ARCHITECTURE FOR DISTRIBUTED DATA COLLECTION FOR MANAGEMENT OF CRISIS SITUATIONS USING TRUSTED AGENT EXECUTION

Ladislav HLUCHÝ, Zoltán BALOGH, Emil GATIAL
Institute of Informatics, Slovak Academy of Sciences/Department of Parallel and Distributed Computing,
Bratislava, Slovak Republic, tel.: +421 2 5477 1004, e-mail: ladislav.hluchy@savba.sk

ABSTRACT

This article proposes a distributed architecture designed for management of crisis situations where multiple actors are involved from various organizations with different competences and communicating over IP-based networks including wireless devices. In such settings requirements exist for secure communication and trusted collection of data from various sources. The primary role of agents in the proposed architecture is coordinated collection of information. In respect to requirements the overall agent infrastructure must be a secure, robust and fail resistant system. The required level of trust for agents is based on a special hardware module which provides trusted computing functionality. In the article we describe such architecture in terms of detailed requirements, design and decomposition to subsystems. We also provide a sample use case scenario inspired by concrete crisis situation. The architecture described herein is being used within the scope of an EU integrated project called Secricom; therefore we briefly describe the integration points with other systems involved in the project. We conclude with current state of the architecture implementation and with further plans concerning the development of the described architecture.

Keywords: *distributed architecture, agents, trusted code, process execution*

1. INTRODUCTION

One of the challenging demands of the communication infrastructures for nowadays crisis management is to add new smart functions to existing services which would make the communication more effective and helpful for users. The aim is to provide smart functions by distributed IT systems which should provide a secure distributed paradigm to achieve confidentiality and access to resources. Such infrastructure should further provide a smart negotiating system for parameterization and independent handling of access requests to achieve rapid reaction. By fulfilling the above stated goals a pervasive and trusted communication infrastructure satisfying the requirements of crisis management authorities and ready for immediate application could be introduced. More concretely in crisis situations requirements exist to collect information from legacy systems of various organizations and from human operators in order to semi-automatically manage the crisis mitigation process or to enact decisions at various management levels. This collection of information must be enacted in a secure manner while ensuring trust between both parties – information consumers and information providers. Many actors participate in a crisis situation and the competences between all parties are explicitly defined in a crisis mitigation plan. Gathering of information is enacted either from legacy systems or from human end-users through mobile devices by guided dialog. Herein we present the requirements analysis, design, and system decomposition of a distributed architecture which would fulfill all the goals set above. Further we suppose that the communication infrastructure is IP-based.

We decided to design and implement such architecture using agent paradigm. The distributed agent-based infrastructure is designed as a collection of software services with agent-like features (such as code mobility) which would execute in a secure and trusted manner. Agent technology was selected due to the ability to fulfill such requirements through support of mobile and dynamically deployable executable code. Other advantages of agent-based systems are that they can help overcoming temporal or longer term communication network failures, save network bandwidth by being executed remotely and deliver only the execution results, provide means to execute code on remote host platforms in a trusted and secure manner or deploy code on host platforms on demand. The role of agents in the architecture is primarily coordinated collection of information. Gathering of information is enacted either from legacy systems or from human end-users through mobile devices by guided dialog. In respect to requirements the overall agent infrastructure must be a secure, robust and fail resistant system. Because validity and authenticity of gathered information is a key factor for decision making in crisis management, trust must be set between agents and third party information systems. Also, agents must trust the host platform providers - remote sites which provide the computational environment for agents. The required level of trust for agents is based on a special hardware module which provides trusted computing functionality.

This article is written as follows: The next section deals with analysis of requirements and security considerations of the proposed architecture. The third section describes the proposed architecture with decomposition to subsystems and envisaged core agents. The fourths section describes a sample scenario which is used as a reference scenario for the infrastructure implementation. The architecture described herein is being integrated within an EU integrated project called Secricom. Therefore the fifth section introduces the Secricom project and the integration points of the proposed architecture with other systems involved in the project such as the PTT (Push To Talk) system, SDM (Secure Docking Module) or MBR (Multi Barer Router).

The last section concludes the article and presents our current achievements and plans concerning the implementation of the proposed architecture.

2. REQUIREMENTS AND SECURITY CONSIDERATIONS

In order to define concrete security requirements for our architecture we sketch the basic infrastructure in which agents will operate (Fig. 1):

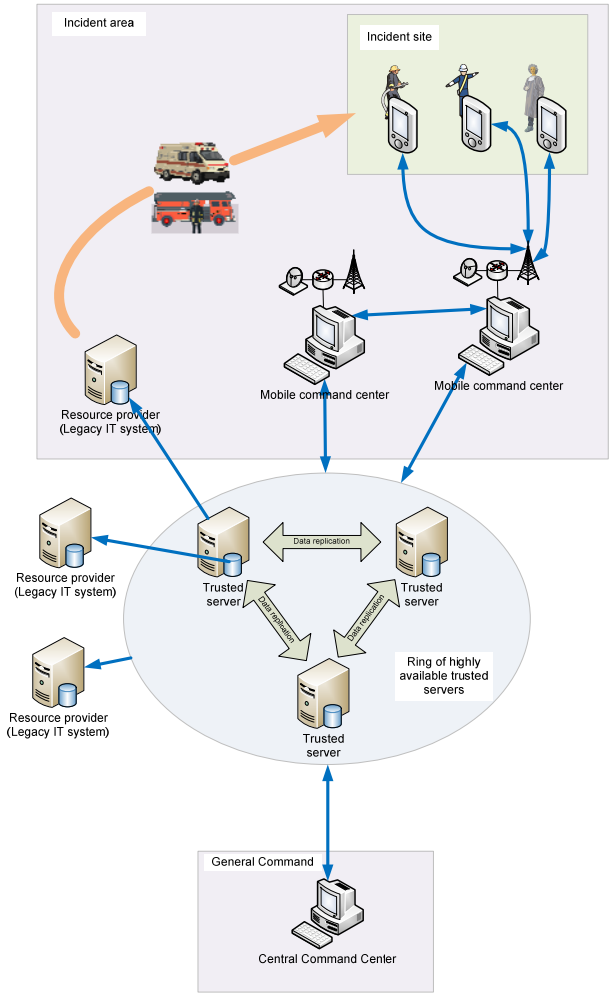

Fig. 1 Infrastructure and Host Platform Providers in the Distributed Agent-based Architecture

The home platform for agents is a network of Trusted Servers (TS). According to [1] the platform from which an agent originates is referred to as the home platform, and normally it is the most trusted environment for an agent. This is also true for our agents – the TS network is a managed set of systems with defined security policies and possibly managed by a central authority. From here agents are delegated to host platforms to gather data and information. Agents are mainly executed on remote sites which provide the computational environment in which agents operate. We will refer to these sites as to host platforms (or agent platforms).

In general, any party which wishes to join the implemented architecture and to provide information from their legacy systems or users must introduce a host platform for agents. We refer to such parties as Host Platform Providers (HPP). From end-user requirements, the following HPPs were identified (Fig. 1): Resource Providers – hospitals, fire brigade, police, warehouses or any other entities which can play a role in mitigation of crisis situation; Command Centers – mobile (nomadic) centers which coordinate the incident site locally; and General Command Center and Operators – usually located at one place or at least closely interconnected.

The features of agents encompass several chosen attributes: code mobility (without execution state) – the ability to move code to different platforms and execute there; within the project we do not plan to support execution state mobility (as there is no such requirement); autonomy – the ability to autonomously deliver gathered data to one or several optional destinations; reactivity – in some cases agents will perceive the context in which they operate and react to it appropriately (e.g. agents can monitor availability of resource and notify the requestor).

Since agents collect information which is often of high sensitivity, confidentiality and security, while at the same time requirements for action or decision traceability exist, agents must be provided with a secure, trusted and attested execution environment. In the following we identify main agent-related security threats. A detailed explanation of generic mobile agent security aspects is discussed in [1]. Generally, four threat categories are identified: an agent platform attacking an agent, an agent attacking an agent platform, an agent attacking another agent on an agent platform and other entities attacking the agent system. The last category covers the cases of an agent attacking an agent on another agent platform, and of an agent platform attacking another platform, since these attacks are primarily focused on the communication capability of the platform to exploit potential vulnerabilities. The last category also includes more conventional attacks against the underlying operating system of the agent platform.

The host platform attacking agent

The main threat for agents in foreign execution environment of host platforms is the “malicious host problem”. This is one of the main problems in the class of “an agent platform attacking an agent“. Simple explanation of the “malicious host problem” is provided in [2]: “Once an agent has arrived at a host, little can be done to stop the host from treating the agent as it likes”. Therefore, the main requirements from the agent-side are laid out in respect to the “malicious host problem”. Concrete security requirements of agents in respect to the host platform are as follows: isolated execution environment for agent execution – not only virtual isolated execution environment but dedicated isolated hardware preferred; means to attest the platform required in order to detect whether the host platform is in trusted state; and protected storage for credential data (such as PKI’s secret key).

Agent attacking the host platform

There are also threats stemming from an agent attacking an agent host platform. Therefore reversely a host platform has also requirements in respect to agents. These requirements are more evident when provided in context of HPPs security requirements:

1. HPPs do not want to install and execute any external application on their systems in line with their strategic legacy applications.

2. HPPs prefer to have a dedicated and isolated system for agent system which would connect to their legacy system in a secure predefined way.

3. HPPs want to be able to control what (data), when and by who (traceability) is provided to agents.

4. HPPs want to be able to configure the set of applications executable on their side. Agents must be therefore audited and verified and therefore mediate trust to executable agent code.

The agent platform has the following security requirements in respect to agents: isolated execution environment for agent execution - agents must be executed in isolated environment (isolated hardware preferred), so an agent can not harm legacy systems; means to monitor and trace agents activity; and means to configure the set of agents executable on the host platform. In order to track agents, any agent in the platform must be cryptographically signed. Only agents signed with trusted authority and assigned to selected category will be trusted by a host system. Agents need to send signed messages to Trusted Servers.

An agent attacking another agent

It is required that any agent which will be used in the system will need to be audited and certified by a central authority. In turn every host platform will be configured to execute only agents which are certified. These two security policies should ensure that malicious agents will not be deployed into the infrastructure. Only a breach of the set security policies might lead to potential agent-to-agent security risk.

Moreover, each agent should be executed in a relatively isolated virtual environment with limited access to data of other parallel executed agents on the same host platform.

Other entities attacking the agent system

Agents will also connect to legacy systems (third party software). Therefore there is a risk of an agent being attacked by a legacy system but also vice versa – the risk of attacking legacy system by an agent also exists. The host platforms will need to provide a kind of connection to legacy systems. We explicitly presume that this will be a network connection. On any network connection there is an eavesdropping risk. Therefore another requirement which arises from agents to the host platform is secure protected connection to legacy systems. Physical security of network connection can be achieved either by direct cable connection of the host platform with legacy system or by managed network security (managed switch with well defined security policies). The data transport security will be achieved primarily through encryption.

3. ARCHITECTURE

In this section we present a distributed architecture designed for management of crisis situations were multiple actors are involved from various organizations with different competences and communicating over IP-based networks including wireless.

The architecture (Fig. 2) is designed for mobile services with agent-like features (mobility, pro-activity) which would execute on secure devices. In general it consists of interconnected trusted (TS) and untrusted servers (US). TS carry out the following tasks: registry of services, users and modules, public encryption keys, the agent base (base of mobile code) or generic security politics. Each agent has features and “abilities”, which are used for the enactment of certain processes.

The enactment of processes is inspired by the domain of management of crisis situations in which collection of information from multiple systems is required. The whole process starts with the specification of a problem in the form of dialog. Further, an agent specifies the most serious problems which were rendered by the crisis situation. Based on the type of crisis situation and on the region where the crisis has occurred appropriate actions

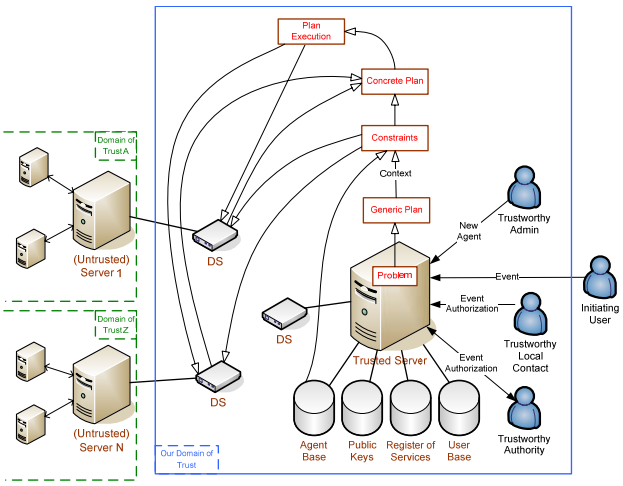

Fig. 2 Distributed architecture designed for management of crisis situations

are initiated for each crisis situation type. The system will semi-automatically generate plausible generic plans of possible solutions (mitigation plans) of identified problems. In the next step the specification of context will be enacted in order to be able to generate the constraints of the crisis situation. Relevant resource providers will be identified in the central database based on constraints generated in the previous step. Agents which are able to query selected servers will be selected from the agent base. Information about available capacities of resource providers will be retrieved and sent back to central trusted server base. The system will then generate a concrete plan

Table 1 The architecture subsystems

Subsystem	Basic description and functionality
Distributed Secure Agent Platform (DSAP)	The core agent platform. Will provide means for agent deployment, execution, migration and communication.
Process Management Subsystem (PMS)	Based on the plan collected from users it will generate a plan of activities. Executes the plan.
Agent Repository (AR)	Database of system users, agents and their certificates. Process of accreditation of agents.
Public Key Infrastructure (PKI)	Certification and verification of agents, users and resources.
Resource Inquire System (RIS)	Will provide information which system to query for specific information.

of crisis situation resolution based on the retrieved disposable resource capacities. The last step is execution of the plan prepared for the concrete crisis situation.

In order to fulfil the architectural requirements set the infrastructure was decomposed into subsystems. These subsystems are related and will cooperate together through defined interfaces. The list of all subsystems is in Table 1.

The purpose of the Distributed Secure Agent Platform (DSAP) is to provide an execution environment for different types of agents. The main aim of Process Management Subsystem (PMS) is execution of processes and coordination of involved agents in the emerged crisis situation. The plan scenario for each type of crisis situation will need to be pre-prepared in the form of an abstract process. The exact execution of such plan in a concrete situation will depend on the context of the crisis situation. The agents available within the herein proposed infrastructure have to be stored on Trusted Servers, from which they can be requested for deployment on the side of HPPs – this functionality is encompassed within the Agent Registry (AR). Public Key Infrastructure (PKI) will allow certifying, and subsequently verifying, all the objects deployed in the infrastructure. Agents will require having information about the information sources which can be queried in order to retrieve information about resource availabilities. The Resource Inquire System (RIS) will provide an interface which will provide such capability.

In addition there is a set of core agents which are required in order to ensure functionalities of the architecture. List of agents including their brief functionality description is given in Table 2.

Table 2 Core agents used in the architecture

Agent	Functionality
Information Delivery Agents (IDA)	IDA agents will need to connect to legacy information systems of third parties to retrieve information about available resource capacities.
User Communication Agent (UCA)	Will communicate with users in a form of guided dialog through electronic device. Will include authentication and interface to authorization of the user.
IP Agent (IPA)	An agent able to configure IP devices such as routers.

4. AGENT PLATFORM DESIGN AND IMPLEMENTATION

The following paragraph explains the agent code migration and agent lifecycle, while the next one introduces the security mechanisms of this agent platform.

Since distributed algorithms were elaborated by computer science, the designed platform focuses merely on remote code execution with emphasis on establishing trust among multiple parties. Moreover, the agents in our scenario are designed to access resources in order to help a person make a decision but not make a decision autonomously. This does not express that the agents are meant to execute only simple code accessing resources (i.e. querying DB systems or temperature sensors). The selection of resources or resource providers can be optimized within the agent's code, while a responsible person has a full control upon agent's activities. Such optimizations of resource selection are beyond the focus of the agents described in this paper. According to [1] the platform from which an agent originates is referred to as the home platform and the platform where an agent's execution takes place is called host platform (HP). In general, any party which wishes to join the implemented architecture and to provide information from its legacy systems or users must introduce a host platform for agents. We refer to such parties as host platform providers. One of the main focuses of designed agent platform is securing and migration the agent from a home platform to a HP and their mutual communication.

DSAP is built upon the Java and Jini framework making use of the service proxy's remote method call feature for modeling a HP. The HP, issuing an agent, locates possible service providers by querying LS(s) (either using known LS location or finding all possible LS by multicast) for specific DSAP service proxy. The DSAP services registered in LS usually hold additional service attributes (service metadata) like service location and service capabilities. The DSAP client discovers the right DSAP service implementation by matching the required service attributes with DSAP service attributes stored at LS and makes use of a DSAP proxy instance to access the HP.

The lifecycle of agent brings the following issues:

- Finding suitable DSAP service in order to deploy an agent with specific goal: In order to successfully accomplish the goal of an agent, the suitable DSAP service has to be identified. One way how to successfully identify the suitable DSAP service deployment is to discover every DSAP service among LS and to filter out the most appropriate service by service attributes' matchmaking. This process includes comparison of DSAP service attributes with goal specific attributes of an agent. Such process should be done every time the agent has to be deployed, which leads to cumbersome and time consuming operations. Another way is to implement special kind of service whose special goal is to register event listener notifying a service attribute change among discovered available DSAP services. This kind of service should periodically discover new DSAP services or discard no longer available services. RIS service should implement methods searching for the most appropriate DSAP service according to specific criteria, for example to find the DSAP service deployed in the vicinity to given geographical location, etc.
- Deployment of an agent in short- or long-term manner using DSAP client: The short-term agent deployment occurs in on-demand information acquisition where an agent responds almost immediately to a DSAP client with return messages. Typically, the agent is terminated and cleared from HP or suspended to be used later on. In case of the long-term deployment, the agent may reside on the HP and respond to a DSAP client continuously or may check the HP environment by invoking events on DSAP client.

In Fig. 3, the process of discovery, join and agent communication is shown within the scope of the DSAP services. Here, HP discovers LS and joins the registrar

object. When a DSAP client (residing in a home environment) wants to deploy an agent it searches the LS for the DSAP services conforming the attributes of an agent. Commonly, the attributes describe a location of a DSAP service, the capabilities reflecting the actions that can be made by an agent using specific Java libraries and an organizational unit that an agent needs to cooperate with. This attribute concept can be easily extended using other attribute types. Once an appropriate DSAP service was found an agent is uploaded and run in HP environment under the control of the DSAP service. Each agent is assigned with globally unique identifier (GUID) before the process of deployment will take place in order a client can communicate with specific agent. A DSAP client is able to send messages to agent using GUID and to receive immediate response, or a DSAP client is notified by firing the events processed by event handler of DSAP client.

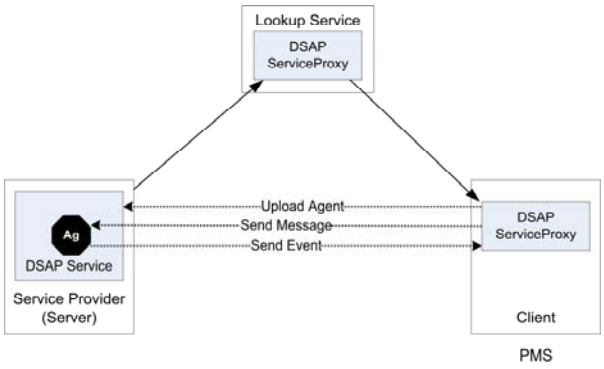

Fig. 3 The concept of DSAP based on the Jini framework

Security mechanism for agent platform is incorporated into the DSAP service using PKI standards with respect to security requirements on the agent platform. Moreover, the DSAP relies on a Secure Docking Module (SDM) storage holding private keys and a Trusted Docking Station (TDS) quoting a trusted platform state. The HP, deployed within the TDS, is measured by auditing the BIOS and operating system booting sequence measurements evaluated as SHA-1 hash values and stored in a Trusted Platform Module (TPM). The SDM only releases private keys if the host platform adheres to a configuration (trusted state) that enforces a key protection policy. The root of trust is established between the agents' home platform and HP by audited agent code before its usage will take place. The audit process must ensure that the agent does only what its creator states it should do, and that it does not contain any malicious code, which may jeopardize the integrity of the HP. Establishing the trust between an agent and a HP is depicted in Fig. 4.

Agent repository (AR) holds the set of certified agent Java classes or .jar files. The code of agents may vary from executing simple DB query to complex management of HP resources. It is up to agent designer to implement an agent's functionality, but with respect to the fact that the code must be audited and certified whether by the HP provider or by trusted third-party authority. Based on the code certification the HP provider can trust the code running his or her HP.

When PMS decides to issue an agent, it queries AR to obtain the classes implementing the agent. Here, PMS is able to verify the certificate of agent classes. Next, an instance of agent object is created by PMS where the agent attributes are set. The agent object and its classes are encrypted using AES key secured by $TDS_1PubK_{E/D}$ public key (referred to as key wrapping) of HP.

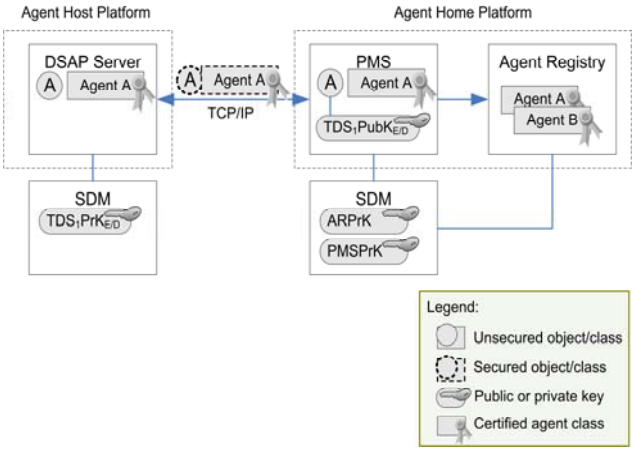

Fig. 4 The scheme of DSAP concept to establish secure and trusted communication of agents

After the encrypted agent is moved on the HP, the DSAP service decrypts the AES key using $TDS_1PrK_{E/D}$ private key of the HP (received from SDM) and uses this key to decrypt an agent. The HPs usually provide access to some resources that a specific agent is able to process. Here, PMS is responsible for choosing the right type of agent and for setting it up to provide the required results. The results are encrypted using the same AES key and sent back to PMS.

5. AGENT COORDINATION

Since DSAP platform provides convenient mechanism for secured agent migration within trusted environment, the coordination of agents is managed by centralized PMS capable of tracking sequence of actions to achieve a specific goal. PMS holds ontological description of processes that can possibly occur when some external event is fired.

Formally, the process P means to execute any possible actions to achieve the process goal. The process state or context C denotes the set of resources available during a process execution. The process goal G_P is to execute every ending action in the specific process P. Action templates of the process $AT_P = \{at_{P1}, at_{P2},\ldots, at_{Pm}\}$ hold the set of action templates available for execution in the given process P. Action templates prescribe what action will be instantiated while fulfilling the preconditions of AT_P. Such precondition set $PC_{AT} = \{pc_{AT1}, pc_{AT2},\ldots, pc_{ATn}\}$ specifying the set of resources that must be available before the action will be executed. Here, each AT_P holds references to the role of responsible actor R_{AT}, which is mapped to specific actor (person or legacy resource responsible for process invocation) later in the deriving process of action instance. AT_P also describes the type of its effects $E_{AT} = \{e_{AT1}, e_{AT2},\ldots, e_{ATp}\}$ containing the set of activity output types. Each action derived from AT_P produces resources of E_{AT} (subset of Resource ontological

concept) that are stored in C. Agents AG = {ag_1, ag_2,…, ag_q} contain set of agents that are capable of solving a specific task. Action A_P = {a_{P1}, a_{P2},…, a_{Pr}} represents an instance of AT_P coupled with ag_i zero or more agents while resources PC_{AT} are available in C. Formally, the task of PMS is to initiate a process P and to manage execution of actions A_P prescribed as AT_i in order to achieve the goal G_P.

Initially, PMS searches for initial action templates in AT_P, where the precondition is not specified, thus PC_{AT} = {}, for each process P and therefore instances of such AT_P can be executed without any required resources. For every at_{Pi} where PC_{AT} = {} the starting action a_{Pi} is created. The instances of agents ag_j, described as E_{AT} in a_{Pi} are created and deployed in DSAP service in order to communicate with the specific actor matching the role R_{AT} for process P. Actually, the PMS will issue the starting agents which contact actors (persons or legacy resources) to initiate a process P. In case of communicating with a person, the UCA long-term agent is utilized to query the responsible person for initiating the process; in case of communicating with legacy system, the ICA long-term agent is sent to HP to check availability of specific resource. Starting agents usually sit in the HP till the required resource is made available to initiate process P and update the context C. Since multiple processes can be enacted simultaneously, the starting agent is redeployed right after the agent has started a new process.

If specific process P is initiated, PMS searches every action template in AT_P set, where the precondition set PC_{AT} contains a subset of required resources included in context C, formally expressed as selecting AT_P; $PC_{AT} \subseteq C$. For every found action template at_{Pi} the action a_{Pi} is created. The instances of agents ag_j, described as E_{AT} in a_{Pi} are created and deployed in DSAP service in order to communicate with the specific actor matching R_{AT} for process P. Here, agents can be issued as long-term usually for monitoring purposes or short-term requesting immediate response. Each agent a_i produces resources of type e_{ATi} that are included into context C. Each time the PMS changes the context C, a new set of AT_P is found according to selecting AT_P; $PC_{AT} \subseteq C$. The PMS finishes the process execution P if the G_P condition is matched.

6. SAMPLE SCENARIO

Herein we present a sample scenario in which coordinated information collection using agents takes place. The presented scenario is not a typical crisis scenario were emergency responders are involved but demonstrates all the important and useful abilities of agents in such distributed settings.

The schema in Fig. 5 depicts an imaginary epidemic crisis scenario: A country has a sudden rise in the number of people sick from an epidemic flu. There are many infected people and others are suspected to be sick soon. The organization responsible for mitigation of epidemic is UVZ. Personally a Chief Officer (CO) at UVZ is responsible for such situations. CO decides to set warning level to 5. As part of this warning level UVZ needs to make sure that there are sufficient supplies of vaccines in regional UVZ branches (RUVZ).

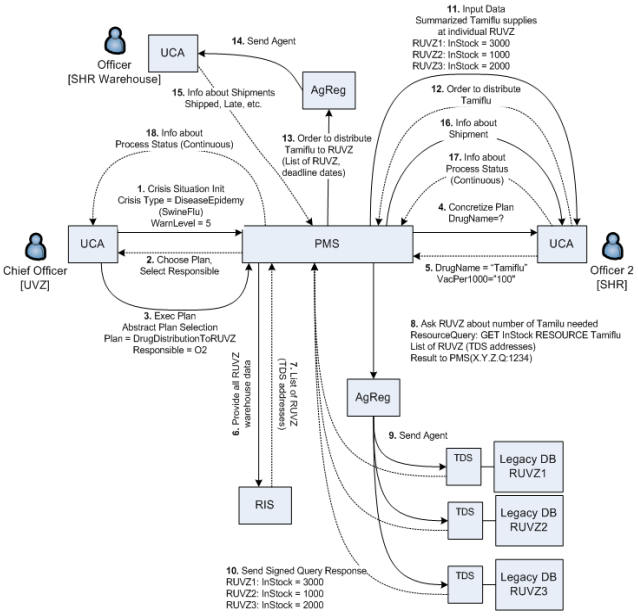

Fig. 5 Schema of a sample crisis scenario

Such information must be retrieved from legacy systems of each RUVZ. CO must delegate this information collection to an officer at another organization called SHR (O2). After the officer at SHR finds out about the supplies at individual RUVZ he needs to delegate the task of distributing additional sufficient amount of vaccines to an Officer at SHR Warehouse. The information about complement shipments of vaccines to RUVZ is sent by SHR Warehouse Officer directly to SHR Officer who redirects this information to CO at UVZ. Concrete steps of the scenario are:

1. CO initiates a new Crisis Situation in the UCA user interface, where CO opens the UCA and selects “Initiate Crisis Scenario” of type “DiseaseEpidemic” and sets “Level” to value 5.

2. PMS is informed about new crisis. PMS checks if the request is signed and whether CO is trusted and has rights to initiate the mitigation. After confirmation is sent back to CO’s UCA, possible (pre-prepared) mitigation plans and list of qualified responsible persons (officers O1-O3) is generated.

3. CO selects the right mitigation plan and decides to ask O2 to supervise this process. In this stage the process is in an abstract format, i.e. details are not concretized.

4. PMS informs O2’s UCA that he is responsible for supervising the process. He is also asked to concretize the process, in this case by specifying the “DrugName” and “VacPer1000” properties.

5. O2 accepts to supervise the process and specifies the required properties.

6. PMS is informed about “DrugName”. PMS needs to find out who is able to supply the “DrugName” resource. PMS contacts RIS with a query to provide all suppliers of “DrugName” resource.

7. RIS replies with a list of RUVZ.

8. PMS now can query all RUVZ for availability of the resource called “Tamiflu”. PMS formulates the query and sends List of RUVZ and where to send the result. Query is send to AR (AgentRepository). Deadline for result delivery is specified as well.

9. AR must select an appropriate agent (of IDA type) for each RUVZ because each RUVZ might have different legacy systems. AR sends out agents to collect relevant data. Agents are deployed to each resource provider (RUVZ in this case).

10. Agents send back their response to query.

11. Data is collected by PMS and after the deadline it is sent in consolidated form to O2. O2 reviews the data where he can see current stock amounts at each RUVZ warehouse.

12. O2 creates order to distribute missing drugs to RUVZ. This will be a request for resources to be ordered/delivered. The request is sent through PMS to Officer at SHR Warehouse. O2 is able to specify that each region should be equipped with 100 vaccines per 1000 people. Based on information about population of regions the vaccine numbers are computed and order is created.

13. PMS requests AR to send OrderAgents to the officer at SHR warehouse.

14. ShipmentAgent is sent out to each RUVZ.

15. ShipmentAgent informs PMS about the status of deliveries.

16. PMS informs O2 about status of deliveries.

17. O2 informs PMS about process status.

18. PMS informs CO about process status.

Please note that in this scenario communication between users is proposed to be done using UCA – User Communication Agents. UCA is able to communicate with users either through computer or through a mobile device. UCA collects information from a user through a sequence of simple forms. UCA summarizes the form results and sends it to PMS for further processing. The IDA – Information Delivery Agent is used for retrieving information from legacy systems. There might be different types of IDA suitable for different legacy systems of various resource providers. There are also other agents used in the scenario such as OrderAgent or ShipmentAgent – they are specific purpose agents. The only agent not mentioned in this scenario is the IP Agent – this agent is intended to configure routers or other active configurable IP devices. IPA can semi-automatically configure the network according to current need of the crisis responders. For example in our sample scenario we could use IPA to prioritise the communication between the officers at UVZ and SHR.

7. INTEGRATION WITH OTHER SYSTEMS

The architecture described herein is being used also whithin an EU integrated project called Secricom [9]. The implementation of the architecture in the project is called Secure Agent Infrastructure (SAI). SAI solves timely delivery of relevant information, obtains the information about available resources (material or human) and helps the authorities manage the distribution of such resources. SAI also communicates with legacy information systems operated by agencies and institutions involved in the crisis resolution. There are several systems to which SAI gets connected. Concretely we describe integration with SDM - Secure Docking Module, PTT - Push To Talk system and MBR - Multi Barer Router systems.

In order to overcome the threats described in section II, agents require safe secured place to store cryptographic credentials (PKI secret keys) and provide interfaces to retrieve these keys, ways to attested platform (execute on a host platform which is in a trusted state) and provide interface to safely communicate with legacy systems. All these functionalities are provided by a hardware module called Secure Docking Module (SDM) [5, 6]: SDM is a key storage device with local attestation and verification capabilities. SDM establishes trust on the host platform where agents are being executed – called Trusted Docking Station (TDS). A trusted state is a specific software configuration. This software configuration is measured by using a Trusted Platform Module (TPM). A TPM is a special security chip, which amongst other functionalities, provides the protected capability of measuring the software configuration of its host device. A TPM must be present in the TDS. The combination of a SDM and a TDS is called a Secure Docking Station (SDS) as shown in Fig. 6:

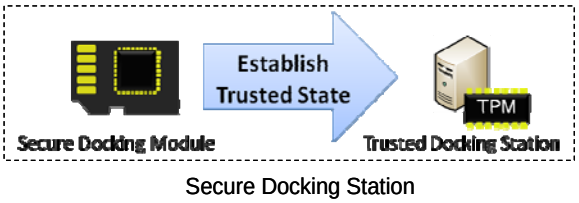

Fig. 6 Schema of how SDM, TPM, TDS and SDS relate

SAI uses SDS deployed in a physical proximity of the legacy information system, preferably in the same room, and acts as a secured and trusted extension of the Secricom infrastructure. SAI executed in SDS eliminates the exposure of the legacy IS to the outside world and allows the operator of the legacy IS to have increased trust in the information consuming party. SAI can process the information received from the legacy IS while conserving the network bandwidth, limiting possible exposure of sensitive data – sending back the results only and continuing data processing even if the connection to the outside world is intermittent. More information about these technologies can be found in [5, 6].

Secricom PTT (Push To Talk) is a client-server communication system using IP protocol and is developed by a Slovak company Ardaco [7]. PTT optimizes and protects the way teams of people communicate without being concerned about misuse of information. Regardless of communication endpoint (mobile, laptop or handheld) the communication is secure and safe. SAI connects to PTT servers in order to communicate with users. Concretely, UCA agent is being integrated with PTT through implementation of simple forms. PTT takes care of delivering and displaying the forms on the user side, while UCA is responsible for the form processing. Forms are being automatically generated by PMS during run-time in accordance to overall process status and process configuration. Integration of UCA with PTT adds a more flexible way of data collection and user communication to Secricom infrastructure.

The Secricom Multi Bearer Router (MBR) is a modular router development platform and is developed by a UK-based company QinetiQ [8]. MBR provides one of the core Secricom platforms and delivers the IPv6 network enabling overlay. It provides seamless, ad-hoc

end-to-end connectivity between various legacy and emerging next generation, static and mobile bearers, networks and user access devices. SAI integrates with MBR using the IPA agent. MBR must be equipped with SDS in order to provide trusted and attested execution environment for agents. IPA agent is able to configure different properties of network such as communication prioritization or bandwidth control between different bearers.

8. CONCLUSION

In this article we have analyzed the requirements for agent-based systems and have proposed a distributed architecture designed for management of crisis situations. We have decomposed the proposed agent architecture to subsystems and identified several core agents to be used in an architecture implementation. A sample scenario was described, which demonstrates possible use of individual agents in case of a crisis in a distributed IP-based communication infrastructure. Lastly we described the use of the proposed architecture within an EU integrated project called Secricom.

Currently the proposed architecture is being implemented in Java [10] using a Jini [11] services technology framework. All the subsystems identified in Table I are implemented and are in pre-prototype version. Core agents (Table II) are also implemented and deployed in the system. Currently integration work is in progress with SDM, PTT and MBR systems as described in section V.

Our overall goal is to provide full prototype implementation of the proposed framework. We believe that beside crisis management there are many other application domains where trusted code execution using agents is appropriate to use and where the proposed distributed agent-based architecture would suit well. In the future we plan to identify other suitable problem domains for our architecture and to customize the system for use in other challenging distributed infrastructures.

ACKNOWLEDGMENTS

This work is supported by the projects SECRICOM FP7-218123, APVV DO7RP-0007-08, SMART ITMS: 26240120005, VEGA No. 2/0211/09.

REFERENCES

[1] JANSEN, W. – KARYGIANNIS, T.: Mobile Agent Security – NIST Special Publication 800-19. National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD 20899, 1999.

[2] BORSELIUS, N.: Mobile agent security, Electronics & Communication Engineering Journal, Volume 14, no 5, IEE, London, UK, pp 211-218, Oct. 2002.

[3] KOCIS et al.: SECRICOM – Analysis of external and internal system requirements. Deliverable report D2.1, the SECRICOM project, Feb. 2009.

[4] O'NEILL et al.: SECRICOM – Analysis of Crisis Management System Requirements. Deliverable report D2.2, the SECRICOM project, Feb. 2009.

[5] ŠIMO et al.: SECRICOM - Security requirements and specification for docking station module. Deliverble report D4.1, the SECRICOM project, Apr. 2009, http://www.secricom.eu/public-deliverables

[6] HEIN et al.: Functional specification of the Secure Docking Module. Deliverable report D5.1, the SECRICOM project, May 2009, http://www.secricom.eu/public-deliverables

[7] Ardaco homepage. http://www.ardaco.com/

[8] QinetiQ homepage. http://www.qinetiq.com/global.html

[9] Secricom Project Homepage. http://www.secricom.eu/

[10] JAVA Home. http://java.sun.com/

[11] Jini Homepage. http://www.jini.org/

Received January 2, 2010, accepted April 14, 2010

BIOGRAPHIES

Ladislav Hluchý is the Director of the Institute of Informatics of the Slovak Academy of Sciences and also the Head of the Department of Parallel and Distributed Computing at the Institute. He received his MSc and PhD degrees, both in computer science. He is R&D Project Manager, Work-package Leader in a number of 4FP, 5FP, 6FP and 7FP projects, as well as in Slovak R&D projects (VEGA, APVT, SPVV). He is a member of IEEE, ERCIM, SRCIM, and EuroMicro consortiums, the Editor-Chief of the journal Computing and Informatics. He is also (co-)author of scientific books and numerous scientific papers, contributions and invited lectures at international scientific conferences and workshops. He is also a supervisor and consultant for Ph.D., master and bachelor studies.

Zoltán Balogh is a researcher at the Institute of Informatics of the Slovak Academy of Sciences. In 1999 he received his M. Sc. degree in management. He received his PhD in applied informatics in 2007. Since then he is employed at the Institute of Informatics, SAS. He is the author or co-author of several scientific papers. He participated in many European IST projects as well as in several national projects (VEGA, APVV, APVT, SPVV). His research interests include Cloud computing, services-oriented architectures, knowledge-based systems, ontologies, case-based reasoning, and application of these approaches in business systems.

Emil Gatial is a researcher at the Institute of Informatics of the Slovak Academy of Sciences. In 2002 he received his M. Sc. degree in information systems. Since 2003 he is employed at the Institute. He is author or co-author of several scientific papers. He participates in the CrossGRID FP5, K-Wf Grid FP6, DEGREE FP6, SECRICOM FP7 as well as in several national projects (VEGA, APVV, SPVV). His research interests cover the domain of mobile agents, knowledge driven processes and semantics in business systems.