Proceedings of the International Multiconference on Computer Science and Information Technology pp. 55–61
ISBN 978-83-60810-14-9
ISSN 1896-7094

An Agent Based System for Distributed Information Management: a case study

Martijn Warnier, Reinier Timmer, Michel Oey, and Frances Brazier
Intelligent Interactive Distributed Systems, Faculty of Sciences
VU University Amsterdam, the Netherlands
{warnier, rjtimmer ,michel, frances}@cs.vu.nl

Anja Oskamp
Computer Law Institute, Faculty of Law
VU University Amsterdam, the Netherlands
a.oskamp@rechten.vu.nl

***Abstract*—Securely managing shared information in distributed environments across multiple organizations is a challenge. Distributed information management systems designed to this purpose, must be able to support individual organizations' information policies whilst ensuring global consistency and completeness of information. This paper describes a multi-agent based prototype implementation of a distributed information management system, for distributed digital criminal dossiers. The prototype implementation runs on the multi-agent platform AgentScape.**

I. Introduction

MANAGING shared information securely and efficiently across multiple independent organizations is a challenge [2], [14]—the challenge this paper addresses. Individual organizations manage their own information locally on their own systems, according to their own information policies. If information is to be shared, however, additional system-wide policies and mechanisms are needed to manage global correctness and consistency.

One example of an environment in which information needs to be shared between multiple semi-independent organizations is the environment in which the Public Prosecution compiles and manages digital criminal dossiers. These criminal dossiers are currently being used in the Courts of Amsterdam and Rotterdam in a number of pilot studies. Earlier work in the Agent-based Criminal Court Electronic Support Systems (ACCESS) project [11], [12], [13] has focused on the high-level design and security of a distributed agent based architecture [4] that manages information based on the notion of *distributed digital criminal dossiers*. This paper describes a prototype implementation of a distributed information management system, that runs on the multi-agent platform AgentScape [7].

The main advantage of a distributed architecture is that it allows *physically* distributed information sources, such as Municipalities and the Prison Systems, to be, and thus remain, responsible for the integrity of their own information content in a digital dossier. By law, the Public Prosecution is responsible for the compilation and maintenance of criminal dossiers. Dutch law also dictates which information can and must be exchanged to facilitate compiling a criminal dossier[1].

In our approach the Public Prosecution has a central role and is responsible for providing the infrastructure that enables other organizations to securely add information and securely access information in criminal dossiers. Together, these organizations form a semi-open environment: an environment in which organizations are known and have control over their own information. This paper discusses some of the details involved in implementing a distributed agent-based information management system that allows *secure* information *exchange* across multiple organizations. In contrast, earlier work [1], [6] on distributed agent-based information management has mainly focused on *finding* information in *open* network, i.e., the Internet.

The remainder of this paper is organized as follows. The next section gives some background on the context in which the ACCESS prototype should function as well as on the AgentScape platform. Section III illustrates the design of the system in some detail and Section IV gives more information on the actual ACCESS prototype implementation. The paper ends with a discussion and conclusions.

II. Background

This section briefly introduces the notion of *Distributed Digital Criminal Dossiers* compiled and managed by the Public Prosecution. See [11], [12], [13] for more detail of compilation and management of distributed dossiers in complex and adaptive environments. This section also briefly introduces the AgentScape [7] agent platform on which the prototype runs.

A. Context

In the semi-open environment in which the Public Prosecution compiles

and manages digital criminal dossiers, information is shared between multiple semi-independent organizations. On an experimental basis and in specific pilot studies such criminal dossiers have been used in the Courts of Amsterdam and

[1]Please note that all legal and procedural details discussed in this paper are interpreted in the context of Dutch law, but can be extended to legislation of other jurisdictions.

Rotterdam. This paper describes the architecture and implementation of a distributed multi-agent system architecture [4], designed to improve consistency, completeness, integrity and security of the information in digital criminal dossiers.

The physically distributed sources of information distributed over different organizations are depicted in Figure 1 below[2].

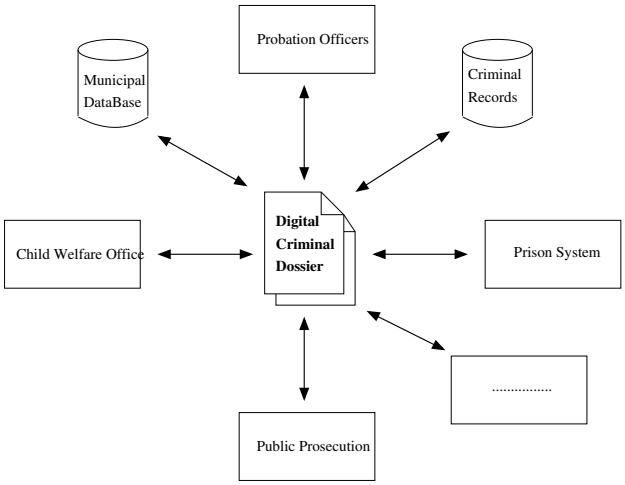

Fig. 1. Information Sources for a Digital Criminal Dossier

The Public Prosecution determines if (and when) an individual is to be prosecuted based on the law and the information available. Once decided, the Public Prosecution creates a criminal dossier for the case. Each newly created criminal dossier contains standard (meta-)data, and crime specific (meta-)data.

Different organizations are responsible for different information in the criminal dossier. Administrative information, for example, is managed and maintained by the defendant's local authorities[3] and information on a juvenile defendant's family situation and social environment is the responsibility of the Council for Child Welfare[4]. Distributing both the *data* (information) and the *responsibility* for the data ensures that information in digital dossiers is kept as up-to-date as possible. Changes in data are flagged by the relevant organizations and transmitted to the Public Prosecution for synchronization of the complete (distributed) criminal dossier.

The criminal dossier as a whole is the responsibility of the Public Prosecution whilst relevant data are maintained and stored by the organizations responsible for the data. The Public Prosecution is responsible for synchronization of the data within the digital criminal dossier as a whole and for the final version of the dossier that is sent to the judge for judgement in trial.

Note that the term 'dossier' is overloaded. In this paper the focus is mostly on *criminal* dossiers. Other dossiers include administrative dossiers, dossiers managed by the Council for Child Welfare, and, in general, all documents types that can contain references to other documents. In the remainder of this paper the term 'dossier' refers to dossiers in general, while 'criminal dossiers', 'administrative dossiers' etc, refer to specific dossier types.

B. AgentScape

AgentScape [7] is a multi-language mobile agent middleware platform designed to support scalable, secure, distributed multi-agent applications. Agents can migrate between virtual domains called *locations* and can negotiate resource requirements before migrating [5]. An AgentScape location consists of one or more hosts running the AgentScape middleware, typically within a single administrative domain. Each AgentScape location runs a *location manager* process on one of its hosts. Each host has its own *host manager*. An example of two AgentScape hosts is depicted below in Figure 2. Multiple locations together with a lookup service form an AgentScape *world*: a distributed multi-agent system. The prototype implementation, discussed in this paper, is an example of an AgentScape world.

The AgentScape *kernel* provides low-level secure communication between the higher level middleware processes and between agents, and provides secure agent mobility. *Agent servers* provide language dependent run-time environments for agents, *Web service gateways* [8] support Web service access and monitoring using the SOAP/XML protocol. Two versions of the kernel have been implemented, in Java and in C.

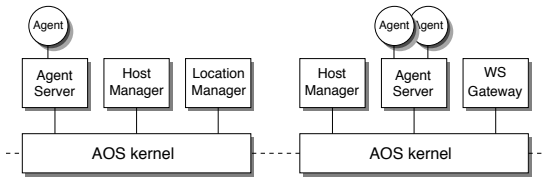

Fig. 2. The AgentScape middleware architecture.

III. Application Design

The ACCESS prototype application is designed to be managed decentrally. Data is stored in digital criminal dossiers, fields of which are distributed among different locations (organizations). The contents of each field in a criminal dossier is automatically updated by the location responsible. This section describes how the information and locations are structured.

A. Distributed Digital Criminal Dossiers

Dossiers are implemented with a light weight 'skeleton' framework based on XML. Fields in this XML document

[2]In this context, the Police have a special role. The Police provide information needed for compiling criminal dossiers by others (Public Prosecution, Probation officers etc), but cannot, for example, change information in criminal dossiers once a criminal dossier has been created by the Public Prosecution. Thus, information exchange between the Police and other organizations does not occur via distributed digital criminal dossiers.

[3]In the Dutch context all citizens are obliged to be registered in one and only one Municipality. Municipalities are responsible for keeping administrative records of their residents. Please note that not all countries have such administrative records at the governmental level, e.g. the USA do not maintain any administrative records at this level.

[4]In the Dutch context, it is the task of the Council for Child Welfare to investigate all aspects of a crime that involves minors, including the family situation and other relevant social factors, and to provide a motivated advice for suitable punishment (if applicable) of the suspect to the Courts.

contain keywords and references to other (possibly, but not necessarily, XML-based) documents. The distributed nature of the dossier is acquired through references to documents on both local and remote systems.

The root of each criminal dossier, created by the Public Prosecution, always includes the dossier number, creation date, access control lists, type of offense, (a reference to) the defendant's administrative data (in an administrative dossier), and the original police report concerning an incident. It also includes mandatory and optional information for each specific offence. Such information is used by the Public Prosecution to check completeness of a criminal dossier—a criminal dossier is only complete if all mandatory information is present. It provides the structure needed for automated completeness checks [12].

This paper assumes that standard XML templates exist for each type of offence, and that these templates are used by the Public Prosecution to structure the meta-data in a criminal dossier[5].

A distributed digital criminal dossier thus consists of a number of documents in a networked structure distributed over physically distributed locations with the Public Prosecution as the central coordinator. Note that a digital criminal dossier may only be referenced by other digital criminal dossiers. All digital criminal dossiers themselves are controlled by the Public Prosecution, an important security requirement. The root of each criminal dossier, maintained by the Public Prosecution, also contains information such as logging information on who altered or accessed information when, and when the last backup of the dossier was made.

B. Principals

Many users interact with the information management system (using agents) concurrently, from different locations (organizations). These users can be divided into three types with different administrative rights:

1) *Users of the information management system*
   Users of the information management system interact (read, write) with dossiers to perform their jobs as judges, (legal) clerks, public prosecutors, etc. As users of the information management system they cannot add or remove dossiers and templates, and they don't have permission to change the configurations of the system.
2) *System administrators with administrative rights of the infrastructure*
   Administrators can be further divided into world and location administrators. The world administrator, a person assigned this role by the Public Prosecution, maintains the AgentScape world. Note that the world includes the lookup service and the central user and template repositories (also see Section IV-C). Location administrators, are assigned their roles by the locations they represent, have the task to maintain local configurations and infrastructure, i.e., the host machines, within their location. Both types of administrators can add users to the system and specify their roles (judge, prosecutor etc).
3) *The owner of the criminal dossiers—the Public Prosecutor*
   The Public Prosecutor is the owner of all criminal dossiers. The Public Prosecutor is the only person who can create (or destroy) new criminal dossiers. The Public Prosecutor assigns access rights to roles (see below), and roles to users.

Note that in our approach, access to (and subsequent manipulation of) a dossier by users of the first type, users of the information management system, is only possible via agents. The owner of the criminal dossiers, the Public Prosecutor, also uses agents to create and delete dossiers. These agents are their only means of accessing the system—the 'single point of entry' for each user into the information management system.

C. Access Control

The semi-open nature of the application domain makes access control a particularly interesting challenge.

The solution proposed in this paper is based on a two-tier access model. The first level in the model is based on role based access control (RBAC) [10]. Each 'role', such as judge, lawyer, public prosecutor, or clerk, has certain rights with regard to the criminal dossier. This typically depends on specific security policies, for example, a clerk may only add information to a criminal dossier, not delete or modify anything, while a public prosecutor may change existing information in the criminal dossier and even create new criminal dossiers. Roles are also distinguished for all of the organizations associated with a dossier. The role based access control list is generated from the high level security policies. Each template includes the RBAC list associated with the offence for which the template has been designed, specifying which access rights are associated with which role.

Additionally, the second access level uses access control lists (ACL) [3] to determine the access rights of an individual user, within each of these organizations, *after* the RBAC list has granted access based on the user's role. Each individual dossier, i.e., an instantiation of a template, specifies, in its meta data, the names of specific users and their access rights: it specifies per user who may read or change, information in a dossier. For example, suppose that a specific RBAC assigns judges (those with the role of judge) read access to a specific type of dossier. The ACL, however, specifies which individual judges have permission to read a specific instance of a dossier. It may specify that only Judge A has read access. In this case, Judge B, although a judge, may not read the contents of this particular dossier instance. Thus in order to read fields in a digital dossier (via an agent) a user not only has to be assigned a specific role, he/she also has to be on the access control list of a specific dossier.

This distinction between roles and individual users is crucial in all dynamic environments. Security policies are based on this distinction. Security policies based on *roles* are relatively

[5]See [12] for a proposal to use automatic clustering techniques to construct (part of) this information automatically.

static (or at least 'long lived') and are typically globally valid (at all possible locations), while *individual*-based access control lists are typically dynamic (or 'short lived') and often apply to specific dossiers. Individual-based control lists change, for example, when a dossier is handed over to another clerk within the same organization, or to another judge. The combination of static and dynamic access control rules are designed to limit 'label creep' [9], (i.e., that access control regulations make *all* data more difficult to access, including the less sensitive information.) Section III-D describes how this two-tier access control mechanism is represented in dossier templates and instances. Note that the ACL is not mandatory—if not specified the globally defined RBAC lists are used to regulate access to dossiers.

Once roles have been distinguished and access rights assigned to users, users need to be able to be identified by the different systems within the different organizations. Authentication is needed. In practice, authentication is done by assigning each user a name and a password. Each username is assigned one or more roles. This information is stored in a central user data repository[6] which is unique for the specific environment, i.e., an ACCESS world and managed by the world administrator. This repository can only be accessed by users within this world who have been assigned the role and the right to maintain this information.

Users, however, do not interact directly with any one of the systems or repositories. A user interacts with an agent that represents that user, see Section IV-A below. Once authenticated, a user's agent is provided with a unique set of credentials with which his/her agent can be identified/authenticated.

Note that access control is not the only security issue in this environment. Other security features, such as secure communication between location, backup mechanisms, integrity checks and others, are discussed in detail in [13].

D. Data Types

As stated earlier, dossiers are structured according to templates. Which template is used depends on the offence (for criminal dossiers) and may also depend on other information (e.g. administrative dossiers, probation dossiers, etc). A template specifies the constraints and the (type of) information that a dossier must or may contain. The listing in Figure 3 shows a (simplified) example of a template in XML format. The template describes administrative dossiers that store information of a subject.

This template shows a role based access control list (RBAC). The RBAC list in a template states the permissions each of the roles have on dossiers that are based on this template. The RBAC list in the template in Figure 3 states that (all) mayors have the right to both read and write entries in dossier instances (but not in dossier templates) and that all mayors furthermore have the right to change the (optional) ACL of a dossier instance. Administrative clerks may both read and write administrative dossiers based on this template and judges are only granted read access.

[6]A data repository is typically a database running on a location.

```
<Template>
<Meta>
<Name>AdminInfo</Name>
<ID type="Value" mandatory="true"
    content="Integer" />
<RBAC>
    Mayor:R-W-ACL,
    AdminClerk:R-W,
    Judge:R
</RBAC>
<ACL type="Value" mandatory="false"
     content="String List"/>
</Meta>
<Fields>
<Field name="Name" mandatory="true"
     type="Value" content="String"/>
<Field name="Title" mandatory="false"
     type="Value" content="String"/>
<Field name="SocialNum" mandatory=true"
     type="link" content="SocNum"/>
</Fields>
</Template>
```

Fig. 3. An example of a dossier template containing administrative information

The template also names the fields in a dossier instance, including their format, and specifies whether a field is mandatory or optional. In this listing, "Name" and "SocialNum" (social security number) are mandatory, "Title" is optional. Name and Title are String values, SocialNum is a link to a dossier. The link must in this case, always point to a dossier of type "SocNum".

Depending on their assigned roles and permissions, users may be able to modify a dossier instance, however they are never allowed to modify the template. The RBAC list inside a dossier can only be modified by the world administrator, if permission is given by a public prosecutor. Note that the RBAC list is mostly static and will typically not change often.

The following listing shows an example of an instance of part of the administrative dossier, based on the template "AdminInfo" depicted in Figure 3. Consistency of required fields and types are checked automatically. More sophisticated consistency checks (spanning multiple dossiers) are performed by specialized agents, see Section IV-A. As required by the template, string values for the mandatory fields "Name" and "SocialNum" are specified.

The mandatory social security number field "SocialNum" contains a link (format `[refId]@[location]`) pointing to dossier ID 12432, located at location "SocNumRepos". The link points to a dossier possibly at another location. The precise content of this dossier is beyond the scope of this paper. In the AgentScape implementation of the ACCESS prototype, different locations store different kinds of information. Links point to different types of documents, e.g. pdf files (such as the original arrest warrant), movie files, dna profiles and other document types, stored locally or remotely.

The administrative dossier has the RBAC list inherited from the administrative dossier template. Note that the RBAC

```
<Dossier>
<Meta>
<Template>AdminInfo</Template>
<ID value="123876"/>
<RBAC> <!--Inherited from template-->
     Mayor:R-W-ACL,
     AdminClerk:R-W,
     Judge:R
</RBAC>
<ACL> Judge:Judy:R <!-- only Judge Judy
   is allowed to read this dossier.
   Access rights for mayors and
   administrative clerks are still
   determined by the RBAC list. -->
</ACL>
</Meta>
<Fields>
<Field name="Name" value="George"/>
<Field name="Title" value="Dr"/>
<Field name="SocialNum"
      value="12432@SocNumRepos"/>
</Fields>
</Dossier>
```

Fig. 4. An example of an (administrative) dossier instance –based on the template in Figure 3– containing administrative information

inside a dossier instance is a local cache of the template RBAC and is only used in case of malfunctions (e.g. when the network is down). But whereas the RBAC specifies the permissions for all dossiers based on this template, the dossier ACL specifies the permissions for this specific dossier, thereby refining the RBAC list. For the dossier instance in Figure 4 the ACL refines the access of judges: According to the RBAC list, of the template in Figure 3, all judges are allowed to read administrative dossiers of this type. The ACL of the administrative dossier instance refines the RBAC and restricts the read access to only one judge,the judge named “Judy”. The ACL does not specify any information on users with role “Mayor” or “AdminClerk”. Thus users that have these roles are not further restrained in their access rights. In summary:

- The dossier ACL is optional. If the ACL is not specified in the dossier, then the RBAC list in a document’s template determines the access rights assigned to users on the basis of the roles they have been assigned.
- The dossier ACL is more restrictive than the template RBAC. If a user’s role is not assigned specific rights in a document template’s RBAC list, the user cannot be assigned these rights in the dossier’s ACL. If this error occurs, it is noted, and access denied.
- If the dossier ACL contains a rule for role R, then users with role R will only be granted access if he/she is mentioned in this rule. (If the dossier ACL does not contain a rule for role R, then users with role R automatically gain all rights from the template’s RBAC list.)

Note once more that a dossier ACL is a refinement of a template’s RBAC list. A dossier ACL can be made whenever appropriate.

E. Data Repositories

Within the ACCESS prototype application, information is stored at different locations. There is a central repository for user information and templates (managed by the world administrator), and one or more central repository for the root files of individual distributed digital criminal dossiers. These criminal dossiers contain references to information stored decentrally in repositories managed by repository owners responsible for the contents.

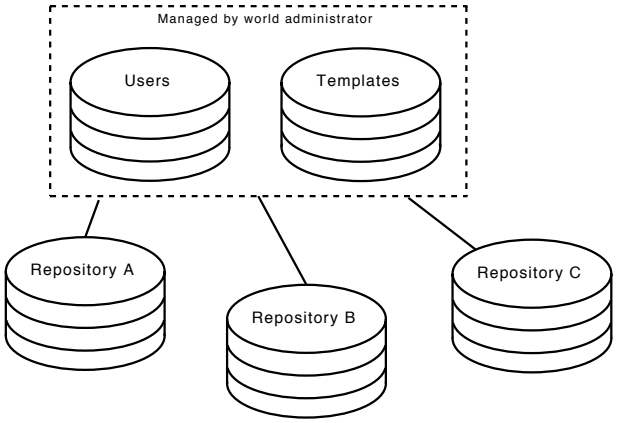

Fig. 5. ACCESS world organization

Users of the information management system at one location can include references to dossiers at other locations in their own dossiers, allowing information at the different locations to be managed by independent owners. Consequently, all updates to a dossier X at location A will automatically be part of all dossiers referring to dossier X from all other locations.

The ACCESS prototype is implemented as an AgentScape world. The world has its own lookup service, with the names of locations and their addresses. Access to services or repositories, outside the AgentScape world is only possible via a WS-Gateway [8]. This issue is not further addressed in this paper. The complete ACCESS world is illustrated in Figure 5.

IV. Prototype Implementation

The ACCESS prototype is agent-based and runs on AgentScape. The implementation is not AgentScape specific and can, with some minor changes, be implemented in other agent platforms. The prototype supports communication between the agents and various repositories, and implements the security policies described above. The architecture of the ACCESS prototype is described below.

A. Agents

The ACCESS architecture distinguishes a number of different types of agents, of which the first two form an intrinsic part of the system.

1) *Repository agents*
   Repository agents regulate access to documents in repositories. Each repository agent is responsible for the documents in one repository. Repository agents regulate access to documents on the basis of agent credentials,

the policies specified in the RBAC lists of document templates, and the ACL of specific dossiers. Note that each repository agent has access to the templates and user information to check if a particular user agent (actually, the user behind the agent) is allowed to perform certain operations.

2) *User agents*
User agents represent the users of the information management systems. These agents interact with the repository agents to acquire access to a repository, i.e., each individual user agent sends the credentials of the user they represent, their dossier ID and the actions the agent wants to perform to the relevant repository agent. Once authorized, a user agent can then perform the action requested e.g. read, write (part of) the dossier managed by the repository agent.
3) *Functional agents*
Functional agents add new functionality to the ACCESS prototype. They can be inserted at runtime. Typical examples include: (i) *Consistency agents* are responsible for consistency checks across different repositories, between different dossiers. The consistency agent at the Public Prosecution is responsible for consistency of the information across repositories, as specified in the crime related template. (ii) *Completeness agents* are responsible for the completeness of the information in criminal dossiers. (iii) *Timeliness guarding agents* monitor time constraints (e.g. statute of limitations) with and across dossiers and take action when necessary.
Other dedicated agents can be deployed for specific tasks, guarding other specific types of constraints and regulations.

Depending on the specific requirements of a domain and the underlying infrastructure, different choices can be made with respect to the agents deployed. For example, repository agents may be responsible for flagging constraint violations (e.g. age < 18) or dedicated agents may be deployed for this purpose. The ACCESS prototype currently supports the first two types of agents and examples of agents of the other types.

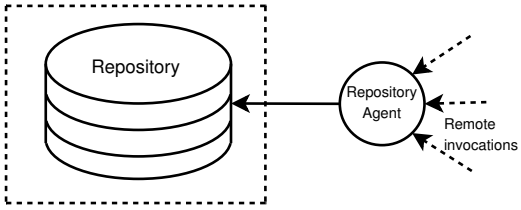

Fig. 6. Repository access by agent

AgentScape is responsible for running the agents and for the communication between agents. An agent's request (for access to a dossier at a different location, in fact, a remote method invocation) is always forwarded to the relevant repository agent, possibly at a different location, on a different host. Figure 6 shows how a repository agent provides other agents access to a repository.

One of the main features of the ACCESS prototype is that it supports incremental design. Agents can be added to the system as needed, adding new functionality, in fact, at runtime.

B. Distribution

One of the main concepts in the ACCESS prototype is that each individual organization manages its own dossiers in its own repositories locally, within an AgentScape location. These locations are (often physically) distributed. A lookup service provides the names and addresses of the repositories within a world. This lookup service contains all the information for a single ACCESS application, i.e., for one Public Prosecution office[7].

C. System Architecture

To access dossiers in the ACCESS world, a user of the information management system interacts with his/her user agent (an AgentScape agent). All communication with other agents is performed through this agent. The user's agent uses the user's credentials (derived from username and password), to acquire access to the different repositories.

In the current implementation, the user is presented with a graphical user interface (GUI) that communicates with the data repositories through the user agent. While the actual GUI window is running outside of AgentScape, all communication and authentication checks run inside the AgentScape platform in the ACCESS world. For an overview see Figure 7.

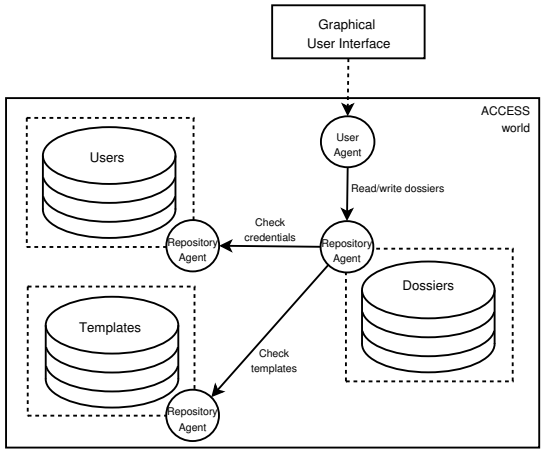

Fig. 7. Architecture of the ACCESS prototype

In theory, all users of the information management system could use their user agent to gain access to all dossiers in all repositories. In practice, however, whether they are actually allowed to view or edit the dossiers, depends on the agent's credentials, the dossier template's RBAC list and the dossier's ACL.

[7]When creating a world, the lookup service is the first thing to be started. Then at the central location, i.e., the Public Prosecution, the user and template repositories are created. These repositories are managed by the world administrator: the user that created the AgentScape world.

V. Discussion & Conclusions

Information management in a distributed environment, such as the Public Prosecution and the organizations with which it cooperates, is an inherently difficult task. The use of multi-agent technology has clear advantages, as this paper demonstrates.

The basic system provides distributed and secure access to criminal dossiers for all users inside the ACCESS prototype. All information in the distributed digital criminal dossiers is maintained by local organizations, such as Municipalities, the Public Prosecution and others, providing local control and global access to information. In addition, information is always kept as up-to-date as possible, e.g. if a defendant moves, this is immediately reflected in the criminal dossier, once a Municipal updates this information in the (linked) administrative dossier.

Additional functionality can be added gradually in a modular fashion by means of new dedicated agents. Local organizations can add their own specialized agents and more globally useful agents can be distributed to all organizations. Again allowing maximum local control and global cooperation.

From a security point of view, as long as the computer system of the Public Prosecution has not been compromised, the digital criminal dossiers are not at risk. The computer systems of the Public Prosecution, however, need to be trusted completely. The lookup service, dossier templates and global authentication mechanism are all hosted by the Public Prosecution. If other systems are compromised the digital criminal dossiers are not necessarily effected. Automatic consistency checking, performed each time part of a dossier is altered, detects modifications. If these modifications are unwarranted and detected by both the dedicated agent and the user, once informed, the original data can be restored from a secure backup service.

The current prototype implementation in AgentScape supports several hundred dossier instances distributed over half a dozen locations. Future research will examine how well the implementation scales in a large scale distributed environment (10,000+ dossier instances on 50+ hosts).

The complete ACCESS prototype including a complete distribution of AgentScape can be found at http://www.agentscape.org.

Acknowledgments

This research is supported by the NLnet Foundation, http://www.nlnet.nl, and is conducted as part of the Agent-based Criminal Court Electronic Support Systems (ACCESS) project, http://www.iids.org/access, initiated by the VU University Amsterdam together with the Courts of Amsterdam and Rotterdam, and financed by the Dutch Court for Jurisdiction (Dutch: Raad voor de Rechtspraak) and by the NWO TOKEN program.

References

[1] J. Dale. *A Mobile Agent Architecture for Distributed Information Management*. PhD thesis, University of Southampton, 1997.

[2] C. A. Ellis and G. J. Nutt. Office Information Systems and Computer Science. *ACM Comput. Surv.*, 12(1):27–60, 1980.

[3] C. Kaufman, R. Perlman, and M. Speciner. *Network Security, PRIVATE Communication in a PUBLIC World*. Prentice Hall, 2nd edition, 2002.

[4] M. Luck, P. McBurney, and C. Preist. *Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing)*. AgentLink, 2003.

[5] D. G. A. Mobach. *Agent-Based Mediated Service Negotiation*. PhD thesis, Computer Science Department, Vrije Universiteit Amsterdam, May 2007.

[6] L. Moreau, N. Gibbins, D. DeRoure, S. El-Beltagy, W. Hall, G. Hughes, D. Joyce, S. Kim, D. Michaelides, D. Millard, et al. SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In *Proceedings of The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pages 369–388, 2000.

[7] B. J. Overeinder and F. M. T. Brazier. Scalable middleware environment for agent-based internet applications. In *Proceedings of the Workshop on State-of-the-Art in Scientific Computing (PARA'04)*, volume 3732 of *Lecture Notes in Computer Science*, pages 675–679, Copenhagen, Denmark, June 2004. Springer.

[8] B. J. Overeinder, P. D. Verkaik, and F. M. T. Brazier. Web service access management for integration with agent systems. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC)*. ACM, March 2008.

[9] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. *IEEE Journal on selected areas in communications*, 21(1), 2003.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996.

[11] M. Warnier, F. M. T. Brazier, M. Apistola, and A. Oskamp. Distributed Digital Data: Keeping files consistent, timely and small. In *Proceedings of the eGovernment Interoperability Campus 2007 Conference (eGovINTEROP'07)*, 2007. to appear.

[12] M. Warnier, F. M. T. Brazier, M. Apistola, and A. Oskamp. Towards automatic identification of completeness and consistency in digital dossiers. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law (ICAIL'07)*, pages 177–182. ACM Press, 2007.

[13] M. Warnier, F. M. T. Brazier, and A. Oskamp. Security of distributed digital criminal dossiers. *Journal of Software*, 3(3):21–29, mar 2008.

[14] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 379–390. ACM Press New York, NY, USA, 2004.