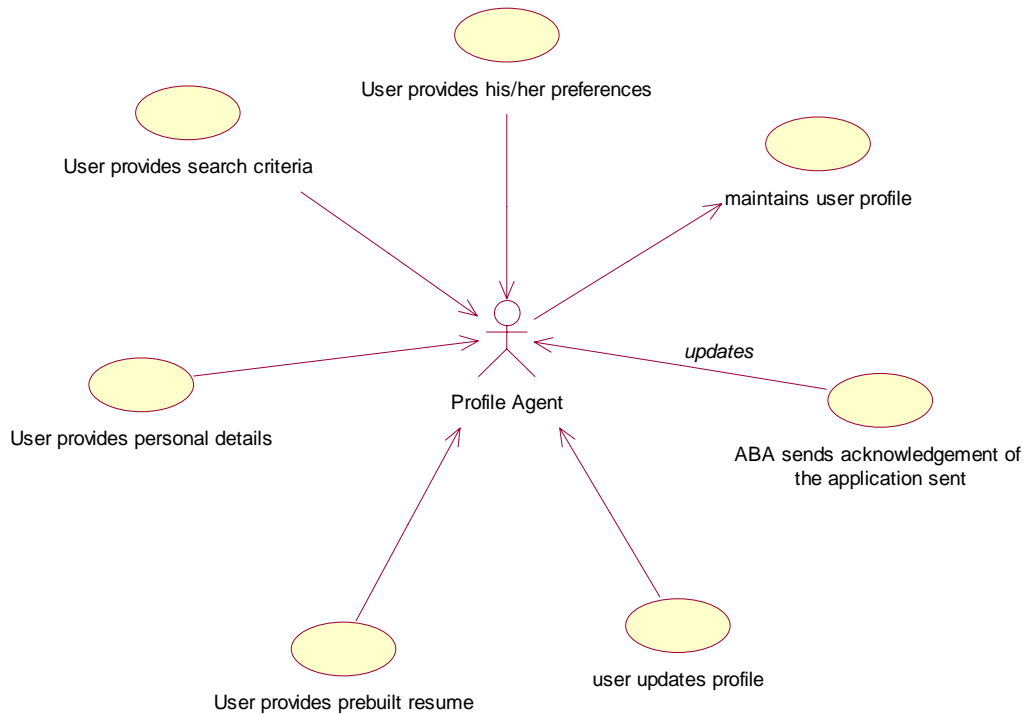
Use case diagram for a Company



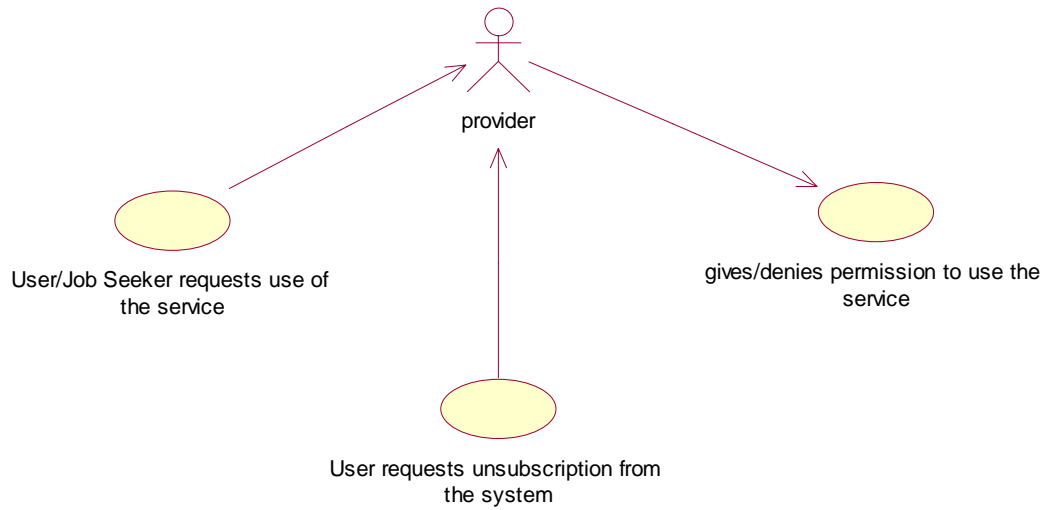
Use case diagram for Job Finding Agent



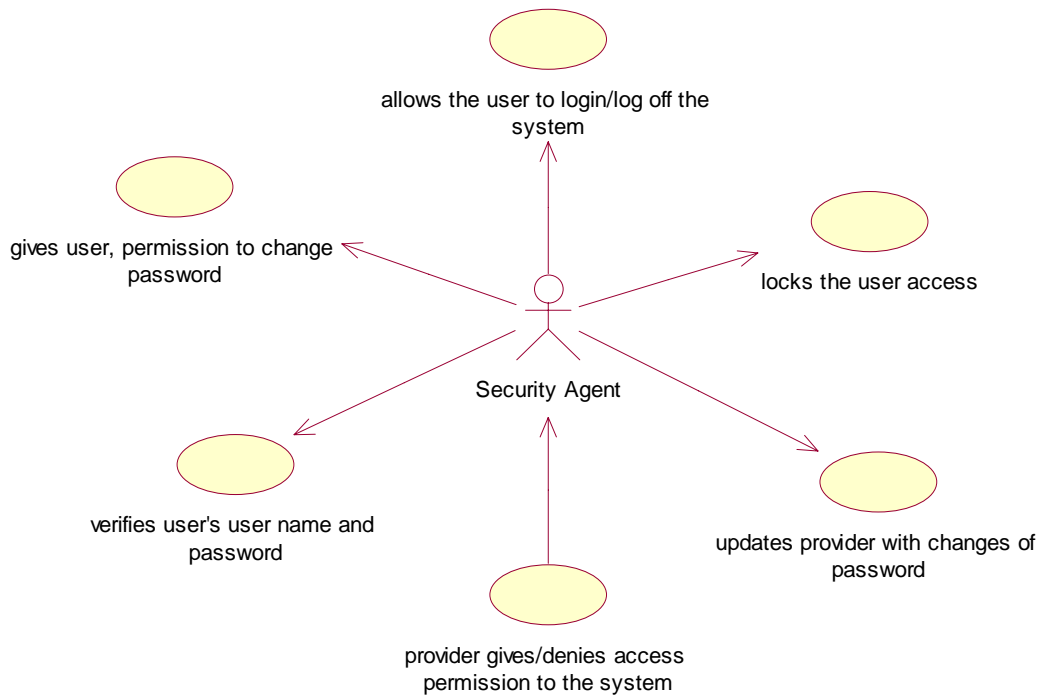
Use case diagram for Profile Agent



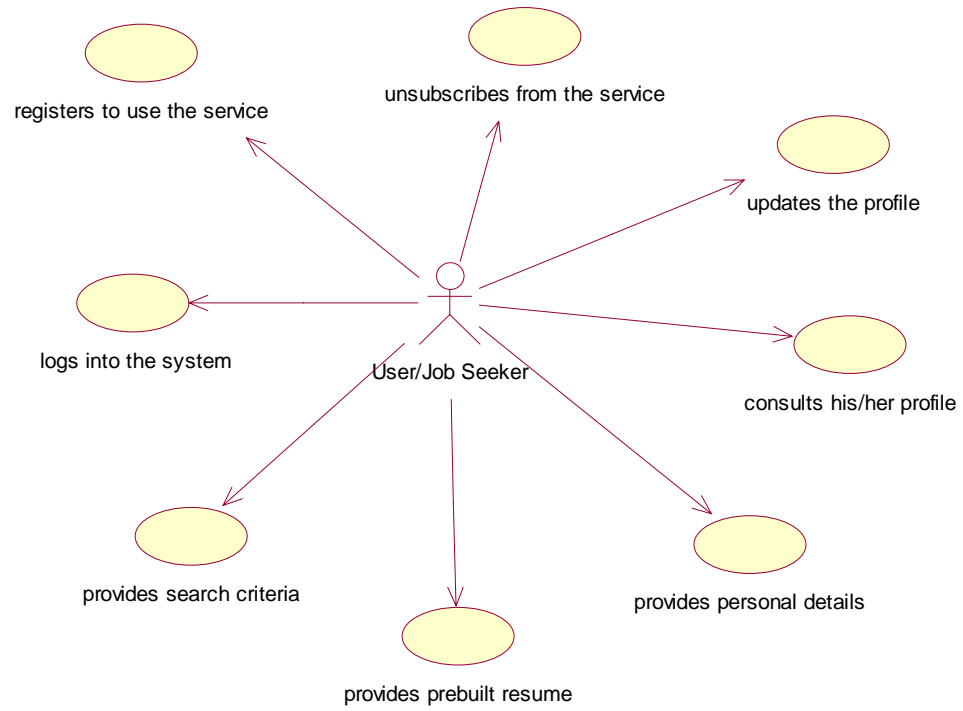
Use case diagram for Provider



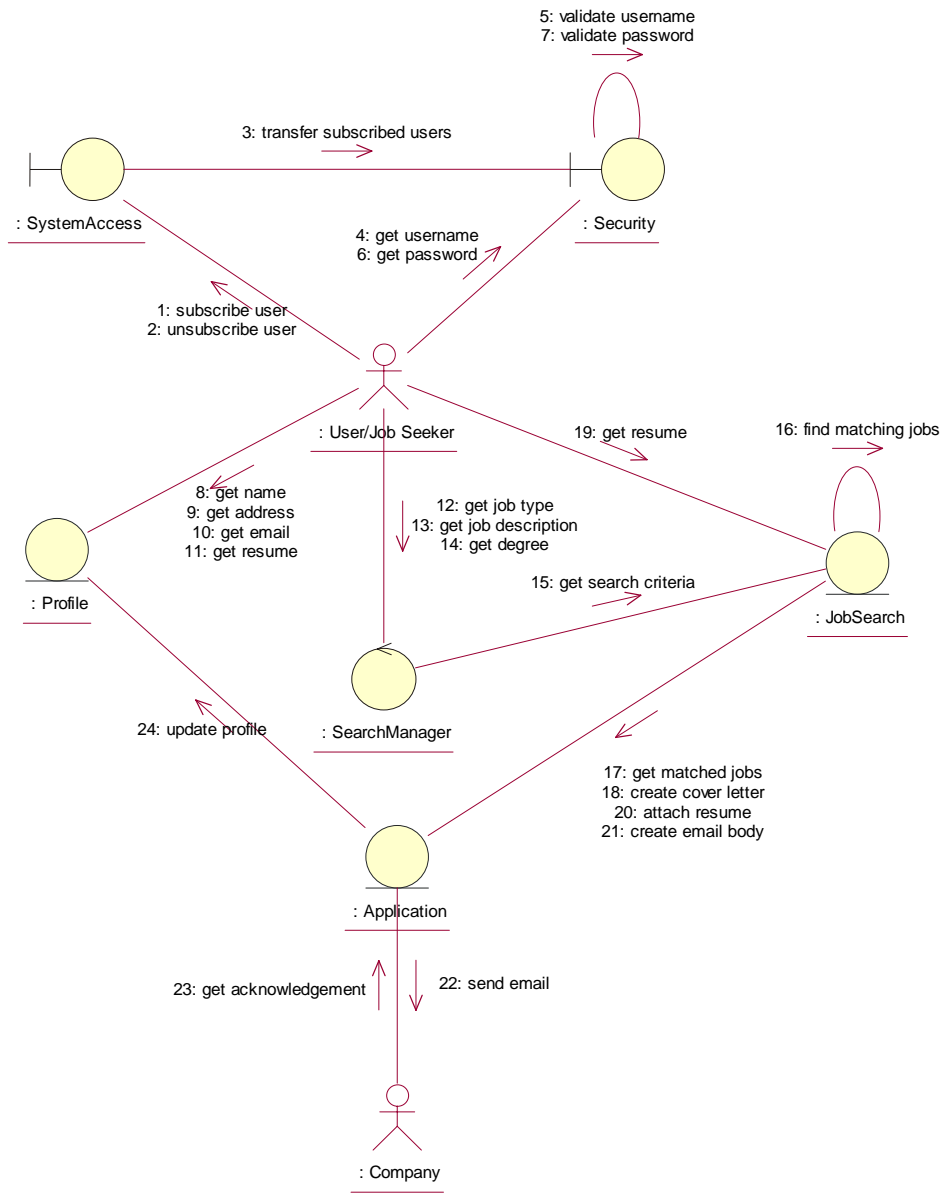
Use case diagram for Security Agent



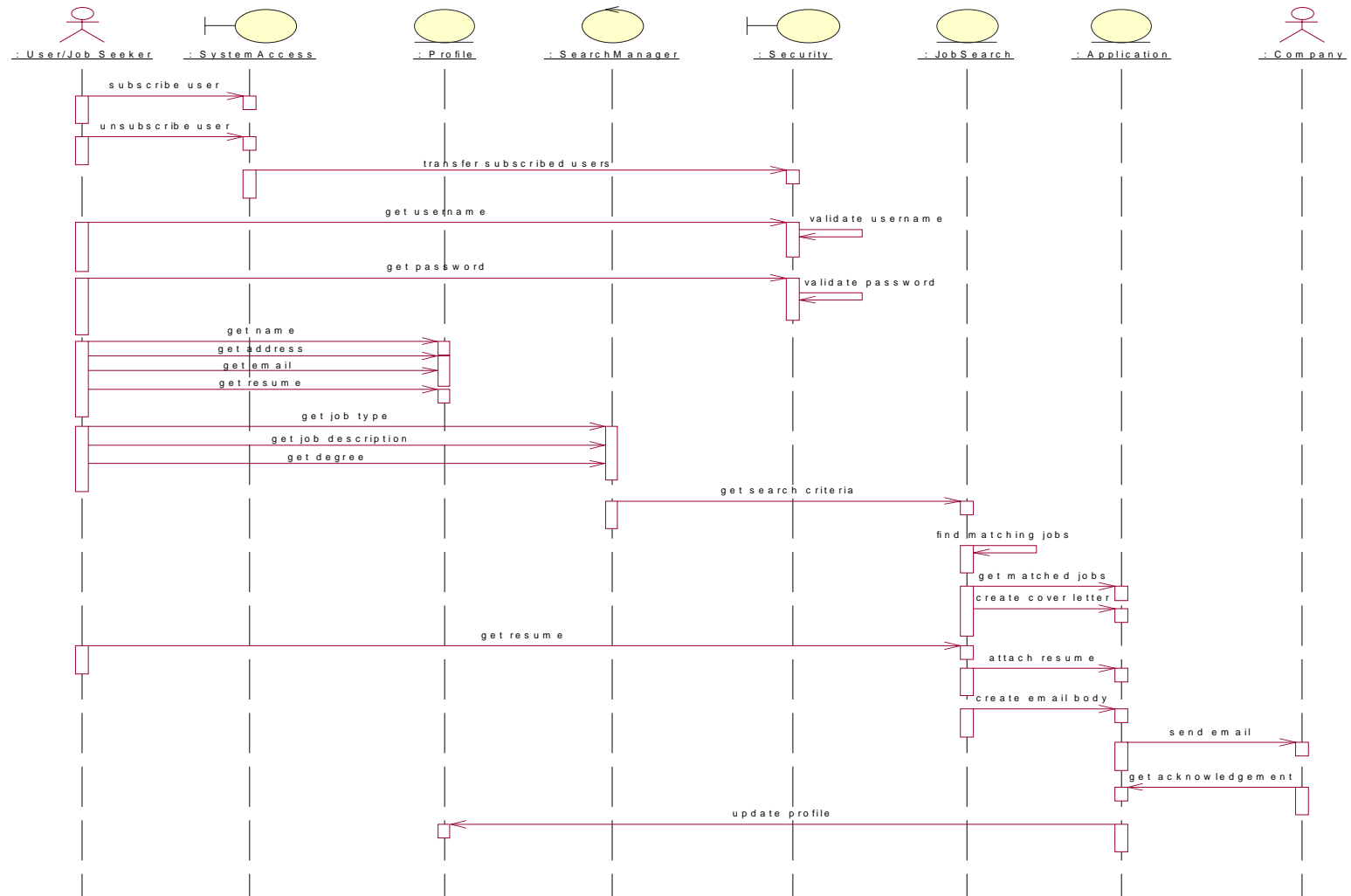
Use case diagram for User/Job Seeker



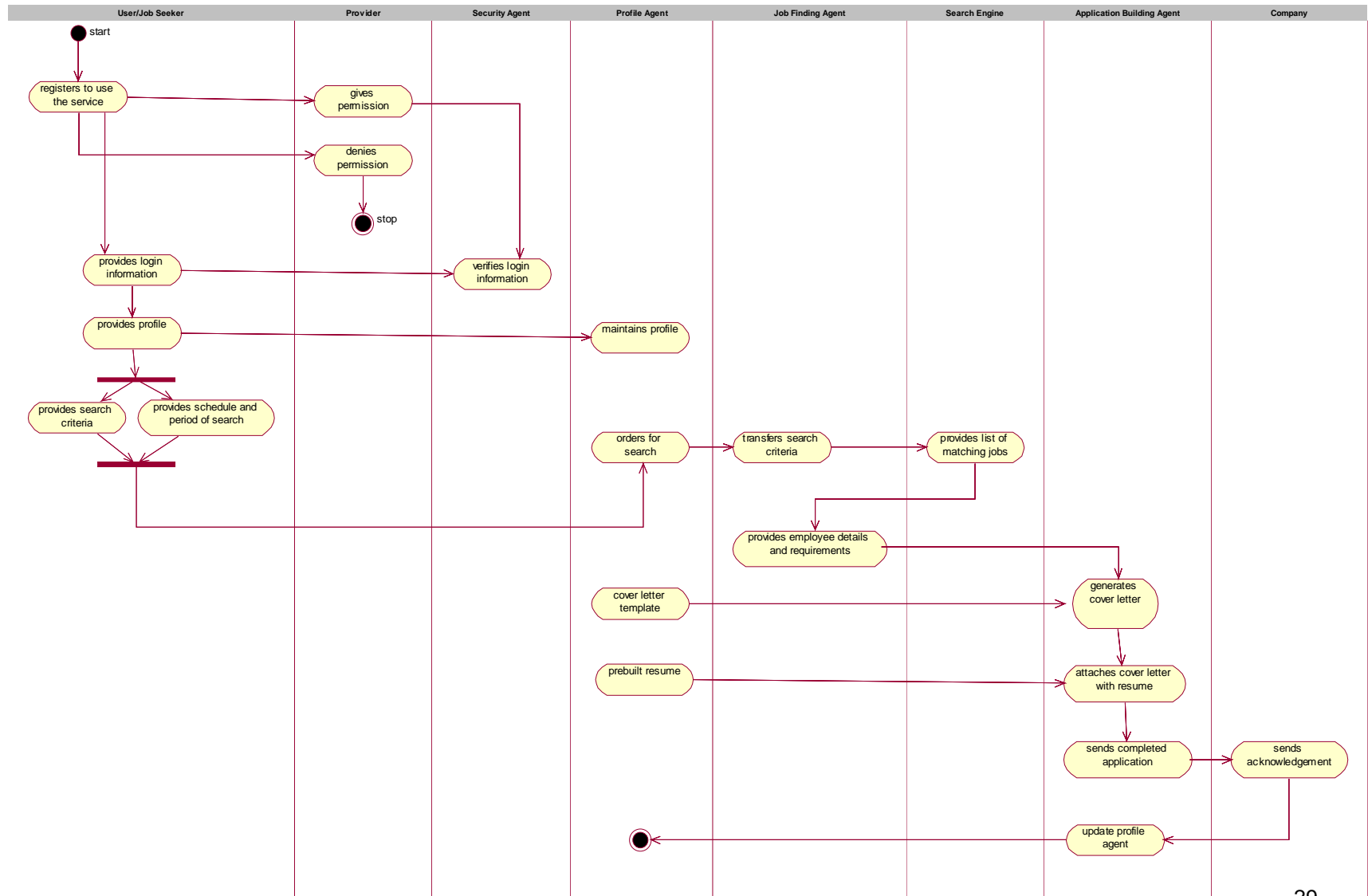
JFAS: Collaboration diagram



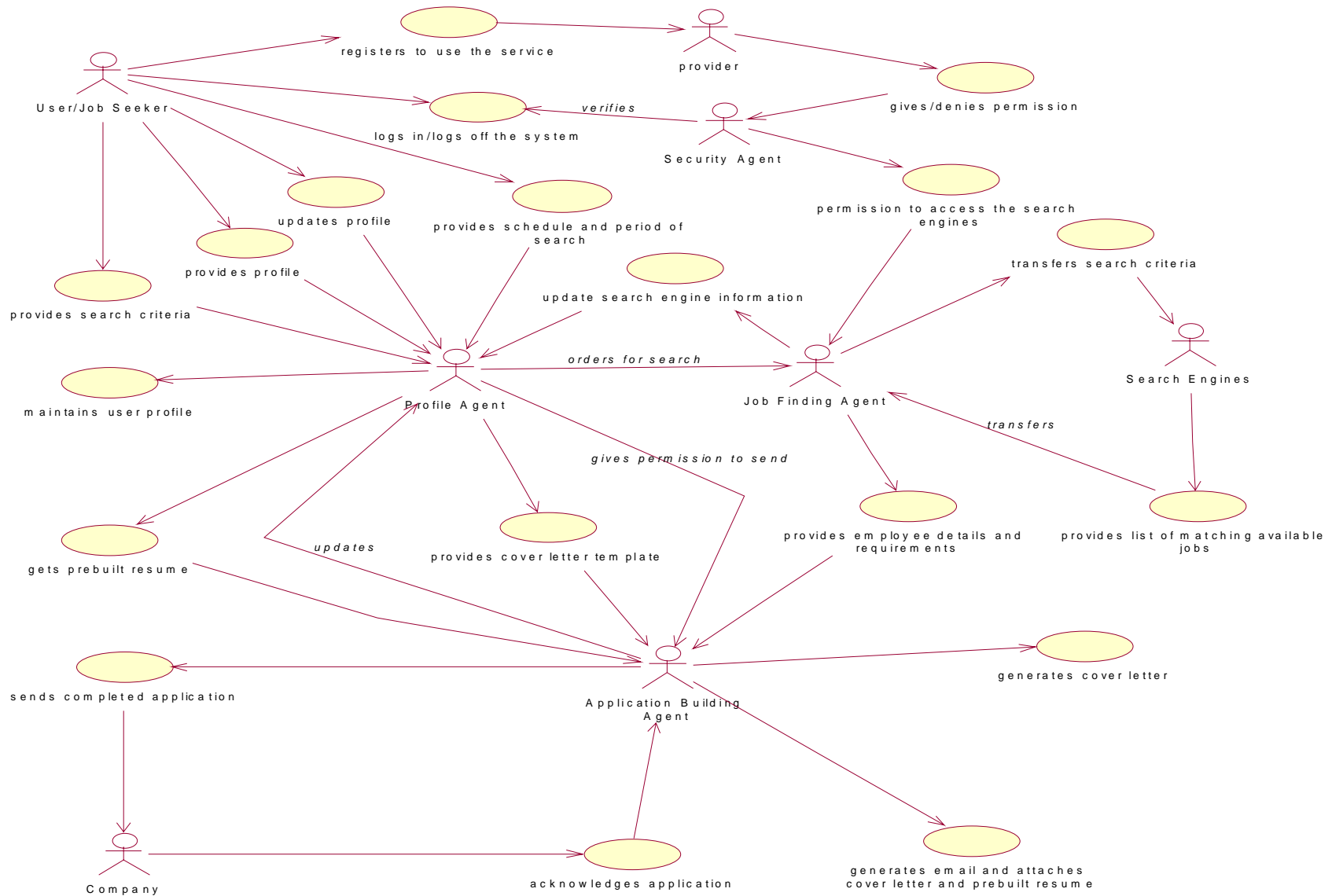
JFAS: SEQUENCE DIAGRAM



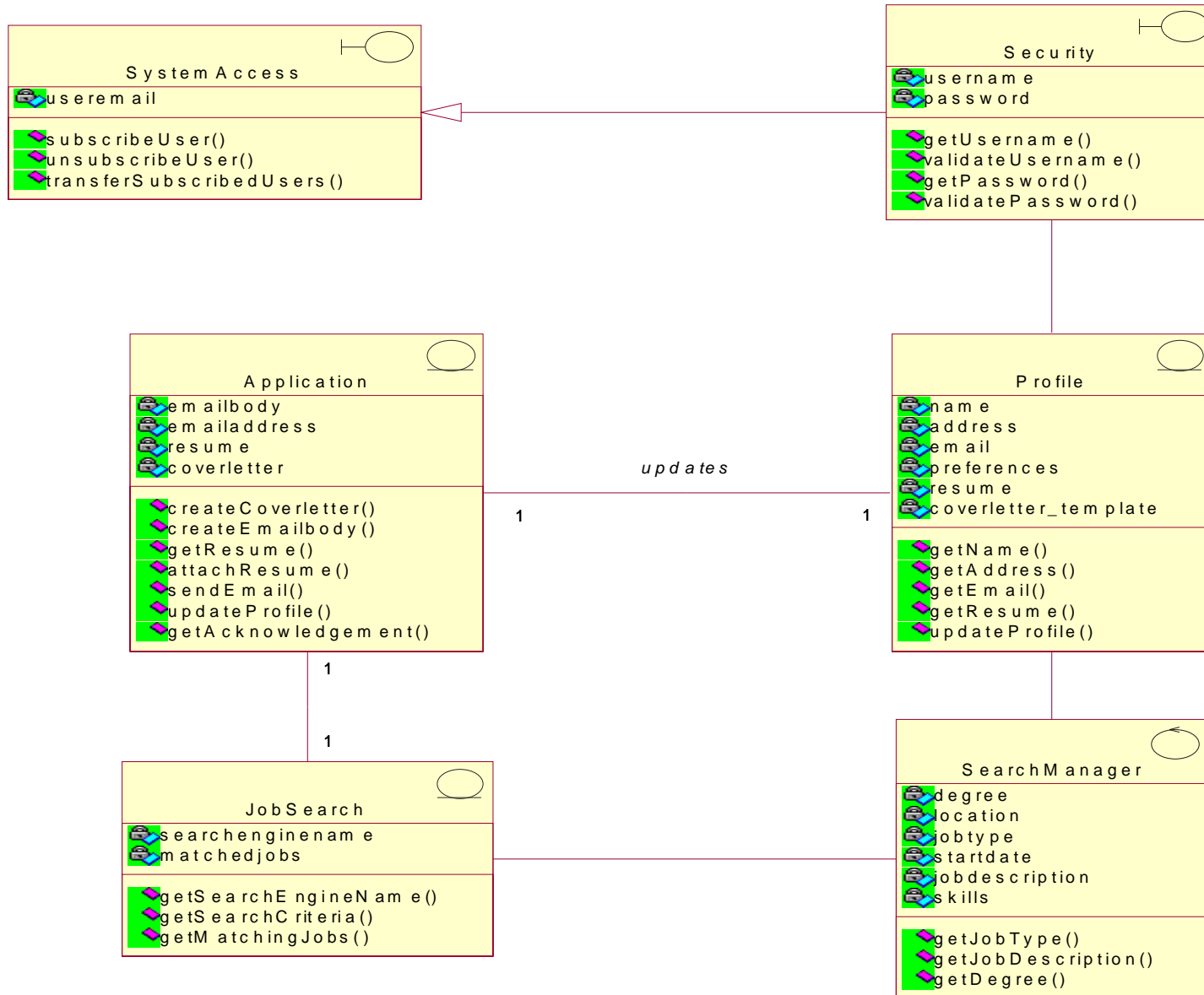
JFAS: ACTIVITY DIAGRAM



JFAS: COMPLETE USECASE DIAGRAM



JFAS: CLASS DIAGRAM



4.6 Use cases specification

Use Case Specification: generates cover letter

Brief description

This use case generates cover letter, which would be sent along with pre-built resume. Application Building Agent is considered an actor in this use case scenario. The cover letter generated by this use case is an important document of the system.

Flow of Events

Basic Flow

1. The system enters the date
2. The system fills in the employee address in the 'To' column
3. The system fills in the job seeker address in the 'From' column
4. The system automatically enters the standard paragraphs with the subject

Alternate Flow

The system generates an error message saying cover letter cannot be generated

Preconditions

1. Job Finding Agent must provide employee details and requirements
2. Profile Agent must provide cover letter template
3. Profile Agent must provide pre-built resume
4. Profile Agent must give permission to send the cover letter to the employee

Postconditions

After building cover letter, Agent Building Agent shall

1. generate an email body
2. attach cover letter and pre-built resume to email body
3. send completed application to the Company
4. update profile agent

Use case specification: provides employee details and requirements

Brief description

This use case provides details and requirements of the company that is suitable for the job seeker. Job Finding Agent is considered an actor in this scenario. The information generated by this use case is necessary for the Application Building Agent to generate cover letter.

Flow of Events

Basic Flow

1. The system collects Company name, address and web address

2. The system collects the name and email address of the contact person in the company to whom job application should be sent
3. The system collects the job requirements

Alternate Flow

The system generates a message – 'No jobs found'

Preconditions

1. Search Engine should be available to search for jobs
2. Job Search Criteria should be provided by the user
3. Job Finding Agent should transfer the job search criteria to Search engines

Postconditions

1. The generated employee details and requirements are passed to Application Building Agent.
2. Based on the above details, cover letter and Email body is generated
3. Pre-built Resume is attached to the above and sent to the employee

Use case specification: maintains user profile

Brief description

This use case maintains job seeker's personal information and job information in a profile. Profile Agent is considered an actor in this use case scenario. The information generated by this use case includes search criteria, pre-built resume and cover letter template. Search Criteria is used by Job Finding Agent. Pre-built resume and Cover letter template is used by Application Building Agent.

Flow of Events

Basic Flow

1. Profile Agent collects name, address, email from the user
2. Profile Agent also collects job search criteria from the user
3. Profile Agent maintains pre-built resume
4. Profile Agent obtains schedule and period of search from the user

Alternate Flow

Profile Agents sends an error message that it cannot maintain the profile

Preconditions

The user should have a valid username and password. Provider should have given permission to Security Agent to allow the user to log in or log off the system.

Postconditions

1. Profile Agent supplies the job search criteria to Job Finding Agent to start the search
2. Profile Agent supplies pre-built resume and cover letter template to Application Building Agent

3. User is allowed to update his/her profile at any time.

Use case specification – logs in/logs off the system

Brief description

This use case allows the user to log in or log off the system. User/Job Seeker and Security Agent are considered as actors in this use case scenario. This use case is important because it serves as an entry point to the system.

Flow of Events

Basic Flow

1. User/Job Seeker enters his username
2. User/Job Seeker enters his password

Alternative Flow

The user must enter his user name and password in order to gain access to the system.

Preconditions

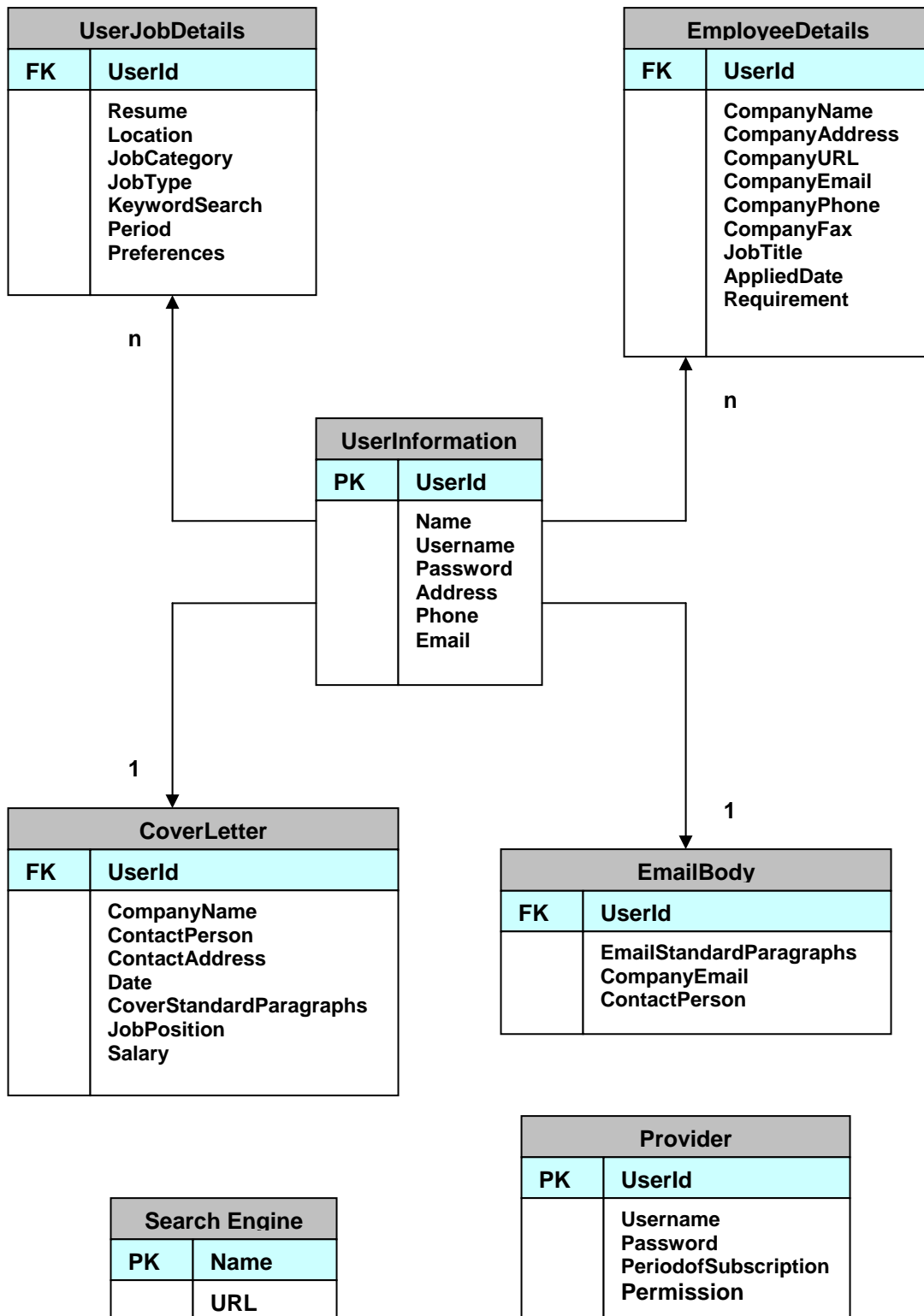
The user must have registered with the provider to use the service. The provider should have given permission to the Security Agent to verify username and password.

Postconditions

After the Security Agent verifies the user's username and password, user gains access to the system. User either uses the system or logs off the system.

5. Data Specification

5.1 ER diagram



5.2 Typical Data Definition

Search Engine

Fields	Descriptions	DataTypes
Name	Search Engine Name	Varchar(50)
URL	Web address of Search Engine	Varchar(50)

Provider

Fields	Descriptions	DataTypes
UserId	Unique user Id	Long
Username	Login name used to enter the system	Varchar(10)
Password	An element that ensures security	Varchar(15)
PeriodofSubscription	Time limit to access the system in days	Integer
Permission	Permission to access the system	Boolean

UserInformation

Fields	Descriptions	DataTypes
UserId	Unique user Id	Long
Name	Name of the user	Varchar(50)
Username	Login name used to enter the system	Varchar(10)
Password	An element that ensures security	Varchar(15)
Address	Address of the user	Varchar(50)
Phone	Phone number of the user	Long
Email	Email address of the user	Varchar(30)

UserJobDetails

Fields	Descriptions	DataTypes
UserId	Unique user Id	Long
Resume	Prebuilt resume of the user	Varchar(500)
Location	Preferred job location for the user	Varchar(25)
JobCategory	Expected job designation in the company	Varchar(25)
JobType	Full Time/Part Time	Varchar(15)
KeywordSearch	Job description	Varchar(30)
Period	Convenient time for the user to work	Time
Preferences	Includes salary, etc	Varchar(30)

EmployeeDetails

Fields	Descriptions	DataTypes
UserId	Unique Id of the user	Long
CompanyName	Name of the company	Varchar(30)

CompanyAddress	Address of the company	Varchar(50)
CompanyURL	Website address of the company	Varchar(30)
CompanyEmail	Email address of the company	Varchar(30)
CompanyPhone	Phone number of the company	Long
CompanyFax	Fax number of the company	Long
JobTitle	Job title to which user applied	Varchar(20)
AppliedDate	Date on which user applied	Date
Requirement	Availability job criteria for the candidate	Varchar(30)

CoverLetter

Fields	Descriptions	DataTypes
UserId	Unique Id of the user	Long
CompanyName	Name of the company	Varchar(30)
ContactPerson	Person to contact in the company	Varchar(20)
ContactAddress	Address of the company	Varchar(50)
Date	Date in which cover letter is written	Date
CoverStandardParagraphs	Standard paragraphs to be included in the cover letter	Varchar(100)
JobPosition	The job title to which user intend to apply	Varchar(20)
Salary	The salary expected from the company	double

EmailBody

Fields	Descriptions	DataTypes
UserId	Unique Id of the user	Long
EmailStandardParagraphs	Standard paragraphs to be included in the email body	Varchar(100)
CompanyEmail	Email Address of the Company	Varchar(30)
ContactPerson	Person to contact in the company	Varchar(20)

5 Inter agents Messages

The main issue in designing an agent is identifying the common language for communication. KQML is one such language. In KQML, messages identify the protocol for message transfer and encode the content.

There are two messaging services like update services and Query/Response services. Messages are framed and communicated between agents using the byte stream service provided by TCP port.

Messages are framed as follows:

Total Length			
Ver	Pcl	MsgTyp	Transaction Identifier
TimeStamp			
Message Specific Information		Padding	

Total Length – Length of the message

Ver – Version Number of the Protocol

Pcl – Protocol Operation Identifier

MsgTyp – Specific Operation Identifier

Transaction Identifier – Identifier assigned by agent invoking an update or query operation

TimeStamp – Time in Seconds

Message Specific Information – Variable length message specific information

Padding – padding the total message length to a multiple of four octets

1. Create Coverletter

Input Parameters

```
<coverletter>
  <companyName>String</companyName>
  <contactPerson>String</contactPerson>
  <contactAddress>String</contactAddress>
  <date>Calendar</date>
  <coverStandardParagraphs>String</coverStandardParagraphs>
  <jobPosition>String</JobPosition>
  <salary>float</Salary>
</coverletter>
```

Output Parameters

```
<coverletter>
  <coverletter>String</coverletter>
</coverletter>
```

Create Coverletter is the inter-agent message. The input parameters required for the message are companyName, contactPerson, contactAddress, date, coverStandardParagraphs, jobPosition and salary. The output parameter is the cover letter.

2. Send Email

Input Parameters

```
<email>
  <emailAddress>String</emailAddress>
  <emailBody>String</emailBody>
  <resume>String</resume>
  <coverletter>String</coverletter>
</email>
```

Output Parameters

```
<email>
  <confirmMessage>String</confirmMessage>
  <errorMessage>String</errorMessage>
</email>
```

Send Email is an inter-agent message. The input parameters include emailAddress, emailBody, resume and coverLetter. The output parameter is the confirmMessage saying that message has been sent successfully. The alternative output may be the error message saying message cannot be sent.

3. Get Acknowledgement

Input Parameters

```
<acknowledge>
  <resume>String</resume>
  <coverLetter>String</coverLetter>
</acknowledge>
```

Output Parameters

```
<acknowledge>
  <receipt>String</receipt>
</acknowledge>
```

Get Acknowledgement is an inter-agent message. The input parameters are resume and coverletter. The receiver sends the acknowledgement for receipt of the application which is the output parameter of the inter-agent message.

4. Update Profile

Input Parameters

```
<profile>
  <name>String</name>
  <address>String</address>
  <email>String</email>
  <preferences>String</preferences>
  <resume>String</resume>
  <coverletter_template>String</coverletter_template>
</profile>
```

Output parameters

```
<profile>
  <name>String</name>
  <address>String</address>
  <email>String</email>
  <preferences>String</preferences>
  <resume>String</resume>
  <coverletter_template>String</coverletter_template>
</profile>
```

Update Profile is an inter-agent message. The input parameters are name, address, email, preferences, resume and coverletter_template. Since the process is updation of input parameters, the output parameters are also same as input parameters.

7. Conclusion

In the course of this project we have been able to achieve a lot of what we had set out to do:

1. Identify the system specifications through the formulation activity that identifies the goals and the objectives of the project and established the scope.
2. Identify the system description through establishment of requirements, assumptions, and the content items that will be incorporated for the design.
3. Design the system with its architecture, components, internal architecture and communication: protocol, interaction, language and transport protocol.
4. The Unified modeling language was used to represent the system architecture.
5. Simple Object Access Protocol (SOAP) was proposed as communication protocol between the agents and the Web services.
6. The input and output parameters of each function introduced bellow has an XML format.

From our experience we found:

1. Building software application is an iterative process. A good software system is that which does not change significantly at the implementation stage. However, the software system is not “complete” until it has been implemented.
2. The software developer must strive to remain descriptive in his/her presentation, but will sometimes be required to be prescriptive (introduce constraints) to avoid software violations.

The produced software system agent based proposed in this project will assist the user for his/her job finding and could be reuse as whole or specific components for other software application.

Due to time constraints we were unable to implement the system and test it. A better understanding of the individual components and their interaction and communication could have helped us to identify design patterns and further promote reusability of the individual components.

In conclusion, Software agent based is a promising technique for the development of today’s business system. They proof their tremendous contribution in assisting individual, small, and large businesses.

8. References

1. H.S. Nwana, "Software Agents:an overview", knowl.Eng. rev., vol.11, no.3 pp 205-244, 1996.
2. Etzioni and D.S. Weld, "Intelligent agents on the internet: Fact, fiction, and forecast", IEEE Exp., vol 10, pp.44-49, 1995.
3. Dr B.H. Far SENG609-22 notes, 2002.
4. ian sommerville "Software Engineering". 6th Edition, 2001.
5. Roger. S. Pressman " Software Engineering a Practitioner Approach", Fifth Edition, 2001.
6. Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide".1999.
7. Gerhard Weiss" Multiagent Systems: a modern approach to distributed artificial Intelligent", 1999.
8. FIPA Standard –PC00089D"FIPA policies and Domains Specifications", 2001/08/10.
9. Terry Quatrani "Visual Modeling With Rational Rose 2000 and UML", 2000.
10. http://agents.umbc.edu/Applications_and_Software/Software/index.shtml

9. Acknowledgements

We would like to thanks Dr Far for the Course SENG-609.22 and the material it covered , It was set in the way that helps students understand the Software agent based technology and open the door four further investigations.