Application of Scenario-driven Role Engineering in Knowledge Management Systems - Requirements and Implementation

Daniel Kimmig*, Andreas Schmidt†*, Klaus Bittner*, and Markus Dickerhof*
*Institute for Applied Computer Science
Karlsruhe Institute of Technology
Karlsruhe, Germany
E-mail: {daniel.kimmig, andreas.schmidt, klaus.bittner, markus.dickerhof}@kit.edu
†Department of Informatics and Business Information Systems,
University of Applied Sciences, Karlsruhe
Karlsruhe, Germany
E-mail: andreas.schmidt@hs-karlsruhe.de

***Abstract*—Collaborative systems, which are often used in short-term virtual enterprises or long-term cooperation networks, often contain information about the manufacturing and fabrication competences of the participating technology partners. These should only be made available to a very restricted group of persons for example to support feasibility studies in the context of actual customer requests. This is a new important feature to be supported in nowadays knowledge management systems. Hence, the goal of this paper is to present a methodology for implementing an access control mechanism based on role based access control. This mechanism supports the definition of fine granular access rights capable of protecting sensible information often found in cooperative process knowledge management systems. In this paper we will discuss models of access control and present an adaption of the scenario-driven role engineering method to the special needs in a collaborative process knowledge management system with very particular access requirements. Beside the adaption of the scenario-driven role engineering method to such a system, the adapted method will be concretely applied to the process knowledge management system *MinaBASE*, which was developed in our institute. To complete, an implementation will be shown with the help of the inversion of control framework "Spring Security" as well as aspect-oriented programming. Here static as well as dynamic aspects of security will be presented. The paper shows in a detailed manner the usability of the scenario-driven role engineering method for applications in the field of collaborative knowledge management.**

***Keywords*-Access control; Knowledge Management; RBAC; Role Engineering.**

I. Introduction

Both corporate groups as well as small and medium-sized enterprises (SME) are experiencing increasing competition and shorter product lifecycles [1]. The resulting necessity of shortening the product development process also has to be mastered by enterprises in the field of microsystems technologies (MST) that are characterized by a high interdisciplinarity and complex, multi-stage, and hardly standardized fabrication processes. Frequently, every product is produced by an individually tailored fabrication process [2]. While larger MST enterprises still manage a wide spectrum of technologies, SME rather tend to offer solutions in a high specialized area. To offer more complex solutions, they establish technical partnerships with other SME. These may have the form of short-term virtual enterprises or long-term cooperation networks. To support such organization forms in the field of MST, the *MinaBASE* process knowledge database was developed by the Institute for Applied Computer Science of Karlsruhe Institute of Technology. By means of this database, the manufacturing and fabrication competences of the technology partners are made available to a central coordinator who then assesses technical and economic feasibility.

This information, however, includes company secrets, whose confidentiality and integrity is of crucial importance to a company's existence. Acceptance of *MinaBASE* therefore does not only depend on meeting functional requirements, but also on aspects like security and access protection. To prevent an undesired disclosure of company secrets, access shall be controlled by the established role-based access control (RBAC) [3]. Here, authorizations are not assigned directly to subjects, i.e., the users of a system, but to abstract roles to which the users are assigned subsequently. In this way, the frequently error-prone maintenance is reduced and security is increased. This requires the definition of an adequate role concept.

Information systems often use standard models with system-wide administrators, owners of information objects, and guests having restricted read access. While this is a reasonable default for establishing a minimal level of access control, it does not consider the business processes in which the system is used and which particular requirements result in terms of confidentiality and data integrity. The lack of a role concept tailored to these specific requirements can severly harm the acceptance and usage of a knowledge management system. For example in *MinaBASE*, it should be possible to grant a partner access to product-related properties of a microsystem during the sales process without disclosing the configuration of machine parameters to produce these properties. Such an application exceeds the modeling capabilities of the standard role concept described above, as external partners are not the owners of this information. This shows that nowadays knowledge management systems have particular security requirements, which require activites of role engineering to create an adequate role concept to support usage in different business processes. Our contribution takes on this problem by showing how a role concept tailored to the particular requirements of process knowledge management systems can be defined and implemented. The main contribution of the paper is twofold: First, the scenario-driven role engineering method will be adapted to the requirements of collaborative knowledge management systems. And second, the suitability of existing access frameworks to implement the adapted method will be shown by means of the framework spring security.

The paper is structured as follows: In the next section, the *MinaBASE* process knowledge database will be presented. Then, mechanisms for access control as well as the scenario-driven role engineering approach [4] and the adaptations due to the background and objective of *MinaBASE* will be outlined. Subsequently, the methodology will be applied and a role concept for RBAC ensuring data integrity and confidentiality for *MinaBASE* will be derived. The final section describes the implementation of this role concept within an "Inversion-of-Control"-Framework (IoC) by demonstrating how Spring Security and technologies such as aspect-oriented programming (AOP) can be used to fulfill static and dynamic security requirements.

II. *MinaBASE* Process Knowledge Database

The knowledge required to produce added value is no public property, but a company resource that has to be administrated efficiently in order to ensure economic success. To support this process, knowledge management systems have been established [5]. In process-oriented knowledge management [6], these methods are applied to highly knowledge-intensive fabrication processes, as those used in MST. The *MinaBASE* process knowledge database is used by the technology partners for the structured storage of technical fabrication parameters of the methods and materials used in MST and of the partner-specific technical competences. In *MinaBASE*, the smallest information entity is the so-called technical aspect (TA). It is used to model materials, machines, and fabrication technologies [7]. By means of generalization hierarchies, TAs are arranged in taxonomies. The number and contents of taxonomy trees can be specified and modified during runtime, such that a flexible structure meeting MST requirements can be defined for the storage of fabrication know-how. TAs may be assigned properties referred to as technical parameters (TP). A TP is a string of characters, integers, or floating-point numbers in a certain

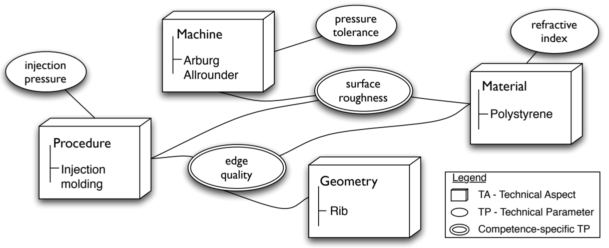


Figure 1. Schematic representation of a *MinaBASE* competence [9].

unit and references an attribute, e.g., the aspect ratio. The TP of a TA are inherited by lower partial hierarchies of the hierarchy tree in analogy to the object-oriented approach. In addition, lower hierarchy levels can further refine the inherited TP. Classification of TP places them in a certain context, such that a TP refers to a product or its fabrication and, hence, is either product-specific or fabrication-specific. Product-specific TP describe the properties of a microsystem, such as the depth of a groove reached by the fabrication process of milling. Fabrication-specific TP refer to the machine configuration needed to produce a specific product property. For modeling the capabilities of a technology partner, competences [8] are considered to be a set of various TA from disjunct hierarchy trees, which is illustrated in Figure 1. This figure schematically represents the competence "injection molding of a rib with polystyrene using the Arburg Allrounder machine" together with some TP. From the hierarchy trees of process, machine, material, and geometry element, the TAs are selected. These TAs are characterized by their TP, such as the injection pressure of the injection molding process. The combination of these TAs results in the competence that is reflected by other TP, such as the edge quality and surface roughness. Consequently, a competence is a type of view of a certain combination of TAs with properties in the form of TP that apply to this combination only, i.e., that characterize the competence in more detail. TAs can be used in several competences. They represent reusable, encapsulated, smallest information entities. An extension of the *MinaBASE* concept has been developed in order to reuse these information entities to allow process modeling of manufacturing sequences based on semantic technologies [9].

III. Access control

Information security is concerned with mitigating risks that affect information systems in the age of growing interconnectedness of computers. The purpose of information security is to establish a state, in which the following three criteria are met for the protected information.

- *Confidentiality* in this context means that only users with certain privileges are allowed to access protected information.

- *Integrity* ensures that information can only be altered or deleted by users that have sufficient permissions.
- *Availability* is the requirement that an authorized access to information is possible at any time.

Access control mainly refers to the first two points above: Confidentiality and integrity. But it also has an impact on the third criterion, since bypassing access control is often a first step to be able to compromise availability [10]. In the digital age massive productivity gains have been achieved due to the integrated availability of enterprise data in business information systems. However, the availability of critical data in such systems involves risks, because the majority of attacks come from within an organization itself [11]. This means that in the long term value added can only be achieved when access to the information systems can be controlled appropriately. The so-called ”Broken Access Control”, i.e., the selective exploitation of unsuitable access control, takes a central place in the OWASP (Open Web Application Security Project - a project, which focuses on the analysis and control of secure software), which reports the most common vulnerabilities of web based applications. From this it can be concluded that the design of an optimal access control for applications such as *MinaBASE* is one of the most important factors for ensuring information security. Essential for this are principles for the design of secure applications that are described briefly below.

- The principle *Fail-safe defaults* stipulates that any attempt to access any object by an arbitrary subject is to be rejected, unless it was explicitly permitted [12] [13]. Instead of asking why one cannot access a resource, it is more important to ask why one should be able to access it in the first place.
- The *Principle of Least Privilege* was first postulated by Saltzer and Schroeder [14] and states that a user within a certain period of time, e.g., during a session, should only possess the minimal amount of privileges that are sufficient for him to fulfill his task. This will ensure that access control cannot be easily circumvented by privileges that were granted too loosely [10].
- The demand for the principle of *Separation of Duty* (SoD) is correlated to the need for integrity of information. This mainly concerns operations on resources that are either very risky or where a situation cannot be excluded in which resources are at risk of being abused even by authorized users. For such cases SoD suggests to split the operation onto multiple users, so that no single user has sufficient authority to that operation [10].

After the objectives and key principles for information security through access control have been introduced, the following sections will present different models of designing access control mechanisms, which are qualified to achieve the described goals of information security.

A. Discretionary Access Control

The model of *Discretionary Access Control* (DAC) is based on the principle of ”object ownership”, which means that one or more owners are assigned a resource, grant permission to access these resources at their discretion. The decision on granting access is based on the identity of a user or his group membership [10]. The term ”discretionary” means that the passing of permissions to specific operations on resources such as files is determined by the owner's confirmation. This means that each user is enabled to spread these rights to another user or his subjects [10]. The implementation of the permission distribution is based on so-called ”Access Control Lists” (ACLs) or capability lists. The advantage of DAC is a high degree of flexibility, since it is possible to grant permissions in a very fine-grained manner. The drawback of DAC is that you can not limit the spread of permissions.

B. Mandatory Access Control

Mandatory Access Control (MAC) relies on weaknesses of DAC, and is therefore specifically designed for the containment of potential information flows [15]. This is achieved as MAC has no principle of object-ownership, but is built upon a classification of information. The sensitivity of information and user status function as a decision criterion for access requests. The sensitivity is determined by the classification of the information depending on how big the damage caused by the loss of confidentiality would be. MAC prevents the proliferation of permissions to users who were not considered to be authorized, as the mechanisms of MAC protect information of a certain level from being accessed by users of an insufficient level. Examples of such mechanisms are the *Bell-LaPadula model* [16] and the *Biba Integrity Model* [17]. The advantage of MAC is the safe limitation of access permissions, however it prevents the flexible sharing of information between users, because the classification of information and users is predetermined and therefore static and rather inflexible [13].

C. Role-based Access Control

In *Role-Based Access Control*, permissions for operations on resources are not assigned directly to users, but an abstraction in between both concepts is created, which is referred to as a role [3]. The meaning of these roles is directly comparable with roles in organizations. Subjects who use a system to accomplish similar tasks, act in a similar role towards the system. Therefore completely different permissions are required, which are limited by the operations necessary to accomplish the different tasks. Permissions are assigned to roles in RBAC [10], thus there is no direct allocation of permissions between users and objects. The reason for the development of the RBAC model is based on two insights: The first is that after investigating the security requirements of commercial organizations, it was

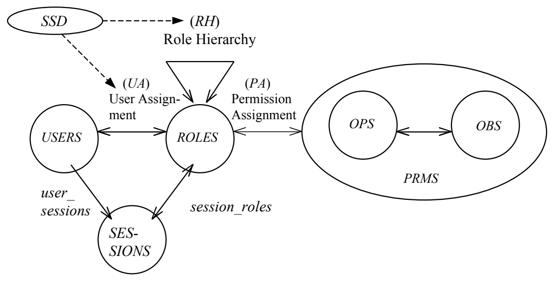


Figure 2. Schematic overview of the extended RBAC model.

found that neither MAC nor DAC cover the needs of these organizations. The origin of the MAC model is the protection of classified information, which means a way of controlling which subject can see what kind of information. In contrast to that, RBAC is concerned with the question of which subjects are allowed to perform which operations on what kind of resources [3]. In addition, it is very difficult to classify information and subjects for a commercial organization, since such a classification is static and inflexible. This lack of flexibility is overcome by the characteristics of DAC to make the decision to limit propagation of access permissions at the discretion of the "Object Owner", since a more dynamic access control becomes possible. The challenge in commercial organizations is, users are not the owners of resources, but the institution, in which they are embedded into [18].

Consequently, the essential principle of "Object Ownership" of DAC is not applicable, as the distribution of access privileges should not be put at the discretion of the users. For this reason RBAC is also referred to as *non-discretionary* [3]. RBAC is less focused on the grouping of users such as DAC, but rather on grouping of permission sets, which enable the execution of operations on resources [3]. Through this concept, the grouping of permissions to roles, administration becomes easier, as changes to users only result in updating the membership to associated roles [18].

It also supports the distribution of RBAC permissions according to the "Principle of Least Privilege", since the roles of an organization can be assigned with a minimal amount of privileges necessary to complete the respective tasks within the organization. If there are conflicts of interest between certain units of the organization, these can be overcome using the technique of "Separation of Duty", which means that restrictions are placed on the distribution of roles. Over time, different stages of RBAC have been developed, which build on each other and will be briefly described below. A schematic structure of these models is given by Figure 2 from [11]. Basic RBAC only consists of three sets, which model users, roles and permissions. Roles exist purely because of the grouped assignment of permissions. In reality, however, there is a hierarchical arrangement, as some roles consist of more permissions than others and thus the privileges of these are included redundantly. The introduction of a hierarchical arrangement for *Hierarchical RBAC* directly in the model decreases the required effort to administrate access control. The arrangement of roles can then be represented by a partial order as a graph or as an inverted tree. With the idea to encapsulate the functions of an organization and the necessary permissions to roles, it becomes apparent that this can easily lead to an abuse of privileges in a commercial environment. To avoid such cases, the principle of "Separation of Duty" is part of the *Constrained RBAC* model, in order to assign permissions to different roles in cases of conflicts of interest. The use of the RBAC model has the potential to reduce complexity and error rate of access control as well as to reduce the cost and duration of administration.

IV. Scenario-driven Role Engineering

The term of role engineering (RE) in the context of RBAC means the design and specification of roles, authorizations, secondary conditions, and restrictions as well as of a hierarchic role model [19]. RE is used to create a concrete model for RBAC-based access control. In [4], Scenario-driven role engineering (SDRE) is defined as an approach based on scenarios, such as sequences of actions and events from the user's perspectives. This sequence in a scenario can be subdivided into subscenarios and atomic steps of chronological user interaction. Scenarios are embedded in a task, i.e., a problem or a work area, which links the scenarios of a system with its users. The users mostly apply a system to fulfill a task of their work profile or their job description. This structurization into various levels serves to break down a job description of a user into atomic steps, each of which may be associated with an authorization to access a resource. For various types of users, the minimum amount of authorizations required for the execution of the tasks can be derived. In this way, the principle of least privilege [14] is implemented. For documentation, various models are generated by the SDRE approach, which are interlinked in terms of contents and used for the derivation of the role concept. The *scenario model* describes all scenarios and steps, *task definitions* serve to structure scenarios, the *work profile* summarizes tasks for job descriptions. The *permission catalog* lists the individual permissions or authorizations. It may be complemented by a constraint catalog of special limitations [4]. While the permission catalog is focused on static assignments of authorizations to specific resources, constraints describe dynamic conditions, which are evaluated at runtime. Hereinafter, the SDRE process will be described in general. First, the use scenarios of the system are compiled and their actions and events are documented. Then, subscenarios and steps are defined and the authorizations required for them are included in the permission catalog or special limitations are listed in the

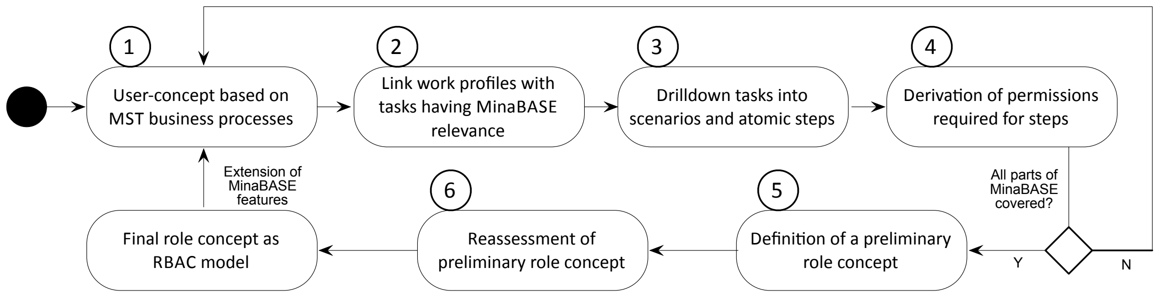

1
User-concept based on MST business processes
2
Link work profiles with tasks having MinaBASE relevance
3
Drilldown tasks into scenarios and atomic steps
4
Derivation of permissions required for steps
All parts of MinaBASE covered?
Y
N
5
Definition of a preliminary role concept
6
Reassessment of preliminary role concept
Final role concept as RBAC model
Extension of MinaBASE features

Figure 3. Scenario-driven role engineering according to [4] with adaptations.

constraint catalog. When this step is completed for all scenarios, similar scenarios are generalized. Very complex scenarios are divided into smaller parts which are then included in the scenario model. On this basis, tasks are formed by grouping scenarios. These tasks are then classified into various work profiles. This results in a preliminary role concept and minimum authorizations can be assigned to the individual activities. As a rule, this preliminary model contains duplicates of roles with identical authorizations, which then have to be fused in a last step. This yields the RBAC model as a role concept. The SDRE process represents a systematic approach to RE. It was applied to information systems for the health care sector by the technical committee of HL7 already [20]. Due to this practical test, SDRE in principle may be applied to *MinaBASE*. However, certain adaptations are required, because the background and objective of *MinaBASE* differ from those of the HL7 systems. The paramount objective of *MinaBASE* is the support of knowledge-intensive business processes of MST enterprises by a structurization of the knowledge required for the execution of these business processes. A criterion for the acceptance of knowledge management is its integration in workflows of the users and an efficient and complete coverage of information needs [21]. As such the SDRE approach is to be applied to the use of *MinaBASE* in business processes of MST enterprises and cooperation networks. The model given in [4] is therefore subjected to the following adaptations:

- In the standard SDRE methodology, scenarios for a system are the main input, to which required authorizations are allocated. Subsequently, these scenarios are generalized and assigned to tasks and work-profiles which create a preliminary role concept.
  For *MinaBASE* however an alternative input is more persuasive. Instead of starting with the scenarios of the system, work areas within the business processes of an MST-company are examined, whether they include tasks in which *MinaBASE* can be used to increase added value. To these tasks scenarios will be assigned in order to obtain the information, which resources are required for fulfilling them and what authorizations are needed. Based on this information, a preliminary role concept can be derived in a similar way to the standard model due to the minimal set of authorizations for each role. By these adapations - a switch of input variables to the methodology - the basic principle of SDRE is preserved, while better results for the creation of the role concept are expected, because of the adapted methodology being closer to the business processes of a MST-company.
- The scenarios to be formulated are not based on consequences of actions and events, but will also contain all definable steps. Although these do not occur in sequential order, they can be characterized by a certain access authorization.
- For reasons of clarity, special limitations extending beyond the static allocation of authorizations are included directly in the permission catalog and not in the constraint catalog, such that both models fuse.

The adapted process is illustrated in Figure 3. It comprises six steps, the execution of which shall be described in more detail in the following section.

V. Application of SDRE to *MinaBASE*

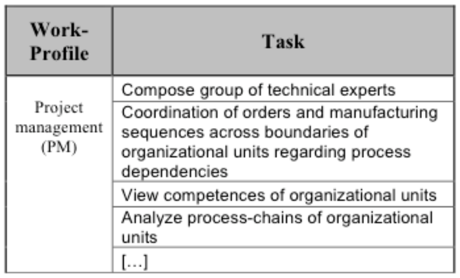
Using parts of the models created by the SDRE process, it shall now be demonstrated how the role concept can be generated systematically.

A. Step 1: Generation of the User Concept

Application of the model is based on an analysis of the business processes of a model MST company for possibilities of using *MinaBASE* and for activities, where the use of *MinaBASE* can result in a added value. Functions and units of an MST company, which may be potential users of *MinaBASE*, are:

- *Sales, external guest: MinaBASE* supports the sales process in the strategic assessment of the feasibility of customer orders, because these decisions can be made based on an IT documentation of competences and fabrication know-how. Strategic means that a general decision is made without taking into account technical details. In addition, the customer order is typed depending on whether a standard product is to be manufactured or a specification has to be met by enforcing the development in a project. The documented competences can be used as a database for sales promotion. External guests, e.g., customers or suppliers, may be given access to the system in order to stay informed about fabrication processes used by the company or the cooperation network.

- *Project management, development:* If a customer order is classified to be not directly producible by the sales division, a project team is established based on the customer's specification. This team is composed of the project manager and technical experts. In an iterative process, they specify general solution alternatives, the commercial feasibility of which is assessed. In addition, solution approaches, such as functional patterns or prototypes, are developed in detail, the technical feasibility of which is guaranteed. Upon successful agreement with the customer, exact fabrication planning is started in the next step. Planning is based on the results of the development of a commercially and technically feasible solution.
- *Construction, fabrication planning:* Planning of fabrication, i.e., of the individual steps of production flow, may be initiated by a successful development process or a directly producible customer order, e.g., the repetition of an already executed fabrication process. In the latter case, *MinaBASE*, a system for process-oriented knowledge management, provides support by the storage of process elements of process steps and process sections and their combination in process chains, as this allows for the direct use of already executed fabrication processes [9]. This principle in weaker form may also be applied to fabrication planning based on a technical solution alternative from development. By copying or adapting existing process models or process elements, planning of the fabrication process can be accelerated. Construction and fabrication planning result in a detailed schedule for production and defined quality management tests, during which data are measured in the production process.
- *Quality management, production:* Production focuses on the execution of the process steps defined by fabrication planning in a process chain to execute the order placed by the customer. Technicians working at the machines have direct access to production and are capable of using technical parameters of the individual process elements of the process chain for adjusting the machine parameters and of measuring real data during the tests. Various areas of quality management are covered. New fabrication knowledge of attributes and parameters of process elements is generated.



Work-Profile	Task
Project management (PM)	Compose group of technical experts
	Coordination of orders and manufacturing sequences across boundaries of organizational units regarding process dependencies
	View competences of organizational units
	Analyze process-chains of organizational units
	[...]

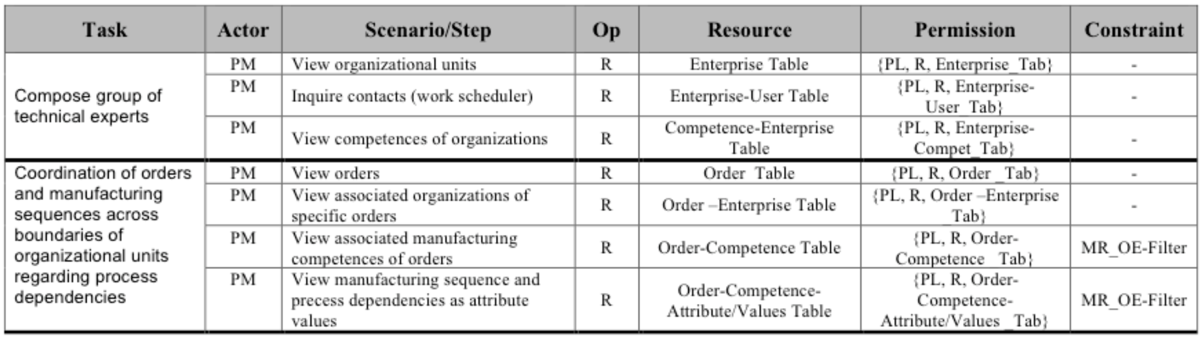
Figure 4. Work area project management with associated tasks from the work profile model.

B. Step 2: Definition of Work Profiles and Task Definitions

According to the adaptations to the SDRE model, *MinaBASE* tasks are assigned to the enterprise units or work areas listed in the previous section. Figure 4 shows a part of the work profile model. In the work area of project management for the iterative development of solutions for an unsolved development problem of a customer order, tasks are identified, to which the *MinaBASE* resources can be applied. These tasks are the pooling of technical experts, the analysis of fabrication competences and process chains, and the coordination of process dependencies beyond organizational units. The complete work profile model contains all tasks to which *MinaBASE* may be applied. These are the input variables for the detailed assignment of scenarios, steps, and authorizations to access resources in the following step.

C. Steps 3/4: Refinement of Scenarios and Assignment of Authorizations

For the first two tasks mentioned in the previous section, namely, the pooling of experts and the coordination of process dependencies, a part of the fused permission and constraint catalog is illustrated in Figure 5. For every scenario or every step, the associated operation on an object or resource is modeled, with R denoting read access (read), C denoting the creation of a new entry (create), U meaning processing (update), and D the deletion (delete) of a resource. The information of which actor accesses which resource with which operation is encapsulated as a permission by a triple of the type (actor, operation, object). At last, limitations or constraints of access are specified. For the first task, the organizational units, contact data of technical experts, and competences of the organizations stored in *MinaBASE* are considered as use scenarios. Read access (R) to the tables of the database and application components is required. The second task is handled similarly, as order data and detailed, production-related attribute values of process dependences are needed. In addition, an entry in the constraint catalog is made to ensure that the actor sees only those attribute



Task	Actor	Scenario/Step	Op	Resource	Permission	Constraint
Compose group of technical experts	PM	View organizational units	R	Enterprise Table	{PL, R, Enterprise_Tab}	-
	PM	Inquire contacts (work scheduler)	R	Enterprise-User Table	{PL, R, Enterprise-User_Tab}	-
	PM	View competences of organizations	R	Competence-Enterprise Table	{PL, R, Enterprise-Compet_Tab}	-
Coordination of orders and manufacturing sequences across boundaries of organizational units regarding process dependencies	PM	View orders	R	Order Table	{PL, R, Order _Tab}	-
	PM	View associated organizations of specific orders	R	Order –Enterprise Table	{PL, R, Order –Enterprise Tab}	-
	PM	View associated manufacturing competences of orders	R	Order-Competence Table	{PL, R, Order-Competence _Tab}	MR_OE-Filter
	PM	View manufacturing sequence and precess dependencies as attribute values	R	Order-Competence-Attribute/Values Table	{PL, R, Order-Competence-Attribute/Values _Tab}	MR_OE-Filter

Figure 5. Refinement of tasks in use scenarios and assignment of authorizations to access the resources needed.

values that are characterized as project-specific property and not as production-specific, internal know-how of an organization. This constraint cannot be implemented as a static authorization, as the assignment is made dynamically during runtime.

D. Steps 5/6: Derivation of the Role Concept

By applying the first steps of the adapted SDRE model to the identified *MinaBASE*-using enterprise units, a hierarchy corresponding to a preliminary model of the role concept may be derived on the basis of the authorizations. This preliminary model is show in Figure 6. The highest point is the administration that is not only responsible for administering users and their assignment to roles, but also has all other authorizations in the system. The lowest point is the external guest, who is given fewest access rights. In between, the graph may be structured horizontally and vertically. Vertical structurization results from the degree of orientation to orders. This means that planning of working steps of a process chain and their execution are much more related to orders than the development of solution alternatives for a certain customer specification by technical experts. Horizontal arrangement results from the respective amount of granted privileges.

Then, the last step of the SDRE process follows, i.e., the analysis of the preliminary model for groupings of authorizations in the form of roles that exist several times and have a comparable amount of authorizations. These roles have to be eliminated. Otherwise, the catalog would list more roles than necessary, which might result in anomalies and undesired side effects in the administration of rights and roles. Review of the preliminary model taking into account the criteria described yields the role concept shown in Figure 7. Documentation of the authorizations for the individual company units shows that a separation between project management and development is not reasonable, as the access rights for the modeled scenarios and steps are identical. For this reason, both units are summarized by the developer role. The same applies to fabrication planning and quality management, as both units use *MinaBASE* for various

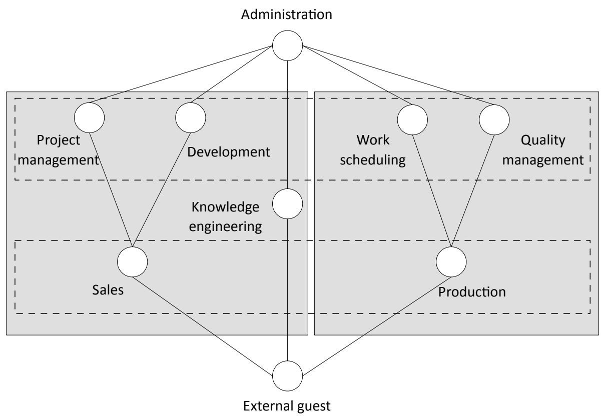


Figure 6. Preliminary role concept based on the permission catalog.

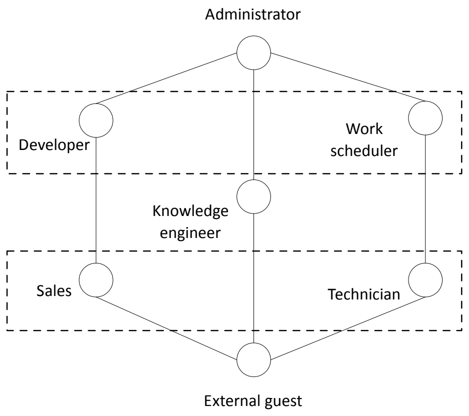


Figure 7. Revised role concept as RBAC model.

objectives, but still have comparable use scenarios and, hence, identical authorizations. Consequently, they assume the same role in the use of *MinaBASE*, the role of the work planner. Underneath the role of the work planner, the role technician exists, who is responsible for the implementation of the plans made by the work planner. The technician is in the position to acquire the measurement data for the tests of the work planner and to define detailed parameters, such as machine instructions, as information added to process elements. To fulfill his task, the developer needs deep insight into the details of the competences and process chains, as he has to extend the strategic assessment of the sales division by a guaranteed technically possible feasibility assessment. The "knowledge engineering" component has already encapsulated the rights to update order-independent knowledge in the preliminary model. In this way, additional authorizations can be assigned specifically to a role.

VI. Implementation in an IoC-Framework

This section describes the implementation of the RBAC model within *MinaBASE*. Firstly, the used framework, Spring Security, is introduced. Subsequently, it is shown how static and dynamic security requirements stemming from the permission catalog as well as the constraint catalog can be fulfilled.

A. Spring Security

Spring Security is a subproject of the application framework "Spring" to control authentication and authorization in the JEE environment, i. e., in the range of business applications based on Java technology [22]. It is empowered by technologies provided by the core of Spring, such as "Inversion of Control" using "Dependency Injection" (DI), which means a passive provisioning of an application component's dependencies by a central container known as the Spring "ApplicationContext". Martin Fowler defined the term dependency injection as means of provisioning an object's dependencies [23], for which several tools have been developed to aid the construction of large object graphs consisting of many interrelated classes mainly based on declarative configuration. The main motivation behind the idea of dependency injection is the overcoming of drawbacks found in previous solutions to the problem of object graph construction. The "ServiceLocator-Pattern" is an example of such a solution. The difference between the two approaches is displayed in Figure 8.

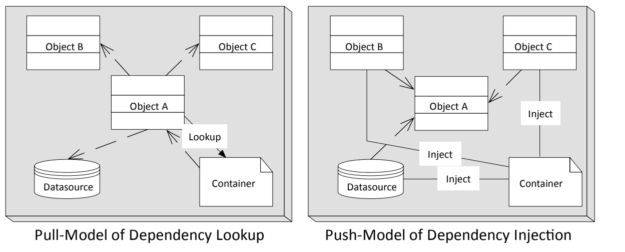


Figure 8. Approaches to constructing object graphs and its dependencies.

The first approach can be characterized as a "Pull-model", in which an object actively requests certain dependencies from a central container that helps in locating services such as a reference to a datasource. The second approach uses a push-model, where the object is passive and the container will manage the object's lifecycle alongside the fulfillment of dependencies that the object requires. The main advantage of the second approach is, the object does not need to know about the central container and is therefore easier to test, refactor and maintain [24].

In addition to that, the ApplicationContext provides AOP capabilities. New programming paradigms usually are invented to overcome the weakness of well established paradigms. In the same way AOP is an extension of traditional object-oriented programming (OOP). In OOP, classes are used as blueprints to model objects from the real world. By following principles such as separation of concerns [25] and information hiding of implementation details, OOP had a drastic effect on the way functional requirements are translated into the structure of application code. However, after decades of experience with OOP, it has been shown that a strict separation of all concerns is not feasible as there are requirements that are the same across all components of a system. Examples for this include but are not limited to transaction management, logging as well as enforcing security policies [26]. Concerns like these are equally relevant to several components while at the same time not capurable as separate components in traditional OOP. As they spread across several components, they are often refered to as cross-cutting concerns. AOP is used for central encapsulation of cross-cutting concerns into so-called aspects, which avoids the scattering of duplicated code for realizing them across the codebase. The difference of the two approaches is shown in Figure 9.

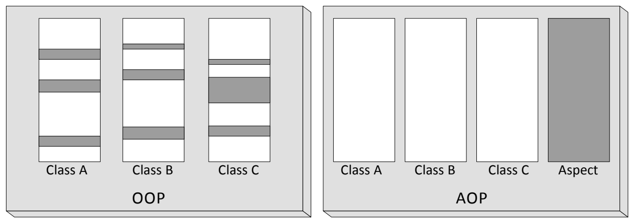


Figure 9. Using AOP to avoid duplication of cross-cutting concerns.

Utilizing DI and AOP, Spring Security allows central definition of security constraints. By integrating with the hosting web container, a central hook is implemented by which a chain of filters can monitor and control the processing of HTTP requests as well as the execution of application components. This enables Spring Security to capture all elements of an application's architecture while thoroughly ensuring its security requirements using a declaratively configurable mechanism. Figure 10 visualizes how a request is processed.

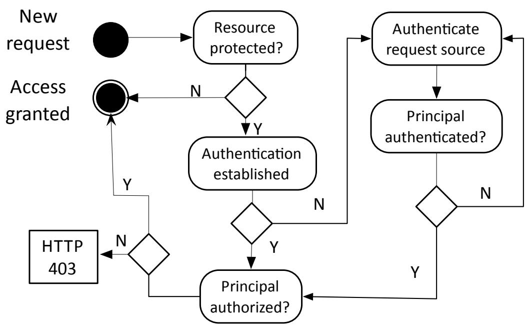


Figure 10. Overview of request processing within Spring Security.

This central hook determines whether an incoming request is trying to access a protected resource according to the supplied configuration. If this is the case, the ”AuthenticationManager“ (AM) is requested to authenticate and return the current principal, an abstract notion for, e. g., the currenlty logged in user, which is used by the ”AccessDecisionManager“ (ADM) to determine whether its role has the permission required for the protected resource in question.

These two components can be controlled in a very flexible manner. For instance, during authentication, the AM can be configured to consult different providers, which in turn compare the principal's credentials by querying relational databases, LDAP directories or Single-Sign-On authentication servers. The ADM can be controlled by assigning static key/value-pairs of resources and required permissions or by enabling the dynamic execution of AOP-driven components. As the flow of the request processing shows, no access is granted unless the source of the request is properly authenticated and the principal possesses sufficient permissions. This effectively realizes the ”Fail Safe Defaults” principle as it will return a security error unless both conditions are met. The following shows how Spring Security can be used to enforce compliance with the static and dynamic security requirements as specified in the permission- and constraint catalog in Figure 5.

B. Static aspects of security

Now that the basic functionality and main components of Spring Security were introduced, this section will focus on how the AM and ADM are used to implement the security requirements stemming from the SDRE role concept defined in chapter V. The following Figure 11 visualizes the architecture of *MinaBASE*. As shown, the application serves multiple types of clients while also utilizing heterogenous datasources. It is structured using the popular 3-tier architecture, which divides the application into presentation logic for request processing, application logic for satisfying business requirements and components for accessing the various datasources.

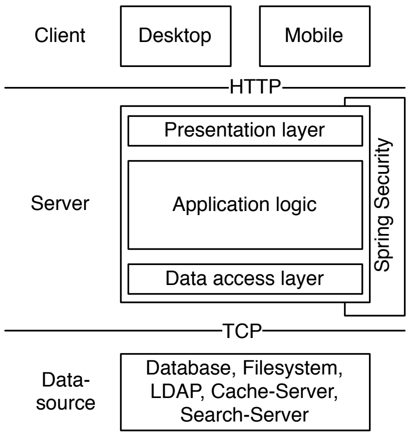


Figure 11. Schematic overview of *MinaBASE* architecture.

The following will show how Spring Security is used to implement the security requirements across the entire application architecture in a coherent way. Extending the security mechanisms requires the ADM to use a FilterSecurityInterceptor“ (FSI) for securing the presentation layer as well as a MethodSecurityInterceptor“ (MSI) for the application layer. To protect the application layer, a configuration of the MSI is required that determines which permissions the role of the current principal must possess to invoke components for data access and application logic. This can be realized by placing annotations directly in the application source code or through a central AOP configuration. The latter variant is used due to easier maintenance and therefore shown in Listing 1. For the protection of method invocations, a so-called pointcut, which is an entry point for the execution of code formulated as AOP-advices, is associated with a permission, whose presence will be checked by the MSI.

```
<global−method−security>
 <protect−pointcut
   expression=”execution(∗
    edu.kit.minabase.∗CompetenceDAO.get∗(..))”
   access=”PERM_R_Competence”/>
 <protect−pointcut
   expression=”execution(∗
    edu.kit.minabase.∗CompetenceDAO.save(..))”
   access=”PERM_W_Competence”/>
 <protect−pointcut
   expression=”execution(∗
    edu.kit.minabase.∗CompetenceDAO.delete(..))”
   access=”PERM_W_Competence”/>
</global−method−security>
```

Listing 1. Configuration of the MethodSecurityInterceptor.

This restricts the data access to competences by requiring the presence of the ”PERM_R_Competence“ permission for the execution of methods, whose names start with ”get“ and are located within the CompetenceDAO class. On top of that the methods to insert, update or delete a competence requires the ”PERM_W_Competence“-permission to be assigned to the role of the current user. The assignments of permissions to roles can be altered using an administrative interface at runtime. The invocation of methods for modifying and deleting other information entities within *MinaBASE* as well as the execution of application logic can be restricted in a similar fashion. To make sure that the pointcut expressions are executed as intended, the following Listing 2 shows how their effectiveness can be tested using standard JUnit tests.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations={
        "file:src/spring-test.xml"})
public class CompetenceDAOSecurityTests {
  @Autowired
  CompetenceDAO dao;

  @Test(expected = AuthenticationException.class)
  public void unauthenticated_get(){
    dao.get(1L);
  }

  @Test
  public void authenticated_get(){
    login("user", "user");
    Competence c = dao.get(1L);
    assertNotNull(c);
  }

  @Test(expected = AccessDeniedException.class)
  public void insufficient_permissions_save(){
    login("user", "user");
    Competence c = new Competence();
    dao.save(c);
  }

  @Test(expected = AccessDeniedException.class)
  public void insufficient_permissions_delete(){
    login("user", "user");
    dao.delete(1L);
  }

  @Test
  public void sufficient_permissions(){
    login("admin", "admin");
    Competence c = new Competence();
    dao.save(c);
    dao.delete(1L);
  }

  private void login(String u, String pw) {
    UsernamePasswordAuthenticationToken token =
    new UsernamePasswordAuthenticationToken(u, pw);
    SecurityContextHolder.getContext()
      .setAuthentication(token);
  }
}
```

Listing 2. Testing effectiveness of pointcut expressions.

The annotations on the class are necessary to specify the JUnit-Test-Runner as well as to provide the location of the configuration file to the Spring Framework. Given this information, Spring will bootstrap the DI-container in the background, inject the dependency CompetenceDAO into the test class and run all methods marked with the JUnit "@Test"-annotation. The first testcase simulates anonymous access and makes sure that no unauthenticated user can run the "get"-method of the CompetenceDAO component by expecting the test code to raise an exception of type "AuthenticationException", which is part of Spring Security. If this exception is not raised, the testcase is considered to have failed by JUnit and will be reported as such. The next testcase is responsible for proving that authenticated users can execute the "get"-method without causing an exception. The credentials specified in the "login"-method refer to an In-memory AuthenticationProvider, which is used during the tests only. The configuration is shown in the following Listing 3.

```
<sec:authentication-manager>
  <sec:authentication-provider>
    <sec:user-service>
      <sec:user name="user" password="user"
         authorities="PERM_R_COMPETENCE" />
      <sec:user name="admin" password="admin"
         authorities=
           "PERM_R_COMPETENCE, PERM_W_COMPETENCE" />
    </sec:user-service>
  </sec:authentication-provider>
</sec:authentication-manager>
```

Listing 3. In-Memory AuthenticationManager.

By placing this into the test-specific Spring configuration, the runtime duration of the tests can be reduced as no database lookups are required. For reasons of clarity, the permissions are not grouped to roles here, but are added directly to the users of the In-Memory AuthenticationProvider. In a production scenario the authentication process is backed by a chain of AuthenticationProviders which consult different datasources (LDAP-repositories, a relational database or the filesystem). But to minimize the dependencies on the surrounding environment, the In-Memory approach was chosen, as it enables the tests to be run from anywhere. As the configuration shows, the first user only possesses the "PERM_R_Competence“ permissions, while the "admin"-user additionally owns "PERM_W_Competence“. The is important for the rest of the testcase from Listing 3, as the third testcase tries to run the "save"-method, which has been configured to require the higher permission. To ensure the effectivness of the pointcut expression in Listing 1, this testcase is expected to raise an "AccessDeniedException" or to fail otherwise. The following testcase assures the same for the "delete"-method. The final testcase is used to prove that no exception is raised when a user with sufficient permissions runs both methods. While the pointcuts in Listing 1 are expressed in a very generic and broad way, these testcases will assure their effectiveness. By minimizing dependencies to external datasources, this way of testing can be used to prove the fulfillment of the security requirements on the level of application logic and data access.

For securing the presentation layer, combinations of URL-patterns for protected regions and required permissions are specified, which are evaluated by the FSI during the monitoring of HTTP request processing. An excerpt of the necessary configuration is shown in Listing 4.

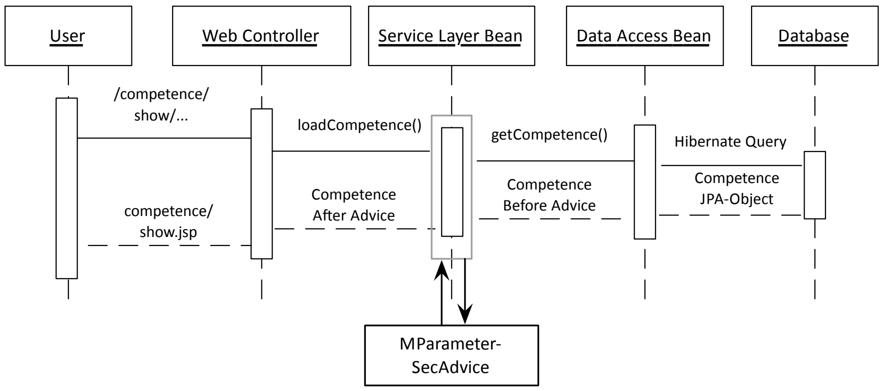


Figure 12. Integration of the MParameterSecAdvice in *MinaBASE*.

```
<http auto-config="false"
  access-denied-page="/denied.jsp">
 <intercept-url
   pattern="/static/*.*" filters="none"/>
 <intercept-url
   pattern="/competence/show/**"
   access="PERM_R_Competence"/>
 <intercept-url
   pattern="/competence/edit/**"
   access="PERM_W_Competence"/>
 <form-login login-page="/login.jsp"
   authentication-failure-url="/login.jsp?error=1"/>
 <logout logout-success-url="/logout.jsp"/>
 </http>
```

Listing 4. Configuration of the FilterSecurityInterceptor.

Due to the URL-patterns being evaluated from top to bottom, the monitoring is at first disabled for static resources to achieve higher performance. Thereafter, permissions for visiting URLs matching the location for display and editing of competences are stated. The final step is the declaration of URLs, to which the AM will redirect unauthenticated users, that try to access a protected resource as well as URLs for authentication failures and the termination of a user's session. These settings ensure that protected areas are not reachable for users that dont possess the required permissions. To improve the user experience, links to sections the user does not have access to should not be displayed in the first place. To achieve this, the generation of HTML needs to be controlled with permissions in mind. Spring Security is bundled with an extension that allows fragments of Java ServerPages (JSP) to be rendered according to the current user's permissions, which is demonstrated in Listing 5.

```
<sec:authorize ifAllGranted="PERM_W_Competence">
 <a href="/competence/edit/...">
    Edit this competence</a>
</sec:authorize>
```

Listing 5. Permission based generation of the user interface.

This JSP-Tag assures that links to the area for editing competences are only rendered to those users that have the "PERM_W_Competence" permission. These three listings show how Spring Security can be used to employ a homogeneous system of permissions that stems from the SDRE permission catalog and covers the entire application architecture from data access to the presentation layer. At runtime these permissions are assigned to roles from the designed RBAC model.

C. Dynamic security aspects

In the previous section, security aspects were considered, which could be fulfilled by statically restricting access to a protected resource by requiring a specific permission to be held by the current principal. While most aspects of the permission catalog are covered by this approach, entries of the constraint-catalog as depicted in Figure 5 cannot be implemented in this fashion, because of their dynamic nature, wich means, these constraints cannot be enforced at build-time, but only at runtime. As an example, the filtering of fabrication-specific TP of a competence's detailed view is used. To avoid code duplication whenever fabrication-specific TP of a *MinaBASE* information entity shall be filtered, this concern is encapsulated into a separate AOP-Advice called "MParameterSecAdvice", whose integration into the method's call flow is illustrated in Figure 12.

A request for the detailed view of a competence is received by a Web-controller, which initiates the data access for the current competence by invoking methods from the service layer. Once this competence is loaded as a database object, the MParameterSecAdvice is hooked into the execution flow using AOP-Weaving. The job of this component is to iterate over the competence's parameter collection and filter out those parameters to which the current principal has no permission. The revised competence object is then returned to the controller which starts the generation of HTML templates and sends the result to the browser.

After explaining the implementation from a high-level point of view, the following will focus on the detailed implementation by describing the source code of the MParameterSecAdvice, which is shown in Listing 6.

```
public class MParameterSecAdvice {
  public void injectAfter(Object ret) {
    SecurityContext ctx =
      SecurityContextHolder.getContext();
    Authentication auth = null;
    boolean fp_permission = false;
    HasAttributes c =
        (HasAttributes)ret;
    Set<CAttribute> cattrs
      = c.getAttributes();
    Set<CAttribute> onlyPPAttrs
      = new HashSet<CAttribute>();

    if (ctx.getAuthentication() != null){
      auth = ctx.getAuthentication();
      GrantedAuthority[] permissions =
        auth.getAuthorities();
      for (int i = 0; i < permissions.length; i++){
        String perm
          = permissions[i].getAuthority();
        if(perm.equals(Constants.FP_PERM)){
          fp_permission = true;
          break;
        }
      }
    }

    if(fp_permission == false){
      if(cattrs != null){
        for(CAttribute current : cattrs){
          if(isFP(current) == false){
            onlyPPAttrs.add(current);
          }
        }
      }
      c.setAttributes(onlyPPAttrs);
    }
  }

  private boolean isFP(CAttribute a){
    Attributetype type = a.getAttribute().getType()
    return type.getId().equals(Constants.FP_ID);
  }
}
```

Listing 6. Parameter filtering constraint within MParameterSecAdvice.

At first, the SecurityContext is retrieved, which contains all information about the current principal. If the current principal can be fetched, its permissions are loaded by invoking the ”getAuthorities”-method from the current Authentication object. The next step is iterating over the list of permissions to find out, whether the current principal has the authority to view the fabrication-related competence parameters. If this is the case, then filtering of parameters can be skipped later on. If this is not the case, the parameters need to be filtered. If the current principal does not possess the required permission, a new collection of parameters is built by iterating over all the parameters of the current competence and inserting only product-related parameters into it. Afterwards the competence is updated by setting the new collection of product-related parameters as the competence's attributes. As the call flow in Figure 12 shows, only the final step, namely rendering of the user interface using HTML templates, is left. As the competence now only consists of product-related parameters, the fabrication-related parameters cannot be rendered to the user. The following Listing 7 shows how the advice is weaved into the execution using a pointcut expression.

```
<bean id="competenceDAO"
  class="edu.kit.minabase.data.CompetenceDAO"/>
<bean id="mParamSecAdvice"
  class="edu.kit.minabase.aop.MParameterSecAdvice"/>

<aop:config>
  <aop:aspect ref="mParamSecAdvice">
    <aop:pointcut id="afterDaoPointcut"
       expression="
     execution(*
  edu.kit.minabase.*
  .CompetenceDAO.getAttributes(..))"/>
    <aop:after-returning returning="ret"
       method="injectAfter"
       pointcut-ref="afterDaoPointcut"/>
  </aop:aspect>
</aop:config>
```

Listing 7. Pointcut expression for the MParameterSecAdvice.

At first the beans ”competenceDAO” and the ”mParamSecAdvice” are declared to the DI container. Subsequently, the pointcut expression states where the advice is to be executed. In our concrete example this expression refers to the ”getAttributes”-method of the CompetenceDAO. As advices can run before, during or after the method referenced in the pointcut expression, the last step is to state when the advice should run. Due to the fact that we need to have the competences populated with all of its parameters, we need to run the advice after the data access code of the CompetenceDAO class. Using the ”returning”-attribute, we specify the name of the method parameter that will be used to tramsit the return value of the data access code to the advice. The value ”ret” corresponds to the method parameter of the ”injectAfter”-method inside of MParameterSecAdvice as can be seen in Listing 6. Inside this method, the ”ret”-variable is assigned and type casted to an interface type called ”HasAttributes”, which is an interface that all information entities within *MinaBASE* implement whenever they can be characterized by parameters. Using this interface instead of a concrete class make this advice useful to all those entities as the implementation is not bound to a single on of them. The advantage of this approach is the fact that the permission based filtering is encapsulated into the MParameterSecAdvice once and can be applied declaratively to multiple application components without code duplication and mixture of concerns by simple configuration in a similar fashion as described in Listing 7.

VII. Related Work

The significance of RE activities when implementing RBAC has led to the development of various RE methodologies. These can be classified as top-down, bottom-up and hybrid approaches. To support enterprise-wide RE, Role Mining has been used to automate parts of the RE activities. In this section, we will discuss these approaches.

Top-down approaches use abstract concepts like work profiles or business functions as a starting point. These are decomposed into smaller parts and mapped onto permissions which enable an aggregation to roles. As an example, in [27] Roeckle et al. define a formal, process-oriented approach for role finding combining an RBAC metamodel with a prodecural model for interfacing business processes in order to automatically derive roles from a process view using a tool called "RoleFinder". In [19] Coyne employs user activities to also identify roles in a top-down manner. Permissions are allocated to roles using the Principle of Least Privilege, as only those permissions are assigned which are necessary to complete a role. Constraints are defined and role hierarchies are built subsequently. Fernandez and Hawkins introduce a semi-formal approach based on textual description of system and user interaction utilizing use-cases [28]. By extending use-cases with rights specification in the form of actors, activity descriptions, preconditions, exceptions and postconditions, all roles and permissions necessary for a system can be determined.

Bottom-up approaches collect permissions as pairs of operations on resources within information systems and use these as a building block for role aggregation using business functions. In [29] Thomsen et al. introduce seven abstract layers to facilitate security management based on RBAC. These layers enable the identification of permissions from objects as well as associated methods and roles for usage by security administration and application developers. Epstein and Sandhu propose the use of the Unified Modeling Language to document each layer introduced by Thomsen et al. in [30].

Hybrid approaches try to combine both technqiues as described above by parallelizing the RE activities or by basing them on an iterative-incremental process [31]. As an example, Epstein and Sandhu propose a conceptual framework in [32] to derive roles in a top-down and bottom-up manner.

RE is useful when the quality of documentation is high and if the amount of tasks, work-profiles or business processes is manageable. In case of dozens of processes, thousands of resources and permissions, a proper decomposition and aggregation to role concepts becomes rather difficult. These conditions have led to Role Mining approaches, in which tools and algorithms from data mining, such as clustering and neural networks are used to derive potential roles automatically [33]. In [34] Fuchs and Pernul introduce HyDRo, which is a tool-supported methodology that facilitates the definition of enterprise-wide roles by combining elements from RE and Role Mining.

As described in detail in section IV, the underlying methodology of this paper is SDRE [4]. As scenarios are used as a building block to derive complete work profiles and associated tasks, but also a mapping of tasks to permissions result in aggregated roles, SDRE - even with our adaptions to it - can be characterized as a hybrid approach. For RE in the context of collaborative knowledge management systems, an alternative configuration of the SDRE mechanism has been used throughout this paper. The approach has been applied with special focus on the use of *MinaBASE* in business processes of MST enterprises and cooperation networks. Instead of using scenarios as the main input, work areas within the business processes of an MST-company are examined, whether they include tasks in which *MinaBASE* can be used to increase added value. To these tasks scenarios are assigned in order to determine which authorizations are needed to fulfill them. Based on this information, a role concept can be derived and further refined. As we only adapt the input variables to the methodology, basic principle of SDRE is preserved, while better results for the creation of the role concept in the context of collaborative knowledge management systems are expected as the adapted methodology is closer to the business processes of a MST-company.

VIII. Conclusion

In this paper, a role concept for the process knowledge database *MinaBASE* has been developed based on a systematic methodology called Scenario-driven role engineering. The implementation of this role concept within an IoC-Framework such as Spring has been demonstrated by utilizing Spring Security and technolgies such as AOP. At first, the *MinaBASE* approach, mechanisms for access control with a special focus on role based access control as well as the Scenario-driven role engineering methodology were introduced. Following this, careful adjustments were made to the inputs of the SDRE process resulting from the background and purpose of *MinaBASE* without hurting the methodology's idea and principles. Then the application of the SDRE process was shown including examples on how to derive a minimal set of permissions enabling each role to fulfill its work profile. In the following section the implementation of the derived role concept using Spring Security is described in detail. Important concepts are Dependency Injection and AOP, as they enable Spring Security to ensure static and dynamic security requirements across the entire application architecture. For the implementation of security requirements that can be decided at runtime only, an example was given in order to prevent the disclosure of fabrication-specific parameters for non-authorized persons.

References

[1] D. Kimmig, A. Schmidt, K. Bittner, and M. Dickerhof, "Application of Scenario-driven Role Engineering to the MinaBASE Process Knowledge Database," in *SECURWARE 2011, Proceedings of the Fifth International Conference on Emerging Security Information, Systems and Technologies*. 978-1-61208-146-5, 2011, pp. 125–132.

[2] U. Hansen, C. Germer, S. Büttgenbach, and H. Franke, "Rule based validation of processing sequences," in *Techn. Proc. MSM*, 2002.

[3] D. F. Ferraiolo and R. Kuhn, "Role-based access control," in *Proceedings of 15th NIST-NCSC National Computer Security Conference*, October 1992, pp. 554–563.

[4] G. Neumann and M. Strembeck, "A scenario-driven role engineering process for functional RBAC roles," in *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM Press, 2002, pp. 33–42.

[5] I. Nonaka and H. Takeuchi, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, USA, 1995.

[6] M. Dickerhof, "Prozesswissensmanagement für die Mikrosystemtechnik." 2003.

[7] M. Dickerhof and A. Parusel, "Bridging the Gap—from Process Related Documentation to an Integrated Process and Application Knowledge Management in Micro Systems Technology," *Micro-Assembly Technologies and Applications*, vol. 260, pp. 109–119, 2008.

[8] M. Dickerhof, O. Kusche, D. Kimmig, and A. Schmidt, "An ontology-based approach to supporting development and production of microsystems," *Proc. of the 4th Internat. Conf. on Web Information Systems and Technologies*, 2008.

[9] D. Kimmig, A. Schmidt, K. Bittner, and M. Dickerhof, "Modeling of Microsystems Production Processes for the MinaBASE Process Knowledge Database Using Semantic Technologies," in *Proc. of the The 3rd Internat. Conf. on Information, Process, and Knowledge Management*, 2011, pp. 17–23.

[10] R. C. David F. Ferraiolo, D. Richard Kuhn, *Role-Based Access Control*, 1st ed., ser. Computer Security Series. Artech House, Inc., 2003.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, no. 3, pp. 224–274, 2001.

[12] E. Glaser, "A brief description of privacy measures in the Multics operating system," in *Proceedings of AFIPS SJCC*, vol. 30. Montvale, N.J.: AFIPS Press, 1967.

[13] M. Benantar, *Acces Control Systems - Security, Identity Management and Trust Models*, 1st ed. Springer Science+Business Media, Inc., 2006.

[14] M. D. S. Jerome H. Saltzer, "The protection of information in computer systems," in *Proceedings of fourth ACM Symposium on Operating System Principles*, 1975.

[15] *Trusted Computer System Evaluation Criteria - DoD 5200.28 Std*. Department of Defense, 1985.

[16] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," Mitre Corporation Bedford, Tech. Rep., March 1973.

[17] K. Biba, "Integrity considerations for secure computer systems," Mitre Corporation Bedford, Tech. Rep., April 1977.

[18] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn, "Role Based Access Control (RBAC): Features and Motivations," in *In Proceedings of 11th Annual Computer Security Application Conference*. IEEE Computer Society Press, 1995, pp. 241–48.

[19] E. J. Coyne, "Role Engineering," in *RBAC '95: Proceedings of the first ACM Workshop on Role-based access control*. New York, NY, USA: ACM Press, 1996, p. 4.

[20] HL7 Security Technical Committee , "HL7 Role Based Access Control (RBAC) Role Engineering Process," January 2005.

[21] K. Boehm, W. Engelbach, J. Härtwig, M. Wilcken, and M. Delp, "Modelling and implementing pre-built information spaces. architecture and methods for process oriented knowledge management," *Journal of Universal Computer Science*, vol. 11, no. 4, pp. 605–633, 2005.

[22] B. Alex and L. Taylor, "Spring Security Reference Documentation," URL: http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html, accessed: 2012-07-12.

[23] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," URL: http://martinfowler.com/articles/injection.html, Januar 2004, accessed: 2012-07-12.

[24] R. Johnson and J. Hoeller, *J2EE Development without EJB*, 1st ed., ser. Expert on-on-one. Wiley Publishing, Inc., 2004.

[25] E. Dijkstra, *A discipline of programming*. Englewood Cliffs, NJ: Prentica Hall, 1976.

[26] O. Böhme, *Aspektorientierte Programmierung mit AspectJ 5*. dpunkt.verlag, 2006.

[27] H. Roeckle, G. Schimpf, and R. Weidinger, "Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization," in *Proceedings of the fifth ACM workshop on Role-based access control*, ser. RBAC '00. New York, NY, USA: ACM, 2000, pp. 103–110.

[28] E. B. Fernandez and J. C. Hawkins, "Determining role rights from use cases," in *Proceedings of the second ACM workshop on Role-based access control*, ser. RBAC '97. New York, NY, USA: ACM, 1997, pp. 121–125.

[29] D. Thomsen, D. O'Brien, and J. Bogle, "Role based access control framework for network enterprises," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*, Dec 1998, pp. 50 –58.

[30] P. Epstein and R. Sandhu, "Towards a UML based approach to role engineering," in *Proceedings of the fourth ACM workshop on Role-based access control*, ser. RBAC '99. New York, NY, USA: ACM, 1999, pp. 135–143.

[31] A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett, "Observations on the role life-cycle in the context of enterprise security management," in *Proceedings of the seventh ACM symposium on Access control models and technologies*, ser. SACMAT '02. New York, NY, USA: ACM, 2002, pp. 43–51.

[32] P. Epstein and R. Sandhu, "Engineering of Role/Permission Assignments," in *Proceedings of the 17th Annual Computer Security Applications Conference*, ser. ACSAC '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 127–.

[33] J. Vaidya, V. Atluri, and Q. Guo, "The role mining problem: finding a minimal descriptive set of roles," in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT '07. New York, NY, USA: ACM, 2007, pp. 175–184.

[34] L. Fuchs and G. Pernul, "HyDRo – Hybrid Development of Roles," in *Information Systems Security*, ser. Lecture Notes in Computer Science, R. Sekar and A. Pujari, Eds. Springer Berlin / Heidelberg, 2008, vol. 5352, pp. 287–302.