A Risk Management Approach for Distributed Event-Based Systems

Sergey Savinov
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
ssavinov@cs.uwaterloo.ca

Paulo Alencar
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
palencar@cs.uwaterloo.ca

Technical Report CS-2014-06

Abstract

In this paper we present a Risk Management System. The goal of the RMS is to decrease risks related to information access. The goal is met by taking the context of operations into account; providing motivation to users through incentives (e.g., punishments or rewards); providing support to track changes and ability to reason about risk-related attributes; providing support to reason about tasks, plans and goals; providing process-based system guidance; decreasing the costs per operation for growing systems by making human intervention optional.

1 Introduction

Distributed Event Based Systems (DEBS) constitute a well-known paradigm for asynchronous communication in which components interact in distributed and heterogeneous environments. This paradigm has become increasingly popular in a wide range of application domains, which include emergency services, internet-based systems, mobile computing and healthcare [1,2].

Information security is a crucial topic due to problems such as unauthorized access and dissemination of critical information and malicious insider threats [3]. In 2010, 251,287 diplomatic cables had leaked from United States State Department information system [3]. In 2011 Facebook applications accidentally leaked access to users' accounts [5]. In 2012, five million of internal emails between the Stratfor's global intelligence company employees and its clients were disclosed [5]. In 2013, a technical contractor working for the United States National Security Agency (NSA) and a former employee of the Central Intelligence Agency (CIA) Edward Joseph Snowden leaked details of several top-secret U.S. and British government mass surveillance programs to the press [7]. Statistical data [8] indicates that the number of these security threats is sharply rising, from 333 reported accidents in 2006 to 801 in 2011. Indeed, these are only a few examples of information security breaches that have been investigated and documented. Each of these accounted accidents was worth over 600 hundred thousand dollars in damages. In total, it adds up to an enormous sum that grows every year and, in 2011 alone, “companies spent more than 500 million dollars on

neutralizing the consequences of leaks" [8]. Similar accidents happen in wide range of industries (e.g., space, social networks, finance), as well as in the government and private sectors.

Concerning malicious insider threats, in August 2012, an attack targeted the South Carolina Department of Revenue, when hackers used a valid login credential of one of the employees and, as a result, 3.6 million Social Security and 387,000 credit- and debit-card numbers were stolen. The cost of this attack was estimated to reach over 14 million dollars [9]. In addition, in April 2013, "the German state of Rhineland-Palatinate has announced that it has bought a CD containing data on secret bank accounts in order to track down suspected tax evaders. It hopes to recover 500 million euro in unpaid taxes" [10].

Although traditional security approaches, such as access control methods, and leak and intruder detection techniques [11], have currently been used to target to some extent the problems previously mentioned, distributed event-based systems require a fundamentally different approach that must deal with their heterogeneous, loosely coupled and dynamic nature [1]. First, in DEBSs, access control and threat assessment needs to take into account the event flow among heterogeneous distributed components [12]. Second, since DEBSs are loosely coupled, the security solutions need to work in settings where event subscribers are not known by event publishers. Third, DEBSs are highly dynamic, and access control and threat assessment cannot be restricted to a fixed set of security attributes and policies, but need to address dynamic context-aware scenarios and support dynamic and adaptive policies [13].

In this paper, I propose a novel risk management approach for DEBSs that involves adaptive access control and agent-based threat assessment (e. g, leak detection, malicious insider attacks).

The rest of the paper is organized as follows. Section 2 describes related work, and Section 3 presents the requirements for my approach to the problem. Section 4 outlines the proposed risk management approach. Section 5 briefly discusses how this approach will be evaluated, and, finally, Section 6 presents conclusions and future work.

2 Related Work

This proposal is related to several areas, including

- Risk Management
- Distributed Event Based Systems (DEBS);
- Access Control models;
- Risk-adaptive access control (RADAC) based on Risk-Benefit analysis;
- Multi-agent systems;
- Prediction and evaluation of component-related processes.

2.1 Distributed Event Based Systems

Distributed Event Based Systems (DEBSs) are systems that support asynchronous communication among components that interact in distributed and heterogeneous environments via events [14]. These systems relies on implicit invocation, meaning that message passing is asynchronous and event-based, and not based on traditional request-reply mechanisms [14]. Any entity interacting within the system is represented as a component, and each component can be an event publisher or subscriber (Figure 1). Events are generated by components called publishers, and components interested in specific events that have been generated are called subscribers. Publishers must declare the events they will be generating by advertising these events. A component subscriber is notified when an event of interest to this component is generated by a publisher component [15]. Publishing and subscribing rely on event schemas, which define the data attributes associated with the event. Subscribers receive the events they are interested in through the publication, routing, filtering and aggregation of events.

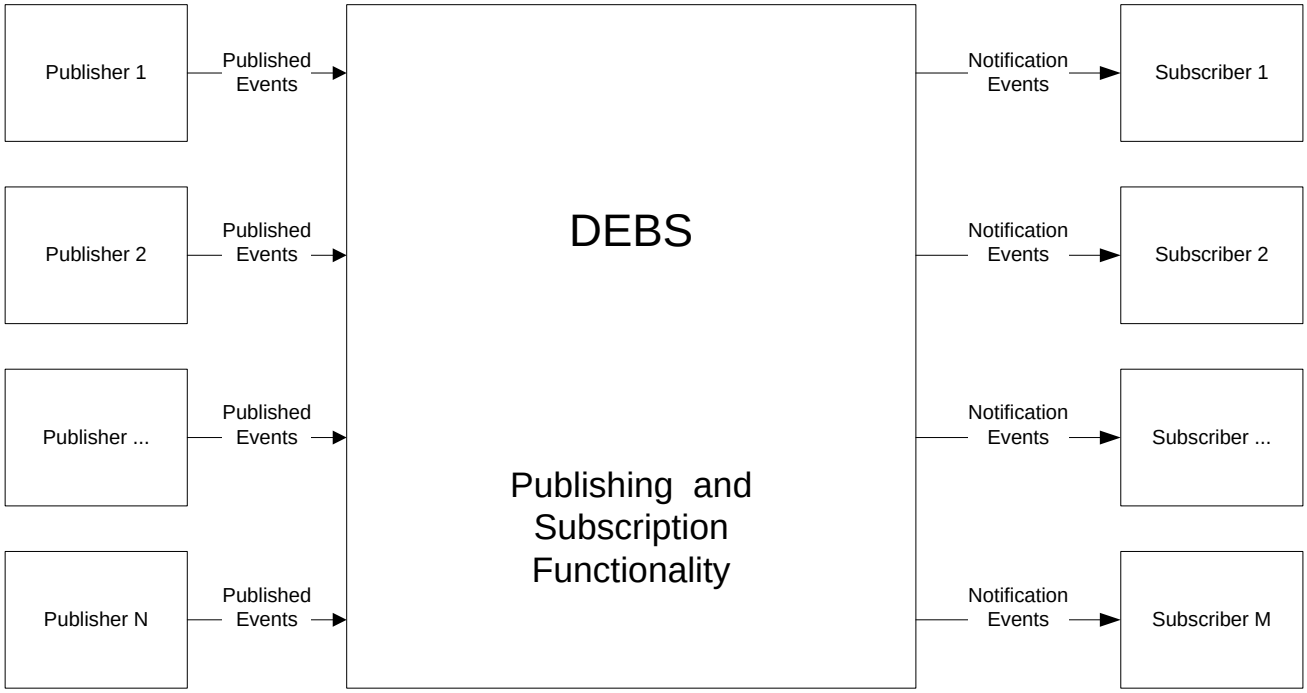

Figure 1. A Typical Distributed Event-Based System.

An approach for DEBS access control, and leak and intruder detection, must deal with the heterogeneous, loosely coupled and dynamic nature of this kind of systems [17]. First, loosely coupled components give potential intruders an advantage because any of these components can be used as an "entry point" for attacks that can compromise the system [18]. Second, time and space decoupling can be abused since a malicious component can ask system to replay previous event

sequences and analyze these sequences to understand the system operations. Third, due to dynamic nature of DEBSs, an intruder can use broadcasting to launch an effective denial of service attack. Fourth, access control and threat assessment needs to take into account the event flow among heterogeneous distributed components, since this complex flow makes the access control and threat analysis much more complex.

To provide an idea about the challenges faced when DEBSs are analyzed in terms of general information security (e. g., access control, threat analysis), a typical example of a DEBS such as SCADA (a supervisory control and data acquisition system) has significant problems [19]:

"Likely impact scenarios for this lab environment include the following:

- Reconfiguring a relay;
- DoS to systems/devices;
- Modifying/disrupting valid alarms;
- Producing fake alarms;
- Sending incorrect commands to relay;
- Manipulating readings from a relay;
- Injecting incorrect data to historian."

"During the assessment only one high level vulnerability was found due to a default password. Most other vulnerabilities fell into a category of security best practice violations like information disclosure and unnecessary open ports."

Although many impact scenarios were presented, only one high level vulnerability was found, since the methods used in this research only account for network level analysis. This conclusion shows that there is a lack of analysis methods to assess the consequences of unauthorized access and dissemination of critical information and malicious insider threats. Indeed, when DEBSs is considered the analysis methods become much more complex than when traditional systems (e. g., client-server) are considered [20]. For example, when a concurrent event is fired, filters are triggered, versions of this and other events are aggregated, and components can recursively fire other distributed events in response to the events previously fired.

There is also a lack of analysis methods to assess the risk of unauthorized access and malicious insider threats [21]. Traditional client-server security architectures assume that servers are reactive components in the sense that they only apply security policies when answering requests, and not in a proactive way. To be proactive, the methods need to address dynamic context-aware scenarios and support dynamic and adaptive policies [22,23]. For example, an emergency situation is considered in which a developer needs to get access to a file that is locked because another developer was working with the file, did not finish his job and is currently not available. Traditional

methods disallow access to the file but a proactive risk-based approach could take into account a high priority for fixing the bug (i. e., context), and adapt to the situation by unlocking the file and sending a notification to the other developer that the file was unlocked in order to maintain consistency.

Regarding malicious insider threats for DEBSs, there are also many challenges. A critical scenario involves situations in which the data is not encrypted and any malicious component can listen to the broadcasted events from all the system nodes that communicate through a limited set of middleware components. To address problem of eavesdropping on the system through a limited set of middleware components, all event data that is not necessary for communications should be encrypted [3,24] and all data that is relevant to communications should have a digest to make sure that this data can only be changed by authorized components.

2.2 Access Control Models

Access control refers to mechanisms and policies that restrict access to resources, thus regulating access to a system or to physical or virtual resources. Access control models are used to represent these mechanisms and policies by which users are granted access and specific privileges to systems, resources or information [25]. There is a variety of widely used access control models, including attribute-based access control (ABAC) [26,27], mandatory access control (MAC) [28], discretionary access control (DAC) [29] and role-based access control (RBAC) [30,31,32,33].

Traditional access control models were designed to be reactive, not proactive [25]. Some of these models are based on a client-server architecture, in which interactions have a reactive nature since they need to be invoked by clients in an explicit way. In contrast, proactive interactions need to be dynamic and rely on implicit invocation. Further, traditional access control models can not address dynamic context-aware scenarios and support dynamic and adaptive policies [34]. For example, client-server application programming interfaces (APIs) require a fixed set of variables as input, but do not take into account the context of operations [35].

Besides the well-known access control models (RBAC, MAC, ABAC, DAC), many other models were proposed in the literature [36,37,38,39]. For example, the break glass access control model suggests that "it is possible for a subject to break-the-glass and explicitly override the denied request" [40,41]. In addition, certain approaches can be used to transform one form of access control model into another, i. e., such as a "framework that uses attribute-based policies to create a more traditional RBAC model" [26].

Role-Based Access Control (RBAC)

Role-Based Access Control models refer to models used to regulate access to systems, resources or information based on the roles of individuals within an organization [42]. According to these models, only individuals performing certain roles have the ability to access required resources

or perform specific tasks such as view or modify data [43]. These models provide fine granularity and scalability, and can be easily implemented for small systems.

The RBAC model used as basic entities users, roles, operations and objects [21]. Permissions to access a resource are represented as pairs of operations and objects, and permission assignments are defined as pairs of permissions and roles. In addition, user assignments are defined as pairs of users and roles. In Figure 2 an overview of this model is provided.

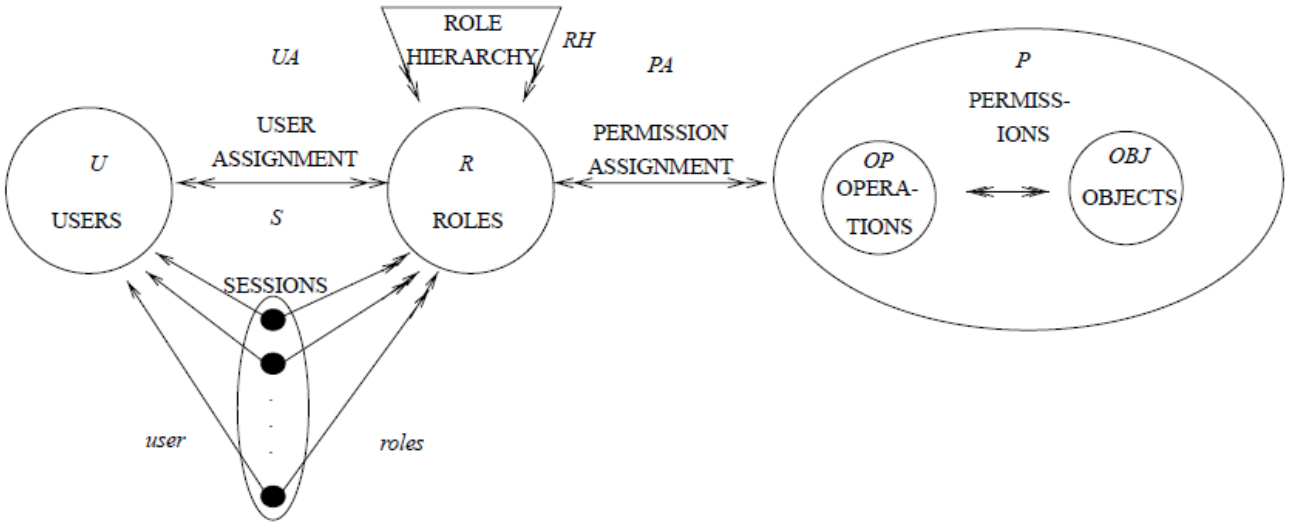

Figure 2. An overview of the RBAC model.

Although RBAC can be reviewed by one or a few analysts when adopted within small systems, for large systems the model can become increasingly complex due to the high number of rule exceptions involving a wide range of users, different domains and diverse scenarios [44,45]. RBAC rules for small systems are not too hard to understand. However, for large systems, these rules become extremely complex and counter-intuitive due to the exceptions and the dynamic nature of event-based systems. This complexity may lead security analysts to omit some critical behaviour. Further, exceptions tend to accumulate and it becomes increasing difficult to resolve and maintain them.

There are attempts to introduce Role Based Access Control into event-driven environments. In one case, a verification framework for the detection and resolution of inconsistencies and conflicts in event-driven RBAC policies was provided [30].

Mandatory Access Control

Mandatory access control models refer to models in which security labels or classifications are assigned to resources and access is granted only to entities (components, users, processes, devices) with distinct level of authorization or clearance [29].

Attribute Based Access Control

Attribute-based access control models refer to models in which access rights are granted to users (e. g., components, persons, devices, processes) via the use of policies that combine attributes [27]. These policies involve different types of attributes (e. g., component attributes, user attributes, resource attributes). Time and space attributes are especially relevant when temporal and geo-spatial access control policies are required [46].

Discretionary access control

Discretionary access control models refer to models that restrict access to resources based on the identity of users (e. g., components, persons, devices, processes) or groups to which they belong [25]. The access control is named discretionary because in this case a user with specific access permission can pass that permission to any other user. In this way, a user owning a resource can grant or deny access to other users.

2.3 Risk Management

Risk management in information technology is a methodology, which is used in a process of reducing or preventing a potential harmful consequences caused by threats to the information systems. Set of standards was developed for the risk management [ISO 27005, NIST SP 800-30]. These standards describe a structured process of risk analysis, assessment, and mitigation. Software solutions for risk management are also available (i.e., SAP Risk Management).

However, existing tools for the risk management do not support a formal reasoning, as well as lack explicit mechanisms of threats consequences quantification, instead, they rely on a human analyst to operate them. Thus, an existing implementations of the risk management methodology cannot be used both on demand (if acceptable delay is in the order of seconds) and in real-time systems.

One of the basic tools extensively used in the risk management is an access control, and while an existing access control approaches can mitigate the security risks, none of the approaches provide a real-time feedback for the risk management systems. Also, providing the access control requires either an access control decisions to be made in response to access control requests, or a system to monitor components actions in real-time, none of which is possible within a frame of the existing risk management methodology.

Interaction with users

Each user interacting with the RMS is modelled using an agents. The agent model is updated whenever any information relating to the actions of that user becomes available to the RMS. There exists three forms of the updates. A first form is a direct observation of the user actions through an access control requests. A second form is a direct observation of the user actions through a context changes. A third form is a discovery of the tasks performed by the users using a process and event

based inference. A goal of the model usage is to provide an accurate prediction of the future actions through an inferred set of goals each user is associated with.

Through prediction of the future actions it is possible to assess the potential risks of the future user (and component) actions and correct a course of events to a more desirable outcome through direct (external action in system description) or indirect (by providing suitable access control decisions or specific processes to follow to get access to requested resource) actions.

RMS operation modes

There are 3 different modes the RMS can operate in. A first, manual or advice mode is a mode when the RMS observes an existing system operations and gives a recommendations to changes an access control policies that will bring the system closer to a system-wide goal compliance. The RMS can also log any difference between an existing access control system decisions and these recommended by the RMS. This way an operator of the RMS can test how it can work before investing significant amount of resources to implement RMS completely, and properly configure it (i.e. to match behaviour of the existing access control system initially).

A second, semi-automatic or balanced mode is a mode when the RMS replaces the existing access control system, but still requires a lot of input from a human operators (including a security analysts, a network administrators, etc.). In this mode the RMS works very similar to the existing access control systems by answering an access control request with an access control decision. The only noticeable difference between behaviour of the RMS and this of the previous access control system is that the RMS can dynamically change policies, basing on a conditions that the security analysts would specify.

Third, automatic or complete mode is a mode when the RMS includes all functional blocks and is able to work autonomously.

2.4 Risk Adaptive Access Control (RADAC) and Risk/Benefit Analysis

Risk Adaptive Access Control uses risk calculations to determine whether access to resources should be granted or denied [47]. The risk calculation is based on a function that takes risks [48] into account through quantitative measures such as the ones involving reputation, security levels, and reliability [48]. In comparison to traditional access control models (RBAC, MAC, DAC, ABAC), this approach allows for the resolution of dynamic access control situations using the quantitative measures in a way that minimizes risks [50]. The functions that compute risks based on these measures rely on risk and benefit calculations.

The risk/benefit calculations indicate that it is not enough to calculate just the risk of granting access [51], and that four types of risks and benefits are required. The first type relates to the risks of granting access to resources. For example, if the access control system grants access to certain information, there is a risk that this information can be maliciously disseminated (e. g., information

leak). The second type relates to the risks of denying access to resources. For example, if access is denied to a requested resource, the developer might be encouraged to follow the a path that will lead to project delays. The third type relates to the benefits of granting access to resources. For example, by granting access to a resource such as software code, possible project delays may be reduced. The fourth type relates to the benefits of denying access to resources. For example, denying access to a file on a heavily loaded server can lead to a load decrease.

Regarding information leakage [52], "the proposed applications that are available to tackle incidents of data leakage from outbound email are proposed as additional plugins in commercial email security platforms (McAfee Data Loss Prevention, Symantec Data Loss Prevention), as standalone applications (WebsenseTruWeb DLP, Proofpoint Enterprise Privacy, MailMarshal) or as features in networking devices (CISCO IronPort email)" [52]. However, "several security software vendors now offer 'data loss prevention' (DLP) solutions that use simple algorithms, such as keyword lists and hashing, which are too coarse to capture the features what makes sensitive documents secret" [53]. In contrast, a Risk Adaptive Access Control approach can provide the basis for a solution that not only makes it possible for critical document features to be captured, but also to relate these features to business processes, process-based context and risk levels.

Indeed, the Risk Adaptive Access Control approach can satisfy the requirements needed to cope with the risks of unauthorized access and dissemination of critical information and malicious insider threats in DEBSs due to the heterogeneous, loosely coupled and dynamic nature of these systems. Essentially, these requirements include: (i) the approach needs to take into account the event flow among heterogeneous distributed components; (ii) since DEBSs are loosely coupled, the security solutions need to work in settings where event subscribers are not known by event publishers; (iii) DEBSs are highly dynamic, and the approach cannot be restricted to a fixed set of security attributes and policies, but need to address dynamic context-aware scenarios and support dynamic and adaptive policies [54].

2.5 Multi-agent systems

Multi-agent systems refer to systems in which several agents interact with each other within a specific environment to solve a problem or to reach a goal that is beyond the individual capabilities or knowledge of each separate agent working on its own [54]. Multi-agent system models are used to represent the individual agents, their goals, beliefs, tasks, plans, as well as the possible interactions among different agents. Multi-agent systems provide several advantages over more traditional centralized methods in that they are well suited to large and complex distributed, dynamic and heterogeneous systems [56].

Multi-agent models can support a risk-based access control decision making process in DEBSs that is based on the prediction of risks and benefits. First, agent components can be used to support system goal compliance [57]. For example, system manageability can be a goal the system should

aim to reach by avoiding heavy loads and making sure that there are valid and accessible administrative accounts in place. Second, agent components can be used to monitor system components in order to simulate their behaviour. Third, agent components can be used to reason about, for example, which resources are needed to reach their goals [58]. Fourth, agent components can support autonomous adaptation in many ways, including policy adaptation [59].

One of the common approaches for multi-agent systems is based on the Belief-Desire-Intention (BDI) framework [59]. An extension of this approach that includes norms such as permissions and obligations is called the Norm-Belief-Desire-Intention (NBDI). According to this approach, agents "are able to reason about their motivations while taking into account the existence of social norms and to determine whether to follow a norm and receive its rewards, or violate it and receive the corresponding punishment" [60].

2.6 Prediction and Evaluation of Component-Related Processes

There are some approaches proposed in the literature that deal with component and user behaviour prediction [62]. These approaches monitor user or component actions and some of them use artificial intelligence techniques such as neural networks and Bayesian statistical models [63]. In one case [64], the proposed system uses short-term prediction to predict behaviour and guide system administrators. The prediction is short-term in the sense that only next actions are forecasted.

There are numerous approaches for evaluating the behaviour about users and components [58]. These approaches use methods such as hidden Markov models, decision trees, support vector machines and Bayesian networks [54]. Further, process mining and user profiling techniques can be used to gather user and component behaviour data to be used in behavioural analysis. When events are considered, process mining approaches can be used in the discovery and analysis of business processes based on raw event data [65,66]. In general, process mining is used to build the business processes that system users perform. In addition, the systems can use the knowledge about business processes to evaluate whether the business processes or the system goals need to be adjusted [65].

However, there is a lack of approaches for predicting the behaviour about users and components [67]. Although process mining techniques support behavioural data gathering and some forms of behavioural evaluation, these techniques are not currently used to make accurate predictions about tasks or processes the system should perform [67]. User profiling techniques have been used "In order to provide personalized assistance, personal agents rely on representations of user information interests and preferences contained in user profiles" [68], but not for behavioural prediction within an event-based setting [69].

Further, there is a lack of techniques that support goals compliance in which component-related processes need to be either evaluated or predicted with respect to the system goals. In some critical cases the behaviour of the system components need to be adjusted in order to become

aligned with the system goals or to avoid contradicting these goals. Further, there is a need of techniques to support evaluating to what degree the system is complying with the predefined systems goals. More complex situations even require the behavioural evaluations and predictions themselves to be adjusted depending on factors such as user profile and task-related context.

3 Problem Statement

There are many challenges in the area of risk management in DEBSs. These challenges lead to many unsolved problems related to the unauthorized access and dissemination of critical information and malicious insider threats.

First, most of the existing Access Control models (RBAC, MAC, ABAC, DAC) do not take the context of operations into account, which limits the data that is available for decision-making and makes the policies defined for such models rigid, not allowing variations in access control outcomes that depend on contextual details (e.g., location, time, task at hand) [70]. Note that introduction of contextual details leads to an access control decision making process that goes beyond the traditional access control methods based on a predefined set of rules and policies.

Second, most of the existing systems do not provide (de-)motivation for their users to improve their behaviour with respect to the risks and compliance with system goals. In other words, in terms of risks, current systems do not provide incentives (e.g., punishments or rewards) for users to be discouraged to access or not to access, for example, a sequence of critical documents, in case the access is increasingly dangerous. In this case, the system can provide access to some parts of the documents or deny access to all documents. In addition, in terms of goal compliance, current systems lack means to provide incentives for users, for example, to be encouraged to comply with task prioritization requirements. When the user tries to access high priority servers instead of low priority ones with low priority tasks, based on incentives, the system can warn the user that access to high priority servers could be blocked in the future if he or she proceeds. In the case of current reputation systems [71], reputation is considered in isolation and without analyzing its relationships with user tasks and context. Further, traditional access control systems focus on prohibitive (permission) actions, but do not consider the user goals or context, or provide feedback so that the user can remediate his or her problems.

Third, existing access control policies are static, meaning that the system is able to execute them, but is not able to track changes or reason about quantitative risk-related measures (e.g., reputation, reliability). Dynamic policies, on the other hand, would allow access control system to update rules for each individual access request. These dynamic updates can change both the policy attributes and the combination of policy components that should be applied in a specific context [23].

Fourth, current systems do not support reasoning involving tasks, plans, goals and context. However, in general, risk evaluation needs to take into account the specific system goals (e. g.,

manageability, reliability) and its operational context (e. g., specific accounts and configurations exist and are accessible). For example, given that an administrative account for the system is not accessible, a task to fix this problem, which may involve actions to restore the power to the server or to send an urgent message to specific technicians, would receive a higher priority [66].

Fifth, existing access control models are non-interactive. This means that systems based on such models provide only final access control decisions (i. e., "yes" or "no" answers), but there is no automated way to circumvent these decisions (i.e., by asking the system administrator for a temporary permission) [72].These circumventing operations are not supported by existing systems. For example, when launching an application that requires an access to a restricted port in a firewall the user has to ask a system administrator to open a port in the firewall, without any automated guidance that would, based on quantitative risk-related measures, context and user tasks, remediate the problem.

Last, existing access control approaches do not support satisfactorily the integration of heterogeneous systems. "Global companies are moving away from monolithic organizations towards a more dynamic business ecosystem that is reflected in the distributed nature of their software infrastructure" [73]. Currently, given that the architect has to combine two systems with different access control approaches, there can be two choices. The first choice is to create a new access control system that would work for both of systems, which is time and resource consuming. The second one is to create a "proxy" which would act in one system on behalf of the other and vice versa, but this option limits the input information that is available to support access control decisions in both systems.

Providing methods to deal with unauthorized access and dissemination of critical and malicious insider threats can result in numerous benefits. The risk of granting an unauthorized access would be lowered, since the risk associated with users operations is evaluated and the context or situation where access is requested is analyzed. The risk of dissemination of critical information would be lowered as well, since the process or sequence of operations that users perform is also analyzed [74]. This analysis would include reasoning about user operations and matching the context of performed operations with patterns of successful and failed processes previously performed [75]. Risks related to malicious intruders would be lowered since user actions are analysed in terms of their risks levels and the policy attributes and the policies themselves change dynamically, it is very difficult for an intruder to formulate simple and efficient strategies against the system.

To briefly summarize, the listed problems:(i) increase the risks related to information access (i.e., by allowing information leaks or an unauthorized access) and do not take the context of operations into account;(ii) limit user motivation by providing only prohibitive or permissive actions without giving rewards for actions that are useful or actual punishments for actions that are harmful to the system;(iii) lack support to track changes and reason about risk-related attributes; (iv) lack support to reason about tasks, plans and goals;(v) lack process-based system guidance by providing

only final access control decisions (i. e., "yes" or "no" answers), and increase the costs per operation for growing systems by making human intervention mandatory to manually update access control policies; and(vi) limit security provided by access control when heterogeneous systems are integrated.

4 Proposed Approach

In this section, we describe a risk management approach for DEBSs that involves adaptive access control and agent-based threat assessment (e. g, leak detection, malicious insider attacks). The approach is shown in Figure 3.

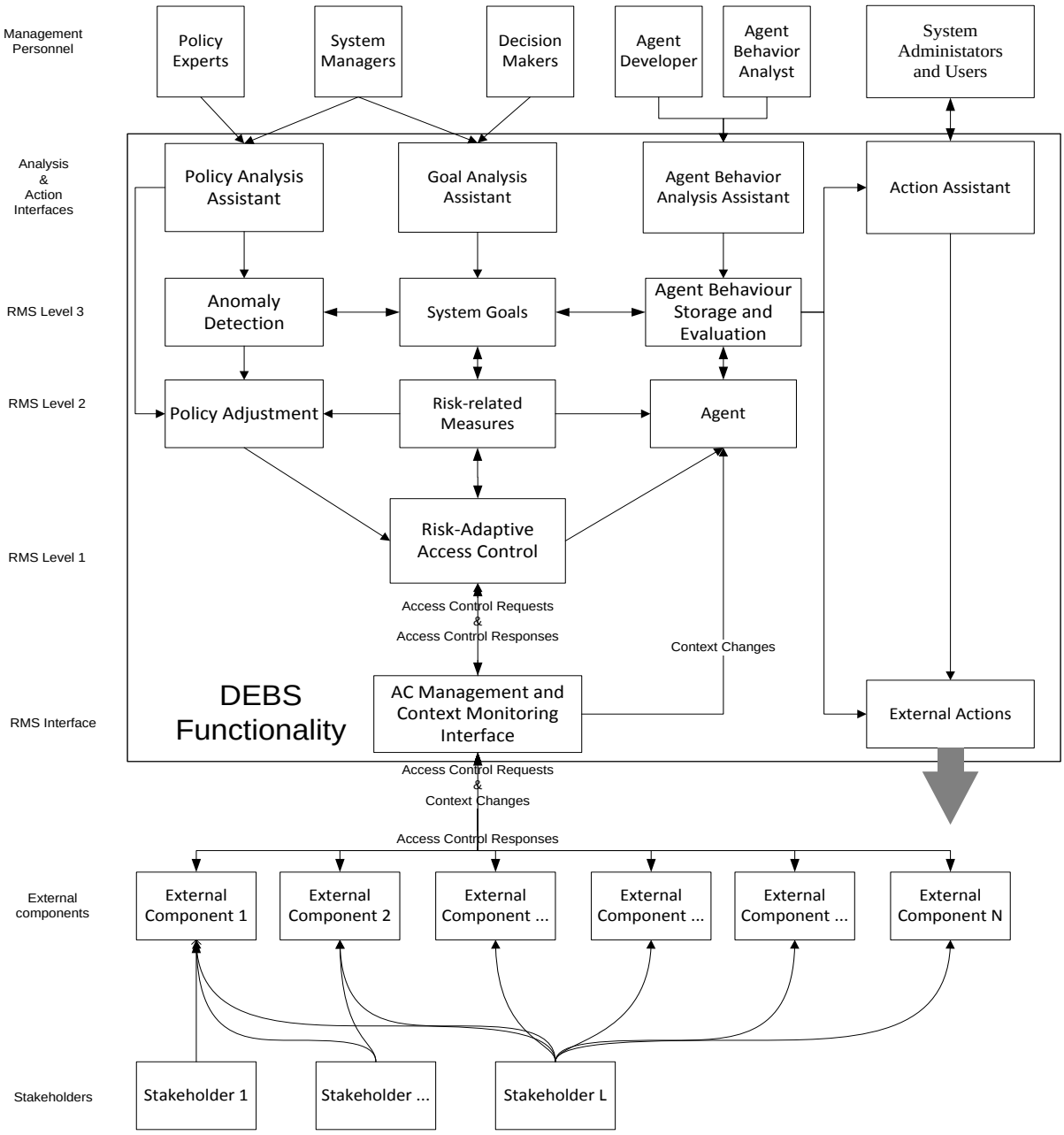

Figure 3. Risk Management approach for DEBS Risk Management: architectural model.

The approach assumes that a number of stakeholders (e. g., clients, developers) interact with external components (e. g., email, firewall, build servers) and the risks related to their actions and operations should be managed. As an interface to the external components, the *AC Management and Context Monitoring Interface* component is used to interact with the DEBS system. This component manages the access control for all of the external components and monitors contextual changes, such as changes in location and time [76]. Each event related to user tasks and user contextual changes need to have the specific user identification as a parameter.

To enable interactions between the system and the management personnel (e. g., policy experts, decision makers, system administrators), the approach introduces some analysis and action assistant interfaces. A *Policy Analysis Assistant* provides an interface for managing and changing policies and their attributes directly (e. g., adding or removing new policies), and altering the *Anomaly Detection* algorithms used to monitor the system event sequences [77] and manage policies. This component is designed to be used by the *Policies Experts* and the *System Managers, who* propose policy changes and authorize these changes, respectively. A *Goal Analysis Assistant* provides an interface for updating, creating, deleting and prioritizing goals in a *System Goal* component, as well for altering the algorithms used to manage and evaluate the system goals [78]. The third interface is an *Agent Behaviour Analysis Assistant,* which provides the ability for the *Agent Developers* to define new agents and to update the *Agent Behaviour Storage and Evaluation* pattern detection and behaviour evaluation algorithms. It also provides the ability for the *Agent Behaviour Analysts* to analyze and evaluate current and previous agent states and to modify these states. The fourth interface on this level is an *Action Assistant*, which provides an interface for the manual execution of External Actions.

The *External Components* interact with the RMS through the *AC Management and Context Monitoring Interface*. This interface receives Access Control Requests from and sends Access Control Responses to the *External Components.* It also monitors contextual changes. Further, this component transforms Access Control Requests into DEBS events and sends them to a *Risk-Adaptive Access Control* component, and receives Access Control Reponses in the form of DEBS events from the *Risk-Adaptive Access Control* component and transforms DEBS events into responses that can be interpreted by the external components. Contextual changes are transformed into events as well and sent to an *Agent* so that this agent's beliefs are updated. This update takes into account contextual changes in the specific component environment the agent is monitoring.

4.1 A Three-Level Model

The approach is based on three levels. In the first level, a *Risk-Adaptive Access Control* component, which interacts with the *AC Management and Context Monitoring Interface* component, is introduced. The risk-adaptive component provides access control decisions for access control requests based on risk and benefit calculations that take into account quantitative risk-related measures and contextual attributes [79,80]. The outcomes of the access control decisions are not

"yes" or "no" answers, but are processes that represent the tasks components have to perform in order to be permitted to access data or perform an action.

The resulting processes are provided based on a Risk Function, which in general can be represented as follows:

RF : {resource, component, qrm(component), risk(<action, resource, component>), benefit(<action, resource, component>),context} → result

where

qrm : component → < ($type_1$,$value_1$), ($type_2$,$value_2$), ... , ($type_m$, value $_m$)>

risk : <action, resource, component> → <<($type_{risk\ 1\text{-}1}$,$value_{risk\ 1\text{-}1}$), ... , ($type_{risk\ 1\text{-}u}$, $value_{risk\ 1\text{-}u}$)>, ... , <($type_{risk\ h\text{-}1}$,$value_{risk\ h\text{-}1}$), ... , ($type_{risk\ h\text{-}y}$, $value_{risk\ h\text{-}y}$)>>

benefit : <action, resource, component> → <<($type_{benefit\ 1\text{-}1}$,$value_{benefit\ 1\text{-}1}$), ... , ($type_{benefit\ 1\text{-}v}$, $value_{benefit\ 1\text{-}v}$)> , ... , <($type_{benefit\ g\text{-}1}$,$value_{benefit\ g\text{-}1}$), ... , ($type_{benefit\ g\text{-}z}$, $value_{benefit\ g\text{-}z}$)>>

context : << action, resource, component>,<st_1, ... , st_n>,<ct_1, ... , ct_s>> → bool

The RF function receives as input the resource to be accessed, the component that is requesting access to the resource, the quantitative risk-related measures (qrm) associated with the component (sequence of types and values), the risk associated with action, resource and component (sequence of types and values for each type of risk), the benefit associated with action, resource and component (sequence of types and values for each type of benefit), and the context associated with action, resource and component, a set of system state values, and a set of context values (a boolean, that is true or false).

The result of the function RF is a sequence of processes, namely

<{$process_1$, resource_form $_1$}, {$process_2$, resource_form $_2$}, ... , {$process_e$, resource_form $_e$}> where

$process_l$ = <$task_1$, $task_2$, ... , $task_d$>, where $l \in \{1, 2, \ldots, e\}$

resource_form is a transformed resource (e. g., obfuscated code, documentation)

Each of these processes consists of a sequence of tasks that could be performed to deal with the risks related to a specific situation.

Each of these processes consists of a sequence of tasks that could be performed to deal with the risks related to a specific situation.

For example, in case RF is applied to a scenario in which a developer is trying to access source code (i. e., resource) through a code-versioning system (i.e., component). We assume in that case that there is only one quantitative risk-related measure, namely trust (i. e., $type_1$=trust level, $value_1$=9). The developer has a high level of trust (i. e., 9 in 10). The action we are considering in this case is to access a source code. The risk can be related to a possible dissemination of the source code (i. e., $type_{risk\ 1\text{-}1}$=risk of the source code dissemination, $value_{risk\ 1\text{-}1}$=0.1). The benefit can be related to decreasing the chance of project delay (i. e., $type_{benefit\ 1\text{-}1}$=benefit of decreasing the chance of project delay, $value_{benefit\ 1\text{-}1}$=0.01). The context can involve a system state type called alert level and a context type called location. We assume the alert level is high and that the developer is working on site.

Taking into account the input we have described, RF evaluates risks against benefits. In this case risks are higher than benefits and, as a result, RF will produce three possible processes to reduce the risks or increase the benefits. The first process consists of two tasks: to ask the user to authenticate himself or herself with the system in an additional way (e. g., by swiping his or her smart card), and to close all other applications. If this process is successfully completed, the user will have the ability to get a full version of the source code file. The second process consists of one task, that is, to get permission from the manager to access the full version of the source code file. When this process is completed, the user will have the ability to get a full version of the source code file. A third process can be defined as empty, meaning that there is no process to be performed. As a result, the user can get access only to the source code documentation.

In the second level, three components are introduced. A *Risk-related Measures* component, which provides a set of numerical attributes (quantitative risk-related measures) to the *Risk-Adaptive Access Control* component for risk and benefit calculations. The numerical attributes relate to contextual information, such as the context of a task or the context of a component. This component also provides methods to obtain certain types and their associated values of quantitative risk-related measures required by policies that an action to be performed needs to be subject to. These types and values of quantitative risk-related measures are acquired through trading them with other types of quantitative risk-related measures that the context of this task or component has. The values of quantitative risk-related measures can be altered by a *Policy Adjustment* component. This component adjusts the values of quantitative risk-related measures of the policies based on the supply, demand and accumulated levels of component quantitative risk-related measures, the exchange rate among quantitative risk-related measures types that are interchangeable, and the frequency of usage of each policy. In addition, this level also contains an *Agent Component,* which monitors all the events associated with a component or task context and builds the sequences of tasks a component has performed. These task sequences are built using process mining and user profiling techniques which gather information about users and components and stored as trees. The *Agent Component* is also used to reason about which types and values of quantitative risk-related measures can be traded. In general, the second level uses an adaptive access control mechanism

defined at level one to support policy adaptation based on agent-mediated quantitative risk-related measures trading.

In the third level, there are also three components. A *System Goal* component is responsible for monitoring the effects of system states and task processes on goal compliance, and managing and evaluating the system goals. An *Agent Behaviour Storage and Evaluation* component evaluates the compliance of agent task processes with system goals. This component also identifies and stores recurring patterns of component's behaviour [78]. It is also responsible for defining external actions, which consist of task sequences used to help components to remediate risky situations and progress towards their goals. The third component in this level is the *Anomaly Detection*. This component monitors the system events to find desirable and undesirable patterns of task sequences [80,81] that could cause the system either to comply or not with system goals. It is also used to manage and change the policies and their attributes. This level relies on the second level to support the detection of risky anomalies and the compliance with system goals [82]. Many types of anomalies can be detected using the approach, including unauthorized access and dissemination of information (e. g., leaks) and malicious insider threats.

4.2 Incremental Development

Even though the proposed approach requires processing enormous amounts of data and involves a large number of components and their intricate and complex event-based interactions, the approach can be implemented gradually. The *Risk-Adaptive Access Control* and *R*isk-related Measures components can use existing access control policies translated into a risk-adaptive form, while preserving the behaviour of the associated access control system. The traditional access control models only provide a "yes" or "no" answer, which can be represented in our model when Risk Function (RF) is evaluated and provide as a result an empty process and either a resource form (which corresponds to "yes") or "empty" as a resource form (which corresponds to "no").

When the *Policy Adjustment* component is included and works with the *Risk-Adaptive Access Control* and *Risk-related Measures* components, a dynamic, self-adjusting, flexible access control mechanism can be supported, which can adjust automatically or semi-automatically risk-related policy attributes according to predefined rules. For example if there was an accident with a policy involving an attribute such as the trust required to gain access to a certain document, the amount of the required trust would increase to a predefined value based on risk-related rules. If there were no accidents during a certain period of time, then this threshold would be lowered to a predefined initial value.

The proposed gradual development will include the following steps:

(i) in the first level, a risk-adaptive mechanism based on quantitative risk-related measures and contextual attributes is provided; (ii) in the second level policies and policy attributes are dynamically adjusted and agents are used to monitor task-related events, build component task sequences, and

reasons about which types and values of quantitative risk-related measures can be traded; (iii) in the third level, the system monitors the effects of system states and task processes on goal compliance, manages and evaluates the system goals, identifies and stores recurring patterns of component's behaviour, defines external actions, (i. e., task sequences) used to help components to remediate risky situations and progress towards their goals, and supports the detection of risky anomalies and the compliance with system goals.

5 Evaluation

The proposed approach can be evaluated using various techniques:

(i) Comparing the approach with alternative methods in terms of software requirements;

(ii) Assessing the improvement of the expressiveness of the design and architectural (meta-) models by considering, for example, the introduction of novel abstractions;

(iii) Validating whether the models and applications satisfy critical requirement and design properties (e. g., dynamic, self-adjusting properties);

(iv) Conducting case studies (e. g., software development scenario);

(v) Performing experimental validation (e. g., risk and benefit quantitative analysis).

6 Conclusion and Future Work

In this paper, a novel risk management approach for DEBSs based on the adaptive access control and agent-based threat assessment (e. g, leak detection, malicious insider attacks) was proposed. This approach decreases the risks related to information access (i.e., by disallowing information leaks or an unauthorized access) and takes the context of operations into account. The approach also improves user motivation by employing rewards for actions that are useful to the system or actual punishments for actions that are harmful to the system. It also provides support to track changes and reason about risk-related attributes, tasks, plans and goals. The approach also provides process-based system guidance by supporting interactive access control decisions based on processes used to minimize the risks, decreases the costs per operation for growing systems by making human intervention optional, and provides support for the integration of access control mechanisms of different heterogeneous systems.

The approach is based on three levels: (i) in the first level, a risk-adaptive mechanism based on quantitative risk-related measures and contextual attributes is provided; (ii) in the second level policies and policy attributes are dynamically adjusted and agents are used to monitor task-related events, build component task sequences, and to reason about which types and values of quantitative risk-related measures can be traded; (iii) in the third level, the system monitors the effects of system states and task processes on goal compliance, manages and evaluates the system goals, identifies

and stores recurring patterns of component's behaviour, defines external actions, (i. e., task sequences) used to help components to remediate risky situations and progress towards their goals, and supports the detection of risky anomalies and the compliance with system goals.

By providing a risk management approach for DEBSs that involves adaptive access control and agent-based threat assessment, we believe the state of the art is advanced in terms of solutions to problems related to unauthorized access and dissemination of critical information and malicious insider threats.

7 References

1. Jayaram, K. R., Eugster P., and Jayalath, C., Parametric Content-Based Publish/Subscribe, ACM Transactions on Computer Systems (TOCS), volume 31, number 2, article 4, 2013.
2. Rathfelder, C., Kounev, S., and Evans D., Capacity Planning for Event-Based Systems Using Automated Performance Predictions, Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering, pages 352-361, 2011.
3. Sun, Y., Ninghui L., and Elisa B., Proactive Defense of Insider Threats through Authorization Management, Proceedings of 2011 international workshop on Ubiquitous affective awareness and intelligent interaction, pages 9-16, 2011.
4. WikiLeaks, Secret US Embassy Cables. Retrieved on August 15, 2013, from http://wikileaks.org/cablegate.html
5. Gross, D., Facebook Apps Leaked Users' Information, Security Firm Says. Retrieved on August 15, 2013, from http://www.cnn.com/2011/TECH/social.media/05/11/facebook.information.leak/index.html
6. PolicyMic, Top 5 Stratfor Wikileaks Revelations (So Far). Retrieved on August 15, 2013, from http://www.policymic.com/articles/4833/top-5-stratfor-wikileaks-revelations-so-far
7. Gerstein, J., DOD Sees U.S. 'Mistake' & 'Failure' in Snowden Leaks. Retrieved on August 15, 2013, from http://www.politico.com/blogs/under-the-radar/2013/07/dod-sees-us-mistake-failure-in-snowden-leaks-168695.html
8. Infowatch, Global Data Leakage Report 2011. Retrieved on August 15, 2013, from http://infowatch.com/analytics/reports/2376
9. Heilman, M., and Glyer, C., South Carolina Department of Revenue Public Incident Response Report. Retrieved on August 18, 2013, from http://governor.sc.gov/Documents/MANDIANT%20Public%20IR%20Report%20-%20Department%20of%20Revenue%20-%2011%2020%202012.pdf
10. Swissinfo.ch and agencies, Germans Buy Another Tax CD. Retrieved on August 18, 2013, from http://www.swissinfo.ch/eng/business/Germans_buy_another_tax_CD.html?cid=35526516
11. Kamra, A., and Bertino, E., Privilege States Based Access Control for Fine-Grained Intrusion Response, Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, volume 6307, pages 402-421, 2010.
12. Migliavacca, M., Papagiannis, I., Eyers, D. M., Shand, B., Bacon, J., and Pietzuch, P., Distributed Middleware Enforcement of Event Flow Security Policy, Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware, pages 334-354, 2010.
13. Kim, M., Jeong, H., and Choi. E., Context-Aware Platform for User Authentication in Cloud Database Computing, International Conference on Future Information Technology and Management Science & Engineering Lecture Notes in Information Technology, volume 14, pages 170-176, 2012.

14. Blanco, R. M., Process Models for Distributed Event-Based Systems, PhD Thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2010. Retrieved from http://hdl.handle.net/10012/5047.
15. Singh, J., Eyers, D. M., and Bacon, J., Disclosure Control in Multi-Domain Publish/Subscribe Systems, Proceedings of the 5th ACM international conference on Distributed Event-Based System, pages 159-170, 2011.
16. Blanco, R. M., Wang, J., and Alencar P., A Metamodel for Distributed Event Based Systems, Proceedings of the second international conference on Distributed Event-Based Systems, pages 221-232, 2008.
17. Srivatsa, M., Ling L., and Iyengar, A., EventGuard: A System Architecture for Securing Publish-Subscribe Networks, ACM Transactions on Computer Systems, volume 29, issue 4, article 10, 2011.
18. Fotiou, N., Marias, G.F., and Polyzos, G.C., Towards a Secure Rendezvous Network for Future Publish/Subscribe Architectures, ACM Transactions on Computer Systems (TOCS), volume 29, issue 4, pages 49-56, 2010.
19. Hahn, A., Kregel, B., Govindarasu, M., Fitzpatrick , F., Adnan, R., Sridhar, S., and Higdon, M., Development of the Powercyber SCADA Security Testbed, Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, article 21, 2010.
20. Hinze, A., and Buchmann, A. P., Principles and Applications of Distributed Event-Based Systems, Information Science Reference, chapter 12, pages 284-306, 2010.
21. Singh, A., Role Based Trust Management Security Policy Analysis, International Journal of Engineering Research and Applications (IJERA), volume 2, issue 6, pages 126-155, 2012.
22. Toninelli, A., Corradi, A., and Montanari, R., A Quality of Context-Aware Approach to Access Control in Pervasive Environments, MobileWireless Middleware, Operating Systems, and Applications, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, volume 7, pages 236-251, 2009.
23. Russo, A., and Sabelfeld, A., Dynamic vs. Static Flow-Sensitive Security Analysis, 2010 23rd IEEE Computer Security Foundations Symposium (CSF), pages 186 - 199, 2010.
24. Tariq, M. A., Koldehofe, B., Altaweel, A., and Rothermel, K., Providing Basic Security Mechanisms in Broker-Less Publish/Subscribe Systems, Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, pages 38-49, 2010.
25. Suhendra, V., A Survey on Access Control Deployment, Security Technology Communications in Computer and Information Science, volume 259, pages 11-20, 2011.
26. Abercrombie, R., Sheldon, F., Aldridge, H., Duren, M., Ricci, T., Bertino, E., Kulatunga, A., and Navaratne, U., Secure Cryptographic Key Management System (CKMS) Considerations for Smart Grid Devices, Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, article 59, 2011.
27. Li, J., Chen, X., Li ,J., Jia, C., Ma, J., and Lou, W., Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption, Computer Security–ESORICS Lecture Notes in Computer Science, volume 8134, pages 592-609, 2013.

28. Vijayakumar, H., Jakka, G., Rueda, S., Schiffman, J., and Jaeger, T., Integrity Walls: Finding Attack Surfaces from Mandatory Access Control Policies, Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pages 75-76, 2012.
29. Jin, X., Krishnan, R., and Sandhu, R., A Unified Attribute-Based Access Control Model Covering DAC, MAC And RBAC, Data and Applications Security and Privacy XXVI, Lecture Notes in Computer Science, volume 7371, pages 41-55,2012.
30. Shafiq, B., Vaidya, J. S., Ghafoor, A., and Bertino, E., A Framework for Verification and Optimal Reconfiguration of Event-Driven Role Based Access Control Policies, Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pages 197-208, 2012.
31. Komlenovic, M., Tripunitara, M., and Zitouni, T., An Empirical Assessment of Approaches to Distributed Enforcement in Role-Based Access Control (RBAC), Proceedings of the First ACM Conference on Data and Application Security and Privacy, pages 121-132, 2011.
32. Tripunitara, M. V., and Carbunar, B., Efficient Access Enforcement in Distributed Role-Based Access Control (RBAC) Deployments, Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, pages 155-164, 2009.
33. Kirkpatrick, M. S., Damiani, M. L., and Bertino, E., Prox-RBAC: a Proximity-Based Spatially Aware RBAC, Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 339-348, 2011.
34. Fuchs, L., Pernul, G., and Sandhu, R., Roles in Information Security–a Survey and Classification of the Research Area, Computers & Security, volume 30, issue 8, pages 748-769, 2011.
35. Hu, S., Muthusamy, V., Li, G., and Jacobsen, H. A., Transactional Mobility in Distributed Content-Based Publish/Subscribe Systems, Distributed Computing Systems, ICDCS'09, 29th IEEE International Conference on, pages 101 - 110, 2009.
36. Bertino, E., and Kirkpatrick, M. S., Location-Based Access Control Systems for Mobile Users: Concepts and Research Directions, Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS, pages 49-52, 2011.
37. Marinovic, S., Craven, R., Ma, J., and Dulay, N., Rumpole: a Flexible Break-Glass Access Control Model, Proceedings of the 16th ACM Symposium on Access Control Models and Technologies, pages 73-82, 2011.
38. Park, J., Sandhu, R., and Cheng Y., A User-Activity-Centric Framework for Access Control in Online Social Networks, Internet Computing, volume 15, issue 5, pages 62-65, 2011.
39. Karp, A. H., Haury, H., and Davis M. H., From ABAC to ZBAC: the Evolution of Access Control Models, Technical Report HPL-2009-30, Hewlett-Packard Laboratories, pages 1-21, 2009.
40. Huang, J., Nicol, D. M., Bobba, R., and Huh, J. H., A Framework Integrating Attribute-Based Policies into Role-Based Access Control, Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pages 187-196, 2012.
41. Schefer-Wenzl, S., and Strembeck, M., Generic Support for RBAC Break-Glass Policies in Process-Aware Information Systems, Proceedings of the 28th Annual ACM Symposium on Applied Computing, pages 1441-1446, 2013.

42. Shafiq, B., Vaidya, J. S., Ghafoor, A., and Bertino, E., A Framework for Verification and Optimal Reconfiguration of Event-Driven Role Based Access Control Policies, Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, pages 197-208, 2012.
43. Lagutin, D., Visala, K., Zahemszky, A., Burbridge, T., and Marias, G. F., Roles and Security in a Publish/Subscribe Network Architecture, IEEE Symposium on Computers and Communications (ISCC), pages 68-74, 2010.
44. Alalfi, M. H., Cordy, J.R., and Dean, T. R., Automated Verification of Role-Based Access Control Security Models Recovered from Dynamic Web Applications, 14th IEEE International Symposium on Web Systems Evolution (WSE), pages 1-10, 2012.
45. Ullah, S., Xuefeng, Z., and Feng, Z., TCloud: A Dynamic Framework and Policies for Access Control across Multiple Domains in Cloud Computing, International Journal of Computer Applications, volume 62, number 2, pages 1-7, 2013.
46. Crampton, J., Practical and Efficient Cryptographic Enforcement of Interval-Based Access Control Policies, ACM Transactions on Information and System Security (TISSEC), volume 14, issue 1, article 14, 2011.
47. Molloy, I., Dickens, L., Morisset, C., Cheng, P. C., Lobo, J., and Russo, A., Risk-Based Security Decisions under Uncertainty, Proceedings of the Second ACM Conference on Data and Application Security and Privacy, pages 157-168, 2012.
48. Allen, L., and Bali, T. G., Cyclicality in Catastrophic and Operational Risk Measurements, Journal of Banking & Finance, volume 31, issue 4, pages 1191–1235 , 2007.
49. Molloy, I, Cheng, P., and Rohatgi, P., Trading in Risk: Using Markets to Improve Access Control, Proceedings of the 2008 Workshop on New Security Paradigms, pages 107-125, 2009.
50. Allen, L., and Saunders, A., Incorporating Systemic Influences into Risk Measurements: A Survey of the Literature, Journal of Financial Services Research, volume 26, issue 2, pages 161-191, 2004.
51. Han, W., Shen, C., Yin, Y., Gu, Y., and Chen, C., Poster: Using Quantified Risk and Benefit to Strengthen the Security of Information Sharing, Proceedings of the 18th ACM Conference on Computer and Communications Security, pages 781-784, 2011.
52. Backes, M., Kopf B., and Rybalchenko, A., Automatic Discovery and Quantification of Information Leaks, 30th IEEE Symposium on Security and Privacy, pages 141 - 153, 2009.
53. Hart, M., Manadhata, P., and Johnson, R., Text Classification for Data Loss Prevention, Proceedings of the 11th international conference on Privacy enhancing technologies (PETS), pages 18-37, 2011.
54. Cugola, G., Margara, A., and Migliavacca, M., Context-Aware Publish-Subscribe: Model, Implementation, IEEE Symposium on Computers and Communications (ISCC), pages 875 - 881, 2009.
55. Horling, B., and Lesser, V., A Survey of Multi-Agent Organizational Paradigms, The Knowledge Engineering Review, volume 19, issue 4, pages 281-316, 2004.
56. Stone, P., and Veloso M., Multiagent Systems: A Survey from a Machine Learning Perspective, Autonomous Robots, volume 8, issue 3, pages 345-383, 2000.

57. Chopra, A. K., Dalpiaz, F., Giorgini, P., and Mylopoulos, J., Reasoning about Agents and Protocols via Goals and Commitments, Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, volume 1, pages 457-464, 2010.
58. Jung, Y., Kim, M., Masoumzadeh, A., and Joshi, J. B., A Survey of Security Issue in Multi-Agent Systems, Artificial Intelligence Review, volume 37, issue 3, pages 239-260, 2012.
59. Jadbabaie, A., Lin, J., and Morse, A. S., Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules, IEEE Transactions on Automatic Control, volume 48, issue 6, pages 988-1001, 2003.
60. Wooldridge, M., and Jennings, N. R., Agent Theories, Architectures, and Languages: A Survey, Intelligent Agents, pages 1-39,1995.
61. Neto, D. S., Baldoino F., Silva, V. T., Lucena, C. J. P., An Architectural Model for Autonomous Normative Agents, Advances in Artificial Intelligence (SBIA), pages 152-161, 2012.
62. Jumadinova, J., and Dasgupta, P., A Multi-Agent Prediction Market Based on Partially Observable Stochastic Game, Proceedings of the 13th International Conference on Electronic Commerce, article 21, 2011.
63. Martinetz, T. M., Berkovich, S. G., and Schulten, K. J., Neural-Gas' Network for Vector Quantization and its Application to Time-Series Prediction, IEEE Transactions on Neural Networks, volume 4, issue 4, pages 558-569, 1993.
64. Pikoulas, J., Buchanan, W., Mannion, M., and Triantafyllopoulos, K., An Intelligent Agent Security Intrusion System, Proceedings Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pages 94 - 99, 2002.
65. Aalst, V. W., and Dustdar, S., Process Mining Put into Context, IEEE Internet Computing, volume 16, issue 1, pages 82-86, 2012.
66. Aalst, V. W., Process Mining: Overview and Opportunities, ACM Transactions on Management Information Systems (TMIS), volume 3, issue 2, article, 2012 .
67. Cao, L., Weiss, G., and Philip, S.Y., A Brief Introduction to Agent Mining, Autonomous Agents and Multi-Agent Systems, volume 25, issue 3, pages 419-424, 2012.
68. Godoy, D., and Amandi, A., User Profiling in Personal Information Agents: A Survey, The Knowledge Engineering Review, volume 20, issue 4, pages 329-361, 2005.
69. Zhang, D., Ramamohanarao, K., Ebringer, T., and Bailey, J., Data Mining for Role Based Access Control, Technical Report, Department of Computer Science and Software Engineering, The University of Melbourne, pages 1-65, 2010.
70. Bertino, E., and Kirkpatrick, M. S., Location-Based Access Control Systems for Mobile Users: Concepts and Research Directions, Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS, pages 49-52, 2011.
71. Jøsang, A., Ismail, R., and Boyd, C., A Survey of Trust and Reputation Systems for Online Service Provision, Decision Support Systems, volume 43, issue 2, pages 618-644, 2007.
72. Han, D. J., Zhuo, H. K., Xia, L. T., and Li, L., Permission and Role Automatic Assigning of User in Role-Based Access Control, Journal of Central South University, volume 19, issue 4, pages 1049-1056, 2012.

73. Adam, N., Stiles, R., Zimdars, A., Timmons, R., Leung, J., Stachnick, G., Merrick, J., Coop, R., Slavin, V., Kruglikov, T., Galmiche, J., and Mehrotra, S., Consequence Analysis of Complex Events on Critical US Infrastructure, Communications of the ACM, volume 56, issue 6, pages 83-91, 2013.
74. Rabin, M., Risk Aversion and Expected-Utility Theory, Econometrica, volume 68, issue 5, pages 1281-1292, 2000.
75. Grimm, V., Revilla, E., Berger, U., Jeltsch, F., Mooij, W. M., Railsback, S. F., Thulke, H., Weiner, J., Wiegand, T., and DeAngelis D., Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology, Science, volume 310, number 5750, pages 987-991, 2005.
76. Patiniotakis, I., Papageorgiou, N., Verginadis, Y., Apostolou, D., and Mentzas, G., Dynamic Event Subscriptions in Distributed Event Based Architectures, Expert Systems with Applications, volume 40, issue 6, pages 1935-1946, 2012.
77. Bauer, L., Garriss, S., and Reiter, M. K., Detecting and Resolving Policy Misconfigurations in Access-Control Systems, ACM Transactions on Information and System Security (TISSEC), volume 14, issue 1 article 2, 2011.
78. Ketter, W., Collins, J., Gini, M., Gupta, A., and Schrater, P., Detecting and Forecasting Economic Regimes in Multi-Agent Automated Exchanges, Decision Support Systems, volume 47, issue, 4, pages 307-318, 2009.
79. Modi, S., Design a Fine Grain Role-Based Access Control Framework for Cloud Computing, MSc Thesis, Computer Science Department, University of Colorado at Denver, pages 1-59, 2012.
80. Fall, D., Blanc, G., Okuda, T., Kadobayashi, Y., and Yamaguchi, S., Toward Quantified Risk-Adaptive Access Control for Multi-tenant Cloud Computing, The 6th Joint Workshop on Information Security, pages 1-13, 2011.
81. Aalst, V. W., Schonenberg, M. H., and Song, M., Time Prediction Based on Process Mining, Information Systems, volume 36, issue 2, pages 450-475, 2011.
82. Chandola, V., Arindam B., and Kumar, V., Anomaly Detection: A Survey, ACM Computing Surveys (CSUR), volume 41, issue 3, article 15, 2009.

Appendix 1: An Illustrative Application of the Approach

We illustrate the applicability of proposed risk management approach using an example involving a software development company. This company is developing a project that is currently in its maintenance phase and developers are dealing with bug requests submitted by users. The project consists of several modules, including network, storage and threads components. The approach application is shown in Figure 4. This figure is an instantiation of Figure 3 to the current application domain.

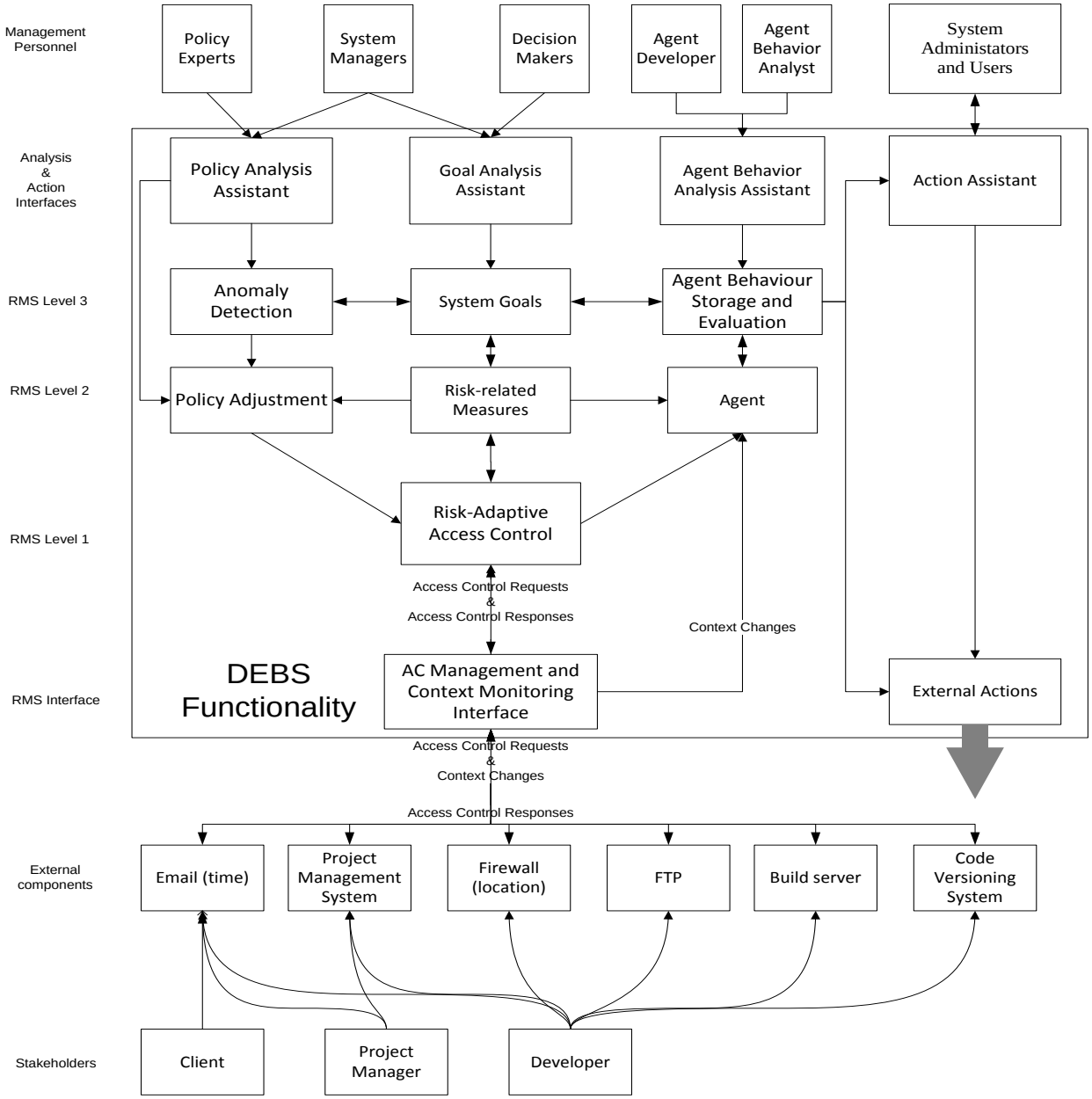

Figure 4. Architecture of RMS with external components and stakeholders

The Stakeholders involved in the project include:

(i) A software developer, who specializes on threads, is the main actor for this case;

(ii) A client that has submitted a request for fixing a bug and that supplies the development team with additional information;

(iii) A second developer is working with the first developer;

(iv) A development manager of development team deals with the administrative aspects of the project.

The main steps related to this case study are described next:

S1. The first developer checks her email messages and answers some them. One of these messages indicate that a bug was found in system and that this bug has a critical priority. The bug is initially categorized as thread-related.
S2. The developer tries to reproduce this bug in the test environment, but is unable to do so.
S3. The developer contacts customer and gets data that can help her to debug system.
S4. She gets a log of the customer activity, but is still unable to reproduce the error. However the debug data suggests that there is a common error in one of the thread modules.
S5. The developer reads the code of that module and finds out that a method belonging to the storage module was called with the wrong attributes.
S6. She reads the documentation for the storage module method and realizes that the method contract was changed and the storage method is outdated. However the required changes are minor.
S7. She makes the necessary change in the module and sends the module to the build server.
S8. The build server is rejecting the change, since another developer is working on the same file.
S9. To confirm that the problem has been solved, she builds a new version locally and sends it to the customer.
S10. The customer replies that the software is still displaying an error message. However, the error that is displayed is different from the previous one. She is familiar with the new error because she was responsible for fixing that kind of error many times in the past.
S11. The workday is almost over and she is planning to go on a vacation on the next day. Therefore, she sends a message to the project manager asking what she should do.
S12. The project manager asks her to stay overtime and fix this bug.
S13. She tries again to reproduce the bug and this time she is successful.
S14. She tries to fix the new bug and the solution seems to work.
S15. She tries to apply changes to the build server. However, it is on maintenance, and she cannot submit code to the repository.

S16. She sends a letter to another developer asking him to apply the code changes, and then leaves the office.

S17. On the next day, the second developer applies the changes and builds the project.

S18. The build server fails to build the project and answers that the regression tests have failed.

S19. The second developer sends a message to the first developer describing the situation.

S20. The first developer has already left the country, but she tries to fix the error remotely.

S21. She requests the source code to be downloaded to her computer.

S22. She makes the necessary changes and updates the code in the repository.

S23. The bug is successfully fixed.

In Figure 5 we illustrate the interaction among participating entities in the case of a bug resolution using a sequence diagram.

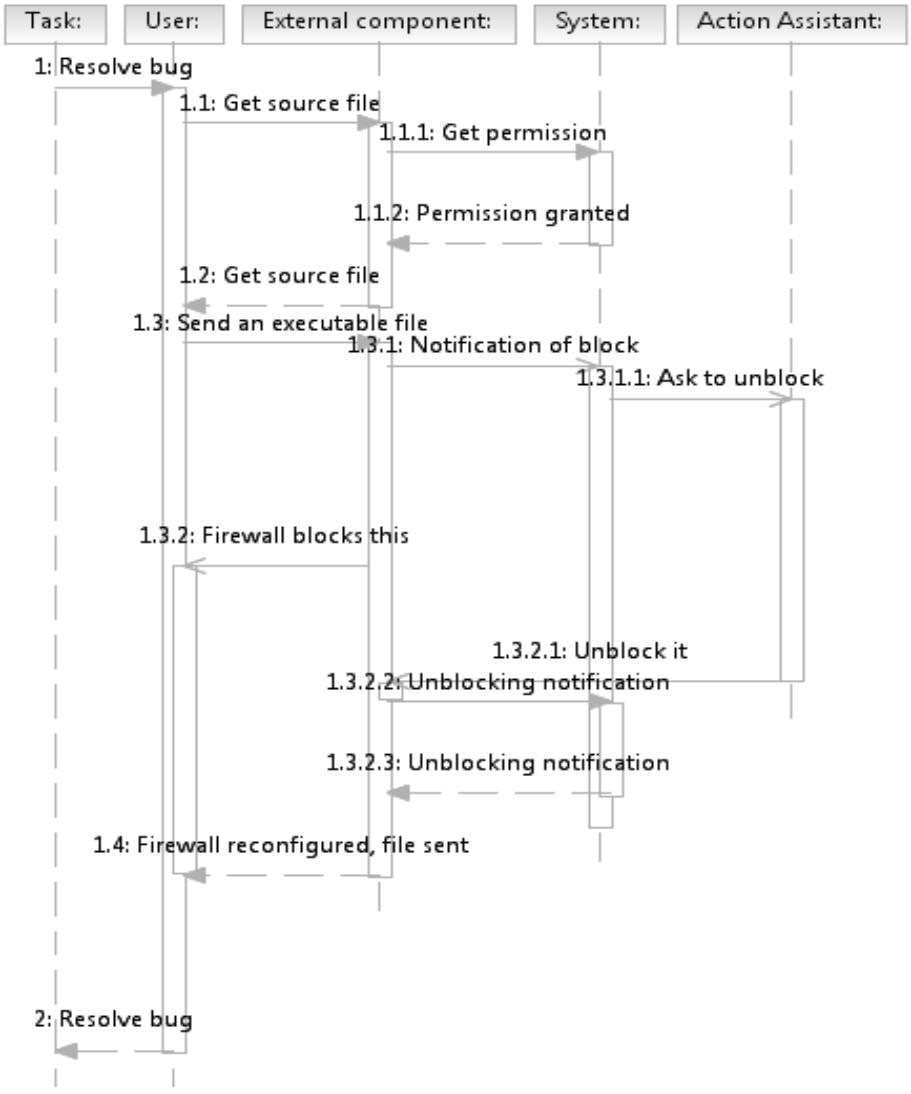

Figure 5. Sequence diagram of partially automated process

According to this diagram, a User receives a task to resolve a certain bug. Then he needs to get a part of source code that is related to this bug. External component responsible for delivering a source code asks System permission to do so, permission was granted. After updating a source file User needs to send an executable file, Firewall blocks this. External component notifies System of that event through error code. System, basing on context of task and reputation of user, estimates that this step is critical for project development and decides to help User to unblock it. System does not yet have functionality to do that automatically so it queries Action Assistant to do that for it. Action assistant creates a temporary exception for a firewall rule. External component receives a message about port status change and forwards via event to System. System executes a script that reiterates last failed activity, thus sending file, this time successfully. Process of bug resolution continues.

Detailed three-level RMS description

First level

In the first level, a *Risk-Adaptive Access Control* component, which interacts with the *AC Management and Context Monitoring Interface* component, is introduced. The risk-adaptive component provides access control decisions for access control requests based on risk and benefit calculations that take into account quantitative risk-related measures and contextual attributes [79,80]. The outcomes of the access control decisions are not "yes" or "no" answers, but are processes that represent the tasks components have to perform in order to be permitted to access data or perform an action.

The Risk-Adaptive Access Control component keeps a list of all system components which represent internal components, external components and users. Furthermore, this component has a set of tasks that can be performed by performed by the components and users. Each task is associated with a set of access control policies, and the system provides support for the creation, interpretation and enforcement of these policies. The *Risk-Adaptive Access Control* and the *AC Management and Context Monitoring Interface* components need to be physically located close to the components and users to be able to explicitly enforce access control decisions.

The resulting processes are provided based on a Risk Function, which in general can be represented as follows:

RF : {resource, component, qrm(component), risk(<action, resource, component>),
benefit(<action, resource, component>),context} → result

where

qrm : component → < ($type_1$,$value_1$), ($type_2$,$value_2$), ... , ($type_m$, value $_m$)>

risk : <action, resource, component> → <<($type_{risk\ 1\text{-}1}$,$value_{risk\ 1\text{-}1}$), ... , ($type_{risk\ 1\text{-}u}$, $value_{risk\ 1\text{-}u}$)>, ... , <($type_{risk\ h\text{-}1}$,$value_{risk\ h\text{-}1}$), ... , ($type_{risk\ h\text{-}y}$, $value_{risk\ h\text{-}y}$)>>

benefit : <action, resource, component> → <<($type_{benefit\ 1\text{-}1}$,$value_{benefit\ 1\text{-}1}$), ... , ($type_{benefit\ 1\text{-}v}$, $value_{benefit\ 1\text{-}v}$)> , ... , <($type_{benefit\ g\text{-}1}$,$value_{benefit\ g\text{-}1}$), ... , ($type_{benefit\ g\text{-}z}$, $value_{benefit\ g\text{-}z}$)>>

context : << action, resource, component>,<st_1, ... , st_n>,<ct_1, ... , ct_s>> → bool

The RF function receives as input the resource to be accessed, the component that is requesting access to the resource, the quantitative risk-related measures (qrm) associated with the component (sequence of types and values), the risk associated with action, resource and component (sequence of types and values for each type of risk), the benefit associated with action, resource and component (sequence of types and values for each type of benefit), and the context associated with action, resource and component, a set of system state values, and a set of context values (a boolean, that is true or false).

The result of the function RF is a sequence of processes, namely

<{$process_1$, resource_form $_1$}, {$process_2$, resource_form $_2$}, ... , {$process_e$, resource_form $_e$}> where

$process_l$ = <$task_1$, $task_2$, ... , $task_d$>, where $l \in \{1, 2, \dots, e\}$

resource_form is a transformed resource (e. g., obfuscated code, documentation)

Each of these processes consists of a sequence of tasks that could be performed to deal with the risks related to a specific situation.

For example, in case RF is applied to a scenario in which a developer is trying to access source code (i. e., resource) through a code-versioning system (i.e., component). We assume in that case that there is only one quantitative risk-related measure, namely trust (i. e., $type_1$=trust level, $value_1$=9). The developer has a high level of trust (i. e., 9 in 10). The action we are considering in this case is to access a source code. The risk can be related to a possible dissemination of the source code (i. e., $type_{risk\ 1\text{-}1}$=risk of the source code dissemination, $value_{risk\ 1\text{-}1}$=0.1). The benefit can be related to decreasing the chance of project delay (i. e., $type_{benefit\ 1\text{-}1}$=benefit of decreasing the chance of project delay, $value_{benefit\ 1\text{-}1}$=0.01). The context can involve a system state type called alert level and a context type called location. We assume the alert level is high and that the developer is working on site.

Taking into account the input we have described, RF evaluates risks against benefits. In this case risks are higher than benefits and, as a result, RF will produce three possible processes to reduce the risks or increase the benefits. The first process consists of two tasks: to ask the user to authenticate himself or herself with the system in an additional way (e. g., by swiping his or her smart card), and to close all other applications. If this process is successfully completed, the user will have

the ability to get a full version of the source code file. The second process consists of one task, that is, to get permission from the manager to access the full version of the source code file. When this process is completed, the user will have the ability to get a full version of the source code file. A third process can be defined as empty, meaning that there is no process to be performed. As a result, the user can get access only to the source code documentation.

Concerning the risk evaluation function, there is a need to balance the risks and benefits when making an access control decision. For example, if the risk that a source code will be maliciously disseminated (e. g., information leak) added to the benefit of denying access (e. g., when denying access to a file on a heavily loaded server can lead to a load decrease), outweighs the risk of denying access (e. g., the developer might be encouraged to follow the a path that will lead to project delays) added to the benefit of granting access (e. g., granting access to the code can reduce possible project delays), then access should not be granted. Note that neither the benefits of granting access nor a required balance between risks and benefits of granting/denying access are taken into consideration by existing (risk and non-risk based) access control systems.

In summary, the function of the Risk-Adaptive Access Control component is to make access control decisions, using the Quantitative Risk-related Measures (QRMs), context and policies. The risks and the benefits of granting and denying access adapt based on the context (e .g., user, task, location). The policies describe the risks and benefits of access control permissions and denials. The QRMs are retrieved from the second level of the RMS, specifically from the Quantitative Risk-related Measures component.

Second level

In the second level, three components are introduced. A *Risk-related Measures* component, which provides a set of numerical attributes (quantitative risk-related measures - QRMs) to the *Risk-Adaptive Access Control* component for risk and benefit calculations. The numerical attributes relate to contextual information, such as the context of a task or the context of a component. This component also provides methods to obtain certain types and their associated values of quantitative risk-related measures required by policies that an action to be performed needs to be subject to. These types and values of quantitative risk-related measures are acquired through trading them with other types of quantitative risk-related measures that the context of this task or component has.

With regard to trading QRMs, the demand and supply for each Quantitative Risk-related Measures creates a dynamic QRM ratio which is used to convert the Quantitative Risk-related Measures of one type to the Quantitative Risk-related Measures of the other types. The QRM ratios are stored as conversion tables in the Quantitative Risk-related Measures component. The conversion tables can change in many ways: first, which QRMs can be converted to which; second, the limits that restrict the QRM conversion ratio; third, the QRM ratio.

The values of quantitative risk-related measures can be altered by a *Policy Adjustment* component. This component adjusts the values of quantitative risk-related measures of the policies

based on the supply, demand and accumulated levels of component quantitative risk-related measures, the exchange rate among quantitative risk-related measures types that are interchangeable, and the frequency of usage of each policy. Besides supporting the adjustment of policy thresholds, the approach also includes methods to grant or revoke quantitative measures to components, to change quantitative measures exchange ratios. The adjustments occur when relevant monitored conditions become true. For example, if components lack a certain type of qrm to perform an action they need (e. g., trust), the system may choose to reduce the required value of this type of qrm in the policies (e. g., by reducing trust thresholds) or to increase the trust qrm of all components. In this way the system restores a balance between supply and demand for trust.

This level also contains an *Agent* component, which monitors all the events associated with a component or task context and builds the sequences of tasks a component has performed. These task sequences are built using process mining and user profiling techniques which gather information about users and components and stored as trees. The *Agent Component* is also used to reason about which types and values of quantitative risk-related measures can be traded. In general, the second level uses an adaptive access control mechanism defined at level one to support policy adaptation based on agent-mediated quantitative risk-related measures trading.

The actions that components perform are monitored by the *Agent* component and using process mining, the Risk Management system builds trees of task sequences which includes information about individual tasks and their context. The procedure defined to build these trees needs to be consistent in the sense that the same tree is built when similar processes and context are involved. The resulting task sequence trees should be accurate in the sense that it represents with high probability the sequence of tasks used to minimize risks in specific cases.

In addition, contracts are used to coordinate component interactions so that some components can provide certain assurances to other components. Further, when contracts fulfilled or violated, components can be rewarded or punished, respectively, and as a result QRMs may increase or decrease (e. g., trust may decrease if user violates a contract by publishing inappropriate pictures).

Third level

In the third level, there are also three components: a *System Goal* component, an *Agent Behaviour Storage and Evaluation* component *and an Anomaly Detection* component. A *System Goal* component is responsible for monitoring the effects of system states and task processes on goal compliance, and managing and evaluating the system goals. This component also includes the final goals that the system must aim to comply with. It also derives a set of intermediate goals that are used in evaluating component behaviour and adjusting policies. In the case, for example, that the final system goal is for the system to stay manageable, and because of an emergency, some roles cannot perform their tasks in the dynamic processes required to minimize risk. As a result, some critical processes are interrupted for an unacceptable period of time, which leads RMS to create an

intermediate goal that involves the creation of a new user account that can be used to perform tasks these roles need to carry out.

In addition, intermediate goals can also be created based on system history. For example, a failure of a specific component which seriously disrupted system activity and, consequently, led to a system goal non-compliance, could result in the creation of intermediate goals which may include replicating the component or providing it with additional resources that can enhance its operation. Further, we note that the *Goal* component decisions also depend on the history of intermediate goals that were created.

An *Agent Behaviour Storage and Evaluation component* evaluates the compliance of agent task processes with system goals. If the evaluation detects that the component behaviour can be critical to maintaining system goal compliance the Agent Behaviour Storage and Evaluation component would help the component to achieve its goals or prevent it to do so, depending on whether the component is identified as helpful or harmful, respectively, to remediate risky situations. When either helping or preventing a component to achieve its goals, actions external to the system can be defined in the form of task sequences that need to be performed to alleviate the risks. The *Agent Behaviour Storage and Evaluation* component also identifies and stores recurring patterns of component's behaviour [78]. The recurring patterns allow the approach to reconstruct the agent-based component process models if the currently used process-based task tree model is inadequate in predicting the subsequent component actions and needs replacement.

The third component in this level is the *Anomaly Detection*. This component monitors the system events to find desirable and undesirable patterns of task sequences [80,81] that could cause the system either to comply or not with system goals. It is also used to manage and change the policies and their attributes. As a possible solution, the *Anomaly Detection* component can search the monitored event sequences for patterns [82], and when a pattern is found it can be used by the *Policy Adjustment* component. The *Policy Adjustment* component uses the patterns as indication of risk-related problems that need to be resolved. In addition, the *Anomaly Detection* component can search for relevant event sequences that make the system more goal compliant, and such sequences can be used by the *Policy Adjustment* component to deal with risky behaviour. Many types of anomalies can be detected using this approach, including unauthorized access and dissemination of information (e. g., leaks) and malicious insider threats.

Appendix 2: Comparison of RMS with current related approaches

In order to show how our approach differs from existing approaches, a comparison with basic security systems and pattern-based systems is provided. By basic security systems we mean systems that would be most likely adopted by companies in the industry. These systems include firewalls, antivirus, passwording systems (user authentication), intrusion detection systems, anti-leak systems and email filters. Pattern-based systems are considered the state of the art and, besides having the

components of existing basic systems, the also use pattern analysis based on machine learning to evaluate user actions.

The comparison is presented in Table 1. In this table we provide in the first column the description of the events related to the scenario steps we have described previously. Each event description refers to the scenario step it relates to. The three other columns refer to how the basic systems, the pattern-based systems, and RMS, deal with each event, respectively.

Based on the example, we show that pattern analysis approaches extend basic access control approaches in the sense that pattern analysis approaches: (i) rely on recurrent patterns to validate processes (i. e., Event 1); (ii) identify suspicious activity (i. e., Event 5); and (iii) identify specific violations (i. e., violation of module ownership in Event 9; violation of task permission, Event 13).

Based on the example, we show in Table 1 that our Risk Management approach goes beyond the pattern analysis approaches in many ways in order to minimize risks based on risk remediation processes that components and users should follow, including by: (i) introducing resource and task management, which allows the approach to minimize risks; (ii) supporting reasoning involving context, tasks and goals of the system, the components, and the interaction processes (e. g., with users, external components and the RMS); (iii) providing agent-based reasoning and goal compliance mechanism to help users and components to reach their goals and prevent them from performing harmful tasks; (iv) using quantitative risk-related measures (e. g., reputation) to support the access control decision making process; (v) supporting decisions based on predicting task sequence using history and process mining; (vi) remediating risks based on interactions with stakeholders; (vii) evaluating and storing the process-based task trees; and (viii) reusing recurrent patterns stored in a process-based task tree library when reconstructing the process-based task tree models if the currently used one is inadequate in predicting the subsequent component actions and needs replacement.

Finally, in Table 2 we present a new features of RMS and events of the case study that relate to these features.

Event description	**Basic systems**	**Pattern analysis**	**Risk Management system**
1. Bug report request (S1)	Valid Process	Bug fix pattern started to execute	Valid Process
2. Bug report reply (S1)	Valid Process	Valid process	Bug is reassigned due to high risk of incompletion and critical priority
3. Failure to reproduce bug (S2)	Valid Process	Valid process	Valid process

4. Debugging data request (S3)	Valid Process	Valid process	Valid process
5. Debugging data sent (S3)	Intrusion detection system warning is detected due to potentially unsafe and uncategorized data transferred over the network	Activity considered suspicious	Potentially unsafe process identified, but due to previous failure and high developer reputation the process is allowed to continue execution
6. Failure to reproduce bug (S4)	Valid Process	Valid process	Valid process
7. Logs read (S4)	Valid Process	Valid process	Valid process
8. Error message found in storage module (S5)	Valid Process	Valid process	Valid process
9. Method documentation in storage module read (S6)	Valid Process	Violation of module ownership (this developer is not supposed to read that module)	Valid process (context association to storage module can be established from previous steps)
10. Change code to update method contract (S7)	Failure; file is locked	Suspicious activity (the developer is trying to access a file that is locked by another developer)	File is unlocked due to high bug priority; notification is sent to the developer who locked the file in the first place to maintain development process integrity
11. Project built (S8)	Valid Process	Valid Process	Valid process
12. Project build rejected (S8)	Valid Process	Suspicious activity confirmed	Valid process
13. Project is built locally (S9)	Valid Process	Violation of task permission	Valid process (task execution is allowed due to failure of previous step and high bug priority)

14. Updated executable is sent to the customer for verification (S9)	Anti-leak system alarm is activated; Email filter does not allow executable file to be sent	Pattern process violation (the bug fixing pattern disallows executable files to be distributed to an outside network)	Operation is considered potentially unsafe, and system requires the developer to ask permission from the project manager
15. The developer was notified that a different error message was displayed to the customer (S10)	Valid Process	Valid process	Valid process. The system confirms that the previous task is safe based on the beliefs of the developer agent and on task-related event context match between the observed event and previous events. The context match involves the developer trying to fix this specific error and the developer having fixed the same error many times in the past.
16. The developer asks the manager what to do (S11)	Valid Process	Valid process	Valid process
17. The manager asks the developer to stay over time and fix the new bug (S12)	Valid Process	Valid process	Valid process. The observed event allows the system to confirm that task sequence is accurate, to increase the developer reputation, and set new time frame for the task.
18. The developer reproduces the new bug (S13)	Violation of time frame	Violation of time frame	Valid process
19. The new bug is fixed (S14)	Valid Process	Valid process	Valid process
20. The developer applies the changes, and starts to build	Valid Process	Valid process	Valid process

the project (S15)			
21. The build server is on maintenance (S15)	Valid Process	Pattern process violation	Valid process. The system increases the priority of completing build server maintenance task defined in the project management system.
22. The first developer sends an email message to the second developer with the code needed to apply the changes (S16)	Anti-leak system alarm is activated.	Valid process	Valid process. According to the previous task sequences, the first developer can either continue to fix the bug or delegate the task to another developer.
23. The next day started (S17)	N/A	Context is updated	Context is updated and the task sequences that were completed on the previous day are labelled as completed and stored
24. The second developer starts to build a new project version with applied changes (S17)	Valid Process	Valid process	Valid process. The first developer delegates the task to the second developer. The context of the task sequence of the first developer is transferred to the second developer.
25. The regression tests have failed (S18)	Valid Process	Valid process	Valid process
26. The second developer asks first developer to solve the problem of regression test failure(S19)	Valid Process	Valid process	Valid process. The second developer delegates the task to the first developer. The context of the task sequence of the second developer is transferred to the first developer.

27. The first developer opens a connection to fix the error remotely (S20)	The firewall does not allow to open a connection to transfer the source code file to a remote location. Anti-leak system alarm is activated.	Violation of role location since the first developer is offsite.	The system considers the task as potentially unsafe. The system asks the first developer to go through an additional authentication step, makes sure the connection is secure (e. g., by changing protocol from ftp to sftp), and notifies the security analyst.
28. The first developer requests to remotely download the required source code (S21)	Valid Process	Violation of role location since the first developer is offsite.	Valid process. The system transfers source code on a per-needed basis. The source code components the developer needs to change are sent in plain text format, and the components that she does not need to change are sent either in binary or obfuscated forms.
28. The first developer updates the source code (S22)	Valid Process	Violation of role location since the first developer is offsite.	The system considers the task as potentially unsafe. The system asks the second developer to verify uploaded components.
29. The code bug is fixed (S23)	Valid Process	The pattern-based models are updated.	The bug fixing task sequence is completed and is stored along with its associated context. The system increases the reputation associated with the developers.

Table 1. Comparison of RMS with alternative approaches.

Novel features of the Risk Management approach	Relating the features to the specific events present in the case study
Introducing resource and task management, which allows the approach to minimize risks	E1-Task management E24 - Task is reassigned to another developer E26 - Task is reassigned back to first developer E28 - Source code is transferred on a per-need basis
Supporting reasoning involving context, tasks and goals of the system, the components, and	E9 - The context associated to the error E10 - The context includes high bug priority

the interaction processes (e. g., with users, external components and the RMS)	E10 - Complying with the system goal (integrity) E13 - The context includes high bug priority E15 - The context related to the task of fixing the second bug E15 - Agent reasoning about beliefs E23 - The context is updated (next day) E24 - The context of the bug fixing task is associated with the second developer E26 - The context of the bug fixing task is associated with the first developer
Providing agent-based reasoning and goal compliance mechanism to help users and components to reach their goals and prevent them from performing harmful tasks	E10 - File is unlocked, notification sent E17 - Agent reasoning to set a new time frame for the bug fixing task defined in the project management system (external component) E21 - Priority of completing the build server maintenance task defined in the project management system (external component) is increased E27 - The system notifies the security analyst, asks the developer to go through an additional authentication step, and makes sure the connection is secure E28 - The system converts the data the developer needs to other forms (i. e., obfuscated, binary)
Using quantitative risk-related measures (e. g., reputation) to support the access control decision making process	E5 - Reputation measure E17 - Increasing the developer reputation E29 - The reputation of the developers is increased since they were successful
Supporting decisions based on predicting task sequence using history and process mining	E5 - Previous failure E9 - Previous steps in current process (tasks related to bug fixing) E13 - Previous failure E15 - Previous steps in past completed processes (tasks related to bug fixing) E22 - Previous steps in past completed processes (tasks trees in bug fixing)
Remediating risks based on interactions with stakeholders	E14 - Asking permission from the project manager
Evaluating and storing the process-based task trees	E17 - Confirming that the current task tree is accurate E23 - Task sequence completed on the previous day is stored

	E29 - The task sequence is labelled as completed, and stored
Reusing recurrent patterns stored in a process-based task tree library when reconstructing the process-based task tree models if the currently used one is inadequate in predicting the subsequent component actions and needs replacement	Not applicable

Table 2. New features of the approach in relation to the events in the case study

As a result of the comparison and illustrative case study, a novel risk management approach for DEBSs should provide:

- continuous access control outcomes ([0..1] instead of {true, false});
- process-based risk mitigation (e. g., management approval to access data);
- access control policy adaptation;
- agent-based goal compliance and task-based risk remediation;
- agent-based incentives (rewards, punishments) related to contracts;
- risk-related measures trading (e. g., reputation for performance);
- agent-based reasoning about system goal compliance;
- agent-based modeling and risk-related prediction of component behavior.

Finally, we have argued in this report that that the introduction of an approach that relies on adaptive access control and agent-based threat assessment (e. g, leak detection, malicious insider attacks), and provides an architectural design and implementation, can lead to a new form of risk management and dynamic risk-based access control for DEBSs.