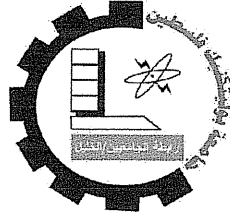


٤٠٠

Palestine Polytechnic University



College of Engineering & Technology
Electrical & Computer Engineering Department

Graduation Project

Plagiarism Detection System Using Mobile Agents (PLADMA)

Project Team

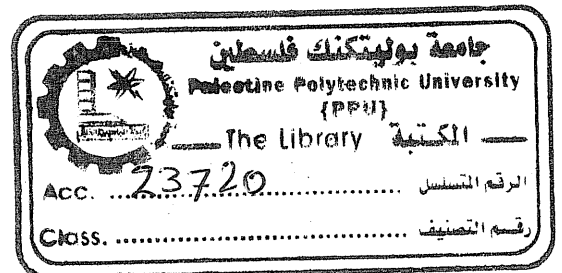
Eid Badr

Abdullah Abdeen

Project Supervisor
Dr. Radwan Tahboub

Hebron-Palestine

June, 2008



جامعة بوليتكنك فلسطين

الخليل- فلسطين

كلية الهندسة والتكنولوجيا

دائرة الهندسة الكهربائية

اسم المشروع

Plagiarism Detection System Using Mobile Agents (PLADMA)

اسماء الطلبة

عبد الله عابدين عيّد بدر

بناء على نظام كلية الهندسة والتكنولوجيا وإشراف ومتابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة حاسوب.

توقيع المشرف

.....

10.7.2008

توقيع اللجنة الممتحنة

.....

توقيع رئيس الدائرة

.....

Abstract

Plagiarism or copyright infringement of protected text on the Internet is a widespread and growing problem. The detection of this kind of plagiarism or infringement by human is time consuming. In this project a survey on different approaches that aimed to solve this problem is to be proposed.

Mobile Agent (MA) is executing program that migrates during execution and present methods for maintaining and using distributed systems [14].

Integrating one of the approaches used in plagiarism or copyright infringement of protected text with mobile agents can be considered as a data mining problem. In this novel approach a design and implementation of a system that makes use of mobile agents that search the Internet for text data and compare it to submitted text data that may contain plagiarized text. This approach will increase the efficiency of the detection process, and reduce both time and cost that are presented in other approaches. Fault tolerance, and security techniques that enhance the mobile agent's navigation mission are applied in this project.

The project aims to obtain high degree of accuracy in detecting plagiarized paragraphs throughout the process of detection, and a performance comparison between different techniques of the searching process is to proposed in this research.

ملخص

الانتحال او خرق حقوق النشر و الطبع المحفوظة للنصوص المحمية هي عملية منتشرة عبر الانترنت , و عملية كشف هذا الخرق و الانتهاك بواسطة الانسان تعتبر عملية متعبة و منهكة و مستنزفة للوقت. في هذا المشروع نقدم مسح عن الطرق و التقنيات المستخدمة في حل هذه المشكلة.

العميل المتنقل (mobile agent) عبارة عن برنامج لديه القدرة على الانتقال عبر الشبكة من جهاز حاسوب الى اخر [14].

عملية التكامل بين الطرق المستخدمة لحل مشكلة الانتحال و العميل المتنقل تصنف تحت عنوان استنباط المعلومات, في هذا المشروع تم تصميم و تنفيذ نظام يستفيد من العملاء المتنقلين لعملية البحث و الكشف عن النصوص المنتحلة عبر الانترنت , هذا التكامل يهدف الى تقليل الوقت و التكلفة اللازمة لهذه العملية, تقنيات اخرى مثل السرية و استدراك الاخطاء استخدمت في هذا النظام لزيادة فعالية هذا النظام.

يهدف النظام الى الحصول على دقة عالية في كشف الانتحال بين المستندات, و مقارنة بين الاساليب المختلفة المستخدمة في عملية التنقل تم تقديمها في هذا البحث.

Dedication

*To our families and friends, we
dedicate this humble effort.*

Project team.

Acknowledgment

We would like to express our deep gratitude to Dr. Radwan Tahboub who very selflessly offered a lot of his time and gave a very good sense of guidance in supervising the writing of this documentation as well as giving the necessary advice and directing during the course.

We also would like to thank Eng. Amal Dweik for the software code she gave to us that helped in developing this project, and for her advices and suggestions to improve the project.

List of contents

<u>Chapter 1: Introduction</u>	1
1.1. Overview	2
1.2. Project Importance	3
1.3. Project Objectives	3
1.4. Literature Review	4
1.5 Requirements	5
1.6. Time Plan	6
1.7. Road Map	8
<u>Chapter 2: Theoretical Background</u>	9
2.1. Copyright	10
2.2. Plagiarism	12
2.3. Mobile Agents	12
2.4. Software Description	16
2.5. Survey on Plagiarism Detection Systems	20
2.6. Questionnaire on Plagiarism at Palestine Polytechnic University	21
2.7. Summary	23
<u>Chapter 3: Conceptual Design</u>	24
3.1. Detailed Project Objectives	25
3.2. Architectural Design	26
3.3. System Modeling	41
3.4. Summary	44

<u>Chapter 4: Detailed System Design</u>	45
4.1. Architecture of PLADMA	46
4.2. Detailed Specifications of PLADMA Components	48
4.3. Agent's Tasks and Interactions in JADE	51
4.4. Detailed System Modeling	54
4.5. GUI Design	88
4.6. Fault Tolerance Module	90
4.7. Security Module	94
4.8. Limitations	96
4.9. Assumptions	96
4.10. Summary	97
<u>Chapter 5: Implementation and Testing</u>	98
5.1. Development Environment	99
5.2. Development Process	100
5.3. Testing	101
5.4. Summary	125
<u>Chapter 6: Conclusion and Future Work</u>	126
6.1. Experimental Results	127
6.2. Conclusions	134
6.3. Future Work	135
6.4. Summary	135
Appendix A	136
Appendix B	155
Appendix C	162
References	207

List of tables

Table 1.1. Tasks description (first semester)	6
Table 1.2. Time plan (first semester)	7
Table 1.3. Tasks description (second semester)	7
Table 1.4. Time plan (second semester)	7
Table 2.1. Fault types in MASs	16
Table 2.2. Comparison among plagiarism detection software	20
Table 3.1. Pseudocode of Karp-Rabin algorithm	37
Table 3.2. Tri-grams generation	39
Table 3.3. Hashes generated for tri-grams (original text).	39
Table 3.4. Hashes generated for tri-grams (modified text)	40
Table 3.5. DFD level 1 processes	42
Table 3.6. DFD level 1 data	42
Table 6.1. Hardware used in experiments	127

List of figures

Figure 2.1. Basic structure of a mobile agent	14
Figure 2.2. Mobile agent paradigm	14
Figure 2.3. Main architectural elements of a JADE platform	17
Figure 2.4. Relationship between the main architectural elements	17
Figure 2.5. Component technologies in Java SE platform	19
Figure 2.6: Plagiarism ratio among graduate students	22
Figure 3.1. General idea of PLADMA	26
Figure 3.2. System block diagram	27
Figure 3.3. Steps of the WA's mission	29
Figure 3.4. Components interaction	30
Figure 3.5. C ³ A and C ² A interaction	31
Figure 3.6. Sequential search	32
Figure 3.7 Sequential search using multiple agents	33
Figure 3.8 Parallel search.	34
Figure 3.9. Distributed search (1)	35
Figure 3.10. Distributed search (2)	35
Figure 3.11. DFD level 0	41
Figure 3.12. DFD level 1	41
Figure 3.13. Use cases at the digital library system	42
Figure 3.14. Use cases at user-system interaction subsystem	43
Figure 3.15. Sequence diagram describes the plagiarism detection operation	43
Figure 3.16. Sequence diagram describes collecting the categories.	44
Figure 4.1: System architecture.	47
Figure 4.2. Agent thread path of execution	51
Figure 4.3. Flowchart of Check Documents	55
Figure 4.4. Flow chart of Signature Generator	56

Figure 4.5. Flowchart of Generate signatures of files	57
Figure 4.6. Flow chart of Set number of agents	58
Figure 4.7. Flowchart of Set the chunk size	59
Figure 4.8. : DCA's life cycle	60
Figure 4.9. Flowchart of Setup	61
Figure 4.10. Sequence diagram of Locations	62
Figure 4.11. Flowchart of Parallel search	63
Figure 4.12. Flowchart of Sequential search	64
Figure 4.13. Flowchart of Distributed sequential and parallel search	65
Figure 4.14. Flowchart of Process results	66
Figure 4.15. Flowchart of distributed chunks filter	67
Figure 4.16. Flowchart of lonely chunks filter	68
Figure 4.17. Flowchart of Generate report	69
Figure 4.18. Flowchart of Show results	70
Figure 4.19. Flowchart of Send killers	71
Figure 4.20. Worker agent's life cycle	72
Figure 4.21. Flowchart of Setup	73
Figure 4.22. Flowchart of afterMove	74
Figure 4.23. Flowchart of Check	75
Figure 4.24. Flowchart of Percentage	76
Figure 4.25. Flow chart of takeDown	77
Figure 4.26. Killer agent's life cycle	78
Figure 4.27. Flowchart of Setup	79
Figure 4.28. Flow chart of afterMove	80
Figure 4.29. Sequence diagram of Agents	81

Figure 4.30. Categories collector agent' life cycle	82
Figure 4.31. Flow chart of afterMove	83
Figure 4.32. C3A life cycle	84
Figure 4.33. Flow chart of Setup	85
Figure 4.34. Flow chart of receiving messages	86
Figure 4.35. BCA life cycle	87
Figure 4.36. Main window	88
Figure 4.37. Digital library	88
Figure 4.38. Results	89
Figure 4.39. Open Documents	89
Figure 4.40. WA is at host container	90
Figure 4.41. WA made a copy of itself (CWA) at the host container and then moves to Digital Library 1	91
Figure 4.42. WA got killed	91
Figure 4.43. WA continued working	92
Figure 4.44. WA is at host container	92
Figure 4.45. WA made a copy of itself (CWA) at the host container and then moved to Digital Library 1	93
Figure 4.46. WA got killed	93
Figure 4.47. WA continued working	94
Figure 4.48. Security step 1: DCA generates a random key (K)	94
Figure 4.49. Security step 2: DCA activates WA provide with K, DCA saves K	95
Figure 4.50. Security step 3: communication between WA and DCA	95
Figure 4.51. Encryption of data	95
Figure 4.52. Decryption of encrypted data	95
Figure 5.1. NetBeans IDE	100
Figure 5.2. Trying to select a folder with no documents	102

Figure 5.42. Centralized: WA 1-2 is at the third container continuing WA 1-1's work, WA 1 is at the home container	122
Figure 5.43. WA 1 is at the first container	123
Figure 5.44. Killing of the third container	123
Figure 5.45. Killing of the fourth container	124
Figure 5.46. WA skipped the lost containers and moved to the fifth container	124
Figure 6.1. Number of agents vs. Time	128
Figure 6.2. Experiment two using 1, 2, and 3 agents	129
Figure 6.3. 1 Agent vs. 2 Agents	129
Figure 6.4. 1 Agent vs. 3 Agents	130
Figure 6.5. Sequential search vs. parallel search	130
Figure 6.6. PPU WAN	131
Figure 6.7. LAN vs. PPU WAN	132
Figure 6.8. Parallel search vs. distributed search	132
Figure 6.9. Size of input documents vs. size of signature files	133
Figure 6.10. Size of input documents vs. time needed to generate signatures	134

List of abbreviations

PLADMA	Plagiarism Detection system using Mobile Agents.
MA	Mobile Agent.
WWW	World Wide Web.
GUI	Graphical User Interface.
DRM	Digital Rights Management.
JADE	Java Agent DEvelopment framework.
FIPA	Foundation of Intelligent Physical Agents.
CPU	Central Processing Unit.
Java SE	Java Standard Edition.
JRE	Java Runtime Environment.
JDK	Java Development Kit.
IDE	Integrated Development Environment.
OS	Operating System.
PPU	Palestine Polytechnic University.
C ³ A	Categories Collectors' Coordinator Agent.
DCA	Documents Coordinator Agents.
BCA	Buttons Controller Agents.
WA	Worker Agent.
KA	Killer Agent.
C ² A	Categories Collector Agents.
TF-IDF	Terms frequency - inverse document frequency..
DL	Digital Library.
DFD	Data Flow Diagram.
AMS	Agent Management System.
MB	Mega Byte.
GB	Giga Byte.
GHz	Giga Hertz.
LAN	Local Area Network.
WAN	Wide Area Network.

CHAPTER 1

INTRODUCTION

1.1. Overview

1.2. Project Importance

1.3. Project Objectives

1.4. Literature Review

1.5. Requirements

1.6. Time Plan

1.7. Road Map

Chapter One

Introduction

1.1. Overview

According to Merriam Webster [46] plagiarizing as a transitive verb means: to steal and pass off (the ideas or words of another) as one's own: use (another's production) without crediting the source; and this is what most of the students do at preparing their homework, reports, and researches, this problem is becoming a habit for most of the students.

Also copyright infringement of digital text is a serious problem which threatened authors of creative works even in the non-electronic world. In the electronic world, easy access to electronic documents and the ease of reproducing and distributing these documents have made copyright infringement even a bigger problem [8].

Although copying a work without obtaining permission may appear to be an easy and convenient solution to an immediate problem, such unauthorized copying can frequently violate the rights of the author or publisher of the copyrighted work [15].


Mobile Agent (MA) is executing program that migrates during execution and present methods for maintaining and using distributed systems [14]. In this project, a new approach and an implementation using mobile agents is developed to solve the problem of efficiently checking plagiarism and copyright infringement of digital text in the Internet.

This new approach depends on a mobile agent that holds signatures derived from a document and travels around the Internet to find where the data was taken from by comparing it to signatures of documents in different digital libraries around

the Internet, also different MA techniques will be studied and implemented accordingly. These techniques include MA navigation, MA security and fault tolerance.

1.2. Project Importance

Plagiarism detection is very difficult and much time consuming process, and if done by human will require much cost, patience and time, so this project comes to help detecting plagiarism by discovering the copying of a text without and citation of the origin of the text and to reduce the cost and time for doing the process by human.

This project will help educators and students to assist in identifying writing that is suspected to have been written by someone other than the student submitting the work for assessment. The idea of copyright detection using mobile agents is a novel one and it was never implemented before. 

1.3. Project Objectives

- To survey the different plagiarism detection techniques and algorithms and choose an appropriate one for this system.
- To design and implement a system that makes use of mobile agents to compare the document on the Internet against documents that may contain plagiarized text.
- Making the process of detecting plagiarism easier and simpler.
- To apply techniques and methods that make the system fault tolerant, that is to mask some failures that might happen throughout the agents' life cycle.
- To apply security enhancement for the agents in order to securely communicate.

1.4. Literature Review

1.4.1 Literature review related to copyright infringement and plagiarism detection

- *"Ikena Copyright"*

This technology was implemented by MotionDSP Company, Ikena is a video copyright detection technology that matches content solely on the video content, it uses a unique technology that creates a 'video fingerprint' by tracking the motion characteristics the video [9].

- *"Automated copyright detection in digital images"*

In [16], Kacker, Dhiraj, Muzzolini, Russell Ennio, presented a computer-assisted method for detecting copyright protection in digital images. It receives a digital image, searches for capture device information associated with the digital image, calculates one or more image characteristics of the digital image, and verifies copyright ownership and usage authorization of the digital image in response to the capture device information and the calculated image characteristics.

- *"SNITCH: A Software Tool for Detecting Cut and Paste Plagiarism"*

In [17], Sebastian Niezgoda and Thomas P. Way, presented a design and implementation of a software tool called SNITCH, which implements a fast and accurate plagiarism detection algorithm using the Google Web API.

- *"SPLAT: A System for Self-Plagiarism Detection"*

In [18], Christian S. Collberg, Stephen G. Kobourov, Joshua Louie, Thomas Slattery at the department of computer science in university of Arizona presented a system for self-plagiarism detection, the system uses a WebL web spider that crawls through the web sites of the top fifty computer science departments, downloading research papers and grouping them by author, and uses a text-comparison algorithm to search for instance of textual reuse.

1.4.2. Literature review related to mobile agents:

- *"Mobile Agents in WWW Access Optimization"*

Dr. Radwan Tahboub, Prof. Dr. V. Lazaerscu, talked in this paper about developing an abstract approach to improve the WWW access optimization using Mobile Agents to replicate web sites into distributed ISP's [14].

- *"A Web-Based Face Recognition System Using Mobile Agent Technology"*

Xinfeng Yang, Yanchao Xing, Ann Nguyen, and Xi' proposed this application of mobile agent technology to solve the problem of web-based face recognition; the proposed system integrates the computing advantages of mobile agents with several improved face recognition algorithms to enhance system robustness [19].

1.5 Requirements

1.5.1 User requirements

- The system should provide a Graphical User Interface (GUI) in order to interact with the user in an easy and a simple way.
- The GUI should have an option for the user to provide the system with the document.
- The GUI should have an option for the user to start the detection process.
- The GUI should provide the user with the results and their explanation.
- The GUI should provide the user with searching, fault tolerance and security options.
- The GUI should provide an option of making signatures for a collection of documents (so as to construct a digital library).
- The GUI should provide the user a stopping criteria specified by the percentage of the found plagiarized text.

1.5.2. System requirements

1.5.2.1 Functional requirements

- The system should check the documents for their suitability to be processed for plagiarism checking.
- The system should analyze the documents and generate their fingerprints (collection of signatures).
- Mobile agents should be sent to collect categories from different digital libraries.
- Mobile agents should be sent, provided with fingerprints of the documents and optionally their category to different digital libraries.
- The mobile agent -holding fingerprints- should perform a matching process at each digital library.
- The mobile agents -holding fingerprints- should send the result of matching after finish checking all digital libraries.

1.5.2.2 Non-functional requirements

- Usability: the system should be easy to use.
- Reliability: the system should be reliable under different circumstances.
- Security: communication between agents should be secured.
- Performance: the system should work with good performance measures.

1.6. Time Plan

Table 1.1. Tasks description (first semester)

Task ID	Task description
T1.1	Selecting the project.
T1.2	Collecting information, literature review and related theory.
T1.3	System analysis and requirements determination.
T1.4	System design.
T1.5	Writing documentation.

Table 1.2. Time plan (first semester)

Task/week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1.1																
T1.2																
T1.3																
T1.4																
T1.5																

Table 1.3. Tasks description (second semester)

Task ID	Task description
T2.1	System design and analysis.
T2.2	Component implementation.
T2.3	Components testing.
T2.4	Integrity testing.
T2.5	System review and incorporating feedback.
T2.6	Requirements refinement.
T2.7	Writing documentation.

Table 1.4. Time plan (second semester)

Task/week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T2.1																
T2.2																
T2.3																
T2.4																
T2.5																
T2.6																
T2.7																

1.7. Road Map

The project consists of six chapters, each chapter talks about a specific area of the project, this page is the final page in chapter one, following is a description of the next chapters.

Chapter Two: Theoretical Background, this chapter gives a clear picture about the system theoretical background related to the main concepts of copyright, agent, mobile agents, and security in mobile agents, fault tolerance and system basic components.

Chapter Three: Conceptual design, this chapter shows system block diagram, design options that can be used in the system, and how each system components interface with each other.

Chapter Four: Detailed system design, this chapter describes the program that operates the project, pseudocode and algorithms that describe project systems processes.

Chapter Five: System Implementation and Testing, this chapter discusses the actual project testing and implementation in details.

Chapter Six: Conclusion and Future Work, this chapter is simplifying the conclusions and results achieved after implementing the entire project and gives suggestions for the future system developing.

CHAPTER 2

THEORETICAL BACKGROUND

2.1. Copyright

2.2. Plagiarism

2.3. Mobile Agents

2.4. Software Description

2.5. Survey on Plagiarism Detection Systems

2.6. Questionnaire on Plagiarism at Palestine Polytechnic University (PPU)

2.7. Summary

Chapter Two

Theoretical Background

This chapter provides an illustrative theoretical background for the project related topics, software and components.

2.1. Copyright

2.1.1. Basic concept

A copyright is a legal tool that provides the creator of a work of authorship the right to control how the work is used, including the exclusive right to reproduce, distribute, adapt, display and perform the work. For example, the author of a book or musical composition has the exclusive right to publish it [3].

2.1.2. The copyright problem

Copyright protects works of authorship. A work of authorship is any creative work created by a human being that can be communicated to other humans, either directly or with the aid of a device such as a film projector. Works of authorship include, but are not limited to: writings of all types, musical works, including song lyrics, plays, photographs, databases, maps, artworks, sculpture and graphics, movies and videos, computer software, sound recordings, pantomimes and choreographic works, and, architectural drawings and blueprints and the design of actual buildings [3].

The digital copyright problem involves a complex set of stakeholders that include copyright holders, users, rights defenders, innovators, and decision makers.

The concerns and interactions of these stakeholders are complicated: many of them share members; others include members with conflicting interests [10].

Recently, the digital world required some changes. While promising to promote progress by enabling copyright holders to reach wider audience at low reproduction and distribution costs, and enabling the public to access a wider range of works to learn from, the digital world also enabled unauthorized reproduction and distribution of works and threatened the royalty-based businesses of copyright holders. The amendments aimed at addressing the concerns related to unauthorized reproduction and distribution of works failed to re-establish the balance between the rights and incentives of different stakeholder groups. One of the most drastically affected groups is the public whose ability to access and use digital works can be controlled and limited by Digital Rights Management (DRM) systems [10].

2.1.3. Approaches to copyright infringement detection for text

There have been two kinds of attempts to solve the problem of copyright violations on the web [8].

- The first kind is based on protection of the work by making it difficult to copy.
- The second kind of attempts can be generally called copy detection. These systems detect copyright violations by finding partial overlaps among documents.

Copy detection methods do not make copyright infringement impossible either. However, they might be effective deterrents.

Detection tools can be: [8]

- *Verbatim*: There have been a few attempts to identify “verbatim” copyright violations, (i.e., violations where part of the document has been copied character for character). These systems prepare fingerprints of documents

from the web in order to form a repository and match target documents against the fingerprints in their repository.

- *Non-verbatim*: None of these attempts forms a conceptual fingerprint of documents. With string based fingerprinting, alternative ways of communicating the same concepts end up being represented differently, and this makes slightly paraphrased but conceptually equivalent documents look completely unrelated.

2.2. Plagiarism

Plagiarism is stealing somebody's work or idea; the process of copying another person's idea or written work and claiming it as original [11].

Plagiarism is becoming a habit for most students running away from creativity, reading and understanding ideas and sentences, "cut and paste" may become one day a star in the flag for the country of education, easy access to text online, by searching Google[47] or Yahoo![48], a student can find free ideas, free articles, free notes, that can be easily grabbed and included in a document within seconds, and then submitted as if the student had done all the work!

Many systems were created to detect plagiarism, many algorithms improved, many techniques were used, but still large number of students plagiarizes.

Whether it is copyright infringement, or plagiarism, the problem is one, and solutions are aimed to solve the problem.

2.3. Mobile Agents

2.3.1. Definition of an agent

There is no commonly agreed upon definition of exactly what an agent is [1]. However there are many definitions of an agent, from the system perspective an agent is a software object that is situated within an execution environment and possesses the following mandatory properties: [5]

- *Reactive* - senses changes in the environment and acts accordingly to those changes.
- *Autonomous* - has control over its own actions.
- *Goal driven* - is pro-active.
- *Temporally continuous* - is continuously executing.

And may possess any of the following orthogonal properties: [5]

- *Communicative* - able to communicate with other agents.
- *Mobile* - can travel from one host to another.
- *Learning* - adapts in accordance with previous experience.
- *Believable* - appears believable to the end-user.

2.3.2. Definition of a mobile agent

A mobile agent can be defined as an object that migrates through many hosts in a heterogeneous network, under its own control, in order to perform tasks using resources of these hosts [2].

2.3.3. Mobile agent components

A mobile agent, as depicted in Figure 2.1., consists of three parts: code, state and data [4].

- The *code* is that aspect of the agent that is executed when it migrates to a platform. In the simplest case there is a single code.
- The *state* is the data execution environment of the agent, including the program counter and the execution stack.
- The *data* consists of the variables used by the agents, such as knowledge, file identifiers, etc.

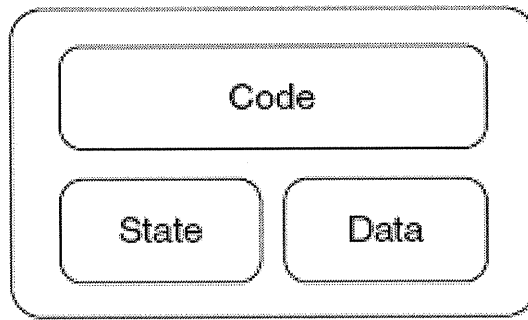


Figure 2.1. Basic structure of a mobile agent [4]

2.3.4. Mobile agent paradigm

A key characteristic of the mobile agent paradigm, see figure 2.2. , is that any host in the network is allowed a high degree of flexibility to possess any mixture of know-how (code), resources, and processors. Its processing capabilities can be combined with local resources. Know-how (in the form of mobile agents) is not tied to a single host but available throughout the network [5].

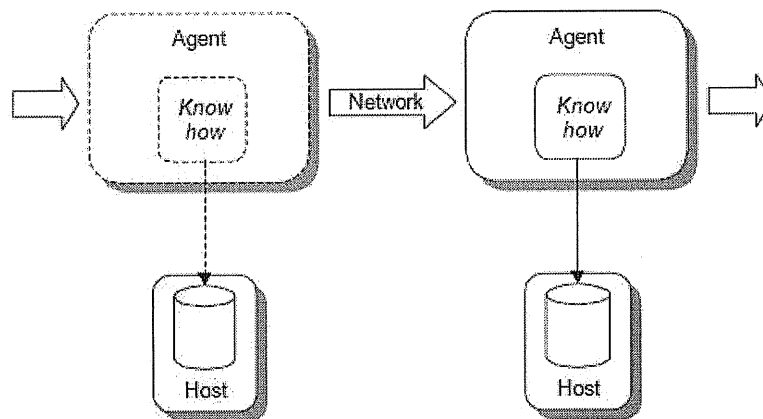


Figure 2.2. Mobile agent paradigm [5]

2.3.5. Mobile agent security

Due to the mobility of the agent it needs protection from other agents and from the hosts on which they execute. Similarly, hosts need to be protected from agents and from other parties that can communicate with the platform [7].

Just as an agent's communication with its environment needs to be protected, the following security properties should be provided: [7]

- *Confidentiality*: assurance that communicated information is not accessible to unauthorized parties.
- *Data integrity*: assurance that communicated information cannot be manipulated by unauthorized parties without being detected.
- *Authentication of origin*: assurance that communication originates from its claimant.
- *Availability*: assurance that communication reaches its intended recipient in a timely fashion
- *Non-repudiation*: assurance that the originating entity can be held responsible for its communications.

2.3.6. Fault tolerance in multi-agent systems (MAS)

2.3.6.1. Definition of fault tolerance

Fault tolerance is the ability of a system to respond gracefully to an unexpected hardware or software failure [6].

2.3.6.2. MAS faults and failure

A failure occurs when the system produces results that do not meet the specified requirements. A fault is defined to be a defect within a component of a MAS that may lead to a failure. Faults in a MAS can be grouped into five categories as shown in Table 2.1. When a fault does occur in a MAS, interactions between agents may cause the fault to spread throughout the system in unpredictable ways [21].

Table 2.1. Fault types in MASs

Fault Type	Description
Program bugs	Errors in programming that are not detected by system testing.
Unforeseen states	Omission errors in programming. The programming does not handle a particular state. System testing does not test for this state.
Processor faults	System crash or a shortage of system resources.
Communication faults	Slow downs, failures or other problems with the communication links.
Emerging behavior	System behavior which is not predicted; Emerging behavior may be beneficial or detrimental.

2.4. Software Description

2.4.1. The JADE platform

Java agent development framework (JADE) is a software platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction [4].

2.4.1.1. JADE architecture

Figure 2.3. shows the main architectural elements of a JADE platform. A JADE platform is composed of agent containers that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents [4].

There is a special container, called the main container, which represents the bootstrap point of a platform, it is the first container to be launched and all other containers must join to a main container by registering with it. The UML diagram in Figure 2.4 schematizes the relationships between the main architectural elements of JADE [4].

The programmer identifies containers by simply using a logical name; by default the main container is named 'Main Container' while the others are named 'Container-1', 'Container-2', etc. Command-line options are available to override default names [4].

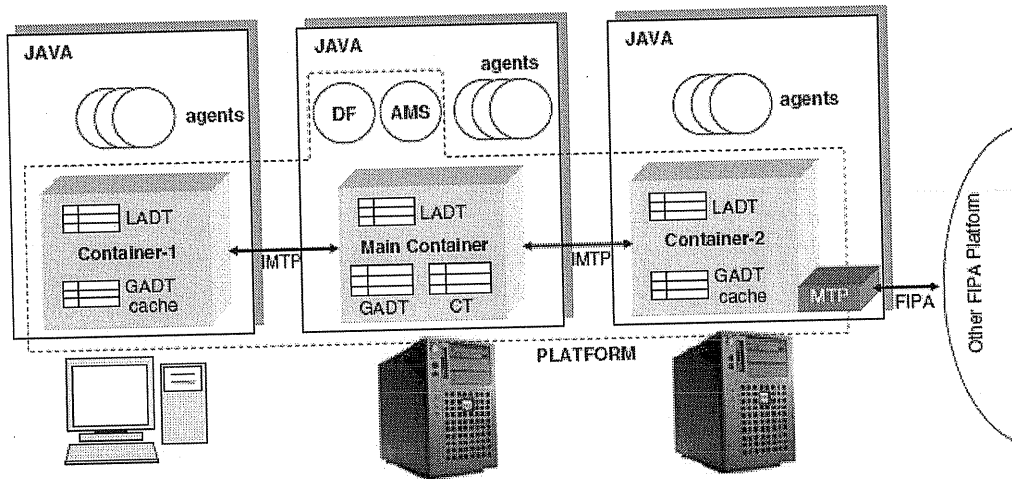


Figure 2.3. Main architectural elements of a JADE platform [4]

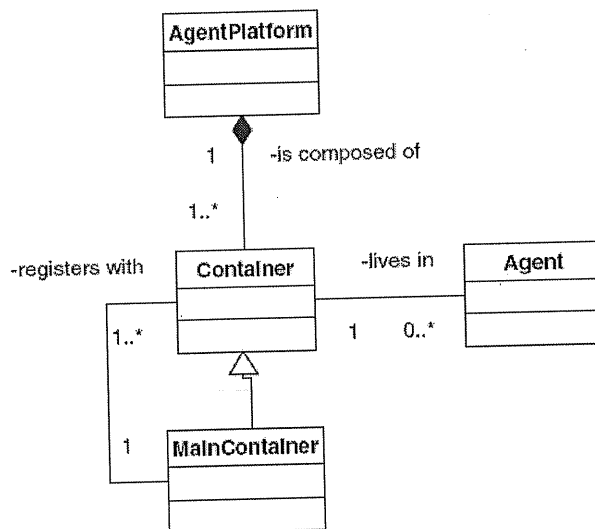


Figure 2.4. Relationship between the main architectural elements [4]

2.4.1.2. Foundation of intelligent physical agents (FIPA)

FIPA is a non-profit making organization that was established in 1996 with the aim of promoting specifications for facilitating interoperability between agent systems. The model proposed by FIPA consists of concrete specifications enabling interoperability, based on a high-level abstract architecture [20].

2.4.2. Java programming language

Java, in computer science, object-oriented programming language introduced in 1995 by Sun Microsystems, Inc. Java facilitates the distribution of both data and small applications programs, called applets, over the Internet. Java applications do not interact directly with a computer's central processing unit (CPU) or operating system and are therefore platform independent, meaning that they can run on any type of personal computer, workstation, or mainframe computer. This cross-platform capability, referred to as "write once, run everywhere," has caught the attention of many software developers and users. With Java, software developers can write applications that will run on otherwise incompatible operating systems such as Windows, the Macintosh operating system, OS/2, or UNIX [11].

2.4.3. Java platform, standard edition (Java SE)

There are two principal products in the Java SE platform family: Java SE Runtime Environment (JRE) and Java Development Kit (JDK) [13].

- **Java runtime environment (JRE)**

The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language [13].

- **Java development kit (JDK)**

The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications. The conceptual diagram in figure 2.5. illustrates all the component technologies in Java SE platform and how they fit together [13].

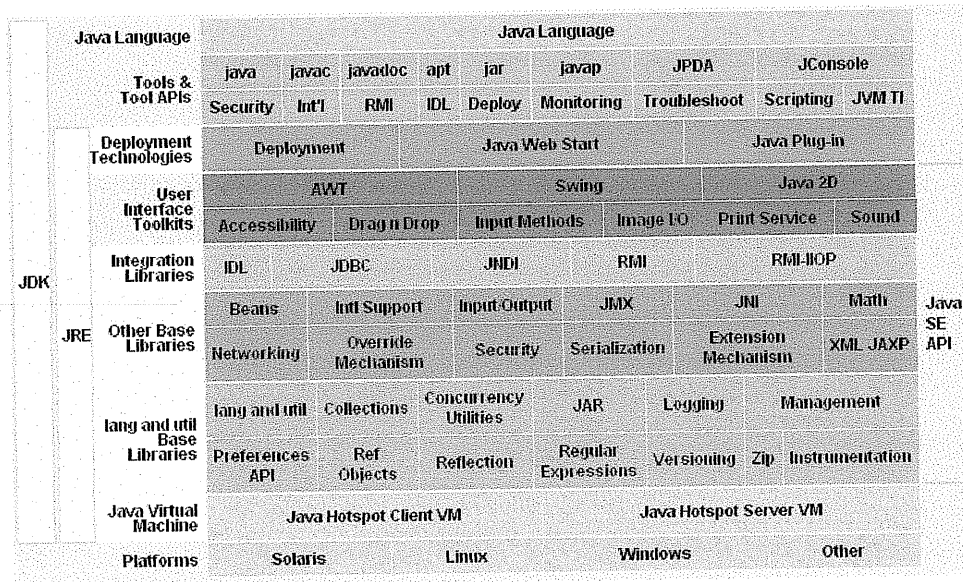


Figure 2.5. Component technologies in Java SE platform [13]

2.4.1. NetBeans IDE

The NetBeans Integrated Development Environment (IDE) is a free, open-source IDE for software developers, programmers get all the tools they need to create professional desktop, enterprise, web and mobile applications, in Java, C/C++ and even Ruby. The IDE runs on many platforms including Windows, Linux, Mac OS X and Solaris; it is easy to install and use straight out of the box [12].

2.5. Survey on Plagiarism Detection Systems.

We made a survey on different plagiarism detection systems, the results is summarized in table 2.2, with five of the systems that we reviewed.

For the complete discussion see appendix A.

Table 2.2. Comparison among plagiarism detection software

	Turnitin[32]	MyDropBox[30]	Plagiarism Checker[22]	WCOPYfind[23]	Copyscape[24]
Technology	Web based, server side processing.	Web based, Server side Processing.	Web base, searches Google and Yahoo.	Window based.	Web based, server side processing.
Supported file types	MS Word, PostScript, PDF, HTML, RTF, and plain text.	ZIP, DOC, TXT, PDF, RTF, HTML.	Direct cut and paste in text box at site.	MS Word, HTML, and plain text.	Only submitted URLs.
Verbatim	Yes.	Yes.	Yes.	Yes.	Yes.
Non-verbatim	No.	No.	No.	No.	No.
Comparison	Compares uploaded documents against the internet.	Compares uploaded documents against the internet.	Compares uploaded documents against the internet	Only compares submitted documents by user against each others.	Compares web page of the submitted URL against the internet.
Cost	Depends on the number of students.	Depends on the number of students.	Free.	Free.	\$0.05 per search.

1. Do you think that such programs will reduce plagiarism?
a. Yes b. No
 2. Do you think that such programs will corrupt the relation between students and their teachers?
a. Yes b. No
- The questionnaire's results were disappointing, about 60% of the students participated in this questionnaire have committed plagiarism either from their colleagues' reports, or from the internet.

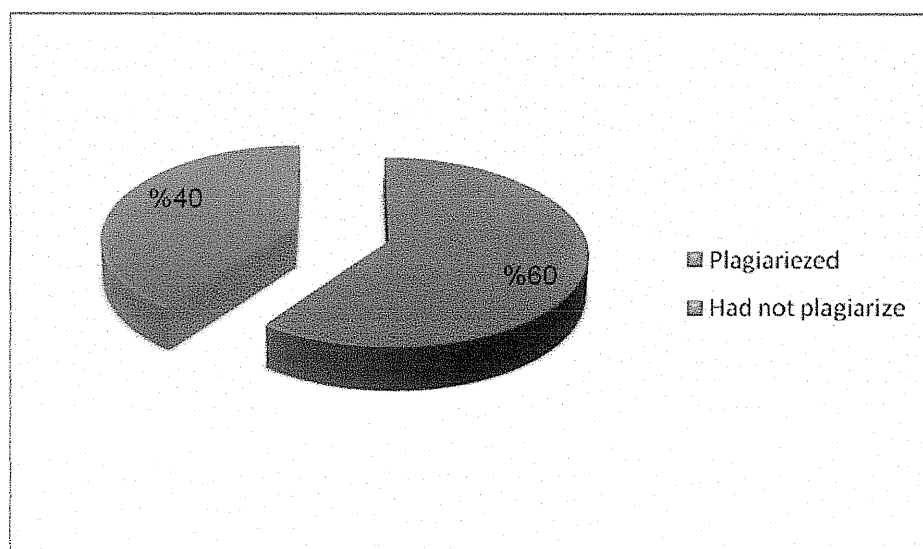


Figure 2.6: Plagiarism ratio among graduate students

- We noticed that students rely too much on online documents to do their reports and researches, that is "cut and paste" is becoming a habit for most of students, also from the questionnaire, about half of the students don't have the ability to collect thought and make an effective writing using their own words, and the cumulative process of surrendering to this weakness reduce more and more the ability of the student to understand and progress in his study at the university.
- Even though most of the students know the possible punishment of the detection of plagiarism, and that plagiarism could violate the copyrights of the owner of the documents, still a large fraction of them commits plagiarism.
- There are a high percentage of students who don't know about programs that detect plagiarism; we think that if students know about plagiarism programs

and how these programs can detect plagiarism, then the percentage of plagiarism among students will decrease in a great manner.

- As a conclusion, Internet is becoming the main source of interest of educational activities done by students, specially written ones, and committing plagiarism will degrade the educational and mental levels of the students.

For the complete results of the questionnaire see appendix B.

2.7. Summary

This chapter provided an illustrative theoretical background for the project related topics, including software used in the project.

CHAPTER 3

CONCEPTUAL DESIGN

3.1. Detailed Project Objectives

3.2. Architectural Design

3.3. System Modeling

3.4. Summary

Chapter Three

Conceptual Design

This chapter includes conceptual design issues, architectural design, system modeling, and detailed description of the project objectives.

3.1. Detailed Project Objectives

- To survey the different plagiarism detection techniques and algorithms and choose an appropriate one for this system.

We will make a survey on different plagiarism detection techniques and algorithms and consider one of them to be implemented in this system.

- To design and implement a system that makes use of mobile agents to compare the document on the Internet against documents that may contain plagiarized text.

This project mainly aims to design and implement a system that make use of mobile agents that travel around the Internet holding documents containing text data, and try to find the origin of this text, and then send the results of what they found.

- Making the process of detecting plagiarism easier and simpler.

Instead of letting a person search the Internet for a text and keeps on comparing it with some text, mobile agents could do all the process by themselves and return the results to the user.

- To apply techniques and methods that make the system fault tolerant, that is to mask some failures that might happen throughout the agents' life cycle.

The agent could face some problems while traversing the digital libraries on the Internet, even the platform of the agent could crash or face some problem, fault tolerance will be applied to the overall system to mask such problems.

- To apply security enhancement for the agents in order to securely communicate among each others.

Two modules are applied to the mobile agents:

- The security module.
- The fault tolerance module.

Two external entities interact with the system:

- The User of the system.
- The digital library system.

See figure 3.2.

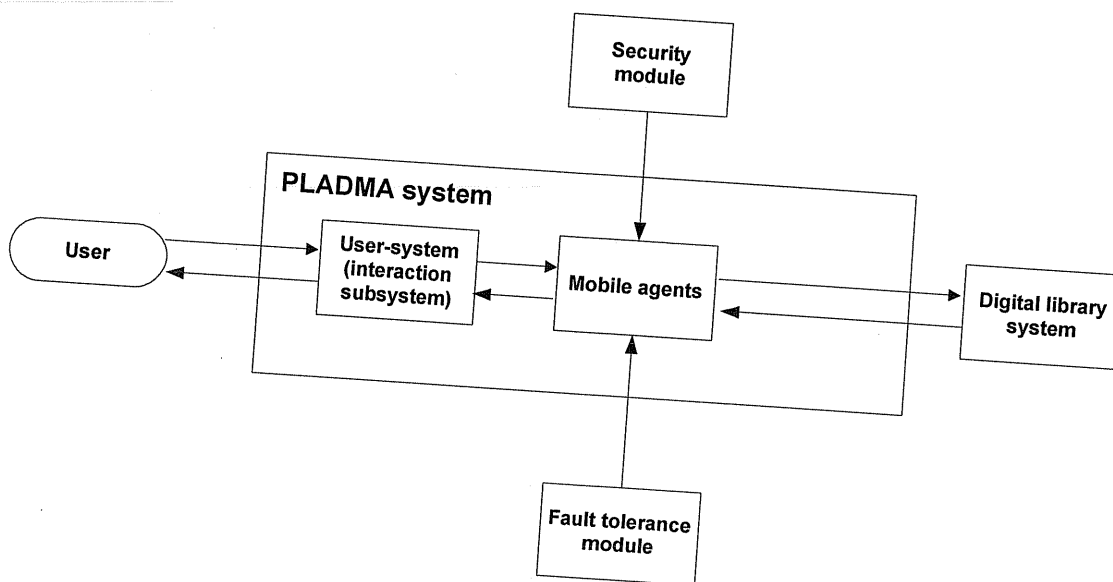


Figure 3.2. System block diagram.

3.2.3. User –system interaction subsystem

Contains the GUI that provides for the user different options including:

- Submission of the document by the user.
- Option to select the type of the document such as: scientific, historical, geographical... etc.
- Start of the copyright detection process.
- Viewing the result of the plagiarism or copyright infringement detection process.

This subsystem contains two types of stationary agents:

- Agents to set the categories provided by one type of the mobile agents we call these agents Categories Collectors' Coordinator Agents (C³A).
- Agents to coordinate the other types of mobile agents we call these agents Documents Coordinator Agents (DCA).
- Agent to control buttons according to the progress of the searching process we call these agents Buttons Controller Agents (BCA).

The first two types of stationary agents interact with the mobile agents in different ways such as:

- Providing mobile agents a migration plan.
- Providing mobile agents fingerprints of the documents.
- Activate mobile agents.

More details in the next chapter.

3.2.4. Mobile agents

There are three types of mobile agents in this system:

- Mobile agents to do the searching and matching process we call them Worker Agents (WA).
- Mobile agents to kill other agents at some stage of searching we call them Killer Agents (KA).
- Mobile agents to collect different categories from different digital libraries we call them Categories Collector Agents (C²A).

The WAs are the main mobile agents in the system, they do the searching and matching process these agents will interact with the DCA and the digital library system, they take instructions from the DCA for its migration plan; also they will carry the fingerprint(s) of the document(s) that it will work on at each digital library, the mission of this type is summarized in the next five steps, see figure 3.3, specifications of the other types are in the next chapter.

- Step 1: Leave the supervisor computer holding a fingerprint(s) of the submitted document(s).
- Step 2: Migrate to a digital library.
- Step 3: Do the searching and matching process and save the results.

- Step 4: Repeat (2, 3) until finishes traversing all digital libraries.
- Step 5: Return results to the user –system interaction subsystem.

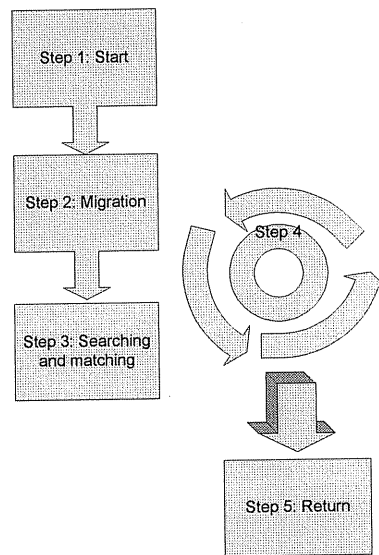


Figure 3.3. Steps of the WA's mission.

3.2.5. Digital library system

A digital library is a system that contains different type of documents; each digital library should have a database of documents categorized according to their type.

3.2.6. Fault tolerance module

In order for this system to be a high performance system it must be fault tolerant; it must be able to continue operating despite the failure of a limited subset of hardware or software or in the presence of failure.

Several approaches to fault-tolerance in MASs, such as using agent replication [21] that means creating one or more duplicates of one or more agents in a multi-agent system. Each of these duplicates is capable of performing the same task as the original agent.

In this system two variants of agent replication are applied, centralized approach and windowing approach [44, 43] more details in the next chapter.

3.2.7. Security module

Security is one important aspect of mobile agent systems. Due to communication, execution, and mobility of a mobile agent, it may face many security threats [7], to protect a mobile agent there have been many attempts to address the threats posed to mobile agents, in order to make the communication secure between different agents that communicate with each others in this system, encryption with symmetric variable key is applied.

3.2.8. How does the system work?

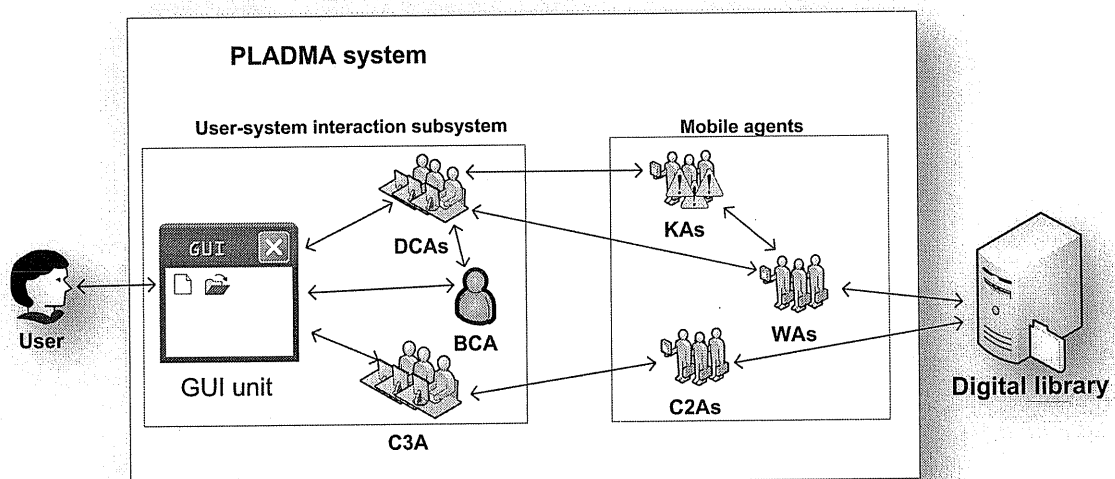


Figure 3.4. Components interaction.

Figure 3.4 describes interactions between different components of the system, as the GUI unit starts up it activates the C³A, this agent activates a number of C²As, each C²A is provided with the address of the digital library it is supposed to go to, C²As collect the categories available at each digital library and send them back to C³A an illustration in figure 3.5, then the user has to submit documents to be check for plagiarism, after submitting the documents, the user clicks start and the processes

starts, the user has 4 different options for the searching techniques described in section 3.2.9 .

When the operation starts, for each document a DCA is activated, each DCA is responsible for starting a number of WAs, each WA is provide by the fingerprint of the documents to be checked, when a WA reaches a digital library, it uses a plagiarism detection technique to check the respective documents at the digital library against the fingerprint it hold, after it finishes the detection process, it goes to another digital library or returns the result to the DCA; depending on the searching technique it uses.

Finally after a DCA gets all results from all WAs it has activated it generate reports that represent the text of the document, and spots the plagiarized sentences.

In the following two section a description of how the plagiarism check is done, and the different searching scenarios used.

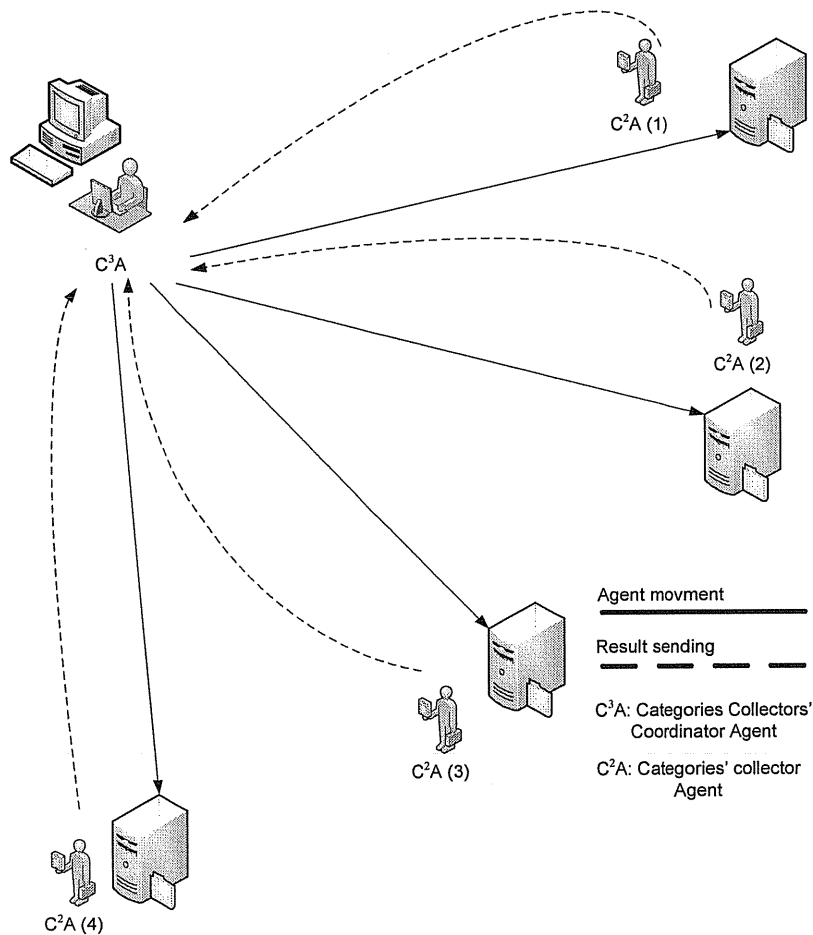


Figure 3.5. C³A and C²A interaction.

3.2.9. Searching scenarios

We suggest 4 scenarios for the searching process described as follows:

3.2.9.1. Scenario 1: Sequential searching using one agent.

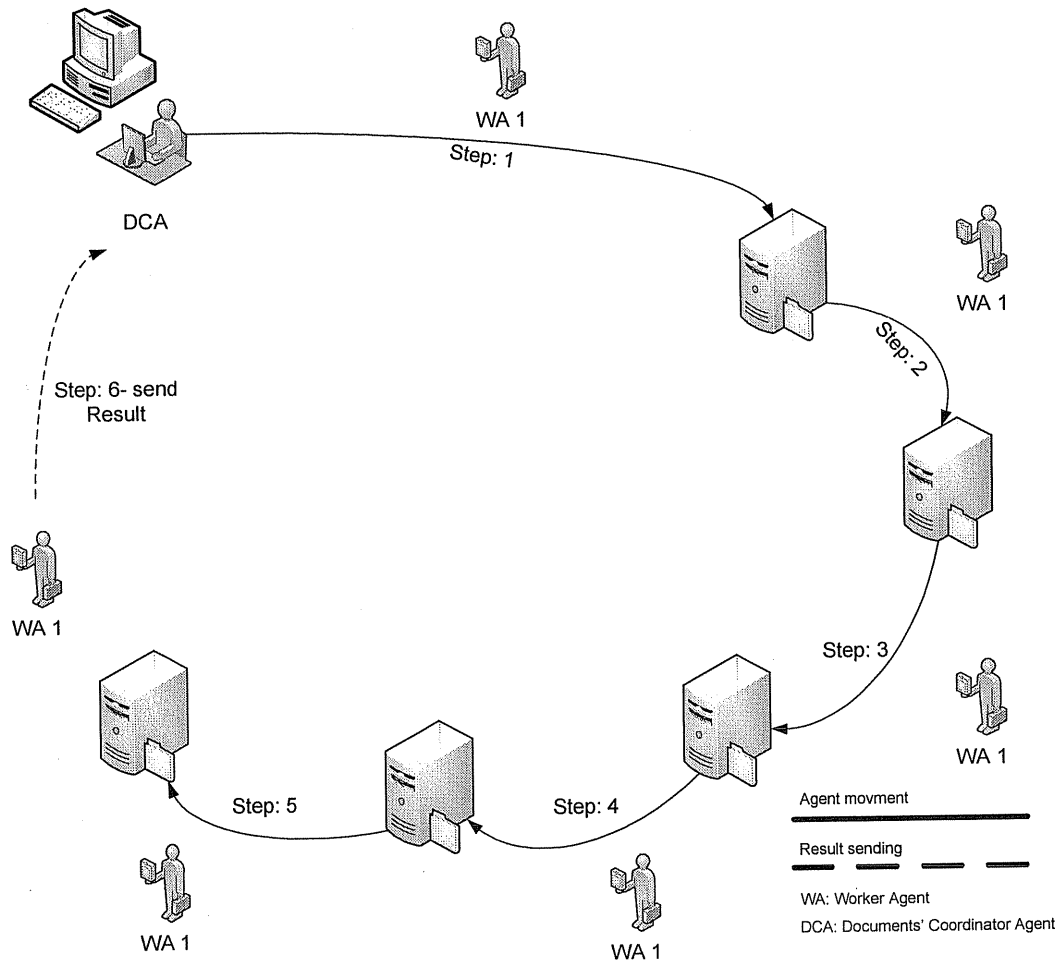


Figure 3.6. Sequential search.

The simplest scenario, where only one agent is activated, going to each digital library in sequential order, doing the "searching and checking" process on them one by one, and finally send the results to the DCA, see figure 3.6.

3.2.9.2. Scenario 2: Sequential searching using multiple agents.

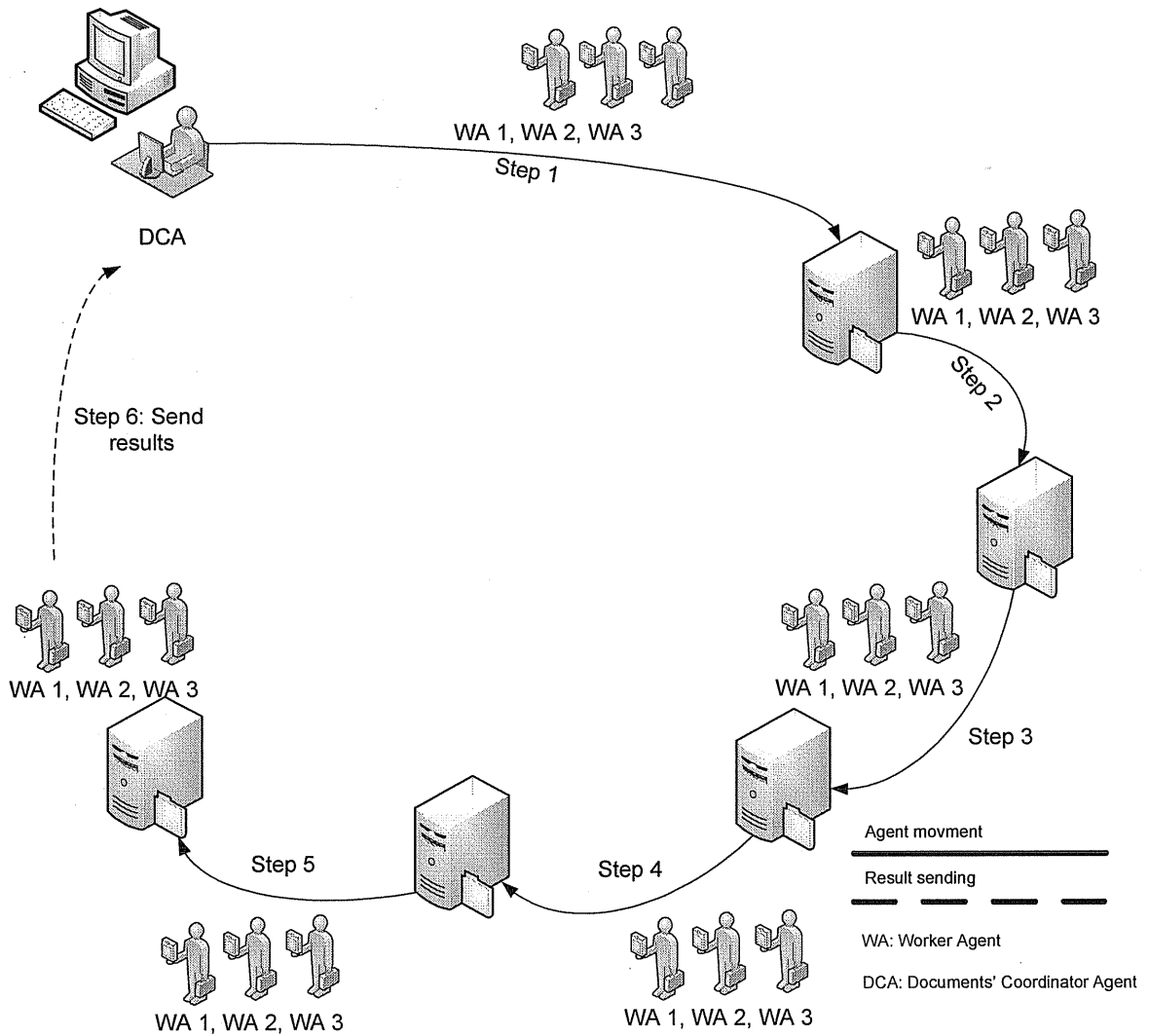


Figure 3.7. Sequential search using multiple agents.

Scenario 1 can be extended so that multiple agents are working on the document, all going in on the same order to different digital libraries, each agent checking a fraction of the signature files depending on the overall number of the agents, every agent is provided with the complete signature of the submitted document, each WA will send the results of the checking after it finishes searching all digital libraries, see figure 3.7.

3.2.9.3. Scenario 3: Parallel searching.

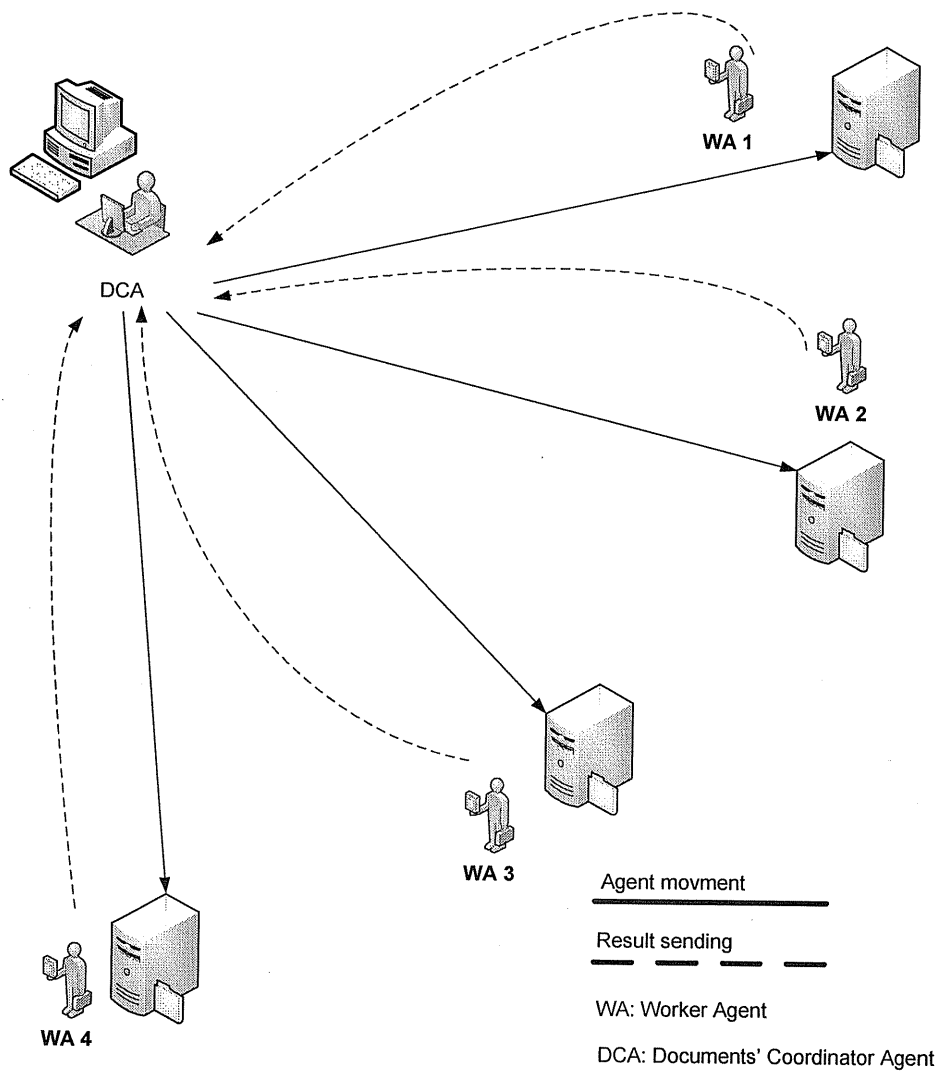


Figure 3.8. Parallel search.

This scenario works by sending an agent to each digital library, the agent works only on the respective digital library, all agents work in parallel, each agent holds the signatures of the submitted document, when an agent finishes work it sends the result to the DCA, the more documents are submitted the more agents are send to the digital libraries see figure 3.8.

3.2.9.4. Scenario 4: Distributed sequential and parallel searching.

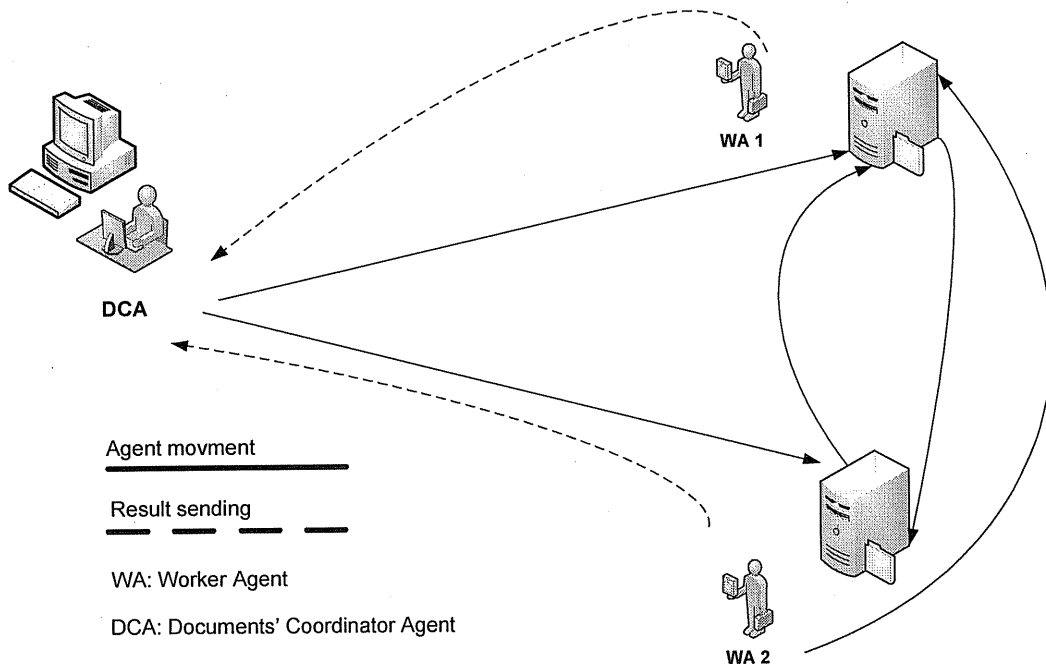


Figure 3.9. Distributed search (1).

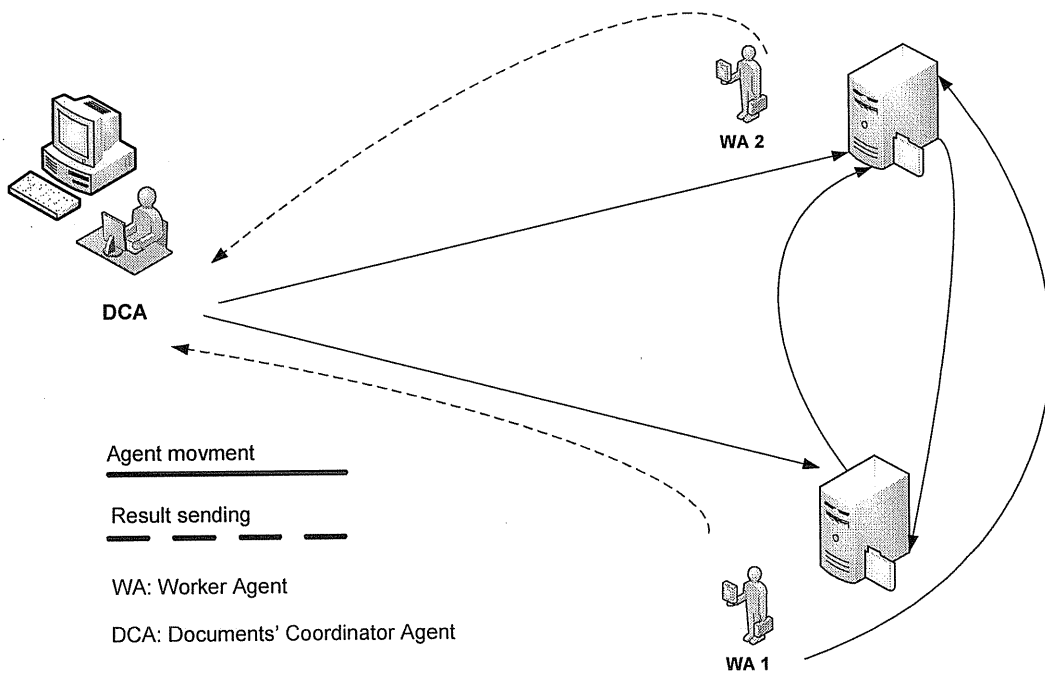


Figure 3.10. Distributed search (2).

Scenario 2 can be extended so that agents start working from different digital libraries in different orders, for example, WA 1 starts at digital library 1 checks through a fraction of the signature files, then goes to digital library 2, and WA 2 goes

to digital library 2 checks through a fraction of the signature files then goes to digital library 1 see figures 3.9 and 3.10.

3.2.10. Plagiarism detection

There are different techniques used in detecting plagiarism of text, either in verbatim detection or in non-verbatim detection.

3.2.10.1. Plagiarism detection techniques

- Frequency of *hapax legomena* [36], which are words that occur only one time in a text, this strategy can be used to compare two documents, where if the common percentage of *hapax legomena* between the two documents is above some percentage, then one of the documents is considered to be plagiarized from the other one.
- Use of TF-IDF approach (Terms frequency - inverse document frequency) [37], that computes a value for word or collection of words in the document, according to its frequency in the document itself, and its frequency in the searched documents, and compare that value to values that are previously calculated for the documents already in the database.
- Use of N-gram [38], which mean N number of words, for example a tri-gram words of the sentence (Palestine polytechnic university is a great university) are: (Palestine polytechnic university), (polytechnic university is), (university is a), (is a great), (a great university). After the generation of the N-gram chunks, they are compared to N-grams chunks generated from documents stored in a database of documents.
- The use of Google or any other search engine, by dividing the text into sentences and search each sentence using Google see [22] [29].
- By using statistical methods such as counting the punctuation marks, misspelled word, words distribution within the document see [40]
- Using linguistic analysis, such as vocabulary substitution detection, passive voice usage detection see [10]

In the plagiarism detection part of this project we used N-gram method, we preferred to use this method because we need to check the whole document, not only one word or one phrase, also the system is to check modified "cut and paste" sentences of the documents, modification of the word with its synonyms will not be check.

The N-gram method is based on a pattern matching algorithm called Karp-Rabin[39] see table 3.1, this algorithm is based on having a preprocessing stage of calculating a hash value for the string, and then calculating hash values to the paragraph according to the length of the string, suppose string length is M and paragraph length is N then N-M+1 hash values will be computed for the paragraph, for example let "abc" be the string and "bcdaabcd" be part of the paragraph, and $S = H(abc)$ be the hash value of "abc", then it compute $H(bcd)$, $H(cda)$, $H(daa)$... and so on, then S is compared to the hash values generated from the paragraph text if a match is found then the string is compared to the subset of the paragraph generated from that hash value, and that is done until a match is found or all hashes are compared.

Table 3.1. Pseudocode of Karp-Rabin algorithm [39]

```
public int KarpRabin(String t, String p) {
    int n = t.length();
    int m = p.length();
    int hpatt = hash(p);
    int htxt = hash(t[0..m-1]);
    for(int i = 0; i < n; i++) {
        if (htxt == hpatt) {
            if (t[i..i+m-1] == p) { return i; }
            htxt = hash(t[i+1..i+m]);
        }
    }
    System.out.println("not found!");
    return -1;
}
```

3.2.10.2. Detecting plagiarism using N-gram

From the Karp-Rabin algorithm for pattern matching, and the idea of N-gram method for detecting plagiarism, we suggest the following method for detecting plagiarism:

Step 1: Read the text of the document.

- Step 2: Define a value N to be the size of the chunk used.
- Step 4: Define a word to be any sequence of characters separated by a space.
- Step 3: Break the text of the document into M tokens of size N.
- Step 4: Generate a hash value for each token generated and save all hashes.
- Step 5: Do all previous steps for all documents that are to be checked against the input document.
- Step 6: Initially there is no hit in the hits list for the hashes.
- Step 7: For all hashes of the input document:
 Search for a match in all checking documents.
- Step 8: Set a hit values for all matched hashes.
- Step 9: apply filters on the hit list.
- Step 10: From the hit list generate a list represents plagiarized words.

3.2.10.1. Example on detecting plagiarism

To illustrate the previous steps we provide in this section this simple example that will explain how the plagiarism detection is done.

Suppose we have the following original paragraph:

Palestine polytechnic university is the best university in Palestine at the field of engineering and technology; it provides the Palestinian society every year with a number of highly qualified engineers who apply their knowledge and experience to make Palestine a better place to live.

And some changes were made to this paragraph as the following:

Palestine polytechnic university is the worst university in Palestine at the field of engineering and technology; it does not provide the Palestinian society every year with a number of highly qualified engineers, they don't care about applying their knowledge and experience to make Palestine a better place to live. To be honest studying engineering is a waste of time.

Red words represent modification of the original paragraph; other parts represent parts of the original text that was simply added using "cut and paste".

Suppose that the chunk size is 3 (tri-gram):

First the original text is broken into the chunks with lower case letters see table 3.2.

Table 3.2. Tri-grams generation

	1	2	3	4	5
1	palestine polytechnic university	polytechnic university is	university is the	is the best	the best university
2	best university in	university in palestine	in palestine at	palestine at the	at the field
3	the field of	field of engineering	of engineering and	engineering and technology;	and technology; it
4	technology; it provides	it provides the	provides the Palestinian	the Palestinian society	Palestinian society every
5	society every year	every year with	year with a	with a number	a number of
6	number of highly	of highly qualified	highly qualified engineers	qualified engineers who	engineers who apply
7	who apply their	apply their knowledge	their knowledge and	knowledge and experience	and experience to
8	experience to make	to make palestine	make palestine a	palestine a better	a better place
9	better place to	place to live.			

For each gram a hash value is computed using the java.lang.String class, for instance, a hash code is computed according to the formula [41]:

$$s[0] * 31^{n-1} + s[1] * 31^{n-2} + \dots + s[n-1]$$

Using Java int (32-bit) arithmetic, where $s[i]$ is the i th character of the string, and n is the length of the string.

Then the hash code is converted to hexadecimal and all hashes are saved in a file that represents the signature of the respective text see table 3.3.

Table 3.3. Hashes generated for tri-grams (original text).

	1	2	3	4	5
1	5bd9c26d	a3b4e49e	837ef373	1148e6d7	d0c842e5
2	74144c5	9933d594	f27548e7	c0ab9971	b272b302
3	c6449814	e2f073f0	5f09913f	c2d95e67	e1a1f91b
4	56aab78a	707e7908	be48edb7	876366c1	3e0ce957
5	2b22dcc6	19fe989c	44696c96	7424e5e	6f6e48f1
6	cf31f33f	377670c	d4635391	b74c26e2	d02866d0
7	18eb8c1a	756d545a	5d5c1b11	ff20578b	80a3e6b8
8	5dc95003	f6989b98	fe6e57dc	b8bbda06	eb4eaaaa
9	841d703c	cf06f50a			

The same previous process is done on the second text (the text to be checked), that is tri-grams are generated from the text using the same technique, and hashes are generated from the try-grams see table 3.4.

Table 3.4. Hashes generated for tri-grams (modified text).

	1	2	3	4	5	6	7
1	5bd9c26d	a3b4e49e	837ef373	3c36364a	b477793e	208b5d4e	9933d594
2	f27548e7	c0ab9971	b272b302	c6449814	e2f073f0	5f09913f	c2d95e67
3	e1a1f91b	983719e3	c313183f	e7a4f293	ee41001b	be59b8e8	876366c1
4	3e0ce957	2b22dcc6	19fe989c	44696c96	7424e5e	6f6e48f1	cf31f33f
5	377670c	b8072023	fbda1ca4	5ce42582	cf9ca695	2ccd9900	687710a
6	25f57b5f	eee1b6d4	5d5c1b11	ff20578b	80a3e6b8	5dc95003	f6989b98
7	fe6e57dc	b8bbda06	eb4eeaaa	841d703c	cf06f50a	ce221d42	7cf4de36
8	466e7409	1f819d03	eeaf5533	5d83a502	8f4f7748	21100b1b	caf5be24
9	48717c56						

Then a search is done on the hashes of the second text, and a hit list is generated as the following:

- If the hash value is found put 1 at the respective position of the hit list.
- If it is not found put 0.

Initially the hit list contains:

{0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0}.

Each 0 represent a non-found hash value.

After searching for all values the list will be :

{1110001 1111111 1000001 1111111 1100000 0111111 1111100 0000000 0}.

Two filters are applied on the hit list:

The first filter is used to remove small hits from a text according to a threshold value, for example if only three chunks were found and the resulted hit list was:

{0000000 0000000 0001000 0000000 0000000 1000000 0010000 0000000 0}.

The filter will remove the three ones and the hit list will return like this:

{0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0}.

The second filter is applied to remove some noisy hits; for example if the hit list was:

{0000000 111111 111111 0000000 0000000 0000001 0000001 0000000 0}.

The first filter will not remove the lonely ones, that's why we need the second filter.

Back to the example above, both filters will keep the hit list as it is,

{1110001 1111111 1000001 1111111 1100000 0111111 1111100 0000000 0}.

Then the hit list is converted to words list, that is a list representing the place of plagiarized words,

{1111101 1111111 1110001 1111111 1111000 0111111 1111111 0000000 000}.

3.3. System Modeling

3.3.1. Data flow diagrams

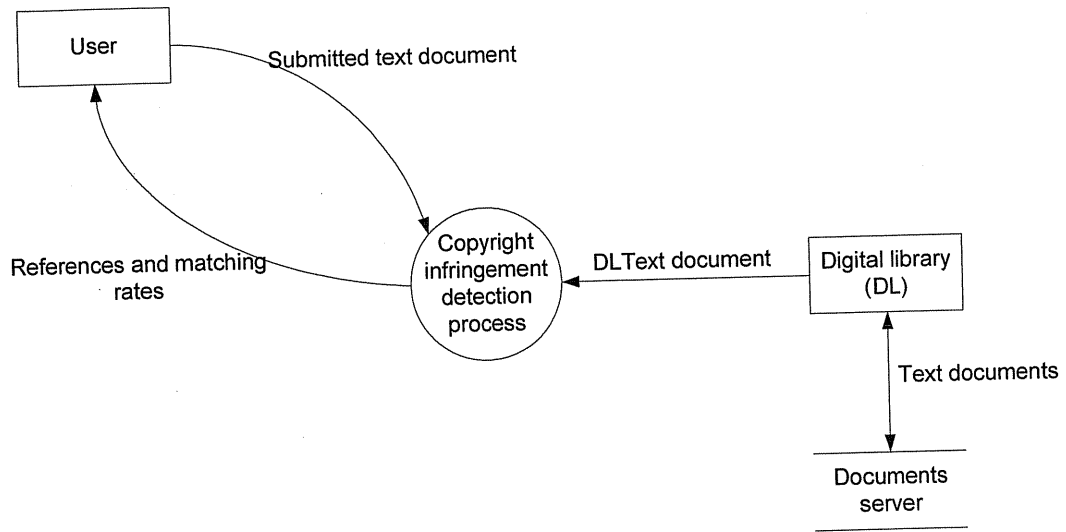


Figure 3.11. DFD level 0.

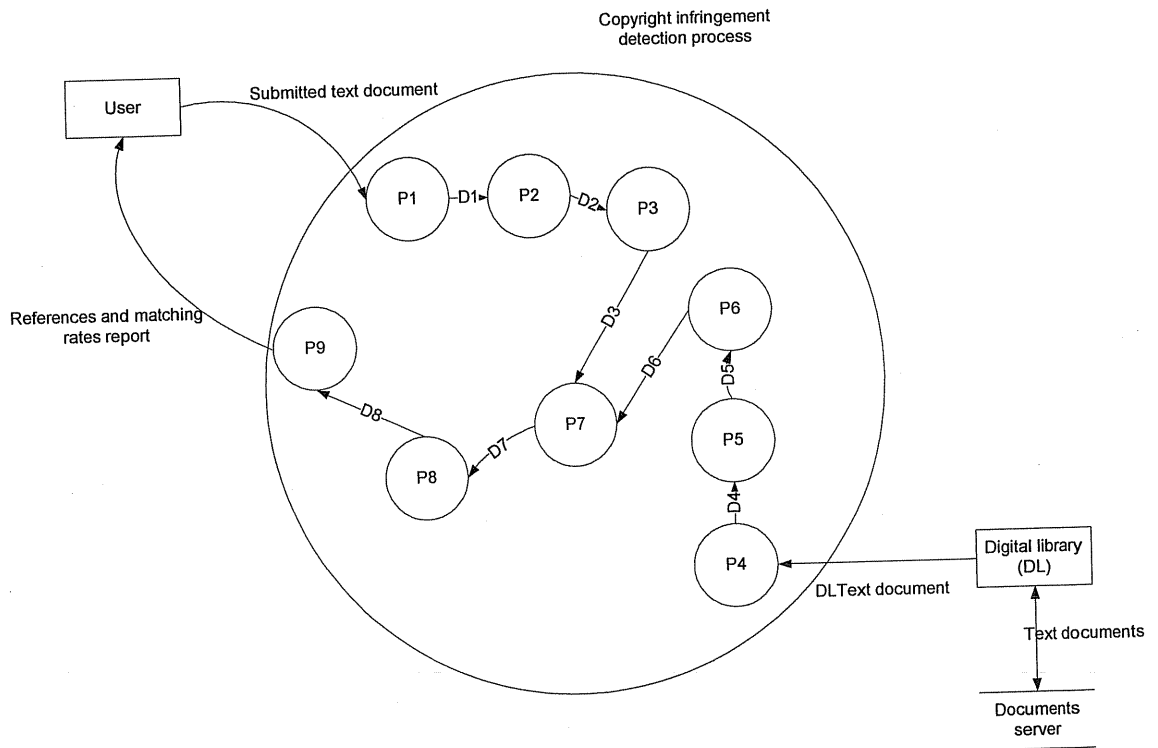


Figure 3.12. DFD level 1

Table 3.5. DFD level 1 processes

Process ID	Process description
P1	Check input document.
P2	Analyze checked document.
P3	Generate fingerprint.
P4	Check DL document.
P5	Analyze checked document.
P6	Generate finger.
P7	Matching process.
P8	Analyze results.
P9	References and matching rates report generation.

Table 3.6. DFD level 1 data

Data ID	Data type
D1	Checked document.
D2	Analyzed document.
D3	Fingerprint of the document.
D4	Checked DL document.
D5	Analyzed DL document.
D6	Fingerprint of DL document.
D7	Analyzed result.
D8	References and matching rate report.

3.3.2. Use cases

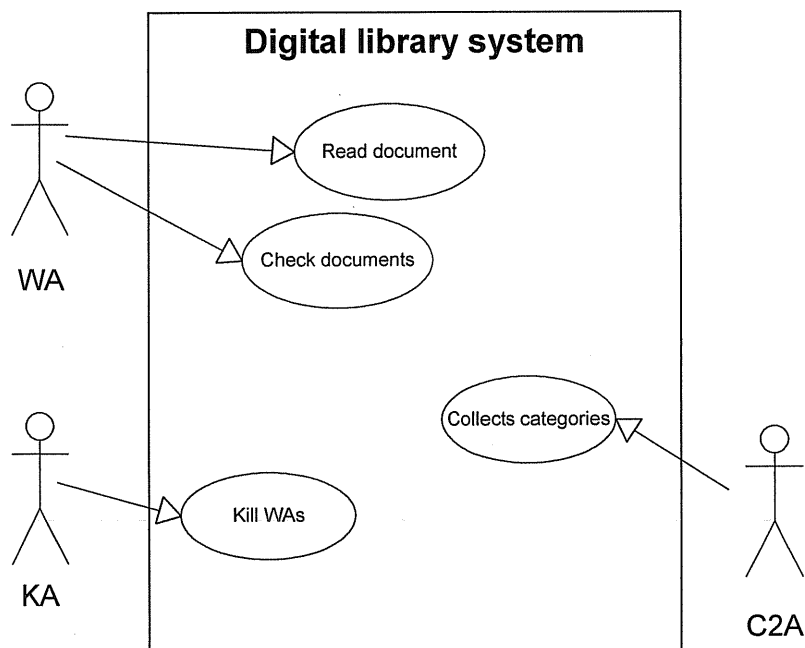


Figure 3.13. Use cases at the digital library system

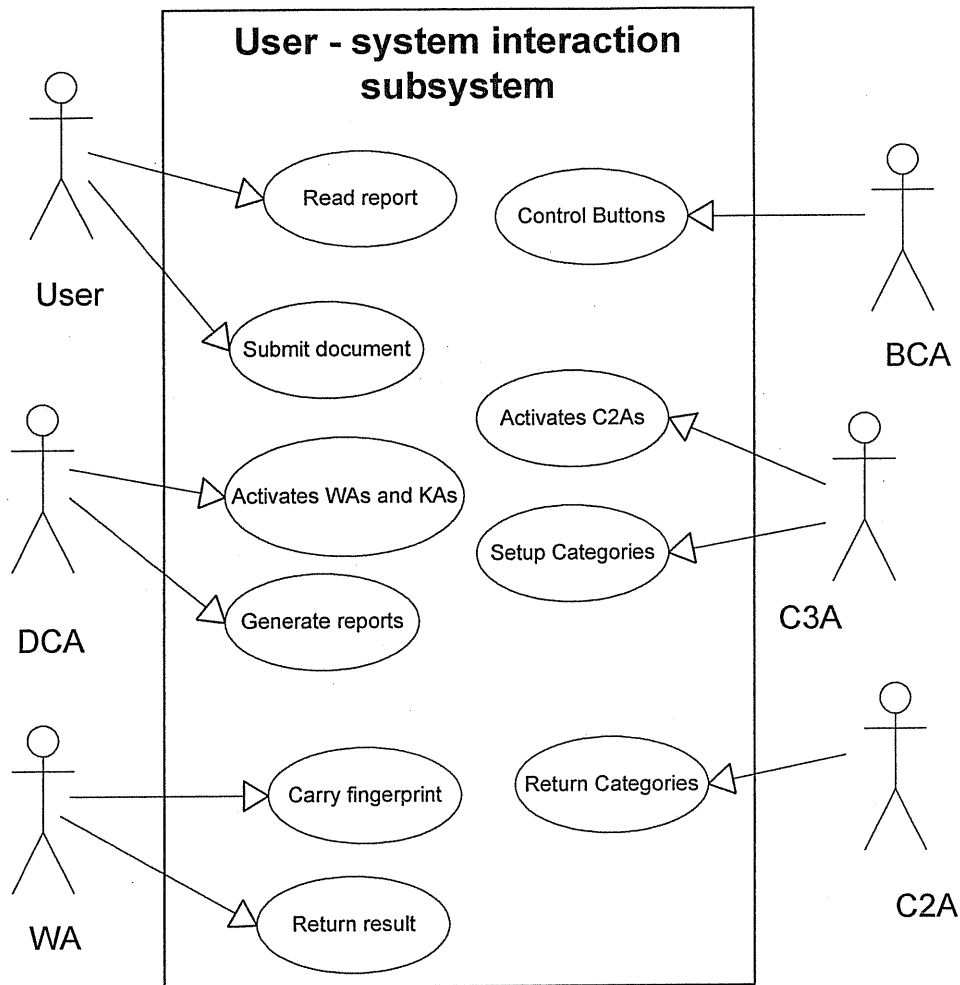


Figure 3.14. Use cases at user-system interaction subsystem

3.3.3. Sequence diagram

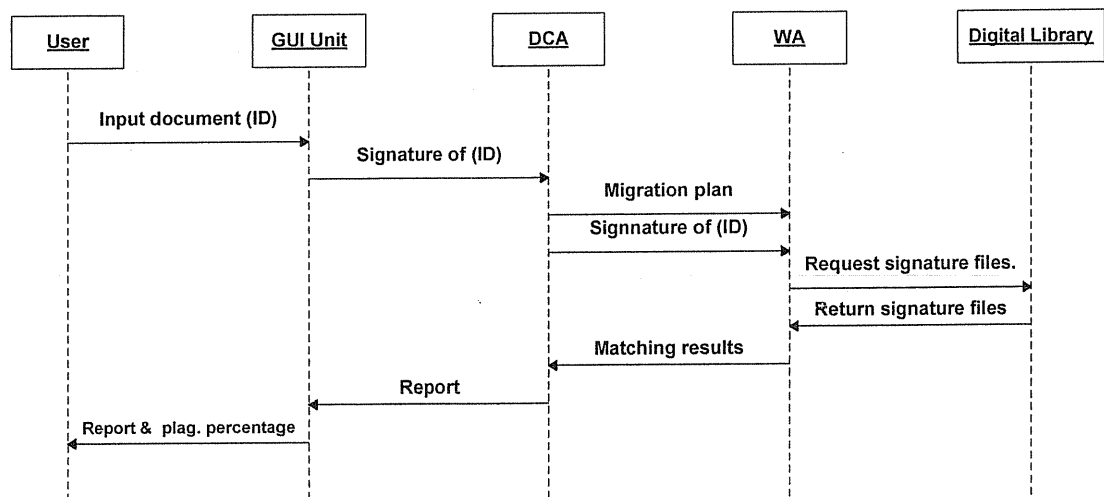


Figure 3.15. Sequence diagram describes the plagiarism detection operation.

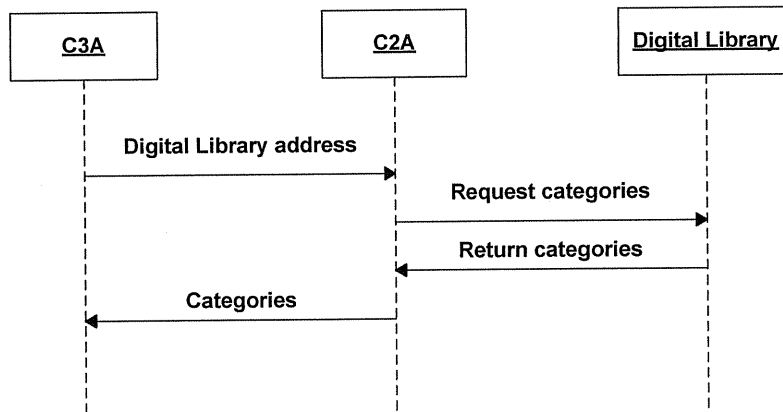


Figure 3.16. Sequence diagram describes collecting the categories.

3.4. Summary

In this chapter we described conceptual design issues, architectural design, system modeling, and detailed description of the project objectives.

CHAPTER 4

DETAILED SYSTEM DESIGN

- 4.1. Architecture of PLADMA**
- 4.2. Detailed Specifications of PLADMA Components**
- 4.3. Agent's Tasks and Interactions in JADE**
- 4.4. Detailed System Modeling**
- 4.5. GUI Design**
- 4.6. Fault Tolerance Module**
- 4.7. Security Module**
- 4.8. Limitations**
- 4.9. Assumptions**
- 4.10. Summary**

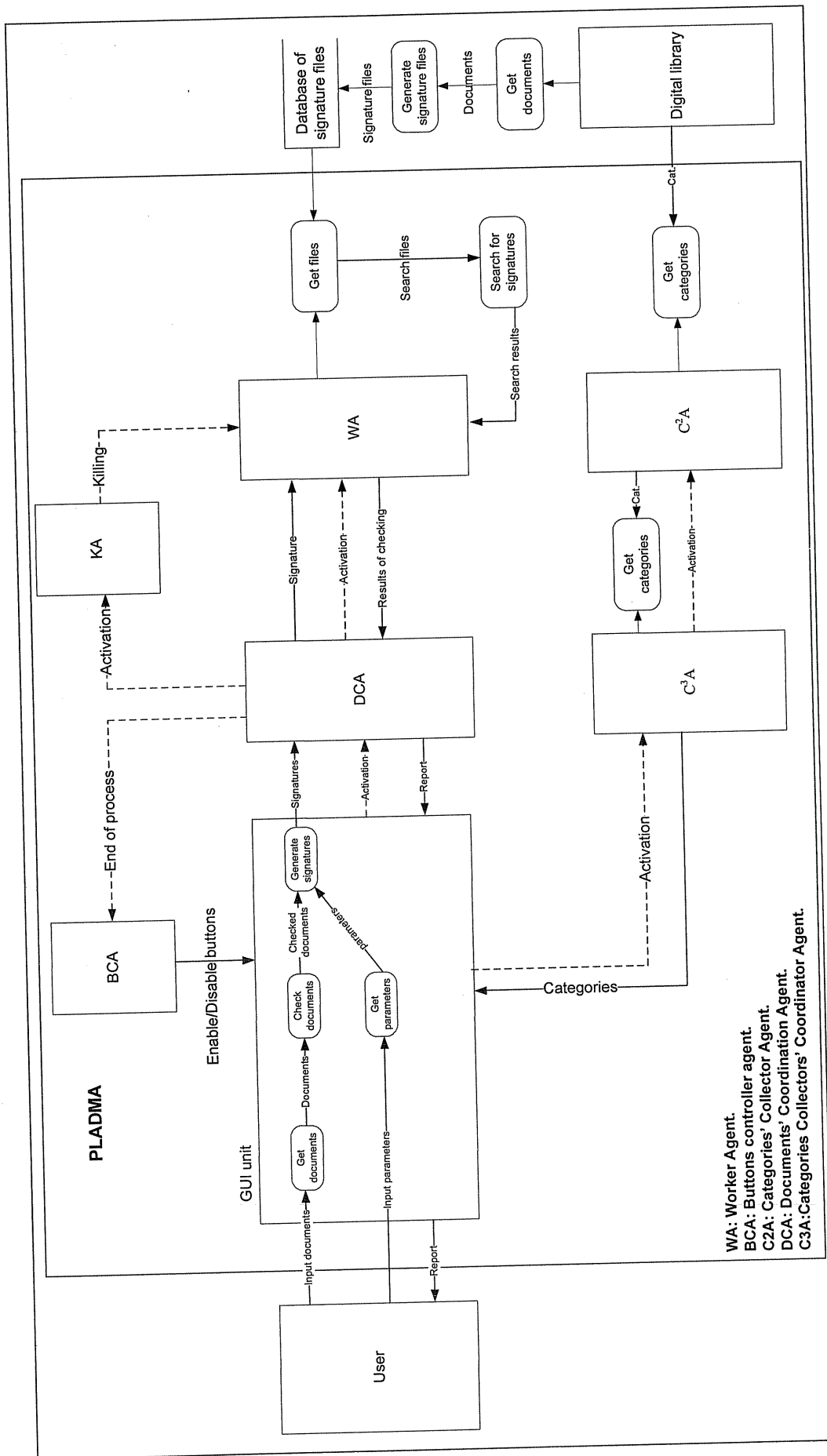


Figure 4.1: System architecture.

4.2. Detailed specifications of PLADMA components

4.2.1. GUI unit:

- A button for the user titled "Open document" that allows the user to select a folder of documents to have all its documents checked, where the document either (.doc) document or (.txt) document.
- A drop down menu that allows the user to select the type of the document such as scientific, physics, historical... etc.
- A Button titled "Start" that allows the user to start the "searching and checking" operation.
- A check box, that let the user determine where to use fault tolerance or not while the agent is moving from container to container.
- An option to change the chunk's size to be used.
- 3 options for the user to choose the different method that to be used while searching, also an option to select the number of agents to be used while searching.
- The home side is responsible for making the signatures for the selected documents.
- The home side starts a "receiver agent" for each document this agent is provided with the signature of the document and other information related to the document and the signature, it is – the receiver agent- to be started when the user clicks on "Start" button.
- A drop down menu that will show the result of searching of different agents.

4.2.2. Documents' Coordinator Agent (DCA)

- Only one DCA is started for each document.
- It starts a number of WAs according to the scenario used in searching.
- It waits for messages containing the result of checking from searching agents.

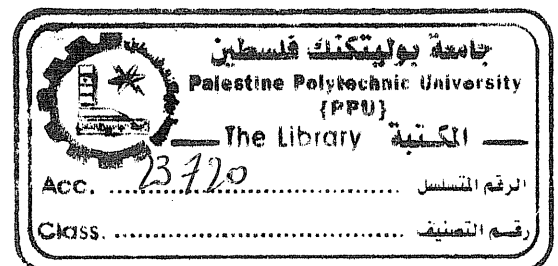
- When a message is received with more than a threshold of percentage of plagiarism the receiver agent generate a report indication the places of the document where plagiarism was committed, and the percentage of plagiarism.
- If any message of results appears to have more than a threshold value of percentage of plagiarized text, or the last message was received, then the receiver agent sends a message to Agent Management System (AMS) asking for available location, and then starts "killer agents" to kill all other agents that are still working on the document on different locations.

4.2.3. Worker Agent (WA)

- As one of the group of worker agents; each worker agent is responsible for checking for matches for the signatures in the signature file it holds.
- According to the scenario indicated, the searching is done; the searching agent goes to specific digital library or a series of digital libraries, doing the checking process on each one, either on all the signature files or a fraction of them.
- If the searching agent found more than a threshold value of percentage of plagiarized text or it finished searching in all digital libraries it is aimed to, then it sends a message to the "receiver agent" that indicates which chunks are found and the location of the file where they found, and then kills itself.

4.2.4. Killer Agent (KA)

- Depending on the number of locations on the platform, a number of killer agents are started by the receiver agent, in order to kill agents that are still working (in case of finishing the checking process by any agent).
- When arriving to a location it sends the AMS a message asking for agents available on that location, and then kills the agents working on that specific document specified by the receiver agent.



4.2.5. Categories' Collector Agent (C²A)

- One categories' collector agent is sent to each digital library to collect all available categories at that digital library.
- Categories' collector agent sends a message to the categories' coordinator agent containing the categories it found on a specific digital library.

4.2.6. Categories Collectors' Coordinator Agent (C³A)

- This agent is responsible for starting categories' collector agents providing each of them the location where it should go i.e. the digital library it should go to and collect the categories it has.
- It waits for messages from categories' collector agents that contain the categories available in the respective digital library.
- The time this agent receives a collection of categories it adds them to the categories list for the user.

4.2.7. Buttons' Controller Agent (BCA)

- This agent disables some buttons when the searching process begins.
- It also waits for all DCA's to complete their jobs.
- It enables the disabled buttons and gives the user a note that the whole process was finished when all DCA's finish.

4.2.8. Specification of a digital library

- It should have folders having the names of different books categories available at the digital library.
- All folders of categories should be inside one folder named "Digital Library".
- Each folder of category contains files of signatures for each book/document available in the digital library that is considered to be under that category.
- Each file of signature is named by the name of the book/document it represents.

- Signature files are constructed in the same way as the signature file of the check document is constructed.
- Chunk size used in constructing the signature files should be the same as the chunk size used in constructing the signature file of the check document.
- The container where the digital library resides should be started authentically using a username and a password.
- A digital library should have all its files of signatures on one container.

4.3. Agents Tasks Scheduling and Interactions in JADE

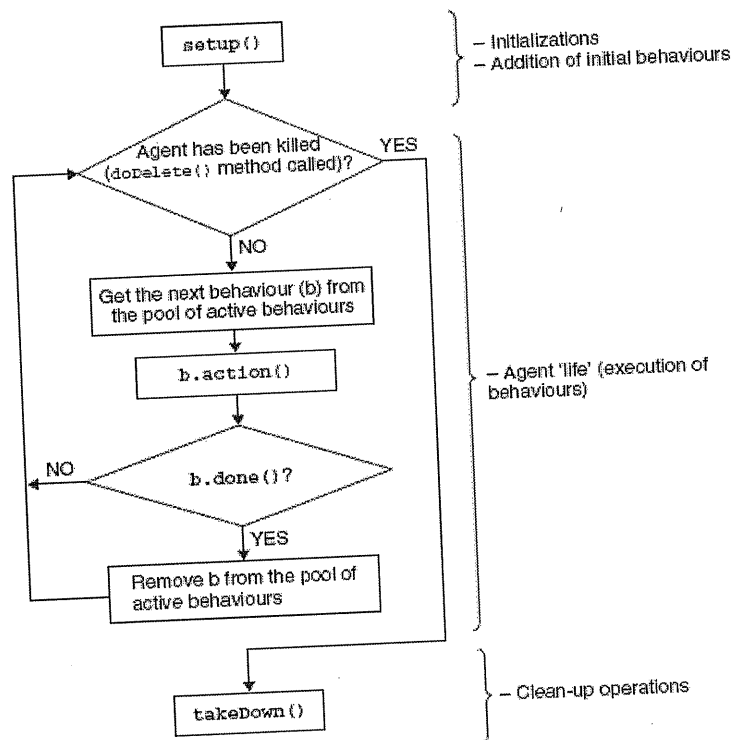


Figure 4.2. Agent thread path of execution [4].

4.3.1. Agent tasks in JADE

The tasks that the agent will do are carried out within "behaviors"; each behavior is implemented as an object of a class extends `jade.core.behaviours.Behaviour`. To make the agent starts a task an object that is implemented by a behavior, the behavior object should be added to the agent using the `addBehaviour()` method.

Behaviors should implement the abstract method `action()` that contains the operations the agent is to perform, and the `done()` method that return true if the behavior completed and false if it didn't.

4.3.2. Primary types of behaviors

There are three primary types of behaviors in JADE, which are:

1. One-shot behaviors, this type of behaviors are designed to have one execution phase, that is their `action()` method executes only once. The `jade.core.behaviours.OneShotBehaviour` class already implements the `done()` method by returning true that is the behavior is done when the execution of `action()` method is done.

For example an implementation of a one-shot behavior class can be done using:

```
public class MyOneShotBehaviour extends OneShotBehaviour {
    public void action() {
        // perform operation X
    }
}
```

Here X is performed only once.

2. Cyclic behaviors, this type of behaviors are designed never to complete, that is the `action()` method executes the same operation every time it is called, the `done()` method always returns false.

For example an implementation of a cyclic behavior class can be done using:

```
public class MyCyclicBehaviour extends CyclicBehaviour {
    public void action() {
        // perform operation Y
    }
}
```

Here Y executes repeatedly until the agent that executing the behavior terminates.

3. Generic behaviors, see [4].

4.3.3. Scheduling operations

The package `jade.core.behaviours` implements classes that do some operation on a selected point of time.

- `WakerBehaviour` has a method `onWake()` that executes after a given period of time.

Example:

```
public class MyAgent extends Agent {
protected void setup() {
System.out.println("Adding waker behaviour");
addBehaviour(new WakerBehaviour(this, 10000) {
protected void onWake() {
// perform operation X
}
} );
}
}
```

Here the operation X is performed after 10 seconds from adding the behavior.

- `TickerBehaviour` has a method `onTick()` that executes repeatedly after a given period of time.

Example:

```
public class MyAgent extends Agent {
protected void setup() {
addBehaviour(new TickerBehaviour(this, 10000) {
protected void onTick() {
// perform operation Y
}
} );
}
}
```

Here operation Y is performed repeatedly every 10 seconds.

4.3.4. Methods invoked during the agent's life cycle

There are different methods in JADE that are used to make the agent do different tasks at different point during its life cycle, these methods are:

- `setup()`: The `setup()` method is used to perform the agent initializations, it is the first method invoked when the agent is activated.
- `afterMove()`: This method is invoked after an agent moves to a container other than its current container.

- `afterClone()`: This method is invoked after an agent is cloned; the method is called at the cloned agent, not the original one.
- `takedown()`: This method is invoked after the agent is terminated.

Other methods are `beforeMove()`, `beforeClone()`... etc. see [4].

4.3.5. Interacting with AMS

Since some agents in this system needs to do operations on the platform level, such as (DCA) and (C3A) needs to know the locations available on the platform, and (KA) needs to know what agents are presented on a container, then there is a need to make some interactions with the AMS, according to FIPA specifications, the AMS does not have to be an agent, but since it sends and receives messages to/from other agents, it's implemented as an agent on JADE.

The AMS performs management duties related to the platform, such as killing an creating agents and containers, it also stores (white pages) contains information about the platform and the agents, where an agents can request from AMS the currently available containers on the platform, or the available agents on the container where the agent resides[4].

Any agent that wishes to perform platform management actions must first request the AMS to perform them.

4.4. Detailed System Modeling

In this section detailed functional description of different system components, and the functions contained within each component, also a life cycle of each agent is described using Agent Unified Modeling Language (AUML) [42].

4.4.1. GUI unit

For easy interaction with the user, this unit is communication link between the user and the system.

Name	Check Documents.
Input	List of documents names.
Output	Boolean value of documents status.
Description	this function gets a list of documents names from the user and checks for any inconvenient documents, only (.txt) and (.doc) documents are accepted, if there is any other types the user will be inform with an error message about this, also directories are not allowed to be one of the list of the documents.

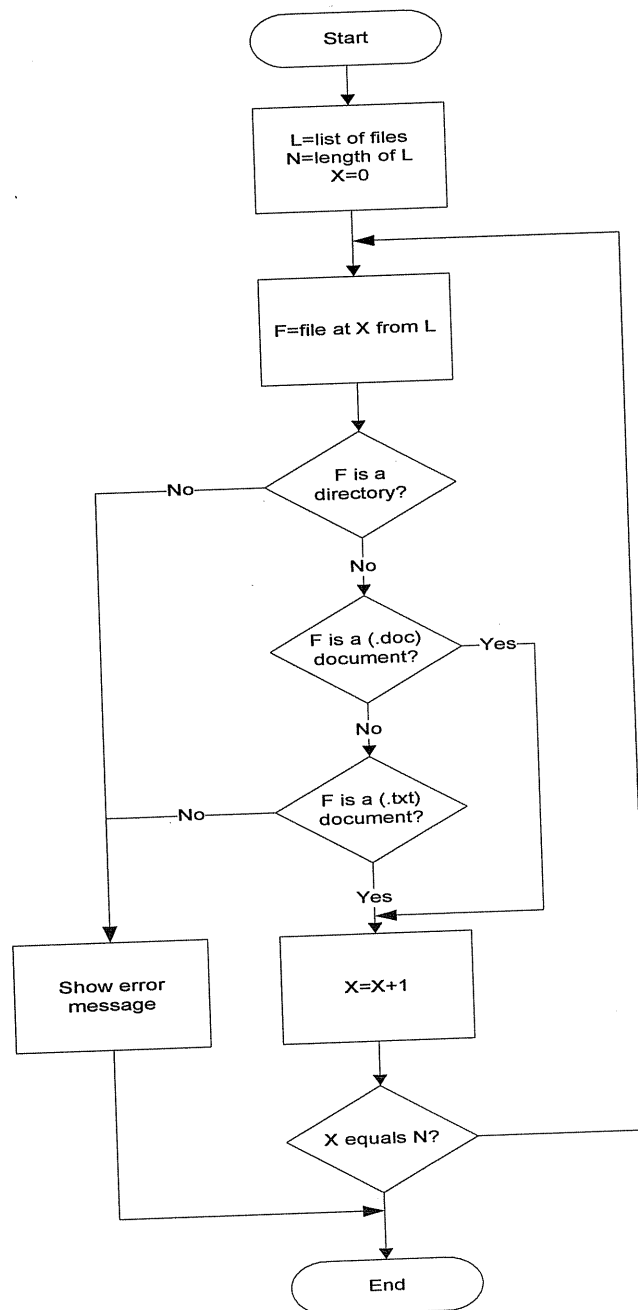


Figure 4.3. Flowchart of Check Documents.

Name	Signature Generator.
Input	Text data.
Output	Signature of the text data.
Description	This function generates a signature for a document.

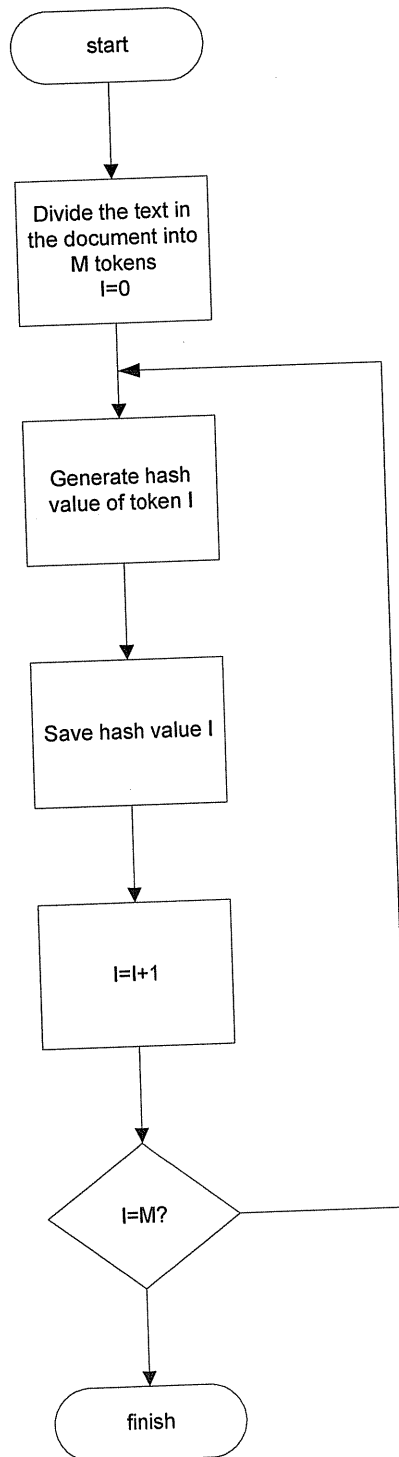


Figure 4.4. Flow chart of Signature Generator.

Name	Generate signatures of files.
Input	Collection of files.
Output	Collection of signature files.
Description	This function generates signature files for input documents.

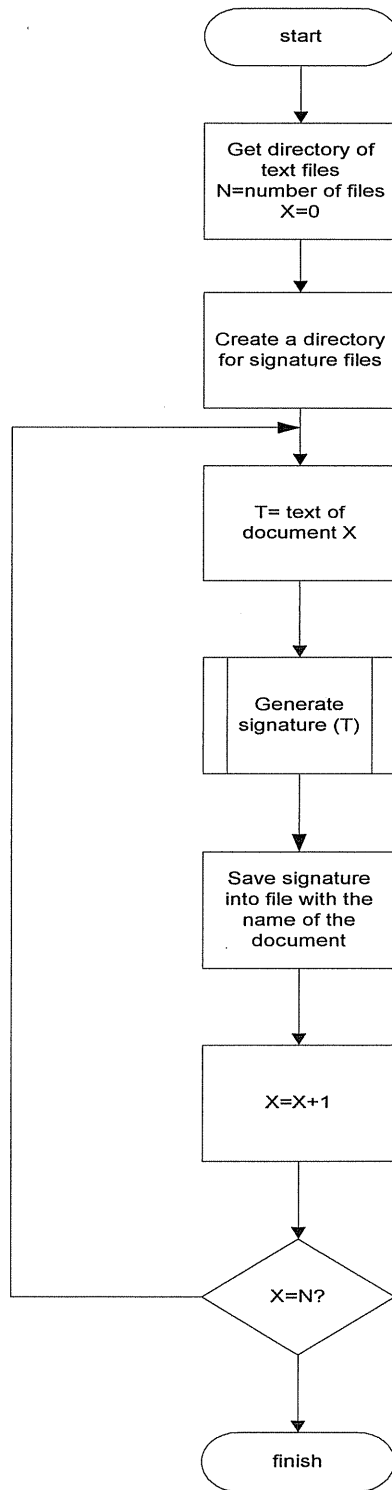


Figure 4.5. Flowchart of Generate signatures of files.

Name	Set number of agents.
Input	Integer number.
Output	Setting the number of agents.
Description	This function takes an integer number from the user and checks the input value, then sets the number of WA's as that number, or informs the user with an error message if there is a problem with the input.

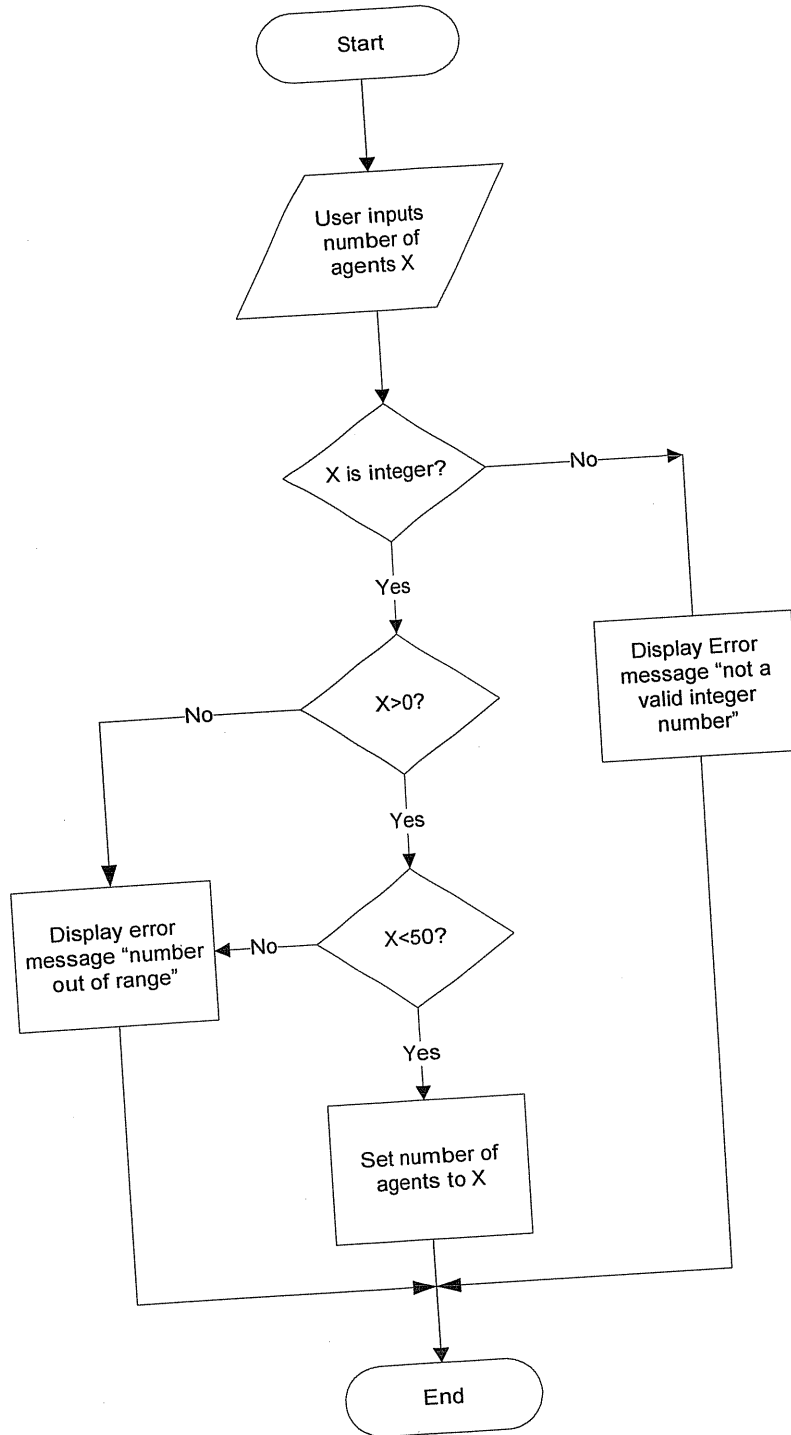


Figure 4.6. Flow chart of Set number of agents.

Name	Locations.
Input	Request of location.
Output	Locations available on the platform.
Description	This function sends a message to AMS asking for available locations on the platform, the AMS then responds with the list of locations.

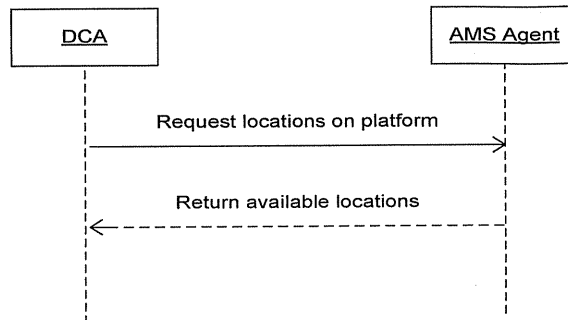


Figure 4.10. Sequence diagram of Locations.

Name	Set number of agents.
Input	Integer number.
Output	Setting the number of agents.
Description	This function takes an integer number from the user and checks the input value, then sets the number of WA's as that number, or informs the user with an error message if there is a problem with the input.

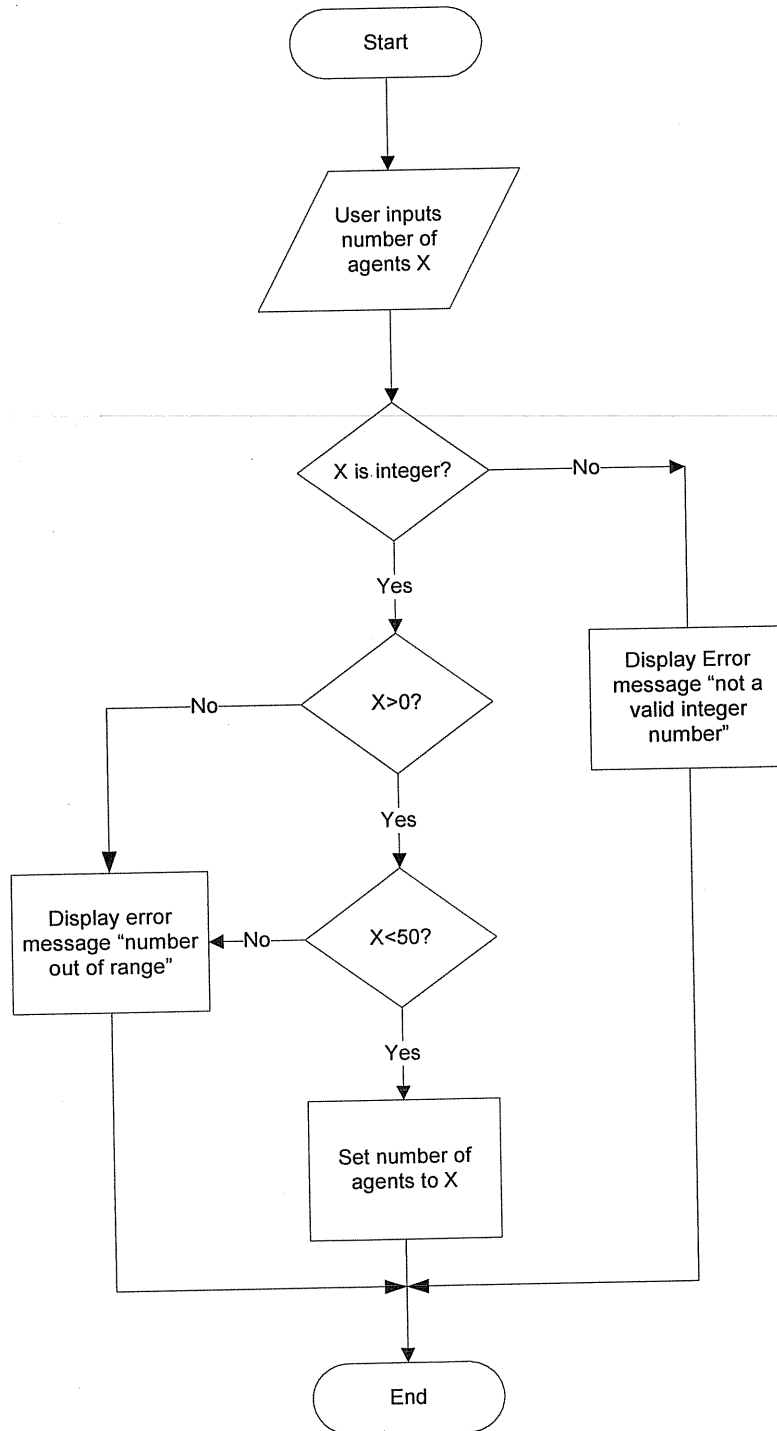


Figure 4.6. Flow chart of Set number of agents.

Name	Set the chunk size.
Input	Integer number.
Output	Setting of the chunk size.
Description	This function takes an integer number from the user and checks the input value, then sets the chunk size as that number, or informs the user with an error message if there is a problem with the input.

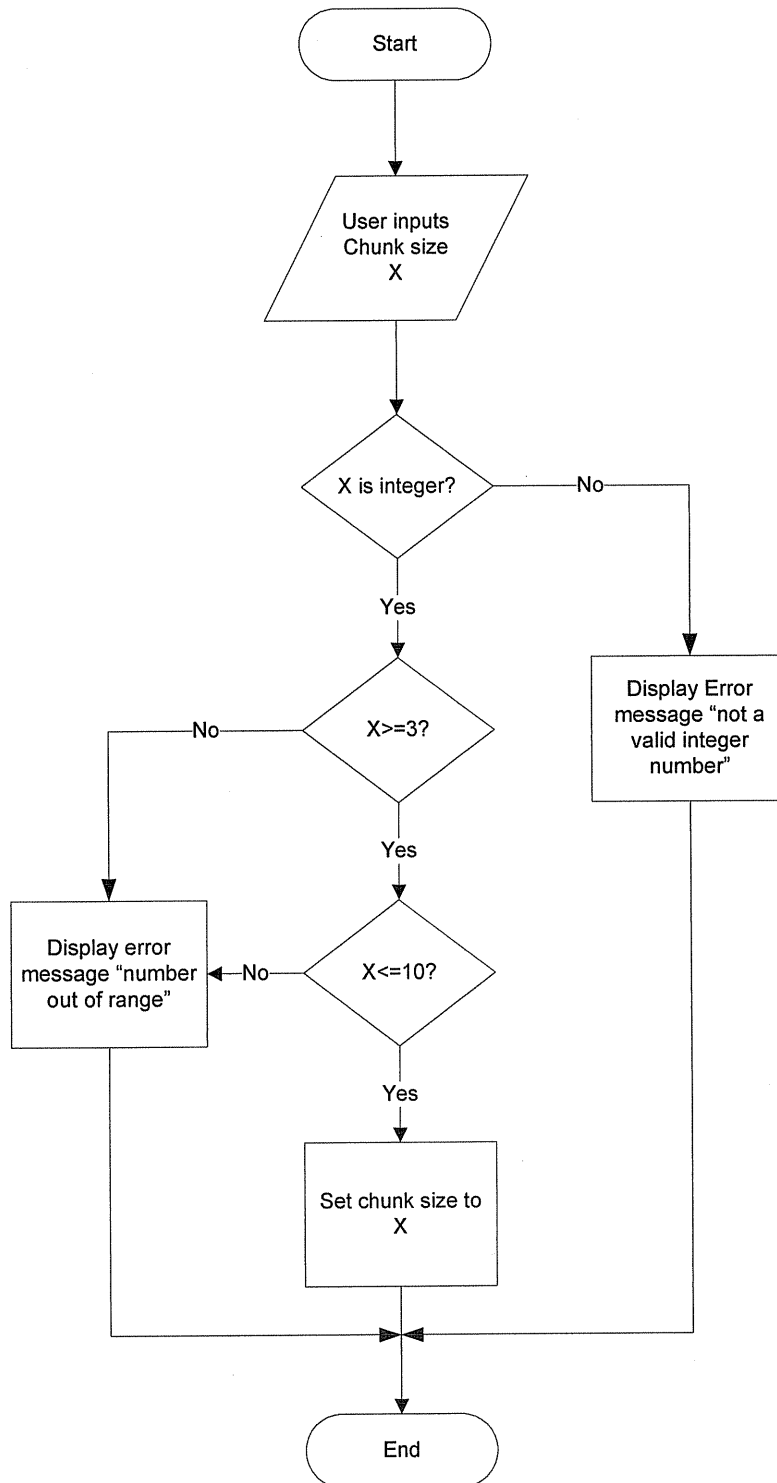


Figure 4.7. Flowchart of Set the chunk size.

4.4.2. Documents' coordinator agent -DCA

Documents' coordinator agent is responsible for coordinating other agents; it starts worker agents providing them with travel plan, and waits for messages of results from them. It also starts killer agents when it receives a message (from worker agent) with more than a threshold value of the percentage of plagiarized text.

It stays at home container, and does not move to any other container, when the checking finishes it kills itself.

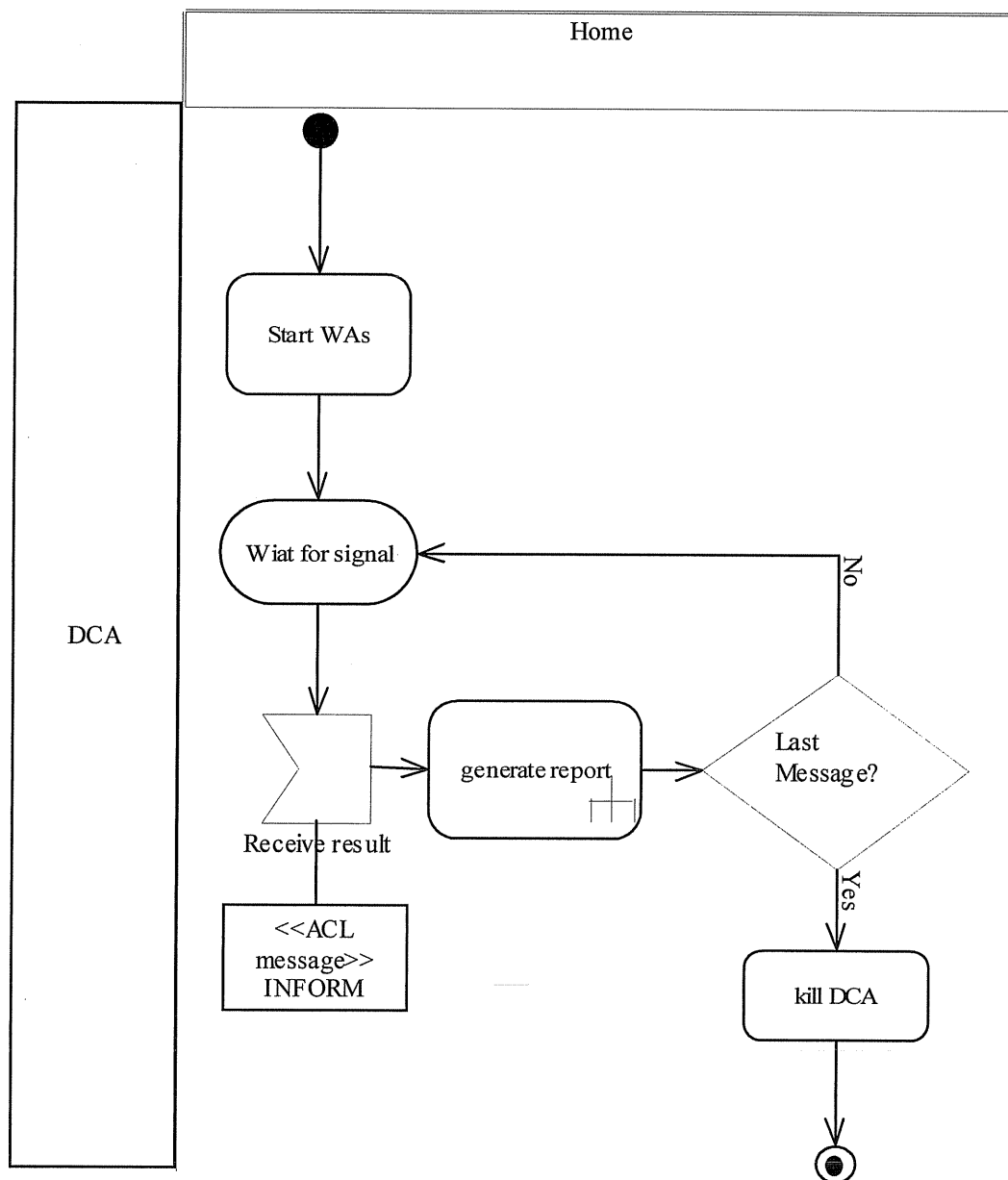


Figure 4.8. : DCA's life cycle.

Name	Setup.
Input	Integer value
Output	Collection of WA started according to the scenario indicated.
Description	This function overrides the function setup() in JADE, it does initialization for DCA, and starts a number of WA according to the scenario indicated by the user.

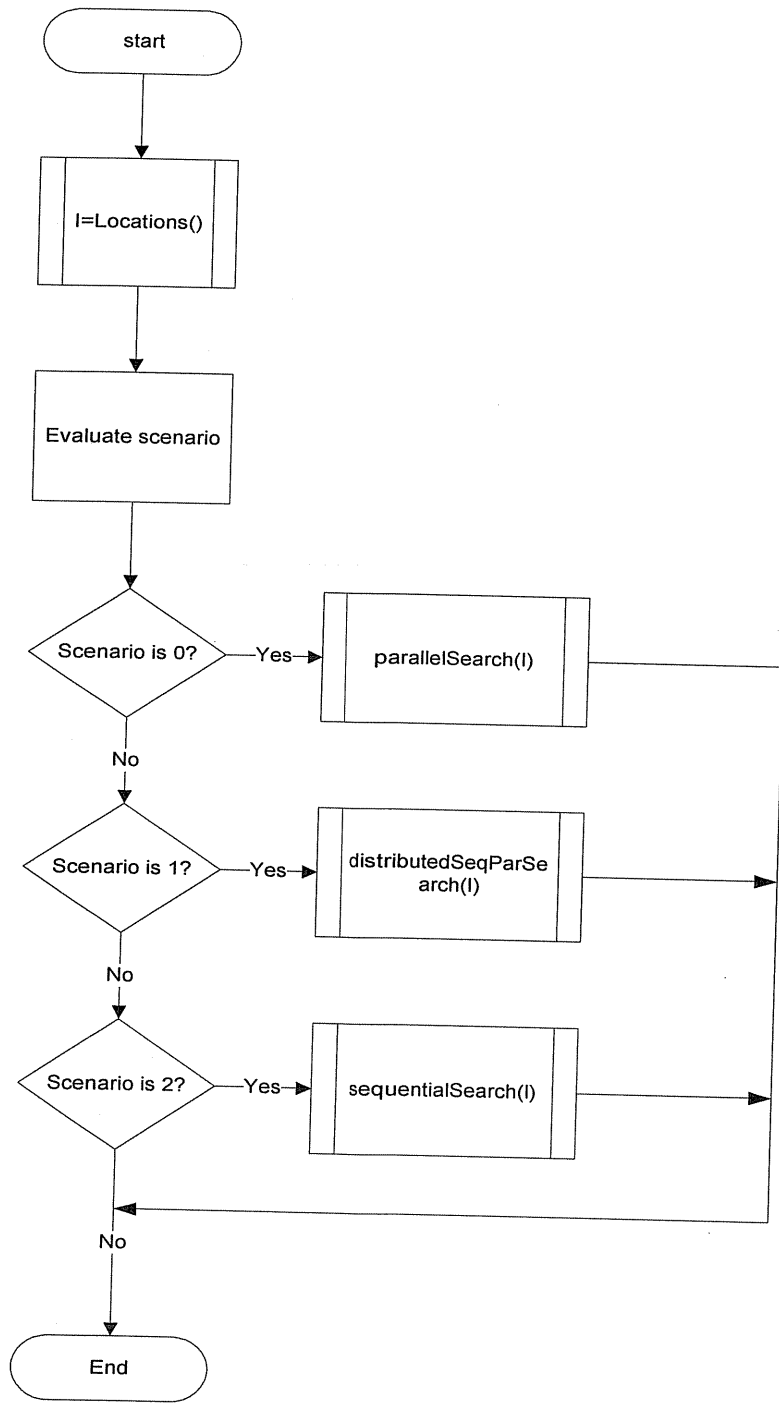


Figure 4.9. Flowchart of Setup.

Name	Parallel search.
Input	List of available locations.
Output	Collection of WA started according to the parallel search scenario.
Description	This function starts a collection of WA's provided with a container name that it should go to in order to do the searching.

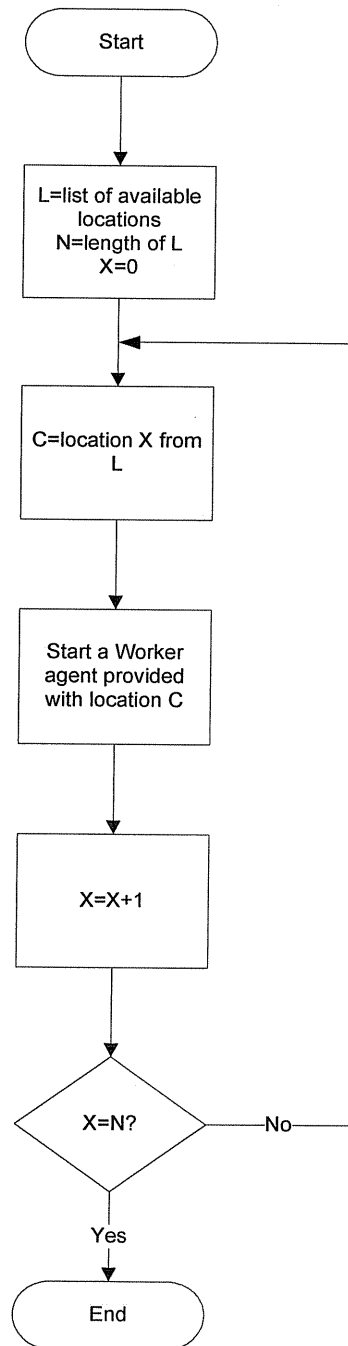


Figure 4.11. Flowchart of Parallel search.

Name	Sequential search.
Input	List of locations.
Output	Collection of WA started according to the sequential search scenario.
Description	This function starts a collection of WA's provided with a list of locations that each one should go to in order to do the searching.

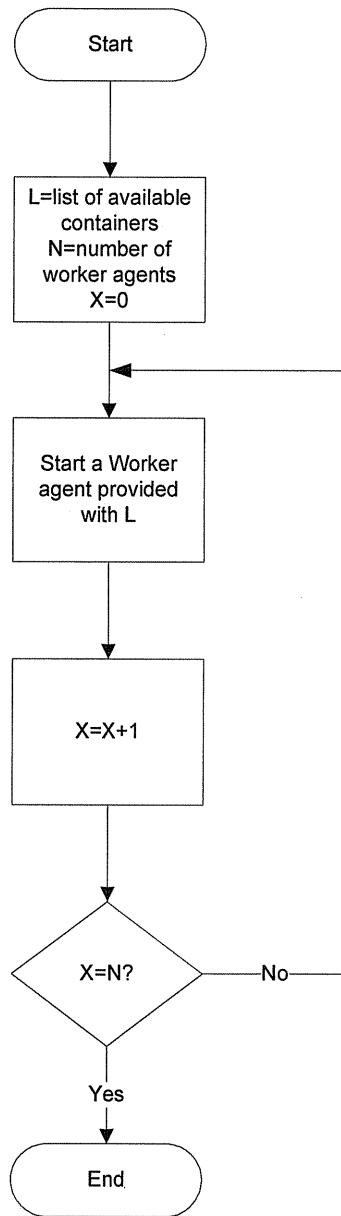


Figure 4.12. Flowchart of Sequential search.

Name	Distributed sequential and parallel search.
Input	List of locations.
Output	Collection of WA started according to the Distributed sequential and parallel search scenario.
Description	This function starts a collection of WA's provided with a list of locations that each one should go to in order to do the searching.

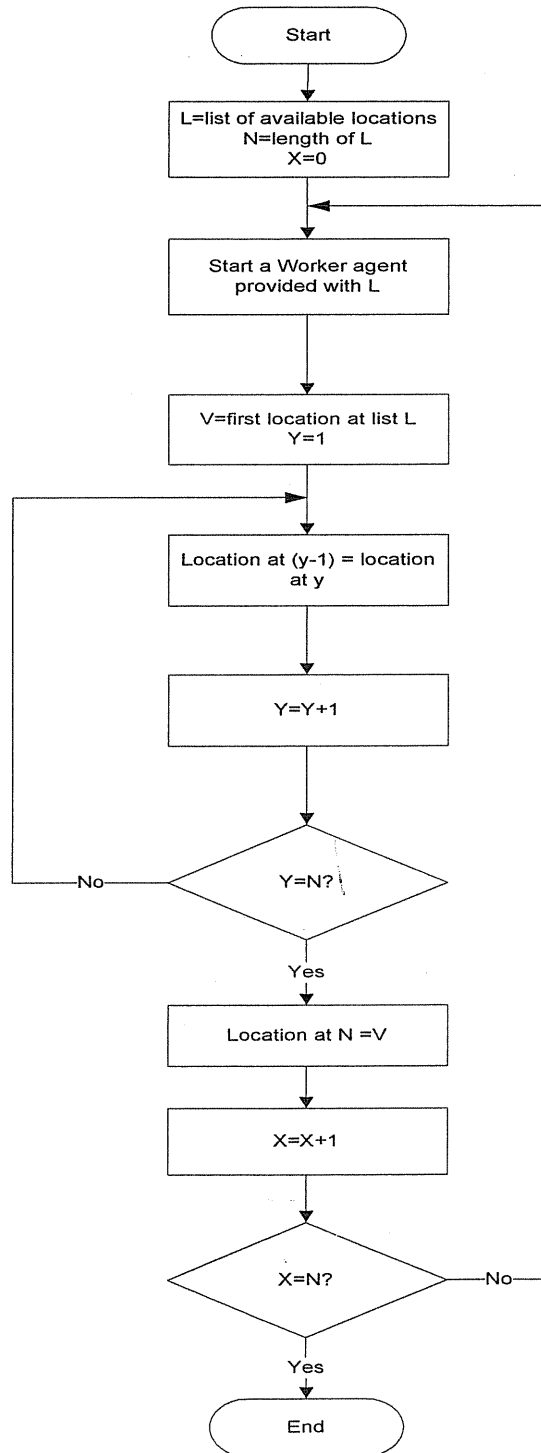


Figure 4.13. Flowchart of Distributed sequential and parallel search.

Name	Process results.
Input	Results as a String.
Output	Filtered and normalized array of hits positions and Hashtable of documents addresses.
Description	This function takes the resulting String from the checking process by a WA, and extracts the different information related to the checked document, and then normalization and filtering is done on the results.

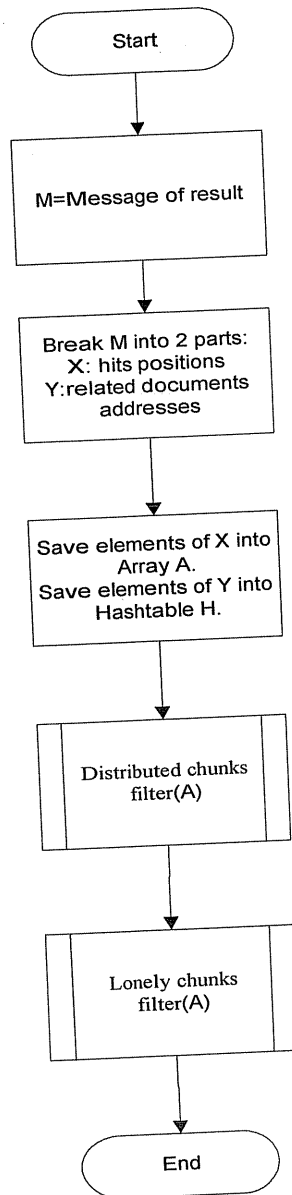


Figure 4.14. Flowchart of Process results.

Name	Process results.
Input	Results as a String.
Output	Filtered and normalized array of hits positions and Hashtable of documents addresses.
Description	This function takes the resulting String from the checking process by a WA, and extracts the different information related to the checked document, and then normalization and filtering is done on the results.

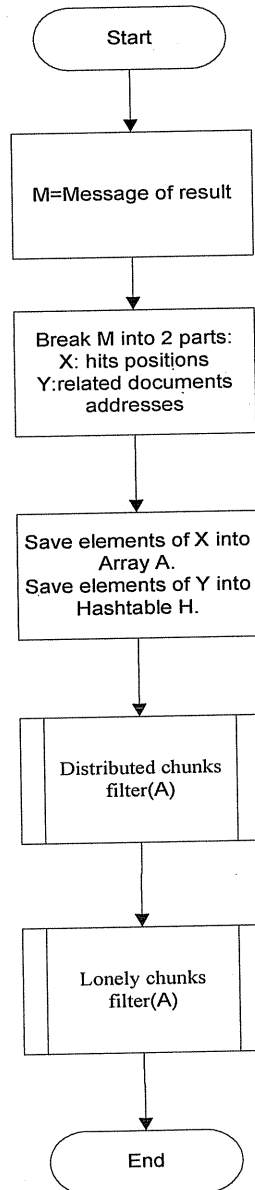


Figure 4.14. Flowchart of Process results.

Name	Distributed chunks filter
Input	Unfiltered, Non-normalized array of hits positions, and a threshold value.
Output	Filtered and normalized array of hits positions and Hashtable of documents addresses.
Description	This function filters the array of hits positions in order to remove false hits of chunks, also normalize the resulted array into smaller range of number.

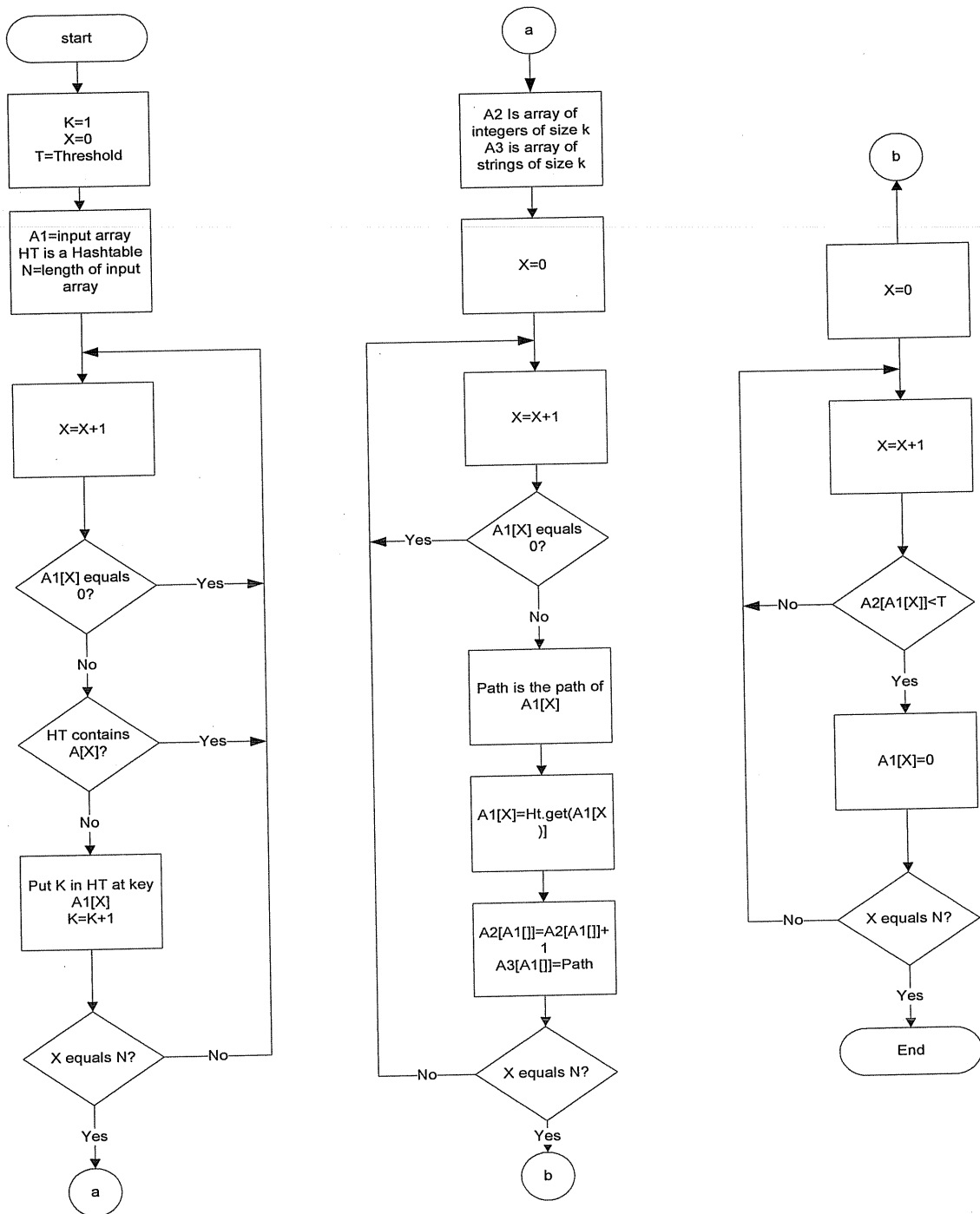


Figure 4.15. Flowchart of distributed chunks filter.

Name	Lonely chunks filter
Input	Normalized array of hits positions.
Output	Filtered array of hits positions using lonely chunks filter.
Description	This function filters the array of hits positions in order to remove noise added by recognized documents.

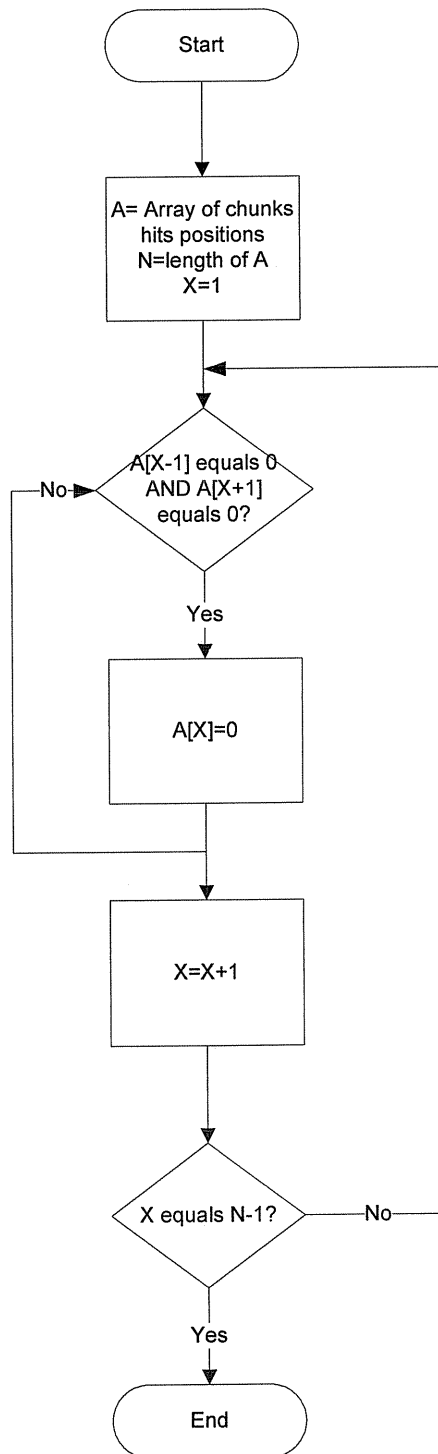


Figure 4.16. Flowchart of lonely chunks filter.

Name	Generate report.
Input	Final array of hits positions, chunk size.
Output	HTML report.
Description	This function generates an HTML report that shows the plagiarized parts of the document, and the percentage of plagiarism in the whole document.

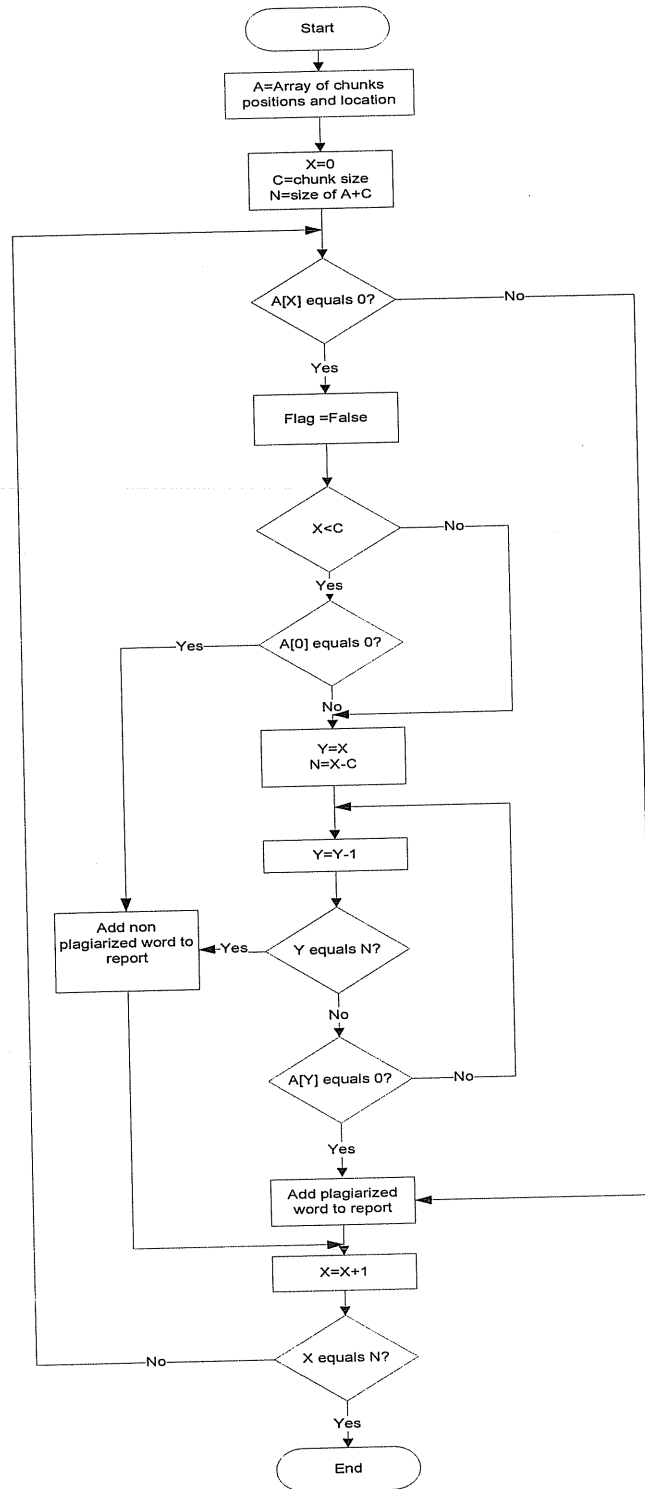


Figure 4.17. Flowchart of Generate report.

Name	Show results.
Input	Name of the report, percentage of plagiarism in the document, a drop down menu.
Output	An item added to the drop down menu.
Description	This function adds to the drop down menu of results, add to it items represents reports of plagiarism for the respective document.

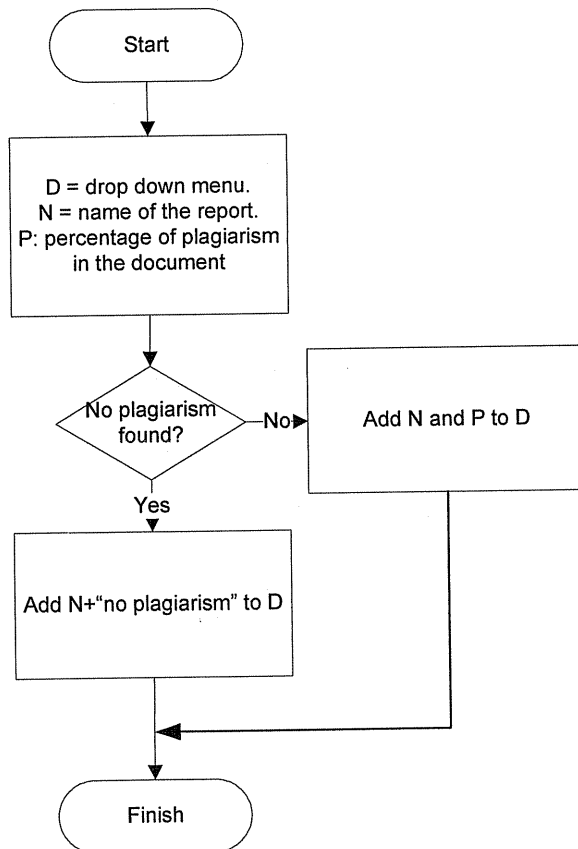


Figure 4.18. Flowchart of Show results.

Name	Send killers.
Input	List of locations.
Output	Sending of a collection of KA's.
Description	This function starts KA's, each one is provided with the location it should go to.

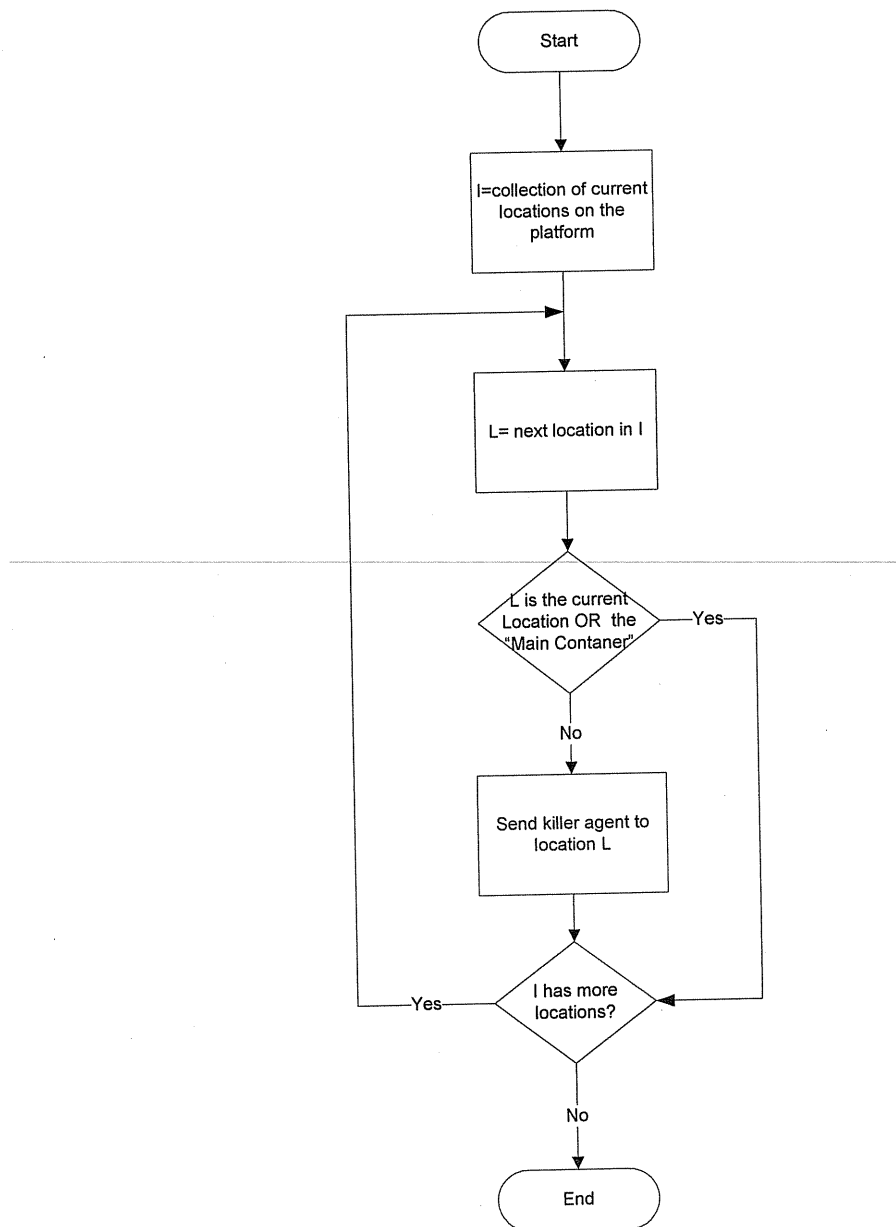


Figure 4.19. Flowchart of Send killers.

4.4.3. Worker agent - WA

Worker agent is responsible for the "searching and checking" process, it gets its travel plan from the DCA , and it goes to the digital libraries according to that travel plan.

After it moves to a location, it starts checking for plagiarized text within the documents available in the digital library, which are related to the category of the held document by the worker agent.

If it find a threshold value of the percentage of plagiarized text in one of the documents it is searching (this is to be specified by the user), then it sends a message of the results to the receiver agent and kills itself.

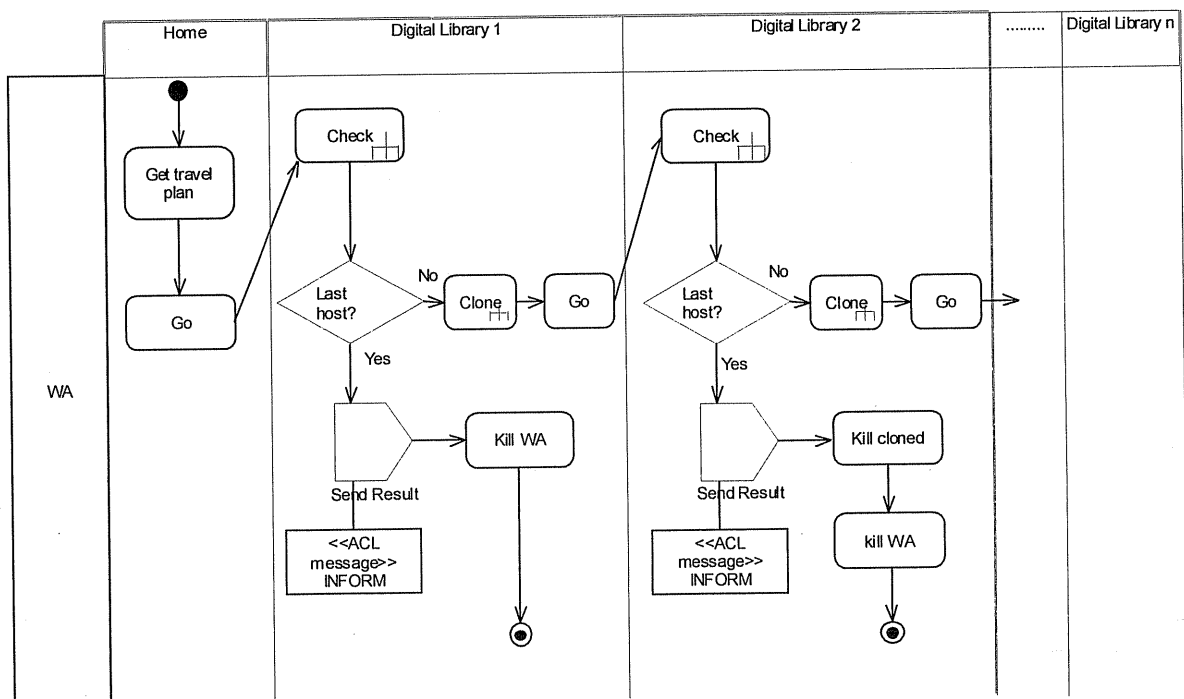


Figure 4.20. Worker agent's life cycle.

Name	Setup.
Input	List of locations, indication of fault tolerance.
Output	Finishing the initial operations of WA.
Description	This function overrides the function setup() in JADE, it does initialization for WA, it also (according to the indicator of fault tolerance), clone or does not clone the agent, then move to the first location in the location's list, <i>doMove</i> and <i>doClone</i> are functions that are already implemented in JADE.

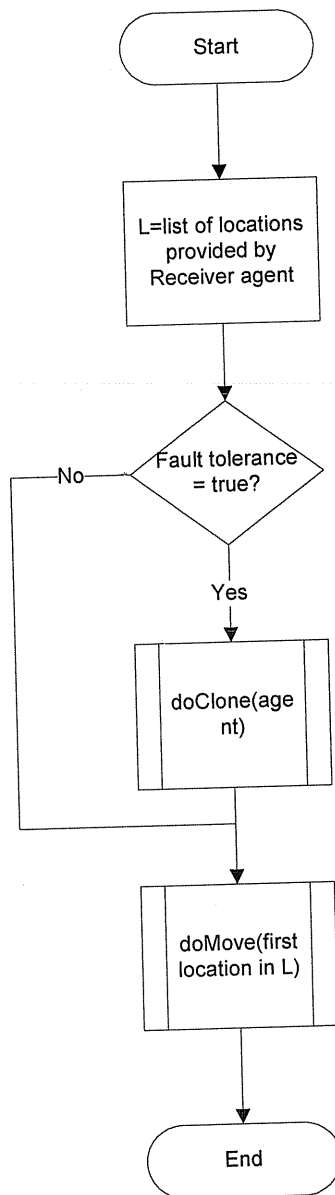


Figure 4.21. Flowchart of Setup.

Name	afterMove
Input	Indicator of fault tolerance, list of locations.
Output	Result of checking.
Description	This function overrides the function afterMove() in JADE, it does the operations that the WA does after moving to a location.

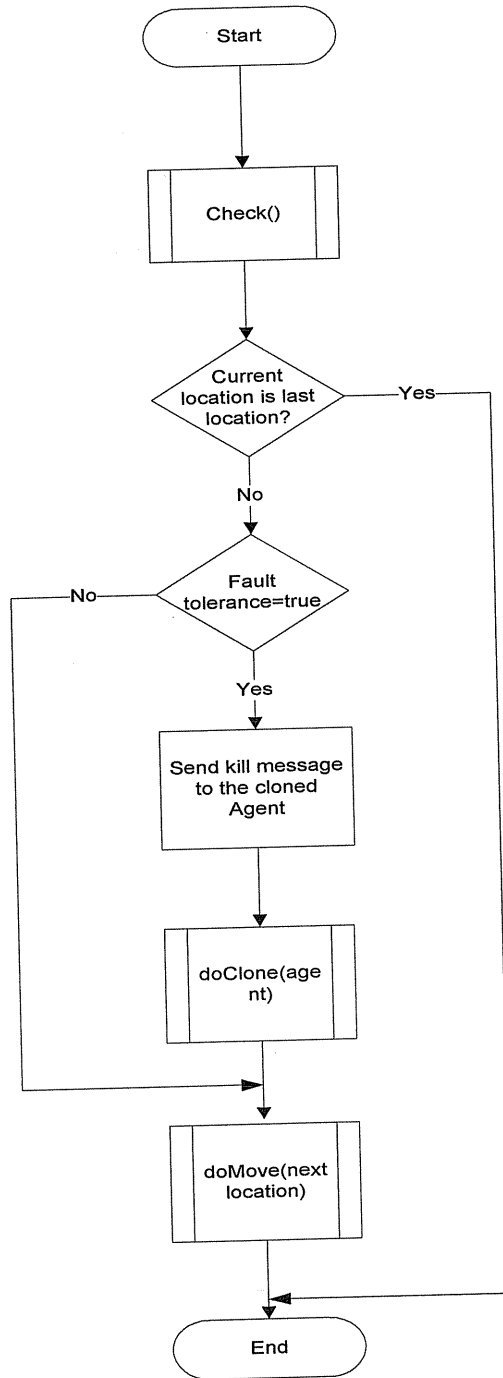


Figure 4.22. Flowchart of afterMove.

Name	Check.
Input	Signatures file of input document.
Output	Array of hits positions.
Description	This function will check the signatures of the input document against the documents in the digital library.

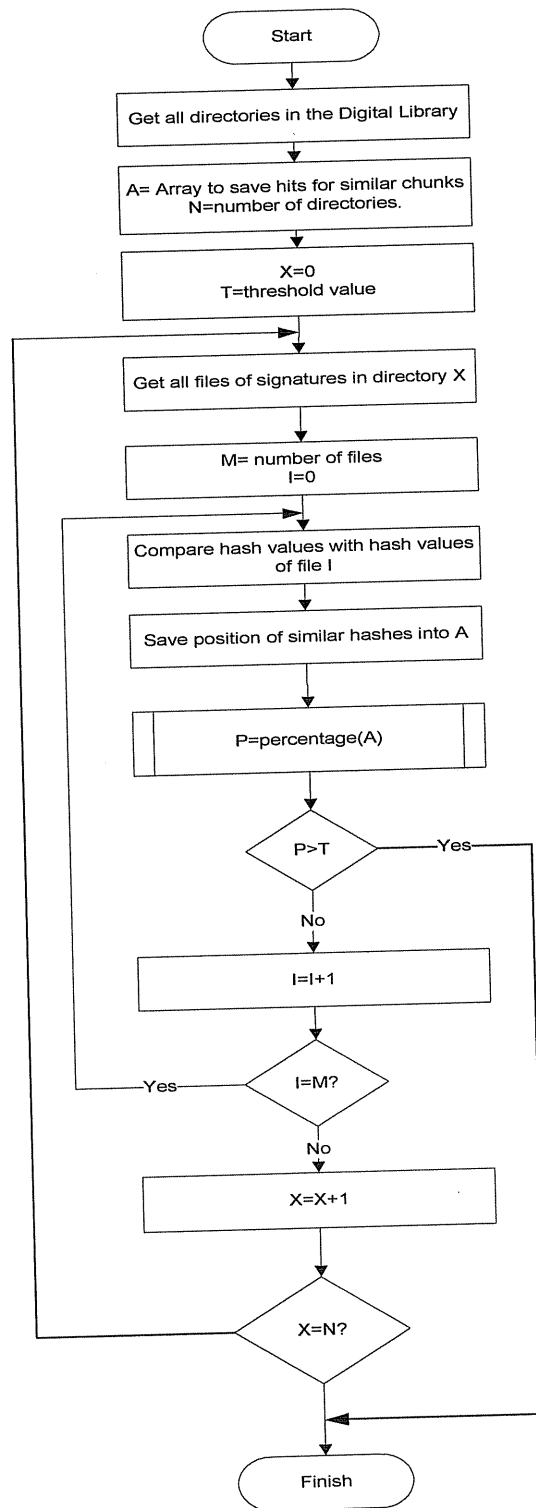


Figure 4.23. Flowchart of Check.

Name	Percentage.
Input	Array of hits positions, chunk size.
Output	Percentage of plagiarism.
Description	This function will calculate the percentage of plagiarism found in the document.

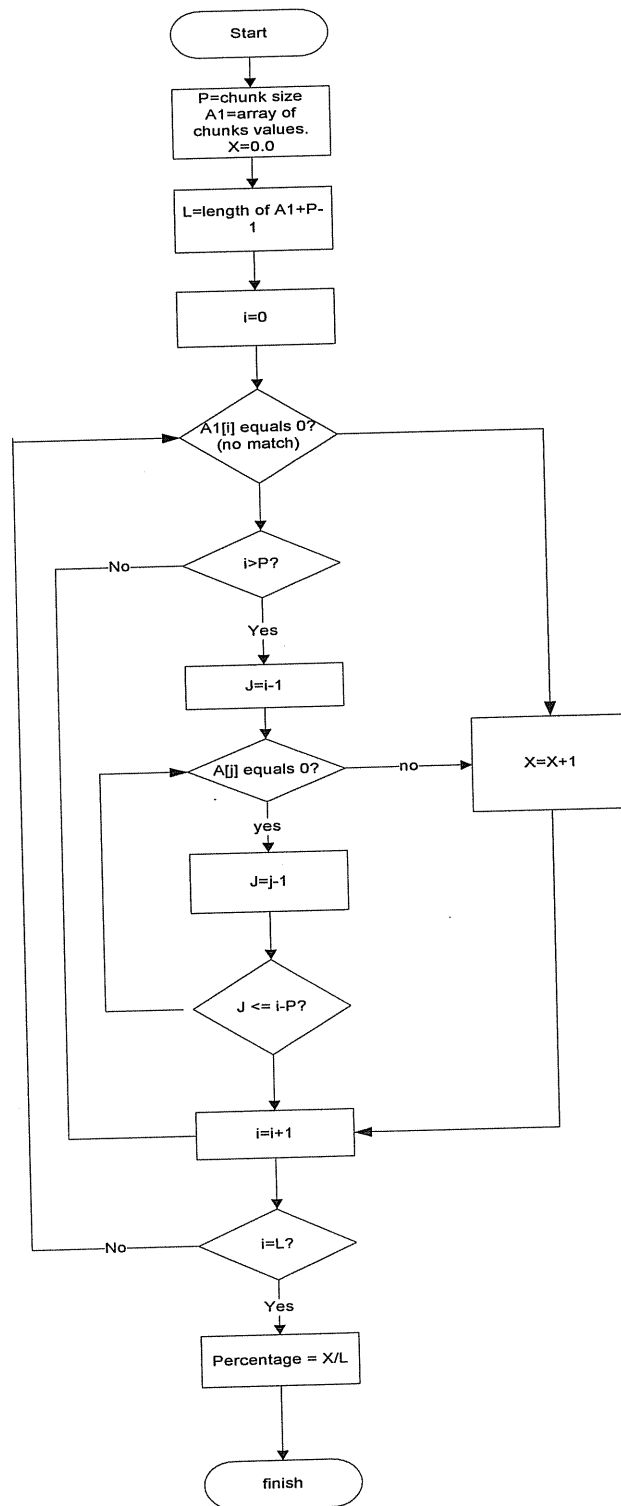


Figure 4.24. Flowchart of Percentage.

4.4.4. Killer agent - KA

Killer agent starts when a WA sends a message with more than a threshold value of a percentage of plagiarized text, DCA starts killer agents provided by the location where they are to go, After the killer agent moves to the location intended to go to, for example (Digital library n), it send a request to AMS agent asking for available agents on the container, Then it kills then one by one, and then it kills itself.

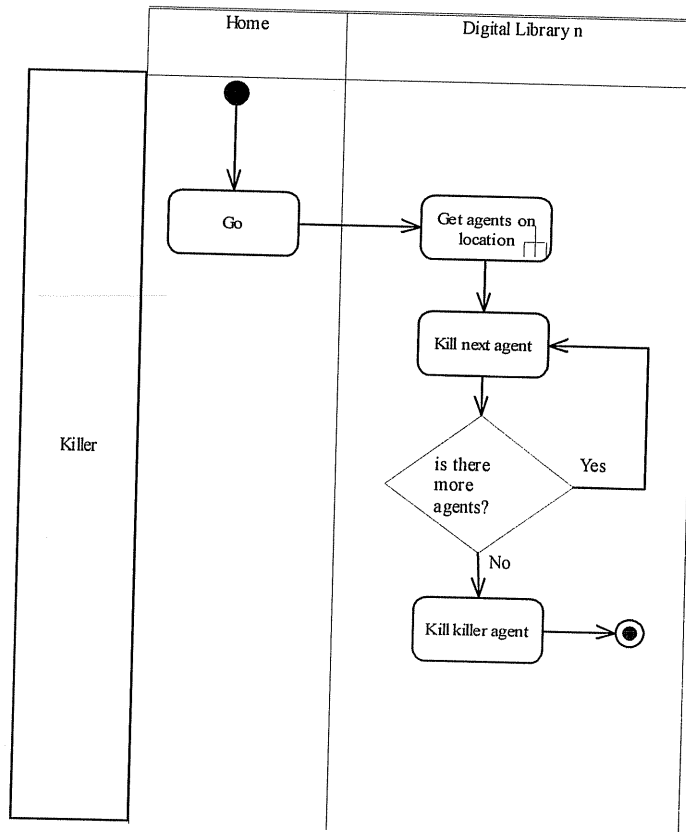


Figure 4.26. Killer agent's life cycle.

Name	Setup.
Input	A location.
Output	Transition of the agent to a location.
Description	This function overrides the function setup() in JADE, it does the initialization of the KA.

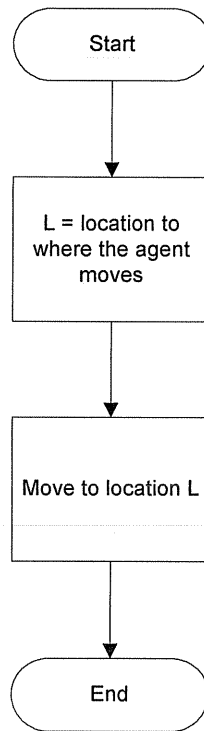


Figure 4.27. Flowchart of Setup.

Name	afterMove
Input	Document number.
Output	Killing of agents.
Description	This function overrides the function afterMove() in JADE, it does the operations that the WA does after moving to a location, that are killing agents that are still working on the respective document.

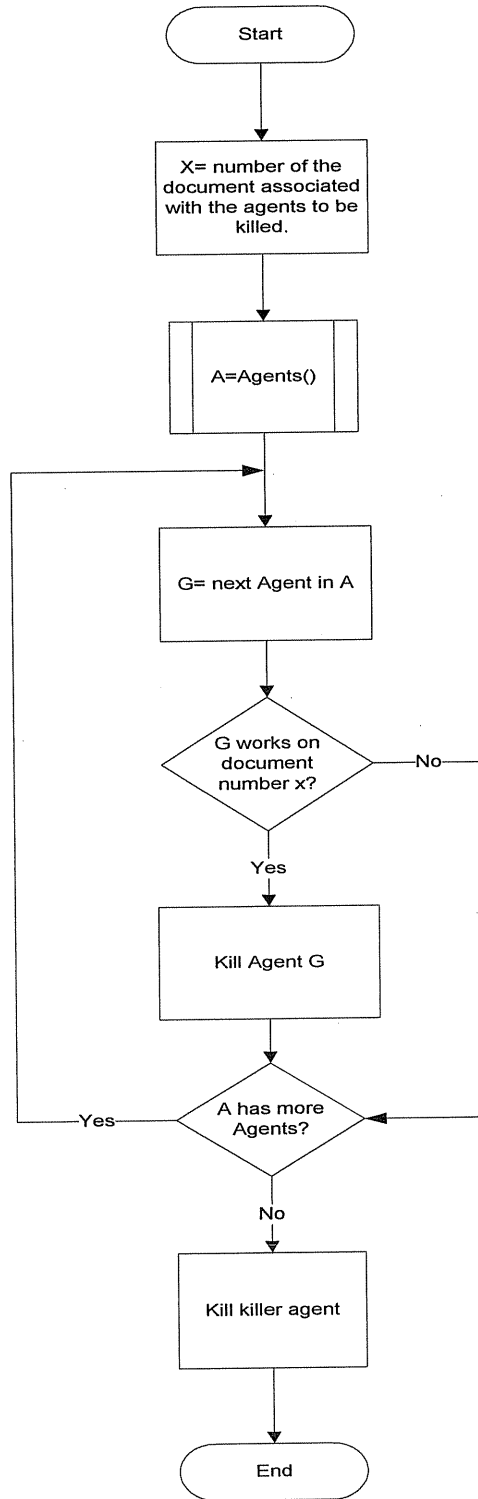


Figure 4.28. Flow chart of afterMove.

4.4.5. Categories collector agent – C²A

As the categories collector agent starts, it goes to the digital library provided by the C3A, it responsible for going to a digital library, and then collecting the different categories available on that digital library, then it sends them as a message to the C3A, and finally it kills itself.

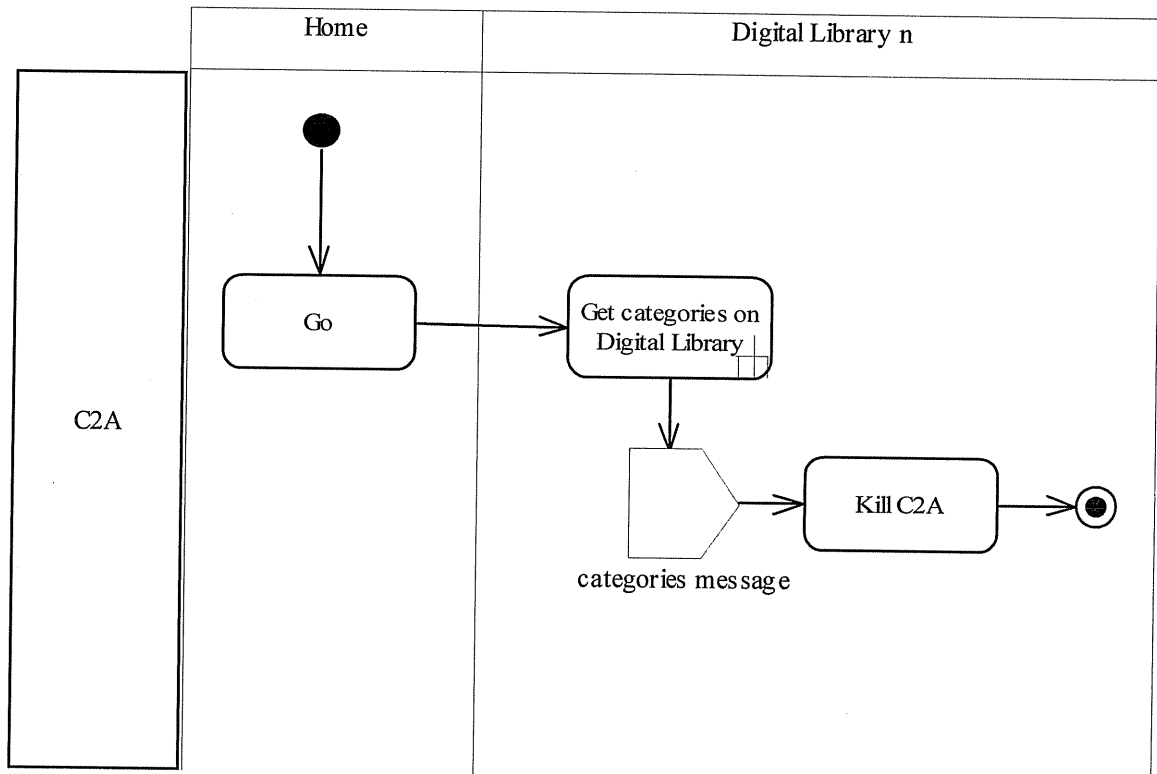


Figure 4.30. Categories collector agent' life cycle.

Name	afterMove.
Input	Name of categories.
Output	Sending message of categories to C3A.
Description	This function overrides the function afterMove() in JADE, it does the operation that the C2A does after moving to a location, that is to collect all available categories and send them to C3A.

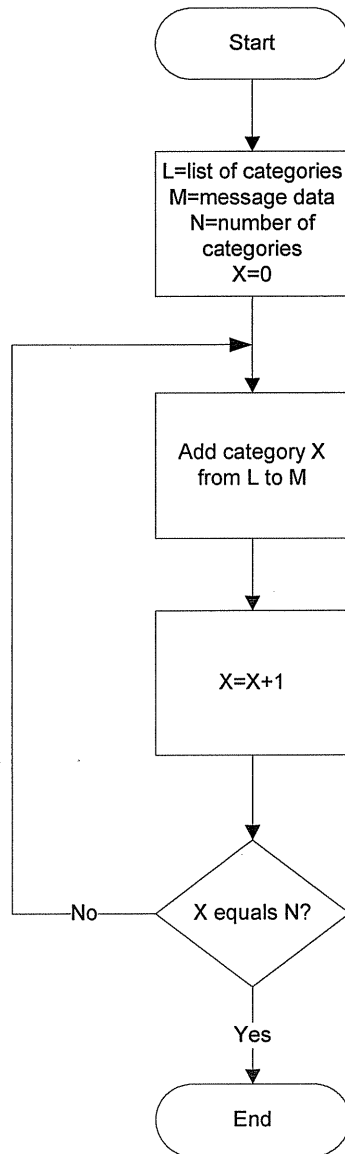


Figure 4.31. Flow chart of afterMove.

4.4.6. Categories collectors' coordinator agent – C³A

The C3A activates C2A agents providing them with the location where each one should go to, and then waits for messages from them contains categories that they found on the respective digital library, this agent shows to the user the different categories collected by the C2A agents.

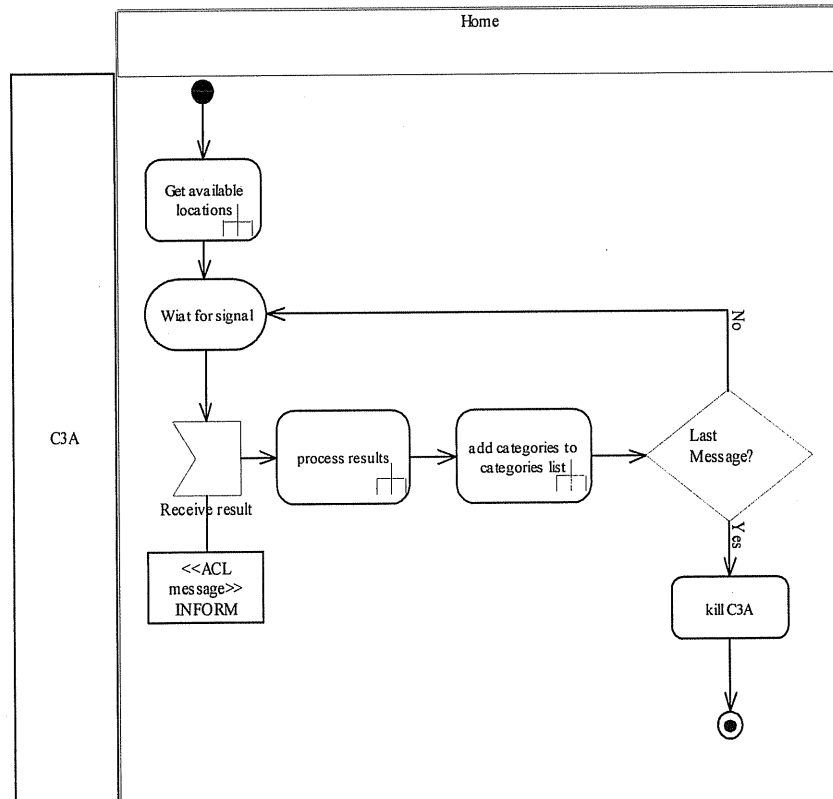


Figure 4.32. C3A life cycle.

Name	Setup.
Input	A list of locations.
Output	Activating C2A agents.
Description	This function overrides the function setup() in JADE, it does the initialization of the C3A, that is it sends a C2A to each location it has in the list

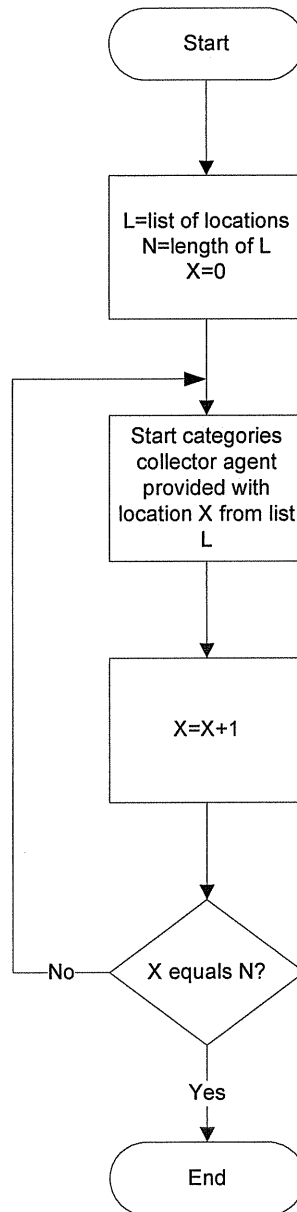


Figure 4.33. Flow chart of Setup.

Name	Receiving messages.
Input	Drop down menu, message data.
Output	Transition of the agent to a location.
Description	This function is called continuously until all C2A send messages to C3A, it processes the message and adds categories to the drop down menu as a message is received.

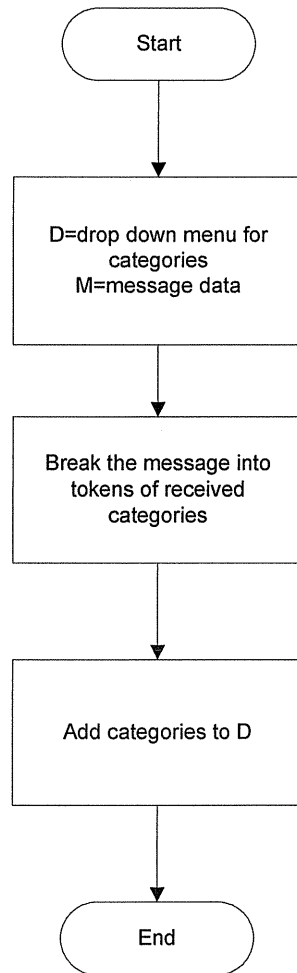


Figure 4.34. Flow chart of receiving messages.

4.4.7. Buttons' controller agent - BCA

This agent is responsible for disabling (Open Documents, Start) buttons when the process starts, then waiting for all DCAs to finish their work and then enables the two buttons.

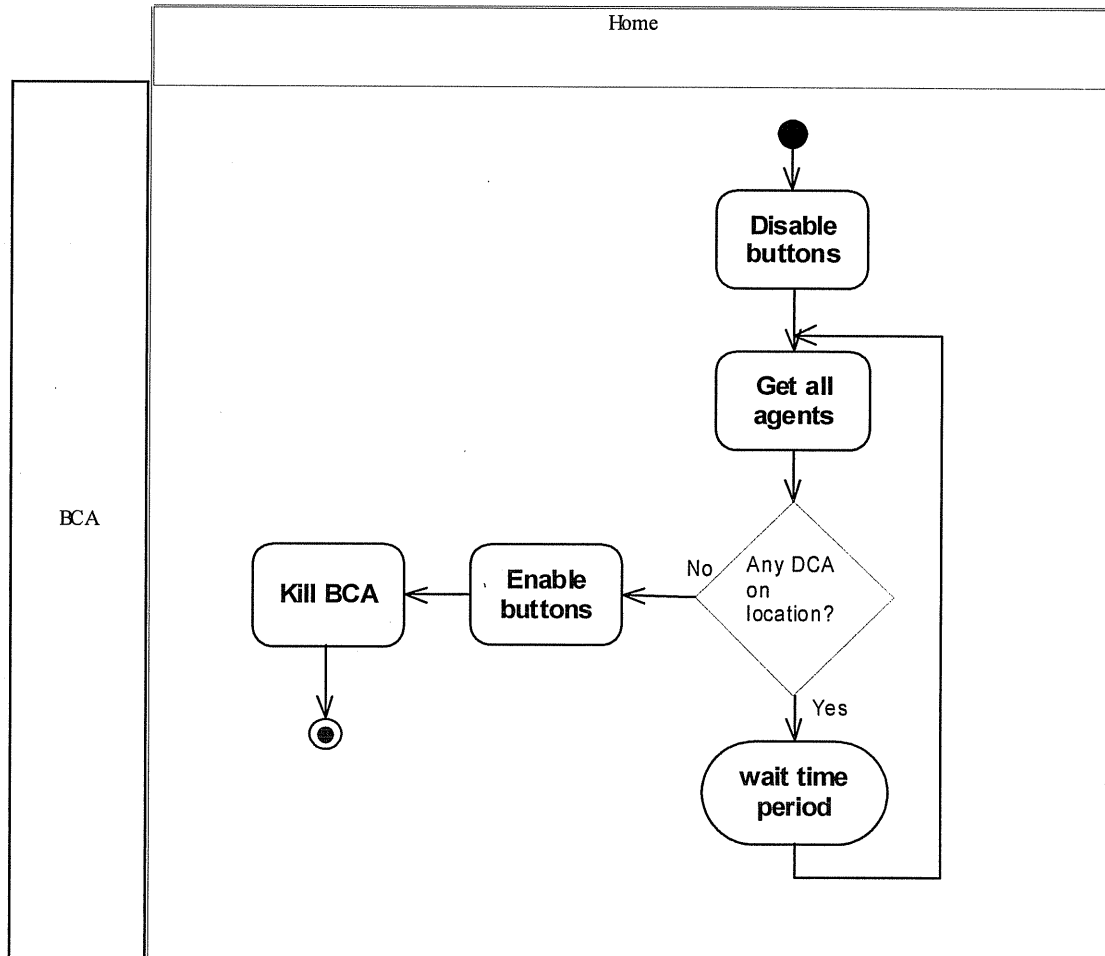


Figure 4.35. BCA life cycle.

4.5. GUI design

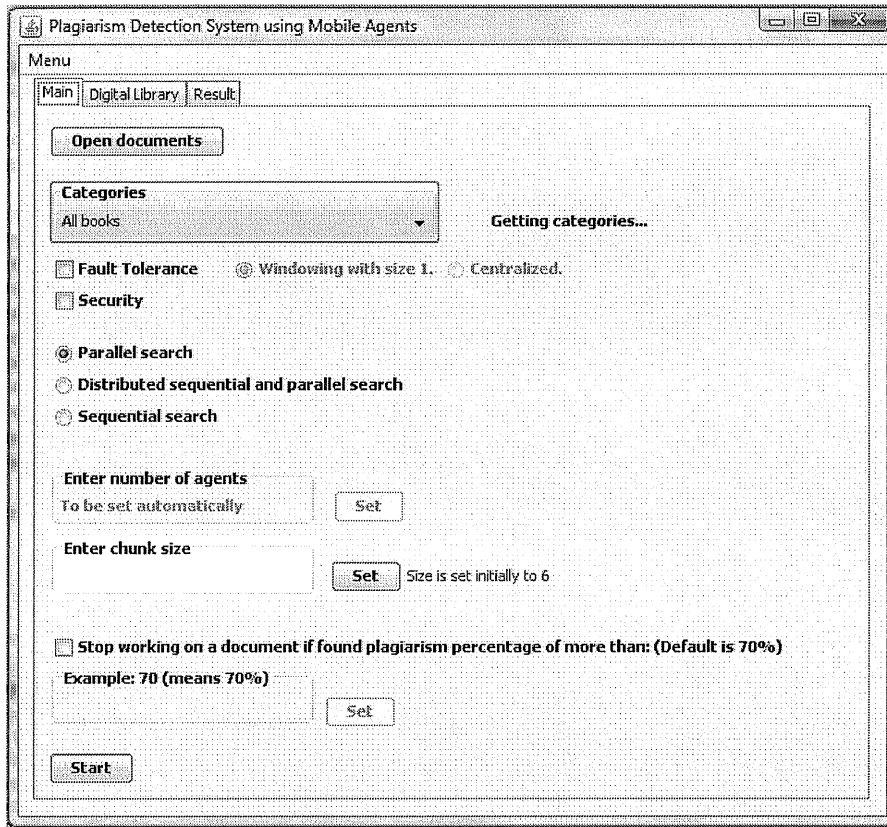


Figure 4.36. Main window.

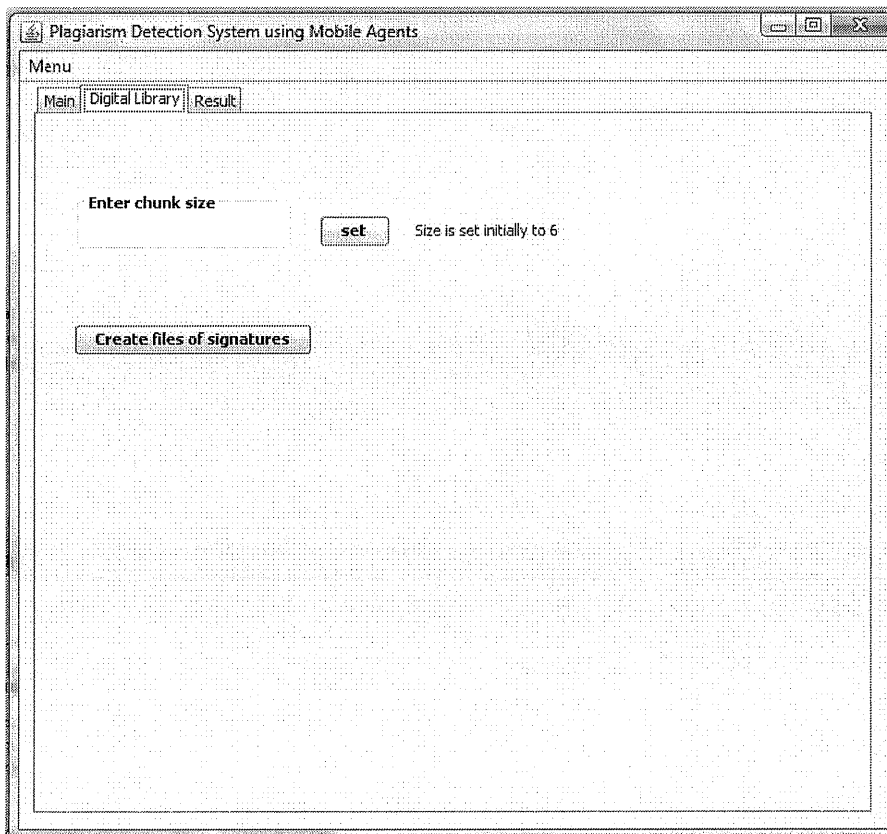


Figure 4.37. Digital library.

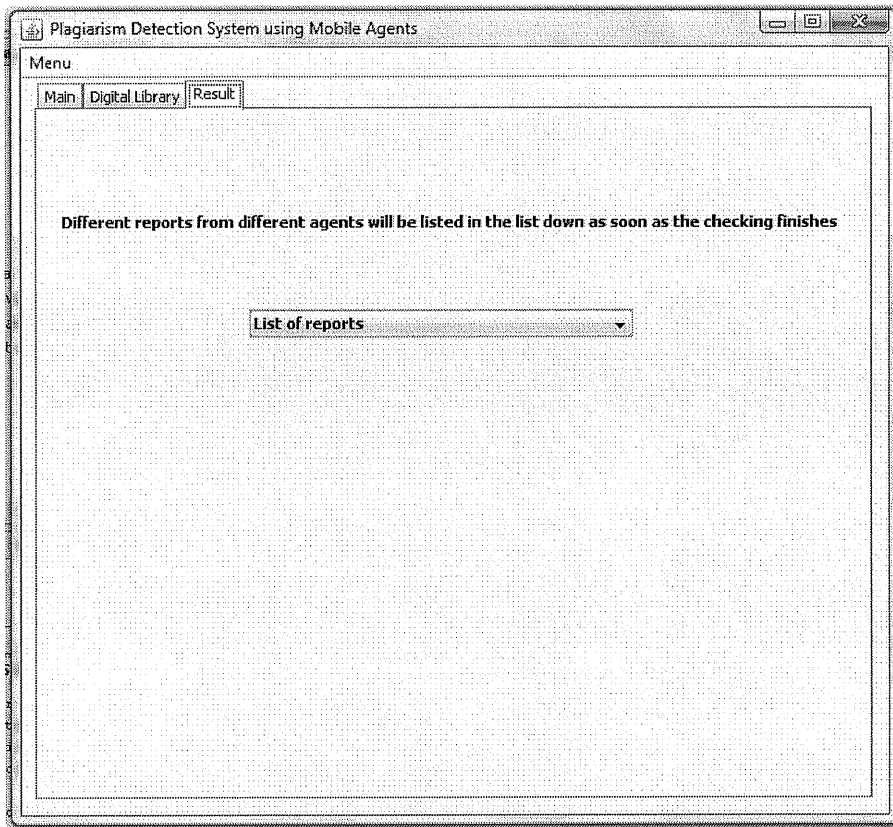


Figure 4.38. Results.

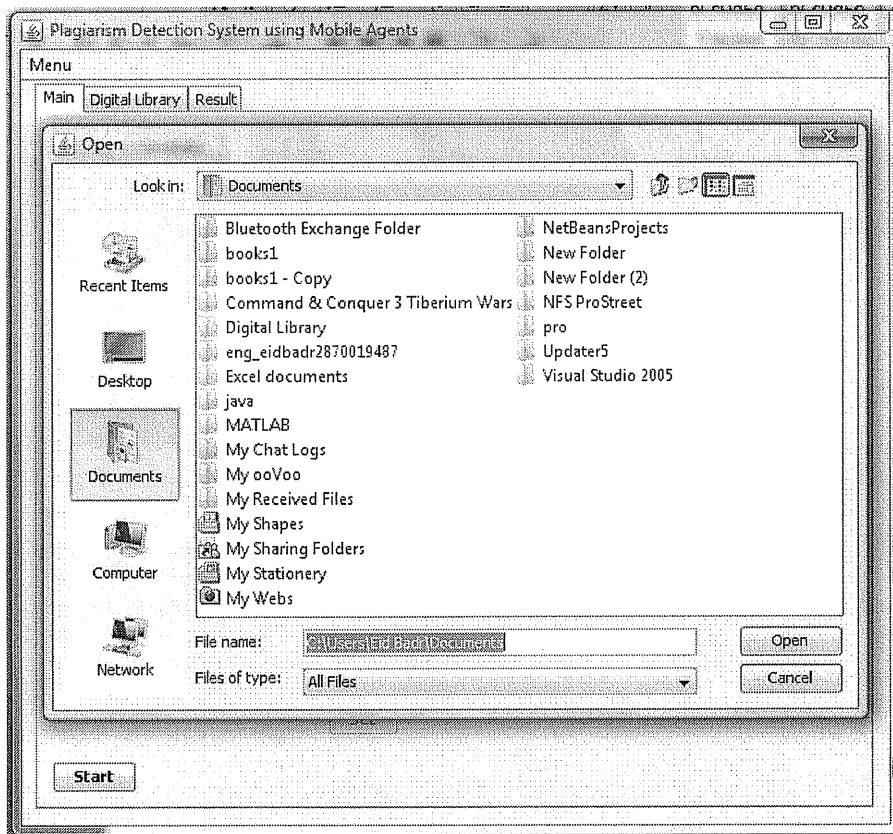


Figure 4.39. Open Documents

4.6. Fault tolerance module

There are different approaches in order to make the mobile agent fault tolerant, one of them is agent replication, under the branch of agent replication there are different approaches, such as [44]:

1. Centralized approach
2. Windowing approach
 - a. Size-1 approach
 - b. Size-n window approach
3. Centralized-windowing approach

4.6.1. Centralized approach

In this approach, any agent that is created on any container on the agent platform, a replica is made on the main container, if the agent migrate to another container then the replica is updated so that the final state of the agent is reflected on the replica, if the agent crashed then the replica continues from where the agent crashed [43].

For example let container M be the main container, agent A is on container 1, agent A done a task and moved to container 2, agent A' (replica of agent A) is created on the main container, agent A moved to container 3, A' is updated to reflect the state of agent A before moving to container 3, if agent A crashed while on container 3 then A' starts working on container 3 starting from the state on which agent A left container 2.

Illustration:

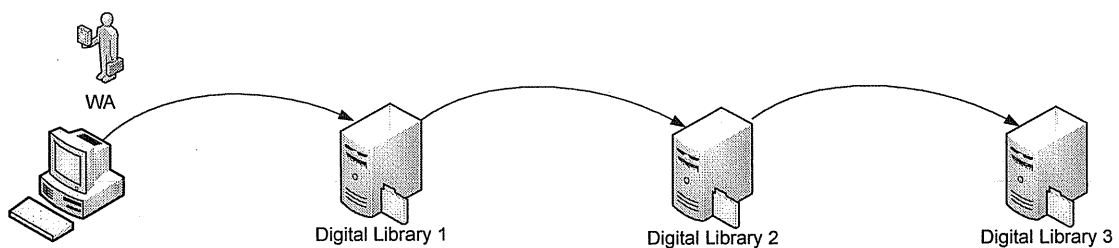


Figure 4.40. WA is at host container.

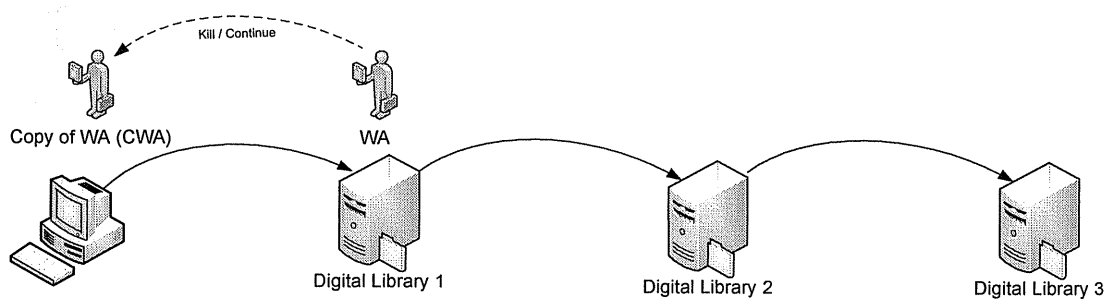


Figure 4.41. WA made a copy of itself (CWA) at the host container and then moves to Digital Library 1.

Two possibilities for WA:

1. A problem happens to it prohibiting it from continuing its work such as getting killed.
2. No problem happens with WA and it continues its work.

- For (1) it sends a message to CWA telling it to continue the work (CWA also makes a copy of itself (copy of CWA) at home container) see figure 4.42.

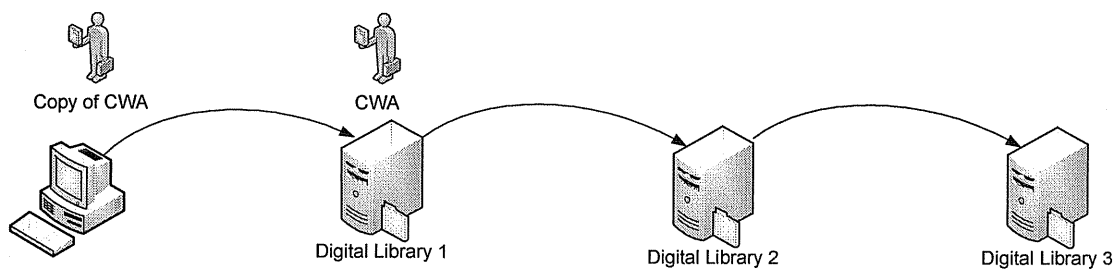


Figure 4.42. WA got killed.

- For (2) it makes an updated copy of itself (C2WA) at home container, and tells CWA to kill itself see figure 4.43.

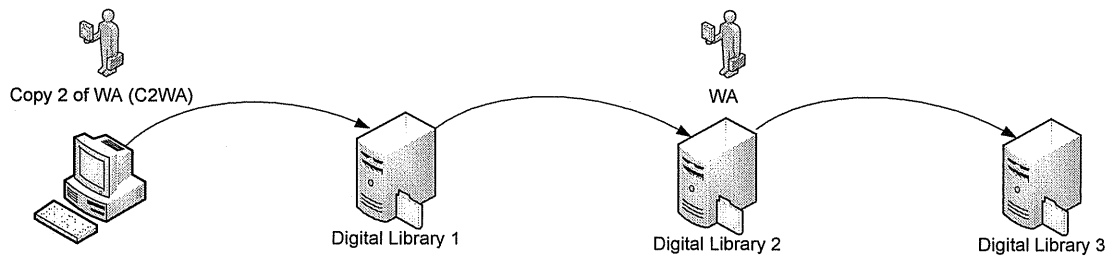


Figure 4.43. WA continued working.

4.6.2. Windowing approach

In this approach, each time a mobile agent leaves a container it makes a replica of itself on that container, when moving to the next container the replica just waits until the agent crashes or moves to the (next) container, if it crashes then the replica continues the trip instead of the crashed agent, if it moves to the (next) container then the replica deletes itself, and another replica is made on the container that the agent just left, (This is in case of size-1 window), in case of n-size window replicas are made on more than the previously visited container [43].

For example (size-1 window), agent A is on container 1, agent A moved to container 2, a replica is made on container 1, if agent A moved to container 3 then the replica deletes itself and another replica is made on container 2 having the same state on which the agent A left container 2, but if agent A crashed then the replica continues the trip from the state on which agent A left container 1.

Illustration:

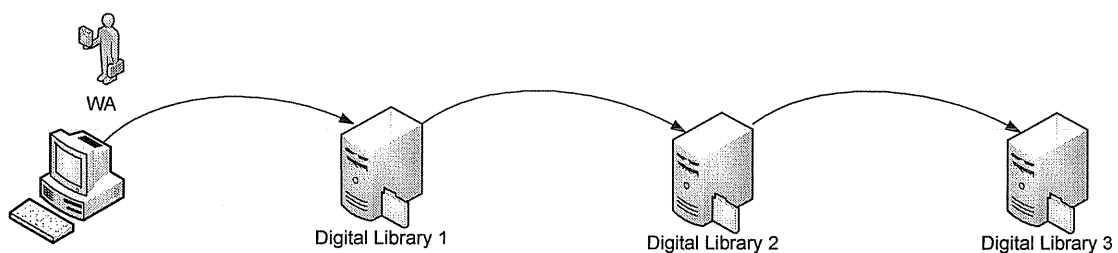


Figure 4.44. WA is at the host container.

- WA made a copy of itself (CWA) at home container and then moved to Digital Library1.

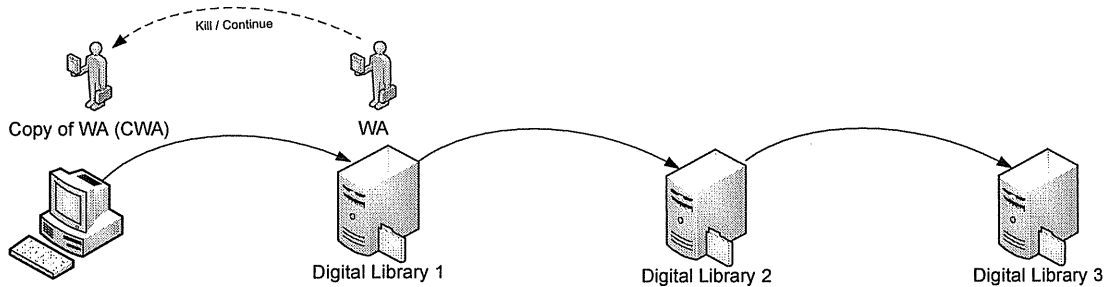


Figure 4.45. WA made a copy of itself (CWA) at the host container and then moved to Digital Library 1.

Also two possibilities for WA:

1. A problem happens to it prohibiting it from continuing its work such as getting killed.
2. No problem happens with WA and it continues its work.

For (1) it sends a message to CWA telling it to continue the work (CWA also makes a copy of itself (copy of CWA) at home container) see figure 4.46.

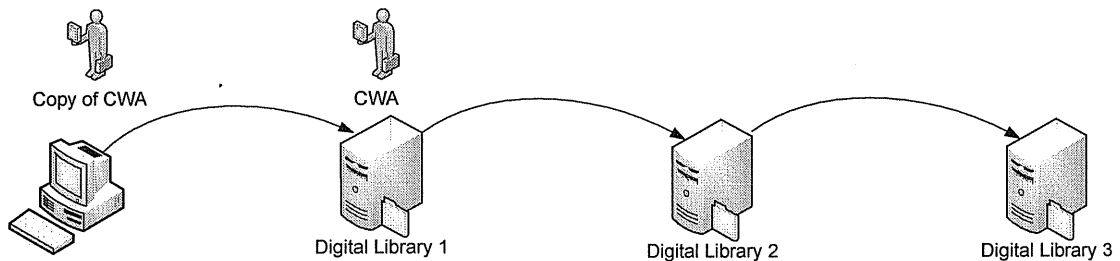


Figure 4.46. WA got killed

- For (2) it makes an updated copy of itself (C2WA) at Digital Library 1 (just left it), and tells CWA to kill itself see figure 4.47.

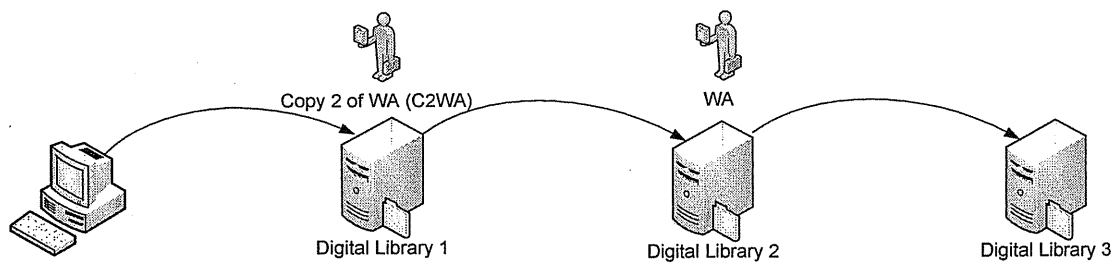


Figure 4.47. WA continued working.

4.6.3. Centralized-windowing approach:

This approach combines the previously discussed approaches.

In this project centralized and windowing with size-1 approaches was applied.

4.7. Security module

As previously described, there are many security aspects that can be applied in a multi-agent system, encryption of messages sent by an agent and received by another agent would be necessary in a multi-agent system.

To make all data communication between agents secure, a symmetric key encryption algorithm is applied, a key (K) is generated every time an agent is activated, and so it uses this key to encrypt the data in each message it sends, the key is saved at the receiver in order to decrypt the encrypted message that it receives and the agent rotates the key by n , when moving from digital library to another.

Three steps to secure the communication between the agents illustrated in figures 4.48, 4.49, 4.50.

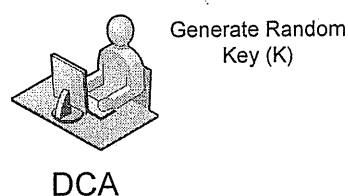


Figure 4.48. Security step 1: DCA generates a random key (K).

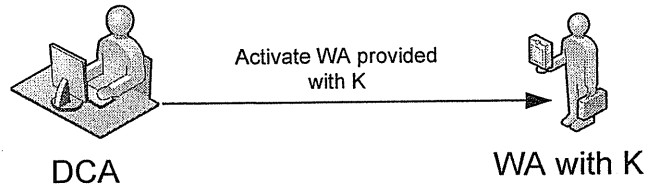


Figure 4.49. Security step 2: DCA activates WA provide with K, DCA saves K.

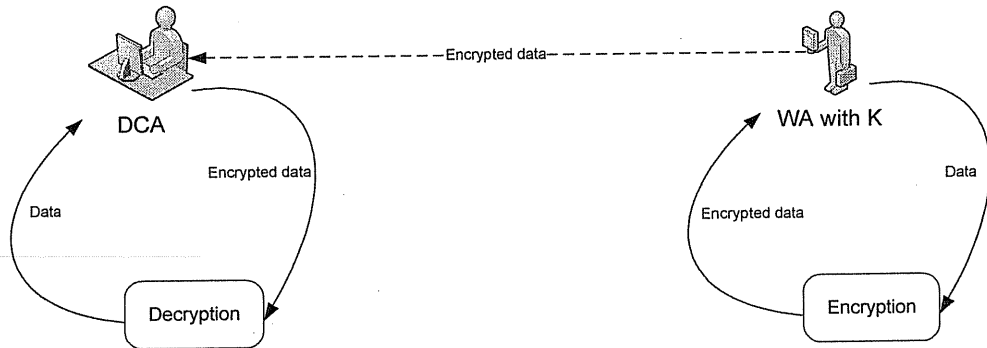


Figure 4.50. Security step 3: communication between WA and DCA.

- **Encryption**

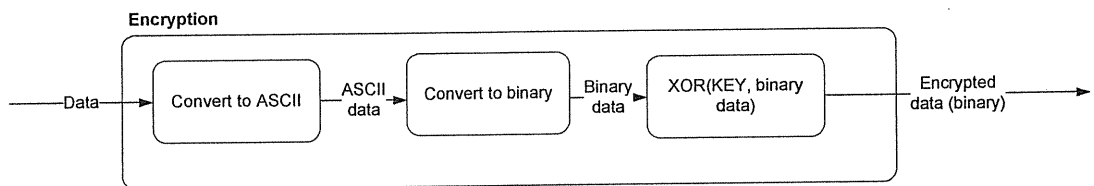


Figure 4.51. Encryption of data.

- **Decryption**

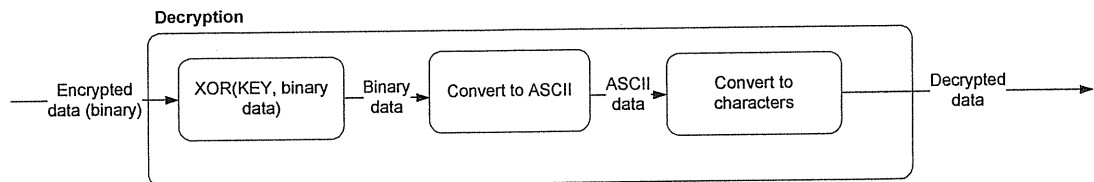


Figure 4.52. Decryption of encrypted data.

- **Generation of a key**

Generation of a key was done by taking a sequence of character of some random length (N) between two threshold values, each character is either 1 or 0, by using Math library available in java, a 0 or 1 can be generated randomly using

```
(int)Math.round(Math.random());
```

then *x* is converted to a string and concatenated to rest of the sequence and finally a key with length (*N*) is generated.

Conversion of data to ASCII and binary:

Using `Integer.toString(c)` a character (*c*) could be converted to a binary representation of its ASCII equivalence.

For characters that need only 6 digits to be represented, an additional 0 is added at the front of the result.

Conversion of binary to data:

A binary sequence could be converted back to characters by dividing the sequence into chunks of 7 digits and then using `(char)(Integer.parseInt(x, 2))` the chunk is converted back to its equivalent character.

XORing:

XORing is done simply by checking the two respective characters from two input strings, if both are 0's or both are 1's then the result is 0 else the result is 1 and continuing until finishing checking all characters.

4.8. Limitations

- Documents to be checked are limited to be of extension (doc) and (txt).
- Documents in the digital library are limited to be of extension (txt).
- The language in the documents is limited to English only.
- The system does not check paraphrasing or the meaning of the words.
- Chunk size is limited to be minimally 3 and maximally 10.

4.9. Assumptions

- The user of the system is assumed to be familiar with computer programs.

- The Main Container is assumed to stay alive all over the time of processing.
- At least one digital library is assumed to be presented on the platform.
- All WAs are assumed to be able to move to any container on the platform.
- All WAs are assumed to be able to clone themselves on any container on the platform.

4.10. Summary

In this chapter a detailed description of the design of different system components and modules, including system architecture, flow charts, GUI design, modules specifications, assumptions and limitations.

CHAPTER 5

IMPLEMENTATION

AND TESTING

5.1. Development Environment

5.2. Development Process

5.3. Testing

5.4. Summary

Chapter Five

Implementation and Testing

In this chapter description of how the components of the system were implemented, and testing of the system.

5.1. Development Environment

The implementation of the system was done on two environments, since JADE (as a simulator) can be used with full functionality on one computer, most of the implementation and software components testing was made on one computer, however, to become closer to the real usage of the system, some testing and partial modifications on the implementation was done over a local area network (LAN).

The software that was used in the implementation and testing phases consist of:

- JADE package: this package should be installed on the working computer and the testing computers as well, in order to start the different JADE containers.
- JDK: should be installed on working computer, in order to compile to java source code into bytecode, then interpreted as a machine code to become executable, also for the testing computers it should be installed as well, in order to make JADE containers runnable on the test computers.
- NetBeans IDE: a free and very helpful IDE that provided us ready implemented GUI components that facilitated the design and implementation of the system, we used this IDE to design the GUI, and to write the source code of all functions.

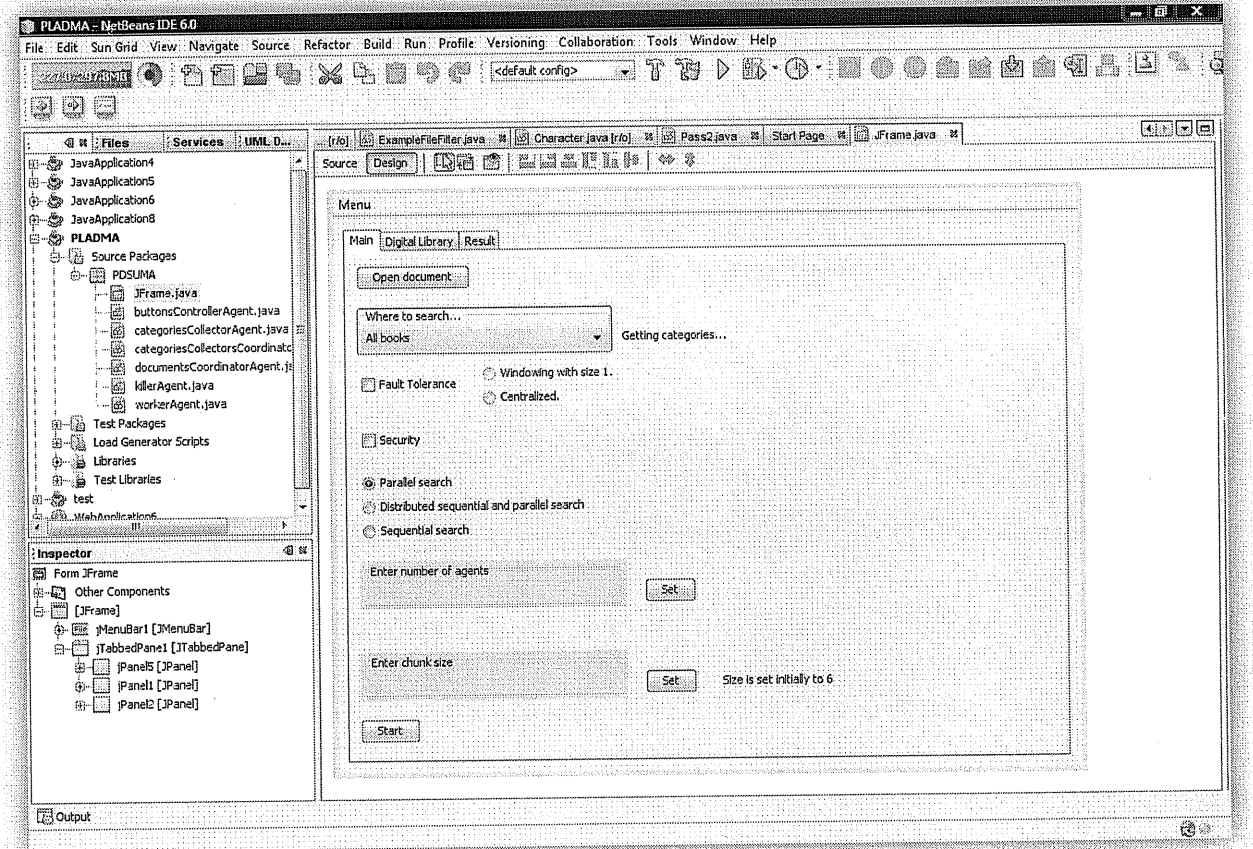


Figure 5.1. NetBeans IDE

5.2. Development Process

Spiral software engineering model was used in the development process, because the overall features and requirements of the system was not clear at the beginning of the development process, many prototypes of the system was delivered, refinement of the requirements after reviewing each prototype was done, and components was tested individually and many integrity tests was done, until a final version of the system was delivered.

5.2.1. Phases of system development

Phase 1: Development of a plagiarism checker:

- Implementation of a basic tool that detects plagiarism among two documents.

- A basic report that highlight the plagiarized text in a different color other than the color of the non-plagiarized text.
- The percentage of the plagiarized text was computed.

Phase 2: setting up a mobile agent:

- The agent does nothing but moving from a container to another.
- The agent can do some basic operations such as cloning and killing itself.

Phase 3: Integration of the output of phase 1 and 2:

- The mobile agent can go to containers and check only one document against another document resides on these containers.

Phase 4: Integrity testing of the output of phase 3 and review of the system.

Phase 5: Design more features and specifications of the system:

- Design of a fault tolerance module.
- Design of a security module.
- Design of different searching techniques.

Phase 6: Implementation of the output of phase 5.

Phase 7: Integrity testing of the output of phase 6 and review the system.

Phase 8: Expansion of the system to check more than one document.

Phase 9: Intensive integrity testing of the system.

Phase 10: Delivery of the system.

Component testing was done during each phase that included implementing of any component of the system.

5.3. Testing

The following figures show the results of different inputs to the system.

Figures describe:

- Error messages.
- Getting categories.
- Checking documents and getting results.
- Searching scenarios.
- Fault tolerance: Windowing size-1.
- Fault tolerance: Centralized.
- Killing Containers.

5.3.1. Error messages

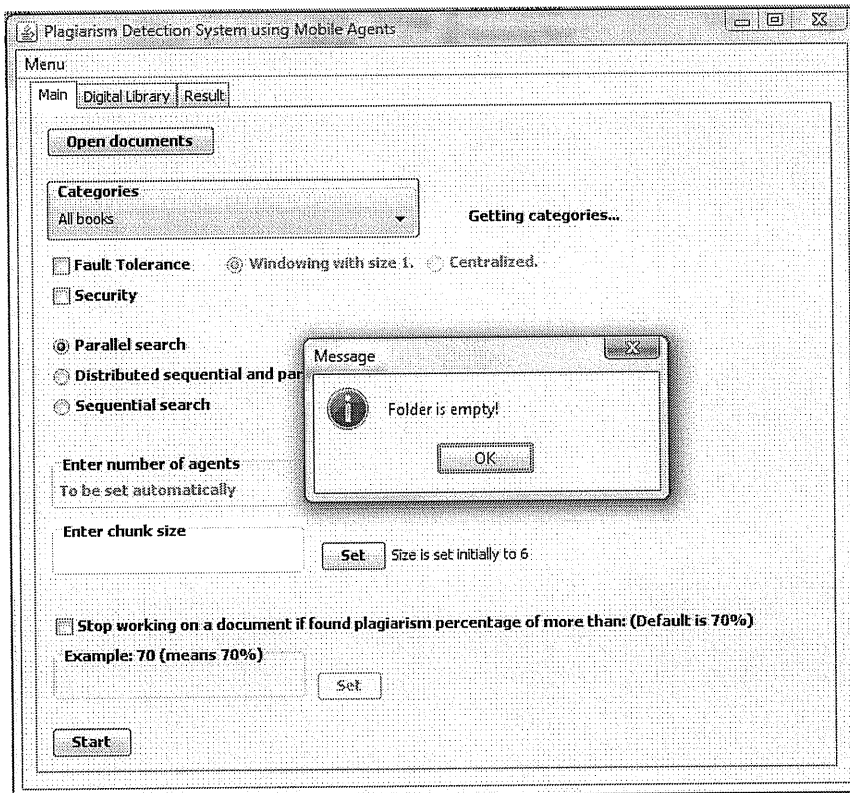


Figure 5.2. Trying to select a folder with no documents.

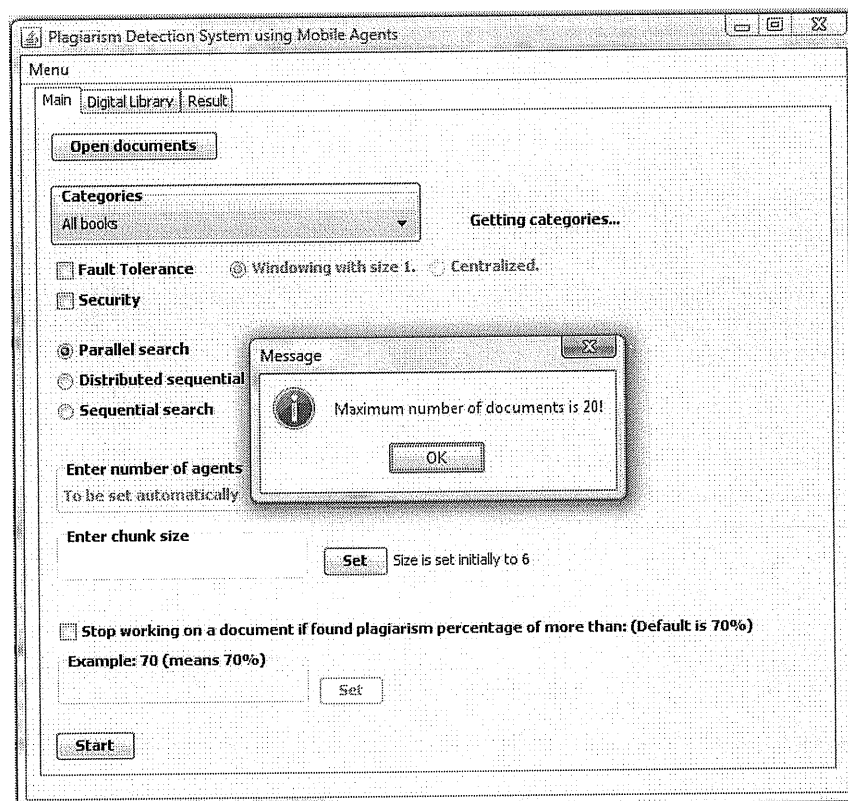


Figure 5.3. Trying to select a folder with more than 20 documents.

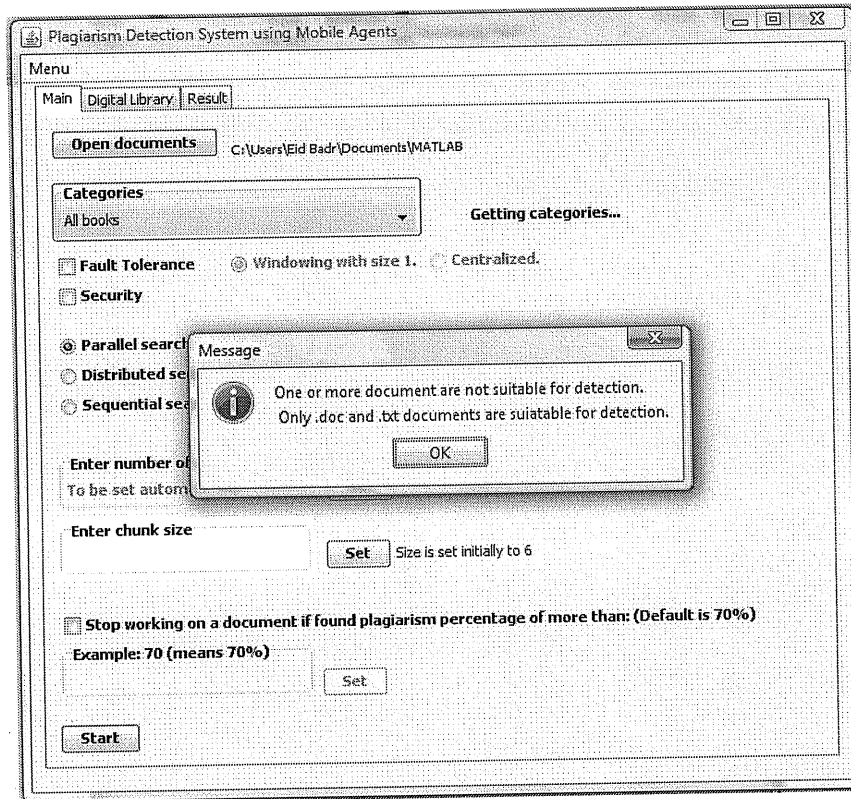


Figure 5.4. Trying to select a folder with documents with unsuitable extensions.

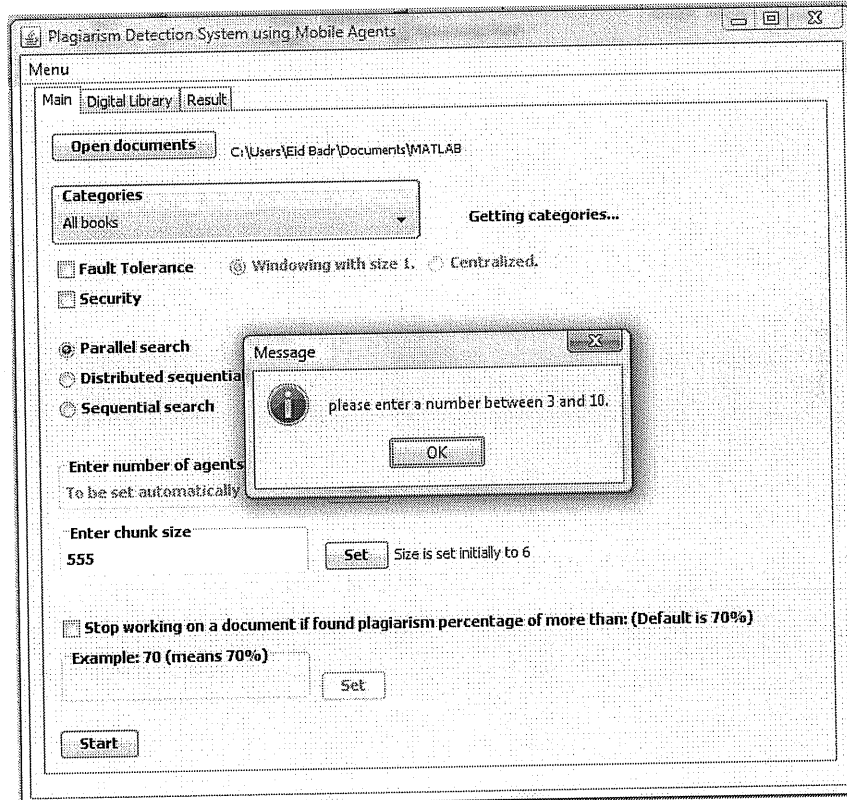


Figure 5.5. Trying to set the chunk size to more than 10.

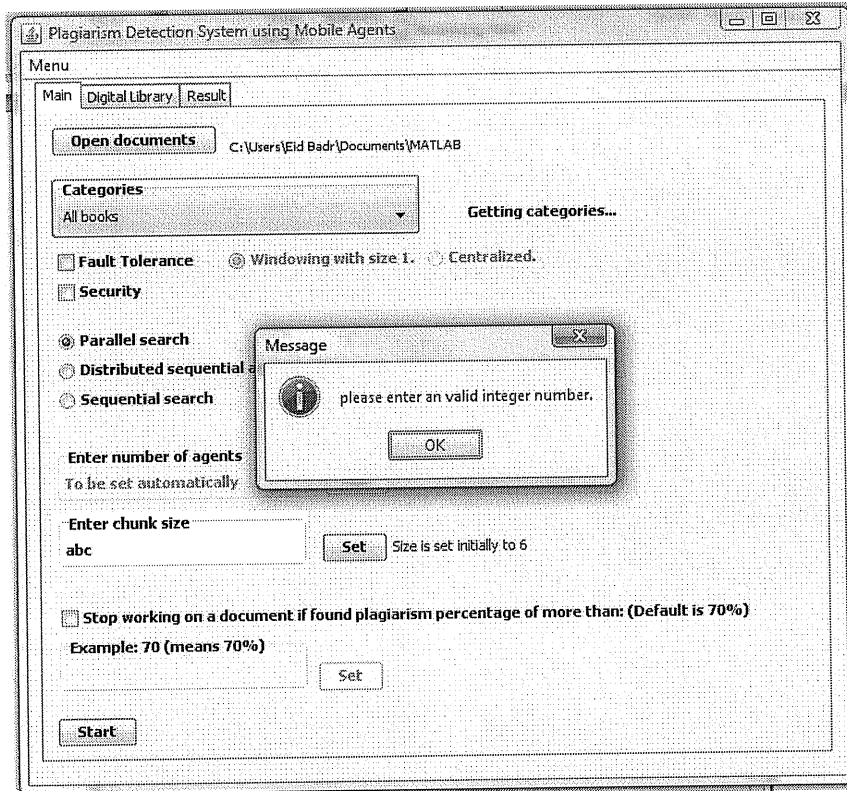


Figure 5.6. Trying to set the chunk size to a non numeric value.

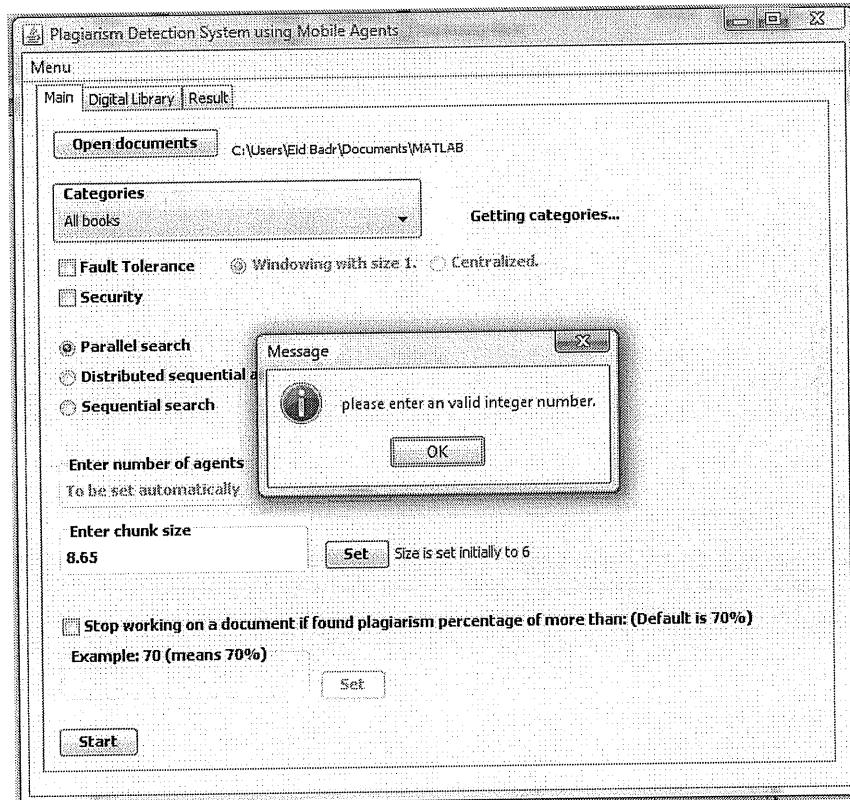


Figure 5.7. Trying to set the chunk size to a non integer value

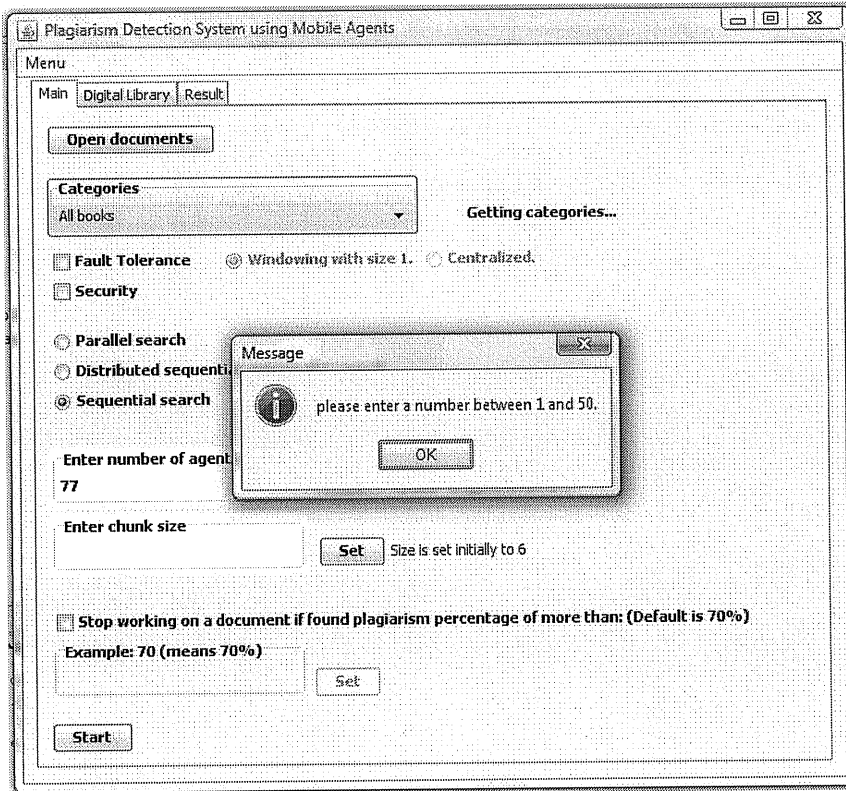


Figure 5.8. Trying to set the number of agent to more than 50.

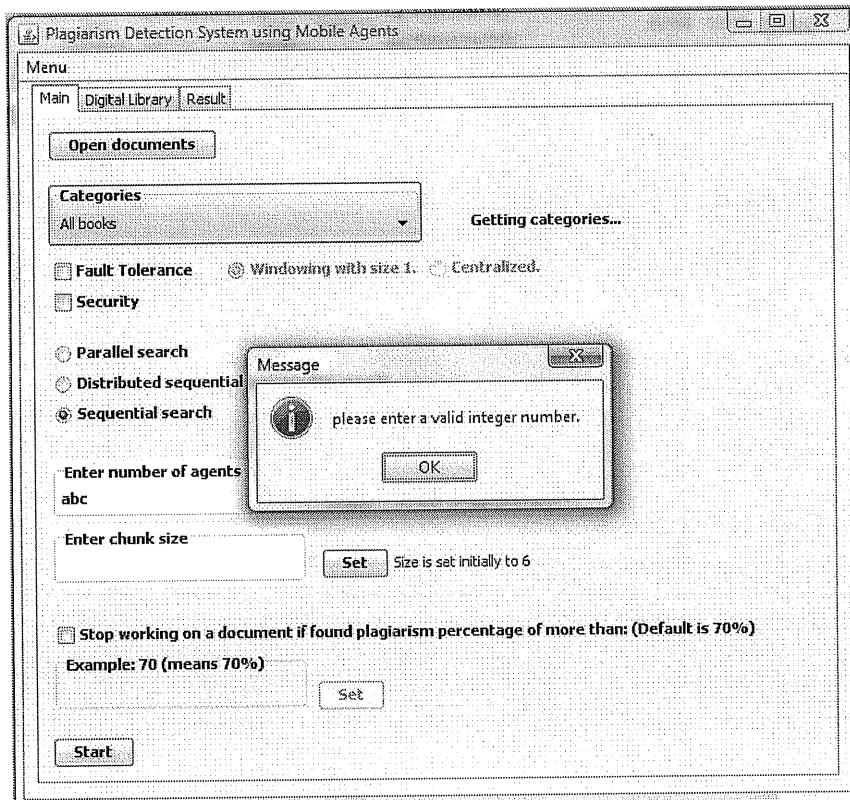


Figure 5.9. Trying to set the number of agents to a non numeric value.

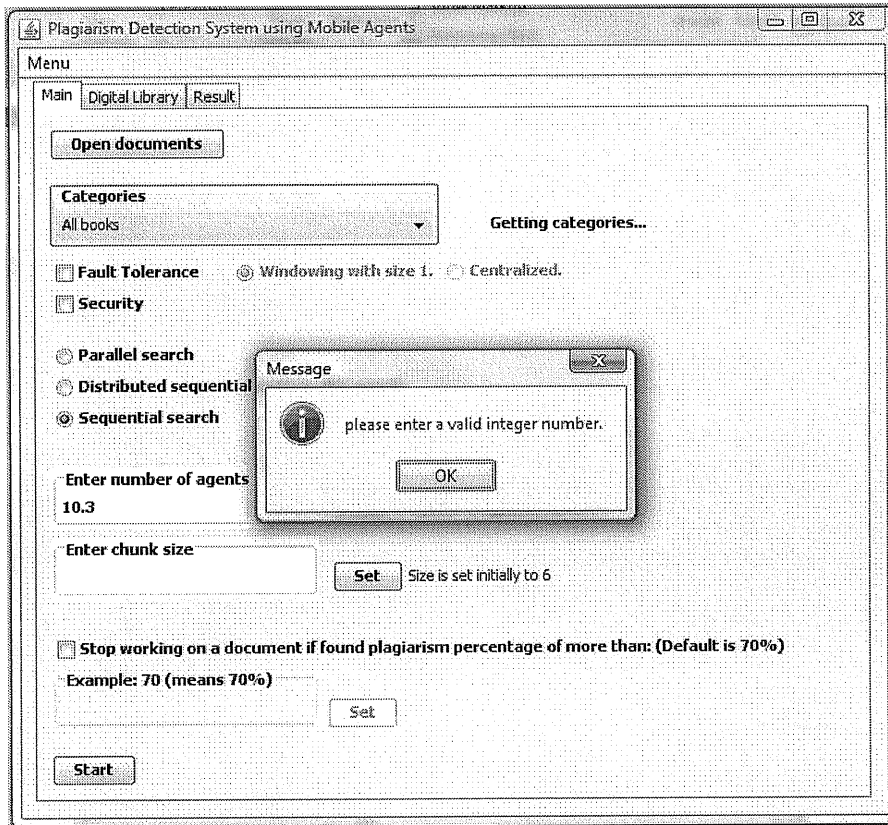


Figure 5.10. Trying to set the number of agents to a non integer value.

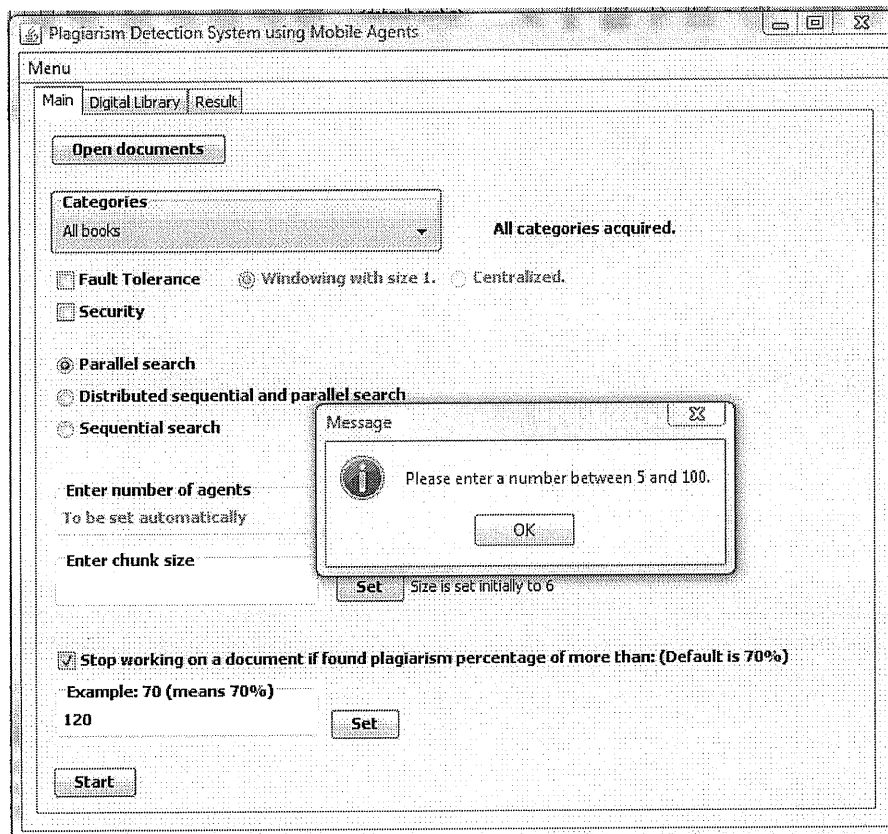


Figure 5.11. Trying to set the percentage to a value out of range.

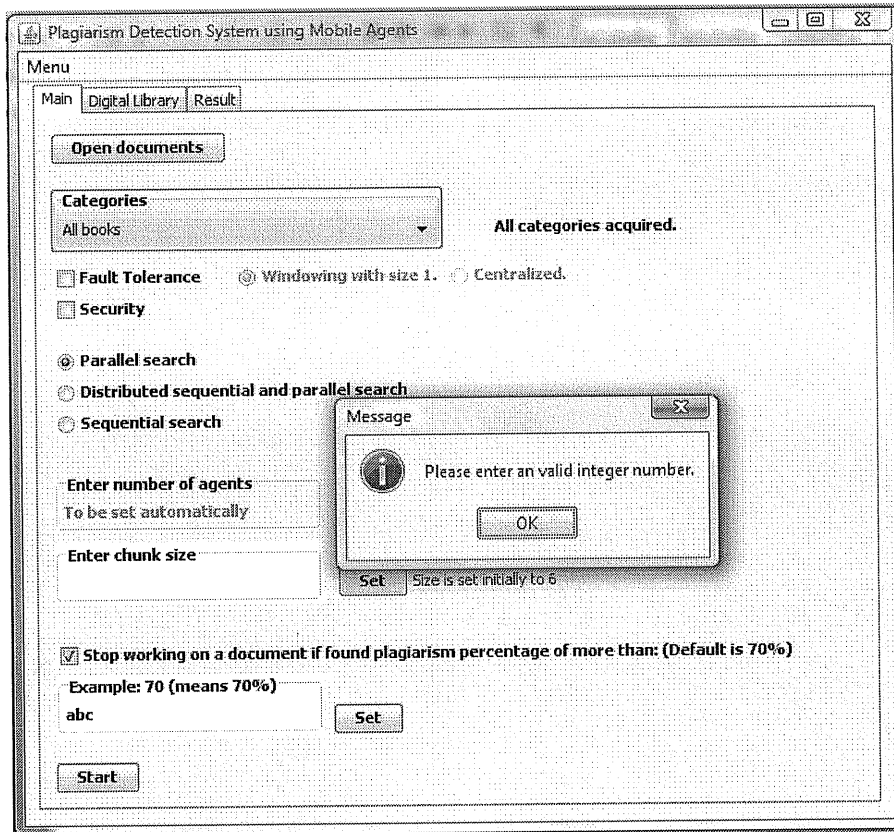


Figure 5.12. Ttrying to set the percentage non-numeric value.

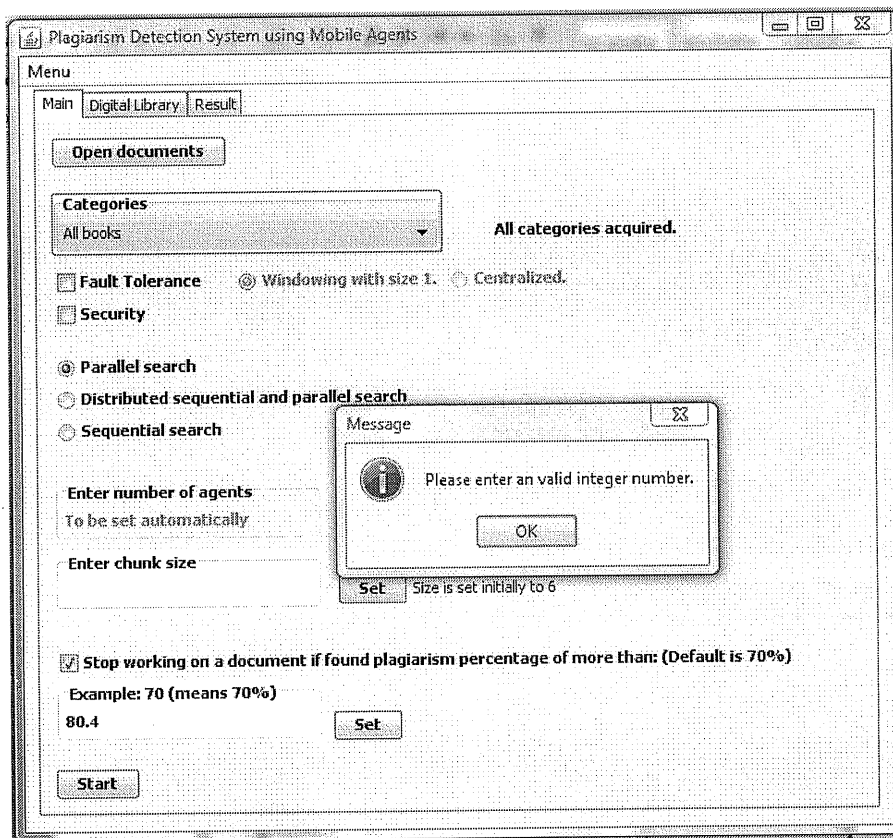


Figure 5.13. Ttrying to set the percentage non-integer value.

5.3.2. Getting categories

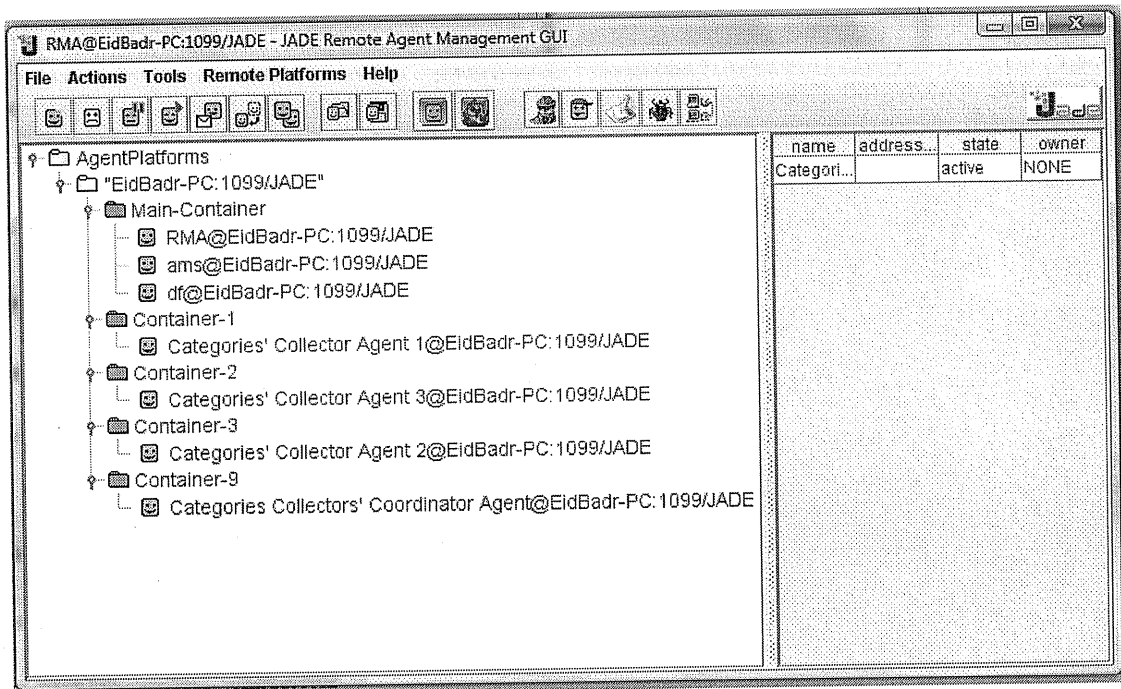


Figure 5.14. The C2As are collecting categories and the C3A is waiting for results.

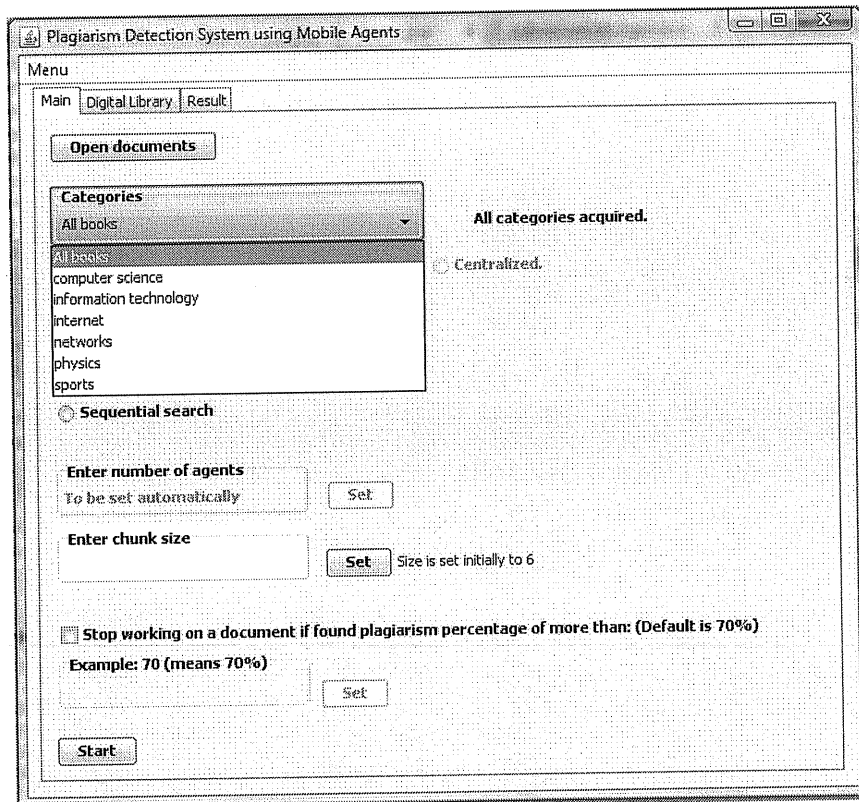


Figure 5.15. Categories acquired.

5.3.3. Checking documents and getting results

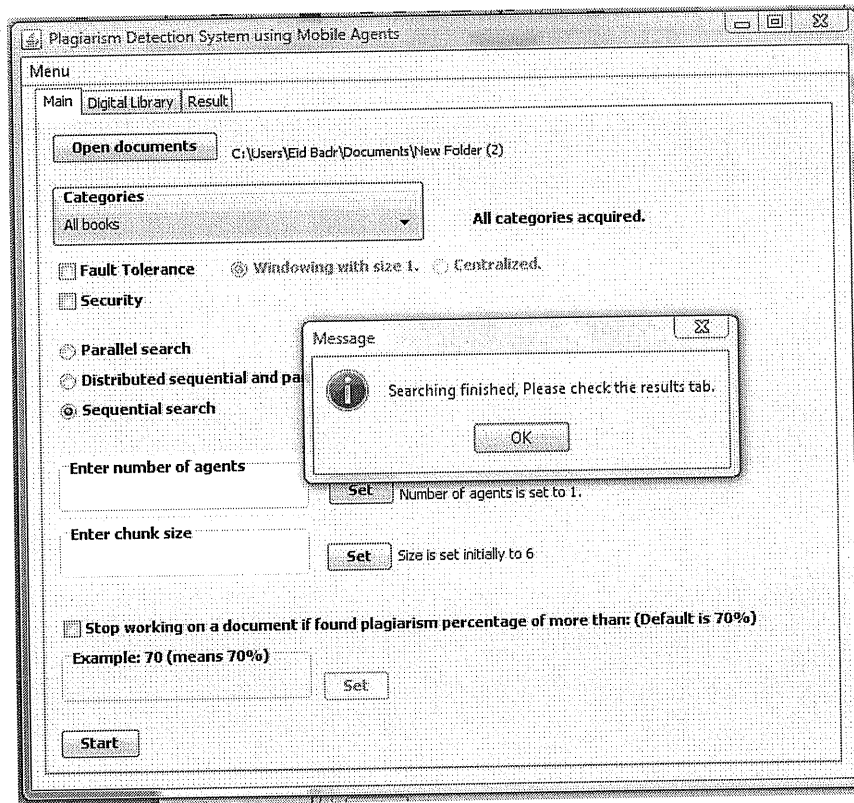


Figure 5.16. Searching and checking of all documents has finished

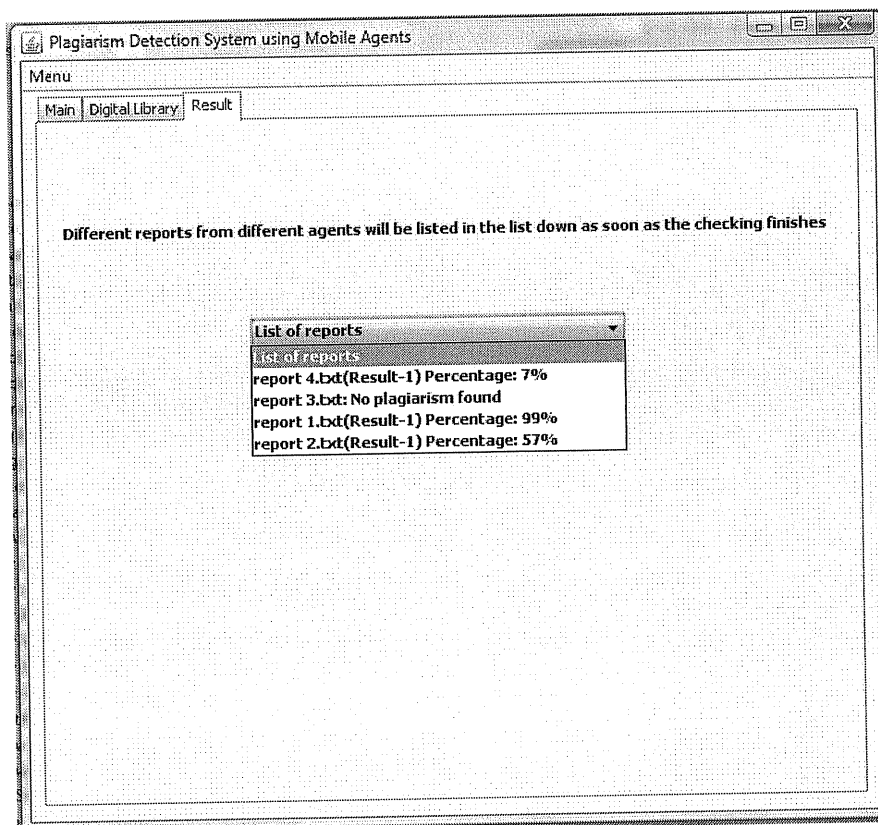


Figure 5.17. List of reports.

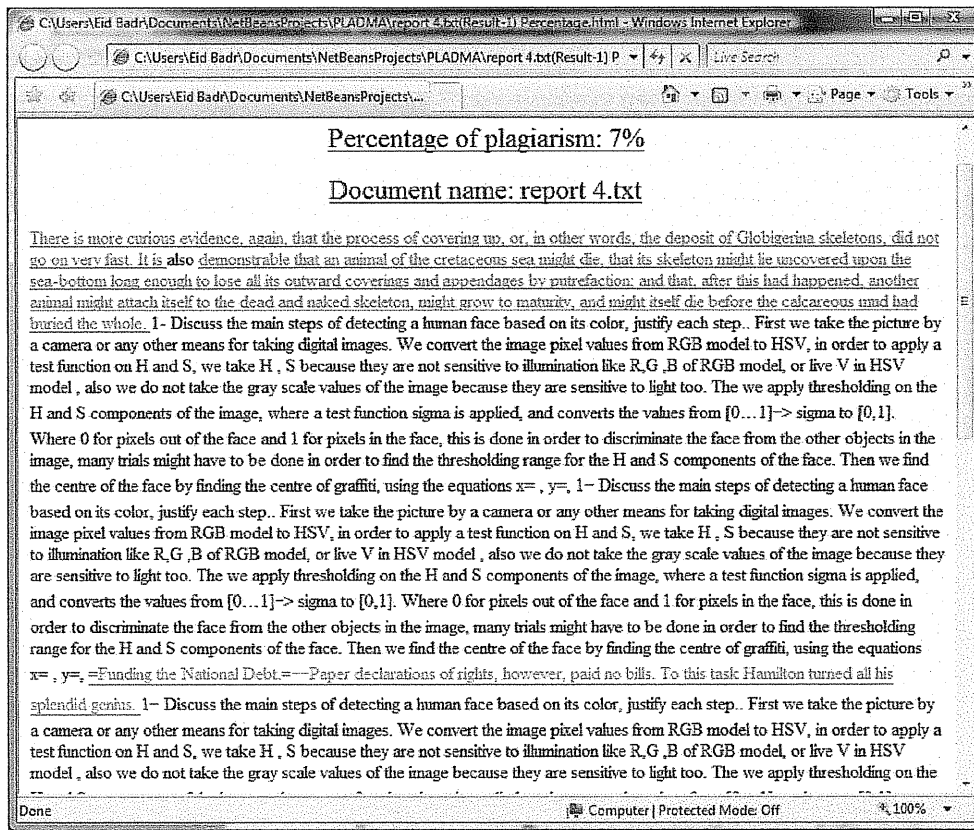


Figure 5.18. Report for document (report 4.txt).

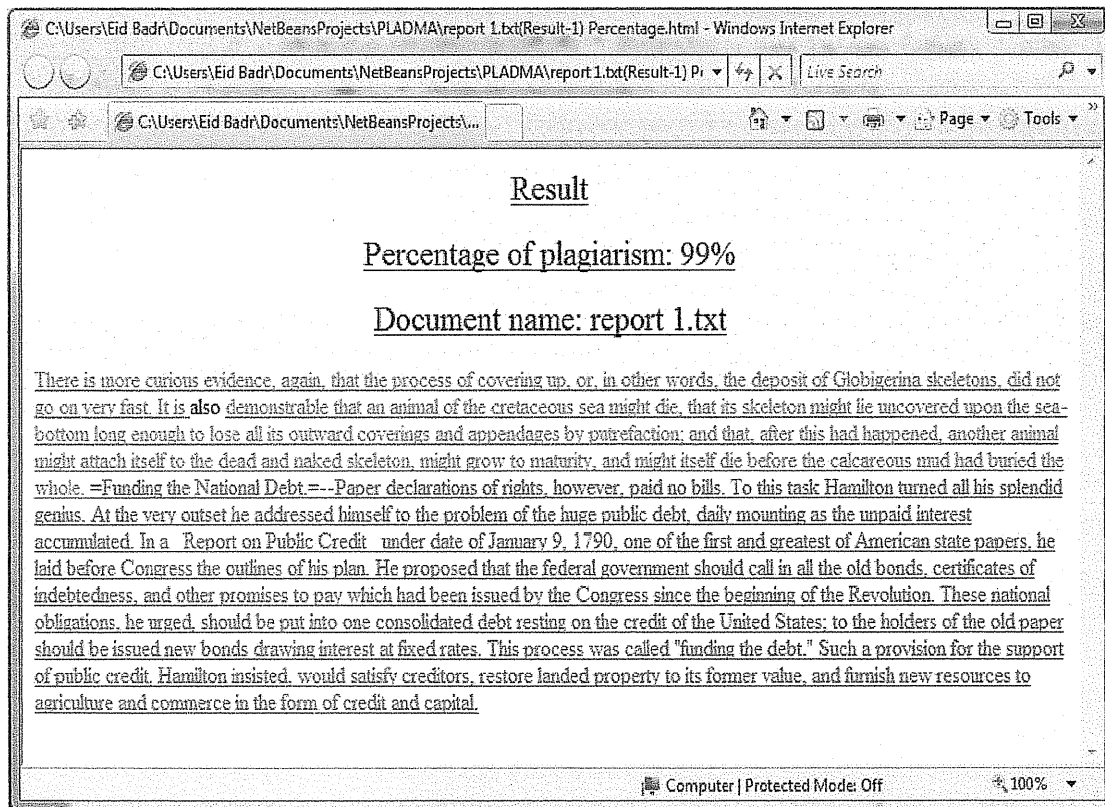


Figure 5.19. Report for document (report 1.txt).

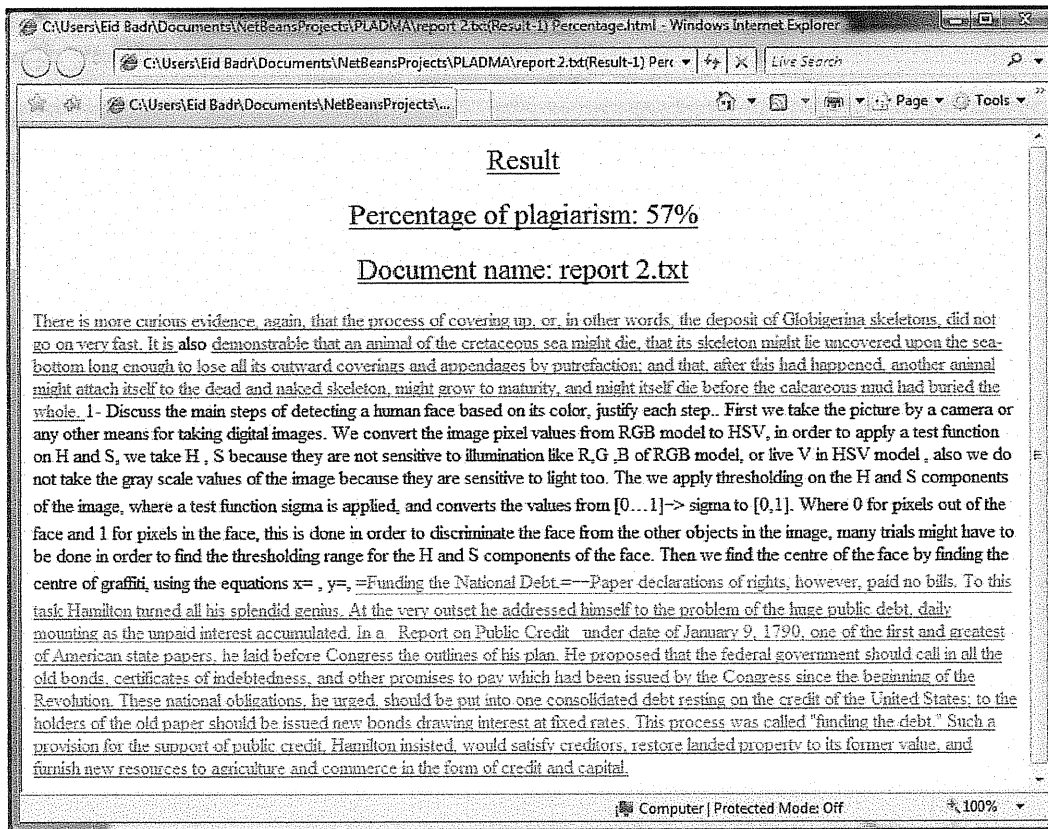


Figure 5.20. Report for document (report 2.txt).

5.3.1. Searching scenarios

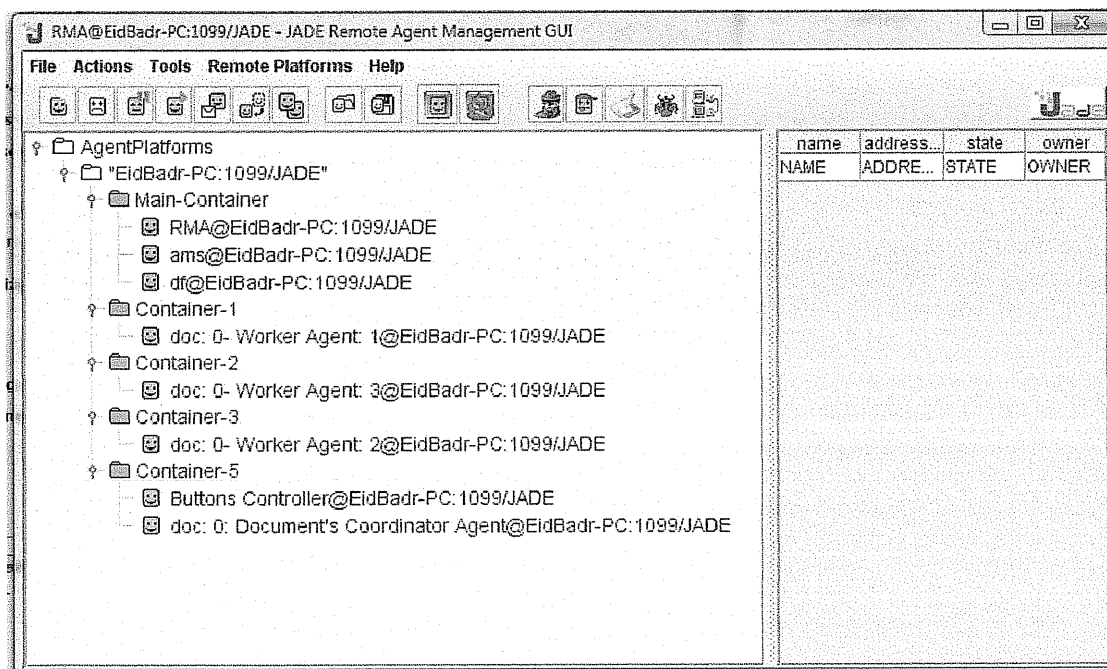


Figure 5.21. Parallel search scenario.

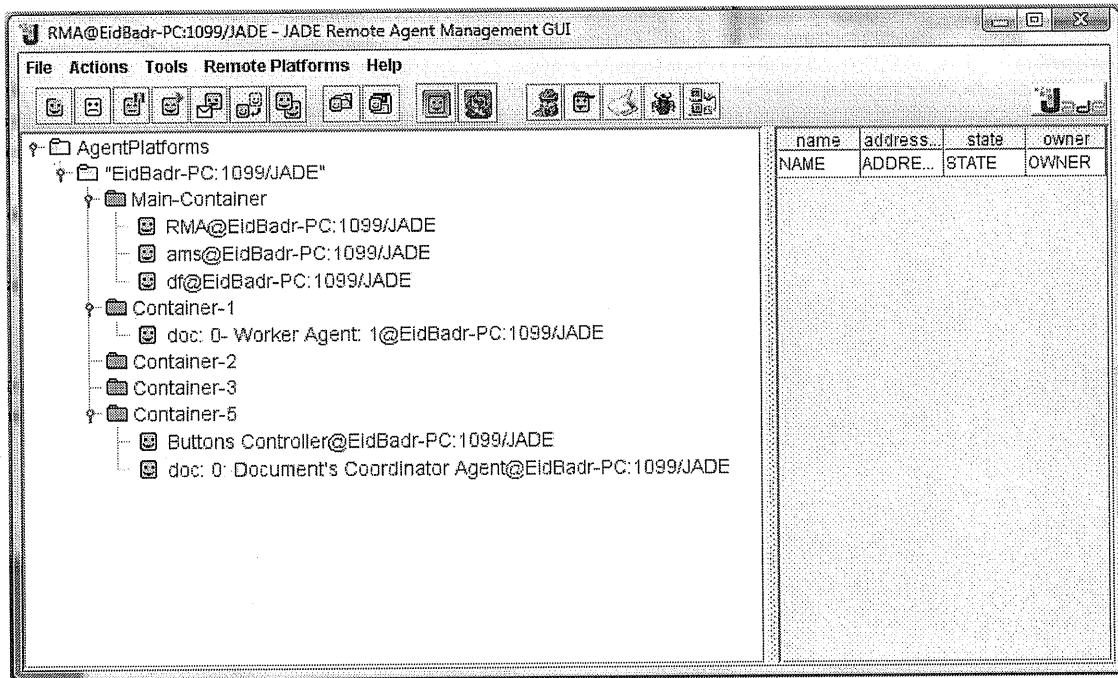


Figure 5.22. Sequential search scenario (part 1).

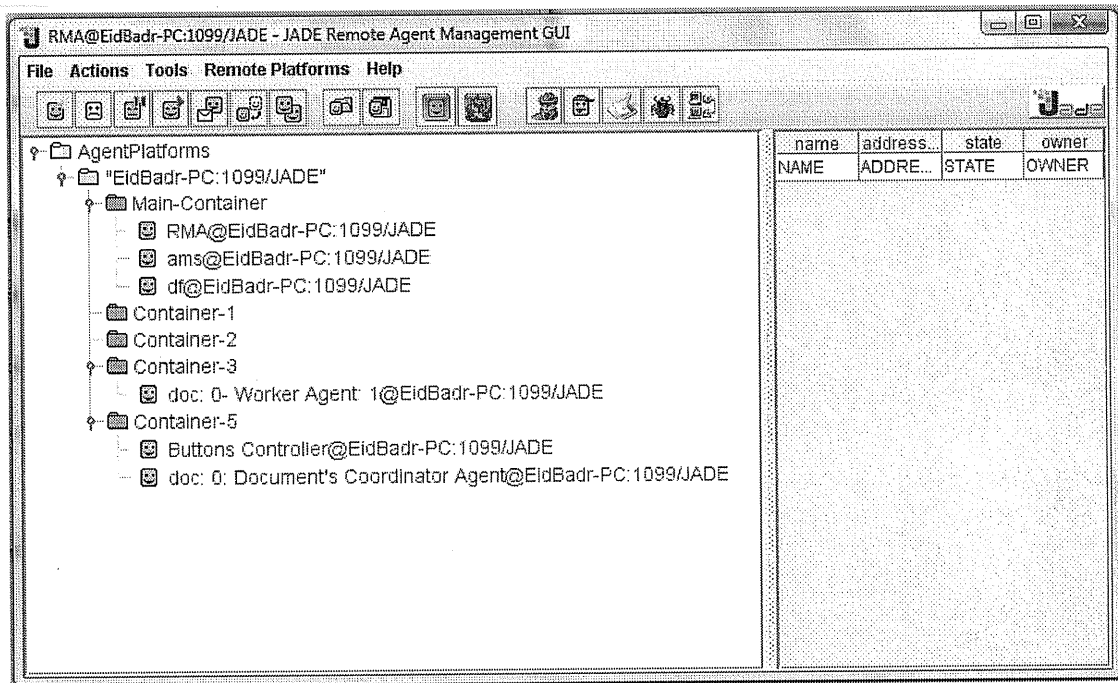


Figure 5.23. Sequential search scenario (part 2).

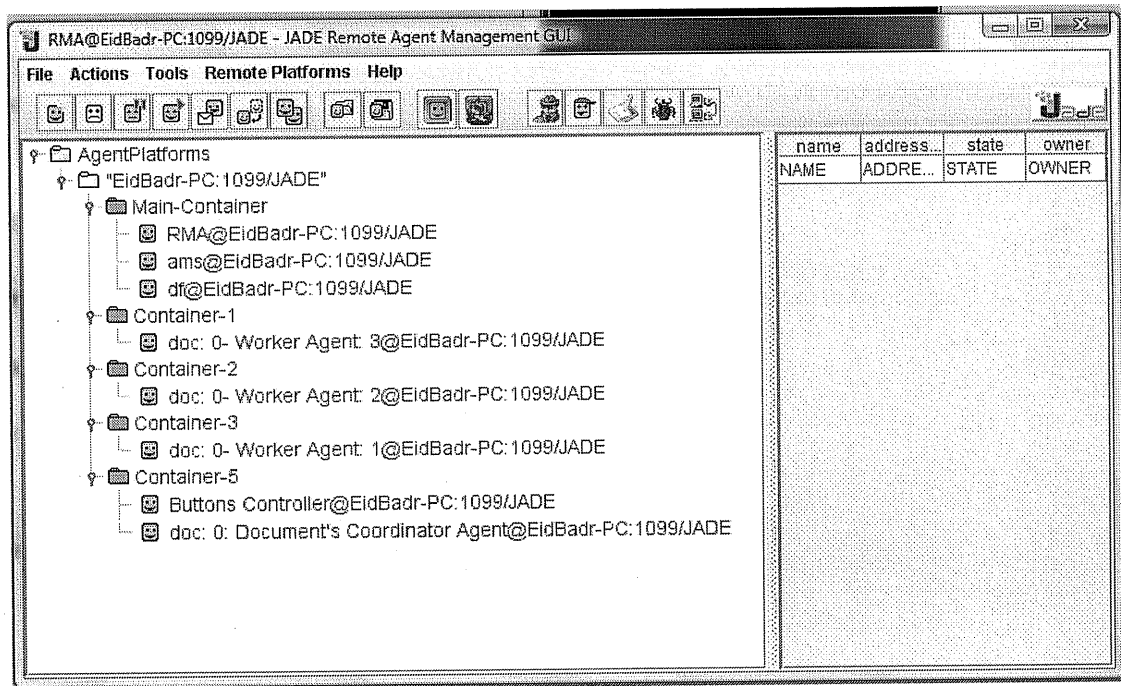


Figure 5.24. Distributed sequential and parallel search scenario (part 1).

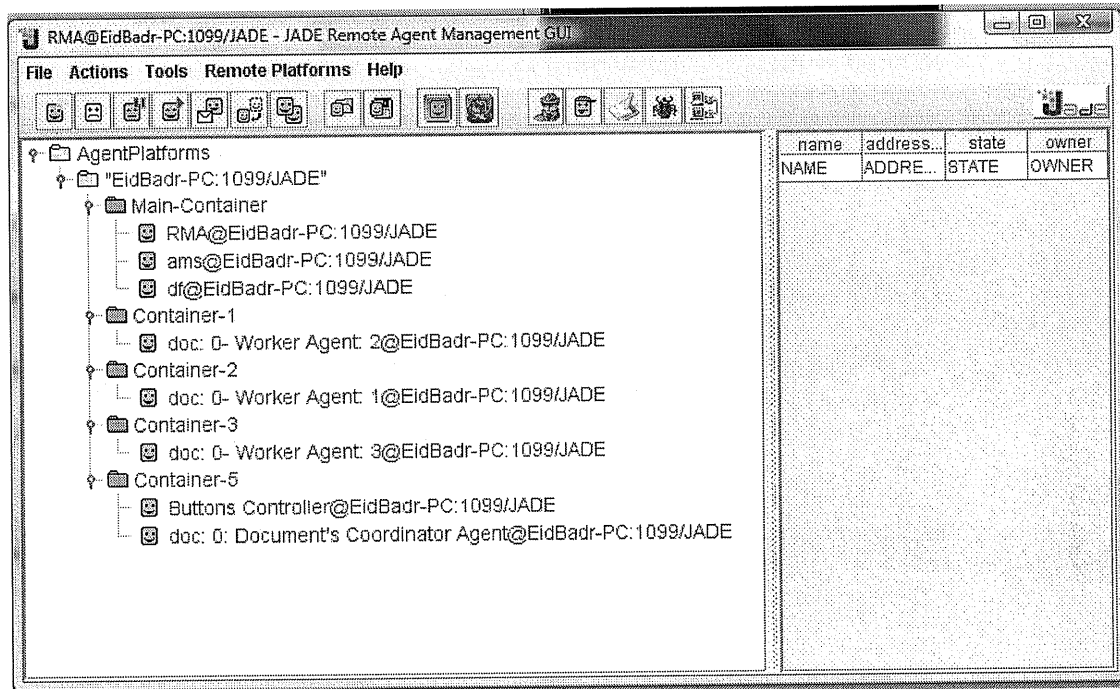


Figure 5.25. Distributed sequential and parallel search scenario (part 2).

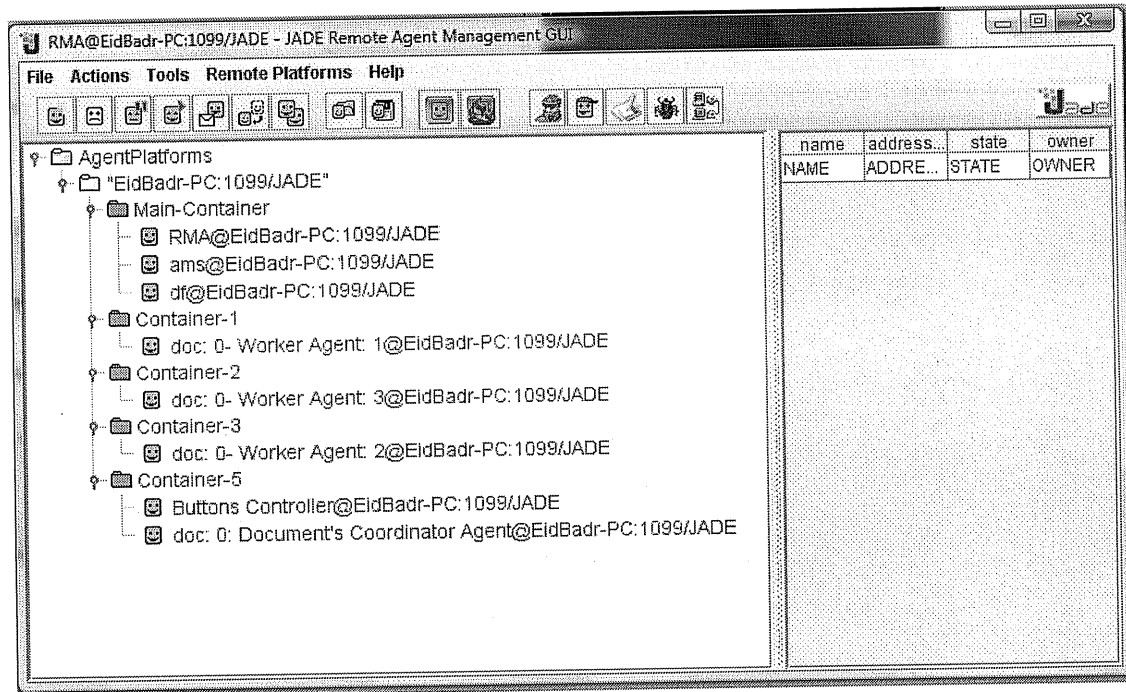


Figure 5.26. Distributed sequential and parallel search scenario (part 3).

5.3.5. Fault tolerance: Windowing size-1

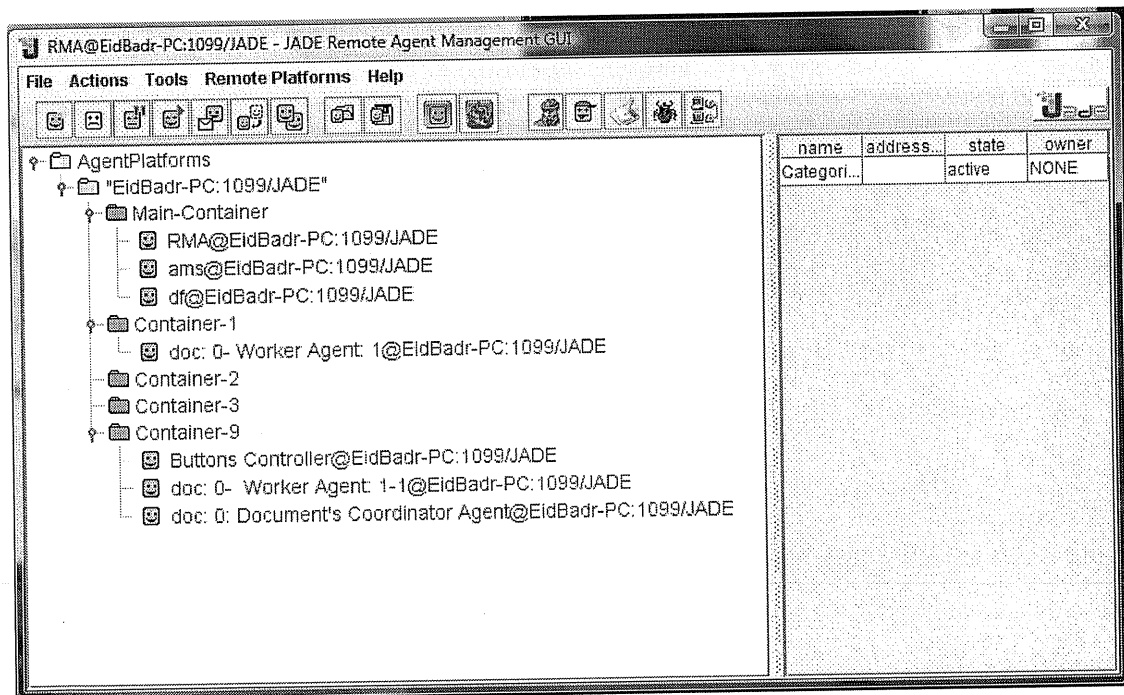


Figure 5.27. Windowing size-1: WA 1 is at the first container, WA 1-1 is at the home container.

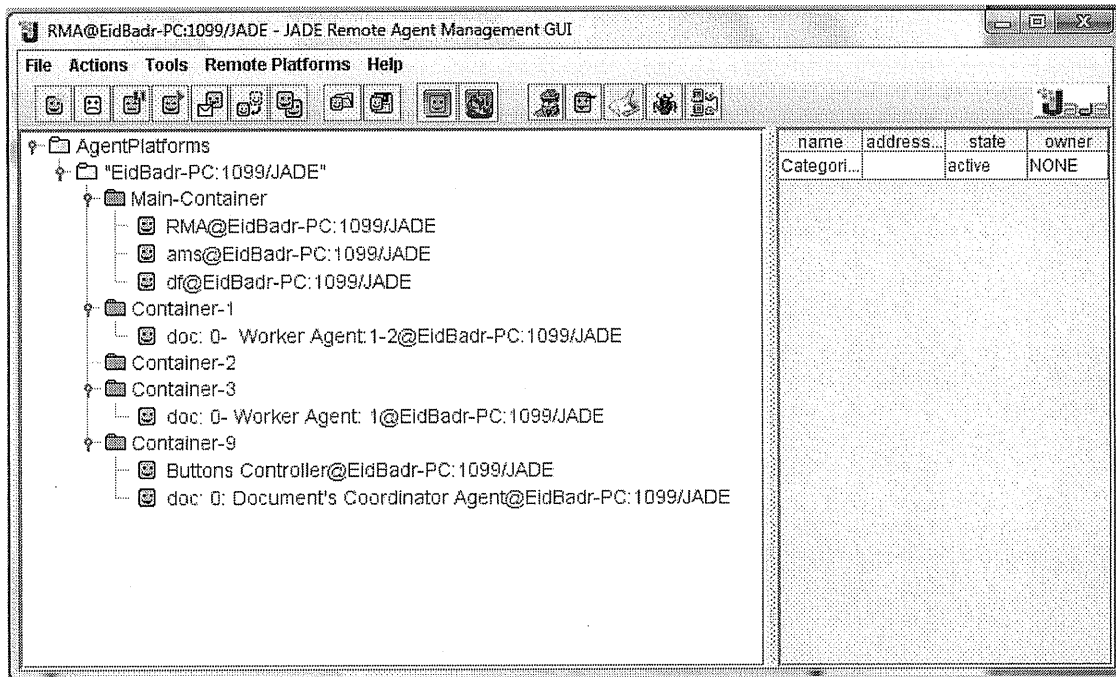


Figure 5.28. Windowing size-1: WA 1 is at the second container, WA 1-2 is at the first container.

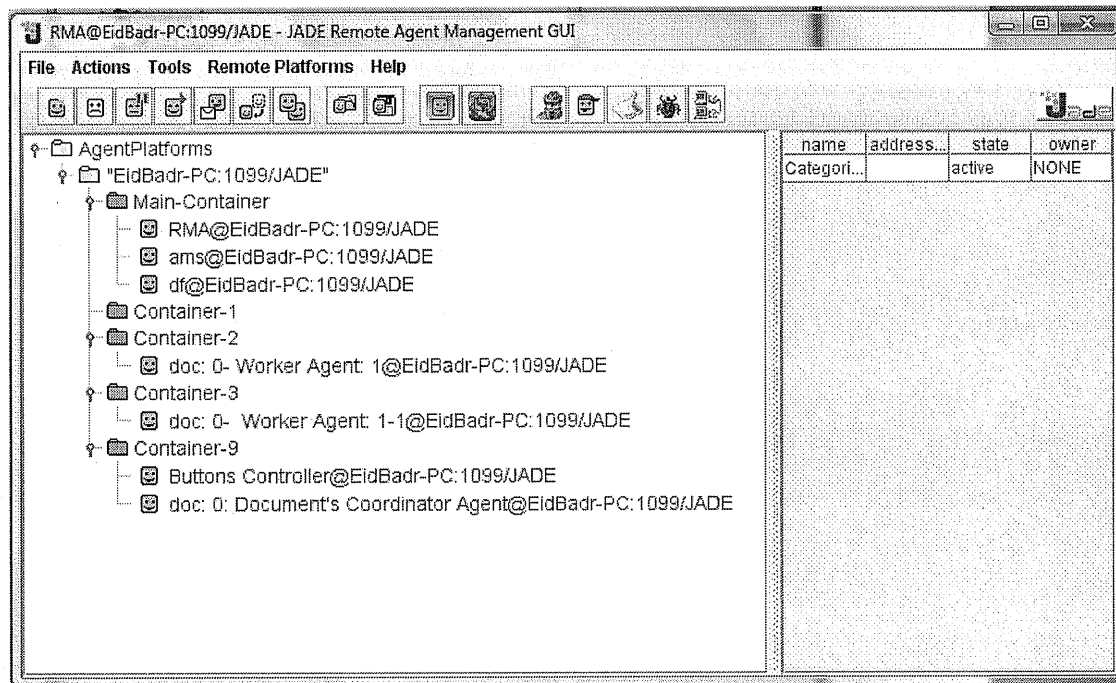


Figure 5.29. Windowing size-1: WA 1 is at the third container, WA 1-1 is at the second container.

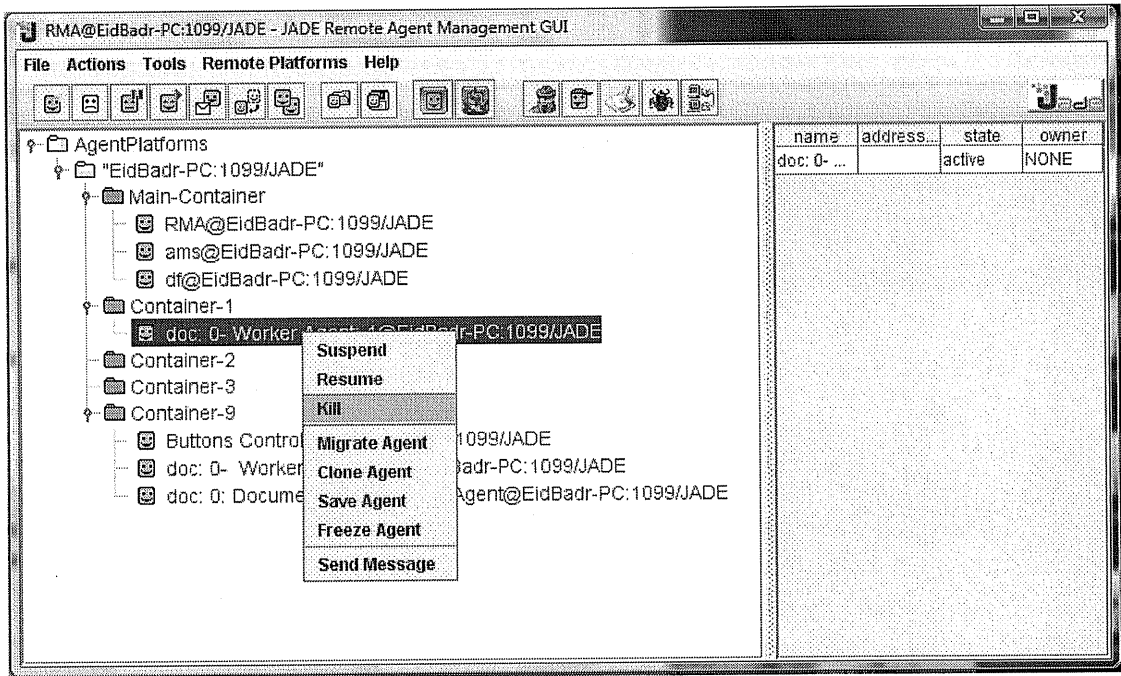


Figure 5.30. Windowing size-1: Killing of WA 1 while it's at the first container.

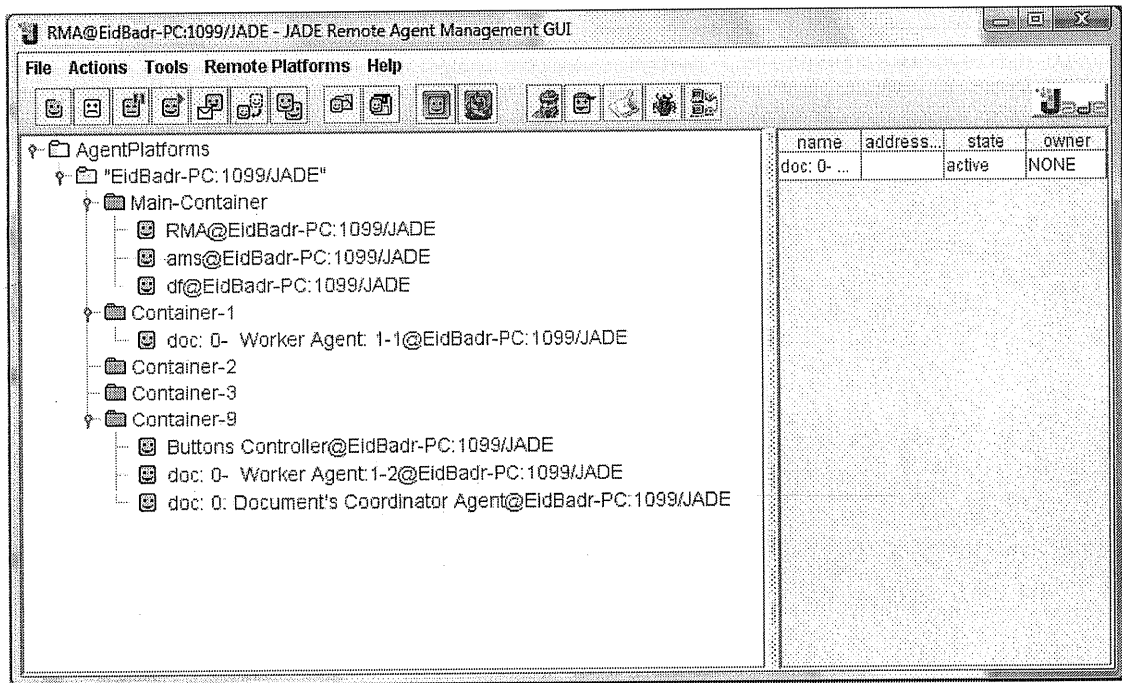


Figure 5.31. Windowing size-1: WA 1-1 is at the first container continuing WA 1's work.

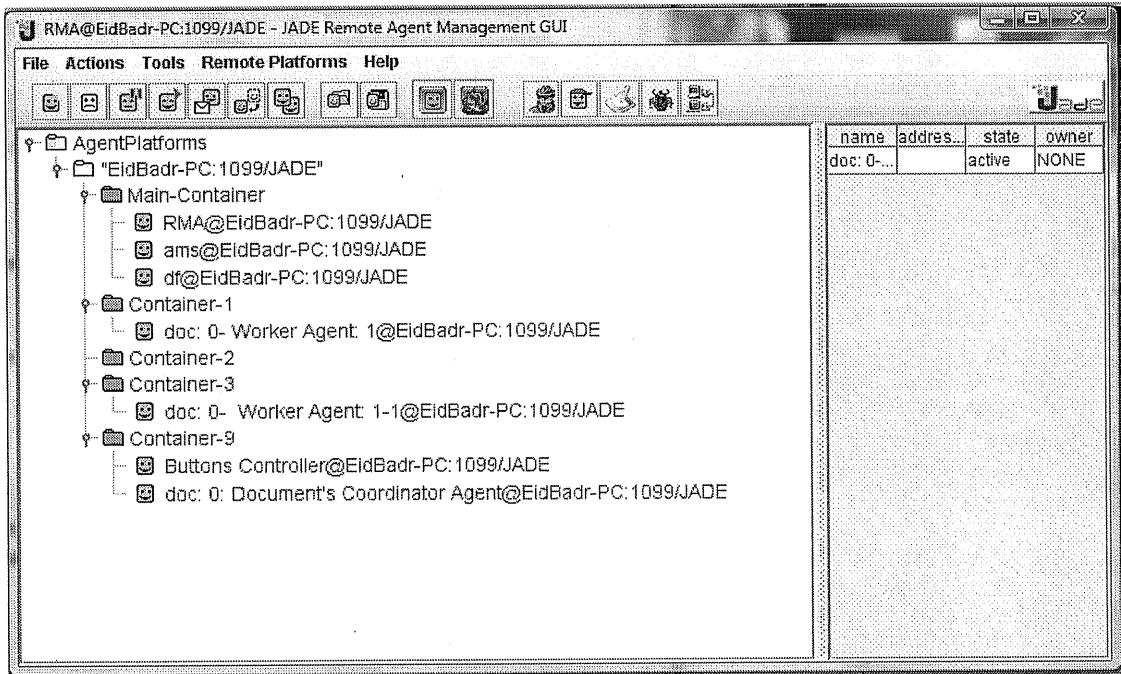


Figure 5.32. Windowing size-1: WA 1-1 is at the second container, WA 1 is at the first container.

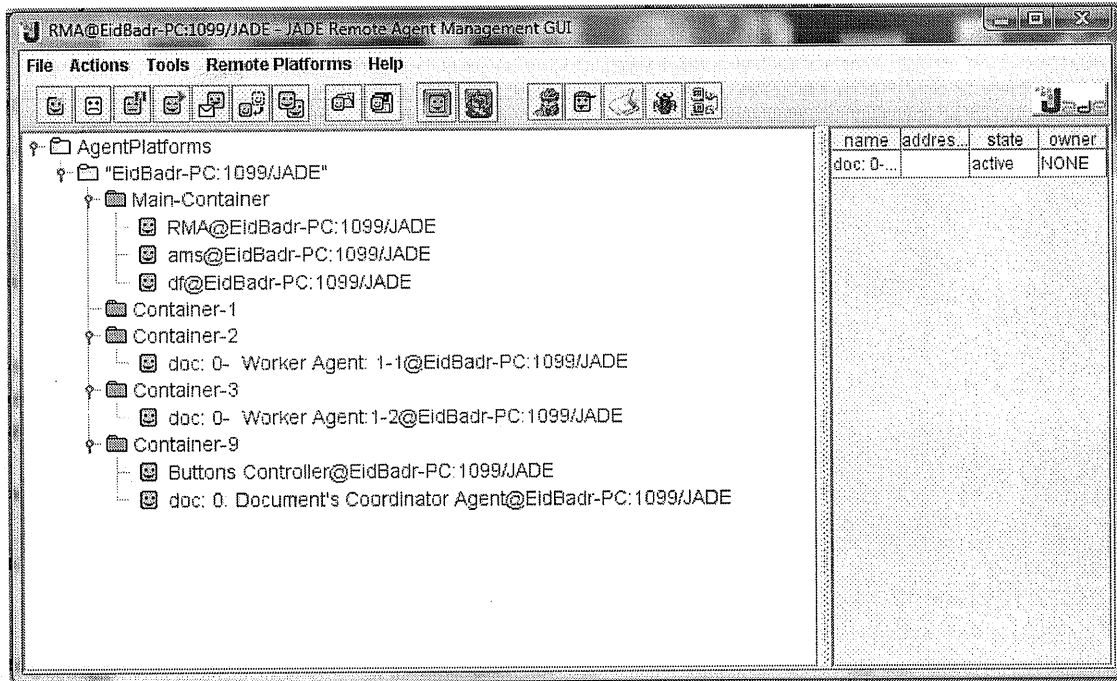


Figure 5.33. Windowing size-1: WA 1-1 is at the third container, WA 1-2 is at the second container.

5.3.6. Fault tolerance: Centralized

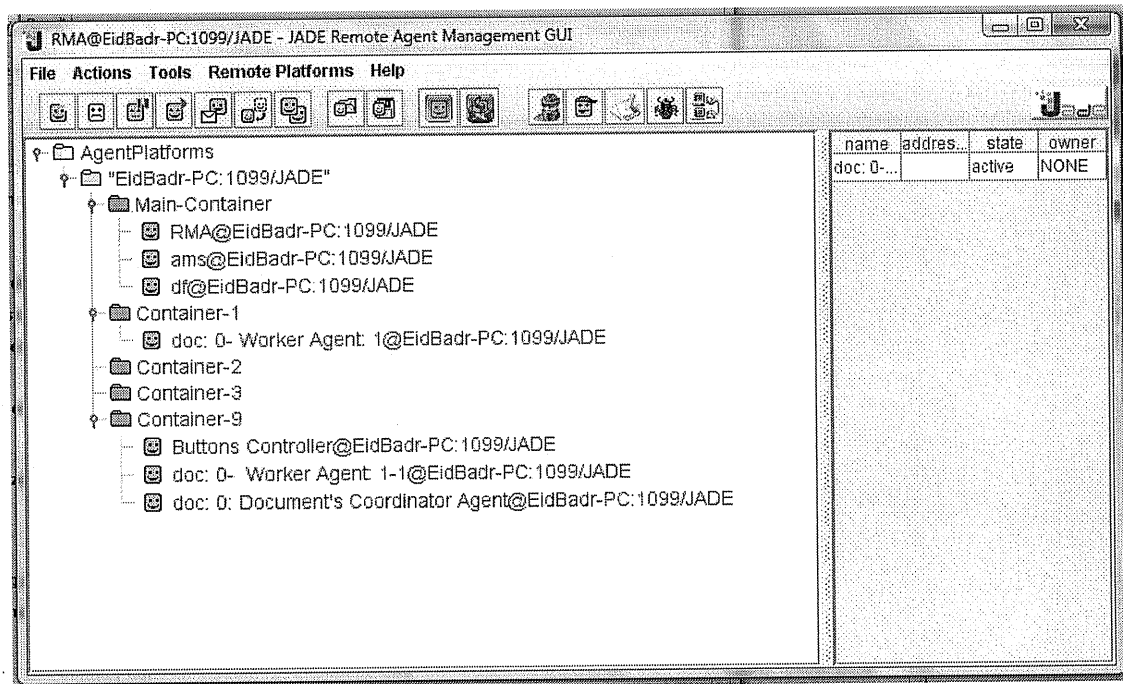


Figure 5.34. Centralized: WA 1 is at the first container, WA 1-1 is at the home container.

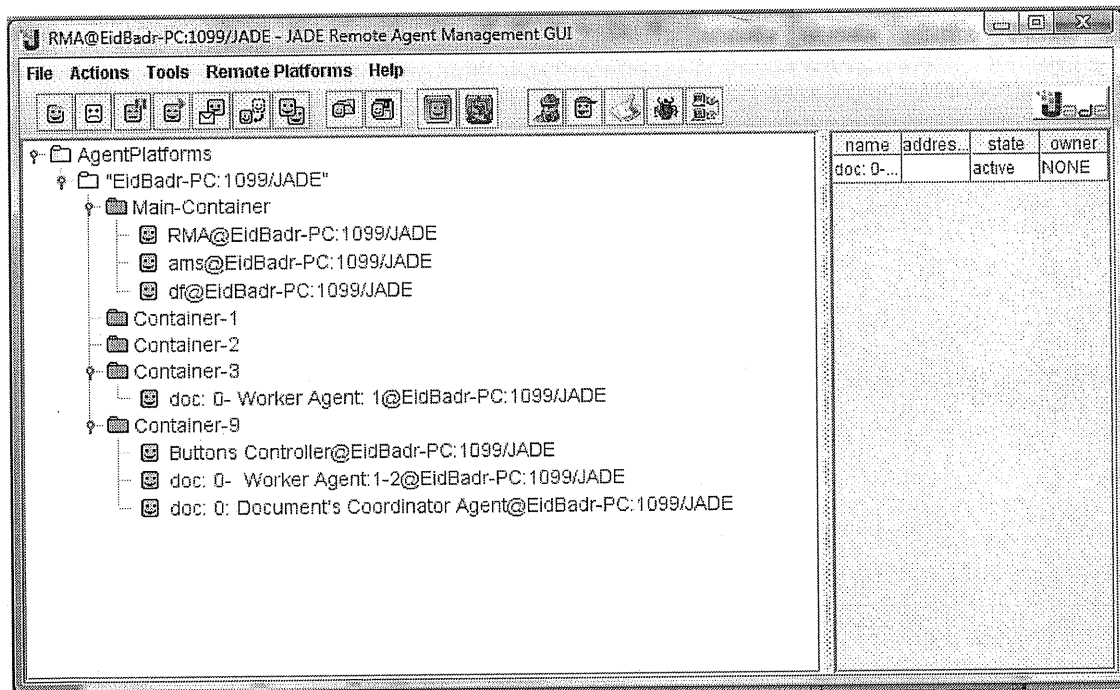


Figure 5.35. Centralized: WA 1 is at the second container, WA 1-2 is at the home container.

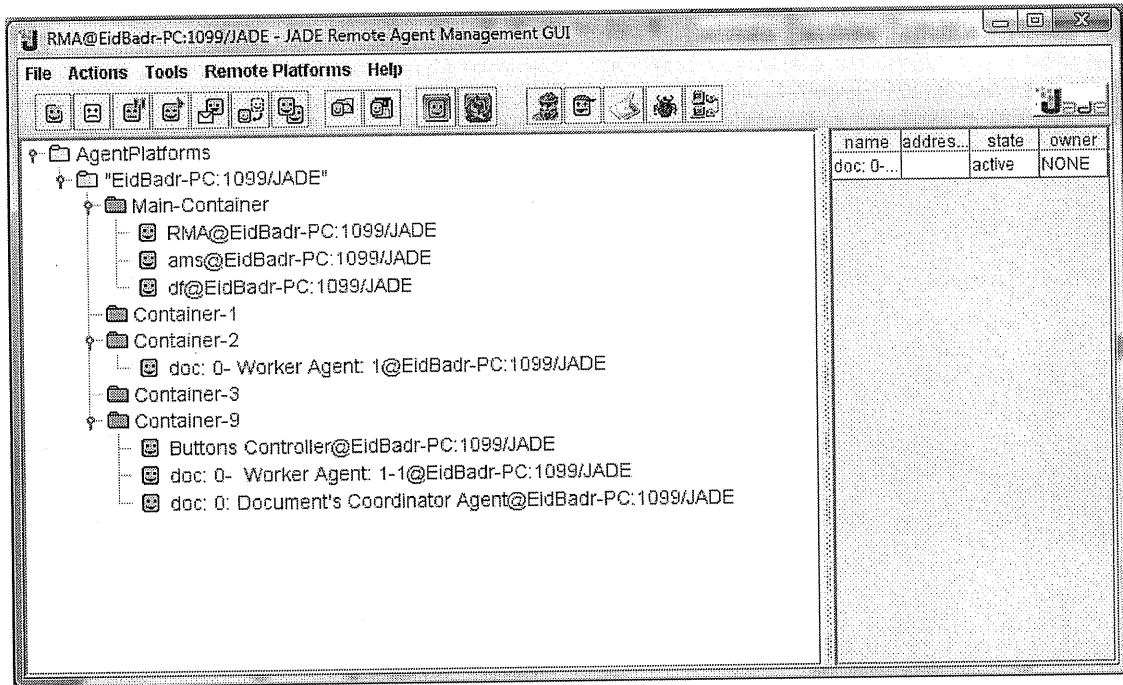


Figure 5.36. Centralized: WA 1 is at the third container, WA 1-1 is at the home container.

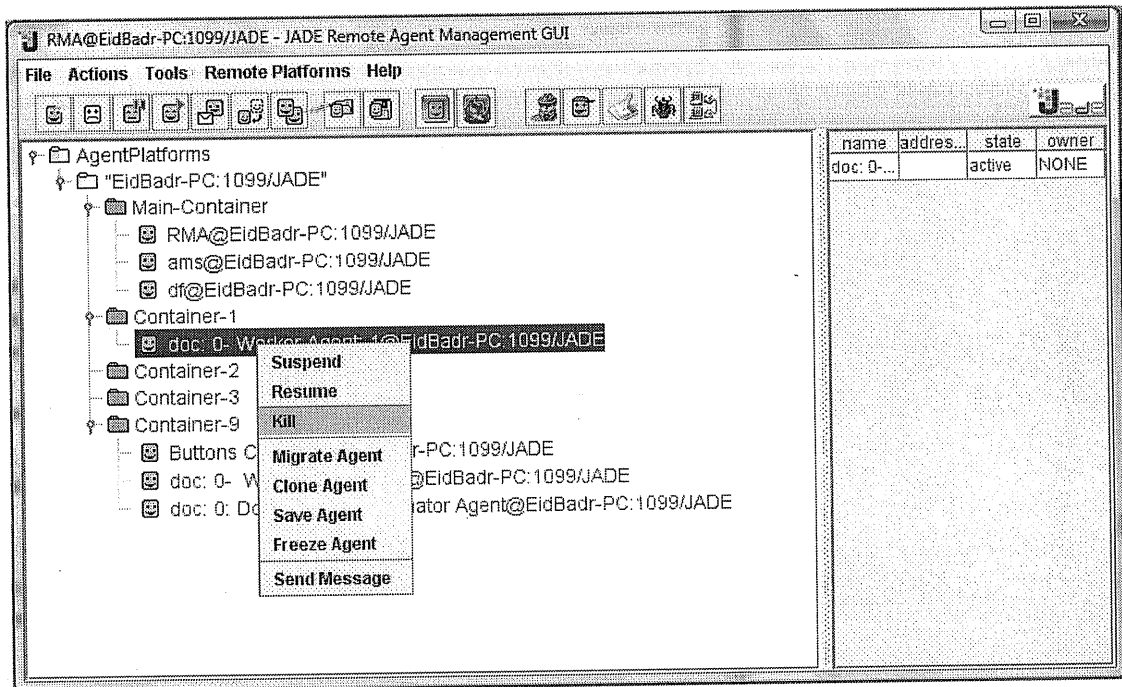


Figure 5.37. Centralized: Killing WA 1 while it's at the first container.

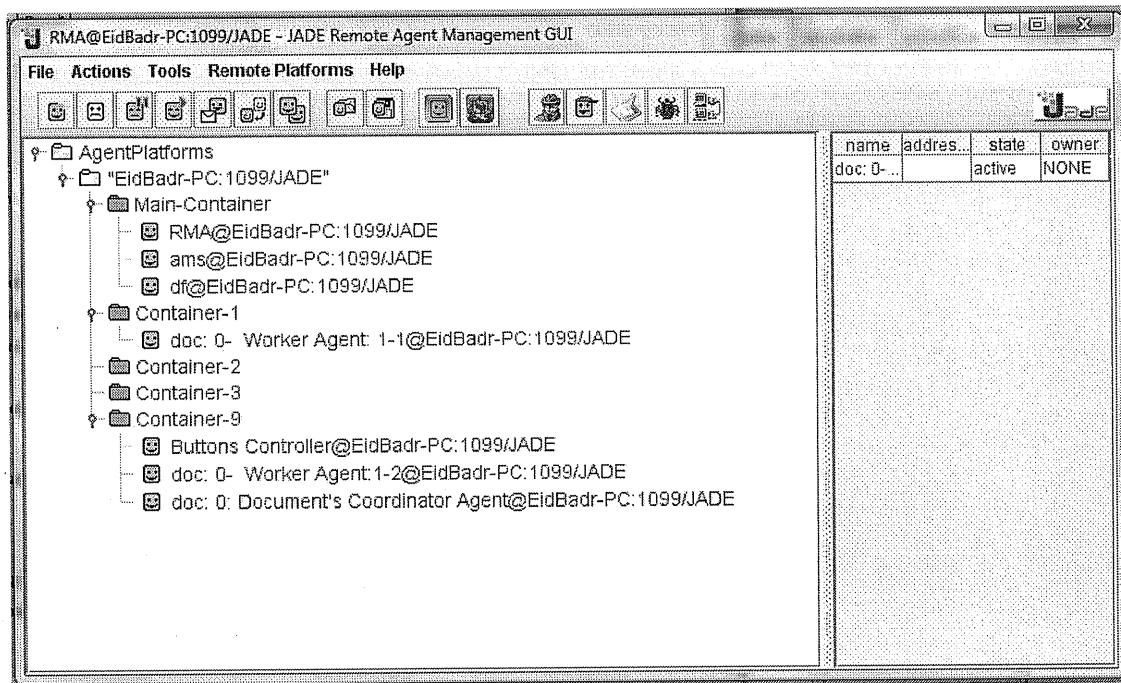


Figure 5.38. Centralized: WA 1-1 is at the first container continuing WA1's work, WA 1-2 is at the home container.

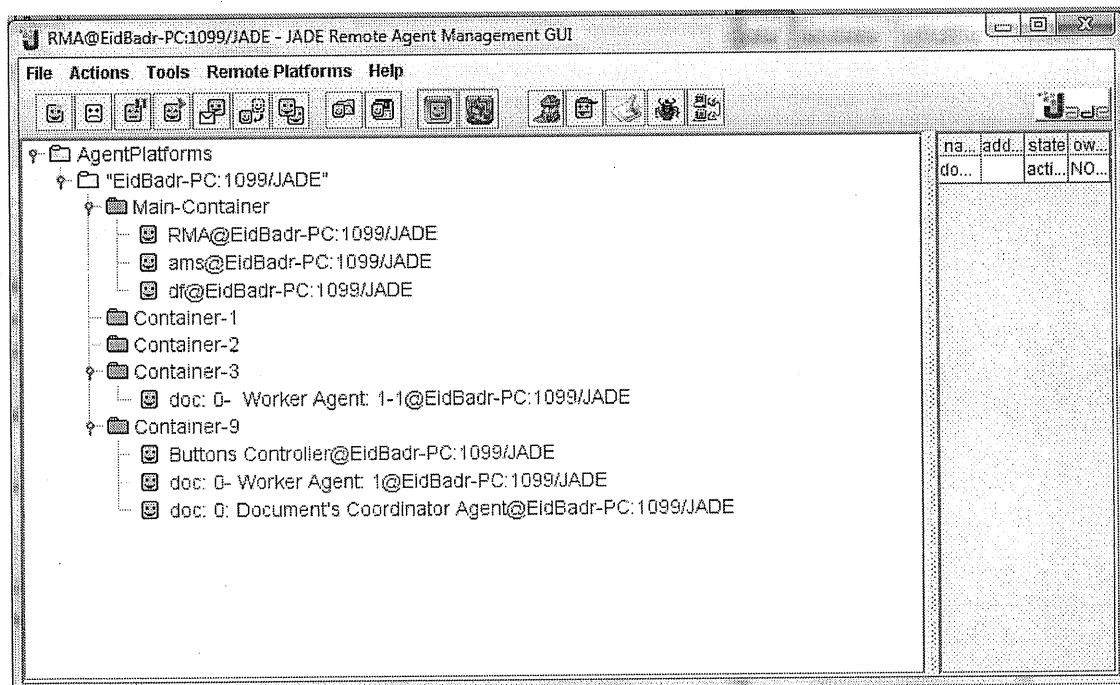


Figure 5.39. Centralized: WA 1-1 is at the second container, WA 1 is at the home container.

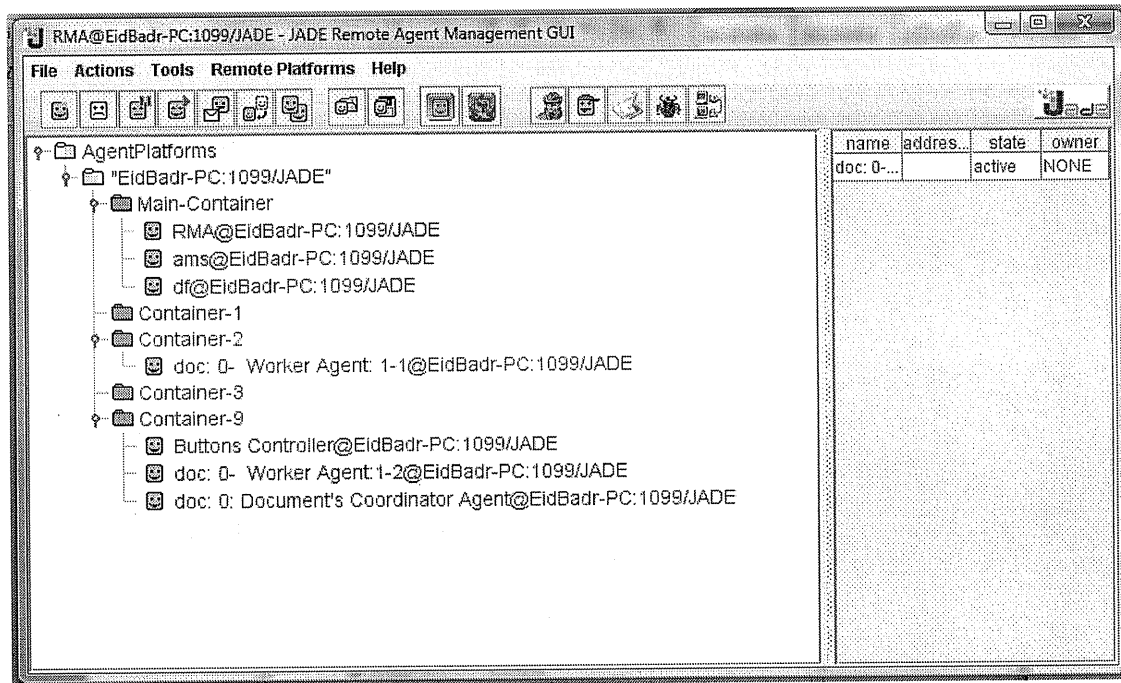


Figure 5.40. Centralized: WA 1-1 is at the third container, WA 1-2 is at the home container.

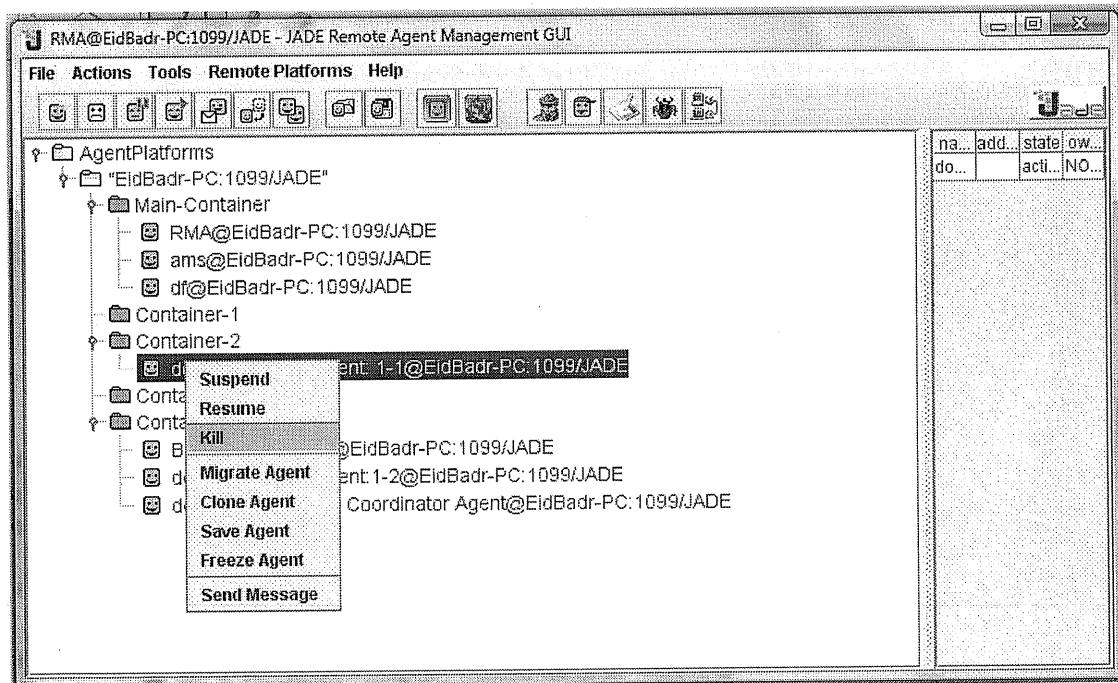


Figure 5.41. Centralized: Killing of WA 1-1 while it's at the third container.

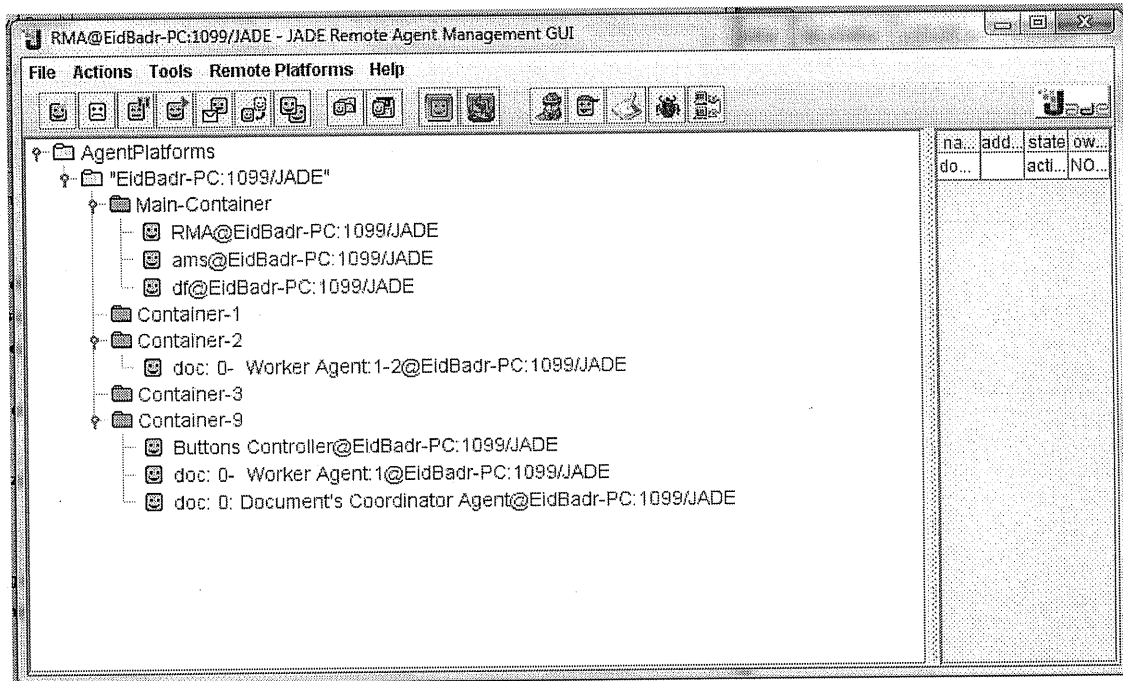


Figure 5.42. Centralized: WA 1-2 is at the third container continuing WA 1-1's work, WA 1 is at the home container.

5.3.7. Killing Containers

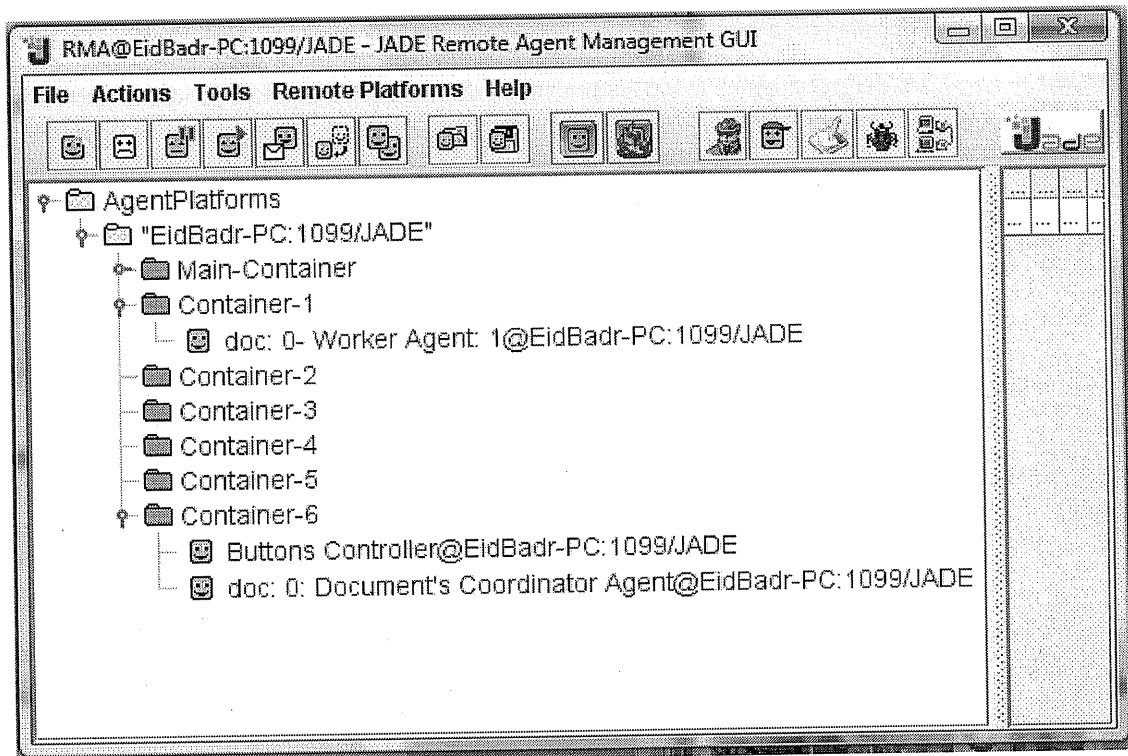


Figure 5.43. WA 1 is at the first container.

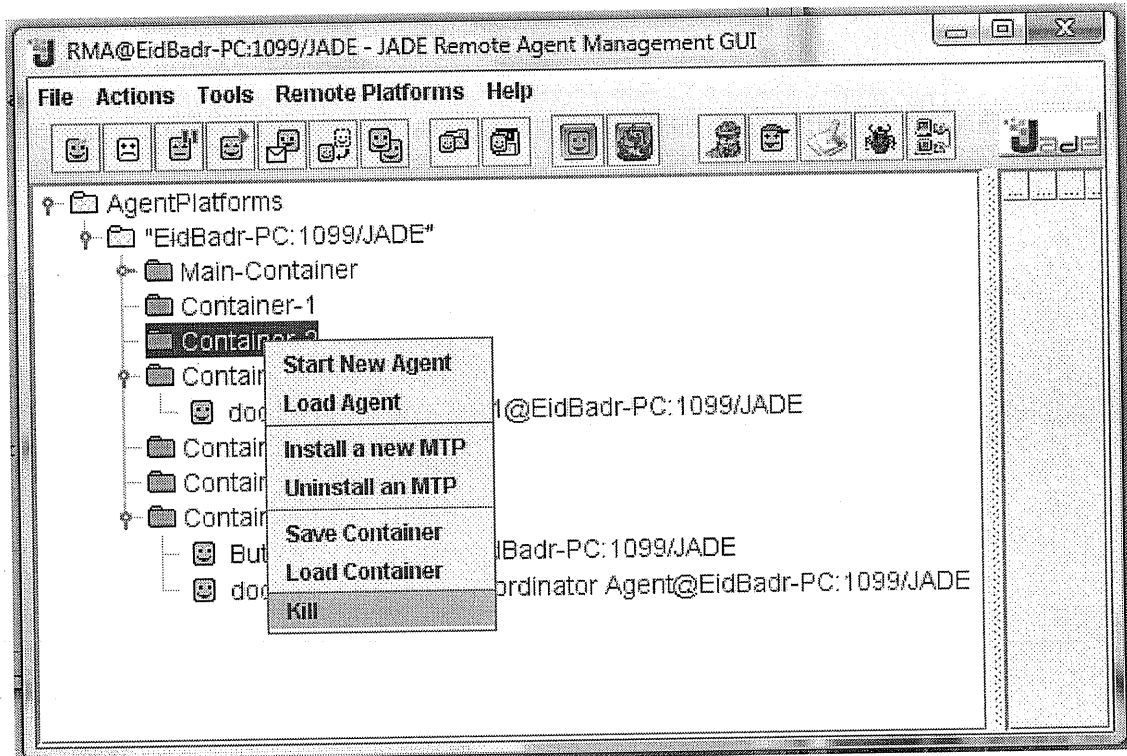


Figure 5.44. Killing of the third container.

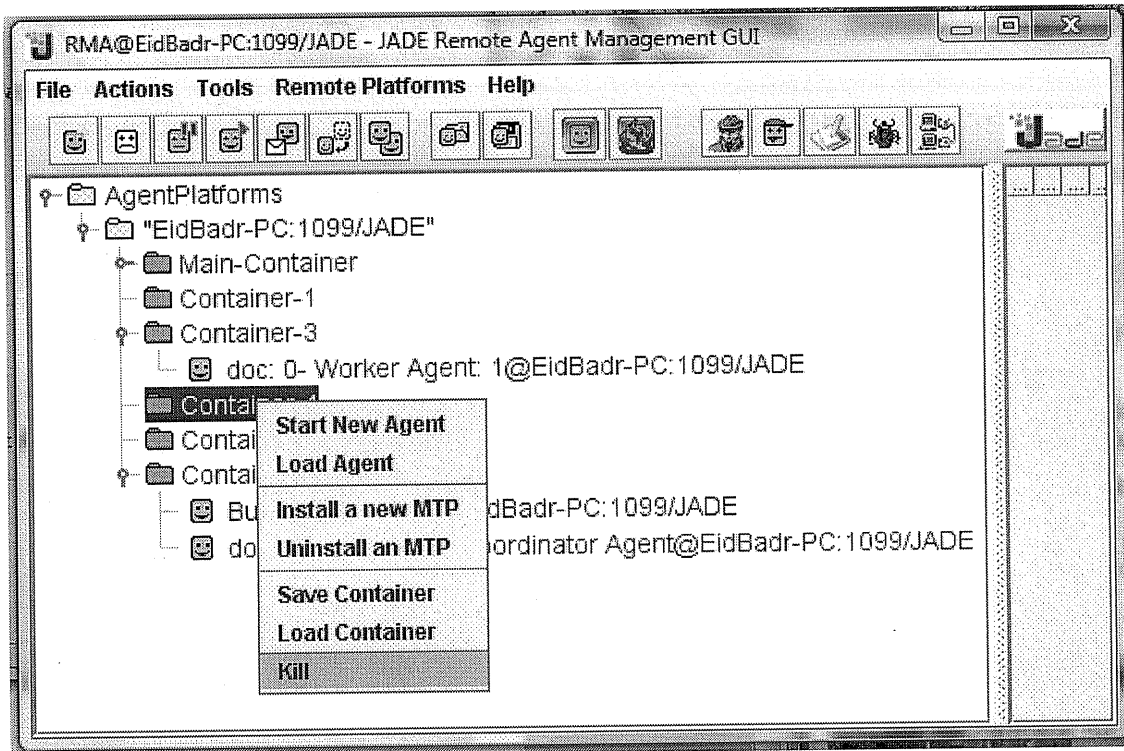


Figure 5.45. Killing of the fourth container.

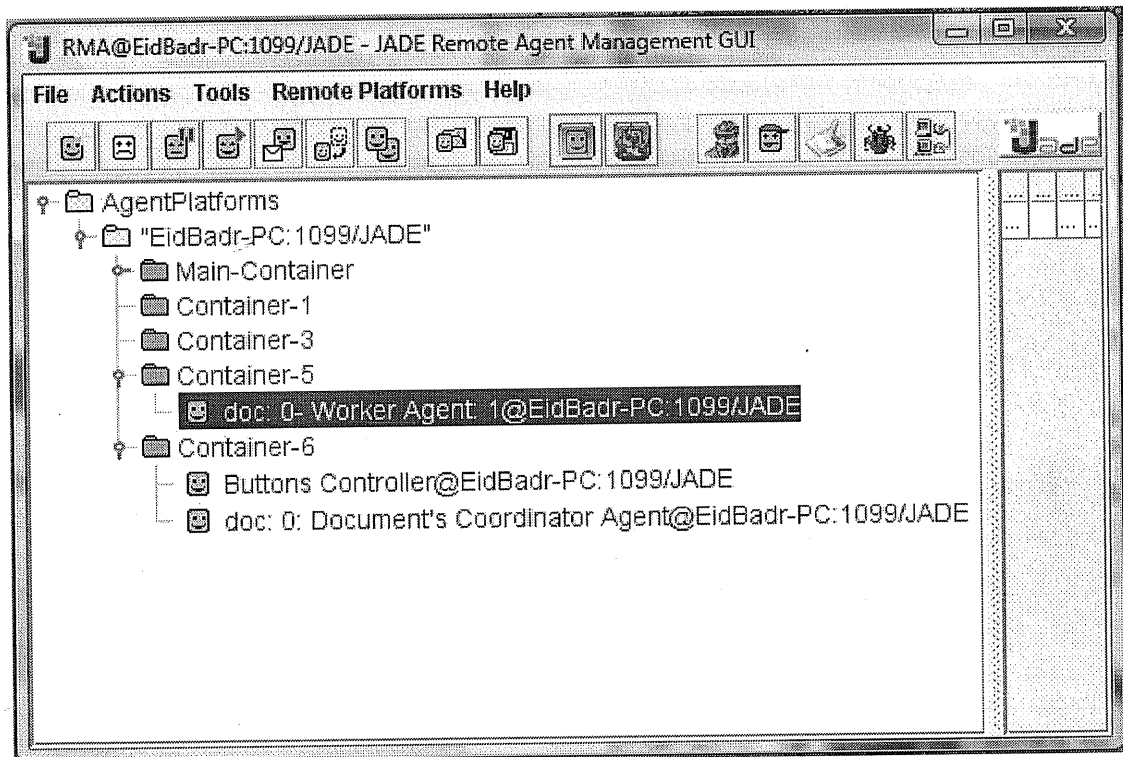


Figure 5.46. WA skipped the lost containers and moved to the fifth container.

5.4. Summary

In this chapter description of how the components of the system were implemented, testing of the system, development process and phases.

CHAPTER 6

CONCLUSION AND

FUTURE WORK

6.1. Experimental Results

6.2. Conclusions

6.3. Future Work

6.4. Summary

Chapter six

Conclusion and Future Work

In this chapter a description of the experiments that were done, conclusions concluded from working on the project, future work and improvements to be done on the system.

6.1. Experimental Results

In this section different comparisons for the performance of the system under different condition and using different scenarios, the test data was downloaded from the Project Gutenberg [45]; about 4 Giga bytes (GB) of text files was downloaded, and 6 experiments were done, hardware specification for each experiment in table 6.1.

Table 6.1. Hardware used in experiments.

Experiment 1	Computer 1: Core 2 duo, 2.66 GHz CPU, 2 GB RAM.
Experiment 2	Computer 1: Core 2 duo, 2.66 GHz CPU, 2 GB RAM.
Experiment 3	Computer 1: Pentium 4, 3 GHz (2 CPU), 512 MB RAM. Computer 2: Pentium 4, 3 GHz (2 CPU), 256 MB RAM. Computer 3: Pentium 4, 3 GHz, 512 MB RAM.
Experiment 4	Computer 1: Core 2 duo, 2.66 GHz CPU, 2 GB RAM. Computer 2: Centrino mobile, 1.7 GHz CPU, 512 MB RAM.
Experiment 5	Computer 1: Pentium 4, 3 GHz (2 CPU), 512 MB RAM. Computer 2: Pentium 4, 3 GHz (2 CPU), 512 MB RAM.
Experiment 6	Computer 1: Core 2 duo, 2.66 GHz CPU, 2 GB RAM.

6.1.1. Experiment one

Performance comparison using sequential search, with different number of agents working on the same document, each checking a different fraction of the document, on one digital library with signature files of total size (512 MB).

For example: in figure 6.1 the point (10, 550000) means 10 agents are working only on one document with size 500 KB, all of them are checking files on the same digital library, and the time needed for all of them to finish working is 550000 ms, here each

agent is in charge of checking 1/10 of the total number of the documents, this does not mean that each agent is working on the same size of data, where different files have different sizes.

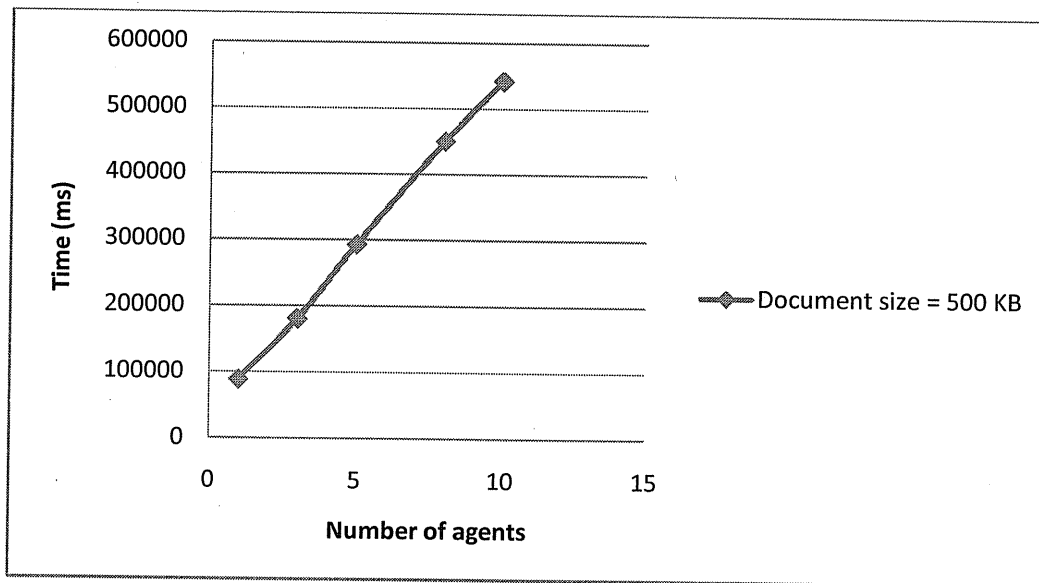


Figure 6.1. Number of agents vs. Time

From this results we can conclude that the best case in sequential search is to let one agent work on a document, that is each document submitted by the user should be assign to one agent to work on it, letting more agents work on the document does not only will increase linearly the time needed to check the document, but also increases the size of the overall traffic used on that document, that is for example having 10 agents working on a document will duplicate the size by 10, which will put more load on the network.

6.1.2. Experiment two

Performance comparison using sequential search, with n documents, n agents, each agent working on one document, on one digital library with signature files of total size (512 MB).

The three curves in figure 6.2 represent the test using 1, 2, and 3 agents; that is 1, 2, and 3 documents of different sizes.

For example the curve with triangles at point (800, 590000) means the user submitted three documents; each document is assigned to one agent, and the three of them are working on the same digital library, document size is 800 KB, and the time needed for all the agents to finish working is 590000 ms.

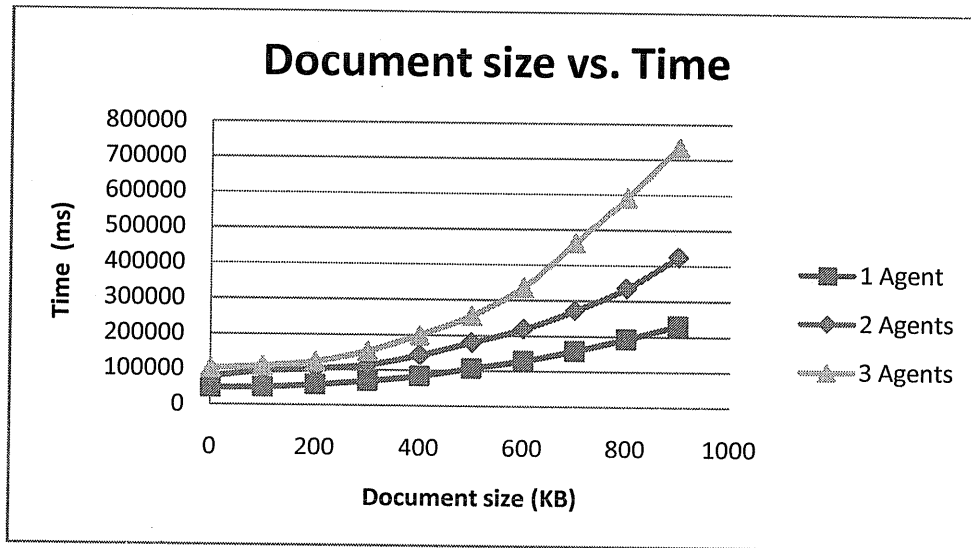


Figure 6.2. Experiment two using 1, 2, and 3 agents.

Figure 6.3. shows the results of the experiment, using 1 and 2 agents, and the results of 1 agent multiplied by 2, having 2 agents working does not mean duplication of the time of using 1 agent, the curve that represents the duplication of the time needed for one agent, is not equal to the curve that represent the experimant with 2 agents.

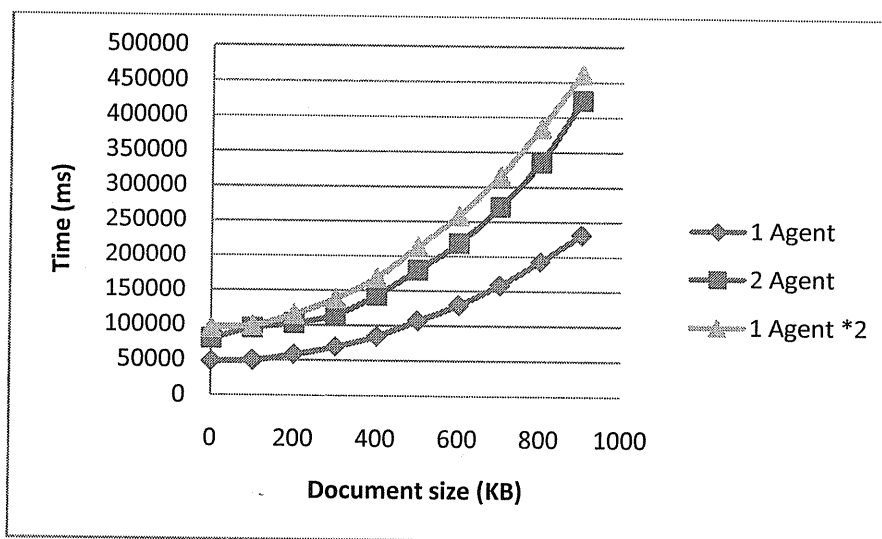


Figure 6.3. 1 Agent vs. 2 Agents.

Figure 6.4. shows the results of the experiment, using 1 and 3 agents, and the results of 1 agent multiplied by 3, until document size 750KB having 3 agents working gave a result with less than 1 agent working multiplied by 3, after 750 KB this is inverted, 3 agents working gave a higher time value from 1 agent *3, so to some size of the document (here 750 KB), increasing the size of the documents will not affect the performance, however, after that point performance will degrade.

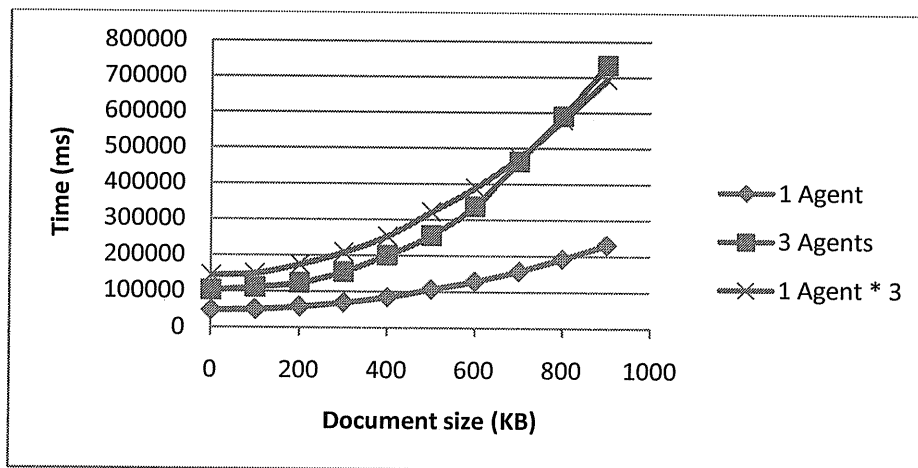


Figure 6.4. 1 Agent vs. 3 Agents.

6.1.3. Experiment three

performance comparison between parallel search and sequential search; three digital libraries was setup, with sizes: 512 MB, 439 MB, and 234 MB, for parallel search three agents was activated, each going to a digital library, for sequential search one agent was activated going to all the digital libraries in sequential order.

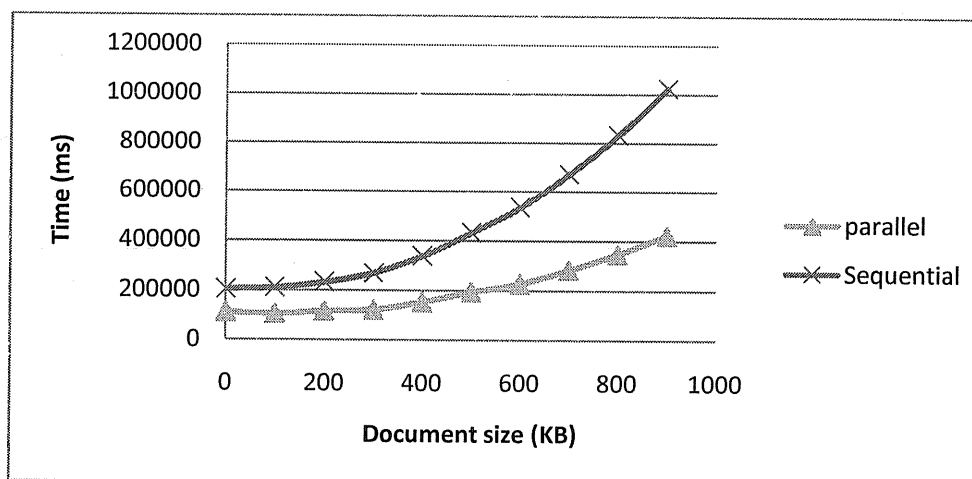


Figure 6.5. Sequential search vs. parallel search.

From this experiment, we conclude that, parallel search depends on the size of largest digital library, in figure 6.5 the line for the parallel search represents the time needed to check the largest digital library (512 MB), sequential search depends on the overall size of all digital libraries, however in parallel search the number of agents activated depends on the number of digital libraries available in the system, that is n digital libraries, will require n agents; this means more load is put on the network, whereas in sequential search only one agent is working, putting the load of only one agent on the network.

6.1.4. Experiment four

Performance comparison between a LAN (100 Mbps) and PPU WAN (2 Mbps) (see figure 6.6) using one agent checking different documents with different sizes, in this experiment two digital libraries was setup at Abu Roman (AR), and another one at Wadi Al-Hariya (WH), and made the agent go from WH to AR back to WH then to AR an finally send the result to WH.

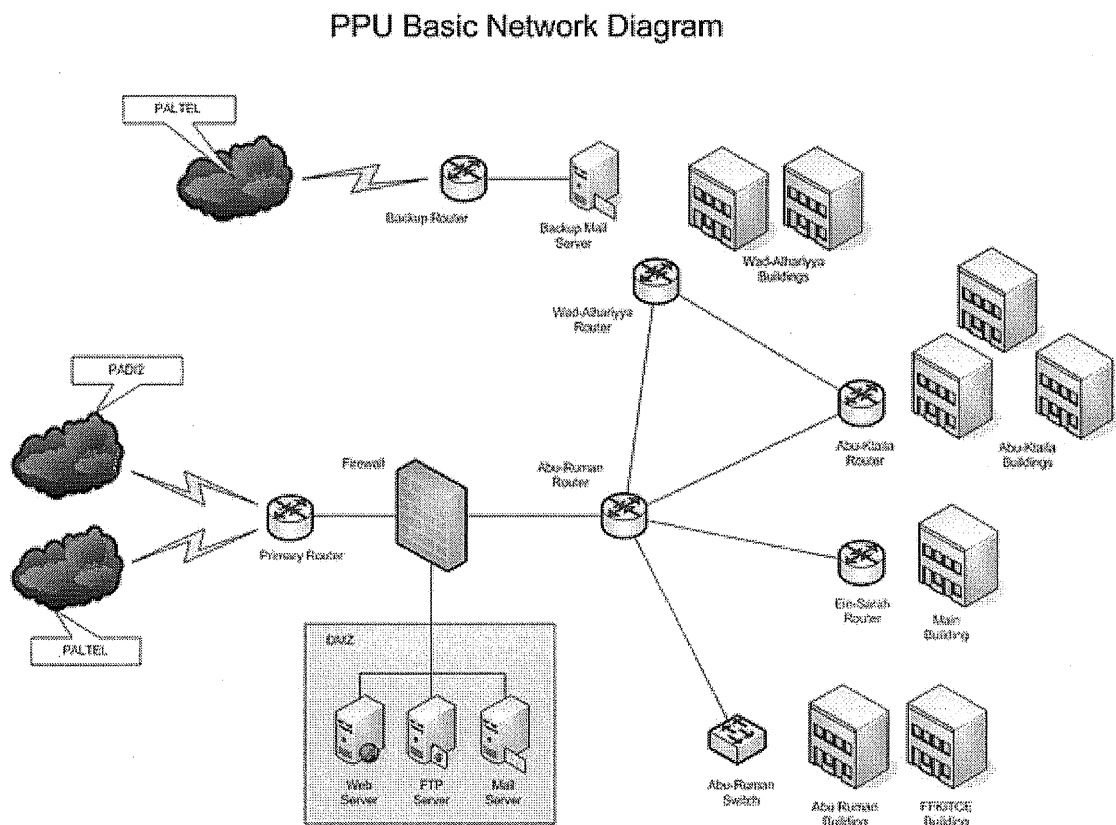


Figure 6.6. PPU WAN [49].

Then the same experiment was repeated on a LAN using a switch see figure 6.7.

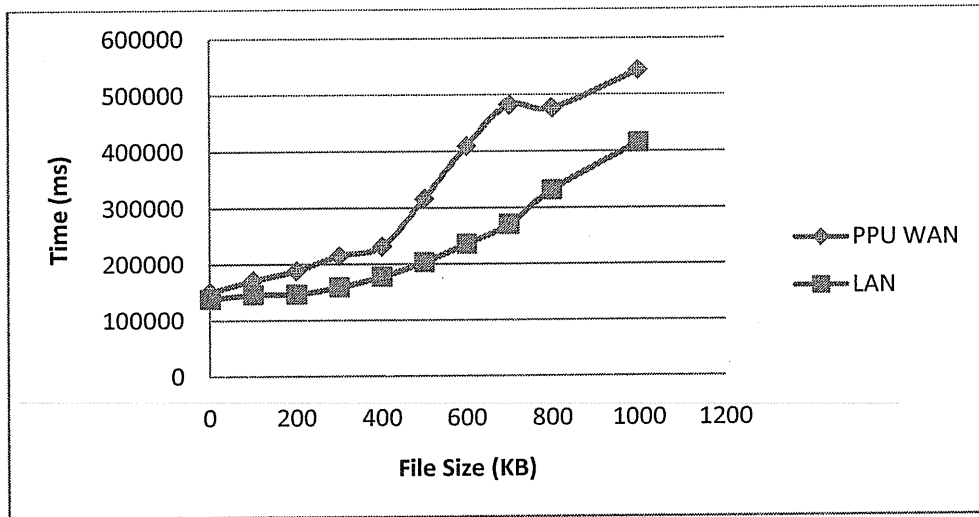


Figure 6.7. LAN vs. PPU WAN.

From this experiment, we found that using a WAN, after reaching a file of size 400 KB the performance degraded with about 30%, also different traffic that is done between different computers connected on the PPU WAN, made some variations on the performance, that made less sized document (700 KB) takes longer time to be processed than higher sized document (800 KB).

6.1.5. Experiment five

Performance comparison between parallel search, and distributed search with different overlapping times see figure 6.8.

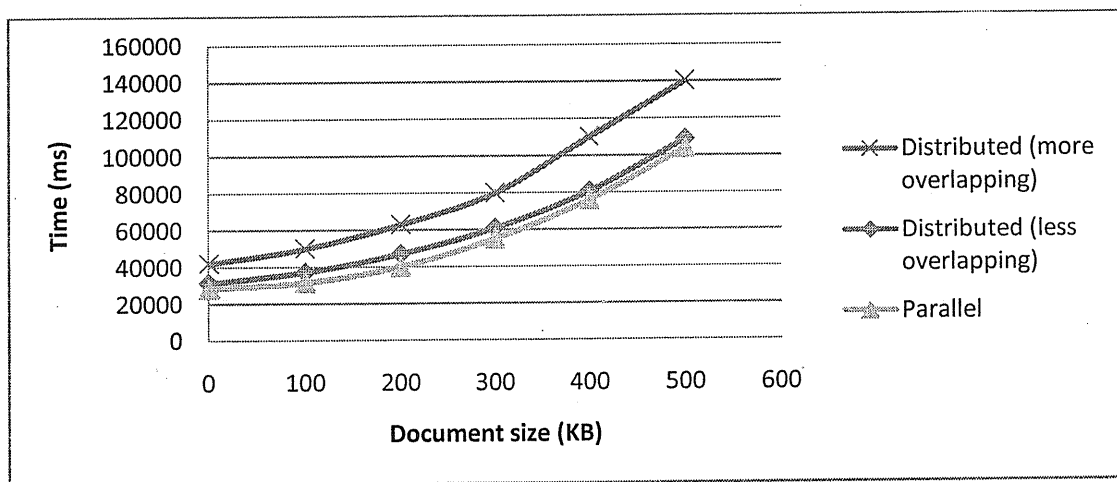


Figure 6.8. Parallel search vs. distributed search.

From this experiment, distributed search with small overlapping of agents gives approximately the same performance of parallel search, however, increasing the overlapping time will increase the gap between distributed and parallel search.

6.1.6. Experiment six

Ratio between size of input documents and size of files resulted from generation of signatures, different sized documents was submitted to the system, and we noted the resulted signature files in two cases, using hashing of chunks and with no hashing.

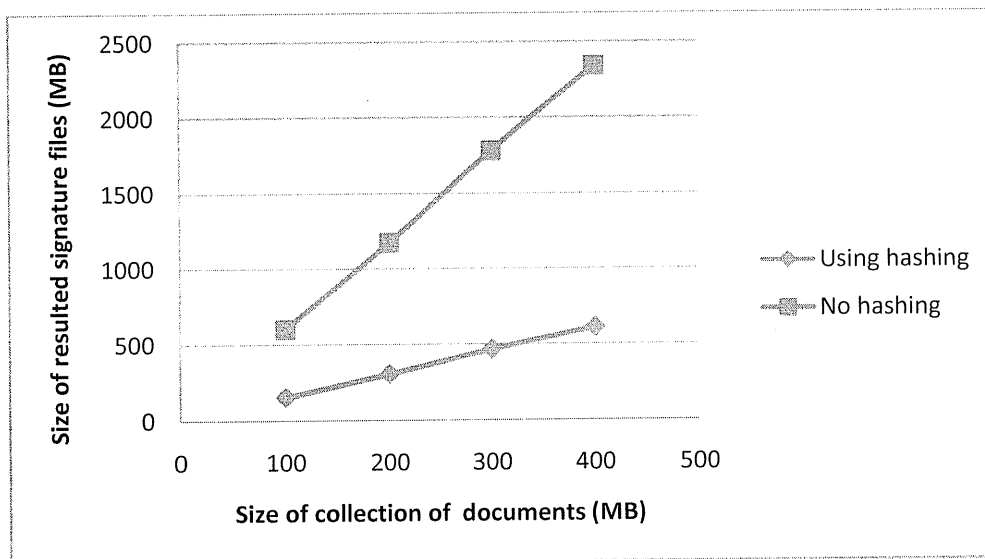


Figure 6.9. Size of input documents vs. size of signature files.

From figure 6.9. when using hashing, the size of signature files increases linearly with increasing the size of the input documents with a factor of about 1.528, however without hashing the curve changes for different sized of document, this is probably because when using hashing, the signatures generated are the same size, (32 bit value), but with no hashing different chunks have different sizes depending on the characters in the chunk.

In this experiment we also noted the time needed to generate the signatures files, from figure 6.10 when using hashing the curve acts as a polynomial of power 2, this is probably because of the operation of generating hash values, and conversion to hexadecimal, however, with no hashing time increases linearly with increasing the size of the input documents, at first (100MB to 350MB) we notice the effect of the size of the chunk that needs more time in writing the chunks into file, but later (after

350MB) the effect of signature generation and hexadecimal conversion appears that makes time needed to generate hashes superior to the time needed to write signatures into memory.

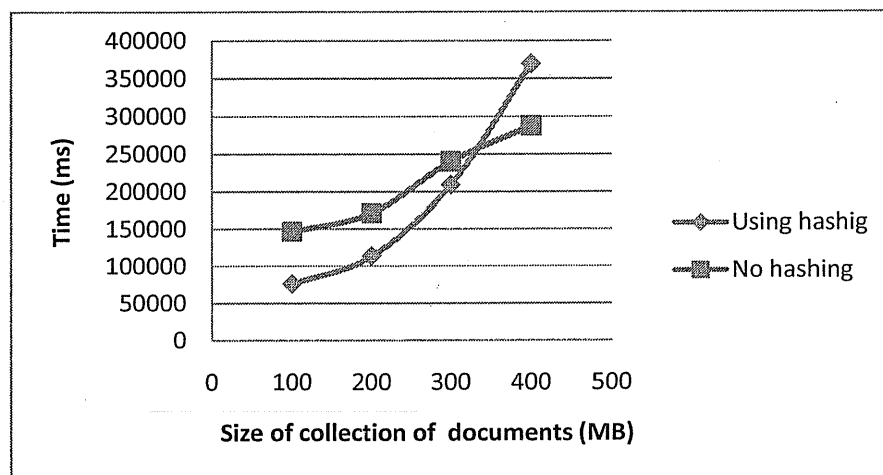


Figure 6.10. Size of input documents vs. time needed to generate signatures.

6.2. Conclusions

From the experiments that was done, we conclude the following:

1. Parallel search gives the best performance (regarding time) needed to check all digital libraries.
2. Distributed search with no overlapping of agents gives the same performance (regarding time) as parallel search, but making no overlapping between agents is not easy.
3. In sequential search the best performance (regarding time) is reached when only one agent is assigned to a document, having more than one agent doing the checking on the document reduces the performance linearly.
4. Parallel search requires sending agent to each digital library, putting a load on the network that is proportional to the number of digital libraries available.
5. Distributed search puts more load on the network due to the increased movement of the agents among digital libraries.

6. Sequential search using one agent puts the least load on the network, where only an agent is moving, and sending only one result.

From working on this project the following is concluded:

1. Plagiarism and copyright infringement of digital text are two dangerous problems facing authors, publishers, students, and teachers.
2. Agent oriented programming is very flexible for data mining problems.
3. There are many differences between simulation on a single computer, simulation on a LAN and a WAN.
4. Fault tolerance is essential to systems that use agents.
5. Team work is very effective in large systems' development.

6.3. Future Work

Doing improvements on:

- The searching scenario used in this system.
- The text comparison method used to compare text.

Expanding the system to cover more than one platform.

6.4. Summary

In this chapter we explained the experiments that were done, conclusions concluded from working on the project, conclusions from the experiments, and future work and improvements on the system.

Appendix A

Survey on plagiarism detection software

A.1. Plagiarismchecker.com [22]

PlagiarismChecker.com is a web-based tool for text plagiarism detection, it's based on a graphical user interface, where the user provides the phrase(s), and an internet search is done on them.

In this system the user has to cut and paste the phrase he wants to check, also has to select between Google and yahoo search engines to be used in the search process, this tool relies on Google and Yahoo! search to do the searching for the phrases, The result is a casual Google/Yahoo! search with the phrases provided by the user.

The system is somehow limited where a phrase should be less than 32 words, more than 6 word, the plagiarism check is done only on online web pages (not possible to check two local documents for partial or full copy).

It has two different options whether to search using Google or Yahoo! search, also it has an option for checking plagiarized web page.

Cost: free.

Website: <http://www.plagiarismchecker.com/>

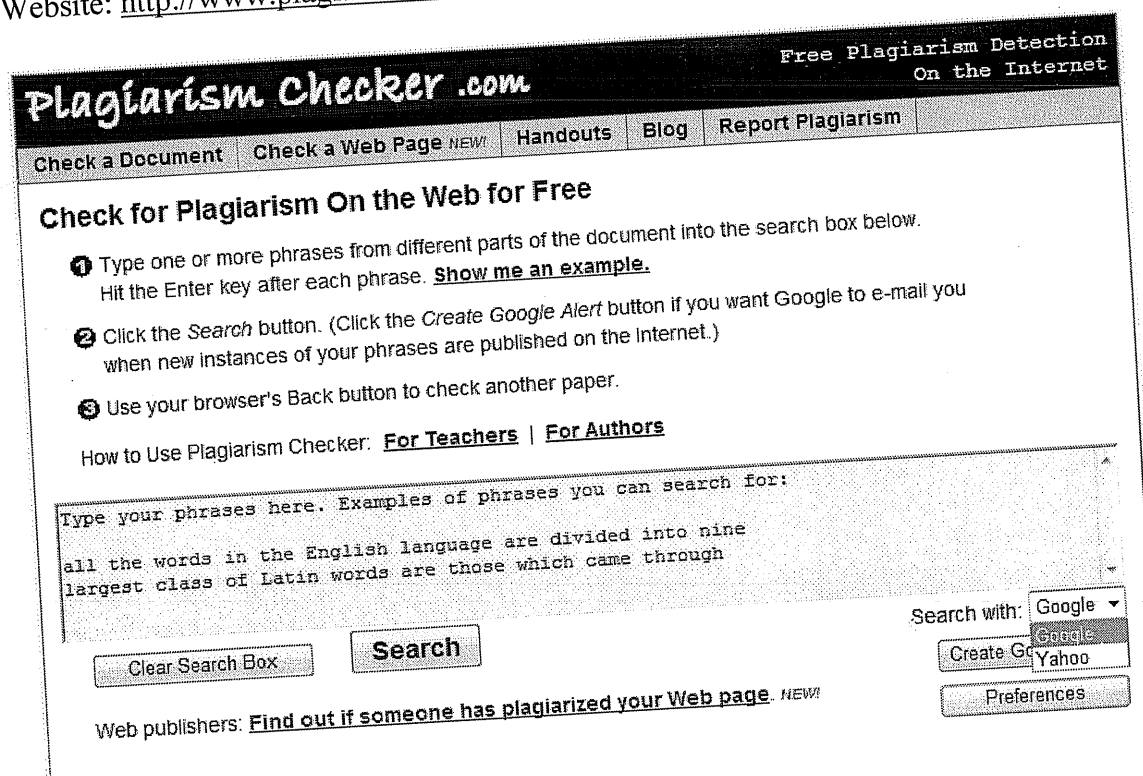


Figure A.1: PlagiarismCheker.com home page.

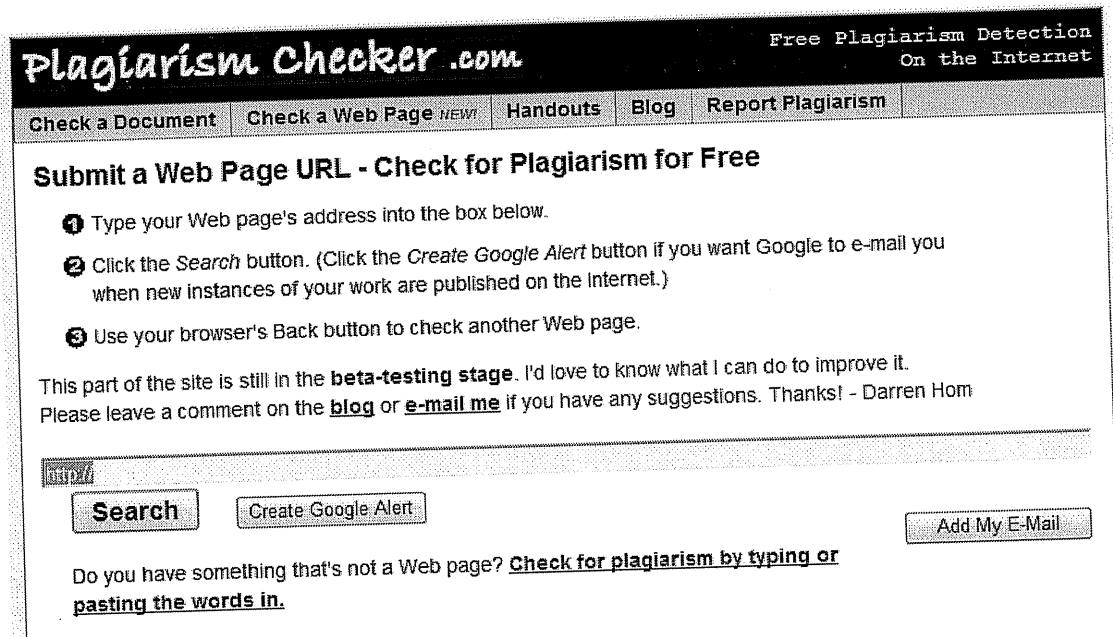


Figure A.2: PlagiarismChecker.com: check a web page.

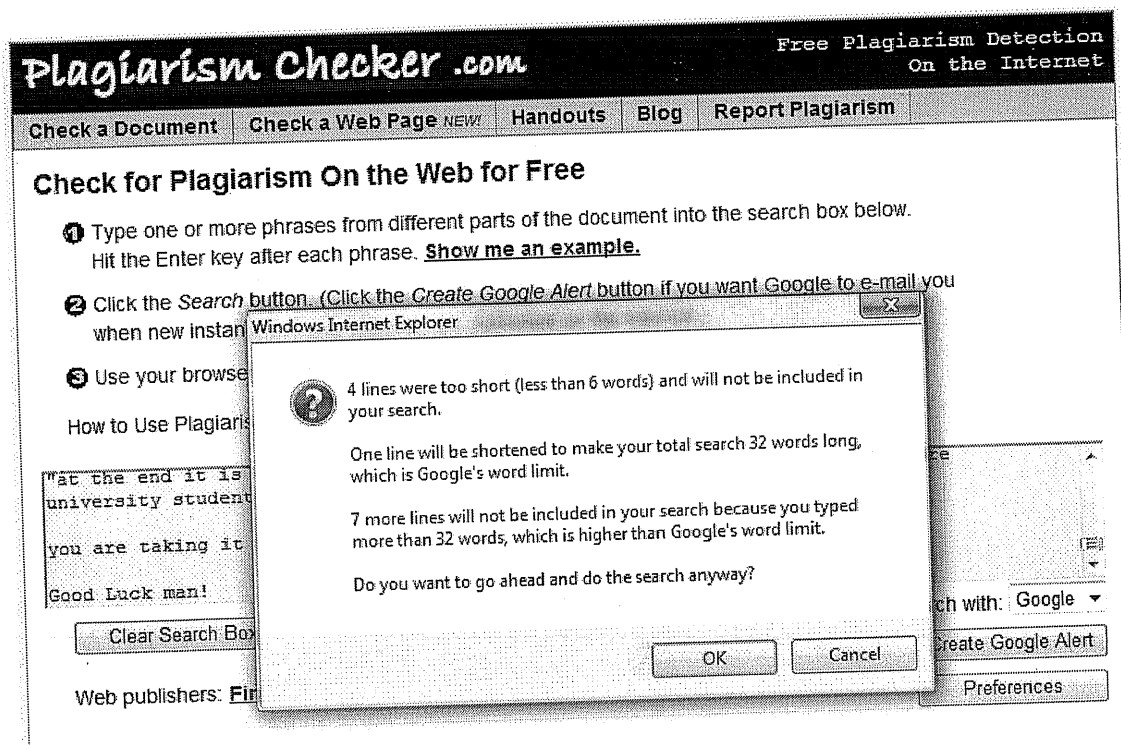


Figure A.3: PlagiarismChecker.com: limitations.

A.2. WCopyfind [23]

WCopyfind is an open source tool that checks plagiarism among local documents; it looks for matches between specific length phrases in the documents.

The user drags the documents to be checked into a box, specifies different options to be considered during the check as comparison rules, then he clicks (Run) and a report is generated viewing the copied portions (from another document) in an HTML format report.

This tool only checks local documents, supports MS WORD, plain text, HTML formats, and does not support PDF files. It has a lot of configurations that may confuse the user, as in figure B.4, and two options for the documents, 1- compare only with new documents, not with one another, 2- compare with old files and with one another.

Cost: free.

Website: <http://plagiarism.phys.virginia.edu/Wsoftware.html>

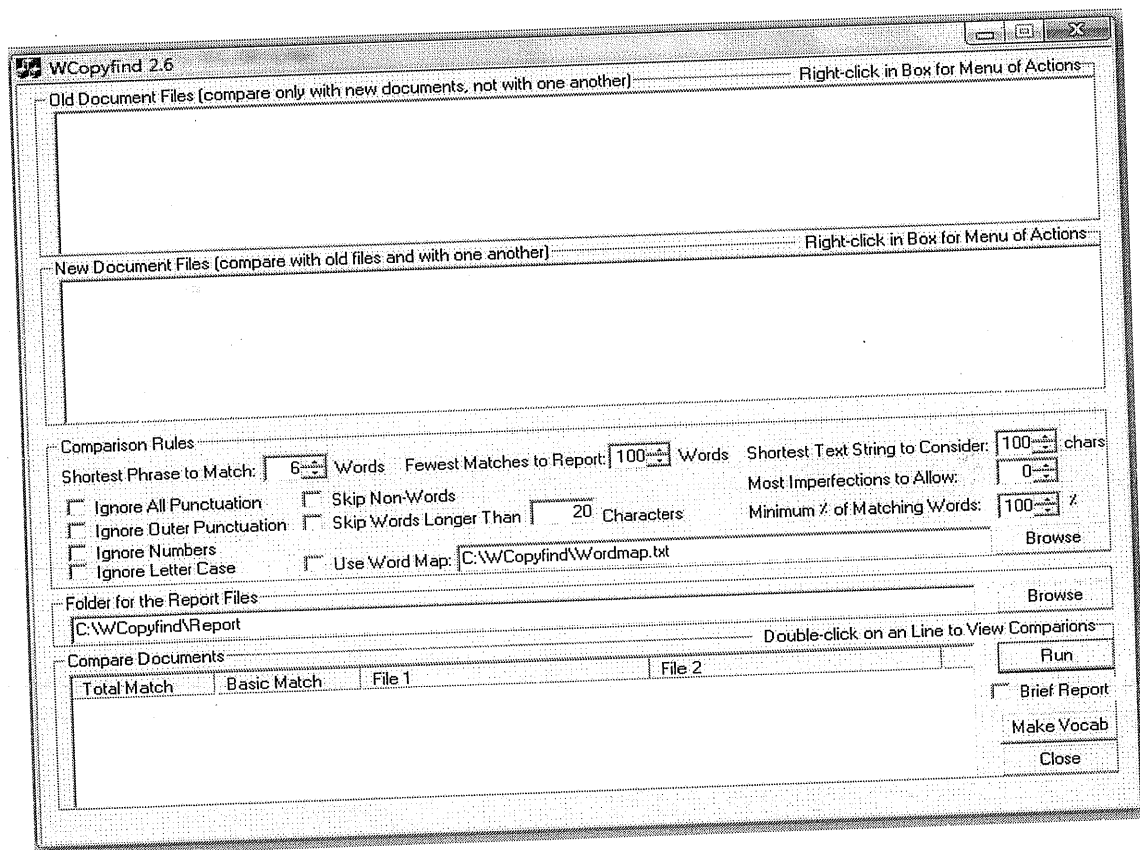


Figure A.4: WCopyfind 2.6.

A.3. Copyscape[24]

Copyscape is a web based system that offers two different services (free, paid); it basically aims detect whether a web page is copied fully or mostly on another website or not.

Simply the user cuts and pastes the uniform resource locator (URL) of his web page and clicks (Go), the result would show links to the web pages that has around 70% of the text in the provides URL, the copied text in these web pages in colored with different colors to be noticed by the user.

The free service allows only 10 times check for a web page, the paid service allows unlimited check for a web page, Copysentry as an extension to Copyscape is another paid services that monitors web site for unauthorized copy.

We have tried several URL's some with good results, where the resulted web pages have more than 70% of the text in the web page, for example see figure 2.16 , however some web pages produced bad results, such as when we tried to check a page of some news event from Al-Jazeera[25] website on (<http://www.aljazeera.net/NR/exeres/B66179D5-E586-4251-AFE6-55BC16231DA3.htm>), and –as a free service- we get 10 results having less than 3% of the contents of the original page.

This system only checks online web page, and it does not check local documents, it uses a WebCrawler, that visits all URL in any page it visits, and downloads these pages, where a recent copy of each visited page is cashed on Copyscape's servers, and some kind of hashing is done on the text in the document, where the minimum chunks available for search is 3 tokens separated by a space. Copyscape detects cut and pasted phrases of the text.

Cost: free service is free, paid service is \$0.05 per search.

Website: <http://www.copyscape.com/>

Copyscape

Search for copies of your page on the Web.

http://

Go

PAGE PROTECTED BY COPYSCAPE DO NOT COPY

Defend your site with a plagiarism warning banner!

Premium - Unlimited searches, copy and paste. Copysentry - Automatic web monitoring.

About Example Plagiarism Press Testimonials Forum Log In

Copyscape © 2008 Indigo Stream Technologies, providers of Google Alert. All rights reserved. Terms of Use.

Figure A.5: Copyscape web page.

الرئيسية | العودة

لقاء حول تكنولوجيا المعلومات في البلديات ومؤسسات المجتمع المدني بالخليل

Publication date:
الأحد فبراير 17 2008

جهد التواضعي - نظم المجلس الأعلى للتنمية في الخليل بالتعاون مع مركز اصدقاء فوزي كعوش للتعمير وتكنولوجيا المعلومات التابع لجامعة بوليتكنك فلسطين وبدعم من مؤسسة فريدريش نومان الألمانية لقاء بعنوان لقاء حول تكنولوجيا المعلومات في البلديات ومؤسسات المجتمع المدني حضره ممثلون عن بلديات ومؤسسات المجتمع المدني في محافظة الخليل.

ورحب المهندس مجدي المختار رئيس مجلس الإدارة بالحضور ، وتدرج الدور الذي يقوم به المجلس في تكنولوجيا المعلومات لتطوير عمل المؤسسات الأهلية.

وأشاد المهندس أسامة التواضعي مستشار وزير الاتصالات وتكنولوجيا المعلومات بالدور الذي يلعبه المجلس في تطوير مؤسسات المجتمع المدني والتكاتف بين البلديات ومؤسسات المجتمع المدني في دعم وتنشيط مثل هذه المبادرات.

وتدرج د. رضوان طهوب مدير مركز فوزي كعوش اهداف المركز وتوجيه نحو تشكيل مرجعية رائدة ومميزة في مجال تكنولوجيا المعلومات.

المهندس جلال السليمه . مدير دائرة التعليم المستمر في جامعة بوليتكنك فلسطين ، المحاضرات التدريبية ، مشيراً إلى تجربته الخاصة في تكنولوجيا المعلومات ومدى إمكانية انعكاس هذه التجربة على المؤسسات المشاركة .

د. فاهر العويبي محاضرة تعريفية بأهمية تطبيق أنظمة المعلومات الجغرافية في البلديات وكيفية الاستفادة من هذه الأنظمة التكنولوجية الحديثة (GIS) والتي تتميز في أنظمة المعلومات الجغرافية.

و استتمت ورشة العمل بمحاضرة القاها د. رضوان ريان تحدث فيها عن أهمية المحاسبة الإلكترونية للبلديات ومؤسسات المجتمع المدني.

وألقي المحاضر في جامعة البوالتكنك المهندس هاني صلاح محاضرة حول أهمية المواقع الإلكترونية للبلديات ومؤسسات المجتمع المدني وكيف يمكن تسخيرها لخدمة أهداف المؤسسات المشاركة.

وقد اختتم اللقاء بمحاضرة عن المطبوع الذي تحدث فيها عن دور برامج إدارة الموارد البشرية لتسهيل وتطوير العمل داخل المؤسسات.

Figure A.6: Example of a web page checked with Copyscape.

A.4. Article checker[26]

Another web-based plagiarism detection tool, which uses Yahoo! and Google to find phrases.

Nearly as the process of (plagiarism checker), however it does not provide the user with notes about the search phrases, also a check can be made using both Yahoo! and Google at the same time.

This system accepts only .txt documents, but it allows the user to upload up to 5 .txt documents to be checked, allows the user to choose between Yahoo! and Google or to use both.

Cost: Free.

Website: <http://www.articlechecker.com/>

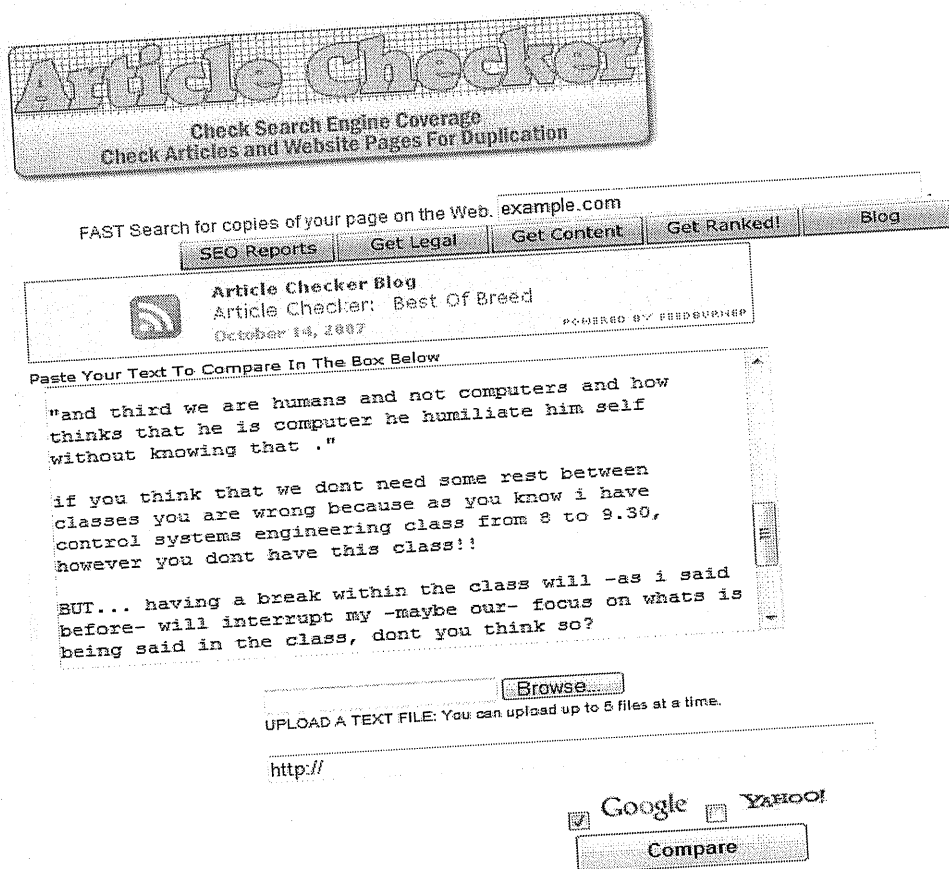


Figure A.7: Article Checker home page.

Search Summary

Item count: 17

Selected Search Engines

Selected search engines: Google Yahoo!

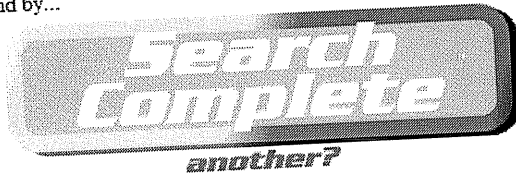
Expected Report Execution Time (peak)

CURRENT STATUS

5.2 seconds to perform for Google.

20.15 seconds to perform on Yahoo!

Stand by...



Matched Phrases		Google Results Found	Yahoo Results Found
wait a minute make	(Set Alert)	0	8+
dont you think so	(Set Alert)	0	8+
and third we are humans and not computers and how thinks that he is computer he humiliate him self without knowing that	0	8+	
if you think that we dont need some rest between classes you are wrong because as you know i have control systems engineering class from 8 to 9	0	8+	
you are taking it too much personally	0	8+	
Total Matched Phrases / Total Submitted		0% (0/13)	38% (5/13)

Figure A.8: Results of checking some text on Article checker.

Tracking Actions...

Source is url <http://www.ppu.edu/newportal/uploads/cn4d%20news%202003%20doc%20without%20pics.htm>
Fetching url ... Completed
Source is TEXTAREA

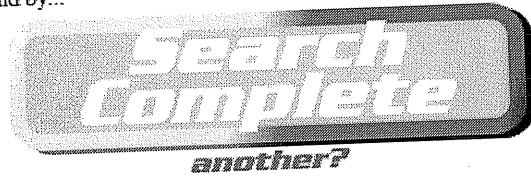
Search Summary

Item count: 1
Selected Search Engines
Selected search engines: Google Yahoo!

Expected Report Execution Time (peak)

CURRENT STATUS

0 seconds to perform for Google.
0 seconds to perform on Yahoo!
Stand by...



Matched Phrases

Total Matched Phrases / Total Submitted

Figure A.9: Result of checking a web page using Article checker.

A.5. Scriptum [27]

A paid plagiarism detection system that helps students and teacher to communicate, the system is designed to be used by teachers in educational institutes.

As a paid system, the institute interested in using this system should buy the product suite at first, the teacher will get a user name and a password, so will his students, the teacher schedule assignment for the student on the specified course page, and students sign in with their usernames and passwords, and upload their assignment, the teacher view the assignments with the degree of originality noted next to the student report.

The teacher has many things to do with the student report, the system make use of Adobe Acrobat Editor so that the teacher can add notes, line marks on the student report, also the teacher has the option to grade the student report.

The system does not compare local documents with each others, i.e. comparing the different students report, accepts only MS WORD documents.

Cost: Subscription to the full Scriptum product suite (with online assignment marking and plagiarism detection): \$99/course per 4-month term (up to 50 students).

Subscription with Plagiarism Detection only: \$49/course per 4-month term (up to 50 students).

Website: <http://www.scriptum.ca>

Scriptum

Login Id:

Password:

Student Login

First time logging in?
[Create an account](#)

[Forgot your password?](#)

Instructor Login

Adobe Reader

© Scriptum 2003

Technical Support

Figure A.10: Scriptum's login page[27].

The screenshot shows the Scriptum web application interface. At the top, there is a header with the text "Aa Bb Cc Dd" and "Scriptum" in a stylized font. Below the header, there are navigation links: "Assignments", "Search Documents", "Administration", "Logout", and "Help".

	Note	Assignment	Author(s)	Grade	Submitted	% Original	
<input type="checkbox"/>		Edit	Midterm Essay	Jim Snow	08/17/03 20:58 EST	100%	Options...
<input type="checkbox"/>		Edit	Midterm Essay	Daisy May	08/17/03 21:09 EST	100%	Options...
<input type="checkbox"/>		Edit	Midterm Essay	John B. Apple	08/17/03 21:10 EST	100% 29%	Options...
<input type="checkbox"/>		Edit	Midterm Essay	Barb Black	08/17/03 21:17 EST	100%	Options...

Below the table, there are three buttons: "Edit Assignment", "Return Selected Assignments", and "Enter Grades".

At the bottom of the page, there is a footer with the Adobe Reader logo, a mouse cursor, the copyright notice "© Scriptum 2003", and a link for "Technical Support".

Figure A.11: Scriptum's results page[27].

A.5. Pl@giarism [28]

Aim of this tool is to detect plagiarism in short reports on the same subject, this tool only detects plagiarism between local documents i.e. it does not compare phrases to the contents of the web pages.

By starting a new project, the user selects the directory where the documents reside, and then he clicks (Process) where the operation begins, each document is compared to all other documents in the directory, the users then can see where plagiarism is spotted in any 2 documents.

The user can check where a 3 words chunk is situation in any 2 compared documents, where it can be checked with a check box and then colored with pink.

Cost: free.

Website: <http://www.plagiarism.tk/>

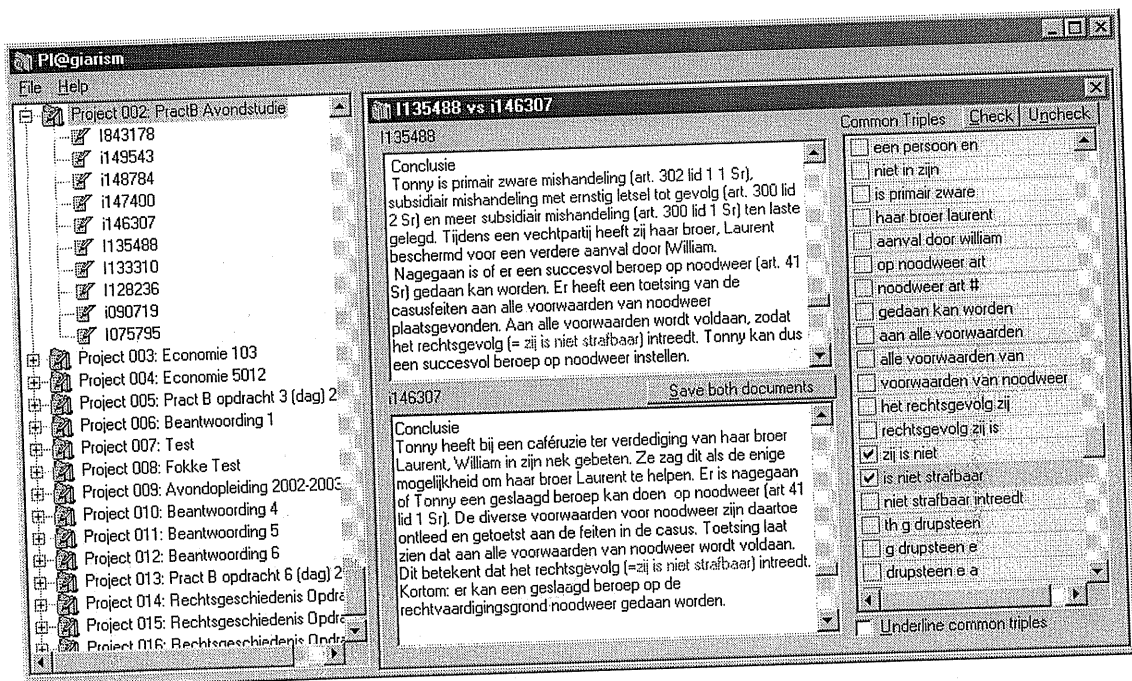


Figure A.12: Snapshot of Pl@giarism[28].

A.6. DOC COP [29]

A web based system for plagiarism detection, it provides two services, (file check, web check), file check is for comparing local documents against each others, web check for checking the coping of a text through Internet web pages.

For file check only accepts MS WORD and PDF files, up to 250000 words a document, however the free services allows only 100 word, for file check the user uploads up to 7 files and clicks (Process) then a report within 1 hour is sent to him as an email with the result. for web check, the user cuts and pastes the text to be checked in the text box available in the system's web page, then clicks (Process), and a report then is sent to him within 1 hour where the plagiarized chunks are colored and links to popular search engines are available to give the user a chance to search for them.

It generates a report in HTML format or MS WORD format, chunks ranges from 6-12 word, but it does not provide the user with the source of a given copied paragraph, it only puts some links for the user to allow him search for the found chunk over the internet using search engines, the percentage of plagiarism between documents is based on number of found chunks, instead of found tokens, which yield to a wrong estimation of the percentage of plagiarism.

Cost: it offers a limited file and web check for free, and paid services to check a corpus for \$4.00 Australian dollar (AUD), and to check a document against the web for \$4.00 AUD.

Website: <http://www.doccop.com/>

Case	Document	Correlation	Document
1 of 10	m1.doc	100%	mobile agent.doc
2 of 10	m3.doc	95%	m1.doc
3 of 10	m3.doc	95%	mobile agent.doc
4 of 10	m1.doc	93%	m4.doc
5 of 10	mobile agent.doc	93%	m4.doc
6 of 10	m3.doc	88%	m4.doc
7 of 10	m2.doc	88%	m1.doc
8 of 10	m2.doc	86%	mobile agent.doc
9 of 10	m2.doc	82%	m4.doc
10 of 10	m2.doc	80%	m3.doc

Return to case list ^ Go forward one

Case	Document	Correlation	Document
1 of 10	m1.doc	100%	mobile agent.doc

MOBILE AGENT, NAMELY IS A TYPE OF SOFTWARE AGENT WITH THE FEATURE OF AUTONOMY SOCIAL ABILITY LEARNING AND MOST IMPORTANT MOBILITY.

WHEN THE TERM MOBILE AGENT IS USED IT REFERS TO A PROCESS THAT CAN TRANSPORT ITS STATE FROM ONE ENVIRONMENT TO ANOTHER WITH ITS DATA INTACT AND STILL BEING ABLE TO PERFORM APPROPRIATELY IN THE NEW ENVIRONMENT. MOBILE AGENTS DECIDE WHEN AND WHERE TO MOVE NEXT WHICH IS EVOLVED FROM RPC. SO HOW EXACTLY DOES A MOBILE AGENT MOVE JUST LIKE A USER DOESN'T REALLY VISIT A WEBSITE BUT ONLY MAKE A COPY OF IT A MOBILE AGENT ACCOMPLISHES THIS MOVE THROUGH DATA DUPLICATION. WHEN A MOBILE AGENT DECIDES TO MOVE IT SAVES ITS OWN STATE AND TRANSPORTS THIS SAVED STATE TO NEXT HOST AND RESUME EXECUTION FROM THE SAVED STATE.

MOBILE AGENT, NAMELY IS A TYPE OF SOFTWARE AGENT WITH THE FEATURE OF AUTONOMY SOCIAL ABILITY LEARNING AND MOST IMPORTANT MOBILITY.

WHEN THE TERM MOBILE AGENT IS USED IT REFERS TO A PROCESS THAT CAN TRANSPORT ITS STATE FROM ONE ENVIRONMENT TO ANOTHER WITH ITS DATA INTACT AND STILL BEING ABLE TO PERFORM APPROPRIATELY IN THE NEW ENVIRONMENT. MOBILE AGENTS DECIDE WHEN AND WHERE TO MOVE NEXT WHICH IS EVOLVED FROM RPC. SO HOW EXACTLY DOES A MOBILE AGENT MOVE JUST LIKE A USER DOESN'T REALLY VISIT A WEBSITE BUT ONLY MAKE A COPY OF IT A MOBILE AGENT ACCOMPLISHES THIS MOVE THROUGH DATA DUPLICATION. WHEN A MOBILE AGENT DECIDES TO MOVE IT SAVES ITS OWN STATE AND TRANSPORTS THIS SAVED STATE TO NEXT HOST AND RESUME EXECUTION FROM THE SAVED STATE.

< Go back one case Return to case list ^ Go forward one

Case	Document	Correlation	Document
2 of 10	m3.doc	95%	m1.doc

WHEN THE TERM MOBILE AGENT IS USED IT REFERS TO A PROCESS THAT CAN TRANSPORT ITS STATE FROM ONE ENVIRONMENT TO ANOTHER WITH ITS DATA INTACT AND STILL BEING ABLE TO PERFORM APPROPRIATELY IN THE NEW ENVIRONMENT.

mobile agent namely is a type of software agent with the feature of autonomy social ab learning and most important mobility.

Figure A.13: DOC COP results of checking local documents against each others.

Summary

Submitted: 23-Feb-2008 at 12:02:56 AM (UTC+11:00)
 Words: 92
 String Length (Words): 6
 Strings found in your document and on the web: 04 of 87 or 97% of document (strings found = highlighted + uppercase text)

Document

in computer science, A MOBILE AGENT IS A COMPOSITION OF COMPUTER SOFTWARE AND DATA WHICH IS ABLE TO MIGRATE MOVE FROM ONE COMPUTER TO ANOTHER AUTONOMOUSLY AND CONTINUE ITS EXECUTION ON THE DESTINATION COMPUTER. MOBILE AGENT, NAMELY IS A TYPE OF SOFTWARE AGENT, WITH THE FEATURE OF AUTONOMY, SOCIAL ABILITY, LEARNING, AND MOST IMPORTANT, MOBILITY. WHEN THE TERM MOBILE AGENT IS USED, IT REFERS TO A PROCESS THAT CAN TRANSPORT ITS STATE FROM ONE ENVIRONMENT TO ANOTHER, WITH ITS DATA INTACT, AND STILL BEING ABLE TO PERFORM APPROPRIATELY IN THE NEW ENVIRONMENT.

Strings

No.	Scroll down to display the status of each string in your document	Conduct your own search on this string using:
1 of 87	in computer science, a mobile agent	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
2 of 87	computer science, a mobile agent is	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
3 of 87	science, a mobile agent is a	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
4 of 87	A MOBILE AGENT IS A COMPOSITION	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
5 of 87	MOBILE AGENT IS A COMPOSITION OF	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
6 of 87	AGENT IS A COMPOSITION OF COMPUTER	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
7 of 87	IS A COMPOSITION OF COMPUTER SOFTWARE	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>
8 of 87	A COMPOSITION OF COMPUTER SOFTWARE AND	Yahoo! <input type="checkbox"/> Google <input type="checkbox"/> Live <input type="checkbox"/> Alexa <input type="checkbox"/>

Figure A.14: DOC COP results of checking local document against the internet.

DOC Cop HOME | TERMS | REGISTER | FILE CHECK | WEB CHECK | FAQ | NEWS | STATUS | EMAIL

Check Material Against The Web (What is PARAGRAPH CHECK?)

I have viewed the Paragraph Report, acknowledge the Word Limit and accept all TERMS

* Guest ID: Need a Guest ID?

* Guest Email:

* Text:

Windows Internet Explorer

Text must be less than or equal to 100 words

OK

Words: 224 Used 0 Remaining

String Length: Words Report Format:

DOC Cop recommends using an 08 word string length.

© 2005-2008 DOC Cop Plagiarism Detection. All rights reserved.

Figure A.15: DOC COP limitations on free service.

A.7. MyDropBox [30]

MyDropBox offers a plagiarism detection service called safeassignment, which can be integrated with BlackBoard[31] system; it helps instructors finding plagiarized text within students' reports.

As a commercial product the educational institute should buy a license for using the product, instructor and students will have usernames, and passwords to enter their accounts, where students upload their assignments and the system will check the assignment and provides the instructor with a report for each student's assignment, the system supports file type of: ZIP, MS WORD, TXT, PDF, RTF, HTML.

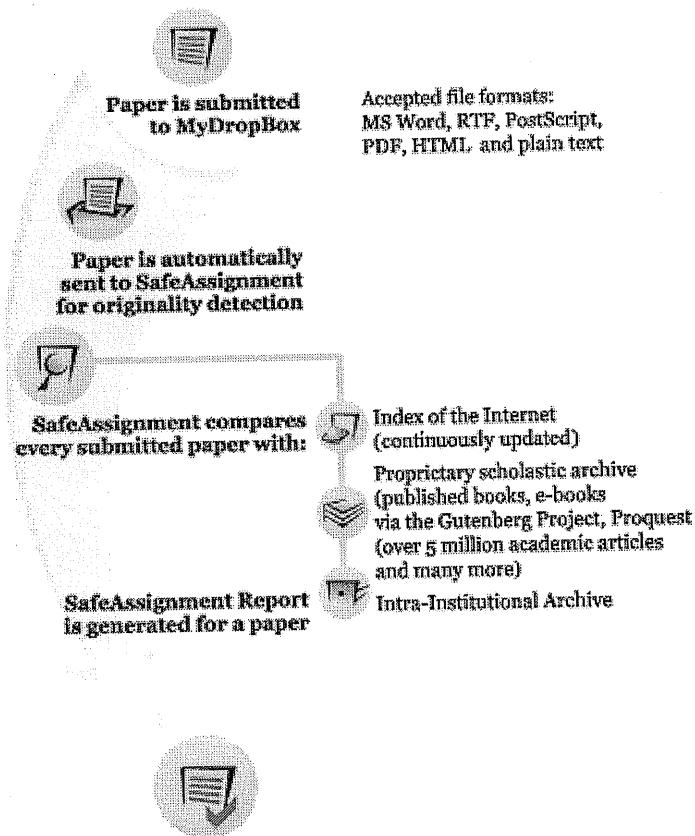


Figure A.16: MyDropBox system [30].

Many options for the report as in figure 3.27 are available for the system.

Cost: for an individual membership: \$49.95 per 3 months, institute membership: depends on the number of students.

Website: <http://www.mydropbox.com/>

This is a sample Save Assignment Report demonstrating the look and functionality of real reports.

Report Information
 Author: John Smith
 Title: Test Paper
 Matching: 72%

Course: Some Course
 Assignment: Some Assignment
 Submitted: 05/05
 Paper ID: 1045

Save Report icon opens the Save As window, allowing a user to store the report on a local disk.
 Save report to disk:
 Email the report:
 Printable version:
 Print Version icon opens a special version of the report, optimized for monochrome office printers.
 Email Report function allows a user to send a link to the report to any email address.

Report information section contains all information necessary to identify the paper.

Suspected Sources
 http://www.babesinspace.net/book/reviews/39.html
 http://www.scifi.com/sfw/issue100/books.html
 http://www.sfreviews.com/docs/Octavia%20Butler_1998_Parable%20Of%20The%20Talents.htm
 Excluded Sources:
 http://www.twbookmark.com/books/59/0446610380/
 http://members.aol.com/firoane/butler.htm
 Do not process the paper without these sources.

Suspected Sources section lists the links to the sources from which the paper was supposedly plagiarized. Users can click on any source link to see the corresponding source.

Delete Sources feature (available to instructors only) allows an instructor to reprocess the paper ignoring any particular sources.

Source Comparison window opens when a user clicks on any color-coded sentence.
 Source Comparison window shows the address of the source document, the percentage of similarity between the matching sentences from the plagiarized paper and from the source, and provides a direct comparison of the two sentences.

Source Highlighting icon opens the source document and highlights all chunks of text that were supposedly copied to the submitted paper.

Misspelled Text
 Octavia E. Butler, author of Dawn, Bloodchild, and Parable of the Talents, the last in a series of futuristic communities inevitable of, with a tale of the survival, destruction, and rebirth of a community called Olamina's love is divided among her young daughter, her community, and the revelation that led Lauren to find a new faith that teaches "God Is Change." However, in the wake of environmental and economic chaos, the U.S. government turns a blind eye to violent bigots who follow her or forsak
 URL: http://members.aol.com/firoane/butler.htm
 Matching: 95%

Uploaded Manuscript:
 Olamina, ment followers found th
 share their lives.

Internet Source:
 Lauren Olamina's love is divided among her young daughter, her community, and the revelation that led Lauren to find a new faith that teaches God is Change

Different colors of sentences in the report indicate different sources of copied text.

Figure A.17: Sample report from MyDropBox system [30].

A.8. Turnitin [32]

Turnitin is another commercial plagiarism detection system; it helps instructors finding plagiarized text within students' assignments, it can be integrated into different Course management systems, such as BlackBoard [31], Moodle [33], and ANGEL[34].

From figure 2.28, the student submits his document through Turnitin web site, the paper is compared to databases of documents, these documents contains pages of books, journals, and internet pages, also the uploaded student papers are stored for later use, and the result are returned to the instructor, this system supports file types of MS Word, PostScript, PDF, HTML, RTF, and plain text.

Cost: for institute membership depends on the number of students, for example as found in [35]; for about 54000 students' campus: \$44000 per annum.

Website: <http://turnitin.com>

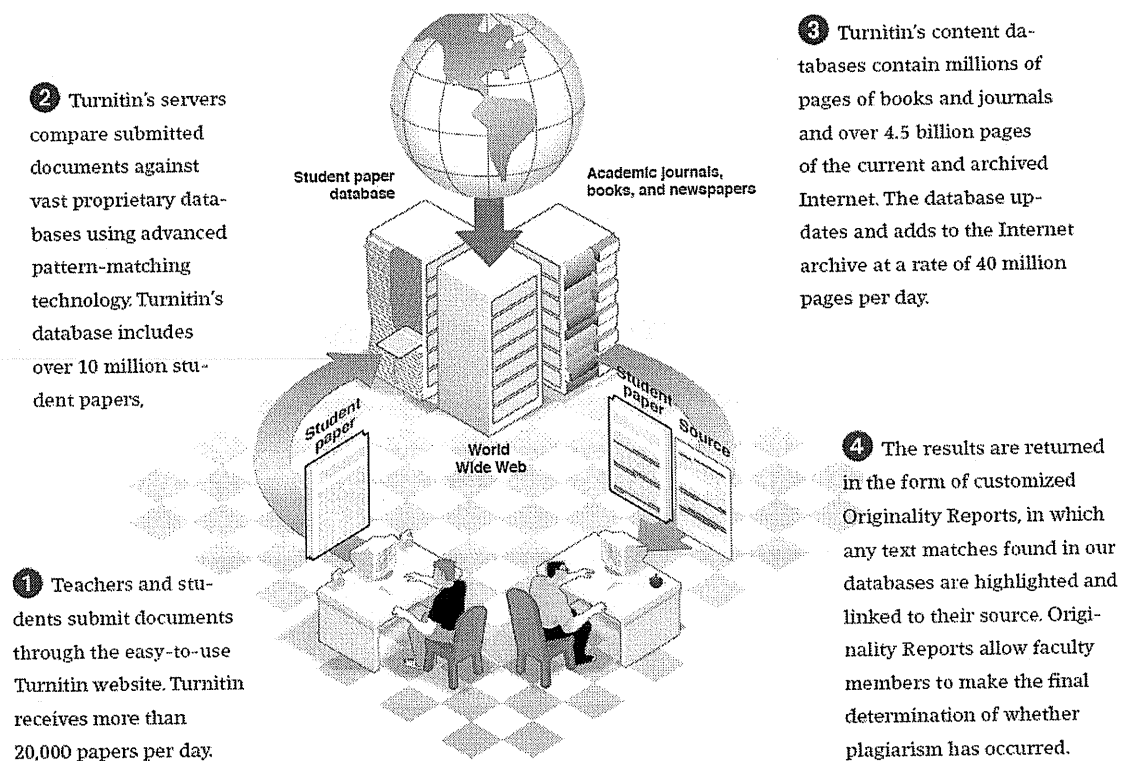


Figure A.18: Turnitin system [32].

Appendix B

Questionnaire on plagiarism at Palestine Polytechnic University (PPU)

1. During your university courses, have you used "cut and paste" from the internet as a method for preparing reports or researches?
- a. Yes b. No

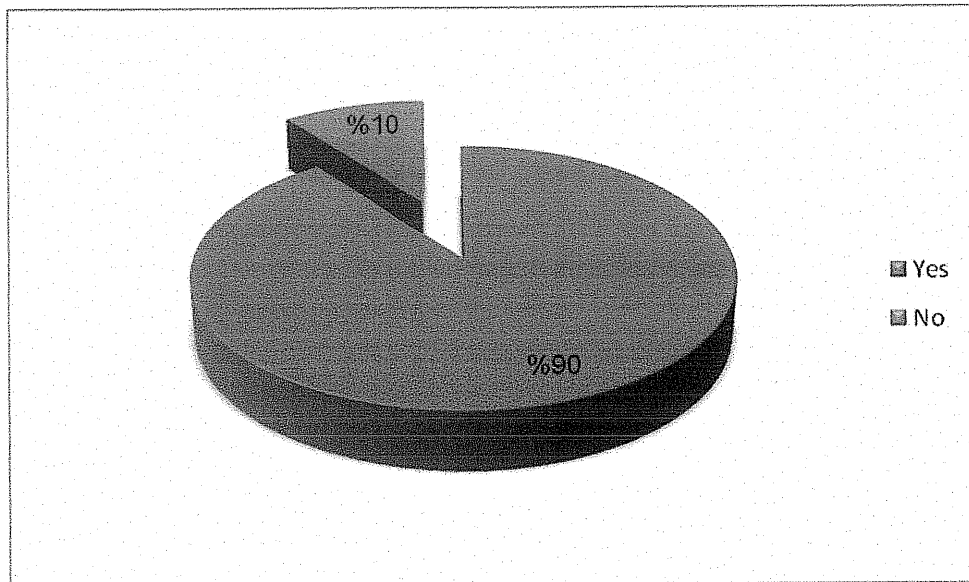


Figure B.1: Results of question 1.

2. If your answer was "Yes" in the first question, were you mentioning the source from where you took the text?
- a. Yes b. No

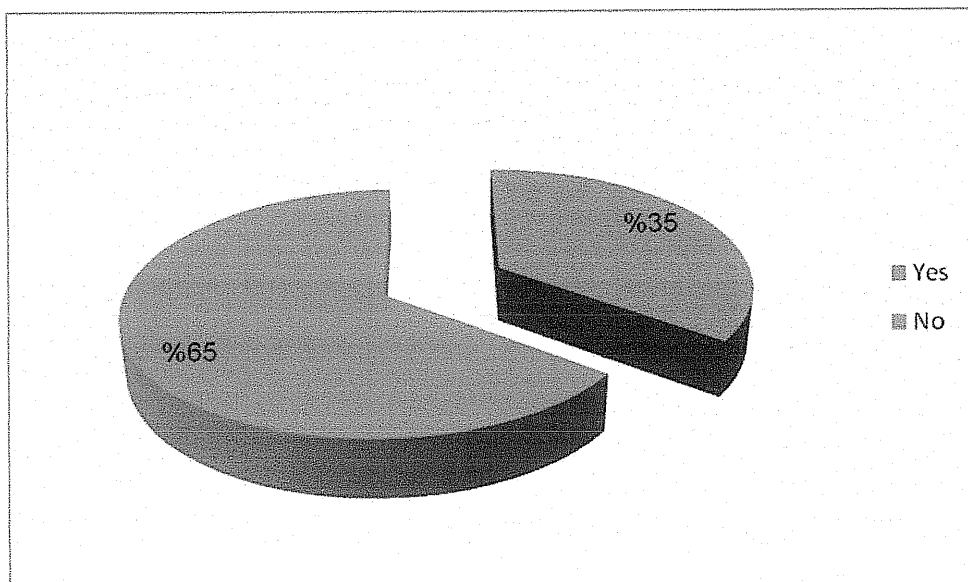


Figure B.2: Results of question 2.

3. Have you used "cut and paste" from your colleagues' reports in preparing reports of researches?
a. Yes b. No

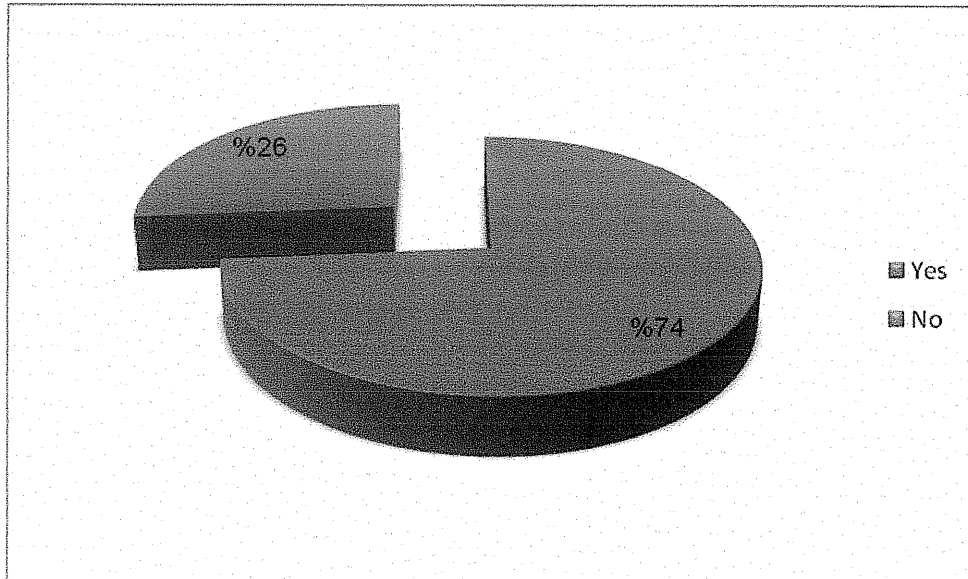


Figure B.3: Results of question 3.

4. If you answered "Yes" on either question 1 or 3 then what made you to use "cut and paste" in preparing reports or researches?
a. Not having enough time.
b. Laziness
c. Not having the ability to understand the text or the ideas and write them using my own word.

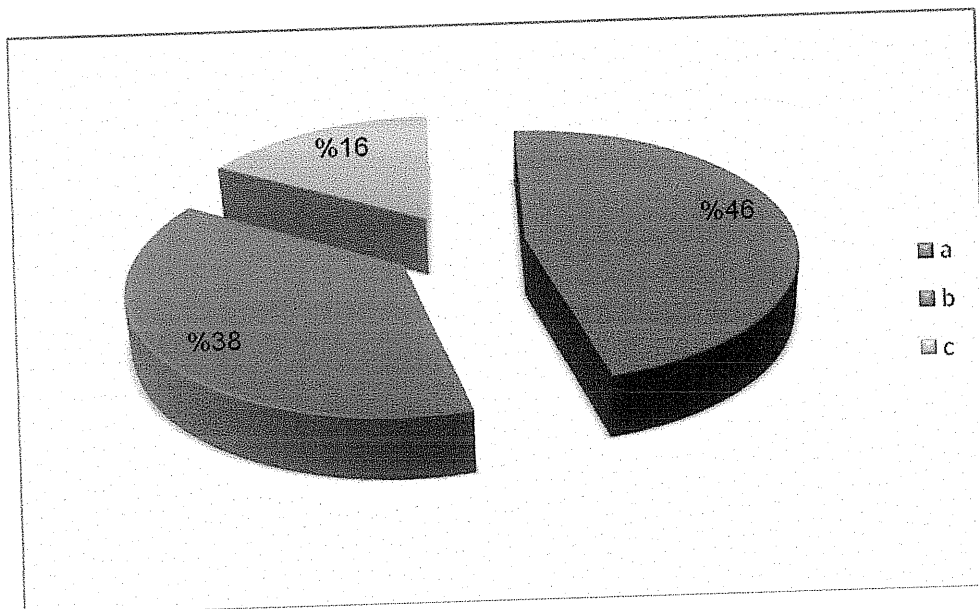


Figure B.4: Results of question 4.

5. Do you know that at some universities a student caught with a plagiarized report gets zero on that report, and might get dismissed from the university?
- a. Yes b. No

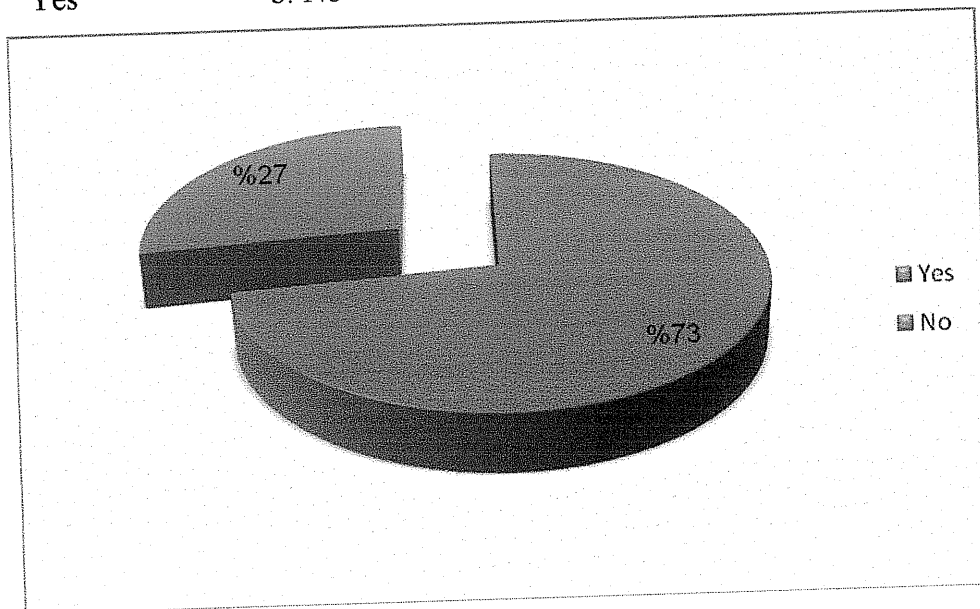


Figure B.5: Results of question 5.

6. Do you know that plagiarism of online text might violate the Copyrights of the owner of the text?
- a. Yes b. No

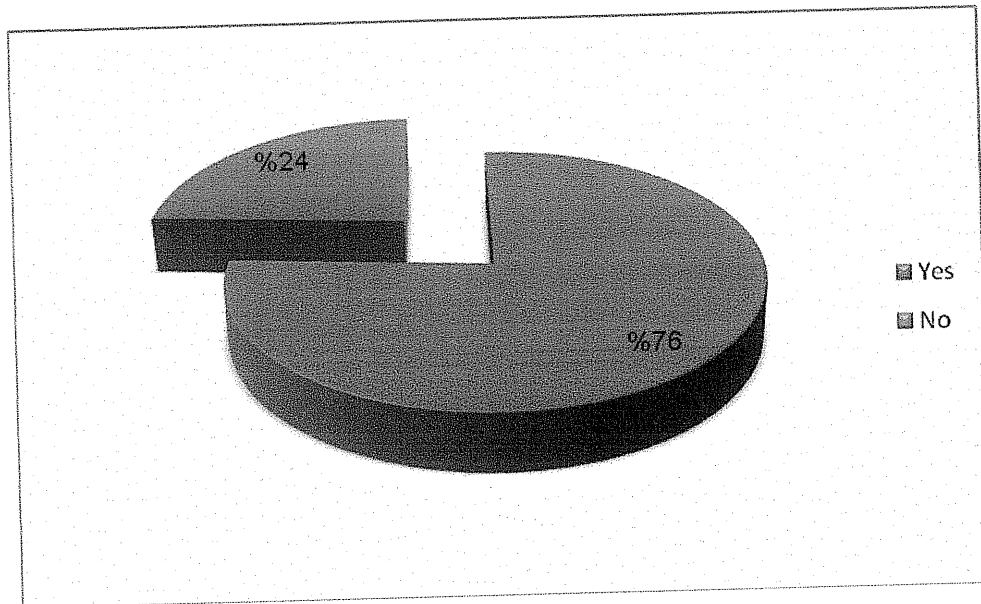


Figure B.6: Results of question 6.

7. Do you think that the popularity of the internet and the availability of documents online make it easier to plagiarize?
- a. Yes b. No

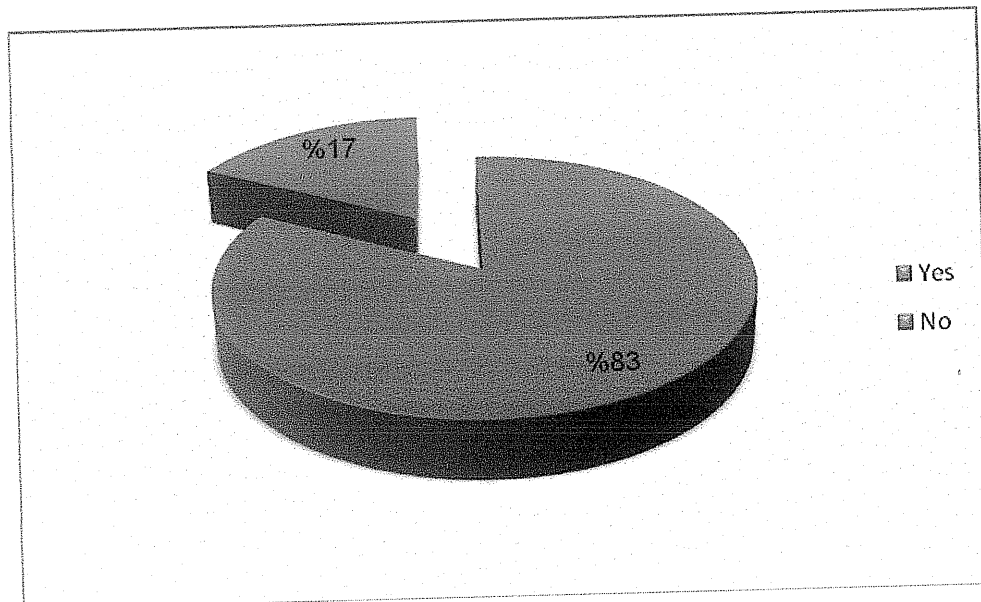


Figure B.7: Results of question 7.

8. Do you know that there exist Software programs that detect plagiarism?
- a. Yes b. No

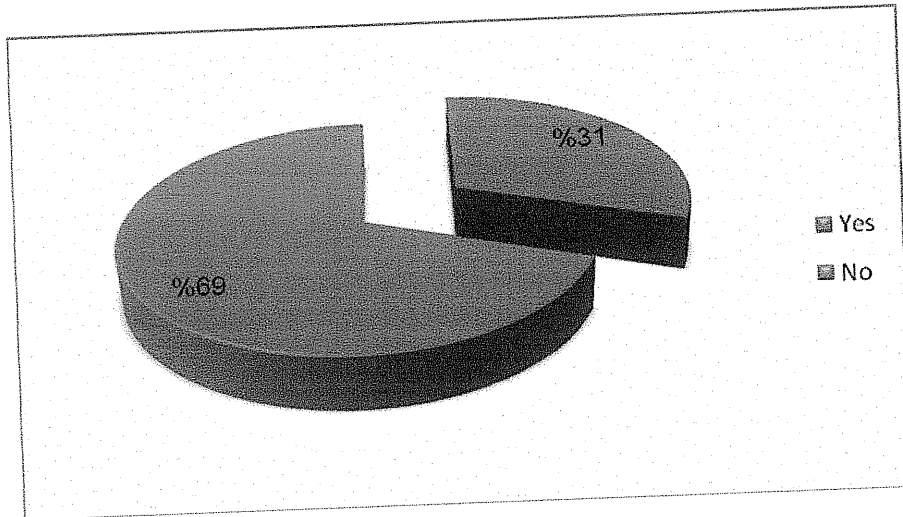


Figure B.8: Results of question 8.

9. Do you think that such programs will reduce plagiarism?
 a. Yes b. No

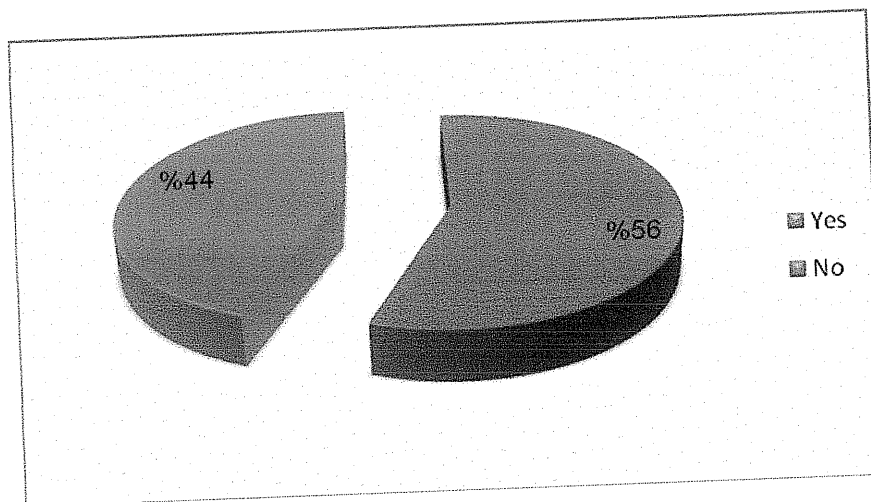


Figure B.9: Results of question 9.

10. Do you think that such programs will corrupt the relation between students and their teachers?
 a. Yes b. No

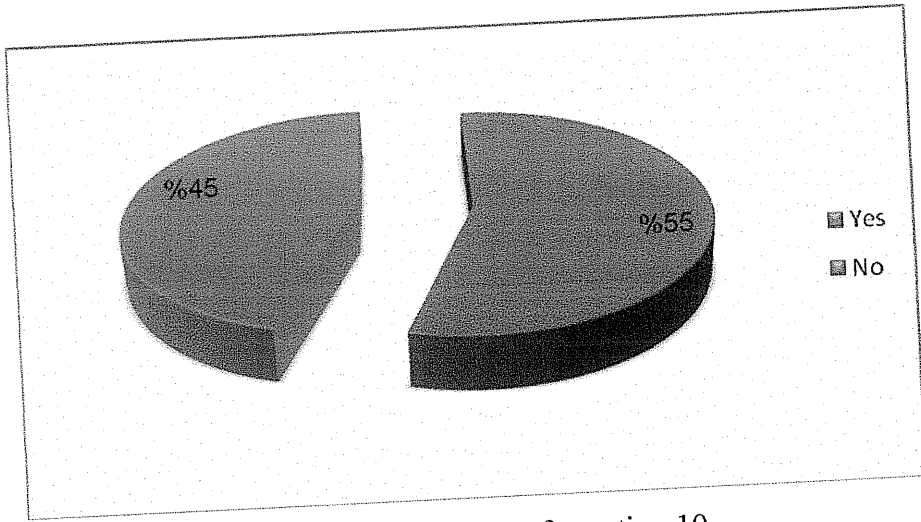


Figure B.10: Results of question 10.

Appendix C
Listing of full code

```

/*
 * JFrame.java
 *
 * Created on January 18, 2008, 3:29 PM
 */

//~--- non-JDK imports ~---
---
import jade.core.ProfileImpl;
import jade.core.Runtime;
import jade.core.PlatformID;
import jade.core.AID;
import jade.wrapper.AgentContainer;

import org.apache.poi.hwpf.extractor.*;

//~--- JDK imports ~---
---

import java.awt.*;
import java.awt.event.*;

import java.io.*;

import java.util.*;

import javax.swing.*;
import javax.swing.UIManager.*;

/**
 *
 * @author Eid Badr
 */
public class JFrame extends javax.swing.JFrame {

    public static int counter = 0;
    public static String documentText[] = null;
    public static String[] documentsNames = null;
    public static int numberOfDocuments;
    public int agentsCounter = 1;
    public int chunkSize = 6;
    public int chunkSizeForDL = 6;
    public double chunksCounter = 0.0;
    public int faultToleranceType = 1;
    boolean flag = false;
    public boolean radioFlag = true, perFlag = true;
    public AgentContainer c;
    public int scenario;
    public int percentage=70;

    /** Creates new form NewJFrame */
    public JFrame() {

        initComponents();
        startContainer();
        jTextField1.setText("To be set automatically");
        jTextField1.setEnabled(false);
        jButton1.setEnabled(false);
        jLabel4.setText("");
        jButton4.setEnabled(false);
        jButton4.setSelected(true);
        jButton5.setEnabled(false);
        jButton3.setEnabled(false);
    }
}

```

```

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    buttonGroup2 = new javax.swing.ButtonGroup();
    jTabbedPane1 = new javax.swing.JTabbedPane();
    jPanel5 = new javax.swing.JPanel();
    jButton9 = new javax.swing.JButton();
    jLabel10 = new javax.swing.JLabel();
    jComboBox1 = new javax.swing.JComboBox();
    jButton7 = new javax.swing.JButton();
    jButton1 = new javax.swing.JButton();
    jTextField1 = new javax.swing.JTextField();
    jLabel2 = new javax.swing.JLabel();
    jCheckBox1 = new javax.swing.JCheckBox();
    jButton10 = new javax.swing.JButton();
    jButton11 = new javax.swing.JButton();
    jButton12 = new javax.swing.JButton();
    jTextField2 = new javax.swing.JTextField();
    jButton2 = new javax.swing.JButton();
    jLabel15 = new javax.swing.JLabel();
    jLabel16 = new javax.swing.JLabel();
    jButton13 = new javax.swing.JButton();
    jButton14 = new javax.swing.JButton();
    jCheckBox2 = new javax.swing.JCheckBox();
    jCheckBox3 = new javax.swing.JCheckBox();
    jTextField3 = new javax.swing.JTextField();
    jButton3 = new javax.swing.JButton();
    jLabel7 = new javax.swing.JLabel();
    jPanel1 = new javax.swing.JPanel();
    jTextField6 = new javax.swing.JTextField();
    jButton12 = new javax.swing.JButton();
    jButton8 = new javax.swing.JButton();
    jLabel11 = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jComboBox2 = new javax.swing.JComboBox();
    jLabel13 = new javax.swing.JLabel();
    jLabel14 = new javax.swing.JLabel();
    jMenuItem1 = new javax.swing.JMenuItem();
    jMenuItem2 = new javax.swing.JMenuItem();
    jMenuItem3 = new javax.swing.JMenuItem();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Plagiarism Detection System using Mobile Agents");

    jButton9.setFont(new java.awt.Font("Tahoma", 1, 11));
    jButton9.setText("Open documents"); // NOI18N
    jButton9.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton9ActionPerformed(evt);
        }
    });

    jLabel10.setFont(new java.awt.Font("Tahoma", 0, 10));

    jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "All books" }));

    jComboBox1.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Categories", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));

```

```

jComboBox1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jComboBox1ActionPerformed(evt);
    }
});

jButton7.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton7.setText("Start"); // NOI18N
jButton7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton7ActionPerformed(evt);
    }
});

jButton1.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton1.setText("Set"); // NOI18N
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jTextField1.setBackground(javax.swing.UIManager.getDefaults().getColor("Button.light"));
jTextField1.setFont(new java.awt.Font("Tahoma", 1, 11));

jTextField1.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Enter number of agents",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));
jTextField1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1ActionPerformed(evt);
    }
});

jCheckBox1.setFont(new java.awt.Font("Tahoma", 1, 11));
jCheckBox1.setText("Fault Tolerance"); // NOI18N
jCheckBox1.addChangeListener(new javax.swing.event.ChangeListener() {
    {
        public void stateChanged(javax.swing.event.ChangeEvent evt) {
            jCheckBox1StateChanged(evt);
        }
    }
});
jCheckBox1.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        jCheckBox1ItemStateChanged(evt);
    }
});
jCheckBox1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jCheckBox1ActionPerformed(evt);
    }
});
jCheckBox1.addPropertyChangeListener(new
java.beans.PropertyChangeListener() {
    public void propertyChange(java.beans.PropertyChangeEvent evt) {
        jCheckBox1PropertyChange(evt);
    }
});

buttonGroup1.add(jRadioButton1);
jRadioButton1.setFont(new java.awt.Font("Tahoma", 1, 11));
jRadioButton1.setSelected(true);
jRadioButton1.setText("Parallel search");

```

```

jRadioButton1.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton1ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton2);
jRadioButton2.setFont(new java.awt.Font("Tahoma", 1, 11));
jRadioButton2.setText("Distributed sequential and parallel search");
jRadioButton2.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton2ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton3);
jRadioButton3.setFont(new java.awt.Font("Tahoma", 1, 11));
jRadioButton3.setText("Sequential search");
jRadioButton3.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton3ActionPerformed(evt);
    }
});

jTextField2.setFont(new java.awt.Font("Tahoma", 1, 11));

jTextField2.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Enter chunk size", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));

jButton2.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton2.setText("Set");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jLabel5.setText("Size is set initially to 6");

jLabel6.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel6.setText("Getting categories...");

buttonGroup2.add(jRadioButton4);
jRadioButton4.setFont(new java.awt.Font("Tahoma", 1, 11));
jRadioButton4.setText("Windowing with size 1.");
jRadioButton4.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton4ActionPerformed(evt);
    }
});

buttonGroup2.add(jRadioButton5);
jRadioButton5.setFont(new java.awt.Font("Tahoma", 1, 11));
jRadioButton5.setText("Centralized.");
jRadioButton5.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton5ActionPerformed(evt);
    }
});

jCheckBox2.setFont(new java.awt.Font("Tahoma", 1, 11));

```



```

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jRadioButton3)
    .add(jRadioButton1)
    .add(jRadioButton2)

.add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel5Layout.createSequentialGroup()
    .add(jTextField1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 194, Short.MAX_VALUE)
    .add(12, 12, 12)
    .add(jButton1)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jLabel2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 266,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
    .add(94, 94, 94)))
    .add(247, 247, 247))
    .add(jPanel5Layout.createSequentialGroup()
    .add(jCheckBox2)
    .add(ContainerGap())
    .add(jPanel5Layout.createSequentialGroup()

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING, false)
    .add(jComboBox1, 0,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.add(org.jdesktop.layout.GroupLayout.LEADING,
jPanel5Layout.createSequentialGroup()
    .add(jCheckBox1)
    .add(18, 18, 18)
    .add(jRadioButton4)))

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jPanel5Layout.createSequentialGroup()

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jRadioButton5)

.add(jPanel5Layout.createSequentialGroup()
    .add(38, 38, 38)
    .add(jLabel6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 169,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
    .add(299, 299, 299)))
    .add(jPanel5Layout.createSequentialGroup()

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 194,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.UNRELATED)
    .add(jButton2)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jLabel5,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 599, Short.MAX_VALUE)
    .add(ContainerGap())
    .add(jPanel5Layout.createSequentialGroup()

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

```



```

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jCheckBox3)
    .add(jPanel5Layout.createSequentialGroup()
        .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 194,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jButton3)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jLabel7,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 205,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        .add(331, 331, 331))
    .add(jPanel5Layout.createSequentialGroup()
        .add(jButton7)
        .add(ContainerGap(532, Short.MAX_VALUE))))
);
jPanel5Layout.setVerticalGroup(
jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jPanel5Layout.createSequentialGroup()
        .add(ContainerGap()

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
    .add(jButton9)
    .add(jLabel10,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 12,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(18, 18, 18)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASILINE)
    .add(jComboBox1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(jLabel6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 23,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(8, 8, 8)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASILINE)
    .add(jRadioButton4)
    .add(jRadioButton5)
    .add(jCheckBox1))
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
    .add(jPanel5Layout.createSequentialGroup()
        .add(jCheckBox2)
        .add(15, 15, 15)
        .add(jRadioButton1)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jRadioButton2)

.addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
    .add(jRadioButton3)
    .add(24, 24, 24)

```

```

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
    .add(jTextField1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
    .add(jButton1,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
    .add(jLabel2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 23,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
    .add(jTextField2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
    .add(jButton2)
    .add(jLabel5))
    .add(25, 25, 25)
    .add(jCheckBox3)
    .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
    .add(jTextField3,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.add(jPanel5Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jLabel7, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jButton3, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
    .add(18, 18, 18)
    .add(jButton7)
    .add(34, 34, 34))
);

jTabbedPane1.addTab("Main", jPanel5);

jTextField6.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Enter chunk size", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 11)));
jTextField6.setMaximumSize(new java.awt.Dimension(6, 20));
jTextField6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField6ActionPerformed(evt);
    }
});

jButton12.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton12.setText("Set"); // NOI18N
jButton12.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton12ActionPerformed(evt);
    }
});

```

```

jButton8.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton8.setText("Create files of signatures"); // NOI18N
jButton8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton8ActionPerformed(evt);
    }
});

jLabel1.setText("Size is set initially to 6");

org.jdesktop.layout.GroupLayout jPanel1Layout = new
org.jdesktop.layout.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jPanel1Layout.createSequentialGroup()
        .add(27, 27, 27)

.add(jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
    .add(jButton8)
    .add(jPanel1Layout.createSequentialGroup()
        .add(jTextField6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 160,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(18, 18, 18)
        .add(jButton12)
        .add(18, 18, 18)
        .add(jLabel1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 158,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(171, Short.MAX_VALUE)
    );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
    .add(jPanel1Layout.createSequentialGroup()
        .addContainerGap(52, Short.MAX_VALUE)

.add(jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASEL
INE)
    .add(jTextField6,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(jButton12)
        .add(jLabel1,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 16,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        .add(52, 52, 52)
        .add(jButton8)
        .add(328, 328, 328)
    );

jTabbedPane.addTab("Digital Library", jPanel1);

jComboBox2.setFont(new java.awt.Font("Tahoma", 1, 11));
jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new
String[] { "List of reports" }));
jComboBox2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jComboBox2ActionPerformed(evt);
    }
});

jLabel13.setFont(new java.awt.Font("Tahoma", 1, 11));

```

```

        jLabel3.setText("Different reports from different agents will be
listed in the list down as soon as the checking finishes"); // NOI18N

        jLabel4.setForeground(new java.awt.Color(255, 51, 51));

        org.jdesktop.layout.GroupLayout jPanel2Layout = new
org.jdesktop.layout.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel2Layout.createSequentialGroup())

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADI
NG)
            .add(jPanel2Layout.createSequentialGroup()
                .add(jPanel2Layout.createSequentialGroup()
                    .add(221, 221, 221)
                    .add(jLabel4,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 372, Short.MAX_VALUE))
                .add(jPanel2Layout.createSequentialGroup()
                    .add(154, 154, 154)
                    .add(jComboBox2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 278,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(jPanel2Layout.createSequentialGroup()
                    .add(18, 18, 18)
                    .add(jLabel3))
                .addContainerGap())
            );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel2Layout.createSequentialGroup()
                .add(jPanel2Layout.createSequentialGroup()
                    .add(74, 74, 74)
                    .add(jLabel3)
                    .add(18, 18, 18)
                    .add(jLabel4,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 21,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(18, 18, 18)
                    .add(jComboBox2,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(335, Short.MAX_VALUE))
                );

        jTabbedPane1.addTab("Result", jPanel2);

        jMenu1.setText("Menu");
        jMenu1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenu1ActionPerformed(evt);
            }
        });

        jMenuItem1.setText("About");
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem1);

        jMenuBar1.add(jMenu1);

        setJMenuBar(jMenuBar1);

```

```

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup())
                .addContainerGap()
                .add(jTabbedPane,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 608,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(layout.createSequentialGroup())
                .add(jTabbedPane,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 528,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)

.addContainerGap(org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    } // </editor-fold>
    private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {
        appendResultsToComboBox();
    }

    public void appendResultsToComboBox() {
        String x =
jComboBox2.getItemAt(jComboBox2.getSelectedIndex()).toString();
        if (!x.equals("List of reports")) {
            StringTokenizer st = new StringTokenizer(x, ":");
            String y = st.nextToken();
            if (st.nextToken().trim().equals("No plagiarism found")) {
                JFrame.jComboBox2.setSelectedIndex(0);
                return;
            }
            File z = new File("");
            try {
                java.lang.Runtime rt = java.lang.Runtime.getRuntime();
                Process p = rt.exec("C:/Program Files/Internet
Explorer/iexplore.exe " + z.getAbsolutePath() + z.separator + y.trim() +
".html");
            } catch (Exception e) {
            }
            jComboBox2.setSelectedIndex(0);
        }

    private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    }
    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
    {
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        int actemp;
        try {
            actemp = Integer.parseInt(jTextField1.getText());
            if (actemp > 50 || actemp < 1) {
                JOptionPane.showMessageDialog(null, "please enter a number
between 1 and 50.");
            }
            return;
        }
    }

```

```

        }
        agentsCounter=actemp;
        jLabel2.setText("Number of agents is set to " + agentsCounter +
".");
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "please enter a valid
integer number.");
    }
}
private void jTextField6ActionPerformed(java.awt.event.ActionEvent evt)
{
}
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        chunkSizeForDL = Integer.parseInt(jTextField6.getText());
        if (chunkSizeForDL < 3 || chunkSizeForDL > 10) {
            JOptionPane.showMessageDialog(null, "please enter a number
between 3 and 10.");
            return;
        }
        jLabel1.setText("Chunk size is set to " + chunkSizeForDL);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "please enter an valid
integer number.");
    }
}
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    counter = 0;
    StringTokenizer inputDocumentTokenizer = null;
    String hashValues1[];
    if (flag) {
        while (jComboBox2.getItemCount() > 1) {
            for (int i = 1; i < jComboBox2.getItemCount(); ++i) {
                JFrame.jComboBox2.removeItemAt(i);
            }
        }
        try {
            c.acceptNewAgent("Buttons Controller", new
buttonsControllerAgent()).start();
        } catch (Exception e) {
        }

        for (int i = 0; i < numberOfDocuments; ++i) {
            inputDocumentTokenizer = new
StringTokenizer(documentText[i].trim().toLowerCase());
            int index = inputDocumentTokenizer.countTokens();
            hashValues1 = null;
            hashValues1 = new String[index];
            chunksCounter = 0.0;
            chunksCounter =
generateSignaturesIntoArray(inputDocumentTokenizer, hashValues1);
            flag = true;
            String type = (String) jComboBox1.getSelectedItem();
            if (type.equals("All books")) {
                type = "";
            }
            try {
                System.out.print(!perFlag);
                c.acceptNewAgent("doc:" + i + ": Document's Coordinator
Agent", new documentsCoordinatorAgent(chunksCounter, chunkSize, hashValues1,
type, agentsCounter, jCheckBox1.isSelected(), scenario, i,
faultToleranceType, jCheckBox2.isSelected(), !perFlag, percentage)).start();
            } catch (Exception e) {
            }
        }
    } else {

```

```

        JOptionPane.showMessageDialog(null, "Please select a collection
of documents!");
    }
}
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
    createFileOfSignatures(chunkSizeForDL);
}
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
}
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    documentText = null;
    documentText = openFile(jLabel10);
}
private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
    scenario = 0;
    jTextField1.setText("To be set automatically");
    jTextField1.setEnabled(false);
    jButton1.setEnabled(false);
    jLabel2.setText("");
}
private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
    scenario = 1;
    jTextField1.setText("To be set automatically");
    jTextField1.setEnabled(false);
    jButton1.setEnabled(false);
    jLabel2.setText("");
}
private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
    scenario = 2;
    jTextField1.setText("");
    jTextField1.setEnabled(true);
    jButton1.setEnabled(true);
    jLabel2.setText("Number of agents is set to " + agentsCounter +
".");
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    int cstemp;
    try {
        cstemp = Integer.parseInt(jTextField2.getText());
        if (cstemp < 3 || cstemp > 10) {
            JOptionPane.showMessageDialog(null, "please enter a number
between 3 and 10.");
            return;
        }
        chunkSize=cstemp;
        jLabel5.setText("Chunk size is set to " + chunkSize);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "please enter an valid
integer number.");
    }
}
private void jCheckBox1StateChanged(javax.swing.event.ChangeEvent evt) {
}
private void jCheckBox1PropertyChange(java.beans.PropertyChangeEvent
evt) {
}
private void jCheckBox1ItemStateChanged(java.awt.event.ItemEvent evt) {
    radioButton4.setEnabled(radioFlag);
    radioButton5.setEnabled(radioFlag);
    radioFlag = !radioFlag;
}
private void jRadioButton4ActionPerformed(java.awt.event.ActionEvent
evt) {
}

```

```

        faultToleranceType = 1;
    }
    private void jButton5ActionPerformed(java.awt.event.ActionEvent
    evt) {
        faultToleranceType = 2;
    }
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(null, "Palestine Polytechnic
        University\nGraduation project\nDesign and implementation by:\nEid Badr &
        Abdullah Abdeen.\nSupervisor: Dr. Radwan Tahboub.\nSpecial thanks to Eng.
        Amal Dweik.");
    }
    private void jMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    }
    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt)
    {
    }
    private void jCheckBox3ActionPerformed(java.awt.event.ActionEvent evt) {
    }
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    int ptemp;
        try {
            ptemp = Integer.parseInt(jTextField3.getText());
            if (ptemp < 5 || ptemp > 100) {
                JOptionPane.showMessageDialog(null, "Please enter a number
                between 1 and 100.");
                return;
            }
            percentage = ptemp;
            jLabel7.setText("Percentage is set to: " + percentage + "%");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Please enter an valid
            integer number.");
        }
    }
    private void jCheckBox3ItemStateChanged(java.awt.event.ItemEvent evt) {
        if (perFlag) {
            jTextField3.setText("");
        }

        jTextField3.setEnabled(perFlag);
        jButton3.setEnabled(perFlag);
        perFlag = !perFlag;
    }
    public void createFileOfSignatures(int size) {
        StringBuffer sigs = new StringBuffer();
        JFileChooser chooser = new JFileChooser();
        chooser.setSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int returnVal = chooser.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            try {
                File file = chooser.getSelectedFile();
                File[] childs = file.listFiles();
                File dir = new File(file.getAbsolutePath() +
                "\\signatures");
                dir.mkdir();
                for (int i = 0; i < childs.length; ++i) {
                    File out = new File(dir + "\\" + childs[i].getName());
                    FileReader reader = new FileReader(childs[i]);
                    BufferedReader bread = new BufferedReader(reader);
                    FileWriter fileWriter = new FileWriter(out);
                    StringBuffer dataBuff = new StringBuffer();
                    String fileContents = new String();
                    String temp;
                    while ((temp = bread.readLine()) != null) {
                        dataBuff.append(temp + "\n");
                    }
                }
            }
        }
    }

```



```

        fileContents = dataBuff.toString();
        sigs = generateSignatures(dataBuff.toString(), size);
        fileWriter.write(sigs.toString());
        fileWriter.close();
        bread.close();
    }
} catch (Exception e) {
}
} else {
    return;
}
}

public void startContainer() {
    ProfileImpl pr = new ProfileImpl();
    try {
        Runtime r = Runtime.instance();
        r.setCloseVM(true);
        pr = new ProfileImpl("127.0.0.1", -1, null);

        c = r.createAgentContainer(pr);
        c.acceptNewAgent("Categories Collectors' Coordinator Agent", new
categoriesCollectorsCoordinatorAgent()).start();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

double generateSignaturesIntoArray(StringTokenizer stringTokenizer,
String[] signaturesArray) {
    int j = 0;
    double counter = 0.0;
    String temp[] = new String[chunkSize];
    String temp2 = new String();
    for (int i = 0; i < chunkSize; ++i) {
        if (stringTokenizer.hasMoreTokens()) {
            if (i == 0) {
                temp[i] = stringTokenizer.nextToken();
            } else {
                temp[i] = stringTokenizer.nextToken();
            }
        }
    }
    while (stringTokenizer.hasMoreTokens()) {
        counter++;
        temp2 = "";
        for (int i = 0; i < chunkSize; ++i) {
            temp2 += temp[i] + " ";
        }
        signaturesArray[j] =
Integer.toHexString(temp2.hashCode()); // (double)total / (double)temp2.length()
; //
        j++;
        for (int i = 0; i < chunkSize; ++i) {
            if (i != chunkSize - 1) {
                temp[i] = temp[i + 1];
            } else {
                temp[i] = stringTokenizer.nextToken();
            }
        }
    }
    return counter;
}

StringBuffer generateSignatures(String a, int size) {
    StringBuffer signaturesData = new StringBuffer();
    String s;
    int j = 0;
    a = a.toLowerCase();
    StringTokenizer t = new StringTokenizer(a);

```

```

String temp[] = new String[size];
String temp2 = new String();
for (int i = 0; i < size; ++i) {
    if (t.hasMoreTokens()) {
        if (i == 0) {
            temp[i] = t.nextToken();
        } else {
            temp[i] = t.nextToken();
        }
    }
}
while (t.hasMoreTokens()) {
    temp2 = "";
    for (int i = 0; i < size; ++i) {
        temp2 += temp[i] + " ";
    }
    // s = Integer.toHexString(temp2.hashCode());
    signaturesData.append(temp2+ " ");
    for (int i = 0; i < size; ++i) {
        if (i != size - 1) {
            temp[i] = temp[i + 1];
        } else {
            temp[i] = t.nextToken();
        }
    }
}
return signaturesData;
}
boolean checkDocuments(File[] files) {
    for (int i = 0; i < files.length; ++i) {
        if (!files[i].getName().endsWith("doc") &&
!files[i].getName().endsWith("txt") || files[i].isDirectory()) {
            return false;
        }
    }
    return true;
}
String[] openFile(JLabel label) {
    int i = 0;
    WordExtractor extractor = null;
    StringBuffer temp1 = new StringBuffer();
    File file = null;
    JFileChooser chooser = new JFileChooser();
    chooser.setSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    String[] docs = null;
    int returnVal = chooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        flag = true;
        try {
            file = chooser.getSelectedFile();
            File[] files = file.listFiles();
            numberOfDocuments = files.length;
            if (numberOfDocuments == 0) {
                JOptionPane.showMessageDialog(null, "Folder is empty!");
                flag = false;
            }
            if (numberOfDocuments > 20 && flag) {
                JOptionPane.showMessageDialog(null, "Maximum number of
documents is 20!");
                flag = false;
            }
        }
        documentsNames = new String[files.length];
        docs = new String[files.length];
        if (!checkDocuments(files) && flag) {
            JOptionPane.showMessageDialog(null, "One or more
document are not suitable for detection.\n Only .doc and .txt documents are
suitable for detection.");
            flag = false;
        }
    }
}

```

```

    }
    if (flag) {
        for (i = 0; i < files.length; ++i) {
            documentsNames[i] = files[i].getName();
            if ((files[i].getName().endsWith("doc"))) {
                extractor = new WordExtractor(new
FileInputStream(file));
                docs[i] = (extractor.getText());
            } else if (files[i].getName().endsWith("txt")) {

                FileReader reader = new FileReader(files[i]);
                BufferedReader bread = new
BufferedReader(reader);

                String temp = null;
                templ = new StringBuffer();
                while ((temp = bread.readLine()) != null) {
                    templ.append(temp + "\n");
                }
                docs[i] = templ.toString();
            }
        }
    }
    } catch (Exception e) {
    }
    label.setText(file.getAbsolutePath());
} else {
    flag = false;
}
return docs;
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    try {
        UIManager.setLookAndFeel(
            UIManager.getInstalledLookAndFeels()[2].getClassName());

    } catch (Exception e) {
        JOptionPane.showInternalMessageDialog(null, "Look and feel
download error");
    }

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new JFrame().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton12;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
public static javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
public static javax.swing.JButton jButton9;
private javax.swing.JCheckBox jCheckBox1;
private javax.swing.JCheckBox jCheckBox2;
private javax.swing.JCheckBox jCheckBox3;
public static javax.swing.JComboBox jComboBox1;
public static javax.swing.JComboBox jComboBox2;
private javax.swing.JLabel jLabel1;

```

```
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
public static javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
public static javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel5;
private javax.swing.JRadioButton jButton1;
private javax.swing.JRadioButton jButton2;
private javax.swing.JRadioButton jButton3;
private javax.swing.JRadioButton jButton4;
private javax.swing.JRadioButton jButton5;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField6;
// End of variables declaration
}
```

```

//~--- non-JDK imports -----
---

import jade.content.*;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Result;

import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.TickerBehaviour;
import jade.core.mobility.*;

import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.JADEAgentManagement.QueryAgentsOnLocation;
import jade.domain.introspection.*;
import jade.domain.mobility.MobilityOntology;

import jade.lang.acl.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class buttonsControllerAgent extends Agent {
    @Override
    public void setup() {
        getContentManager().registerLanguage(new SLCodec());

        getContentManager().registerOntology(MobilityOntology.getInstance());
        setControl(false);
        addBehaviour(new TickerBehaviour(this, 10000) {
            protected void onTick() {
                boolean flag = false;
                jade.util.leap.Iterator it = Agents();
                while (it.hasNext()) {
                    String x = ((AID) it.next()).getName();
                    if (x.contains("Document's Coordinator Agent")) {
                        flag = true;
                        break;
                    }
                }
                if (!flag) {
                    setControl(true);
                    javax.swing.JOptionPane.showMessageDialog(null,
                        "Searching finished, Please check the results
tab.");
                    doDelete();
                }
            }
        });
    }

    public void setControl(boolean flag) {
        JFrame.jButton7.setEnabled(flag);
        JFrame.jButton9.setEnabled(flag);
    }

    public jade.util.leap.Iterator Agents() {
        jade.util.leap.Iterator it = null;
        try {
            Action act = new Action();
            act.setActor(getAMS());
            QueryAgentsOnLocation action = new QueryAgentsOnLocation();
            action.setLocation(this.here());
            act.setAction(action);
            sendRequest(act);
            MessageTemplate mt =
MessageTemplate.and(MessageTemplate.MatchSender(getAMS()),

```

```

MessageTemplate.MatchPerformative(ACLMessage.INFORM));
    ACLMessage resp = blockingReceive(mt);
    ContentElement ce =
getContentManager().extractContent(resp);
    Result result = (Result) ce;
    it = result.getItems().iterator();
    return it;
} catch (Exception e) {
    e.printStackTrace();
}
return it;
}

public void sendRequest(Action action) {
    ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
    request.setLanguage(new SLCodec().getName());
    request.setOntology(MobilityOntology.getInstance().getName());
    try {
        getContentManager().fillContent(request, action);
        request.addReceiver(action.getActor());
        send(request);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}

```

//~ Formatted by Jindent --- <http://www.jindent.com>

```

//~--- non-JDK imports -----
---

import jade.content.*;

import jade.core.*;
import jade.core.AID;
import jade.core.Agent;
import jade.core.Location;
import jade.core.mobility.*;

import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.introspection.*;

import jade.lang.acl.ACLMessage;

//~--- JDK imports -----
---

import java.io.File;

public class categoriesCollectorAgent extends Agent {
    public int      docNumber;
    public Location location;
    public categoriesCollectorAgent() {}
    public categoriesCollectorAgent(Location loc) {
        this.location = loc;
    }

    @Override
    public void setup() {
        System.out.print("setup");
        this.doMove(location);
    }

    @Override
    public void afterMove() {
        // doWait(5000);
        System.out.print("aftermove");
        String categories = getCategories();
        sendMessage(categories);
        doDelete();
    }

    public String getCategories() {
        String data = "";
        String[] files = null;
        try {
            File file = new File("Digital Library");
            files = file.list();
        } catch (Exception e) {}

        for (int i = 0; i < files.length; ++i) {
            data += "%" + files[i];
        }
        return data;
    }

    public void sendMessage(String msgData) {
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.addReceiver(new AID("Categories Collectors' Coordinator Agent",
AID.ISLOCALNAME));
        msg.setContent(msgData);
        send(msg);
    }
}

```

}

//~ Formatted by Jindent --- <http://www.jindent.com>


```

//~--- non-JDK imports -----
---

import jade.content.*;
import jade.content.lang.sl.SLCodec;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Result;
import jade.content.onto.basic.Result;

import jade.core.Agent;
import jade.core.Agent;
import jade.core.Agent;
import jade.core.Location;
import jade.core.Location;
import jade.core.Location;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.mobility.*;

import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.introspection.*;
import jade.domain.mobility.MobilityOntology;
import jade.domain.mobility.MobilityOntology;
import jade.domain.mobility.MobilityOntology;

import jade.lang.acl.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

//~--- JDK imports -----
---

import java.io.*;

import java.util.ArrayList;
import java.util.StringTokenizer;

public class categoriesCollectorsCoordinatorAgent extends Agent {
    public int j = 0;
    public int k = 0;

    @Override
    public void setup() {
        getContentManager().registerLanguage(new SLCodec());

        getContentManager().registerOntology(MobilityOntology.getInstance());
        jade.util.leap.Iterator it = Locations();
        String locationString;
        Location location;
        java.util.List list = new ArrayList();
        while (it.hasNext()) {
            location = (Location) it.next();
            locationString = location.getName();
            if (!locationString.equals("Main-Container"))
                &&!locationString.equals(this.here().getName()) {
                    ++k;
                }
        }
        Behaviour by = new recBeh();
    }
}

```

```

        this.addBehaviour(by);
        try {
            startCollectors();
        } catch (Exception e) {}
    }

    public void startCollectors() throws Exception {
        int i = 0;
        jade.util.leap.Iterator it = Locations();
        while (it.hasNext()) {
            Location lo = (Location) it.next();
            if (!lo.getName().equals("Main-Container")
                &&!lo.getName().equals(here().getName())) {
                this.getContainerController().acceptNewAgent("Categories'
Collector Agent " + ++i,
                    new categoriesCollectorAgent(lo)).start();
            }
        }
    }

    @Override
    public Location here() {
        return super.here();
    }

    @Override
    public void takeDown() {
        System.out.print("taken down");
    }

    public void done() {
        System.out.println("done()");
    }

    public jade.util.leap.Iterator Locations() {
        jade.util.leap.Iterator it = null;
        try {
            sendRequest(new Action(getAMS(), new
QueryPlatformLocationsAction());
                MessageTemplate mt =
MessageTemplate.and(MessageTemplate.MatchSender(getAMS()),
MessageTemplate.MatchPerformative(ACLMessage.INFORM));
                ACLMessage resp = blockingReceive(mt);
                ContentElement ce =
getContentManager().extractContent(resp);
                Result result = (Result) ce;
                it = result.getItems().iterator();
                return it;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return it;
    }

    public void sendRequest(Action action) {
        ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
        request.setLanguage(new SLCodec().getName());
        request.setOntology(MobilityOntology.getInstance().getName());
        try {
            getContentManager().fillContent(request, action);
            request.addReceiver(action.getActor());
            send(request);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

```

```

    }

    class recBeh extends CyclicBehaviour {
        public void action() {
            try {
                ACLMessage msg = myAgent.receive();
                if (msg != null) {
                    j++;
                    StringTokenizer st = new
StringTokenizer(msg.getContent(), "%");
                    while (st.hasMoreTokens()) {
                        String category = st.nextToken().toLowerCase();
                        JFrame.jComboBox1.removeItem(category);
                        JFrame.jComboBox1.addItem(category);
                    }
                    if (j == k) {
                        JFrame.jLabel6.setText("All categories acquired.");
                        doDelete();
                    }
                } else {
                    block();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

//~ Formatted by Jindent --- <http://www.jindent.com>

```

import jade.content.*;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Result;

import jade.core.Agent;
import jade.core.Location;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.mobility.*;

import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.introspection.*;
import jade.domain.mobility.MobilityOntology;

import jade.lang.acl.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

//~--- JDK imports -----
---

import java.io.*;
import java.io.File;

import java.util.ArrayList;
import java.util.Hashtable;
import java.util.StringTokenizer;

public class documentsCoordinatorAgent extends Agent {
public long t1,t2,t3;
    public int agentsCounter = 0;
    public String[] key = null;
    java.util.List list = new ArrayList();
    public Hashtable htForReport = new Hashtable();
    boolean flag2 = false;
    boolean flag = false;
    public String s1 = new String();
    public Hashtable xht = new Hashtable();
    public boolean security = false;
    int chunkSize;
    public double chunksCounter;
    public int docNumber;
    public boolean faultTolerance;
    public int faultToleranceType;
    public String[] hashValues;
    public int[] hashValuesPositions;
    public int numberOfWorkers;
    String[] perArray;
    int s[];
    public int scenario;
    public String typeFactor;
    public boolean percentageFlag=false;
    public int percentageValue;

    public documentsCoordinatorAgent() {
    }

    public documentsCoordinatorAgent(double c2, int chunkSize, String[]
hashValues, String typeFactor, int numberOfWorkers, boolean ft, int
scenario, int docNumber, int ftType, boolean security, boolean
percentageFlag,int percentageValue) {
        this.chunksCounter = c2;
        this.chunkSize = chunkSize;

```

```

        this.s = new int[(int) this.chunksCounter + this.chunkSize];
        this.numberOfWorkers = numberOfWorkers;
        this.hashValues = hashValues;
        this.perArray = new String[this.numberOfWorkers];
        this.hashValuesPositions = new int[(int) this.chunksCounter +
this.chunkSize];
        this.typeFactor = typeFactor;
        this.faultTolerance = ft;
        this.scenario = scenario;
        this.docNumber = docNumber;
        this.faultToleranceType = ftType;
        this.security = security;
        this.percentageFlag=percentageFlag;
        this.percentageValue=percentageValue;
    }

    @Override
    public void setup() {
        getContentManager().registerLanguage(new SLCodec());

        getContentManager().registerOntology(MobilityOntology.getInstance());
        jade.util.leap.Iterator it = Locations();
        int i = 0;
        String locationString;
        Location location;
        java.util.List list = new ArrayList();
        while (it.hasNext()) {
            location = (Location) it.next();
            locationString = location.getName();
            if (!locationString.equals("Main-Container") &&
!locationString.equals(this.here().getName())) {
                list.add(i++, locationString);
            }
        }
        t1=System.currentTimeMillis();
        switch (scenario) {
            case 0:
                parallelSearch(list);
                break;
            case 1:
                distributedSequentialAndParallelSearch(list);
                break;
            case 2:
                sequentialSearch(list);
                break;
        }
        Behaviour by = new recBeh(chunksCounter);
        this.addBehaviour(by);
    }

    public void setControl(boolean flag) {
        JFrame.jButton7.setEnabled(flag);
        JFrame.jButton9.setEnabled(flag);
    }

    public void generateAllKeys(int number) {
        key = new String[number];
        if (security) {
            for (int i = 0; i < number; ++i) {
                key[i] = generateKey((int) (Math.round(Math.random() *
1000)));
            }
        }
    }

    public void distributedSequentialAndParallelSearch(java.util.List list)
    {

```

```

generateAllKeys(list.size());
int i;
String[] locations1 = new String[list.size() + 1];
String[] locations2 = new String[list.size() + 1];
for (i = 0; i < list.size(); ++i) {
    locations1[i] = (String) list.get(i);
}
numberOfWorkers = list.size();
for (i = 0; i < list.size(); ++i) {
    try {
        int j;
        String var2;

        var2 = locations1[0];
        for (j = 1; j < locations1.length - 1; ++j) {
            locations1[j - 1] = locations1[j];
        }
        locations1[locations1.length - 2] = var2;
        locations1[list.size()] = null;
        int number = i + 1;
        this.getContainerController().acceptNewAgent("doc: " +
docNumber + "- Worker Agent: " + number, new workerAgent(chunksCounter,
chunkSize, hashValues, typeFactor, i + 1, list.size(), faultTolerance,
false, locations1, docNumber, faultToleranceType, security,
key[i],percentageFlag,percentageValue)).start();
        this.doWait(100);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

public void sequentialSearch(java.util.List list) {
    generateAllKeys(numberOfWorkers);
    int i;
    String[] locations = new String[list.size() + 1];
    for (i = 0; i < list.size(); ++i) {
        locations[i] = (String) list.get(i);
    }
    locations[i] = null;
    for (i = 1; i <= numberOfWorkers; ++i) {
        try {
            doWait(100);
            this.getContainerController().acceptNewAgent("doc: " +
docNumber + "- Worker Agent: " + i, new workerAgent(chunksCounter,
chunkSize, hashValues, typeFactor, i, numberOfWorkers, faultTolerance,
false, locations, docNumber, faultToleranceType, security, key[i -
1],percentageFlag,percentageValue)).start();
        } catch (Exception e) {
        }
    }
}

public void parallelSearch(java.util.List list) {
    generateAllKeys(list.size());
    numberOfWorkers = list.size();
    for (int i = 0; i < list.size(); ++i) {
        String[] singleLocation = new String[2];
        singleLocation[0] = (String) list.get(i);
        singleLocation[1] = null;
        int number = i + 1;
        try {
            this.getContainerController().acceptNewAgent("doc: " +
docNumber + "- Worker Agent: " + number, new workerAgent(chunksCounter,
chunkSize, hashValues, typeFactor, number, list.size(), faultTolerance,
true, singleLocation, docNumber, faultToleranceType, security, key[number -
1],percentageFlag,percentageValue)).start();
        } catch (Exception e) {
        }
    }
}

```

```

    }
}

@Override
public Location here() {
    return super.here();
}

@Override
public void takeDown() {

    System.out.print("taken down");
}

public void done() {
    System.out.println("done()");
}

public jade.util.leap.Iterator Locations() {
    jade.util.leap.Iterator it = null;
    try {
        sendRequest(new Action(getAMS(), new
QueryPlatformLocationsAction()));
        MessageTemplate mt = MessageTemplate.and(
            MessageTemplate.MatchSender(getAMS()),
            MessageTemplate.MatchPerformative(ACLMessage.INFORM));
        ACLMessage resp = blockingReceive(mt);
        ContentElement ce = getContentManager().extractContent(resp);
        Result result = (Result) ce;
        it = result.getItems().iterator();
        return it;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return it;
}

public void sendRequest(Action action) {
    ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
    request.setLanguage(new SLCodec().getName());
    request.setOntology(MobilityOntology.getInstance().getName());
    try {
        getContentManager().fillContent(request, action);
        request.addReceiver(action.getActor());
        send(request);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

public String binaryToCharacters(String s) {
    String data = "";
    String temp;
    for (int i = 0; i < s.length(); i = i + 7) {
        temp = "";
        for (int j = i; j < i + 7; ++j) {
            temp += s.charAt(j);
        }
        data += (char) (Integer.parseInt(temp, 2));
    }
    return data;
}

public String XOR(String s1, String key) {
    String result = "";

```

```

    for (int i = 0; i < s1.length(); ++i) {
        if (s1.charAt(i) == key.charAt(i % key.length())) {
            result = result + "0";
        } else {
            result = result + "1";
        }
    }
    return result;
}

public String generateKey(int n) {
    String key = "";
    for (int i = 0; i < n; ++i) {
        key += Integer.toString((int) Math.round(Math.random()));
    }
    return key;
}

class recBeh extends CyclicBehaviour {

    double c;
    String allLetters[] = new String[10];
    String allResources[] = new String[10];
    int i = 0;

    recBeh(double cc) {
        this.c = cc;
    }

    public void action() {
        String contents = null;
        int k = 0;
        try {
            ACLMessage msg = myAgent.receive();
            if (msg != null) {
                t2=System.currentTimeMillis();
                t3=t2-t1;
                System.out.print("\nTime is: "+t3);
                if (security) {
                    StringTokenizer st = new
StringTokenizer(msg.getSender().getLocalName(), "-");
                    String z = st.nextToken();
                    StringTokenizer st2 = new
StringTokenizer(st.nextToken(), ":");
                    z = st2.nextToken();
                    int index =
Integer.parseInt(st2.nextToken().trim());
                    contents = "";
                    contents = XOR(msg.getContent(), key[index - 1]);
                    contents = binaryToCharacters(contents);
                } else {
                    contents = msg.getContent();
                }
                StringTokenizer st = new StringTokenizer(contents, "~");
                if (st.countTokens() > 1) {
                    ++i;
                    JFrame.counter++;
                    allLetters[i] = st.nextToken();
                    allResources[i] = st.nextToken();
                    StringTokenizer x = new
StringTokenizer(allLetters[i]);
                    s = new int[(int) chunksCounter + chunkSize];
                    for (int j = 0; j < s.length; ++j) {
                        s[j] = Integer.parseInt(x.nextToken());
                    }
                    StringTokenizer y = new
StringTokenizer(allResources[i], "%");
                    while (y.hasMoreTokens()) {

```



```

        int yy = Integer.parseInt(y.nextToken().trim());
        String xx = y.nextToken();
        xht.put(yy, xx);
    }
    String array[] = filter(s);
    filter2(s);
    if ((per(s) / 100) * (hashValues.length + chunkSize)
> 30) {
        flag2 = true;
        generateReport(chunksCounter, s,
Integer.toString(i), array);
    }
    if (i == numberOfWorkers) {
        showResult();
        myAgent.doDelete();
    }
    if(percentageFlag){
    if (per(s) > percentageValue && !flag) {
        flag = true;
        startKillers(myAgent);
        showResult();
        myAgent.doDelete();
    }
    }
    } else {
        ++i;
    }
    if (i == numberOfWorkers && !flag2) {
        noPlagiarism();
        JFrame.counter++;
        myAgent.doDelete();
    }
    } else {
        block();
    }
    } catch (Exception e) {
        e.printStackTrace();
    }
    }
}

public void startKillers(Agent myAgent) throws Exception {
    jade.util.leap.Iterator it = Locations();
    int k = 0;
    while (it.hasNext()) {
        k++;
        Location lo = (Location) it.next();
        if (!lo.getName().equals("Main-Container") &&
!lo.getName().equals(myAgent.here().getName())) {
            myAgent.getContainerController().acceptNewAgent("doc: " +
docNumber + " Killer Agent:" + k, new killerAgent(lo, docNumber)).start();
        }
    }
}

public void noPlagiarism() {
    JFrame.jComboBox2.addItem(JFrame.documentsNames[docNumber] + ": No
plagiarism found");
}

public double per(int s[]) {
    double xx = 0.0;
    for (int i = 0; i < chunksCounter + chunkSize; ++i) {
        if (s[i] == 0) {
            if (!(i < chunkSize - 1 && s[0] == 0 && i > -1)) {
                for (int j = i - 1; j > i - chunkSize; --j) {
                    if (s[j] != 0) {
                        xx++;
                        break;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    } else if (s[i] != 0) {
        xx++;
    }
}
return (xx / (chunksCounter + chunkSize)) * 100;
}

public void showResult() {
    JFrame.jLabel4.setText("");
    boolean flag;
    java.util.Collections.sort(list);
    for (int i = list.size() - 1; i >= 0; --i) {
        flag = true;
        String toAdd = JFrame.documentsNames[docNumber] + "(Result-" +
htForReport.get((String) list.get(i)) + ") Percentage: " + list.get(i) +
"%";
        for (int j = 0; j < JFrame.jComboBox2.getItemCount(); ++j) {
            String temp = (String) JFrame.jComboBox2.getItemAt(j);
            if (temp.equals(toAdd)) {
                flag = false;
                break;
            }
        }
        if (flag) {
            JFrame.jComboBox2.addItem(toAdd);
        }
    }
}

public String[] filter(int[] arr) {
    int k = 1;
    Hashtable hm = new Hashtable();
    for (int i = 0; i < arr.length; ++i) {
        if (arr[i] != 0) {
            if (!hm.containsKey(arr[i])) {
                hm.put(arr[i], new Integer(k++));
            }
        }
    }
    int[] arr4 = new int[k];
    String[] arr5 = new String[k];
    for (int i = 0; i < arr.length; ++i) {

        if (arr[i] != 0) {
            String path = (String) xht.get(arr[i]);
            arr[i] = (Integer) hm.get(new Integer(arr[i]));
            arr4[arr[i]]++;
            arr5[arr[i]] = path;
        }
    }
    for (int i = 0; i < arr.length; ++i) {
        if (arr4[arr[i]] < 4) {
            arr[i] = 0;
        }
    }
    return arr5;
}

public void filter2(int[] arr) {
    for (int i = 1; i < arr.length - 1; ++i) {
        if (arr[i - 1] == 0 && arr[i + 1] == 0) {
            arr[i] = 0;
        }
    }
}

```

```

    }
}

public void generateReport(double chunksCount, int arr[], String number,
String arr5[]) {

    if (!htForReport.containsKey(Integer.toString((int) per(arr)))) {
        htForReport.put(Integer.toString((int) per(arr)), new
Integer(number));
        list.add(Integer.toString((int) per(arr)));
        String html = "<html>" + "<p dir=\"ltr\"
align=\"center\"><u><font size=\"5\">Result</font></u></p>";
        html += "<p dir=\"ltr\" align=\"center\"><u><font size=\"5\">" +
"Percentage of plagiarism: " + (int) per(arr) + "%" + "</font></u></p>";
        html += "<p dir=\"ltr\" align=\"center\"><u><font size=\"5\">" +
"Document name: " + JFrame.documentsNames[docNumber] + "</font></u></p>";
        String colors[] = {"<font color=\"#FF6600\">", "<font
color=\"#008000\">", "<font color=\"#FF0000\">", "<font color=\"#FFFF00\">",
"<font color=\"#0000FF\">"};
        String font = "</font>";
        String hyperlink = "</a></u>";
        String href1 = "<u><a title=\"\"";
        String href2 = "\">";
        int i = 0;
        String st3 = new String();
        st3 = "";
        StringTokenizer st4 = new
StringTokenizer(JFrame.documentText[docNumber]);
        i = 0;
        boolean flag = false;
        st3 += html;
        for (i = 0; i < chunksCount + chunkSize; ++i) {
            if (arr[i] == 0) {
                flag = false;
                if (i < chunkSize - 1 && arr[0] == 0 && i > -1) {
                    st3 += st4.nextToken() + " "; //+ font;
                } else {
                    for (int j = i - 1; j > i - chunkSize; --j) {
                        if (arr[j] != 0) {
                            st3 += href1 + arr5[arr[j]] + href2 +
colors[(arr[j] % 5)] + st4.nextToken() + " " + font + hyperlink;
                            flag = true;
                            break;
                        }
                    }
                }
                if (!flag) {
                    st3 += st4.nextToken() + " "; // + font;
                }
            }
            } else if (arr[i] != 0) {
                st3 += href1 + arr5[arr[i]] + href2 + colors[(arr[i] %
5)] + st4.nextToken() + " " + font + hyperlink;
            }
        }
        st3 += "</html>";
        File f1 = new File(JFrame.documentsNames[docNumber] + "(Result-"
+ number + ") Percentage.html");
        try {
            FileWriter wr1 = new FileWriter(f1);
            wr1.write(st3);
            wr1.close();
        } catch (Exception e) {
        }
    }
}
}

```

```

//~--- non-JDK imports -----
---

import jade.content.*;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Result;

import jade.core.*;
import jade.core.AID;
import jade.core.Agent;
import jade.core.Location;
import jade.core.mobility.*;

import jade.domain.FIPAAgentManagement.*;
import jade.domain.JADEAgentManagement.*;
import jade.domain.JADEAgentManagement.QueryAgentsOnLocation;
import jade.domain.introspection.*;
import jade.domain.mobility.MobilityOntology;

import jade.lang.acl.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class killerAgent extends Agent {
    public int      docNumber;
    public Location location;

    public killerAgent() {}

    public killerAgent(Location loc, int docNumber) {
        this.location = loc;
        this.docNumber = docNumber;
    }

    @Override
    public void setup() {
        this.doMove(location);
    }

    @Override
    public void afterMove() {
        getContentManager().registerLanguage(new SLCodec());
    }

    getContentManager().registerOntology(MobilityOntology.getInstance());
    jade.util.leap.Iterator it = Agents();
    try {
        while (it.hasNext()) {
            String x = ((AID) it.next()).getName();
            if (x.startsWith("doc :" + docNumber)) {
                this.getContainerController().getAgent(x).kill();
            }
        }
        this.doDelete();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public jade.util.leap.Iterator Agents() {
    jade.util.leap.Iterator it = null;
    try {
        Action act = new Action();
        act.setActor(getAMS());
        QueryAgentsOnLocation action = new QueryAgentsOnLocation();
    }
}

```

```

        action.setLocation(this.here());
        act.setAction(action);
        sendRequest(act);
        MessageTemplate mt =
MessageTemplate.and(MessageTemplate.MatchSender(getAMS()),
MessageTemplate.MatchPerformative(ACLMessage.INFORM));
        ACLMessage resp = blockingReceive(mt);
        ContentElement ce =
getContentManager().extractContent(resp);
        Result result = (Result) ce;
        it = result.getItems().iterator();
        return it;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return it;
}

public void sendRequest(Action action) {
    ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
    request.setLanguage(new SLCodec().getName());
    request.setOntology(MobilityOntology.getInstance().getName());
    try {
        getContentManager().fillContent(request, action);
        request.addReceiver(action.getActor());
        send(request);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}

```

//~ Formatted by Jindent --- <http://www.jindent.com>

```

//package mobileagent;

//~--- non-JDK imports -----
---
import jade.content.*;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.basic.Action;
import jade.content.onto.basic.Result;

import jade.core.AID;
import jade.core.Agent;
import jade.core.ContainerID;
import jade.core.Location;
import jade.core.behaviours.Behaviour;
import jade.core.behaviours.OneShotBehaviour;

import jade.domain.JADEAgentManagement.*;
import jade.domain.mobility.MobilityOntology;

import jade.lang.acl.*;

//~--- JDK imports -----
---

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;

import java.util.HashMap;
import java.util.Hashtable;
import java.util.StringTokenizer;

public class workerAgent extends Agent {

    public String cloneTo = "";
    public int docsCounter = 0;
    public String key = null;
    public boolean normalDeath = false;
    public String x = "~";
    public boolean security = false;
    public boolean faultTolerance = false;
    public int agentNumber;
    public int chunkSize;
    public double chunksCounter;
    public int docNumber;
    public int faultToleranceType;
    public String[] hashValues;
    public int hashValuesPositions[];
    public String home;
    public int i, count;
    public boolean killed;
    public Hashtable locations;
    public String locs[];
    public int numberOfWorkers;
    public String original, toBeKilled, toBeCloned;
    public boolean parallel;
    public String slaveAgent;
    public String typeFactor;
    public boolean percentageFlag=false;
    public int percentageValue=100;

    public workerAgent(double chunksCounter, int chunkSize, String[]
hashValues, String typeFactor, int agentNumber, int numberOfWorkers, boolean
ft, boolean parallel, String locations[], int docNumber, int ftType, boolean
security, String key,boolean percentageFlag,int percentageValue) {
        this.chunksCounter = chunksCounter;
        this.chunkSize = chunkSize;

```

```

        this.hashValues = hashValues;
        this.typeFactor = typeFactor;
        this.hashValuesPositions = new int[(int) this.chunksCounter +
this.chunkSize];
        this.agentNumber = agentNumber;
        this.numberOfWorkers = numberOfWorkers;
        this.faultTolerance = ft;
        this.locs = locations;
        this.parallel = parallel;
        this.docNumber = docNumber;
        this.faultToleranceType = ftType;
        this.security = security;
        this.key = key;
        this.percentageFlag=percentageFlag;
        this.percentageValue=percentageValue;
    }

    @Override
    public void setup() {
        System.out.print(percentageFlag);
        original = "doc: " + docNumber + "- Worker Agent: " + agentNumber;
        toBeKilled = "doc: " + docNumber + "- Worker Agent: " + agentNumber
+ "-1";
        toBeCloned = "doc: " + docNumber + "- Worker Agent:" + agentNumber
+ "-2";
        getContentManager().registerLanguage(new SLCodec());

        getContentManager().registerOntology(MobilityOntology.getInstance());
        i = 0;
        home = this.here().getName();
        if (faultTolerance) {
            Behaviour bx = new cloneBeh(toBeKilled, this.here().getName(),
"kill");
            this.addBehaviour(bx);
        }
        Behaviour by = new moveBeh(locs[i]);
        this.addBehaviour(by);
    }

    public String to7characters(String x) {
        String data = "";
        for (int i = 0; i < x.length(); ++i) {
            if (Integer.toBinaryString(x.charAt(i)).length() != 7) {
                data += "0" + (Integer.toBinaryString(x.charAt(i)));
            } else {
                data += (Integer.toBinaryString(x.charAt(i)));
            }
        }
        return data;
    }

    public String XOR(String s1, String key) {
        String result = "";
        for (int i = 0; i < s1.length(); ++i) {
            if (s1.charAt(i) == key.charAt(i % key.length())) {
                result = result + "0";
            } else {
                result = result + "1";
            }
        }
        return result;
    }

    public String binaryToCharacters(String s) {
        String data = "";
        String temp;

```

```

    for (int i = 0; i < s.length(); i = i + 7) {
        temp = "";
        for (int j = i; j < i + 7; ++j) {
            temp += s.charAt(j);
        }
        data += (char) (Integer.parseInt(temp, 2));
    }
    return data;
}

@Override
public Location here() {
    return super.here();
}

@Override
public void takeDown() {
    if (faultTolerance) {
        try {
            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
            msg.addReceiver(new AID(toBeKilled, AID.ISLOCALNAME));
            if (!normalDeath) {
                if (security) {
                    msg.setContent(XOR(to7characters("go"), key));
                } else {
                    msg.setContent("go");
                }
            }
            send(msg);
        } else {
            if (security) {
                msg.setContent(XOR(to7characters("kill"), key));
            } else {
                msg.setContent("kill");
            }
            send(msg);
        }
    } catch (Exception e) {
    }
}

public void done() {
    System.out.println("done()");
}

@Override
public void afterClone() {
    getContentManager().registerLanguage(new SLCodec());
    getContentManager().registerOntology(MobilityOntology.getInstance());
    String contents;

    ACLMessage msg = this.blockingReceive();
    if (msg != null) {
        if (security) {
            contents = "";
            contents = XOR(msg.getContent(), key);
            contents = binaryToCharacters(contents);
        } else {
            contents = msg.getContent();
        }
        if (contents.equals("kill")) {
            doDelete();
        } else if (contents.equals("go")) {

```



```

        if (locs[i + 1] == null &&
this.here().getName().equals(home) && !exists(locs[i])) {
            finish();
            doDelete();
        } else {
            switch (faultToleranceType) {
                case 1:
                    cloneTo = this.here().getName();
                    break;
                case 2:
                    cloneTo = home;
                    break;
            }
            String temp = toBeCloned;
            toBeCloned = msg.getSender().getLocalName();
            toBeKilled = temp;
            workerAgent.this.addBehaviour(new
cloneBeh(msg.getSender().getLocalName(), cloneTo, "go"));
        }
    }
}

@Override
public void afterMove() {
    Behaviour b = new check(this.here().getName());
    this.addBehaviour(b);

    if (locs[i] != null) {
        if (faultTolerance) {
            Behaviour buu = new msgSend("kill");
            this.addBehaviour(buu);
            switch (faultToleranceType) {
                case 1:
                    cloneTo = this.here().getName();
                    break;
                case 2:
                    cloneTo = home;
                    break;
            }
            Behaviour bx = new cloneBeh(toBeCloned, cloneTo, "kill");
            this.addBehaviour(bx);
        }
        Behaviour b2 = new moveBeh(locs[i]);
        this.addBehaviour(b2);
    }
    System.out.println("aftermove(): agent " +
workerAgent.this.getLocalName() + " is now in " + this.here().getName());
}

public jade.util.leap.Iterator Locations() {
    jade.util.leap.Iterator it = null;
    try {
        sendRequest(new Action(getAMS(), new
QueryPlatformLocationsAction()));
        //Receive response from AMS
        MessageTemplate mt = MessageTemplate.and(
            MessageTemplate.MatchSender(getAMS()),
            MessageTemplate.MatchPerformative(ACLMessage.INFORM));
        ACLMessage resp = blockingReceive(mt);
        ContentElement ce = getContentManager().extractContent(resp);
        Result result = (Result) ce;
        it = result.getItems().iterator();
        return it;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return it;
}

```

```

}

public void sendRequest(Action action) {
    ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
    request.setLanguage(new SLLocale().getName());
    request.setOntology(MobilityOntology.getInstance().getName());
    try {
        getContentManager().fillContent(request, action);
        request.addReceiver(action.getActor());
        send(request);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

```

```

class msgSend extends OneShotBehaviour {

```

```

    String msgR;

```

```

    msgSend(String msg) {
        this.msgR = msg;
    }

```

```

    public void action() {
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.addReceiver(new AID(toBeKilled, AID.ISLOCALNAME));
        if (security) {
            msg.setContent(XOR(to7characters(msgR), key));
        } else {
            msg.setContent(msgR);
        }
        send(msg);
    }
}

```

```

class cloneBeh extends OneShotBehaviour {

```

```

    String cloner = "";
    String oldLocation = "";
    String status = "go";

```

```

    cloneBeh() {
    }

```

```

    cloneBeh(String cloner) {
        this.cloner = cloner;
    }

```

```

    cloneBeh(String cloner, String oldLocation) {
        this.cloner = cloner;
        this.oldLocation = oldLocation;
    }

```

```

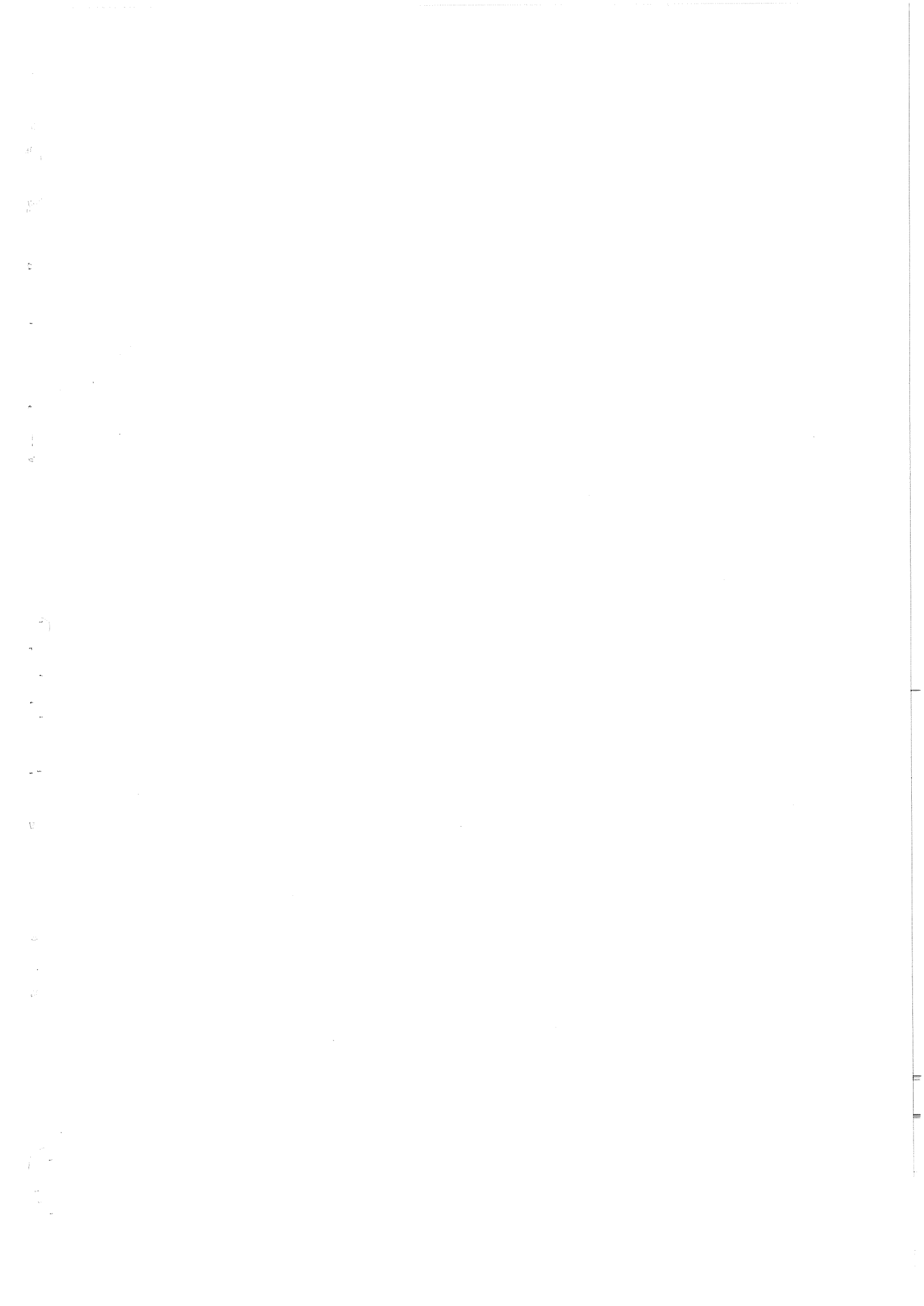
    cloneBeh(String cloner, String oldLocation, String status) {
        this.cloner = cloner;
        this.oldLocation = oldLocation;
        this.status = status;
    }

```

```

    public void action() {
        ContainerID loc1;
        if (i == 0) {
            loc1 = new ContainerID(home, null);
            try {
                workerAgent.this.doClone(loc1, cloner);
            } catch (Exception ex) {
                System.out.println("doClone");
            }
        }
    }

```




```

int filesAfterMod = filesCounter - (filesCounter
% numberOfWorkers);
(agentNumber - 1);
(agentNumber) - 1;
    int filesAfterMod = filesCounter - (filesCounter
start = (filesAfterMod / numberOfWorkers) *
end = (filesAfterMod / numberOfWorkers) *
    if (agentNumber == numberOfWorkers) {
        end += filesCounter % numberOfWorkers;
    }
}
for (int l = start; l <= end; ++l) {
    ++docsCounter;
    ht2.put(docsCounter, "" + container +
childsLevel2[l].getPath());
    HashMap ht = new HashMap();
    try {
        FileReader fr = new
FileReader(childsLevel2[l]);
        BufferedReader br = new BufferedReader(fr);
        fr = null;
        StringTokenizer st = new
StringTokenizer(br.readLine());
        br.close();
        br = null;
        while (st.hasMoreTokens()) {
            ht.put(st.nextToken(), null);
        }
        st = null;
    } catch (Exception e) {
    }
    boolean flag = false;
    for (int j = 0; j < chunksCounter; ++j) {
        boolean flag1 = false;
        if (ht.containsKey(hashValues[j])) {
            if (hashValuesPositions[j] == 0) {
                hashValuesPositions[j] =
docsCounter;
            }
        }
    }
    int j;
    int m = (int) chunksCounter;
    int y = hashValuesPositions[m - 1];
    for (j = m; j < m + chunkSize; ++j) {
        hashValuesPositions[j] = y;
    }
    if (percentageFlag)
    {
        System.out.print("\n" + (int) per());

        if ((int) per() > percentageValue)
            System.out.print(percentageValue);
        break;
    }

    if (myAgent.getState() == 6) {
        break;
    }
}
if (percentageFlag)
{
    if ((int) per() > percentageValue)
        break;
}
if (myAgent.getState() == 6) {
    break;
}
}
}

```

```

    }
    } catch (Exception e) {
        e.printStackTrace();
    }
    for (int z = 0; z < chunksCounter; ++z) {
        try {
            if (!x.contains((String)
ht2.get(hashValuesPositions[z]))) {
                x += "%" + hashValuesPositions[z] + "%" + ((String)
ht2.get(hashValuesPositions[z]));
            }
        } catch (Exception e) {
        }
    }
    ht2.clear();
    if (per() > 70 || locs[i] == null) {
        if (myAgent.getState() != 6) {
            finish();
            normalDeath = true;
            myAgent.doDelete();
        }
    }
}
}
}

```

```

public void finish() {
    ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
    msg.addReceiver(new AID("doc: " + docNumber + ": Document's
Coordinator Agent", AID.ISLOCALNAME));
    String letter = "";
    for (int i1 = 0; i1 < hashValuesPositions.length; ++i1) {
        letter += hashValuesPositions[i1] + " ";
    }
    letter += x;
    if (security) {
        msg.setContent(XOR(to7characters(letter), key));
    } else {
        msg.setContent(letter);
    }
    send(msg);
}

```

```

public double per() {
    double xx = 0.0;
    for (int i2 = 0; i2 < chunksCounter + chunkSize; ++i2) {
        if (hashValuesPositions[i2] == 0) {
            if (!(i2 < chunkSize - 1 && hashValuesPositions[0] == 0 &&
i2 > -1)) {
                for (int j = i2 - 1; j > i2 - chunkSize; --j) {
                    if (hashValuesPositions[j] != 0) {
                        xx++;
                        break;
                    }
                }
            }
        }
        } else if (hashValuesPositions[i2] != 0) {
            xx++;
        }
    }
    return (xx / (chunksCounter + chunkSize)) * 100;
}

```

```

class moveBeh extends OneShotBehaviour {
    String container;
}

```

```

        moveBeh(String cont) {
            this.container = cont;
        }
        public void action() {
            ContainerID loc1;
            getContentManager().registerLanguage(new SLCodec());
        }
        getContentManager().registerOntology(MobilityOntology.getInstance());
        while (!exists(locs[i])) {
            if (locs[i] == null) {
                finish();
                doDelete();
                break;
            }
            ++i;
        }
        loc1 = new ContainerID(locs[i], null);
        ++i;
        workerAgent.this.doMove(loc1);
    }
}

public boolean exists(String loc) {
    jade.util.leap.Iterator it = Locations();
    int i = 0;
    String locationString;
    Location location;
    java.util.List list = new java.util.ArrayList();
    while (it.hasNext()) {
        location = (Location) it.next();
        locationString = location.getName();
        if (locationString.equals(loc)) {
            return true;
        }
    }
    return false;
}
}
}

```

References

- [1]: Nick Jennings and Michael Wooldridge, "Software agents", IEE Review, pp: 17-20, January 1996.
- [2]: R. S. Silva Filho. "Mobile Agents and Software Deployment", ICS280 - Configuration Management and Runtime Change (Fall 2000). University of California, Irvine.
- [3]: Fishman, Stephen. "The Copyright Handbook: How to Protect & Use Written Works", 7th Ed. Nolo, Berkeley, CA, 2003.
- [4]: Fabio Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing Multi-Agent Systems with JADE", John Wiley & Sons, Ltd, Australia, pp: 30-33, 2007.
- [5]: Danny B. Lange. "Mobile Objects and Mobile Agents: The Future of Distributed Computing", In Proceedings of the European Conference on Object-Oriented Programming '98, pp: 1-12, 1998.
- [6]: A. Tanenbaum and M. Van Steen, "Distributed Systems: Principles and Paradigms", Prentice Hall, Pearson Education, USA, 2002.
- [7]: Niklas Borselius, "Mobile agent security" , Electronics & Communication Engineering Journal, ,Volume 14, no 5, IEE, London, UK, pp: 211-218 , October 2002.
- [8]: Uzuner, O., Katz, B. "Non-verbatim copyright infringement detection for text" Retrieved from <http://www.ai.mit.edu/research/abstracts/abstracts2001/information-access/09uzuner.pdf>
- [9]: <http://www.motiondsp.com>
- [10]: O. Uzuner. "Identifying Expression Fingerprints Using Linguistic Information" PhD thesis, Massachusetts Institute of Technology, 2005. Retrieved from <http://publications.csail.mit.edu/tmp/MIT-CSAIL-TR-2005-077.pdf>
- [11]: Microsoft ® Encarta ® 2008. © 1993-2007 Microsoft Corporation. All rights reserved.
- [12]: <http://www.netbeans.org>
- [13]: <http://java.sun.com/javase/technologies/index.jsp#overview>
- [14]: R. Tahboub, V. Lazarescu, "Mobile Agents in WWW Access Optimization", ACIDCA-ICMI'2005 International Conference, IEEE, Tunis, Nov 5-7, 2005.
- [15]: This material comes from Questions and Answers on Copyright for the Campus Community, Copyright ©1997, Association of American Publishers, National Association of College Stores, and Software Publishers Association.
- [16]: www.freepatentsonline.com/7120274.html
- [17]: S. Niezgodna and T.P. Way, "SNITCH: A Software Tool for Detecting Cut and Paste Plagiarism", ACM SIGCSE Technical Symposium on Computer Science Education, pp: 51-55, 2006.

- [18]: C. Collberg, S. G. Kobourov, J. Louie, and T. Slattery, "SPLAT: A System for Self-Plagiarism Detection", IADIS Conference on WWW/Internet, pp: 508-514, 2003.
- [19]: <http://ausweb.scu.edu.au/aw02/papers/refereed/nguyen/paper.html>
- [20]: Joan Ametller, Sergi Robles, Joan Borrell: "Agent Migration over FIPA ACL Messages", Mobile Agents for Telecommunication Applications (MATA) vol.2881 pp: 210-219, 2003.
- [21]: Fedoruk, A., Deters, R.: "Improving fault-tolerance by replicating agents", In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2, ACM Press New York, NY, USA , pp:737-744, 2002
- [22]: <http://www.plagiarismchecker.com/>
- [23]: <http://plagiarism.phys.virginia.edu/Wsoftware.html>
- [24]: <http://www.copyscape.com/>
- [25]: www.aljazeera.net
- [26]: <http://www.articlechecker.com/>
- [27]: <http://www.scriptum.ca>
- [28]: <http://www.plagiarism.tk/>
- [29]: <http://www.doccop.com/>
- [30]: <http://www.mydropbox.com/>
- [31]: <http://www.blackboard.com>
- [32]: <http://turnitin.com>
- [33]: <http://moodle.org/>
- [34]: <http://www.angellearning.com/>
- [35]: <http://www.calt.monash.edu.au/Quality/ETC/agenda/content/>
- [36]: Wentian Li (1992). "Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution". IEEE Transactions on Information Theory 38 (6): 1842-1845. Can be found at: http://www.nslj-genetics.org/wli/pub/ieee92_pre.pdf
- [37]: <http://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf>
- [38]: Cavnar, William B. and Trenkle, John M., "Ngram Based Text Categorization", Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval", April 1994, pp 161-169.
- [39]: Karp and Rabin's original paper: Karp, Richard M.; Rabin, Michael O. (March 1987). "Efficient randomized pattern-matching algorithms". IBM Journal of Research and Development 31 (2), 249-260.
- [40]: Daniel Cubitt , Automatic Detection of Plagiarism. Can be found at: www.dcs.shef.ac.uk/intranet/teaching/projects/archive/ug2005/pdf/u2dc.pdf
- [41]: String (java 2 platform SE v1.4.2), URL: [http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html#hashCode\(\)](http://java.sun.com/j2se/1.4.2/docs/api/java/lang/String.html#hashCode())
- [42]: Miao Kang, Lan Wang, and Kenji Taguchi, Modelling Mobile Agent Applications in UML 2.0 Activity Diagrams, 2004/04/21, <http://www.auml.org/auml/supplements/UML2-AD.pdf>
- [43]: Amal M. Al Dweik "Fault tolerance of Mobile Agents – Centralized Approach". 2nd International Computer Engineering Conference- Engineering the Information

Society ICENCO'2007". Faculty of Engineering, Cairo University. Cairo, EGYPT.
December 26-28, 2006

[44]: Amal M. Al Dweik , "Fault tolerance of Mobile Agents". The International Arab
conference on Information Technology "ACIT2005". 6th to 8th Dec.2005, Amman.

[45]: <http://www.gutenberg.org>

[46]: <http://www.merriam-webster.com/>

[47]: <http://www.google.com>

[48]: <http://www.yahoo.com>

[49]: <http://www.ppu.edu>