2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

A Cooperative Multi-Agent Approach to Computer Forensics

Bruno W. P. Hoelz, Célia G. Ralha, Rajiv Geeverghese and Hugo C. Junior
Computer Science Department, University of Brasília
Campus Universitário Darcy Ribeiro
Caixa Postal 4466 - CEP 70.919-970 - Brazil
werneck.bwph@dpf.gov.br, ghedini@cic.unb.br,
vect0rius@yahoo.com.br, hugo.csj@gmail.com

Abstract

This article proposes the use of a collaborative multi-agent approach to develop a toolkit to assist the experts during the forensic examination process: **MADIK** *- a* **M***ulti-***A***gent* **D***igital* **I***nvestigation Tool***K***it. The use of a multi-agent approach has been proved adequate, specially regarding the cooperative action of the autonomous specialized agents: HashSetAgent, FilePathAgent, TimelineAgent, FileSignatureAgent. Also the distributed nature of the multi-agent approach allows for better usage of computational resources, since agents can operate autonomously in different machines and environments. As part of our work, we have defined a four layer multi-agent architecture, as a metaphor to the organizational hierarchy levels, which is divided in strategic, tactical, operational and specialist levels. The proposed architecture was the base to the development of the toolkit, which was developed with a blackboard approach, implemented over the* Java Agent DEvelopment Framework - JADE, *using* Java Expert System Shell - JESS. *We have done some experiments with MADIK using real data and the results are encouraging. This paper focuses on the benefits of using the multi-agent approach to aid in the forensic examination process, specially regarding the cooperative action of the autonomous specialized agents, which we deem as a flexible and promising possibility that should be further explored in the computer forensics scenario.*

1. Introduction

Computer Forensics consists of examination and analysis of computational systems, which demands a lot of resources due to the large amount of data involved. Thus, the success of computer forensic examinations depend on the ability to examine large amounts of digital forensic data, in search of important evidences. Forensic examination consists of several steps to preserve, collect and analyze evidences found in digital storage media, so they can be presented and used as evidence of unlawful actions. The constant growth in the capacity of digital storage media and the wide-spread presence in everybody's daily life represents also a growth in the demand for forensic examinations.

But the current set of forensic tools are not robust enough when it comes to the need to analyze a great number of evidences and correlate the findings. As a consequence, computer forensic experts' work is excessively time consuming. Also the computational resources required to do such examinations are a problem, since most of the forensic tools have no distributed processing capabilities. What we aim with this research is to develop a toolkit to help experts during specialized forensic examinations. The tool is intended to improve the experts performance, in a more intelligent and flexible way, when compared to the help offered by the current available tools. In addition, the tool includes distribution of tasks in the examination processes and correlation of findings, which is not available in any tool at the moment.

In order to reach a better use of human and computational resources, we propose the use of a multi-agent approach. A multi-agent system (MAS), can be defined as a computational system composed by more than one intelligent software agent [6, 17, 18, 4]. An intelligent software agent (ISA) uses Artificial Intelligence (AI) modern approach to interact with the environment, perceiving and acting autonomously over it to achieve defined goals [11, 15]. Thus, a MAS is a system, where many agents interact with the environment in a cooperative or competitive way to achieve individual or group objectives.

The rest of the article presents in Section 2 an overview of the computer forensics problems. Section 3 explains the proposed architecture and MADIK. Section 4 discuss some experiments and results; while Section 5, presents some conclusions and points to future work.

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.55

2. Overview of the Problem

At real computer forensic cases, experts can't define at first what evidence is more relevant to the incident or crime under investigation. Thus, a pre-analysis of the suspect machines would limit the number of evidences collected for examination, reducing the time of investigation and analysis by the forensic experts. But the lack of intelligent and flexible tools to help forensic experts with the pre-analysis phase, and also with a concrete cross-analysis of large number of potential correlated evidences, is a reality. What really happens is that the computers and storage media are analyzed separately, because of the lack of these tools. As a result, a large number of potentially related evidences are lost during the forensic examinations. This is not only a problem to forensic computing, but also to network incident response.

Some solutions to these problems have been proposed by many research projects. In [1], they propose a multi-tier, hierarchical framework to guide digital investigations. The framework includes objectives-based phases and sub-phases, that are applicable to various layers of abstraction, and to which additional layers of detail can easily be added as needed. Authors propose six phases which includes many tasks: preparation, incident response, data collection, data analysis, presentation of findings and incident closure.

As presented, the forensic examination occurs in the data analysis phase and includes various tasks. This research work focus on data collection, data analysis and presentation of findings. We propose a framework, for distributed digital forensics, based on a multi-agent approach; more specifically, we propose a four layer multi-agent architecture and MADIK - a **M**ulti-**A**gent **D**igital **I**nvestigation Tool**K**it, developed as a proof-of-concept prototype, implemented using the JADE framework [8].

The characteristics of forensic examination of computational systems result in a complex task, which demands special abilities from forensic experts. The resources are always limited, not only in terms of human experts and their knowledge, but also in terms of computational resources and time available to perform the examinations. These limitations have a direct impact on the examination's quality. The process cited by [1], serve as basis of analysis that might be considered for the conception of new specialized agents. There are some automated tools to help the investigation process as analyzed in [7], such as: (i) full-text indexing, e.g. dtSearch^{TM}, (ii) password breaking, e.g. AccessData Password Recovery ToolKit (PRTK^{TM}), (iii) file recovery, e.g. Ontrack Easy Recovery^{TM}, (iv) comparing file hashes against hash sets, e.g. KFF functionality of AccessData Forensic ToolKit (FTK^{TM}), and (v) file path and time line analysis, e.g. EnCase^{TM} (Guidance Software Inc.).

3. The Proposed Architecture and MADIK

In real forensic examinations, not all of the cited tasks are necessary or possible, because of the limitations already discussed. But a constant necessity in all investigations is the distribution and coordination of tasks amongst the team of specialists. Thus, in the proposed approach, different roles are played by different levels of agents, similarly to organizational hierarchy levels, as used for example in the work of intelligent distributed system for strategic decision making of [14]. Thus, we propose a four layer multi-agent architecture, as a metaphor to the organizational hierarchy levels, which is divided in: strategic, tactical, operational and specialist.

With the four layer architecture, we define autonomous agents, each specialized in a small and distinct subset of the overall objectives and constraints. With this mechanism, agents can collaborate by observing and modifying one another's work, through the use of a common base named blackboard [13, 9, 16, 3]. With a blackboard approach, the convergence to good solutions, for a variety of real problems, has been obtained, by embedding a few simple rules in each agent. Figure 1 presents the idea of the operational manager, where the blackboard is being used by the specialized agents accessing local data of a specific computer.

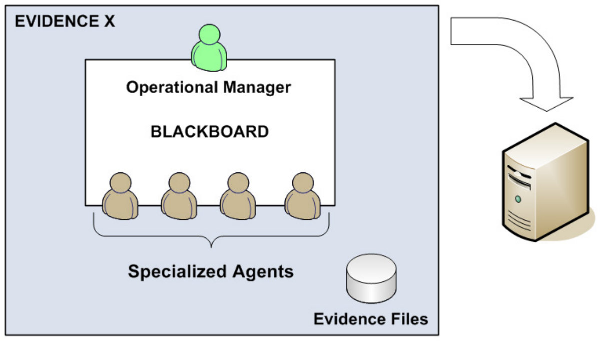


Figure 1. The operational manager actuation.

The distributed nature of MAS allows for better usage of resources, since agents can operate autonomously in different machines and environments. Also, with the hierarchy used, the distribution of tasks among specialized agents might be properly used. Thus, the four layers architecture defined includes at the operational level the specialized agents, at the higher levels managers to strategic and tactical decisions, that will distribute and coordinate the execution of tasks by the specialized agents. For these, the communication protocol established uses the hierarchy to communicate, since the specialized agents communicate only with the operational manager, the operational manager with the tactics manager and the tactics to the strategic manager.

Figure 2 presents the architecture, which is divided into four layers, as a metaphor to the organizational hierarchy levels, named strategic, tactical, operational and specialist levels. In MADIK, the strategic manager receives the requests for investigation cases and distributes them to the tactical managers. The tactical manager will assign each evidence that belong to its case to one of its operational managers. The operational manager has the objective of finding appropriate specialized agents to examine the evidence it has received from its manager. Note that the operational manager has an important role in the architecture, since it determines which specialists will be employed.

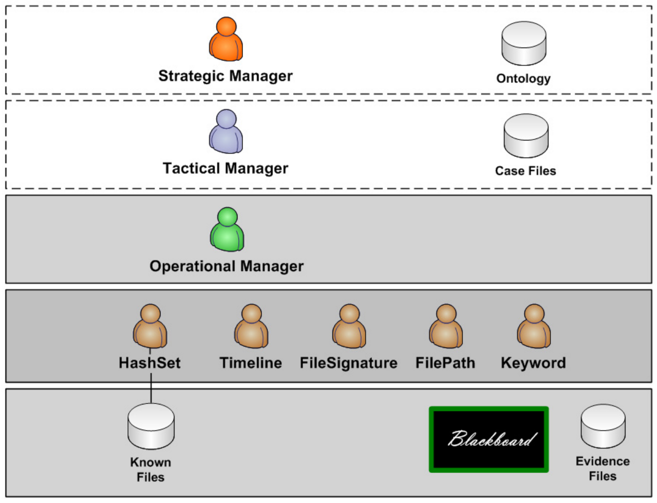


Figure 2. MADIK's architecture.

3.1. MADIK's Overview

MADIK was implemented using the *Java Agent DEvelopment Framework - JADE*, which is fully developed with Java language [8, 2]. JADE was used since it simplifies the implementation of multi-agent systems, through a middleware, that claims to comply with the *Foundation for Intelligent Physical Agents - FIPA*[1] specifications. JADE has a set of tools that supports the debugging and deployment phase. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can also be changed at run-time, by moving agents from one machine to another. The reasoning process of the intelligent software agents was implemented using *Java Expert System Shell - JESS* [5] as an inference engine.

[1]FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies (`http://www.fipa.org/`).

MADIK has been implemented to help experts during the forensic examination process. The specialized intelligent software agents already implemented include:

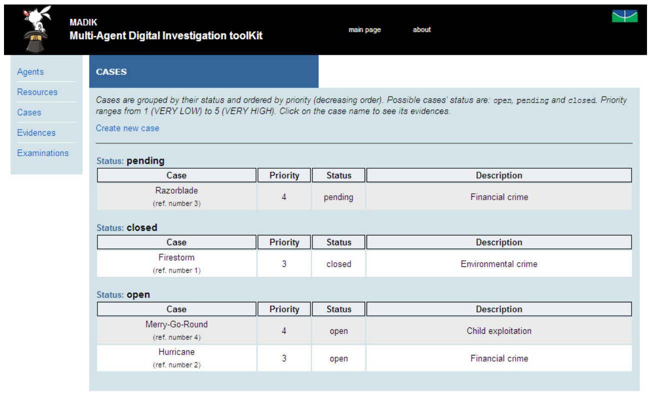
- `HashSetAgent` calculates the MD5 hash from a file and compares it with its knowledge base, which contains sets of files known to be ignorable (e.g. system files) or important (e.g. contraband files like child exploitation imagery). We might cite that some of these hash sets contain more than 10 million hash values, from different softwares, as cited in [12]. Also we might note that the bigger the hash set the longer the comparison takes.
- `FilePathAgent` keeps on it's knowledge base a collection of folders which are commonly used by several application which may be of interest to the investigation like P2P (peer-to-peer), VoIP and instant messaging applications.
- `FileSignatureAgent` examines the file headers (the first 8 bytes of the file), to determine if they match the file extension. If someone changes the file extension in order to "hide" the true purpose of the file, this will be detected by this agent. It also keeps a list of commonly used prefixes and file names, such as the ones used by digital cameras.
- `TimelineAgent` examines dates of creation, access and modification to determine events like system and software installation, backups, web browser usage and other activities, some which can be relevant to the investigation.

Note that in Figure 2, there is also a KeywordAgent, which is already implemented, but has not been used in this article's experiments.

The following *JESS* code presents the `HashSetAgent` inference rule. Note that there are two templates being used: (i) `file` with file attributes like name, extension and size and (ii) `known-file` with input data from the HashSetAgent knowledge base.

```
(defrule known-file-filter
  "Known file filter"
  ?f <- (file)
  ?kf <- (known-file { hash == f.hash }
  =>
  (assert (blackboard-fact
    (id ?f.id)
    (agent ?f.agent)
    (decision ?kf.decision)
    (description ?kf.description)))
    (printout t "Known file with status "
    ?kf.decision " inserted in the
blackboard (" ?kf.description ")" crlf))
```

Figure 3 presents MADIK's web interface displaying some cases being examined. Note that there are four cases registered, which are grouped by their status and ordered by priority. The possible case status are: `open`, `pending` and `closed`. The `pending` status is used when the case is completed, but the examiner has not yet reviewed its results. Note that each case has a short description and the priority, which ranges from 1 (VERY LOW), 2 (LOW), 3 (NORMAL), 4 (HIGH) to 5 (VERY HIGH). In the figure, we can see the options menu of the web interface which includes: `Agents`, `Resources`, `Cases`, `Evidences` and `Examinations`.



Cases are grouped by their status and ordered by priority (decreasing order). Possible cases' status are: `open`, `pending` and `closed`. Priority ranges from 1 (VERY LOW) to 5 (VERY HIGH). Click on the case name to see its evidences.

Status: **pending**

Case	Priority	Status	Description
Razorblade (ref. number 3)	4	pending	Financial crime

Status: **closed**

Case	Priority	Status	Description
Firestorm (ref. number 1)	3	closed	Environmental crime

Status: **open**

Case	Priority	Status	Description
Merry-Go-Round (ref. number 4)	4	open	Child exploitation
Hurricane (ref. number 2)	3	open	Financial crime

Figure 3. Overview of some cases being examined by the MADIK.

During the specialists' execution, they will insert their conclusions and remarks in the blackboard. The most important fields are:

- `recommendation` - the actual recommendation issued by the agent. There are three distinct levels: (i) `ignore` - the strongest recommendation to ignore a file, indicating its unimportant according to the agent, (ii) `alert` - strongly recommends the selection of a file, and (iii) `inform` - this recommendation is an intermediate value, which contains information to help the human reviewer to decide whether to select that file or not. There can be an additional sign (**+** or **-**) representing an ignore or include bias, respectively. When the recommendations diverge, we have a conflict that must be solved by the operational manager.

- `description` - the description is part of the explanation mechanism, since it describes the reasoning behind the recommendation and

- `time taken` - this field is for benchmark and profiling, since it registers the time taken (in mileseconds) by the agent to examine the file.

At the moment, the conflicts are being solved in a very naïve way, since all the facts inserted in the blackboard, concerning evidences to the forensic examination, are considered independently of other specialized agents opinion. This is being done based on the fact that it's better to include an evidence than leave it out of investigation analysis, if we are not sure of its' importance. As future work, we may implement a conflict resolution mechanism to consider uncertainty level, through the use of applied statistics and probability theory, for example using dimensional dependence modeling [10].

3.2. Performed Calculations

The different managers of the architecture layers use different calculations to distribute work, according to organizations' priorities, resources' availability and capability, file sizes and case evidences related to different areas. The strategic manager uses the relative case priority (RCP) to determine how the resources will be allocated to each case. The RCP is calculated using the sum of all cases' priorities (SCP). The calculated resources are passed to the tactical managers.

$$SCP = \sum_{i=1}^{n} P(C_i) \quad and \quad RCP(c) = \frac{P(C)}{SCP}$$

On their turn, the tactical managers have to calculate each evidence they have to examine, so they can determine how many of the available computers will be assigned to each evidence. Similarly to the strategic manager, they use the relative evidence priority within a case (REP), which is calculated using the sum of evidence's priorities within a case (SEP).

$$SEP(c) = \sum_{i=1}^{n} P(E_i), \quad \text{where } E \in c$$

$$REP(e) = \frac{P(e)}{SEP(C(e))}, \text{ where } C(e) \text{ returns the case with } e \text{ evidence}$$

In case there is the necessity to know the global priority of an evidence (GEP), considering all the evidences from every case, we use REP and RCP as follows:

$$GEP(e) = REP(e) * RCP(C(e))$$

4. Experiments

In order to validate MADIK we did experiments using data from a real investigation case (Case 3). The data consist of 450 thousand files from seven different hard drives belonging to the same investigation. The investigation's main objective was to get evidence of smuggling activities and to clarify if possible the *modus operandi* of the suspects.

To test the priority calculations and resource allocation, additional evidences where added to two fictional cases (Cases 1 and 2). Doing that we have a situation with three open cases. The platform starts with the strategic manager which determines the existence of three open cases. In these experiments the case priorities were assigned arbitrarily. The number of machines was set to 12, also arbitrarily to simulate a situation of insufficient resources. The strategic manager calculates the relative priority of each case (RCP), using the formulas presented in Section 3.2 and shown in Table 1.

Table 1. Results of three open cases.

CASE	PRIORITY	RCP	Resources
Case 1	NORMAL (3)	3/10 = 0.3	3.6
Case 2	NORMAL (3)	3/10 = 0.3	3.6
Case 3	HIGH (4)	4/10 = 0.4	4.8

The fractional parts are given to the case with highest priority. Cases 1 and 2 get 3 machines each (resources = 3.6 in Table 1). Case 3 gets 6 machines (resource = 4.8 + 1.2 from the fractional parts of Cases 1 and 2 in Table 1). After determining the available resources to each case, the strategic manager sends a message to the tactical managers, informing which case is assigned to him and which computers are available to him. Let's follow TacticalManager-3, responsible for Case 3 (the real case), which has six machines at his disposal. The TacticalManager-3 queries the database to find the evidence related to Case 3. It then performs the same calculations to determine which evidences will receive the resources first. It finds seven evidences in the database, as shown in Table 2. The tactical managers responsible for other cases will do the same, with their respective evidences.

One interesting possibility we explored to determine the evidences priority level was to perform a pre-analysis, to quickly obtain some interesting facts about the evidences. For instance, while counting the number of e-mail messages found on the case, Evidence 7 presented 83% of all e-mail messages found in this case. Thus, this one was given the `VERY HIGH` priority. On the other hand, Evidence 1 presented only ten thousand files, less than 3% of the case files and almost 10% of its files were zero-sized files (including folders). Thus, Evidence 1 got the `VERY LOW` priority.

Keep in mind that those priorities are currently set by human experts, generally based on prior knowledge of the investigation and examiner experience. This pre-analysis is a new possibility to determine which evidence is more likely to produce better results when examined first.

Table 2. Results of Case 3 done by TacticalManager-3.

EVIDENCE	PRIORITY	REP	Resour.
Evidence 1	VERY LOW (1)	1/20 = 0.05	0.3
Evidence 2	LOW (2)	2/20 = 0.1	0.6
Evidence 3	LOW (2)	2/20 = 0.1	0.6
Evidence 4	NORMAL (3)	3/20 = 0.15	0.9
Evidence 5	NORMAL (3)	3/20 = 0.15	0.9
Evidence 6	HIGH (4)	4/20 = 0.2	1.2
Evidence 7	VERY HIGH (5)	5/20 = 0.25	1.5

Since the resources are insufficient to examine all the evidences at the same time, the manager decides to limit the number of evidences to three at a time. The calculations are performed again, with the three highest priority evidences. Evidence 7 gets 3 machines, Evidence 6, 2 machines and Evidence 5, 1 machine.

One operational manager is then assigned to each case. They instantiate the agents they have available, which were described earlier. Each one of this agents examines the files and put their findings on the blackboard. With this collaborative approach, the tactical level is now able to correlate findings from different evidences. Such correlation is not possible with the most commonly used tools. One simple example is the discovery of communications that originated on one computer and were received on the other, as shown in Figure 4.

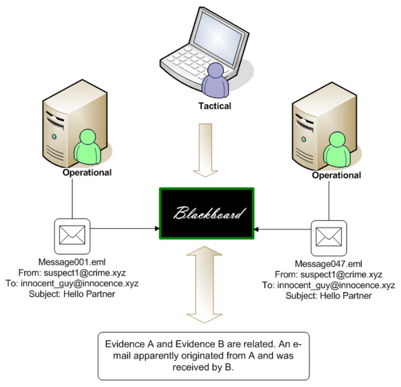

Tactical
Operational
Operational
Blackboard
Message001.eml
From: suspect1@crime.xyz
To: innocent_guy@innocence.xyz
Subject: Hello Partner
Message047.eml
From: suspect1@crime.xyz
To: innocent_guy@innocence.xyz
Subject: Hello Partner
Evidence A and Evidence B are related. An e-mail apparently originated from A and was received by B.

Figure 4. The correlation scheme through communication.

4.1 Results

After the execution of the four specialized agents in all seven evidences, we evaluated the obtained results considering the factors: (i) reduction, (ii) coverage, (iii) time taken to examine the evidences. We also observed the number of conflicts found in the blackboard, that is, the number of conflicting decisions inserted by different agents. A final correlation examination was made, although at the moment no specialized agent was conceived to do such task. Table 3 presents the results of reduction and coverage obtained by the four specialized agents.

Table 3. Results obtained by the specialized agents.

AGENT	REDUCTION	COVERAGE
HashSetAgent	42%	69%
FilePathAgent	30%	64%
FileSignatureAgent	25%	45%
TimelineAgent	5%	74%

- The `HashSetAgent` found 140,000 duplicate files, 30,000 zero-sized files and another 6% of files that belonged to the Windows 98 hash set. Considering some overlapping in these categories, the final reduction suggested by the `HashSetAgent` was 42%. In terms of coverage, it analyzed 69% of the case files where a hash calculation were possible, just like any other hash analysis tool. The `HashSetAgent` found no files with the `Alert` status.

- The `FilePathAgent` suggested the removal of 42,000 small image files (less than one cluster of logical size) located on the Temporary Internet Files folder. It also suggested that the 1,761 cookies should be ignored. Other suggestions included ignoring almost 89,000 Java-related files and another 3,528 program documentation files. The total reduction suggested was of 30% of the files. The `FilePathAgent` suggested the inclusion of the 5,875 files that were located in user-related folders (e.g. My Documents). The coverage of this agent in this case reached 64%.

- The `FileSignatureAgent` suggested the inclusion of 17 digital photographies, 155 cookies with possible interesting information about webmail sites and another 1,709 temporary internet files which could be related to webmail. Some 153 files had an extension that did not match with the file header (first bytes of the file), thus being assigned an `Alert` status. The agent suggested to ignore 113,000 files based on their types (executable files like DLL, EXE, VXD and Java class files, which represented approximately 25% of the files). The coverage was of about 45%.

- The `TimelineAgent` detected the creation of about 23,000 files related to the installation of Java-related files on two evidences on the same day, possibly an update, a reduction of 5%. It suggested the examination of 8,800 files used on the last 30 days, which represents only 2% of the files. Its analysis reached 74% of the files.

Regarding the reduction factor (the number of files which the agents suggested we could ignore), the final percentage obtained was of 62%, disregarding the repeated occurrences of files which were ignored by more than one agent. The total coverage of the analysis was about 80%, meaning that 80% of the files were examined by at least one agent. The examination of the seven hard-drives took about four hours using one machine with four cores and four gigabytes of RAM. The MD5 hashes were already calculated and were not added to the total time. The total size of all the files in the case where approximately 113 GB. The time taken by the human examiner to perform the same examination was of about 25 hours as taken from the examiner's report. This time does not take into account the time required by the forensic tool used (FTK^{TM}) to recover files, generate thumbnails from images or calculate the MD5 hashes. We considered only the time needed to examine the file, do keyword searches the examiner deems necessary and select the interesting files.

Most of the decisions of an agent presented no conflicts with the others. The conflict set was smaller than 1% of the files. One conflict, for instance, occurred between the `FilePathAgent` and the `FileSignatureAgent`. The first one suggested to ignore all the cookie files, while the second suggested the examination of 155 of those cookies. On the other hand, both agents agreed on ignoring the Java-related files.

In terms of correlation, we searched the blackboard for executable files in two or more evidences. The results indicated the presence of an custom-made application found on 3 evidences. The number of user-related files on multiple evidences was of only 51 files, consisting mostly of documents and spreadsheets.

5. Conclusions

This paper described a four layer multi-agent architecture, as a metaphor to the organizational hierarchy levels, which is divided in strategic, tactical, operational and specialist levels. **MADIK** - **M**ulti-**A**gent **D**igital **I**nvestigation Tool**K**it a proof-of-concept prototype was implemented.

Its goal is to help experts during the forensic examination process, through the use of four autonomous specialized agents: HashSetAgent, FilePathAgent, TimelineAgent, FileSignatureAgent.

MADIK seeks to perform forensic examinations in a more intelligent and flexible way, specially when compared to the currently available tools, since the toolkit includes distribution of tasks in the examination processes and allows for correlation of findings. With the proposed architecture, we look forward to the possibility that all the information retrieved by the specialized agents can be correlated amongst computers of an investigation, thus achieving a higher level of completeness than today's tools and procedures can provide.

The results obtained by this set of agents were compared to those obtained by the human examiner. The results were similar in the set of selected files. The expert's results showed a greater reduction factor, but the time required to do so was about 6 times greater than that of the MADIK. With the creation of new specialized agents, we expect to obtain results that more closely resemble those of the human expert. In terms of better computational resource usage, MADIK proved to be an interesting solution, with great reduction in the total time and also in the volume of files that would require further examination. The reduction factor of 62% was surprisingly high, what proves the importance of using the MADIK as a pre-analysis tool. The coverage factor of 80% was also a good result, which can be improved by the inclusion of other specialized agents, but may vary depending on the nature of the investigation.

MADIK has prove to be useful in the assistance of the experts during the forensic examination process. The cooperative action of the autonomous specialized agents showed good results considering the reduction of time and number of files that require examination by an expert with adequate coverage of the analysis.

As future work we consider important the definition and implementation of new specialized agents such as the WindowsRegistryAgent, capable of examining the Windows operating system registry files. We also consider important making new experiments with these new agents, including the KeywordAgent, which is already implemented. The planning mechanism among the four hierarchical levels of the architecture must be better defined and implemented. The inclusion of case-based reasoning is also considered to improve the reasoning of the specialized agents, thus reaching more accurate results.

References

[1] N. Beebe and J. G. Clark. A hierarchical, objectives-based framework for the digital investigations process. *Digital Investigation*, 2(2):147–167, 2005.

[2] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology, Sussex, England, 2007. ISBN 978-0-470-05747-6.

[3] D. D. Corkill. Collaborating Software: Blackboard and Multi-Agent Systems & the Future. In *Proceedings of the International Lisp Conference*, New York, USA, October 2003.

[4] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer Series in Agent Technology, Berlin, Germany, 2^{nd} revised and extended edition, 2004. ISBN 3-540-40700-6.

[5] E. F. Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

[6] M. N. Huhns and M. P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francico, USA, 1998. ISBN 1-55860-495-2.

[7] G. Inc. Marketscope for e-discovery and litigation support vendors 2007. *Gartner RAS Core Research Note G00152876*, ID Number: R2582 12212008, 2007.

[8] T. Italia Lab (TILAB). Java Agent DEvelopment framework - JADE. Online. `http://jade.tilab.com`.

[9] V. Jagannathan, R. Dodhiawala, and L. Baum, editors. *Blackboard Architectures and Applications*. Academic Press, Orlando, FL, USA, 1989.

[10] D. Kurowicka and R. Cooke. *Uncertainty Analysis with High Dimensional Dependence Modelling*. Wiley Series in Probability and Statistics, 2006. ISBN 978-0-470-86306-0.

[11] G. F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, USA, 4^{th} edition, 2002. ISBN 0-201-64866-0.

[12] S. Mead. Unique file identification in the national software reference library. *Digital Investigation*, 3(3):138–150, 2006.

[13] H. P. Nii. Blackboard systems, part one: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53, 1986.

[14] S. Pinson and P. Moraïtis. An intelligent distributed system for strategic decision making. *Group Decision and Negotiation*, 6:77–108, 1996.

[15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, USA, 2^{nd} edition, 2002. ISBN 0-13-790395-2.

[16] H. Velthuijsen, editor. *The Nature and Applicability of the Blackboard Architecture*. PTT-Research, Maastricht, 1992.

[17] G. Weiss, editor. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, USA, 2^{nd} edition, 2000. ISBN 0-262-23203-0.

[18] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Ltd., Sussex, England, 2002. ISBN 0-471-49691-X.