 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2003	Department: Electrical and Computer Engineering
	Document Name: An agent- based system for job finding	Document Type: Group Project

Group Project
An Agent-based System for
Job Finding

Course Instructor: Dr. Behrouz. H. Far
Name: Dong Liu, Zhizhong Li, Junling Huang
Date: Dec. 2003

Table of contents

Table of contents	- 1 -
List of Figures	- 2 -
List of Tables	- 3 -
1. Introduction.....	- 4 -
2. System Specifications	- 5 -
2.1 Business Case.....	- 5 -
2.2 System Description	- 5 -
2.3 Assumption	- 6 -
2.4 Requirement.....	- 6 -
2.4.1 General View	- 6 -
2.4.2 System Use Case.....	- 7 -
2.5 Wish List (not implemented)	- 18 -
3. Agent-based analysis and design	- 19 -
3.1 System architecture and agent schemata.....	- 19 -
3.2 Agent Collaboration as Use Case Realization	- 23 -
3.3 Message Specification	- 24 -
4. Detailed Design Document.....	- 30 -
4.1 Search internal database.....	- 30 -
4.2 Search outer sources	- 33 -
4.3 Post job.....	- 35 -
4.4 Search candidates.....	- 40 -
4.5 Class Diagram.....	- 46 -
5. Data Specification.....	- 47 -
5.1 Typical Data Definition	- 47 -
5.2 ER Diagram	- 49 -
6. Conclusion.....	- 50 -
7. References	- 51 -
8. Acknowledgement	- 52 -
Appendix	- 53 -
A.1 Message 1 Document.....	- 53 -
A.2 Message 2 Document.....	- 55 -
A.3 Message 3 Document.....	- 59 -
A.4 Message 4 Document.....	- 65 -
A.5 Message 5 Document.....	- 71 -
A.6 Message 6 Document.....	- 79 -
A.7 Message 7 Document.....	- 81 -
A.8 Message 8 Document.....	- 85 -

List of Figures

Figure 2-1 System overview	- 6 -
Figure 2-2 System use case diagram.....	- 8 -
Figure 2-3 Use case search job diagram	- 15 -
Figure 3-1 System architecture in agents.....	- 22 -
Figure 3-2 Agent collaboration in a scenario of use case “search job”	- 24 -
Figure 3-3 Diagram of message 1	- 26 -
Figure 3-4 Diagram of message 2.....	- 26 -
Figure 3-5 Diagram of message 3.....	- 26 -
Figure 3-6 Diagram of message 4.....	- 27 -
Figure 3-7 Diagram of message 5.....	- 28 -
Figure 3-8 Diagram of message 6.....	- 28 -
Figure 3-9 Diagram of message 7	- 29 -
Figure 3-10 Diagram of message 8.....	- 29 -
Figure 4-1 Collaboration diagram for use case Search internal database.....	- 31 -
Figure 4-2 Sequence diagram for use case Search internal database.....	- 32 -
Figure 4-3 Collaboration diagram for use case Search outer sources.....	- 34 -
Figure 4-4 Collaboration diagram for use case Search outer sources.....	- 35 -
Figure 4-5 Collaboration diagram for use case Post job.....	- 37 -
Figure 4-6 Sequence diagram for use case Post job	- 39 -
Figure 4-7 Collaboration diagram for use case Search candidates	- 41 -
Figure 4-8 Sequence diagram for use case Search candidates.....	- 45 -
Figure 4-9 Six important class diagrams applied in four use cases	- 46 -
Figure 5-1 Relationships of typical tables	- 49 -

List of Tables

Table 2-1 Use case register	- 8 -
Table 2-2 Use case logging in.....	- 9 -
Table 2-3 Use case logging out.....	- 10 -
Table 2-4 Use case maintain profile	- 11 -
Table 2-5 Use case post job	- 12 -
Table 2-6 Use case search candidates	- 13 -
Table 2-7 Use case search job.....	- 15 -
Table 2-8 Use case search internal database.....	- 16 -
Table 2-9 Use case search outer source	- 16 -
Table 2-10 Use case send express email.....	- 17 -
Table 3-1 Actors and their suggested roles.....	- 20 -
Table 3-2 Inner components and their suggested roles.....	- 20 -
Table 3-3 Functionalities and their suggested roles.....	- 20 -
Table 3-4 Role schema of individual agent	- 20 -
Table 3-5 Role schema of company agent.....	- 21 -
Table 3-6 Role schema of bank agent.....	- 21 -
Table 3-7 Role schema of outer-search agent.....	- 21 -
Table 3-8 Role schema of database agent.....	- 21 -
Table 3-9 Role schema of email agent.....	- 22 -
Table 3-10 Role schema of analysis agent.....	- 22 -
Table 3-11 Use cases and the agents involved.....	- 23 -
Table 3-12 The messages arranged by receiver agents.....	- 25 -
Table 3-13 Message diagrams and documents	- 25 -
Table 4-1 Use case search internal database.....	- 30 -
Table 4-2 Use case search outer source	- 33 -
Table 4-3 Use case post job	- 35 -
Table 4-4 Use case search candidates	- 40 -
Table 5-1 Individual profile definition	- 47 -
Table 5-2 Company profile definition	- 47 -
Table 5-3 Bill profile definition.....	- 48 -
Table 5-4 FavoriteJob profile definition.....	- 48 -
Table 5-5 JobPost profile definition	- 48 -

1. Introduction

Nowadays, job finding to some degree is the lifelong career for most people. Although there are many job searching engines available on Internet, job finding still can be very time consuming and frustrated. On the other hand, it is almost impossible for companies or recruiters to find qualified candidates quickly and at a low cost.

In this document, we discuss a multi-agent system for job finding, which is designed to provide autonomous and intelligent services not only for job seekers who want to find a decent job, but also for employers or recruiters with much more than the ability to post jobs. The document structure is as follows:

- For the system analysis, the system specifications are presented. Apart from the business case and system description, we make great efforts on the requirement specification in which the system use case is discussed in detail. We employ the requirements-driven development approach in this paper, and determined the multi-agent technology should be used for the system design and implementation
- For the general design, the system architecture is discussed, and we apply the “methodologies for agent-based analysis and design” for role identification and agent description
- For the detail design, OOAD and UML technology are used to specify the classes, sequences, which are lower level abstractions that can be implemented directly
- Last but not least, the ER Diagram, data directory and internal agent messages make the information for the system to based upon clearly

2. System Specifications

2.1 Business Case

Job finding for job seekers as well as candidate finding for companies or recruiters is time consuming and high costs. Although search engines on Internet greatly simplify the matching process, the continuously changing characteristics make the process difficult to be effective and convenient.

The existing job search system only provides services as follows:

- For Job Seekers, usually the service is free, and the only need is to register or to be a member. The information needed for registration is usually personal profile information including individual preference. Then, it provides the registered individual services like maintaining individual profile, job searching, and some featuring resources for resumes, interviews and self-assessment.
- For companies or recruiters, usually services are charged, and the company registration information needs to be verified. Then, the services provided can be activated. Usually it provides job posting, online recruitment, application instruction, and information resources for company culture, HR process or recruitment standards.

We try to add three more features, which autonomous and intelligent agents play a key role in helping job seekers and companies or recruiters make a perfect and immediate match:

- The personalized favorite job information for job seekers, and the personalized qualified candidates for companies or recruiters is presented when the user login. All the information is displayed and generated by dynamic analysis and statistics based on user's searching history and preference
- The express hot mail for job seeker's favorite job, and for companies' or recruiters' favorite candidate is sent out on time
- Many job search systems retain a great deal of information for clients but they do not share with other systems due to business reasons. We try to break it, and make the industry as a whole. The job searching is not only searching in our inner database, the outer search resources are also included.

2.2 System Description

The agent-based system for job finding proposed in this document is a multi-agent system aimed at helping job seekers and companies or recruiters to find each other in an immediate and effective way. Apart from a great deal of information we have, the outer search resources like the Job bank, the Monster are also searched by our intelligent agent,

which means users of our system instead of going from one search engine to another manually, they have a tireless search agent to do this tedious job automatically. In addition, an analysis agent works as a senior counselor to analyze and evaluate user's preference history. For the companies or recruiters, the system also provides a billing agent to manage payment issues. The system overview is shown in Figure 2-1.

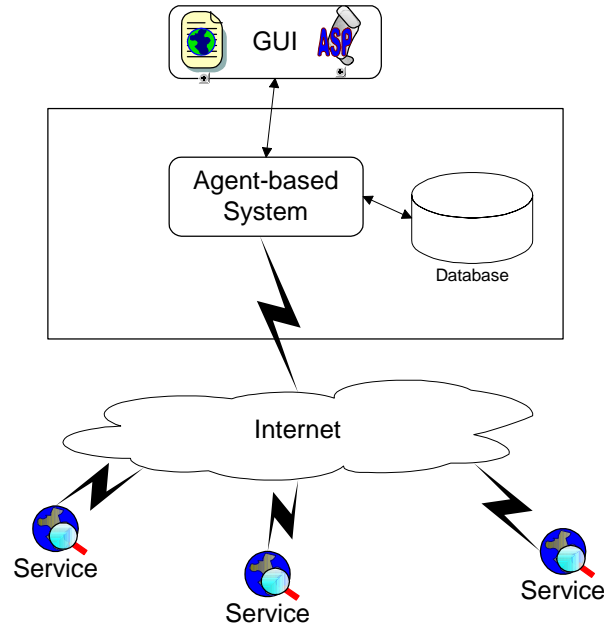


Figure 2-1 System overview

2.3 Assumption

- We assume that resources of the exist search engines on Internet are open, and have the same WSDL description meaning, their input and output messages have a standard syntax.
- We assume that the registered information of companies or recruiters is true and accurate so that the system do not need to verify the information
- We assume that the individual job seeker registered in this system has priority in applying for the job provided by companies or recruiters
- We assume that the preference and search history for individual job seeker and companies or recruiters is complete and up to date

2.4 Requirement

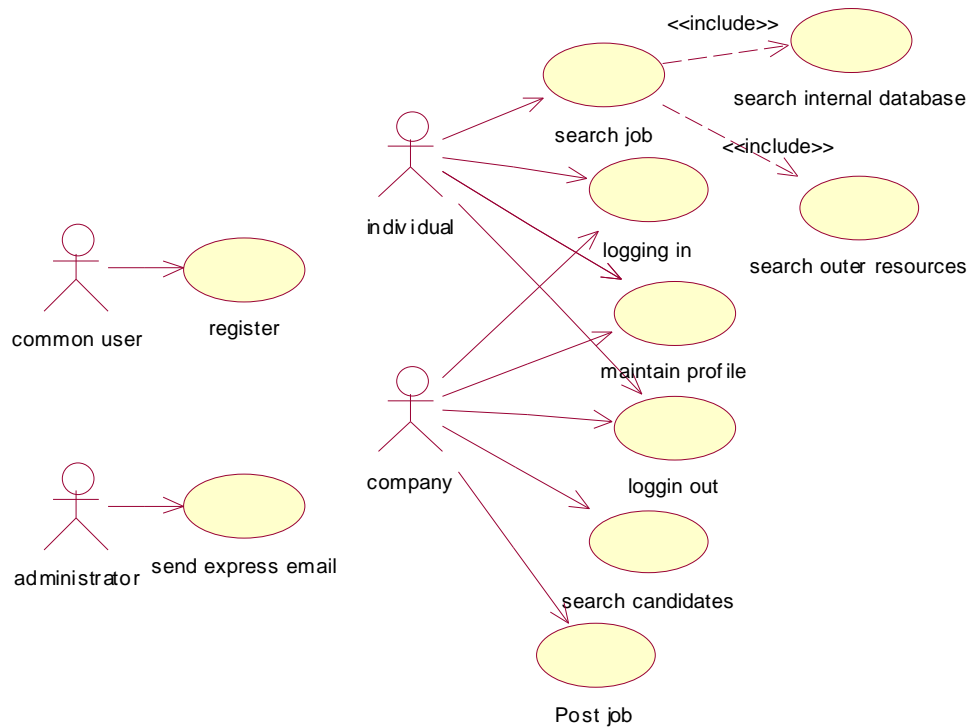
2.4.1 General View

The general process in this agent-based system for job finding is as follows:

- The common user (potential job seeker and companies or recruiters) browses the welcome home page for this system
- The common user registers in this system becoming the registered member, the registered job seeker in this system is called “Individual”, and the registered company or recruiter in this system is called “Company”
- The individual or company logs in this system
- For the individual, the personalized welcome home page is presented, and he/she can maintain his/her profile information, search job, or browse the newest job information generated and presented according to individual’s preference and job searching history. All services and information provided for individual are no cost at all
- Individual would get express mail form this system, which is about the latest individual favorite job information
- For the company, the personalized welcome home page, which presents the qualified candidates’ brief information for it’s latest job posting, provides the company a clear and fast way to find whom they want. However, in order to get the detail information about these candidates, certain amount of money should be paid by the company
- For the company, it also can maintain its profile information, do search, and post job advertisement
- Company would get express mail form this system, which is about the latest qualified candidates

2.4.2 System Use Case

The detail system Use case is discussed below, and it would give us much more clear specifications about the whole system:



System use case Diagram

Figure 2-2 System use case diagram

Table 2-1 Use case register

Use Case:	Register
Actor:	Common user
Short description:	The Registration use case is where the User(company or individual) builds his/her new account file
Pre-Conditions:	None
Post-Conditions:	Account file is created and added to the account database. The User is logged in
Main flow:	<ul style="list-style-type: none"> •The system prompts the User for all the necessary information. •The User fills all the required information and then submits. (A1) (A2) (A3) •The system validates the entered information, making sure that the entered username is new and valid, and that all the required information is complete •The system adds the registration file to the account database.

	<p>The current date is regarded as the creation data</p> <ul style="list-style-type: none"> •The User is logged in. The system displays a message indicating that the account is successfully created and the User is logged in. Once the account is created, the User is considered signed in
Alternative flow:	<p><A1> Already existed account name .If the account name already exists, the System will give the User the opportunity to change to a new account name. The Basic Flow continues where the User enters new information</p> <p><A2> Incomplete or invalid information. If the User doesn't enter complete information, the following occurs:</p> <ol style="list-style-type: none"> 1. The system describes the reasons why the User failed registration. 2. The system presents the User with suggestions for changes necessary to allow the User to pass registration. 3. The system prompts the User to re-enter the incomplete or invalid information. 4. The Basic Flow continues where the User enters new information (see step 2 of the Basic Flow) <p><A3> Registration cancellation. If the User doesn't want to continue the registration, he or she can cancel the registration process at any time. Once this happens, the System will return to the home page</p>

Table 2-2 Use case logging in

Use Case:	Logging in
Actor:	Individual or Company
Short description:	The login page allows a registered user (individual or company) to sign in to his/her personal account in this system.
Pre-Conditions:	The user has registered in this system
Post-Conditions:	The valid user can access his/her account profile, can use services provided by the system, and can view the personalized information automatically collected and organized by the system
Main flow:	<ul style="list-style-type: none"> •A registered user enters the user name and password in the fields provided, and then, submits to the job finding system. The user name and password input fields would be Case-sensitive •The system checks the input integrity <A1>

	<ul style="list-style-type: none"> •The system checks the validity of the user name and password <A2> •The system gets the valid user’s profile from its database, and set the valid user or company status to “Log in” •The valid user can access to his/her account profile, and services provided by the system •The system collects and organizes personalized information, and then, presents this information to the valid user (individual or company). For the valid individual, the personalized information is the latest job postings match the individual’s favorite career goal; for the valid company, the personalized information can be the qualified candidates’ contact information and resume summary for the company’s latest job posting
Alternative flow:	<p>< A1 >If the user did not enter user name or password, then the system would display “please input your user name” or “please input your password” message, and keep the cursor focusing on the corresponded field</p> <p>< A2 >If the system could not find the user name or the password does not match, then the system would display “Invalid username or password” message, and keep the cursor focusing on the corresponded field</p> <p>< A3 >If the user could not recall his/her password, then the system would bring the user to a page or given steps that the user could use to access his/her account. The page or given steps may be some questions about the user’s telephone number, email address, or “forget password questions”</p>

Table 2-3 Use case logging out

Use Case:	Logging out
Actor:	Individual or Company
Short description:	To ensure the user’s information is secure, the user should click the “Logout” button when he/she finished his/her operations or left his/her computer. The system would end the user’s session, save all changes, clean up and draw back all the resources
Pre-Conditions:	The user has registered in this system
Post-Conditions:	The system would end the user’s session, save all changes, clean up and draw all the resources back, and then, the system would present “Logout successful” page to the user
Main flow:	<ul style="list-style-type: none"> •The user clicks the “Logout” button

	<ul style="list-style-type: none"> •The system checks the user’s “Log in” status <A1> •The system saves all changes made by the valid user •The system ends the valid user’s session, and sets the user to “Log out” status •The system cleans up and draws all the resources back
Alternative flow:	< A1 >If the user’s status was not “Log in”, then the system would display “please Logging in first” message to the user

Table 2-4 Use case maintain profile

Use Case:	Maintain profile
Actor:	Individual or Company
Short description:	A valid user (individual or company) can modify any of his/her current account information, overwrite any unwanted information. If the user changed his/her password, the system would ask the user to enter the current password to verify the changes
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system saved all changes into its database, updated the user’s current account information, and then return to the personalized welcome web page
Main flow:	<ul style="list-style-type: none"> •The valid user clicks the “maintain profile” link on the personalized web page •The user’s profile information would be viewed and updated in the fields provided. All fields marked with an asterisk “*” are required •When the valid user finished, he/she should click the “submit” button <A1> •The system checks the input integrity <A2> •The system writes the changes into its database, updates the user’s current account information <A3> •The system displays “re-login” dialog if the password was changed •The system return to the personalized welcome web page
Alternative flow:	< A1 >If the valid user click the “cancel” button, then the system would abort all changes and return to the personalized web page

	<p>< A2 >If not all of fields marked with an asterisk “*” were filled up, then the system would display “please input all the required information” message, and keep the cursor focusing on the corresponded fields</p> <p>< A3 >If there were some abnormal errors happened, then the system could not update the information in database, and it would return fail message</p>
--	---

Table 2-5 Use case post job

Use Case:	Post job
Actor:	Company
Short description:	The Post Job use case is where the company posts a new job.
Pre-Conditions:	The company has successfully logged in
Post-Conditions:	A job post is created and added to the job-post database
Main flow:	<ul style="list-style-type: none"> •The system prompts the User for all the necessary information about the job •The company user fills all the required job information and then submits. (A1) •The system validates the entered information, making sure that all the required information is complete •The system prompts the User for billing information. The following information is included: •The company user fills all the billing information and then submits. (A3) •The system validates the billing information, making sure that all the billing information is correct •The system takes out money from company’s billing account and sends receipt to the company user by Email. (A4) •The system adds the new job post to the job-post database. The current date is recorded to be the creation data •The system adds the new job post to the company user profile. The current date is regarded as the creation data •The system displays a message informing the company that the job is successfully posted
Alternative flow:	<p><A1> If the User doesn’t enter complete information, the following occurs:</p> <ol style="list-style-type: none"> 1. The system describes where the information is incorrect or incomplete.

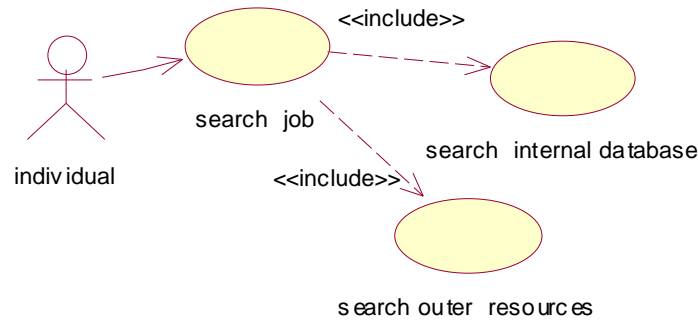
	<ol style="list-style-type: none"> 2. The system presents the User with suggestions for changes necessary to allow the User to make a successful post. 3. The system prompts the User to re-enter the incomplete or invalid information. <p><A2>If the company user doesn't want to continue the posting, he or she can cancel the posting process at any time. Once this happens, the System will return to the company user home page</p> <p><A3>If the User doesn't enter complete billing information, the following occurs:</p> <ol style="list-style-type: none"> 1. The system describes where the information is incorrect or incomplete. 2. The system presents the User with suggestions for changes necessary to allow the User to make a successful post 3. The system prompts the User to re-enter the incomplete or invalid billing information <p><A4>If the System cannot access the billing process, the following occurs:</p> <ol style="list-style-type: none"> 1. The system informs the company that the billing process cannot be accessed 2. The system prompts the User to re-enter the incomplete or invalid billing information
--	--

Table 2-6 Use case search candidates

Use Case:	Search candidates
Actor:	Company
Short description:	The search candidates use case is where the company user searches candidates
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system presents the qualified candidates to the company user
Main flow:	<ul style="list-style-type: none"> •The system prompts the company user for all the search criteria about available candidates •The company user fills parts or all the search criteria and then submits •The system searches the available candidates following company user's request •The System lists all the summaries of matched candidates to

	<p>company user</p> <ul style="list-style-type: none"> •The System prompts company to see the details of the most interested resumes. (A1) (A2) •The company user selects the most interested resumes and submits •The system prompts the User for billing information. The following information is included: •The company user fills all the billing information and then submits. (A3) •The system validates the billing information, making sure that all the billing information is correct •The system takes out money from company’s billing account and send receipt to the company user by Email. (A4) •The system sends all selected resume to company user’s Email address •The system adds the search results to the Search log. The following information is included: •The system displays a message informing the company that the resumes are successfully sent
<p>Alternative flow:</p>	<p><A1>If the company user want to continue the search, he or she can redo the searching process at any time. Once this happens, the System will go to the search page</p> <p><A2>If the company user doesn’t want to continue the search, he or she can cancel the searching process at any time. Once this happens, the System will return to the company user home page</p> <p><A3>If the User doesn’t enter complete billing information, the following occurs:</p> <ol style="list-style-type: none"> 1. The system describes where the information is incorrect or incomplete 2. The system presents the User with suggestions for changes necessary to allow the User to make a successful post. 3. The system prompts the User to re-enter the incomplete or invalid billing information 4. The Basic Flow continues where the User enters new information <p><A4> If the System cannot access the billing process, the following occurs:</p>

	<ol style="list-style-type: none"> 1. The system informs the company that the billing process cannot be accessed 2. The system prompts the User to re-enter the incomplete or invalid billing information
--	---



Search job Diagram

Figure 2-3 Use case search job diagram

Table 2-7 Use case search job

Use Case:	Search job
Actor:	Individual
Short description:	The user who has logged in search the jobs she/he wants
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system search internal database as well as outer resources for users
Main flow:	<ul style="list-style-type: none"> • The user navigates to the search-job page • The system displays the search-job page • The user input information on the search-job page • The user submits a search-job request • The system generates a new search-job result page <p>Con</p> <p>Co1 The system searches the internal database based on the request</p> <p>Co2 The system searches the outer job resources based on the request</p> <p>Noc</p> <ul style="list-style-type: none"> • The system fills the search result into the search-job result page • The system displays the search-job result page to the user • The user selects a job item on the search-job result page • The system displays the detailed job page • The user selects submit-resume on the detailed job page • The system sends the user’s resume to the job-poster

	<ul style="list-style-type: none"> The system displays resume-sent page to the user
Alternative flow:	<p>< A1 > If the user does not input necessary information on the search-job page, then</p> <ol style="list-style-type: none"> The system displays a message that the user should input certain information for the search The user confirms the message The system displays the search-job page <p>< A2 > If the user cancels the search during the system is doing the search job, then</p> <ol style="list-style-type: none"> The system stops the internal database and outer resource search The system displayed the welcome-page
Include use cases	<ol style="list-style-type: none"> Search internal database Search outer resources

Table 2-8 Use case search internal database

Use Case:	Search internal database
Actor:	Individual
Short description:	The system searches the internal database based on the user's request
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system search the internal database information for users
Main flow:	<ul style="list-style-type: none"> The user submits the search-job request The system generates the search-database profile The system searches the database using the profile The system gets the search result The system end the search-database
Alternative flow:	<p>< A1 > If the system cannot get the search-database result for a long time, then</p> <ol style="list-style-type: none"> The system cancels this search-database The system sets the result to be "exception with database" <p>< A2 > If the user cancel the search, then The system stops the search</p>

Table 2-9 Use case search outer source

Use Case:	Search outer sources
Actor:	Individual
Short description:	The system searches the outer resources based on the user's request
Pre-Conditions:	The user has successfully logged in

Post-Conditions	The system searches the outer resources for users
Main flow:	<ul style="list-style-type: none"> • The user submits the search-job request • The system generates the search-outer profile • The system sends the search-outer profile to outer job resource • The system receives the search result from outer job resource • The system ends the search
Alternative flow:	<p>< A1 > If the system cannot get the search-outer result for a long time, then</p> <ol style="list-style-type: none"> 1. The system cancels this search-outer 2. The system sets the result to be “exception with network” <p>< A2 > If the user cancel the search, then The system stops the search</p>

Table 2-10 Use case send express email

Use Case:	Send express email
Actor:	Administrator
Short description:	The system sends express email to the registered user everyday
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	Express email is sent out to user
Main flow:	<ul style="list-style-type: none"> • The system starts to analyze the new job post when it is 2 o'clock; • The system select one user from the registered user list • Do <ol style="list-style-type: none"> 1. The system searches the job posted in the past day in the internal database based on the user's profile 2. The system gets the search result 3. The system composes an email with the search result 4. The system sends out the email 5. The system selects the next user in the registered user list <p>While there are still user not processed in the list</p> <ul style="list-style-type: none"> • The system log that the sending emails of that day has been finished
Alternative flow:	<p>< A1 > If the search result is empty, then</p> <ol style="list-style-type: none"> 1. The system does not compose the email 2. The system selects the next user in the registered user list

2.5 Wish List (not implemented)

- The system should provide services for job seekers to generate and prepare resumes and cover letters
- The system should help job seekers to practice the interview process and provide evaluation tools
- The system should help the companies or recruiters to manage the whole employment process
- The system should apply the automatic verification process for the registered information of companies or recruiters

3. Agent-based analysis and design

The requirements described in the first part of this report are analyzed, and the agents are identified. The analysis method adopted in our project is based on the methodology of Gaia that is first introduced by Wooldridge etc. [2]. The role model is developed and the role agent schemata are presented in this part. The architecture of this system is composed with the agents and their interactions. The messages, as one of the most important artefacts in agent-based analysis and design, are defined as well. This part is comprised with three subsections: system architecture and agent schemata, agent collaboration and messages.

3.1 System architecture and agent schemata

Based on the requirements described in the previous part of this report, we can carry out agent-base analysis and design in this part. The first step is to analyze the requirements and identify the agents that we will use to construct the whole system. We apply the Gaia methodology for agent-based analysis and design in our project, but the details are not exactly the same as Gaia. The concepts of role and role model are explained in the papers. However, how to identify the roles from requirements is not presented. As a methodology, we need some kind of guidelines to help find the roles and develop the role model. Like the activity of object elicitation in object-oriented analysis and design, finding agents is considered as a problem. In our project, we apply such a guideline as the followings.

1. Identify the stakeholders of the system, find those who will interact with the system as either a person or a system, i.e. actors in use-case modeling, and consider what roles inside the system will take care those interactions;
2. Identify the predefined components or parts that is well defined and developed even before the system is designed, e.g. database, and consider what roles will take care the interfaces and logics related with such components and parts;
3. With the roles identifies in the previous 2 steps, review the requirements, consider what roles the system need to perform the functionalities and non-functional requests in the requirements.

Following this guideline, we analyze the system and get the roles. Table 3-1 lists actors and their suggested roles for this system; Table 3-2 inner component and suggested roles; and Table 3-3 the functionalities and suggested roles.

Table 3-1 Actors and their suggested roles

Actors	Suggested Roles	Comments
Individual User	Individual Agent	Responsible for all the tasks that the individual users want the system to perform
Company User	Company Agent	Responsible for all the tasks that the company users want the system to perform
Bank	Bank Agent	Responsible for all the tasks that the system needs to perform with the bank
Outer job resources	Outer-search Agent	Responsible for all the tasks that the system needs to perform with the outer job resources

Table 3-2 Inner components and their suggested roles

Inner Components	Suggested Roles	Comments
Database	Database Agent	Responsible for all the tasks that system needs to perform

Table 3-3 Functionalities and their suggested roles

Functionalities	Suggested Roles	Comments
Analyze the jobs	Analysis Agent	Responsible for analyzing the posted jobs to generate the personalized pages and emails
Send emails	Email Agent	Responsible for sending emails

So far 7 roles have been identified, and we still need to refine the role model in 2 ways: one is to develop the system architecture in agents and their interactions, and the other is to develop the role schemata. According to Gaia, a role is defined by protocols, permissions and responsibilities. In this project, we do not adopt this schema format, but use the one developed by Kendall [4]. A role is defined by its responsibilities and the corresponding collaborators. With the collaborators of each role identified, the system architecture in agents can be developed easily.

Table 3-4 Role schema of individual agent

Role: Individual Agent	
Responsibilities	Collaborators
Generate Personalized welcome page	(send, receive message) Analysis Agent
Maintain profile	(send, receive message) Database Agent

Search job	(send, receive message) Database Agent (send, receive message) Outer-search Agent
------------	--

Table 3-5 Role schema of company agent

Role: Company Agent	
Responsibilities	Collaborators
Maintain profile	(send, receive message) Database Agent
Post job information	(send message) Database Agent (send, receive message) Billing Agent
Search proper workers	(receive message) database agent (send, receive message) billing agent

Table 3-6 Role schema of bank agent

Role: Bank Agent	
Responsibilities	Collaborators
Billing	(receive message) company agent
Log Billing	(send message) email agent (send message) database agent

Table 3-7 Role schema of outer-search agent

Role: Outer-search Agent	
Responsibilities	Collaborators
Supply outer job search result	(send, receive message) Individual agent
Convert the request to SOAP	
Search outer job resource	(send, receive message) outer job resource (send, receive message) outer source agent
Convert the search result to XML	

Table 3-8 Role schema of database agent

Role: Database Agent	
Responsibilities	Collaborators
Search/update the database	(send, receive message) Individual Agent, Company Agent, Analysis Agent, Bank Agent
Convert the request to SQL	

Convert the search result to XML	
----------------------------------	--

Table 3-9 Role schema of email agent

Role: Email Agent	
Responsibilities	Collaborators
Send Email	(receive message) Analysis Agent, Bank Agent

Table 3-10 Role schema of analysis agent

Role: Analysis Agent	
Responsibilities	Collaborators
Compile the request for Database Agent	
Search the database	(send, receive message) Database Agent
Convert the search result to HTML	(send message) Individual Agent, Company Agent, Email Agent

The system architecture of agents is shown in Figure 3-1.

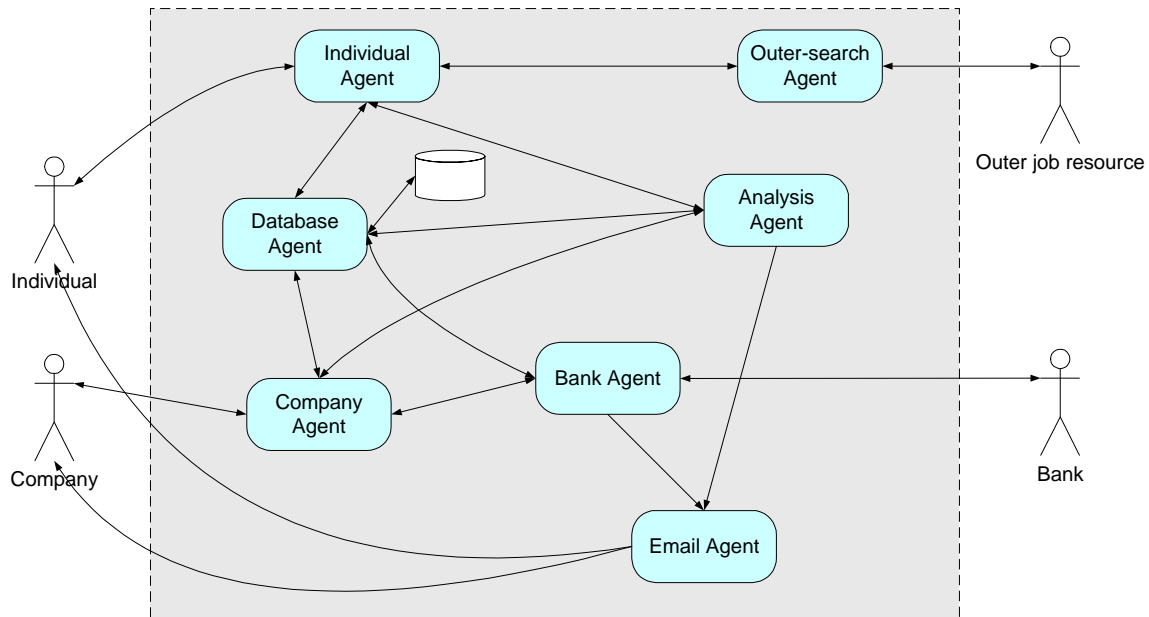


Figure 3-1 System architecture in agents

3.2 Agent Collaboration as Use Case Realization

In OOAD (Object-Oriented Analysis and Design), the use cases are “realized” by collaboration in sequence diagrams or collaboration diagrams. The realization means that all the events are carried out by the objects and their interactions. In agent-based analysis and design, it turns different when we face the agents rather than the objects. In OOAD, the collaboration/sequence diagrams can help find the responsibilities of objects. What we can get if we analyze the collaboration of agents? I think that it is the messages and their flows. I will try to reuse the notations of UML to represent the collaboration among agents and the messaging process.

A scenario of use case “Search job” is showed in the following diagram. In that diagram we can find that there are 5 kinds of request messages and 5 kinds of reply messages, and each agent is responsible for processing and generating some messages. The system’s behavior described in the use case is realized in the messaging process. Developing such collaboration diagrams of agents can help review the use case and find the messages and their manipulation in the system. The use cases and corresponding agents to realize them are listed in the following table.

Table 3-11 Use cases and the agents involved

Use cases	Agents involved
Search job	Individual Agent, Database Agent, Outer-search Agent
Search candidate	Company Agent, Database Agent, Bank Agent, Email Agent
Log in	Individual/Company Agent, Analysis Agent, Database Agent
Send express email	Analysis Agent, Database Agent, Email Agent

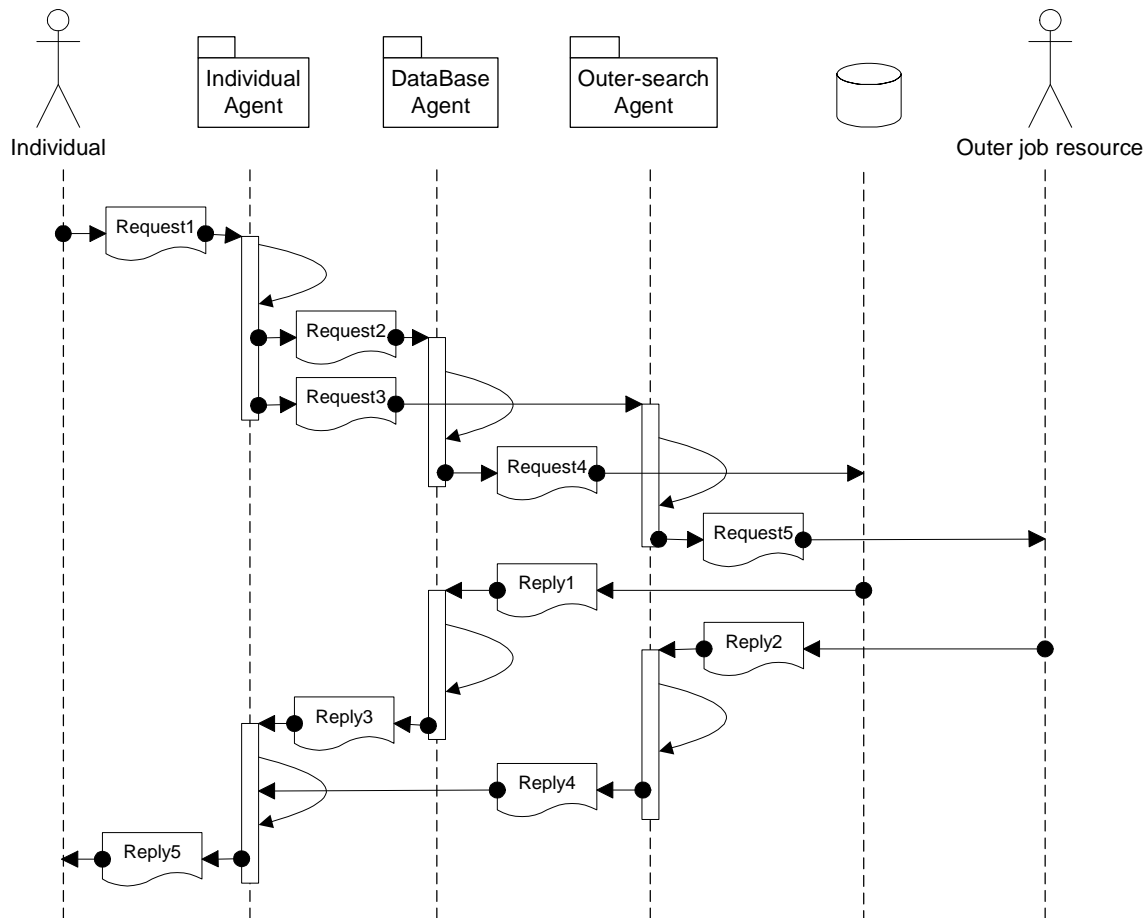


Figure 3-2 Agent collaboration in a scenario of use case “search job”

3.3 Message Specification

The messages involved in the system can be classified into 2 types: the messages that are generated and will be consumed inside the system and those that will be consumed by the outer systems or generated by them. The second type of messages has to be specified according the standard of the outer systems. In this project, it is assumed that the services of outer systems are based on SOAP (Simple Object Access Protocol). However, the details about the messages still need to be considered later. Actually there are only a few web services related with jobs that have been published (<http://uddi.microsoft.com/>). So now we consider only the messages inside the system, especially the information contents and structures.

The contents of messages are related to the data of the system. In this project, the messages are specify after we develop the Entity-Relation model of the system that is arranged in the next part of this report somehow. We find that there are so many different messages in the system just as those shown in Figure 3-2. In this project we use such a process to analyze the messages:

1. Tag each messages with its producer (sender), consumer (receiver) and task, and then categorize all the messages according to the **consumer**, of each message to get a table like Table 3-12;
2. Analyze each class of messages to figure out the information content and structure of that class.

Table 3-12 lists all the agents as receiver and the messages they will consume. The reasons that we analyze the messages in such a process are

1. A message will affect both the receiver and the sender, but the receiver should be considered at the first place because the logic of the receiver can never be realized till the incoming message is known;
2. We need to analyze the similarity of a class of messages to specify them with one schema.

Table 3-12 The messages arranged by receiver agents

Agents (receiver)	Messages	ID
Analysis Agent	(from Company Agent, Individual Agent) User ID	1
	(from Database Agent) user ID, type (job-supplier/job-applicant), content	2
Billing Agent	User ID, type (post jobs/buy resume), content	3
Company Agent	User ID, type (update/search/billing), content	4
Database Agent	User ID, type (update/search), content (user profile/search condition)	5
Email Agent	To address, email subject, content	6
Individual Agent	User ID, type(update/search), content	7
Outer-search Agent	User ID, type(search), search condition	8

The detailed contents and structures about the messages are specified in XML schemata which main diagrams and documents are listed in Table 3-13.

Table 3-13 Message diagrams and documents

Message ID	Diagram	Document
1	Figure 3-3	A.1 Message 1 Document
2	Figure 3-4	A.2 Message 2 Document
3	Figure 3-5	A.3 Message 3 Document
4	Figure 3-6	A.4 Message 4 Document
5	Figure 3-7	A.5 Message 5 Document
6	Figure 3-8	A.6 Message 6 Document
7	Figure 3-9	A.7 Message 7 Document
8	Figure 3-10	A.8 Message 8 Document

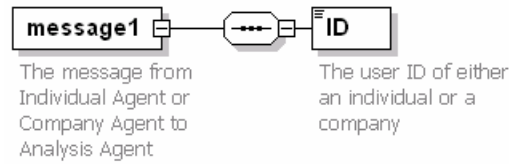


Figure 3-3 Diagram of message 1

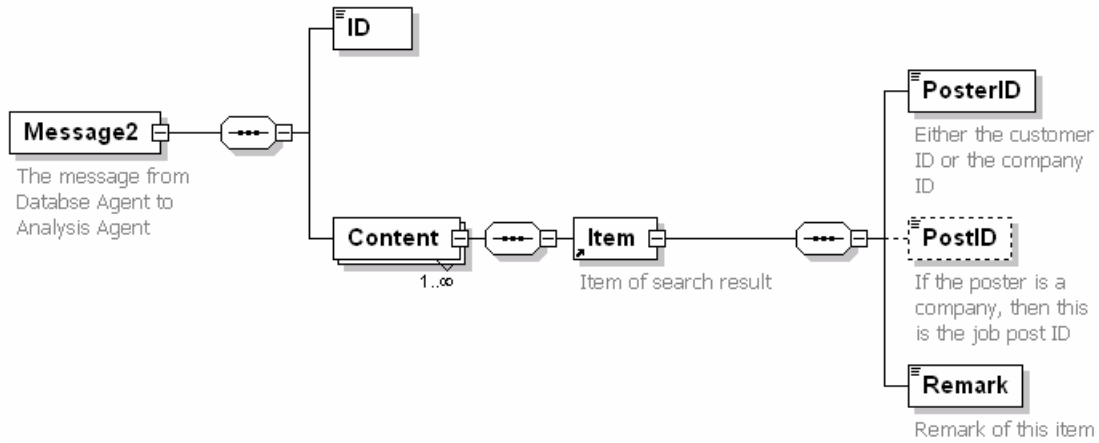


Figure 3-4 Diagram of message 2

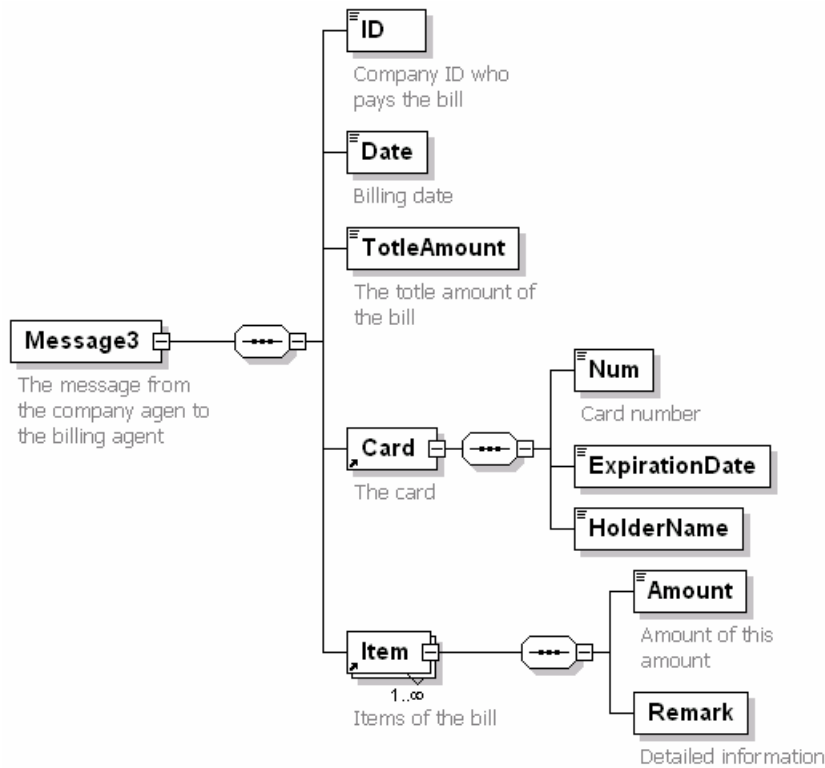


Figure 3-5 Diagram of message 3

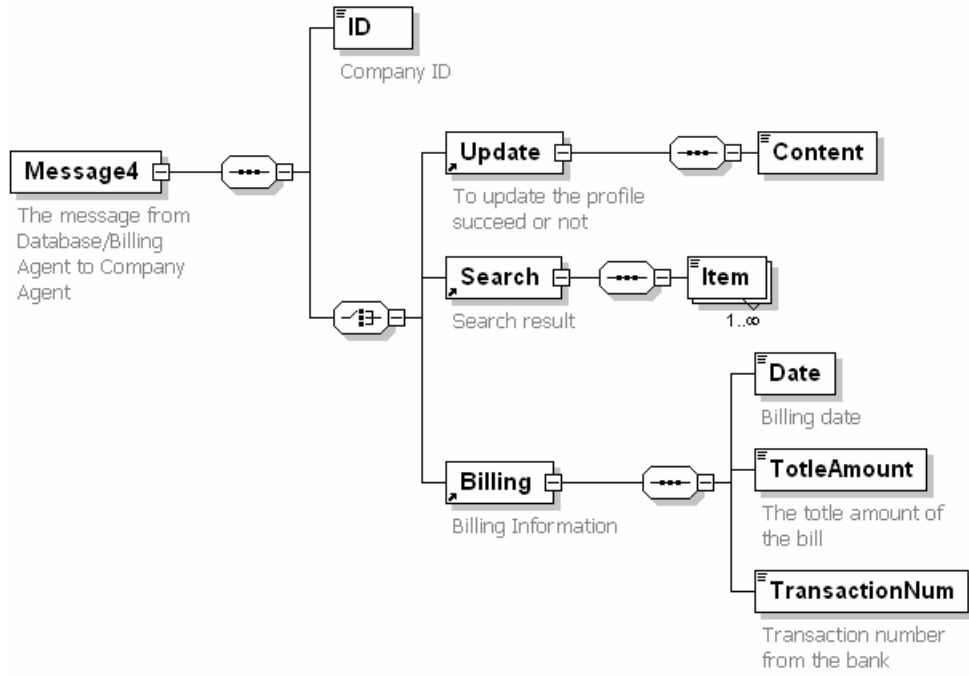


Figure 3-6 Diagram of message 4

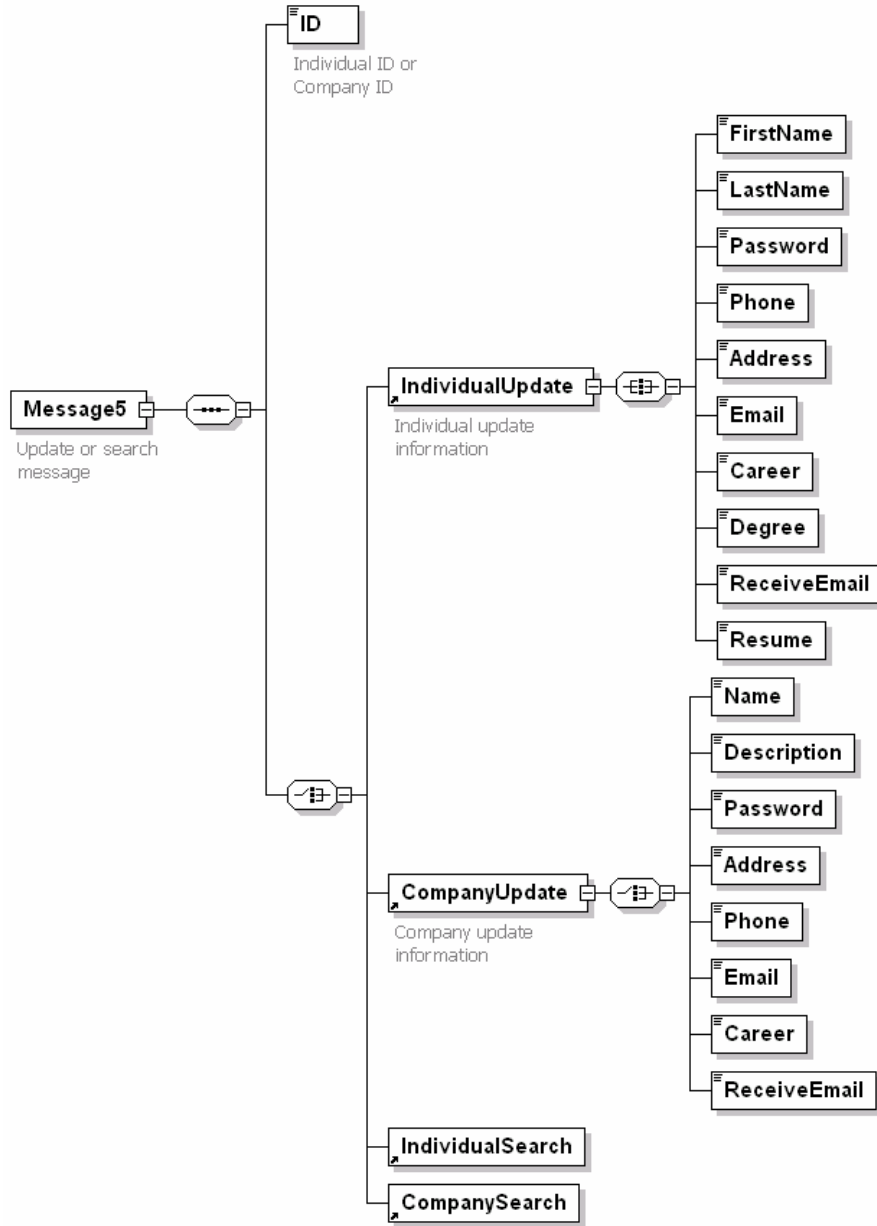


Figure 3-7 Diagram of message 5

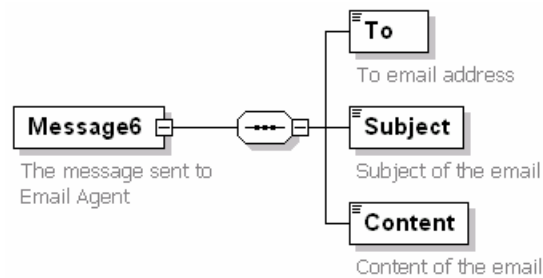


Figure 3-8 Diagram of message 6

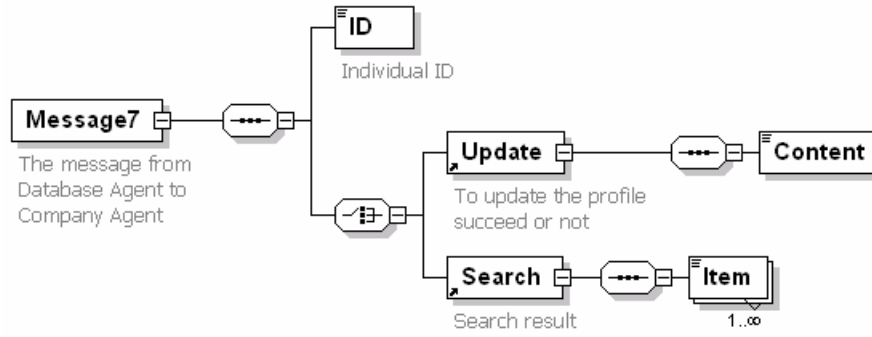


Figure 3-9 Diagram of message 7

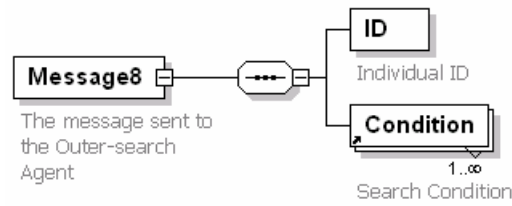


Figure 3-10 Diagram of message 8

4. Detailed Design Document

In this part we applied two behavioral diagrams to view the dynamic parts of a system: Collaboration diagram and Sequence diagram. We build these detailed behavioral diagrams for four use cases: Search internal database, Search outer sources, Post job and Search candidates. These four use cases are implemented by individual agent and company agent. They are the most important use cases in this agent-based system for job finding. After explaining the four use cases, we give class diagrams for six important classes in the detailed design.

4.1 Search internal database

Table 4-1 Use case search internal database

Use Case:	Search internal database
Actor:	Individual
Short description:	The system searches the internal database based on the user's request
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system search the internal database information for users
Main flow:	<ul style="list-style-type: none"> • The user submits the search-job request • The system generates the search-database profile • The system searches the database using the profile • The system gets the search result • The system end the search-database
Alternative flow:	<p>< A1 > If the system cannot get the search-database result for a long time, then</p> <ol style="list-style-type: none"> 3. The system cancels this search-database 4. The system sets the result to be "exception with database" <p>< A2 > If the user cancel the search, then The system stops the search</p>

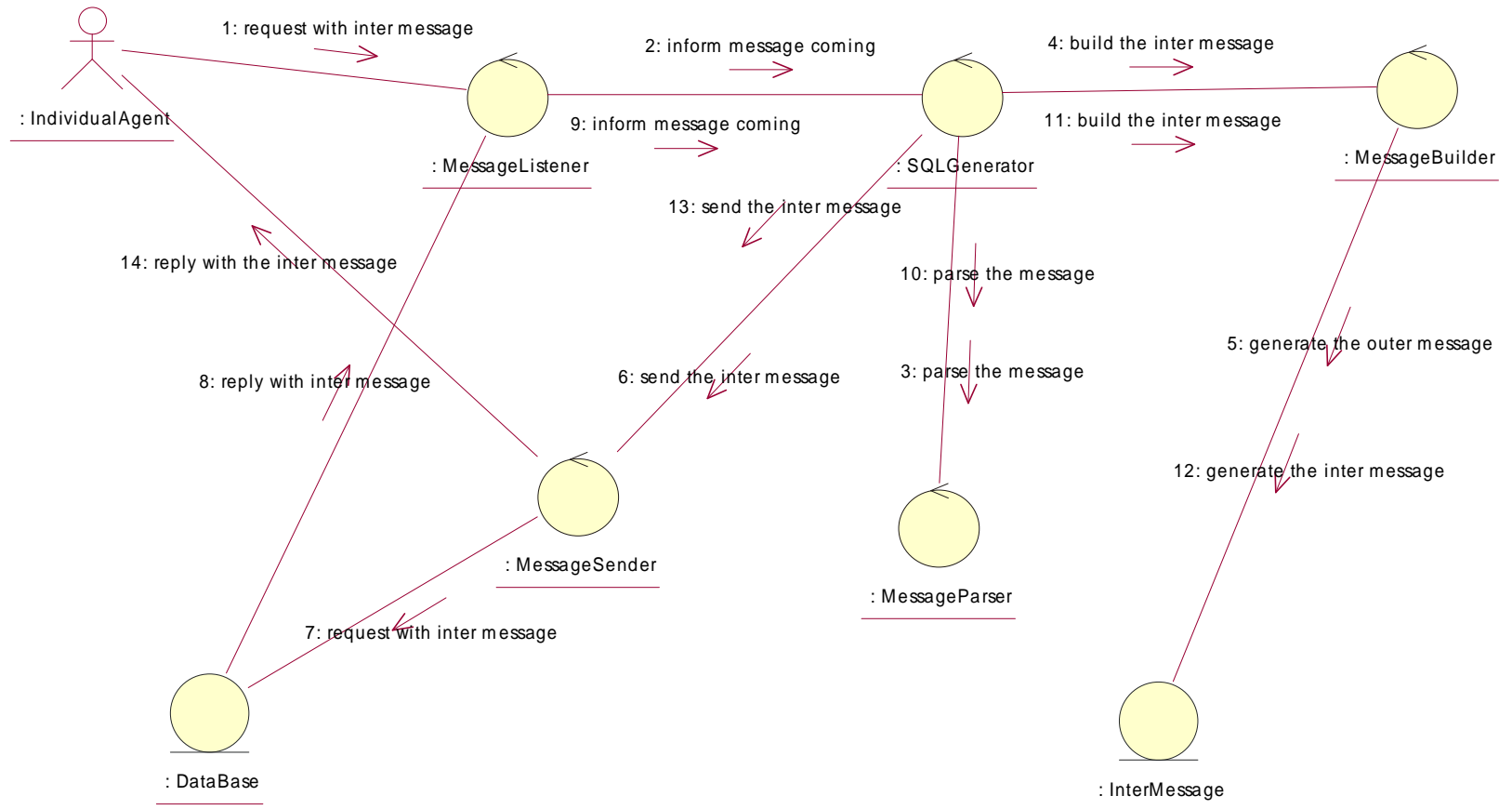


Figure 4-1 Collaboration diagram for use case Search internal database

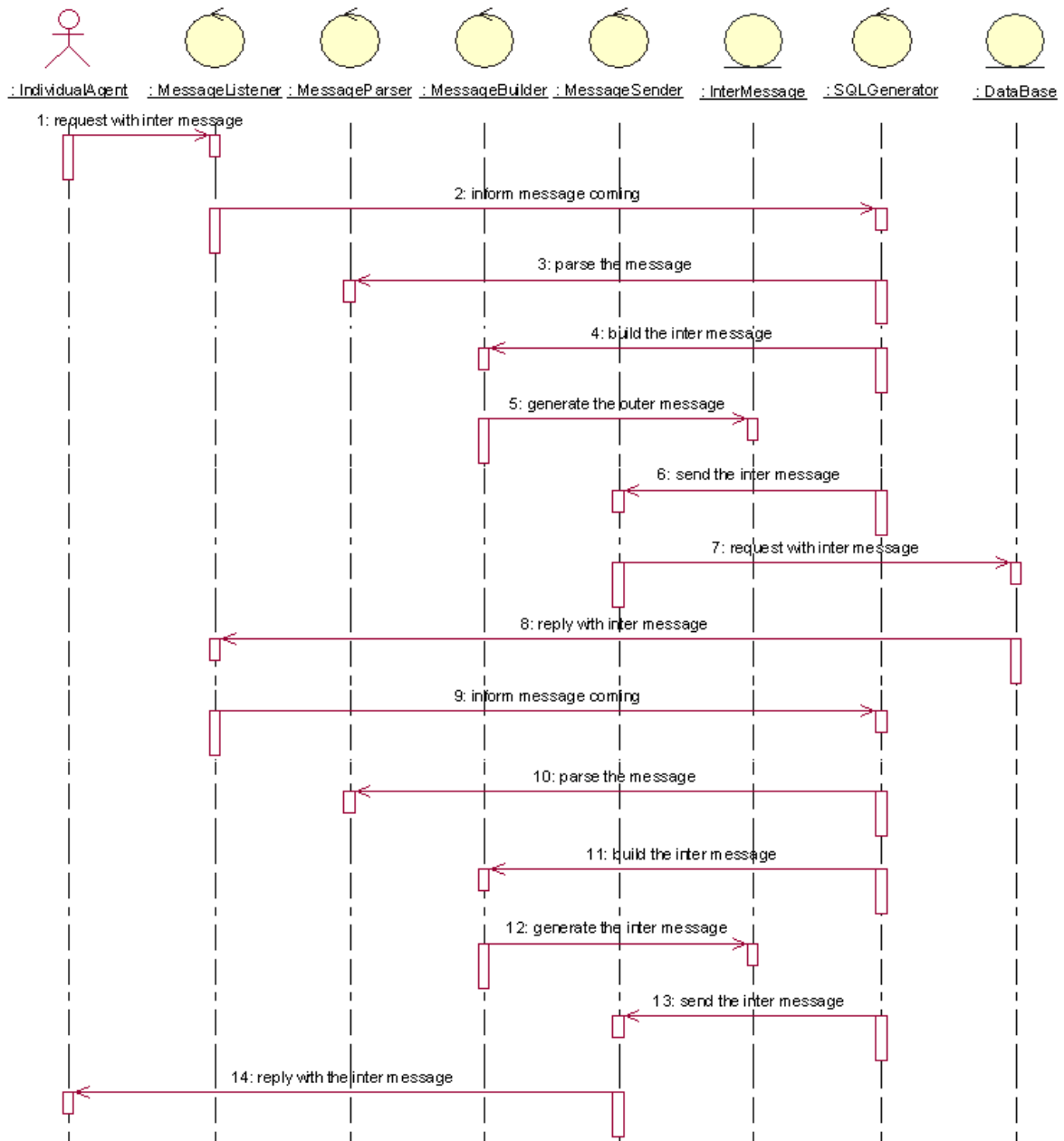


Figure 4-2 Sequence diagram for use case Search internal database

4.2 Search outer sources

Table 4-2 Use case search outer source

Use Case:	Search outer sources
Actor:	Individual
Short description:	The system searches the outer resources based on the user's request
Pre-Conditions:	The user has successfully logged in
Post-Conditions	The system searches the outer resources for users
Main flow:	<ul style="list-style-type: none"> • The user submits the search-job request • The system generates the search-outer profile • The system sends the search-outer profile to outer job resource • The system receives the search result from outer job resource • The system ends the search
Alternative flow:	<p>< A1 > If the system cannot get the search-outer result for a long time, then</p> <ol style="list-style-type: none"> 3. The system cancels this search-outer 4. The system sets the result to be "exception with network" <p>< A2 > If the user cancel the search, then The system stops the search</p>

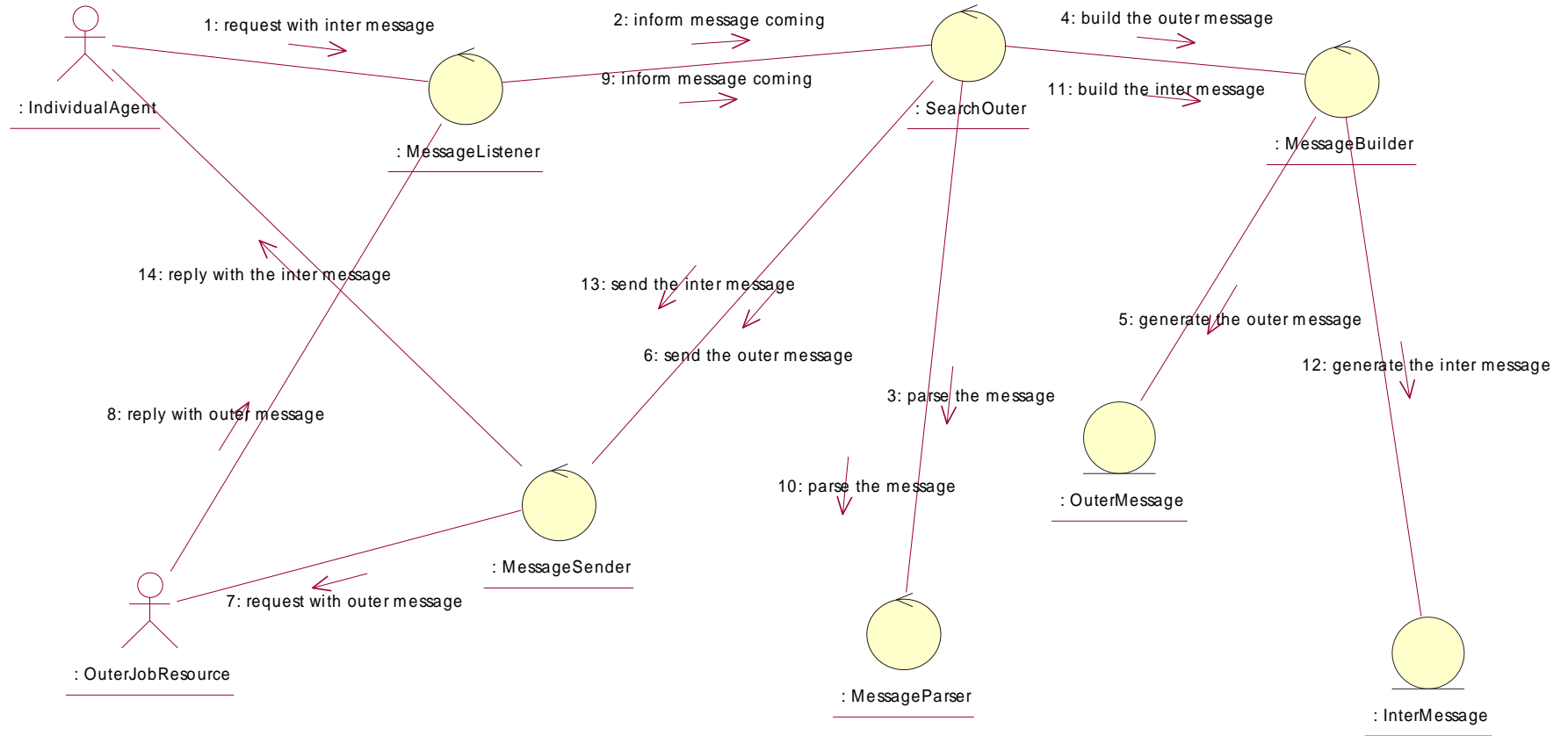


Figure 4-3 Collaboration diagram for use case Search outer sources

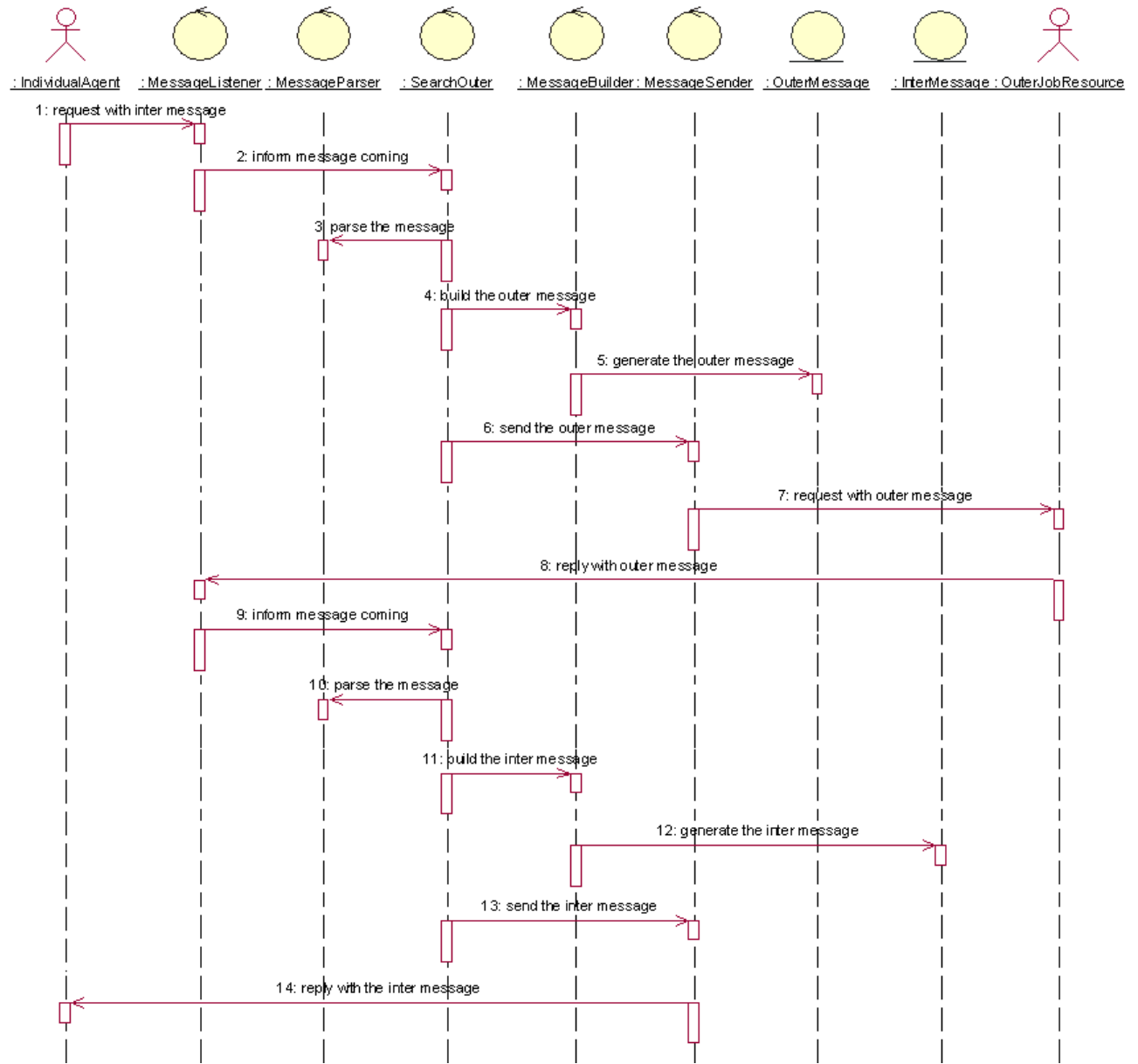


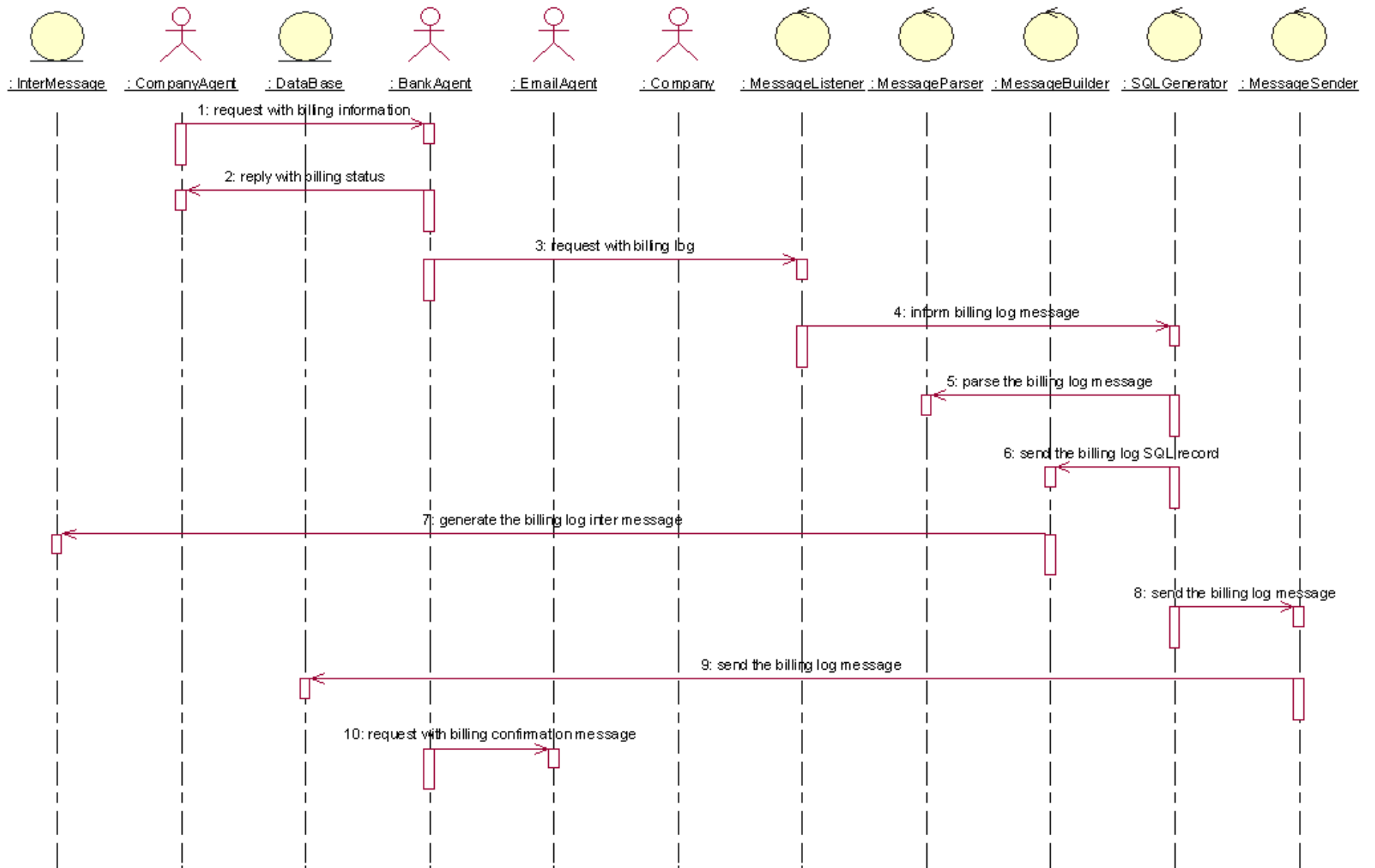
Figure 4-4 Collaboration diagram for use case Search outer sources

4.3 Post job

Table 4-3 Use case post job

Use Case:	Post job
Actor:	Company
Short description:	The Post Job use case is where the company posts a new job.
Pre-Conditions:	The company has successfully logged in
Post-Conditions:	A job post is created and added to the job-post database

<p>Main flow:</p>	<ul style="list-style-type: none"> •The system prompts the User for all the necessary information about the job •The company user fills all the required job information and then submits. (A1) •The system validates the entered information, making sure that all the required information is complete •The system prompts the User for billing information. The following information is included: •The company user fills all the billing information and then submits. (A3) •The system validates the billing information, making sure that all the billing information is correct •The system takes out money from company’s billing account and sends receipt to the company user by Email. (A4) •The system adds the new job post to the job-post database. The current date is recorded to be the creation data •The system adds the new job post to the company user profile. The current date is regarded as the creation data •The system displays a message informing the company that the job is successfully posted
<p>Alternative flow:</p>	<p><A1> If the User doesn’t enter complete information, the following occurs:</p> <ol style="list-style-type: none"> 4. The system describes where the information is incorrect or incomplete. 5. The system presents the User with suggestions for changes necessary to allow the User to make a successful post. 6. The system prompts the User to re-enter the incomplete or invalid information. <p><A2>If the company user doesn’t want to continue the posting, he or she can cancel the posting process at any time. Once this happens, the System will return to the company user home page</p> <p><A3>If the User doesn’t enter complete billing information, the following occurs:</p> <ol style="list-style-type: none"> 4. The system describes where the information is incorrect or incomplete. 5. The system presents the User with suggestions for changes necessary to allow the User to make a successful post 6. The system prompts the User to re-enter the incomplete or invalid billing information <p><A4>If the System cannot access the billing process, the following occurs:</p> <ol style="list-style-type: none"> 3. The system informs the company that the billing process cannot be accessed 4. The system prompts the User to re-enter the incomplete or invalid billing information



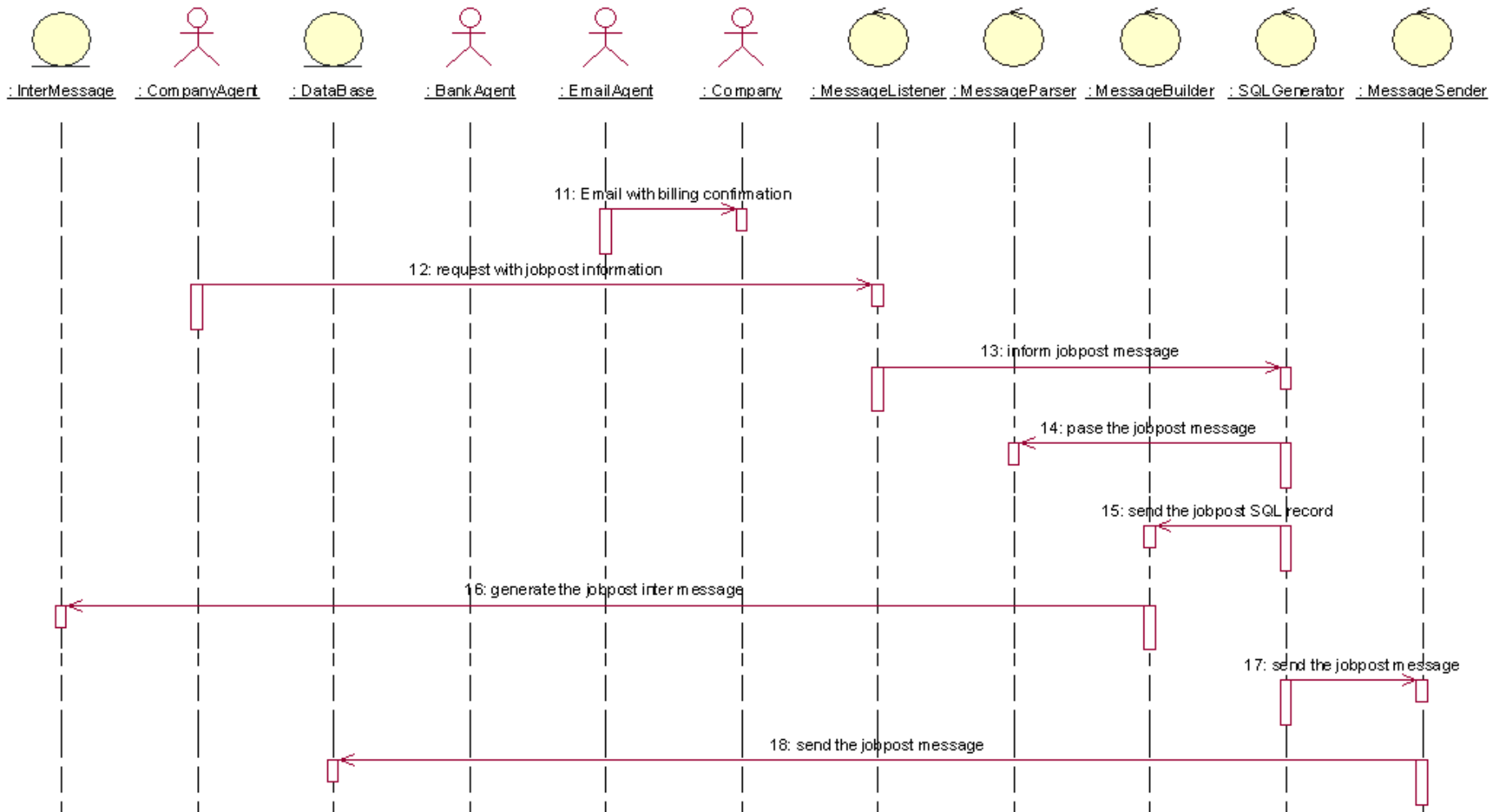


Figure 4-6 Sequence diagram for use case Post job

4.4 Search candidates

Table 4-4 Use case search candidates

Use Case:	Search candidates
Actor:	Company
Short description:	The search candidates use case is where the company user searches candidates
Pre-Conditions:	The user has successfully logged in
Post-Conditions:	The system presents the qualified candidates to the company user
Main flow:	<ul style="list-style-type: none"> • The system prompts the company user for all the search criteria about available candidates • The company user fills parts or all the search criteria and then submits • The system searches available candidates following company's request • The System lists all the summaries of matched candidates to company • The System prompts company to see the details of the most interested resumes. (A1) (A2) • The company user selects the most interested resumes and submits • The system prompts the User for billing information. The following information is included: <ul style="list-style-type: none"> • The company user fills all the billing information then submits. (A3) • The system validates the billing information, making sure that all the billing information is correct • The system takes out money from company's billing account and send receipt to the company user by Email. (A4) • The system sends all selected resume to company user's Email address • The system adds the search results to the Search log. The following information is included: <ul style="list-style-type: none"> • The system displays a message informing the company that the resumes are successfully sent
Alternative flow:	<p><A1>If the company user want to continue the search, he or she can redo the searching process at any time. Once this happens, the System will go to the search page</p> <p><A2>If the company user doesn't want to continue the search, he or she can cancel the searching process at any time. Once this happens, the System will return to the company user home page</p> <p><A3>If the User doesn't enter complete billing information, the following occurs:</p> <ol style="list-style-type: none"> 5. The system describes where information is incorrect/ incomplete 6. The system presents the User with suggestions for changes necessary to allow the User to make a successful post. 7. The system prompts the User to re-enter the incomplete or invalid billing information 8. The Basic Flow continues where the User enters new information <p><A4> If the System cannot access the billing process, following occurs:</p> <ol style="list-style-type: none"> 3. The system informs the company that the billing process cannot be accessed 4. The system prompts the User to re-enter the incomplete or invalid billing information

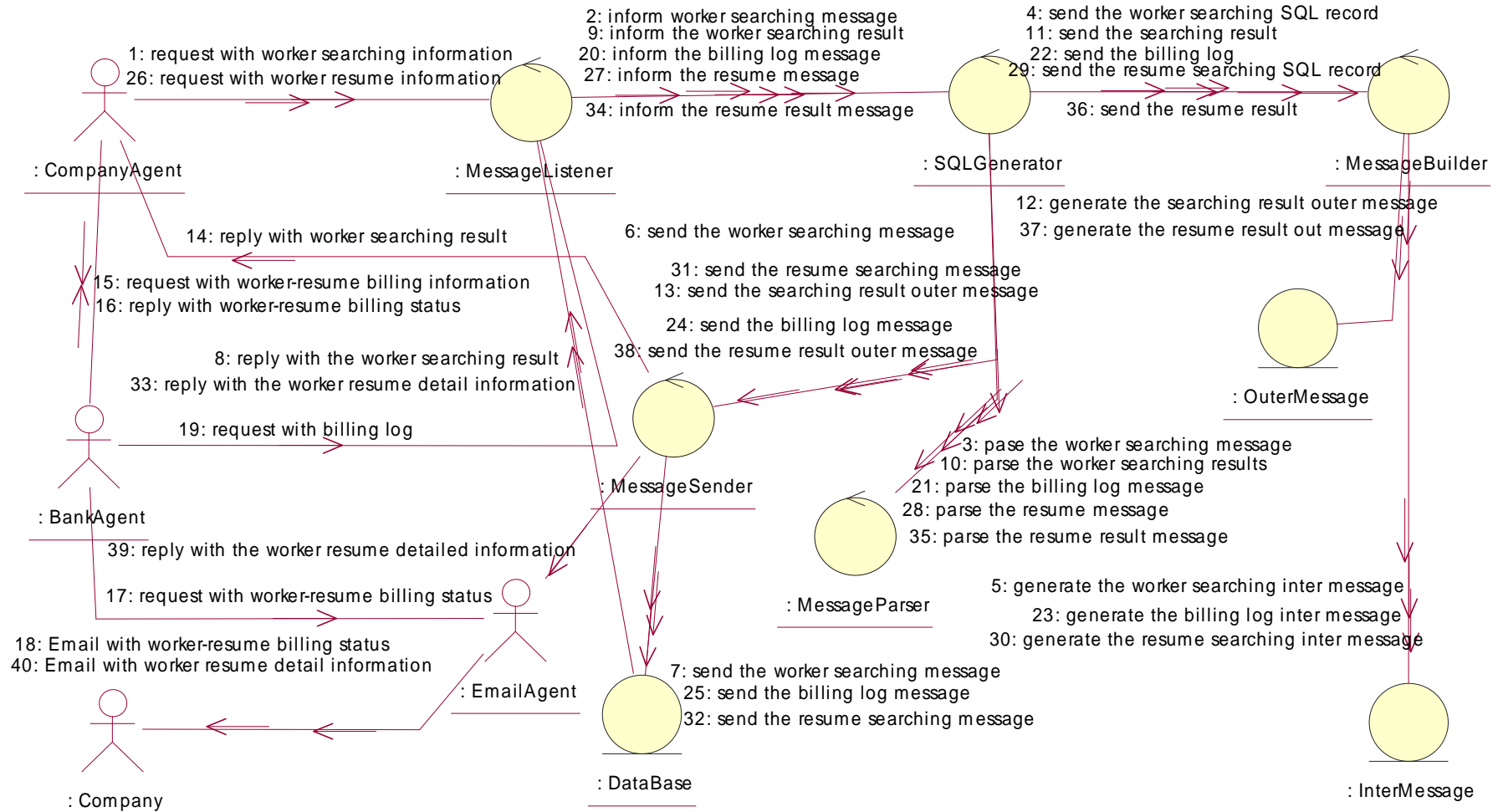
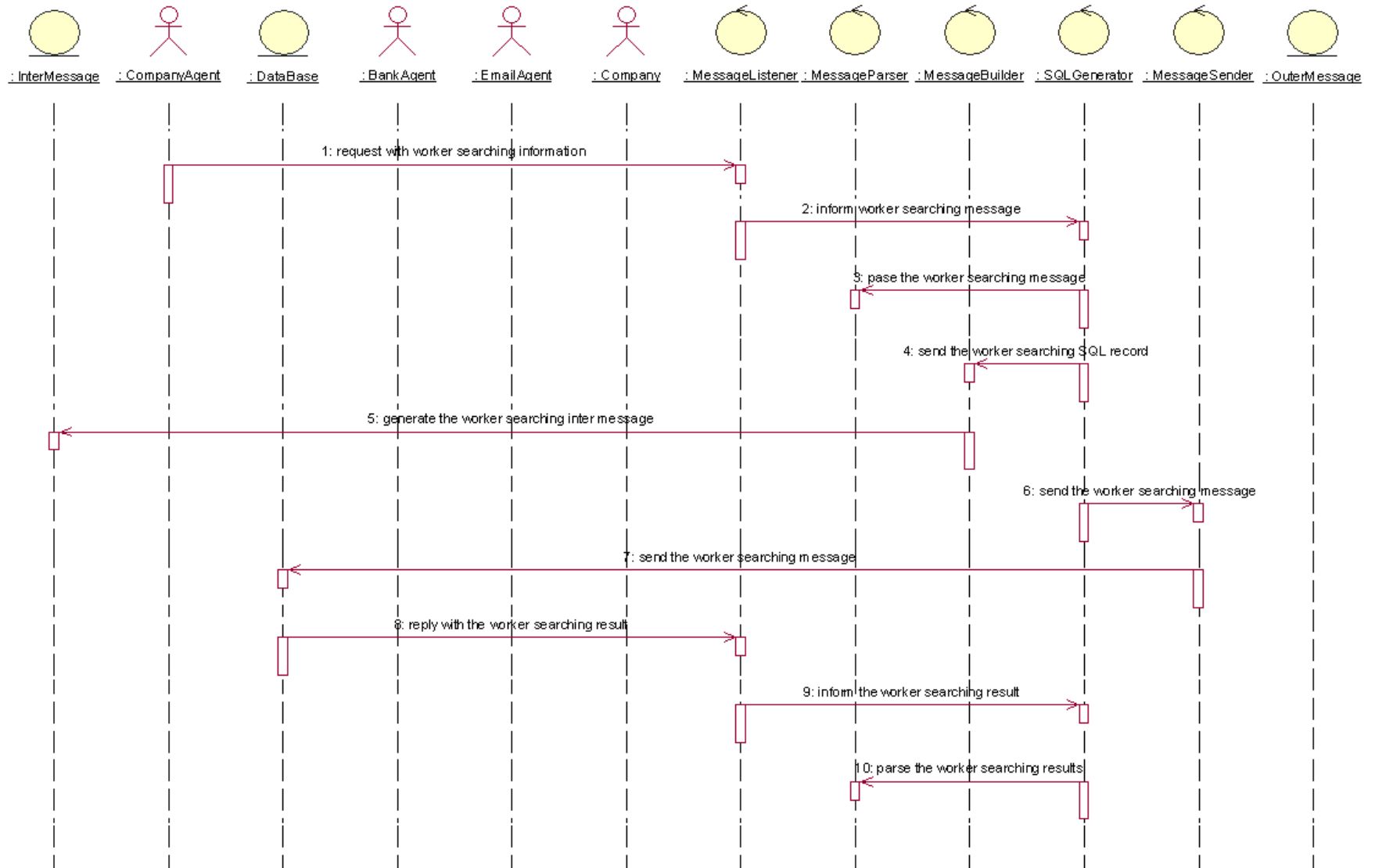
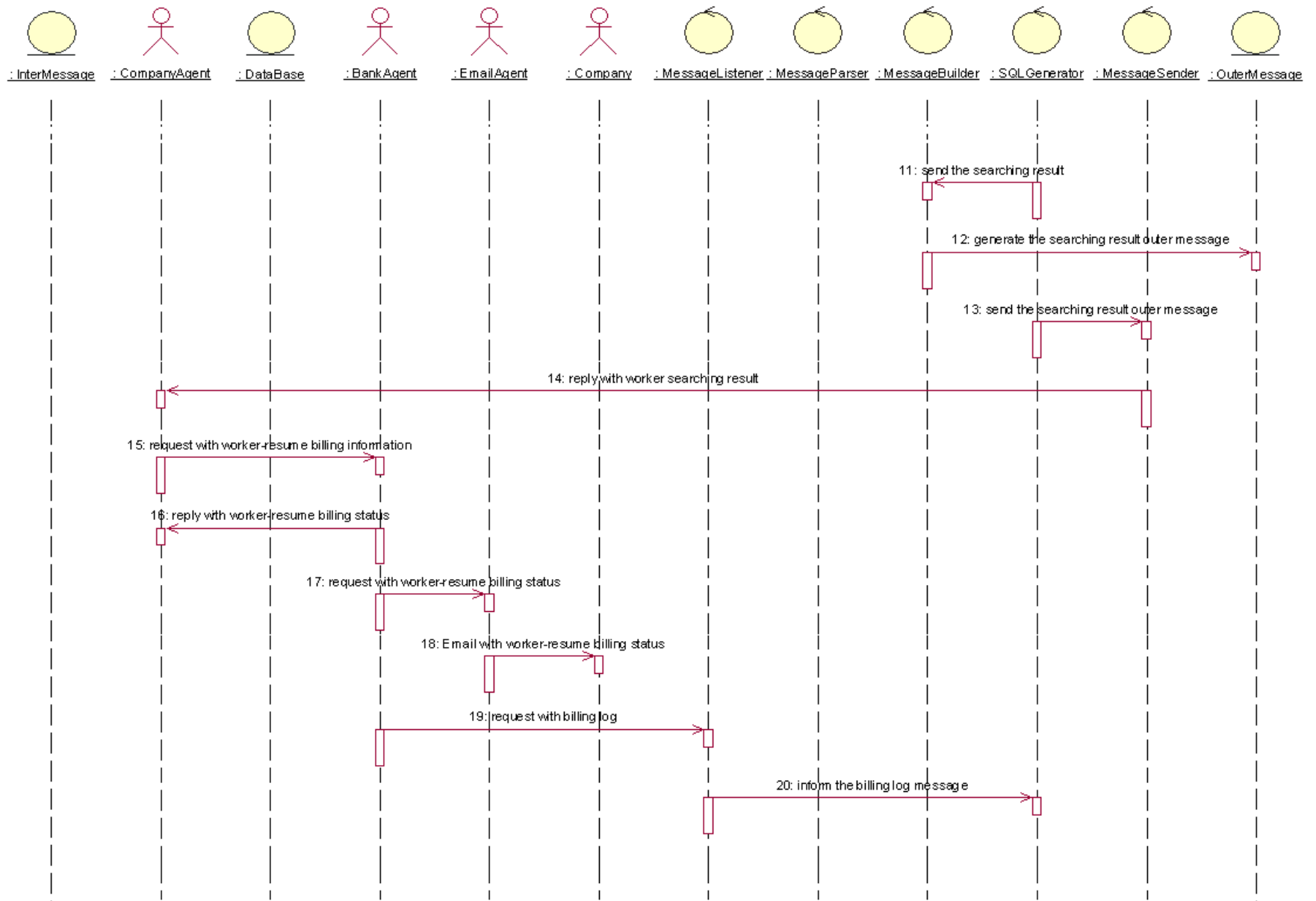
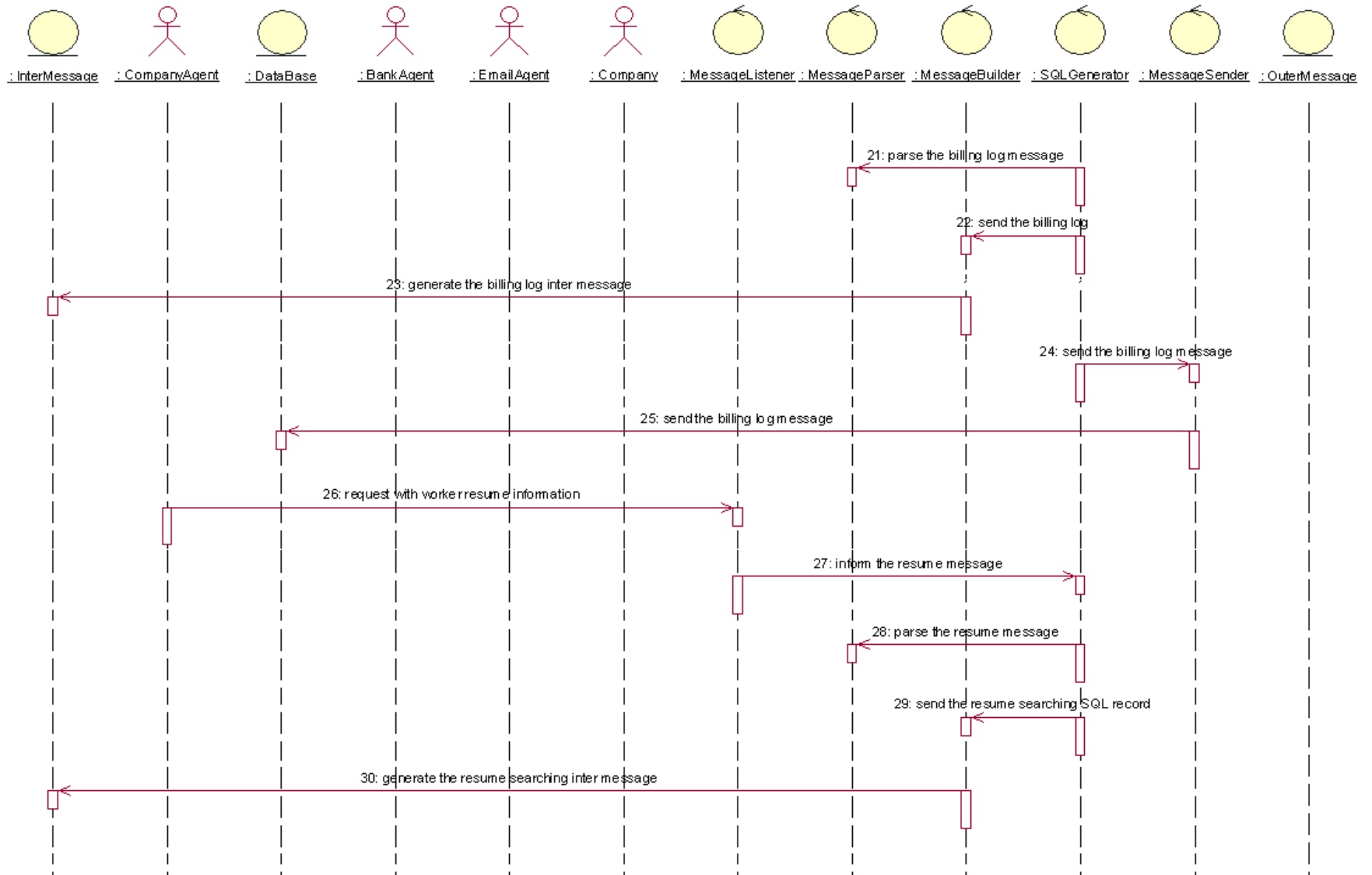


Figure 4-7 Collaboration diagram for use case Search candidates







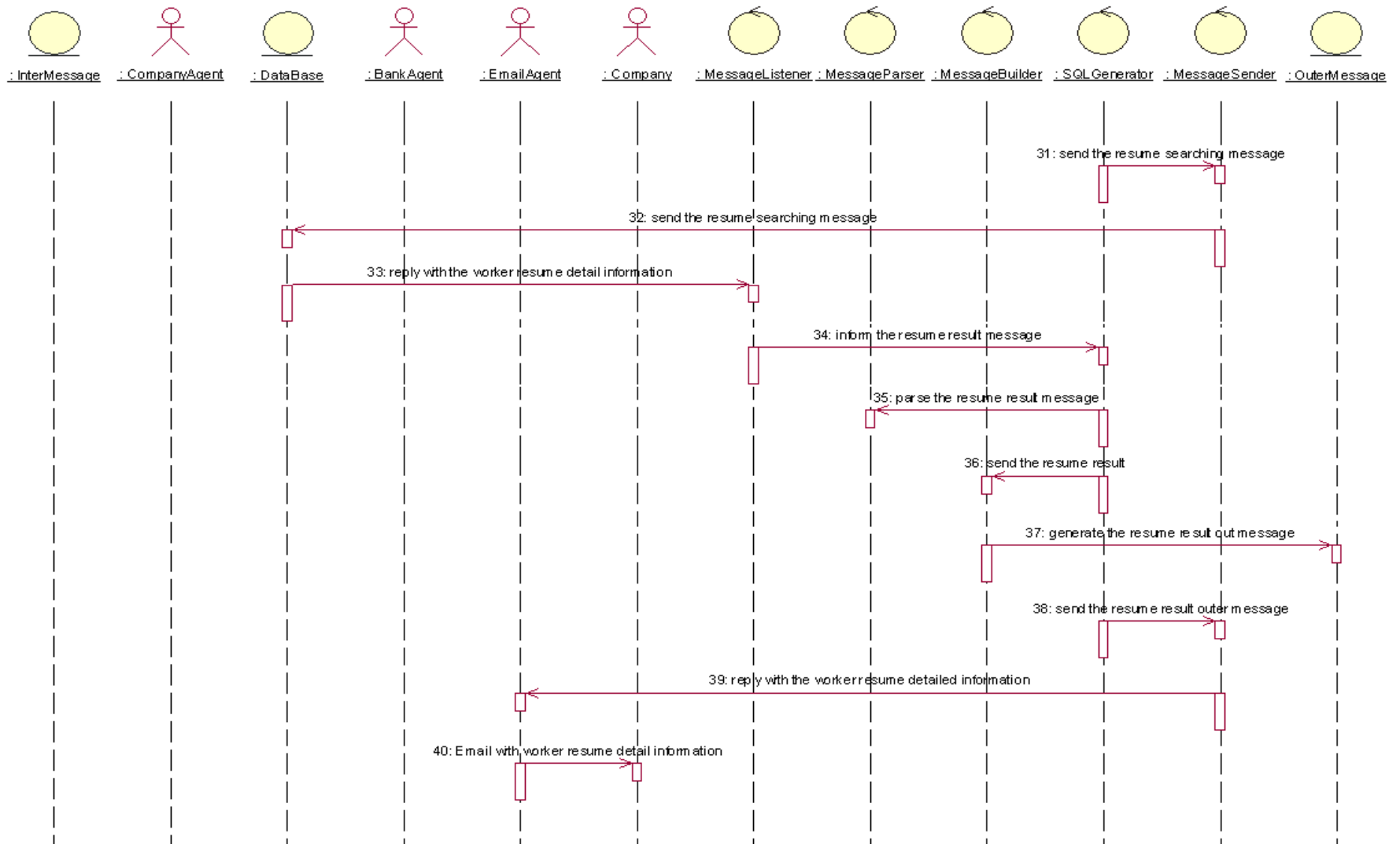


Figure 4-8 Sequence diagram for use case Search candidates

4.5 Class Diagram

Figure 4-9 shows the main six class diagrams of database agent and outer-search agent.

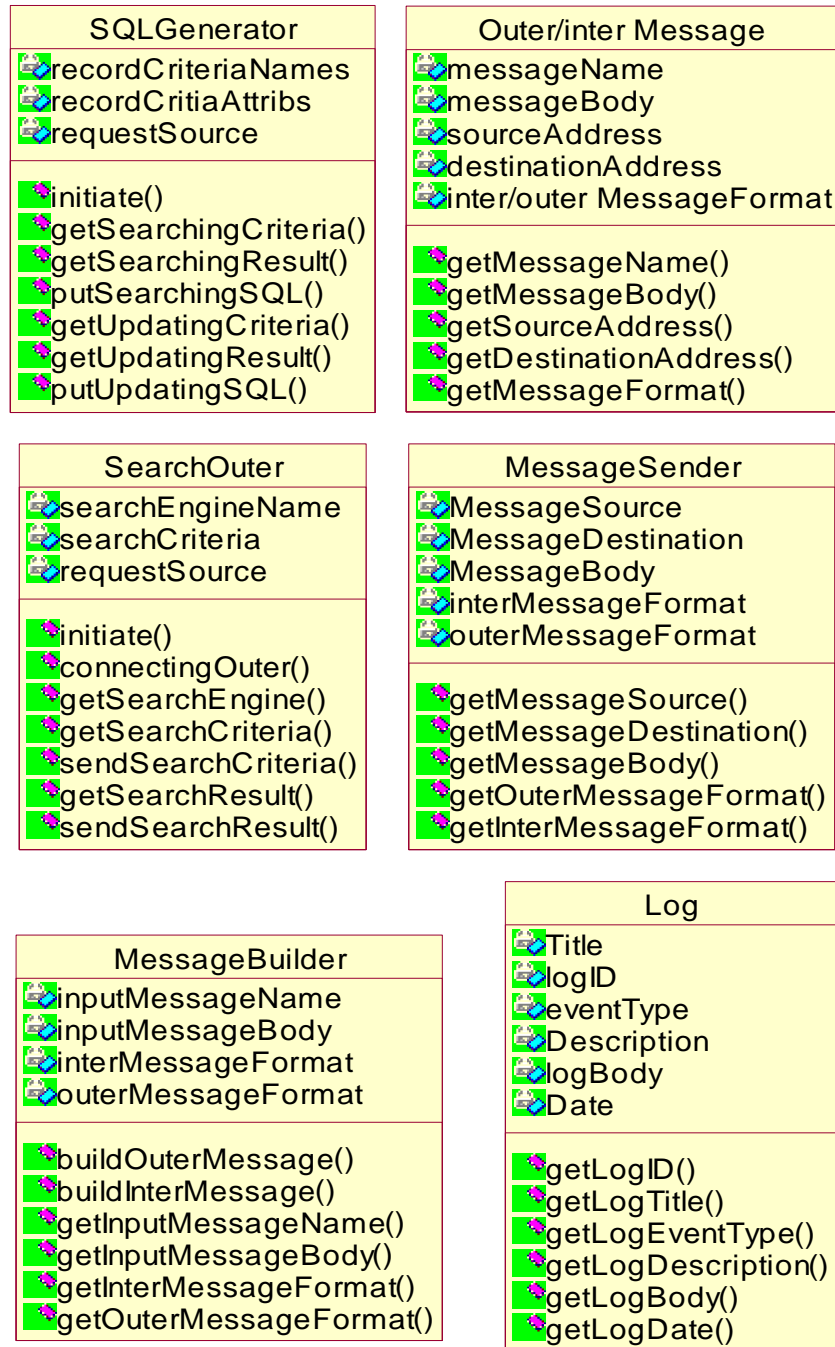


Figure 4-10 Six important class diagrams applied in four use cases

5. Data Specification

5.1 Typical Data Definition

Table 5-1 Individual profile definition

Individual		
Field	Data Type	Description
IndividualID	AutoNumber	Individual ID number
FirstName	Text	Individual first Name
LastName	Text	Individual last Name
AccountName	Text	Account Name
Password	Text	Password
Phone	Number	Phone number
Address	Text	Post address
Email	Text	Email address
Career	Text	Career Level
Degree	Text	Degree Level
ReceiveEmail	Yes/No	Weather or not receive Email information
Resume	OLE Object	Word or PDF document

Table 5-2 Company profile definition

Company		
Field	Data Type	Description
CompanyID	AutoNumber	Company ID number
Name	Text	Company Name
AccountName	Text	Account Name
Description	Memo	Brief description of the company
Password	Text	Password
Address	Text	Post address
Phone	Number	Phone number
Email	Text	Email address
Career	Text	Career Level
ReceiveEmail	Yes/No	Weather or not receive Email information

Table 5-3 Bill profile definition

Bill		
Field	Data Type	Description
BillID	AutoNumber	Bill id
CompanyID	Number	Company ID number
Date	Date/Time	The date of billing
Amount	Currency	The amount of the bill
CardType	Text	The type of the payment card
CardNumber	Number	The number of the card
ExpirationDate	Date/Time	The expiration of the card
HolderName	Text	The card holder's printed name
Finished	Yes/No	The transaction is finished or not

Table 5-4 FavoriteJob profile definition

FavoriteJob		
Field	Data Type	Description
JobID	AutoNumber	Favorite job ID number
IndividualID	Number	Individual ID number
Title	Text	Title
Description	Text	Description
Type	Text	Type
Salary	Text	Salary
Category	Text	Category
Location	Text	Location

Table 5-5 JobPost profile definition

JobPost		
Field	Data Type	Description
PostID	AutoNumber	Job post ID number
CompanyID	Number	Company ID Number
Title	Text	Title
Description	Text	Description
Type	Text	Type
Status	Text	Status
Salary	Text	Salary
Category	Text	Category
Location	Text	Location

5.2 ER Diagram

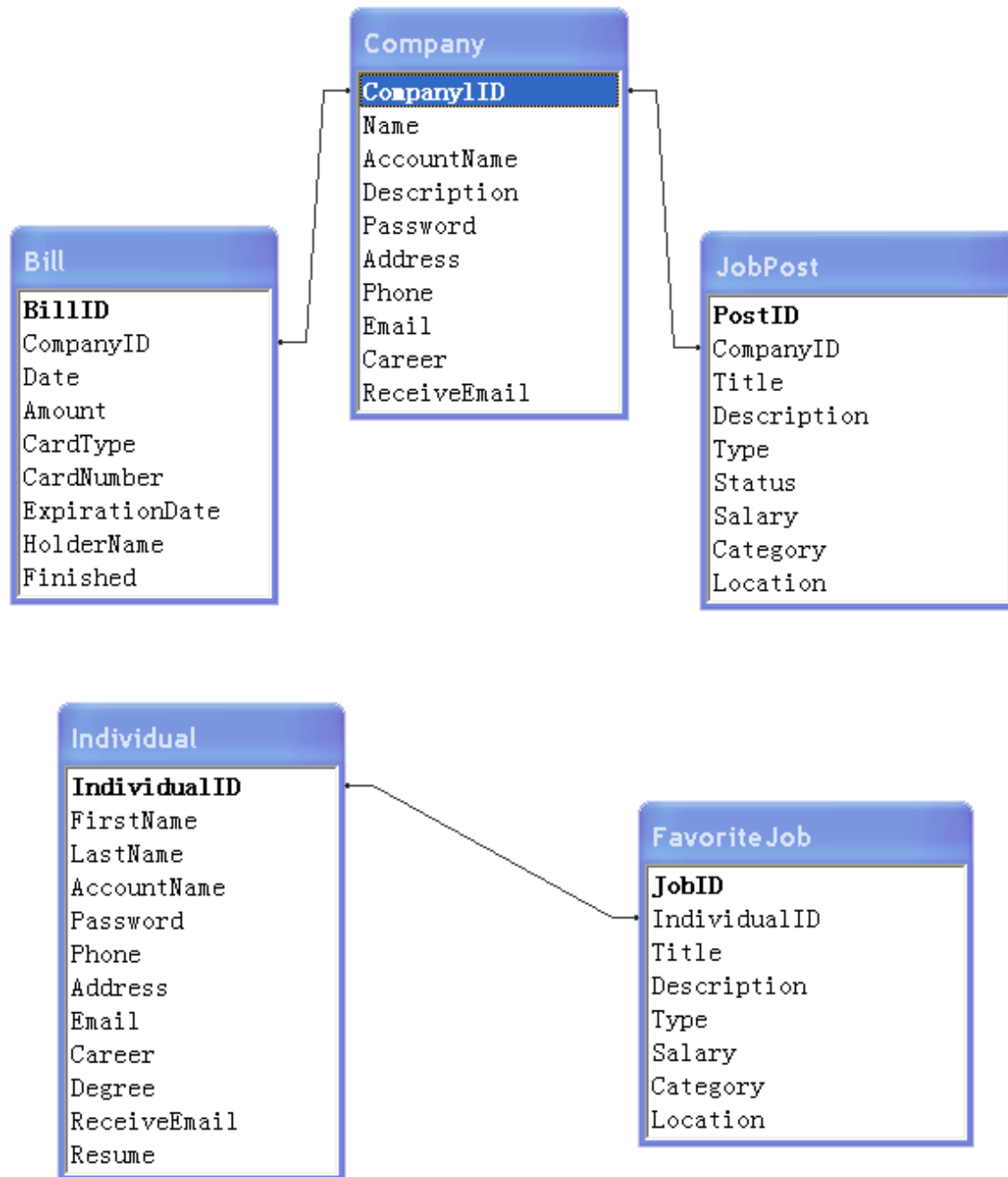


Figure 5-1 Relationships of typical tables

6. Conclusion

In this document, we have described the analysis and design of “An Agent-based System for Job Finding” using the methodology of agent-based systems. We have been able to achieve a lot of what we had set out to do:

- First of all, in the system specification process, we surveyed several famous job finding systems such as www.Monster.ca, www.Jobbank.ca, www.job-hunter.com, www.manpower.ca and so on. From these famous and successful commercial systems we understood the business goal of our system;
- Based on the business case of the system, we settled the requirements of our system. We group discussed together and roughly established most of the necessary roles and functionalities in this system;
- Afterward, we wrote down all the 10 necessary use cases. These use cases were detailed and thus roles and functionalities are finally determined.
- Then we went to the analysis and the high level design process. At this time we divided the system into seven agents: individual agent, company agent, bank agent, outer-search agent, database agent, email agent and analysis agent.
- Till now, the rough system architecture diagram was generated. Consequently, we entered the low-level and detail design process.
- Based on the rough system structure, the agent collaboration and message specification were created. And the detailed system architecture was written by means of the Unified Modeling Language.
- In the data specification, the knowledge query and manipulation language formats are defined for this particular system.
- Finally, we implemented the most important four use cases and generated the complicated collaboration diagrams with help of UML tools. These four use cases are: search internal database, search outer source, post job, search candidates. Based on the collaboration diagrams, sequence diagrams were also generated.

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

7. References

1. OMG Agent Platform Special Interest Group: Agent Technology Green Paper. Object Management Group (2000)
Available: http://www.objs.com/agent/agents_Green_Paper_v100.doc
2. Wooldridge, M., Jennings, N. R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*. 3(3) (2000)
3. Brazier, F., Dunin-Keplicz, B., Jennings, N. R., Treur, J.: DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems*. 6(1) (1997) 67-94
4. Kendall, E.A.: Role modeling for agent system analysis, design, and implementation. *Concurrency, IEEE*. 8 (2) (2000) 34-41
5. Riehle, D., Gross, T.: Role Model Based Framework Design and Integration. *Proc. 1998 Conf. Object-Oriented Programming Systems, Languages and Applications*. ACM Press, New York, (1998)
6. Jacobson, I.: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley. (1992)
7. Gregor Hohpe, Bobby Woolf : *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Pearson Education; (2003)
8. Eric Newcomer: *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley Pub Co, (2002)

8. Acknowledgement

We would like to express our gratefulness to Dr. Behrouz H. Far for the course SENG609.22 and the systematic knowledge about the agent-based software engineering it provided. We also appreciate the opportunity to apply all the Knowledge we learned in this project.

Appendix

A.1 Message 1 Document

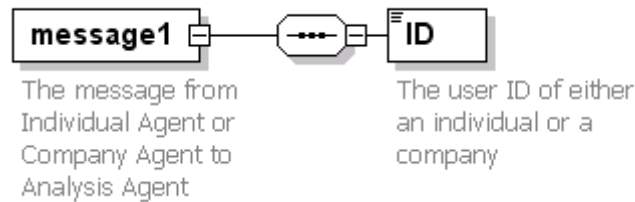
Schema **Message1.xsd**

Elements

[message1](#)

element **message1**

diagram



children [ID](#)

annotation documentation The message from Individual Agent or Company Agent to Analysis Agent

```

source <xs:element name="message1">
  <xs:annotation>
    <xs:documentation>The message from Individual Agent or Company Agent to Analysis Agent </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The user ID of either an individual or a company
        </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

element **message1/ID**

diagram



type **xs:string**

annotation documentation The user ID of either an individual or a company

source `<xs:element name="ID" type="xs:string">`
`<xs:annotation>`
`<xs:documentation>The user ID of either an individual or a`
`company</xs:documentation>`
`</xs:annotation>`
`</xs:element>`

A.2 Message 2 Document

Schema **Message2.xsd**

Elements

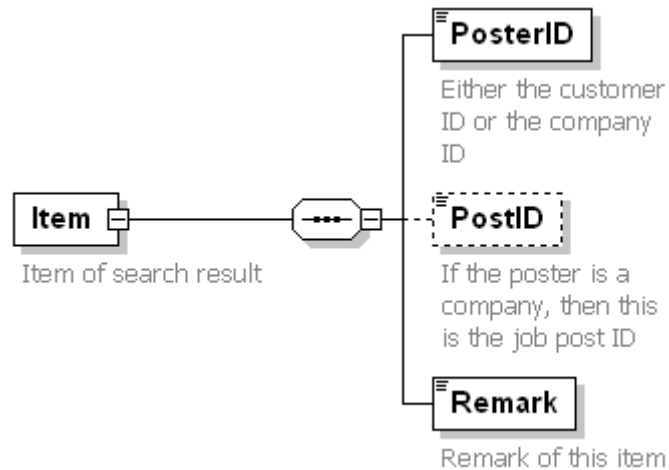
[Item](#)
[Message2](#)

Simple types

[MessageType](#)

element **Item**

diagram



children [PosterID](#) [PostID](#) [Remark](#)

used by element [Message2/Content](#)

annotation documentation Item of search result

```

source <xs:element name="Item">
  <xs:annotation>
    <xs:documentation>Item of search result </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PosterID" type="xs:string">
        <xs:annotation>
          <xs:documentation>Either the customer ID or the company ID </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="PostID" type="xs:ID" minOccurs="0">
        <xs:annotation>
          <xs:documentation>If the poster is a company, then this is the job post ID
        </xs:documentation>
      </xs:element>
      <xs:element name="Remark" type="xs:string">

```

```

    <xs:annotation>
      <xs:documentation>Remark of this item </xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

element Item/PosterID

diagram



Either the customer
ID or the company
ID

type **xs:string**

annotation documentation Either the customer ID or the company ID

```

source <xs:element name="PosterID" type="xs:string">
  <xs:annotation>
    <xs:documentation>Either the customer ID or the company ID </xs:documentation>
  </xs:annotation>
</xs:element>

```

element Item/PostID

diagram



If the poster is a
company, then this
is the job post ID

type **xs:ID**

annotation documentation If the poster is a company, then this is the job post ID

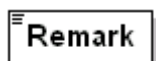
```

source <xs:element name="PostID" type="xs:ID" minOccurs="0">
  <xs:annotation>
    <xs:documentation>If the poster is a company, then this is the job post ID
  </xs:documentation>
  </xs:annotation>
</xs:element>

```

element Item/Remark

diagram

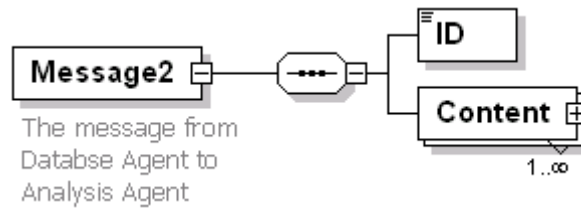


Remark of this item

type **xs:string**
 annotation documentation Remark of this item
 source `<xs:element name="Remark" type="xs:string">
 <xs:annotation>
 <xs:documentation>Remark of this item </xs:documentation>
 </xs:annotation>
 </xs:element>`

element **Message2**

diagram



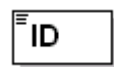
children [ID Content](#)

attributes	Name	Type	Use	Default	Fixed
	Type	MessageType	required		
annotation	documentation	The message from Databse Agent to Analysis Agent			

source `<xs:element name="Message2">
 <xs:annotation>
 <xs:documentation>The message from Databse Agent to Analysis Agent
 </xs:documentation>
 </xs:annotation>
 <xs:complexType>
 <xs:sequence>
 <xs:element name="ID" type="xs:string"/>
 <xs:element name="Content" maxOccurs="unbounded">
 <xs:complexType>
 <xs:sequence>
 <xs:element ref="Item"/>
 </xs:sequence>
 </xs:complexType>
 </xs:element>
 </xs:sequence>
 <xs:attribute name="Type" type="MessageType" use="required"/>
 </xs:complexType>
 </xs:element>`

element **Message2/ID**

diagram

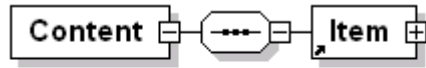


type **xs:string**

source `<xs:element name="ID" type="xs:string"/>`

element **Message2/Content**

diagram



Item of search result

children [Item](#)

source `<xs:element name="Content" maxOccurs="unbounded">
 <xs:complexType>
 <xs:sequence>
 <xs:element ref="Item"/>
 </xs:sequence>
 </xs:complexType>
 </xs:element>`

simpleType **MessageType**

type restriction of **xs:string**

used by attribute [Message2/@Type](#)

facets enumeration JobApplicant
 enumeration JobSupplier

source `<xs:simpleType name="MessageType" final="restriction">
 <xs:restriction base="xs:string">
 <xs:enumeration value="JobApplicant"/>
 <xs:enumeration value="JobSupplier"/>
 </xs:restriction>
 </xs:simpleType>`

A.3 Message 3 Document

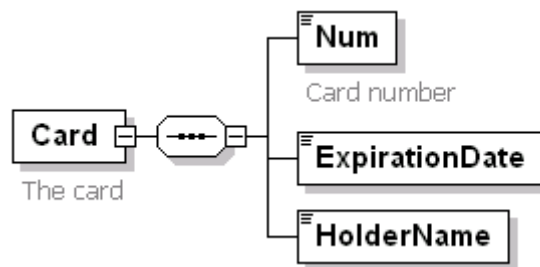
Schema **Message3.xsd**

Elements
[Card](#)
[Item](#)
[Message3](#)

Simple types
[CardType](#)
[ItemType](#)

element **Card**

diagram



children [Num](#) [ExpirationDate](#) [HolderName](#)

used by element [Message3](#)

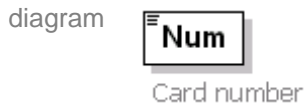
attributes	Name	Type	Use	Default	Fixed
annotation	Type	CardType	required		
	documentation	The card			

source

```

<xs:element name="Card">
  <xs:annotation>
    <xs:documentation>The card</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Num" type="xs:long">
        <xs:annotation>
          <xs:documentation>Card number</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ExpirationDate" type="xs:date"/>
      <xs:element name="HolderName" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="Type" type="CardType" use="required"/>
  </xs:complexType>
</xs:element>
  
```

element **Card/Num**

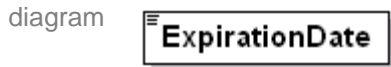


type **xs:long**

annotation documentation Card number

```
source <xs:element name="Num" type="xs:long">
  <xs:annotation>
    <xs:documentation>Card number</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Card/ExpirationDate**



type **xs:date**

```
source <xs:element name="ExpirationDate" type="xs:date"/>
```

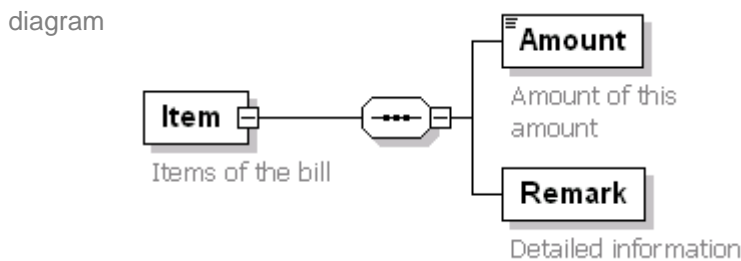
element **Card/HolderName**



type **xs:string**

```
source <xs:element name="HolderName" type="xs:string"/>
```

element **Item**



children [Amount](#) [Remark](#)

used by element [Message3](#)

attributes	Name	Type	Use	Default	Fixed
	Type	ItemType	required		
annotation	documentation	Items of the bill			

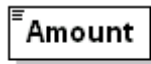
```

source <xs:element name="Item">
  <xs:annotation>
    <xs:documentation>Items of the bill</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Amount" type="xs:decimal">
        <xs:annotation>
          <xs:documentation>Amount of this amount</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Remark">
        <xs:annotation>
          <xs:documentation>Detailed information </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="Type" type="ItemType" use="required"/>
  </xs:complexType>
</xs:element>

```

element Item/Amount

diagram



Amount of this amount

type **xs:decimal**

annotation documentation Amount of this amount

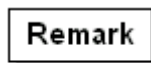
```

source <xs:element name="Amount" type="xs:decimal">
  <xs:annotation>
    <xs:documentation>Amount of this amount</xs:documentation>
  </xs:annotation>
</xs:element>

```

element Item/Remark

diagram



Detailed information

annotation documentation Detailed information

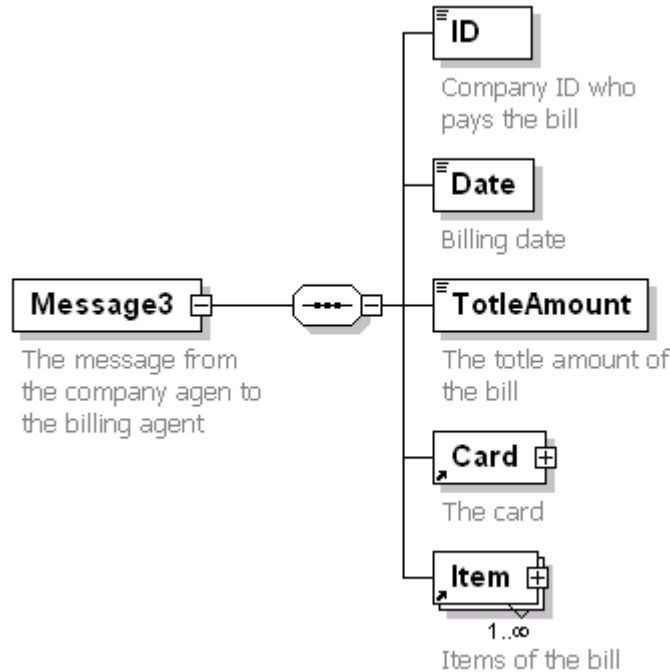
```

source <xs:element name="Remark">
  <xs:annotation>
    <xs:documentation>Detailed information </xs:documentation>
  </xs:annotation>
</xs:element>

```

element **Message3**

diagram



children [ID](#) [Date](#) [TotleAmount](#) [Card](#) [Item](#)

annotation documentation The message from the company agen to the billing agent

```

source <xs:element name="Message3">
  <xs:annotation>
    <xs:documentation>The message from the company agen to the billing agent
  </xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:sequence>
    <xs:element name="ID" type="xs:string">
      <xs:annotation>
        <xs:documentation>Company ID who pays the bill </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Date" type="xs:date">
      <xs:annotation>
        <xs:documentation>Billing date </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="TotleAmount" type="xs:decimal">
      <xs:annotation>
        <xs:documentation>The totle amount of the bill </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element ref="Card">
      <xs:annotation>

```

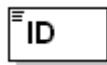
```

    <xs:documentation>The card the company uses to pay </xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element ref="Item" maxOccurs="unbounded">
  <xs:annotation>
    <xs:documentation>The detailed items of the bill </xs:documentation>
  </xs:annotation>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

```

element Message3/ID

diagram



Company ID who
pays the bill

type **xs:string**

annotation documentation Company ID who pays the bill

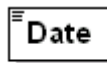
```

source <xs:element name="ID" type="xs:string">
  <xs:annotation>
    <xs:documentation>Company ID who pays the bill </xs:documentation>
  </xs:annotation>
</xs:element>

```

element Message3/Date

diagram



Billing date

type **xs:date**

annotation documentation Billing date

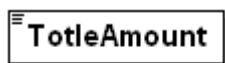
```

source <xs:element name="Date" type="xs:date">
  <xs:annotation>
    <xs:documentation>Billing date </xs:documentation>
  </xs:annotation>
</xs:element>

```

element Message3/TotleAmount

diagram



The totle amount of
the bill

type **xs:decimal**
 annotation documentation The totle amount of the bill
 source `<xs:element name="TotleAmount" type="xs:decimal">
 <xs:annotation>
 <xs:documentation>The totle amount of the bill </xs:documentation>
 </xs:annotation>
</xs:element>`

simpleType **CardType**

type restriction of **xs:string**
 used by attribute [Card/@Type](#)
 facets enumeration Visa
 enumeration Master
 annotation documentation Card type either Visa or Master
 source `<xs:simpleType name="CardType">
 <xs:annotation>
 <xs:documentation>Card type either Visa or Master</xs:documentation>
 </xs:annotation>
 <xs:restriction base="xs:string">
 <xs:enumeration value="Visa"/>
 <xs:enumeration value="Master"/>
 </xs:restriction>
</xs:simpleType>`

simpleType **ItemType**

type restriction of **xs:string**
 used by attribute [Item/@Type](#)
 facets enumeration Resume
 enumeration Post
 annotation documentation The bill item type either Post or Resume
 source `<xs:simpleType name="ItemType">
 <xs:annotation>
 <xs:documentation>The bill item type either Post or Resume</xs:documentation>
 </xs:annotation>
 <xs:restriction base="xs:string">
 <xs:enumeration value="Resume"/>
 <xs:enumeration value="Post"/>
 </xs:restriction>
</xs:simpleType>`

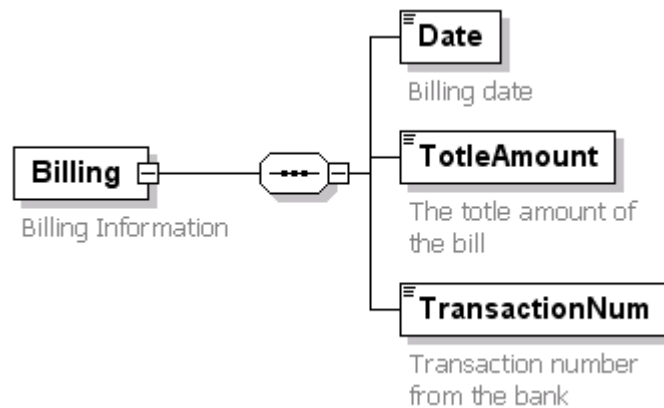
A.4 Message 4 Document

Schema **Message4.xsd**

Elements
[Billing](#)
[Message4](#)
[Search](#)
[Update](#)

Simple types
[MessageType](#)

element **Billing**
 diagram



children [Date](#) [TotleAmount](#) [TransactionNum](#)

used by element [Message4](#)

annotation documentation Billing Information

```

source <xs:element name="Billing">
  <xs:annotation>
    <xs:documentation>Billing Information</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Date" type="xs:date">
        <xs:annotation>
          <xs:documentation>Billing date </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="TotleAmount" type="xs:decimal">
        <xs:annotation>
          <xs:documentation>The totle amount of the bill </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="TransactionNum" type="xs:long">
        <xs:annotation>

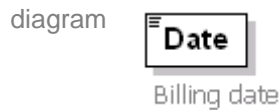
```

```

        <xs:documentation>Transaction number from the bank </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

```

element **Billing/Date**



type **xs:date**

annotation documentation Billing date

```

source <xs:element name="Date" type="xs:date">
  <xs:annotation>
    <xs:documentation>Billing date </xs:documentation>
  </xs:annotation>
</xs:element>

```

element **Billing/TotleAmount**



type **xs:decimal**

annotation documentation The totle amount of the bill

```

source <xs:element name="TotleAmount" type="xs:decimal">
  <xs:annotation>
    <xs:documentation>The totle amount of the bill </xs:documentation>
  </xs:annotation>
</xs:element>

```

element **Billing/TransactionNum**



type **xs:long**

annotation documentation Transaction number from the bank

```

source <xs:element name="TransactionNum" type="xs:long">

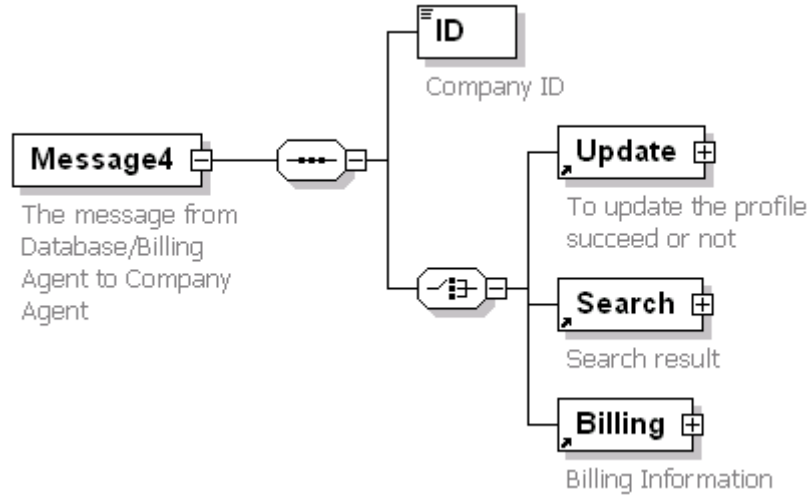
```

```

<xs:annotation>
  <xs:documentation>Transaction number from the bank </xs:documentation>
</xs:annotation>
</xs:element>
    
```

element **Message4**

diagram



children [ID](#) [Update](#) [Search](#) [Billing](#)

attributes	Name	Type	Use	Default	Fixed
annotation	Type	MessageType	required		
annotation	documentation	The message from Database/Billing Agent to Company Agent			

```

source <xs:element name="Message4">
  <xs:annotation>
    <xs:documentation>The message from Database/Billing Agent to Company Agent</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:string">
        <xs:annotation>
          <xs:documentation>Company ID</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:choice>
        <xs:element ref="Update"/>
        <xs:element ref="Search"/>
        <xs:element ref="Billing"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="Type" type="MessageType" use="required"/>
  </xs:complexType>
</xs:element>
    
```

element **Message4/ID**

type **xs:string**

annotation documentation Company ID

```
source <xs:element name="ID" type="xs:string">
  <xs:annotation>
    <xs:documentation>Company ID</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Search**

children [Item](#)

used by element [Message4](#)

annotation documentation Search result

```
source <xs:element name="Search">
  <xs:annotation>
    <xs:documentation>Search result</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Item" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

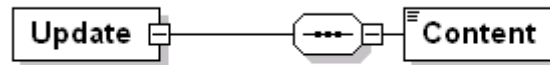
element **Search/Item**

type **xs:string**

```
source <xs:element name="Item" type="xs:string" maxOccurs="unbounded"/>
```

element **Update**

diagram



To update the profile
succeed or not

children [Content](#)used by element [Message4](#)

annotation documentation To update the profile succeed or not

```

source <xs:element name="Update">
  <xs:annotation>
    <xs:documentation>To update the profile succeed or not</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Content" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

element **Update/Content**

diagram

type **xs:string**source `<xs:element name="Content" type="xs:string"/>`simpleType **MessageType**type restriction of **xs:string**used by attribute [Message4/@Type](#)

facets enumeration Update
enumeration Search
enumeration Billing

annotation documentation Message type Update result, Search result or Billing information

```

source <xs:simpleType name="MessageType">
  <xs:annotation>
    <xs:documentation>Message type Update result, Search result or Billing
information</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="Update"/>
    <xs:enumeration value="Search"/>
  </xs:restriction>
</xs:simpleType>

```

```
<xs:enumeration value="Billing"/>  
</xs:restriction>  
</xs:simpleType>
```

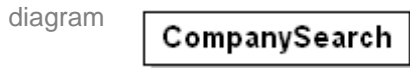
A.5 Message 5 Document

Schema **Message5.xsd**

Elements
[CompanySearch](#)
[CompanyUpdate](#)
[IndividualSearch](#)
[IndividualUpdate](#)
[Message5](#)

Simple types
[MessageType](#)

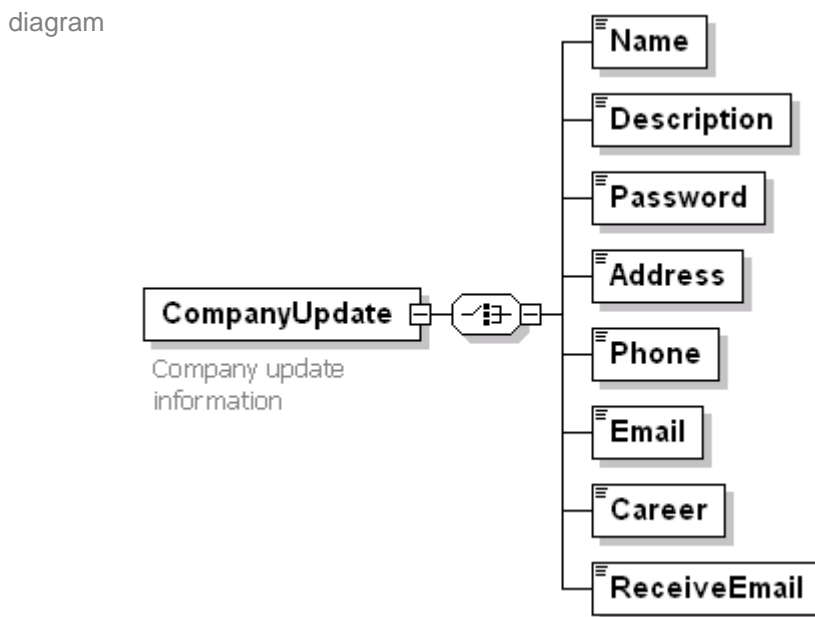
element **CompanySearch**



used by element [Message5](#)

source `<xs:element name="CompanySearch"/>`

element **CompanyUpdate**



children [Name](#) [Description](#) [Password](#) [Address](#) [Phone](#) [Email](#) [Career](#) [ReceiveEmail](#)

used by element [Message5](#)

annotation documentation Company update information

source `<xs:element name="CompanyUpdate">
 <xs:annotation>
 <xs:documentation>Company update information</xs:documentation>`

```
</xs:annotation>
<xs:complexType>
  <xs:choice>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Description" type="xs:string"/>
    <xs:element name="Password" type="xs:string"/>
    <xs:element name="Address" type="xs:string"/>
    <xs:element name="Phone" type="xs:long"/>
    <xs:element name="Email" type="xs:string"/>
    <xs:element name="Career" type="xs:string"/>
    <xs:element name="ReceiveEmail" type="xs:boolean"/>
  </xs:choice>
</xs:complexType>
</xs:element>
```

element CompanyUpdate/Name

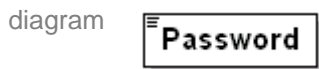
type **xs:string**

source `<xs:element name="Name" type="xs:string"/>`

element CompanyUpdate/Description

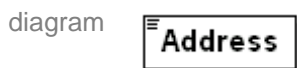
type **xs:string**

source `<xs:element name="Description" type="xs:string"/>`

element CompanyUpdate/Password

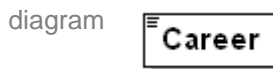
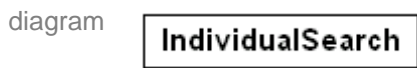
type **xs:string**

source `<xs:element name="Password" type="xs:string"/>`

element CompanyUpdate/Address

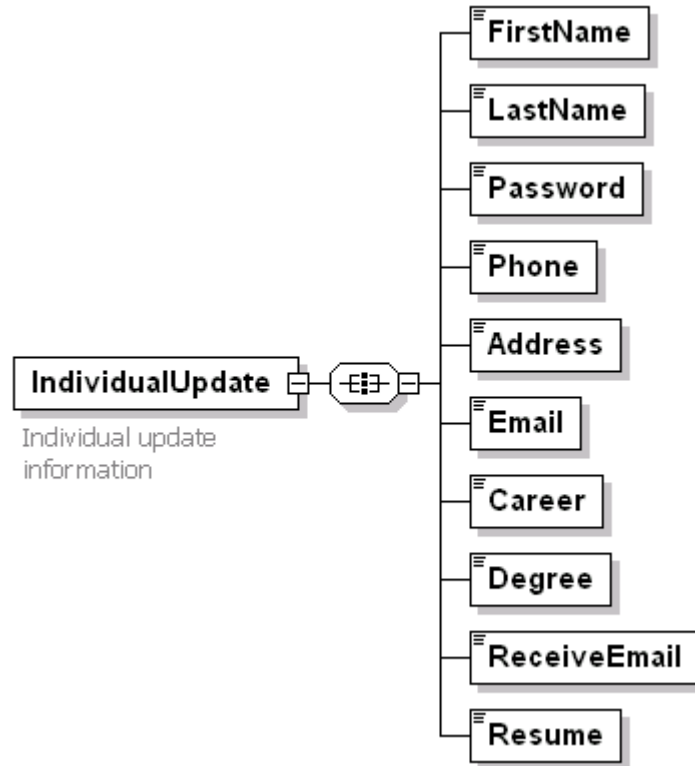
type **xs:string**

source `<xs:element name="Address" type="xs:string"/>`

element **CompanyUpdate/Phone**type **xs:long**source `<xs:element name="Phone" type="xs:long"/>`element **CompanyUpdate/Email**type **xs:string**source `<xs:element name="Email" type="xs:string"/>`element **CompanyUpdate/Career**type **xs:string**source `<xs:element name="Career" type="xs:string"/>`element **CompanyUpdate/ReceiveEmail**type **xs:boolean**source `<xs:element name="ReceiveEmail" type="xs:boolean"/>`element **IndividualSearch**used by element [Message5](#)source `<xs:element name="IndividualSearch"/>`

element **IndividualUpdate**

diagram



children [FirstName](#) [LastName](#) [Password](#) [Phone](#) [Address](#) [Email](#) [Career](#) [Degree](#) [ReceiveEmail](#) [Resume](#)

used by element [Message5](#)

annotation documentation Individual update information

```

source <xs:element name="IndividualUpdate">
  <xs:annotation>
    <xs:documentation>Individual update information </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:all>
      <xs:element name="FirstName" type="xs:string"/>
      <xs:element name="LastName" type="xs:string"/>
      <xs:element name="Password" type="xs:string"/>
      <xs:element name="Phone" type="xs:long"/>
      <xs:element name="Address" type="xs:string"/>
      <xs:element name="Email" type="xs:string"/>
      <xs:element name="Career" type="xs:string"/>
      <xs:element name="Degree" type="xs:string"/>
      <xs:element name="ReceiveEmail" type="xs:boolean"/>
      <xs:element name="Resume" type="xs:string"/>
    </xs:all>
  </xs:complexType>

```

```
</xs:element>
```

element IndividualUpdate/FirstName

diagram 

type **xs:string**

source

```
<xs:element name="FirstName" type="xs:string"/>
```

element IndividualUpdate/LastName

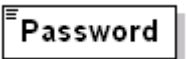
diagram 

type **xs:string**

source

```
<xs:element name="LastName" type="xs:string"/>
```

element IndividualUpdate/Password

diagram 

type **xs:string**

source

```
<xs:element name="Password" type="xs:string"/>
```

element IndividualUpdate/Phone

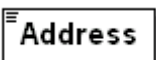
diagram 

type **xs:long**

source

```
<xs:element name="Phone" type="xs:long"/>
```

element IndividualUpdate/Address

diagram 

type **xs:string**

source

```
<xs:element name="Address" type="xs:string"/>
```

element IndividualUpdate/Email

diagram 

type **xs:string**
source `<xs:element name="Email" type="xs:string"/>`

element **IndividualUpdate/Career**



type **xs:string**
source `<xs:element name="Career" type="xs:string"/>`

element **IndividualUpdate/Degree**



type **xs:string**
source `<xs:element name="Degree" type="xs:string"/>`

element **IndividualUpdate/ReceiveEmail**



type **xs:boolean**
source `<xs:element name="ReceiveEmail" type="xs:boolean"/>`

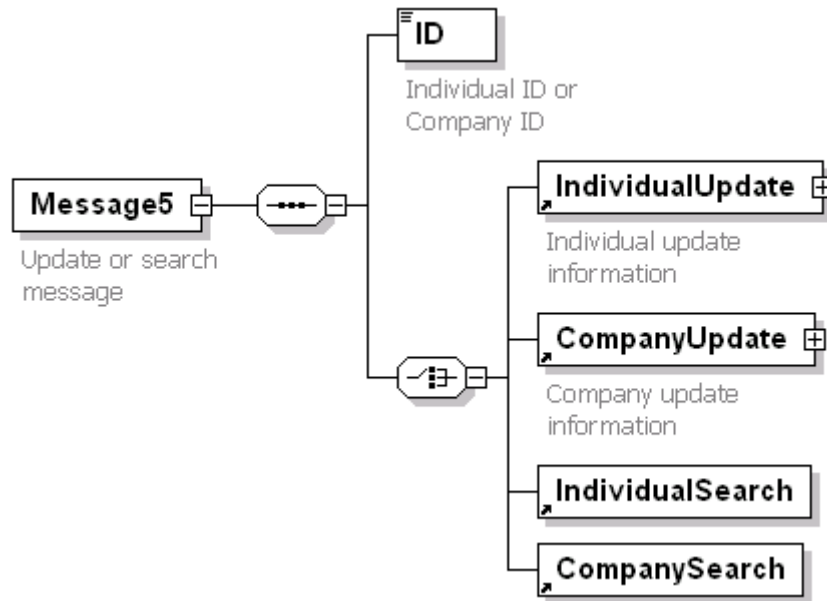
element **IndividualUpdate/Resume**



type **xs:string**
source `<xs:element name="Resume" type="xs:string"/>`

element **Message5**

diagram



children [ID](#) [IndividualUpdate](#) [CompanyUpdate](#) [IndividualSearch](#) [CompanySearch](#)

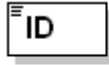
attributes	Name	Type	Use	Default	Fixed
annotation	Type	MessageType	required		
	documentation	Update or search message			

```

source <xs:element name="Message5">
  <xs:annotation>
    <xs:documentation>Update or search message</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:string">
        <xs:annotation>
          <xs:documentation>Individual ID or Company ID </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:choice>
        <xs:element ref="IndividualUpdate"/>
        <xs:element ref="CompanyUpdate"/>
        <xs:element ref="IndividualSearch"/>
        <xs:element ref="CompanySearch"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="Type" type="MessageType" use="required"/>
  </xs:complexType>
</xs:element>
  
```

element **Message5/ID**

diagram

Individual ID or
Company IDtype **xs:string**

annotation documentation Individual ID or Company ID

```
source <xs:element name="ID" type="xs:string">
  <xs:annotation>
    <xs:documentation>Individual ID or Company ID </xs:documentation>
  </xs:annotation>
</xs:element>
```

simpleType **MessageType**type restriction of **xs:string**used by attribute [Message5/@Type](#)facets enumeration Update
enumeration Search

```
source <xs:simpleType name="MessageType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Update"/>
    <xs:enumeration value="Search"/>
  </xs:restriction>
</xs:simpleType>
```

A.6 Message 6 Document

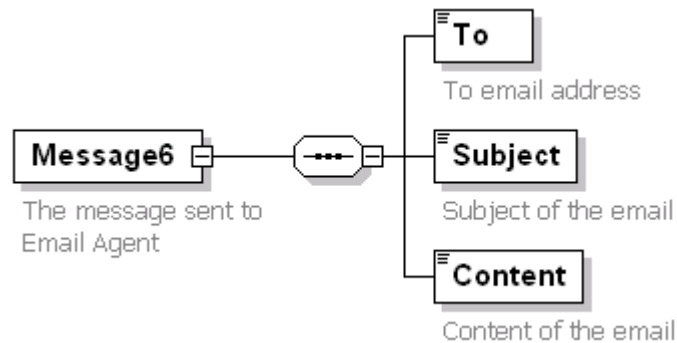
Schema **Message6.xsd**

Elements

[Message6](#)

element **Message6**

diagram



children [To Subject Content](#)

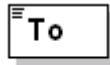
annotation documentation The message sent to Email Agent

```

source <xs:element name="Message6">
  <xs:annotation>
    <xs:documentation>The message sent to Email Agent</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="To" type="xs:string">
        <xs:annotation>
          <xs:documentation>To email address </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Subject" type="xs:string">
        <xs:annotation>
          <xs:documentation>Subject of the email </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Content" type="xs:string">
        <xs:annotation>
          <xs:documentation>Content of the email </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
  
```

element **Message6/To**

diagram



To email address

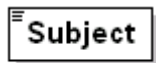
type **xs:string**

annotation documentation To email address

```
source <xs:element name="To" type="xs:string">
  <xs:annotation>
    <xs:documentation>To email address </xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Message6/Subject**

diagram



Subject of the email

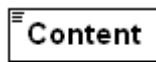
type **xs:string**

annotation documentation Subject of the email

```
source <xs:element name="Subject" type="xs:string">
  <xs:annotation>
    <xs:documentation>Subject of the email </xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Message6/Content**

diagram



Content of the email

type **xs:string**

annotation documentation Content of the email

```
source <xs:element name="Content" type="xs:string">
  <xs:annotation>
    <xs:documentation>Content of the email </xs:documentation>
  </xs:annotation>
</xs:element>
```

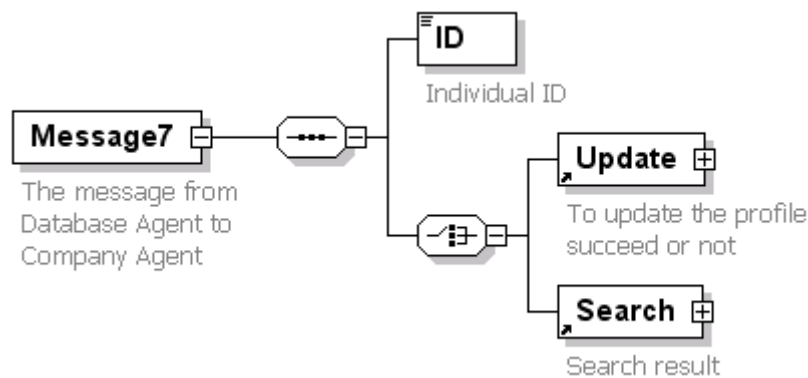
A.7 Message 7 Document

Schema **Message7.xsd**

Elements
[Message7](#)
[Search](#)
[Update](#)

Simple types
[MessageType](#)

element **Message7**
 diagram



children	ID	Update	Search			
attributes	Name	Type	Use	Default	Fixed	
annotation	Type	MessageType	required			
	documentation	The message from Database Agent to Company Agent				

```

source <xs:element name="Message7">
  <xs:annotation>
    <xs:documentation>The message from Database Agent to Company Agent</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:string">
        <xs:annotation>
          <xs:documentation>Individual ID</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:choice>
        <xs:element ref="Update"/>
        <xs:element ref="Search"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="Type" type="MessageType" use="required"/>
  </xs:complexType>
  
```

```
</xs:element>
```

element **Message7/ID**



Individual ID

type **xs:string**

annotation documentation Individual ID

```
source <xs:element name="ID" type="xs:string">
  <xs:annotation>
    <xs:documentation>Individual ID</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Search**



Search result

1..∞

children [Item](#)

used by element [Message7](#)

annotation documentation Search result

```
source <xs:element name="Search">
  <xs:annotation>
    <xs:documentation>Search result</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Item" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **Search/Item**

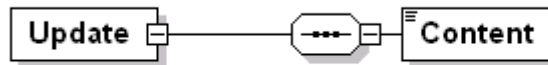


type **xs:string**

```
source <xs:element name="Item" type="xs:string" maxOccurs="unbounded"/>
```

element **Update**

diagram



To update the profile
succeed or not

children **Content**used by element **Message7**

annotation documentation To update the profile succeed or not

```
source <xs:element name="Update">
  <xs:annotation>
    <xs:documentation>To update the profile succeed or not</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Content" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **Update/Content**

diagram

type **xs:string**

source <xs:element name="Content" type="xs:string"/>

simpleType **MessageType**type restriction of **xs:string**used by attribute **Message7/@Type**

facets enumeration Update
enumeration Search

annotation documentation Message type Update result, Search result or Billing information

```
source <xs:simpleType name="MessageType">
  <xs:annotation>
    <xs:documentation>Message type Update result, Search result or Billing
information</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="Update"/>
    <xs:enumeration value="Search"/>
  </xs:restriction>
```

</xs:simpleType>

A.8 Message 8 Document

Schema **Message8.xsd**

Elements

[Condition](#)
[Message8](#)

element **Condition**

diagram



used by

element [Message8](#)

annotation

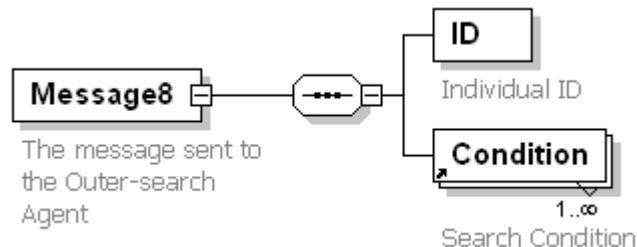
documentation Search Condition

source

```
<xs:element name="Condition">
  <xs:annotation>
    <xs:documentation>Search Condition</xs:documentation>
  </xs:annotation>
</xs:element>
```

element **Message8**

diagram



children

[ID](#) [Condition](#)

annotation

documentation The message sent to the Outer-search Agent

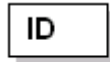
source

```
<xs:element name="Message8">
  <xs:annotation>
    <xs:documentation>The message sent to the Outer-search Agent </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID">
        <xs:annotation>
          <xs:documentation>Individual ID</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element ref="Condition" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

element Message8/ID

diagram



Individual ID

annotation documentation Individual ID

```
source <xs:element name="ID">
  <xs:annotation>
    <xs:documentation>Individual ID</xs:documentation>
  </xs:annotation>
</xs:element>
```