# Controlling Security of Software Development with Multi-agent System

Esmiralda Moradian and Anne Håkansson

Department of Communication Systems, KTH,
The Royal Institute of Technology
Forum 100, 164 40 Kista, Stockholm, Sweden
moradian@kth.se, annehak@kth.se

**Abstract.** Software systems become distributed and complex. Distributed systems are crucial for organizations since they provide possibility to share data and information, resources and services. Nowadays, many software systems are not developed from scratch: system development involves reuse of already developed components. However, with the intrusion in the computer systems, it has become important that systems must fulfill security goals and requirements. Moreover, interdependencies of components create problems during integration phase. Therefore, security properties of components should be considered and evaluated earlier in the lifecycle. In this paper, we propose an agent-oriented process that supports verification of fulfillment of security goals and validation of security requirements during different phases of development lifecycle. Moreover, the system needs to support mapping of security requirements to threat list to determine if any of the attacks in the list is applicable to the system to be developed. This is performed by the meta-agents. These meta-agents automatically create a security checklist, as well as, provide control of actions taken by human agent.

**Keywords:** Multi-agent system, security engineering, risk management, security checklist, control system.

## 1 Introduction

Interdependencies between organizations in different sectors, such as business, financial, government, and medical are increasing. Hence, software systems become distributed and thereby more complex. However, distributed systems must be able to fulfill security goals and requirements and be able to operate securely. Unfortunately, security is often neglected during the development process. When considered, security is "relegated to the status of a few add-on fixes when all other design decisions have been frozen" [1]. Thus, security solutions, when implemented are isolated, and inadequate to business requirements. Lodderstedt *et al.* [2] point out three reasons for isolation, which are lack of understanding that security should be integrated in the development process, lack of tools supporting security engineering and lack of experience of the developers. Several security standards, among others

R. Setchi et al. (Eds.): KES 2010, Part IV, LNAI 6279, pp. 98–107, 2010.

Common Criteria (CC), and ISO 27000 series, make it possible to handle security requirements [3]. However, evaluation according to any of the standards is resource and time demanding [4]. Faults in software systems are unavoidable. To identify problems and find security risks that are not detected by human, automated processes are needed. We propose an agent-oriented process that can support security requirements engineering by performing automated search, provide checking for documents and intelligent matching of information. In the research in this paper, we present search, check and match processes that verify and validate security requirements against goals, risks and standards. Interdependencies of components create problems during integration phase. Hence, combined properties of components should be evaluated earlier in the lifecycle [1]. The work here explores the possibilities to automatically inspect combination of components during design phase and identify and analyze components that are necessary to achieve goals and fulfill requirements.

## 2 Related Work

Mouratidis *et al.* [5] present an approach that integrates security and systems engineering process. The work is done within the context of the Tropos methodology where security requirements are considered as an integral part of the whole development process. Authors [5] introduce security-oriented paradigm to the software engineering process. The process is iterative, which allows redefinition of security requirements in different levels. In our research, we present a semi-autonomous approach of controlling development of new systems or modification of existing systems. We focus on verification and validation of security requirements throughout the development process.

Brændeland and Stølen [6] propose an integrated process for component-based system development and security risk analysis. The authors [6] concentrate on identifying stakeholders and assets at the component level and identify and estimate the risks of component interaction. Moreover, they describe how the basic components can fit together into a composite component that refines the original requirements. In our work, we propose a multi-agent system to verify and validate security requirements and perform analysis of security properties of identified components. Moreover, the agent system inspects interdependencies of components. The output is a security checklist presented to the human agent.

Security checklist is not a new paradigm. Gilliam *et al.* [7] focuses their research on the development of a Software Security Checklist (SSC) for the life cycle. The authors [7] consider requirements engineering and specification, design and code issues, maintenance and decommissioning of software systems. Gilliam *et al.* [7] point at SSC as an instrument to guide organizations and system engineers in integrating security into the software life cycle. [7] Many checklist approaches were developed. However, listing pinpoints, without understanding and testing how they fit together during different phases of the system development, do not help in creating security architecture [1]. Thus, a multi-agent system is necessary in order to provide a holistic view throughout lifecycle. In our work, we propose the checklist that is automatically created by meta-agent as a result of performed analyses.

## 3 Agents in Multi-agent System

Nowadays, agent technologies are used in information and communication systems in order to provide management, search, monitoring, etc. In this paper, we present the multi-agent system that enforces control of security requirements and secure design. Moreover, the multi-agent system enables verification and validation of requirements throughout software development process. To handle this quality assurance, the multi-agent system consists of software agents and meta-agents where agents are working as a team.

The agents are communicative, mobile, cooperative, goal-oriented, autonomous, adaptive, and reactive [8]. Agents are communicative in the sense that they can communicate with users, other agents in the system, and other systems. Software agents communicate with the user in order to receive commands. From these commands, the agents search for information. The agents move between different locations over the networks while searching for available components and are therefore mobile.

Cooperation with the meta-level agents consists of passing significant information. Meta-agents use the information for analyses, control and matching. The meta-agents communicate with each other to united solve the problems [9]. They have ability to cooperate, i.e. work together with other agents to solve tasks. Goal oriented agents work towards a specific goal(s) [10]. The tasks have to be performed by the agents without any external guidance, and, thus, must be autonomous. Autonomous agents can operate without human intervention.

Software agents execute the user input. These agents are used to provide search for documents according to received information. Moreover, software agents can observe and monitor user actions through Interface module. To minimize search time in our system, software agents perform search in parallel in multiple databases.

Additionally, meta-agents are used as management, coordination, matching and checking agents. The meta-agents, in our system, can compare, analyse and combine information received from software agents. The meta-agents are also autonomous, i.e. meta-agents are able to act autonomously, reason and take decision in order to satisfy their objectives.

The meta-agents interact with software agents and other meta-agents to satisfy goals [9]. The communication with the software agents consists of collecting information from the software agents and giving commands that the software agents have to execute [11]. Meta-agents are reactive and adaptive, since they can learn, respond and adapt to changed environments.

The agents, in our multi-agent system, work in a deterministic environment, which means that next state of the environment is determined by the current state and the action that is being executed by an agent [11]. Environment is episodic, since software agents perform one task at the time while searching for information in databases. The task is complex and, therefore, divided into many simple tasks. Thus, we use multiple agents due to task complexity. To increase efficiency and shorten search time, the team of software agents execute in parallel. The environment, agents work in, is accessible, since the agents have access to information needed to accomplish tasks and satisfy goals.

Dynamic environment refers to the environment that can change. In our work, the agents need to monitor changes in the environment during execution because checklist is continuously updated by the human agent and requirements can change during the development process.

## 4 Security Requirements

Requirement describes the activities of the system to fulfil the purpose of the system and specify a system's behaviour [12]. Security is about protecting business goals and assets [1]. Analysis of business requirements enable to determine security goals of the system. Security policy expresses clearly and concisely what the protection mechanisms are to achieve and contains security goals agreed by the management [13]. Security goals help to identify security requirements, i.e. security constraints. Thus, security requirements must be identified and analysed according to business requirements and goals (business and security). The security policy specifies security properties of the system (i.e., identify secure and non-secure states, describe the secure way of information management, conditions of altering data, and identify level of trust), and exists to guide decisions in order to achieve desired goals [14]. However, the security policy does not state how security is to be delivered [1].

Building secure system requires that risk management is a part of development process [15]. The purpose of risk management is to produce knowledge about relevant security characteristics of the system(s). Risk management ensures that information security is managed appropriately during all phases in a system lifecycle. The first step is to assess criticalities of the system to be developed through identification of risks. Criticality is a measure of the extent to which a deficiency of the system in its environment may affect security [16]. Next step is to define or update the baseline of security requirements [17]. The ongoing risk analysis leads to the definition of security requirements and reduction or acceptance of remaining risk. Risk expresses probability (P) that undesirable event will occur and the consequence (C) of it. Security engineers have to understand that risks and threats should be put in content, and made realistic assessments, of what might go wrong [13]. Requirements engineering often suffers from problems such as: requirements are specified without any analysis or analysis restricted to functional end-user requirements; requirements specification is incomplete, inconsistent, and incapable of being validated [18]. Several standards that deal with security requirements exist. However, many researches [4, 19, 3, 20, 21] point out issues such as: lack of guidance on how to fulfill requirements during the development process [21], lack of methodological support [3], evaluation is time consuming [19, 21], and existence of gaps and overlaps [20]. Hence, analysis of security requirements and mapping those to security standards, as well as, control of implemented actions, that supports management, is needed. Therefore, a multi-agent system where meta-agents provide verification and validation of security requirements against goals, standards, risk analysis, as well as, control of performed actions, is highly needed.

## 5 Secure Architectural Design

This section presents secure architectural design of the enterprise system. When building distributed systems, architects need to considerate that resources and systems from different organisations must work together. This involves Web servers, databases, security systems, business logic components etc [22]. It is essential to describe components and analyse their properties, interconnection, and interdependencies, in order to, specify communication and control mechanisms, as well as, control behaviour of the system as a whole. Security architecture should encompass an enterprise-wide view, i.e., from goals and requirements to operation and maintenance.

Secure design embraces logical, physical and component security architecture [1]. The logical security architecture identifies the relationships and interdependencies between various elements of the system. Logical security architecture concerns with specifying logical security services (i.e., confidentiality and integrity protection); entities (i.e., users, administrators); domains; and security processing cycle (i.e., registration, login, session management) [1]. Physical security architecture describes physical devices that deliver logical services. It concerns with specifying data model and structures (i.e., tables, messages, signatures), rules (i.e., conditions, procedures, actions), security mechanisms (i.e., access control, encryption, virus scanning), security technology infrastructure, and time dependency (i.e., events, time intervals) [1].

Each component is an element of the whole system and its integration into the system affects security of the overall system, since components are selected from different vendors. Thus, component selection and component building is an important step, since it may be the case that some components are not available or not suitable for the specific system. During design phase should be determined what components to build or to buy [1]. Sherwood [1] points out that the components with compatible standardized interfaces should be used in order to achieve integration. Security analyses of the components, as well as, security evaluation of each identified component and all components together are necessary to achieve goals and fulfill requirements. As soon as, components are identified, which are necessary for goal fulfilment, the relationships between components are analysed. Thus, all security standards that are required, descriptions and specifications of all selected strategic technologies, products and tools along with guidance on how, why, where and when they should be used, are described during design phase. Moreover, roles, access rights, all processes, procedures and activities and how they relate to each other should also be specified.

Size and complexity of the systems are growing. Therefore, architectural risk management should be performed to refine identified assets and risks. Risk management is the ongoing process throughout development lifecycle. It involves threat and vulnerability analysis and identification, and analysis of design flaws. Vulnerability analysis is performed to understand the internal dependencies as well as the impact of external software dependencies [23]. Threats are mapped to the potential vulnerabilities and the security controls. Thus, architectural risk analysis identifies risks in the architecture, which improves the built-in security of a system by determining the security risks of the system. Misuse cases, i.e., negative scenarios or use cases are designed in order to determine behaviour of the malicious user.

We propose using multi-agent system, where the meta-agent analyse requirements and map requirements and attack patterns to each other in order to determine if any of

attacks in the list are applicable to the system to be developed. Moreover, from knowledge repository the meta-agents select those attack patterns that are relevant to the system to be developed.

## 6 Multi-agent Control System

Many systems are built, operated, and maintained by different groups or organizations. Security plays an essential role in the development lifecycle. Security goals and priorities define security requirements of the overall system [17]. Policies, goals, risks, and requirements are analysed in order to identify conflicting, redundant and missing goals and detect insufficient, inconsistent, incorrect, unrealistic and untraceable requirements. Software system architecture should conform to the goals and requirements that are to be fulfilled, which include performance, security, and interoperability, integration, and supportability. Vulnerabilities embedded in the software and system components affect security of the system [17].

An agent based security requirements engineering consists of a multi-agent system with meta-agents and software agents. The multi-agent system is depicted in Figure 1.

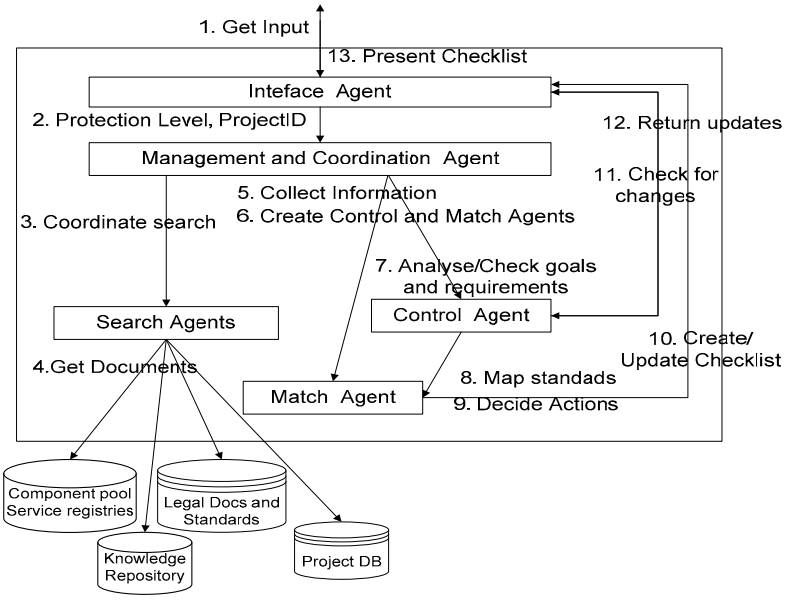


**Fig. 1.** Multi-agent system

The multi-agent system, in this research, is semi-autonomous. The agent system interacts with the external human agent in order to execute commands and capture changes. The process is depicted in Figure 1. The role of human agent is assigned to relevant actor throughout the lifecycle, which means that human agent can be a business and security manager, requirements engineer and risk manager as well as

developer, architect, security specialist, and test specialist. [17] The interaction is going through the Interface Agent (IA). The Human agent (legitimate user) makes an input, by providing protection level and ProjectID. Documents that belong to the specific project should be tagged with its ProjectID. The IA gets this information (1:1) and passes it over to the Management and Coordination Agent (1:2). The Management and Coordination Agent is a meta-level agent that acts as coordination agent at this point. The meta-level agent is responsible for allocating search task to search agents, which are software agents and coordinating them. Depending on task complexity, agents can create new agents by cloning themselves. The replication allows performing tasks in parallel manner. The Management and Coordination agent assigns search tasks to search agents, i.e. gives a command to get documents according to input that are protection level and ProjectID (1:3, 1:4).

Search agents perform parallel search for documents in databases (1:4). For this pattern search is used. The software agents start to execute and a meta-agent is following along to the next node. Meta-agents provide control of software agents by keeping track on them. For the amount of databases to be searched the numbers of meta-agent and software agent are expanded. For the documents, a copy of the meta-agent and software agents are created and expanded with the amount of document. The procedure continues until agents reach every document that is relevant according to the input information. Notice, since there is a database with standards we need to consider the lifecycle phase, i.e. during requirements phase only standards that provide guidance for management security requirements will be valid. The meta-agents collect all information from search (1:5) and collaborate with each other and make decisions on how to proceed. Decision can concern standard(s) that should be used in the current phase, knowledge that is applicable for specific requirement, etc. The Management and Coordination Agent clones itself to Control Agent and Match Agent (1:6). The Control Agent is responsible for analyses and checks of documents (1:7), which contain important information that is necessary to perform the task. The documents such as business goals, system goals, requirements specifications (including security requirements) as well as risk analyses documents are analysed. Analyses include identification of conflicting, redundant and missing goals and insufficient or incorrect requirements. If such goals and requirements are identified, the system development is halted [17]. The agent system interacts with human agent in order to inform and receive new commands. The Match agent is responsible for performing mapping of standards, requirements, goals, security standards, domain expertise (1:8). Moreover, the Match Agent decides actions that should be implemented during development phases in order to satisfy goals (1:9) and create and update the checklist (1:10). The checklist is presented to the user by the Interface Agent (1:13). Goals and requirements can change during the development process. If that is the case then the checklist must be updated. The Control Agent needs iteratively perform checks of changes through Interface agent in order to detect deviations and capture new information (1:11). We assume that every single change is documented. If preconditions (goals and requirements) differ from initially defined conditions, the development of the system is halted until new search and analysis is performed (steps1:3 to 1:9). Moreover, the Control Agent is also responsible for control of the performed actions and validation of the requirements satisfaction. This meta-agent inspects the processes and procedures that have been applied. Check for changes (1:11, 1:12) are performed

at each stage of the development process. All performed actions are documented in the checklist. The process is iterative and terminates when defined goals and requirements are satisfied [17].

Defective design accounts for 50% of security problems [15]. During the design phase software agents search (according to request from human agent) for available components and/or services (both internally within organisation and externally) in the component pool or in the service registry. See Figure 1:4. The meta-agent analyse component security properties and presents available components and analyses to the human agent. Software agents also perform search for documents such as, requirements, risks, list of threats and vulnerabilities. A Meta-agent compares and analyzes requirements and use cases and maps those to the attack patterns in order to determine which attacks can target the system. A Meta-agent collects and interprets the available information and maps the requirements to the threats in order to determine if any of attacks in the list are applicable to the system to be developed. The process is depicted in Figure 2.

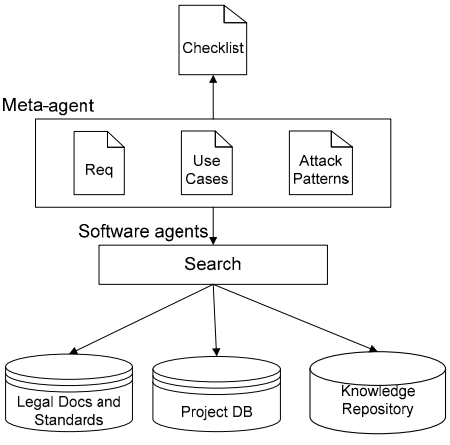


**Fig. 2.** The process of determining attacks

The meta-agent constructs a checklist with possible attacks that may target the system and communicate it to the human agent through the Interface agent, which is depicted in Figure 1. The checklist, created by the meta agent, can contain (but not limited to) following checkpoints: check authentication rules, implement random password generation, identify flaws, set up access control list, perform threat and attack analysis, design abuse cases, perform security testing for system integration, perform penetration tests, analyse network, implement and verify secure connections [17]. The Label "Yes" indicate fulfilled goals and requirements while "No" point to the unfulfilled goals and requirements. The checklist is extended and modified as soon as the change takes place. The resulting checklist is examined continuously both by the human agent and the meta-agent in order to detect changes, determine which of the security relevant activities took place, when (time, which phase of development lifecycle) and who (which human agent) is responsible for the specific activity.

## 7 Conclusion

In this work we presented a multi-agent system that can be used as a semi-automatic control system for verification and validation of security requirements throughout development lifecycle. Moreover, the proposed system can identify essential points in security requirements, such as conflicting, incomplete and inconsistent requirements. Furthermore, the system can compare and analyze requirements and use cases and map those to attack patterns in order to determine which attacks can target the system. The multi-agent system monitors documentation that includes requirements specifications, architecture overviews, component structure, interface specifications, analysis and results in order to keep track on changes in environment and have updated information. The proposed system can also control actions of the human agent during all the phases of system development.

## References

1. Sherwood, J., Clark, A., Lynas, D.: Enterprise Security Architecture A Business-Driven Approach. CMP Books (2005), ISBN 1-57820318-X
2. Lodderstedt, T., Basin, D., Doser, J.: A UML-Based Modeling Language for Model-Driven Security*. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
3. Mellado, D., Fernández-Medina, E., Piattini, M.: A common criteria based security requirements engineering process for the development of secure information systems, vol. 29(2), pp. 244–253. Elsevier Science Publishers B. V, Amsterdam (February 2007), ISSN:0920-5489
4. Abbas, H., Yngström, L., Hemani, A.: Option Based Evaluation: Security Evaluation of IT Products Based on Options Theory. In: First IEEE Eastern European Conference on the Engineering of Computer Based Systems, pp. 134–141 (2009)
5. Mouratidis, H., Giorgini, P., Manson, G.: Integrating Security and System Engineering: Towards the Modelling of Secure Information Systems. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 63–78. Springer, Heidelberg (2003)
6. Brændeland, G., Stølen, K.: Using Model-based Security Analysis in Component-oriented System Development QoP 2006, Alexandria, Virginia, USA. Copyright 2006, ACM 1-59593-553-3/06/0010 (October 30, 2006)
7. Gilliam, D.P., Wolf, T.L., Sherif, J.S., Bishop, M.: Software Security Checklist for the Software Life Cycle. In: Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003), pp. 243–248 (June 2003), ISBN: 0-7695-1963-6
8. Phillips-Wren, G.: Assisting Human Decision Making with Intelligent Technologies. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part I. LNCS (LNAI), vol. 5177, pp. 1–10. Springer, Heidelberg (2008)
9. Håkansson, A., Hartung, R.: Calculating optimal decision using Meta-level agents for Multi-Agents in Networks. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 180–188. Springer, Heidelberg (2007)
10. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1995), ISBN: 0-13-103805-2

11. Moradian, E., Håkansson, A.: Approach to Solving Security Problems Using Meta-Agents in Multi Agent System. In: Nguyen, N.T., Jo, G.-S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS (LNAI), vol. 4953, pp. 122–131. Springer, Heidelberg (2008)
12. Pfleeger, S.L.: Software Engineering Theory an Practice, 2nd edn. Prentice-Hall, Inc., Englewood Cliffs (2001), ISBN 0-13-029049-1
13. Andersson, R.: Security Engineering A guide to building. Dependable Distributed Systems, 2nd edn. Wiley, Chichester (2008), ISBN 978-0-470-06852-6
14. Bishop, M.: Introduction to Computer Security. Pearson Education, Inc., London (2005), ISBN 0-321-24744-2
15. McGraw, G.: Software Security Building Security. Addison-Wesley Pearson Ed., Reading (2006), ISBN 0-321-35670-5
16. Lindström, C., Näsström, S.: Handbook for Software in Safety-Critical Applications. Swedish Armed Forces (2005)
17. Moradian, E., Håkansson, A., Andersson, J.-O.: Multi-Agent System Supporting Security Requirements Engineering. Accepted in The 9th International Conference of Software Engineering Research and Practice, SERP 2010 (2010)
18. Allen, J.H., Barnum, S., Ellisson, R.J., McGraw, G., Mead, N.: Software Security Engineering A Guide for Project Managers. Addison-Wesley, Reading (2008), ISBN: 0-321-50917X
19. Haley, C.B., Moffett, J.D., Laney, R., Nuseibeh, B.: A Framework to security requirements engineering. In: SESS 2006, Shanghai, China. Copyright, May 20-21, ACM 1-59593- 085-X/06/0005 (2006)
20. Magnusson, C.: Corporate Governance, Internal Control and Compliance (September 2007), http://www.svensktnaringsliv.se/material/rapporter/article35898.ece
21. Vetterling and Wimmel Secure Systems Development Based on the Common Criteria: The PalME Project SIGSOFT 2002/FSE10, Charleston, SC, USA, November 18-22. ACM 1581135149/02/0011 (2002)
22. Papazoglou, M.: Web Services: Principles and Technology. Pearson Education, Essex (2008), ISBN: 978-0-321-15555-0
23. Swiderski, F., Snyder, W.: Threat Modelling. Microsoft Press(2004), ISBN 0-7356-1991-3