Network of Excellence

Deliverable D5.1

Common Body of Knowledge
Abstract

Interdisciplinary communities involve people and knowledge from different disciplines in addressing a common challenge. Differing perspectives, processes, methods, tools, vocabularies, and standards are problems that typically arise in this context.

We present in this deliverable an approach to support bringing together disciplines based on a common body of knowledge (CBK), in which knowledge from different disciplines is collected, integrated, and structured. The novelty of our approach is twofold: first, it introduces a CBK ontology, which allows one to semantically enrich contents in order to be able to query the CBK in an elaborate way afterwards. Second, it heavily relies on user participation in building up a CBK, making use of the Semantic MediaWiki as a platform to support collaborative writing. The CBK ontology is backed by a conceptual framework, consisting of concepts to structure the knowledge and to provide access options to it, and a mechanism to build up a common terminology. To ensure a high quality of the provided contents and to sustain the community's commitment, we further present organizational means as part of our approach.

Keyword List

interdisciplinarity, common body of knowledge, knowledge management, software engineering, security engineering, services computing.
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Type of Change</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Version to be reviewed</td>
<td>Eicker, Heisel, Schmidt, Schwittek (UDE)</td>
</tr>
<tr>
<td>0.2</td>
<td>Corrections based on feedback from Wouter Joosen, Marinella Petrocchi, and Pieter Philippaerts</td>
<td>Schmidt (UDE)</td>
</tr>
<tr>
<td>1.0</td>
<td>Further corrections</td>
<td>Schmidt (UDE)</td>
</tr>
<tr>
<td>1.1</td>
<td>Further corrections</td>
<td>Schwittek (UDE)</td>
</tr>
</tbody>
</table>
Table of Contents

**LIST OF FIGURES** ................................................................................. 7

**LIST OF TABLES** .............................................................................. 9

1 **INTRODUCTION** ............................................................................. 11

2 **CONCEPTUAL DESIGN OF THE COMMON BODY OF KNOWLEDGE** .......... 13
   2.1 Scope and Functionality ...................................................................... 13
     2.1.1 The Role of the CBK in NESSoS .................................................. 13
     2.1.2 Use Cases .................................................................................. 13
   2.2 Knowledge Base Structure ................................................................. 14
     2.2.1 Conceptual Framework ............................................................... 14
     2.2.2 An Ontology for a Common Body of Knowledge ....................... 16
   2.3 Technology ...................................................................................... 20
     2.3.1 Platform Evaluation ................................................................. 20
     2.3.2 Our Choice ............................................................................... 21
     2.3.3 Realization ............................................................................... 22
   2.4 Establishing Processes ....................................................................... 22
     2.4.1 Three Phases ........................................................................... 23
     2.4.2 Quality Assurance .................................................................. 23
   2.5 Related Work ................................................................................... 24

3 **CONTENT OF THE COMMON BODY OF KNOWLEDGE** .................... 25

4 **CONCLUSIONS** .............................................................................. 37
   4.1 Summary ....................................................................................... 37
   4.2 Further Development and Exploitation of the CBK ......................... 37

**BIBLIOGRAPHY** .................................................................................. 39

A **ONTHELOGY PROPERTIES** .............................................................. 41

B **PUBLISHED PAPER [16]** .............................................................. 45
List of Acronyms

**BOK**  Body Of Knowledge
**CBK**  Common Body of Knowledge
**CSCW** computer supported collaborative work
**EU**  European Union
**KA**  Knowledge Area
**KO**  Knowledge Object
**NESSoS**  Network of Excellence on Engineering Secure Future Internet Software Services and Systems
**NoE**  Network of Excellence
**OWL**  Web Ontology Language
**QA**  Quality Assurance
**SE**  Software Engineering
**SDLC**  Software Development Lifecycle
**SMW**  Semantic MediaWiki+
**SWEBOK**  Software Engineering Body of Knowledge
**UML**  Unified Modeling Language
List of Figures

Figure 2.1: CBK ontology (presented as a UML class diagram) .................................................. 17
Figure 3.1: Development of KOs according to KO types over time ........................................... 25
Figure 3.2: Development of KAs over time .................................................................................. 27
Figure 3.3: Development of KO's maturity over time ................................................................. 27
Figure 3.4: Example method: UWE (part 1 of 3) ....................................................................... 28
Figure 3.5: Example method: UWE (part 2 of 3) ....................................................................... 29
Figure 3.6: Example method: UWE (part 3 of 3) ....................................................................... 30
Figure 3.7: Example notation: SecureUML (part 1 of 2) ......................................................... 31
Figure 3.8: Example notation: SecureUML (part 2 of 2) ......................................................... 32
Figure 3.9: Example tool: MagicUWE (part 1 of 3) ................................................................. 33
Figure 3.10: Example tool: MagicUWE (part 2 of 3) .............................................................. 34
Figure 3.11: Example tool: MagicUWE (part 3 of 3) .............................................................. 35
Figure 3.12: Example technique: Bayesian belief network ....................................................... 36
List of Tables

Table 2.1: Method Properties (excerpt) ............................................................. 18
Table 2.2: Tool Properties (excerpt) ................................................................. 18
Table 2.3: KnowledgeObject Properties (excerpt) .............................................. 19
Table 2.4: Activity Properties ........................................................................... 19
Table 2.5: LearningTrailStep Properties ............................................................ 19

Table 3.1: Overview of all KOs ordered by KO types ........................................ 26

Table A.1: KnowledgeObject Properties ............................................................ 41
Table A.2: Method Properties ............................................................................ 42
Table A.3: Activity Properties ........................................................................... 42
Table A.4: Tool Properties ................................................................................ 42
Table A.5: Notation Property ............................................................................. 42
Table A.6: Function Properties ......................................................................... 42
Table A.7: UsageExample Properties ................................................................. 42
Table A.8: LearningTrail Property ..................................................................... 43
Table A.9: LearningTrailStep Properties ............................................................ 43
Table A.10: Link Properties ............................................................................... 43
Table A.11: Image Properties ............................................................................ 43
Table A.12: Publication Properties ..................................................................... 43
1 Introduction

In this deliverable, we present the basic concepts we developed to realize the common body of knowledge (CBK) and its current implementation using the SMW+ framework.

The CBK is intended to support one of the main goals of the NESSoS NoE, namely to create a long-lasting research community on engineering secure software services and systems and to bring together researchers and practitioners from security engineering, service computing, and software engineering.

One of the main challenges of the NESSoS CBK is that it must support interdisciplinary work. Bringing together different disciplines harbors a number of problems, such as the combination of differing perspectives, processes, methods, tools, vocabularies, and standards. Moreover, these problems have to be considered with respect to multiple dimensions such as research and practice, which further complicates the situation.

Our CBK is conceived in such a way as to overcome the aforementioned problems. While existing bodies of knowledge (BOKs) like the Software Engineering Body of Knowledge (SWEBOK) [8] solely rely on books or hypertext systems as a medium, our CBK provides several advantages, such as improved flexibility and access possibilities for its users. In fact, the CBK introduces an ontology that allows users to semantically enrich content. The CBK ontology is backed by a conceptual framework consisting of three main pillars: The structuring of knowledge from different disciplines that the CBK collects and integrates, such as specific techniques, methods, industrial best practices, and tools, constitutes the first pillar. To consolidate an interdisciplinary community, we need a common understanding of the key concepts as well as a common vocabulary of the different disciplines. The CBK introduces a common terminology, i.e., necessary basic notions and relations between them. The common terminology is the second pillar, and it allows us to create a mapping between discipline-specific terminology and the notions of the common terminology. The last pillar comprises means to group knowledge in order to provide a variety of access options to the knowledge for a wide range of different target groups. Since the CBK’s content is semantically enriched, it can be precisely queried in order to, e.g., find appropriate methods to analyze and solve a given problem. Moreover, the CBK can be used to contribute to identifying possible research gaps, weaknesses, and interesting directions for future research.

Another difference to existing BOKs is that our CBK heavily relies on user participation realized through a wiki platform. Consequently, the CBK supports collaborative writing and provides mechanisms to build up and update the CBK. Since the CBK will be opened for the public, our approach is complemented by organizational means considering aspects such as quality assurance to ensure a high quality of content.

The deliverable is organized into two main chapters: in Chapter 2 we present the conceptual design of the CBK. We outline use cases for the CBK, we present the ontology underlying our CBK, we discuss our choice of technology for realizing the CBK, we introduce organizational factors of the CBK, and we present related work. Chapter 3 shows the current state of the CBK, where we present some statistics and examples of the actual content of the CBK.

Finally, we conclude and raise ideas for future work in Chapter 4.
2 Conceptual Design of the Common Body of Knowledge

2.1 Scope and Functionality

We briefly place the CBK in the context of the NESSoS NoE in Sect. 2.1.1, and we develop use cases for the CBK and identify knowledge to be stored in it with respect to NESSoS in Sect. 2.1.2.

2.1.1 The Role of the CBK in NESSoS

The major goal of NESSoS is to lay the foundation for a long-lasting research community on engineering secure software-based Future Internet services and systems. Forming such a community became necessary, because research on this topic is distributed among several fields, each working quite independently from the others. NESSoS brings together researchers from software engineering, security engineering, and service engineering from different countries of the EU. For an NoE such as NESSoS, integration and sustainability of new research communities is an important aim. In practice, this means that research activities of all partners from academia and industry are re-addressed, harmonized and integrated, leading to an interdisciplinary and international research setting with a high demand in transferring knowledge from research into practice and triggering research from practical challenges. An impact on training and education activities in Europe is expected as well.

The project itself is structured accordingly: besides the biggest area of “Integrated Research” (concerning the allocated person months) and the two blocks “Spreading of Excellence” and “Consortium Management”, we find another big area addressing integration and sustainability issues of the network named “Network Integration”. The main task of the latter area is to integrate research communities and agendas. It therefore contains a mobility program for researchers, a virtual research lab, a workbench in which tools and methodologies are integrated, and an integrated common body of knowledge, which is the main subject of this deliverable.

The CBK in its current form brings together researchers and practitioners through a wiki-based platform for collaborative editing of knowledge in the fields of security, services, and software engineering. The CBK not only collects the relevant knowledge but it also integrates and structures it in order to support the creation of a long lasting research community on engineering secure software-based Future Internet services and systems.

2.1.2 Use Cases

As outlined in Sects. 1 and 2.1.1, we aim at integrating and structuring overlapping knowledge areas from security engineering, services computing and software engineering (SE) in order to support the creation of a long-lasting research and practice community on engineering secure software services and systems. Having in mind that the CBK should serve as a toolkit and a handbook, the identification of relevant use cases constitutes the center of analyzing the functionality and the scope of the CBK. Consequently, we present typical use cases for the CBK in this section. We assume practice and research as well as administrative perspectives for the uses cases. This is reflected in the actors we identified to interact with the CBK:

**Researcher** belongs to the research community of the CBK’s domain; contributes knowledge to the CBK and retrieves knowledge from it;

**Practitioner** similar to researcher, but belongs to the practice community of the CBK’s domain;

**Administrator** is responsible for administrative aspects; does not add knowledge to the CBK;

**Quality agent** is responsible for quality assurance; might correct knowledge contained in the CBK and improve its quality.

Note that each actor defines a role, i.e., a set of individuals, and these roles might not be disjoint. For instance, a quality agent might be a researcher at the same time.
The CBK supports different content categories, two of them are the categories method and tool (see Sect. 2.2.1 for details). As examples, we present use cases identified for methods and tools in the following. However, similar use cases also apply to the other knowledge categories.

1. **Manage** (*add, delete, modify*) methods and tools.
   - **Actors:** researchers (add and modify), practitioners (add and modify), administrators (delete and modify)

2. **Browse** methods and tools.
   - **Actors:** researchers, practitioners, administrators, quality agents

3. **Identify** methods and tools that fulfill specific criteria.
   - **Actors:** researchers, practitioners

4. **Compare** methods and tools according to specific criteria.
   - **Actors:** researchers, practitioners
   - This use case is highly related to deliverables D2.1 [7] and D2.2 [6], as they focus on the comparison of different tools.

5. **Get recommendations** for methods and tools according to specific criteria.
   - **Actors:** practitioners

6. **Get an overview** of existing methods and tools for a specific purpose.
   - **Actors:** researchers, practitioners

7. **Identify gaps** in practice and in the current research landscape.
   - **Actors:** researchers

Use case 1 summarizes the relevant operations for adding, deleting, and correcting knowledge stored in the CBK with respect to the different actors. Use case 2 is the simplest (non-manipulating) query. The use cases 3 to 5 refer to “specific criteria”, i.e., querying with respect to these criteria has to be explicitly supported by the CBK. For example, we identified the following criteria for methods: name, author, notation used, degree of formality, relevance for national and international standards and in this context the relevance for certification and auditing, publication, maturity level, and different categories such as the **SDLC (software development lifecycle)** phase the method supports. We identified similar criteria for tools. The “specific purpose” mentioned in use case 6 can be interpreted as an additional criterion. Use case 7 is a query that involves a special threshold to define what “gap” means. For instance, gap can mean that there does not exist any method supporting a specific SDLC phase, or that there exists a maximum number of methods only.

Based on the use cases specified in this section, we present the conceptual and technical development of a CBK in the next section.

### 2.2 Knowledge Base Structure

The basic idea behind the structural concept of the CBK is to be able to link arbitrary content classes with each other and to allow actors to browse content along the links between the actual content. Furthermore, the aim is to provide several access possibilities to the CBK, each customized to the target audiences and use cases the CBK addresses. In the following, we introduce the core concepts in Sect. 2.2.1 as well as a formalization of these core concepts as an ontology the CBK is based upon in Sect. 2.2.2.

#### 2.2.1 Conceptual Framework

The conceptual framework consists of four basic concepts: **knowledge objects**, **knowledge areas**, **learning trails**, and the **common terminology**. In our case **knowledge objects** are considered to be direct contributions to a body of knowledge of an engineering discipline, whereas a **knowledge area** and a **learning trail** are used for structuring purposes and provide a variety of access possibilities to the CBK. The aim of the **common terminology** is to provide an instrument to improve the comprehensibility of the contents for the
different stakeholder groups addressed by the CBK. All these concepts can be considered as the building blocks of the CBK. We describe them in more detail in the following sub-sections.

Knowledge Objects

A knowledge object (KO) is a fundamental entity of the CBK. Each KO can be linked to other KOs, resulting in a network of KOs, which as a whole can be considered as a representation of a body of knowledge of a certain discipline. For the initial version of the CBK, we derived four KO types, which we considered as typical types of contributions to a body of knowledge of an engineering discipline, engineering secure software and services in particular. These KO types are methods, tools, techniques, and notations. We consider them as a starting point, open for extensions in the future. Methods define a set of activities, which in combination with a notation or a number of notations are used to tackle problems in engineering secure software and services in a systematic way. For example, the concrete method “Security Engineering Process using Patterns” [14] is an individual of the KO type method. Tools support a software engineer in achieving a development goal in an (at least partially) automated way. For example, a UML (Unified Modeling Language)1 editor is a concrete tool that supports generating UML diagrams. Patterns provide a form through which knowledge about recurring development tasks is codified. For example, design patterns [12] are a quite popular pattern type in SE, besides many other types. A notation defines symbols, a syntax, and semantics to express relevant artifacts. One popular notation among software engineers is UML, which is used to describe software systems.

Knowledge Areas

We adopt the concept of knowledge areas (KA) from the SWEBOK [8] for our CBK. KAs span the research field as a whole, dividing it into smaller parts and providing an easier access to subjects of interest. The SWEBOK was created in a long process from 1998 to 2003, involving approximately 500 reviewers from 42 countries in a first phase and over 120 reviewers from 21 countries in a second phase. One main result is the worldwide accepted common understanding of what is today viewed as SE. This includes the differentiation of the field into ten KAs on which we want to base our KAs, e.g., software requirements and software design as listed in the corresponding class box. We took this decision because we regard the field of engineering secure software and services as a supplement of SE and therefore concerning all SE KAs. In addition, we introduce KAs specific to the fields of security and services based on standard literature. For instance, we introduce the KAs risk analysis and privacy as presented in Anderson's Security Engineering book [2]. Note that the list of KAs depicted in the corresponding class box is not yet complete. It will evolve over time at least until the end of the NESSoS project.

Each KA consists of a description providing an overview of the KA and its scope, as well as relationships to other KAs. KAs are detailed further into sub-areas, topics and sub-topics. Each topic or sub-topic contains the following three items: A short state-of-the-art description of the topic/sub-topic, links to KOs supporting the topic/sub-topic and a list of the most relevant publications for further reading.

Learning Trails

Learning trails are a structuring element meant to provide access to the common body of knowledge on engineering secure software and services for different target groups. This idea is based on the fact that content has to be prepared in accordance to the background of the reader. An expert in this topic area expects more detailed information, whereas a non-expert needs more contextual information in order to be able to understand. Learning trails are therefore written and categorized along different expert levels indicating, e.g., what prior knowledge is required to understand the content. Another differentiation is made concerning the reader’s background: if (s)he is from research or from practice. Learning trails are realized by moderated tours which guide the reader through a set of KOs, which are considered to be part of a certain topic. Each step builds upon the previous step and gives a successive introduction into a topic with respect to the reader’s expert level and background. The overall aim of this approach is to provide access to the CBK for a broad spectrum of people, regardless of whether the reader is a student, an experienced expert, a practitioner, or a researcher.

1http://www.omg.org/spec/UML/2.3/
Common Terminology

The aim of a common terminology is to enable a community to speak the same language; or at least to simplify the translation of a term to another domain with the help of a common reference or common term as we want to call it in the following. A common term is a term with a meaning on which an agreement was reached within the community. With the common terminology, we therefore introduce an instrument for defining a common term with a certain meaning and for relating different terms with the same or a similar meaning to this common term. In the opposite direction, the common terminology serves the purpose of a dictionary from which synonyms and translations can be queried. A term does not always have the same exact meaning of another similar term, so that deviations to the meaning of the common term must be made explicit. In the CBK, this is realized by three different relationship types. A term’s meaning is either synonymical, broader or narrower in relation to another term’s meaning.

The common terminology is created bottom-up. An ontology of terms of the domain “engineering secure software and services” is initially created on basis of existing CBK contents and usage of terms after a period of time. It is then proposed to the community and refined within regular feedback cycles.

We formalize the presented CBK concepts using a special CBK ontology in the following.

2.2.2 An Ontology for a Common Body of Knowledge

Ontologies are used to capture knowledge about some domain of interest. We use the OWL (Web Ontology Language) notation and terminology in the following. An ontology describes concepts and relations between them. In OWL, a concept is specified in terms of a class, i.e., a set of individuals. An individual represents a concrete object in the domain in which we are interested. In general, in OWL a relation is specified as a property, which represents a binary relation between individuals. Object properties and data properties are distinguished: the former represent relations between individuals or classes, and the latter represent relations between individuals or classes and data values. Both, classes as well as properties can be structured to build complex constructs.

As an example, we present the current ontology underlying the NESSoS CBK in Fig. 2.1. This CBK ontology can be considered as the explicit description of the domain of creating a CBK for engineering secure software and services. The depicted CBK ontology is a snapshot from the time this deliverable is written since the CBK is an artifact that is constantly evolving and enhanced by the community participating in this CBK endeavor. The CBK ontology is developed in close collaboration with researchers from LMU, who are responsible for tool and method integration and comparison within WP2. For that reason, the CBK ontology is partly contained in deliverable D2.1 [7], too.

Since the interrelated network of KOs is supported by the CBK ontology, it allows the user to browse or query this network and to possibly uncover new correlations (see use cases in Sect. 2.1.2).

We present details on the ontology and its creation in the following sub-sections.

Ontology Creation

The CBK ontology which we present in the following sections was created taking two forces into account: the use cases defined in Sect. 2.1.2 and the format for describing design patterns [12].

Along the use cases we started analyzing typical KOs, e.g., looking at concrete descriptions of tools and methods, and we developed ontology classes, data properties, and object properties (relations). For each entity of our domain we had to decide to either model it as a class or as a property. For example, we decided to model an activity of a method as a class, because an activity from our use case point of view is defined as an aggregation of name, input and output artifacts, etc., for which one property would not be sufficient. We created a criteria catalogue in order to systemize the decision process for each entity.

Design patterns in the field of SE usually follow a typical structure, essentially consisting of the parts “context description”, “problem description”, “solution description” and “consequences description”. We decided to structure the descriptions of Method, Tool, Technique, and Notation. This decision was taken due to the following reasons:

- Since design patterns are a well-known and widely accepted reuse concept in the SE community, the pattern-based KO structure is easy to recognize and thus improves understanding.
• Explicitly specifying the problem solved by a KO individual and the corresponding context supports access for a broader audience.
• The explicit problem and solution descriptions represent a first step towards an engineering catalogue following a problem-solution approach.
• Using a standardized structure improves the comparability of KOs.

An overview of all pattern-related properties with short descriptions is given in Tab. 2.3.

**Ontology Classes**

The result of the ontology creation process is depicted in Fig. 2.1, which we describe in detail in the following. We use a UML class diagram to represent the underlying text-based OWL notation. In the CBK ontology, each concept from our conceptual framework described in the previous section is represented by an ontology class, namely KnowledgeObject (see Sect. 2.2.1), KnowledgeArea (see Sect. 2.2.1), LearningTrail (see Sect. 2.2.1), and CommonTerm (see Sect. 2.2.1). They are highlighted by a dark gray background color. The KnowledgeObject class is further specified by the more specialized subclasses Tool, Method, Technique, and Notation representing different KO types as described in Sect. 2.2.1. They are highlighted by a light gray background color. Furthermore, classes are introduced that mainly serve...
as supplements for the main concept classes. On the one hand, we have the supplement classes Publication, UsageExample, Image, Link, Function, LearningTrailStep, Role, TargetGroup, and Activity as a container for complex data structures, which cannot be modeled as properties. On the other hand, the classes MaturityLevel and SDLCPhase represent enumeration types, each containing a predefined set of individuals. This means that they define a fixed range of values, which can be referenced by other classes. For instance, the individual “Security Engineering Process based on Patterns” of the class Method references the predefined individual “(2) StableOrEvaluated” from the class MaturityLevel out of in total four possible values.

### Ontology Properties

Our ontology defines relations in terms of object properties (between classes) and data properties (between classes and primitive data types). Appendix A summarizes all properties currently realized in the NESSoS CBK. In the following, we present different kinds of properties as examples.

Tab. 2.1 shows properties central to the KO sub-class Method, i.e., this class is the domain of these properties. A method can be supported by a tool or a number of tools (HasMethodSupport). A method might consist of sub methods (HasMethodSubMethod), and it can be the successor of other methods (HasMethodSuccessor). Each method consists of at least one activity (HasMethodActivity).

Our ontology describes properties for the KO sub-class Tool, some of which we present in Tab. 2.2. A tool can be used with other tools (HasToolUses).

Some properties are common to all kinds of KOs. Therefore, we introduce these properties with the class KnowledgeObject as a domain in Tab. 2.3. We describe each KO by its context (HasKOContextDescription), problem (HasKOProblemDescription), solution (HasKOSolutionDescription), and consequences (HasKOConsequencesDescription). A KO can be associated with KAs (HasKOKA). A KO might be a synonym term (HasKOSynonymTerm), a broader term (HasKOBroaderTerm), or a narrower term (HasKONarrowerTerm) compared to a common term. A KO might be based on a notation (HasKONotation) or might be supported by a tool (HasKOToolSupport). A KO has a certain maturity level expressed by the property HasKOMaturityLevel pointing at one out of four levels.

As an example for those classes that we introduced for structuring purposes, we present some Activity class properties in Tab. 2.4. An activity must have a name (HasActivityName) and a description (HasActivityDescription). An activity might use input artifacts (HasActivityInput) and can generate output artifacts (HasActivityOutput). These properties represent textual descriptions; hence they are String properties.

As another example for a class with a structuring purpose, we present the class LearningTrailStep using the data properties shown in Tab. 2.5. The first learning trail step is associated with the LearningTrail class. Except the first and the last step, learning trail steps have a predecessor (HasLearningTrailStepPredecessor) and successor (successorHasLearningTrailStepSuccessor) step. Moreover, each learning trail step has a description (HasLearningTrailStepDescription) and might refer to a number of KOs (HasLearningTrailStepReferenceToKO).

In the following section, we present knowledge queries utilizing the various properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasMethodSupport</td>
<td>Method</td>
<td>0..*</td>
</tr>
<tr>
<td>HasMethodSubMethod</td>
<td>Method</td>
<td>0..*</td>
</tr>
<tr>
<td>HasMethodPredecessor</td>
<td>Method</td>
<td>0..*</td>
</tr>
<tr>
<td>HasMethodSuccessor</td>
<td>Method</td>
<td>0..*</td>
</tr>
<tr>
<td>HasMethodActivity</td>
<td>Activity</td>
<td>1..*</td>
</tr>
</tbody>
</table>

**Table 2.1: Method Properties (excerpt)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasToolUses</td>
<td>Tool</td>
<td>0..*</td>
</tr>
</tbody>
</table>

**Table 2.2: Tool Properties (excerpt)**
Knowledge Queries

The properties defined in our ontology allow us to specify various knowledge queries realizing the use
cases presented in Sect. 2.1.2.

Use cases 3 and 6 (identify and overview according to specific criteria) in Sect. 2.1.2 can be realized
by queries of the form

Property\_Name \text{ value} \text{ Criterion}

For example, the query

\text{HasKO}Maturity\_Level \text{ value ProofOfConceptOrPrototype}

retrieves all methods validated by a proof of concept or prototype implementation. Use case 5 (get rec-
ommendations according to specific criteria) can be regarded as a conjunction of several queries of the
aforementioned type. For instance, the query

\text{HasKO}Maturity\_Level \text{ value ProofOfConceptOrPrototype and }
\text{HasKO}Relevance\_For\_Standards \text{ value CommonCriteria}

Table 2.3: KnowledgeObject Properties (excerpt)

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasKOContextDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasKOPProblemDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasKOSolutionDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasKOConsequencesDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasKOKA</td>
<td>KnowledgeArea</td>
<td>1..*</td>
</tr>
<tr>
<td>HasKOSynonymTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
</tr>
<tr>
<td>HasKOBroaderTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
</tr>
<tr>
<td>HasKONarrowerTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
</tr>
<tr>
<td>HasKONotation</td>
<td>Notation</td>
<td>0..*</td>
</tr>
<tr>
<td>HasKOToolSupport</td>
<td>Tool</td>
<td>0..*</td>
</tr>
<tr>
<td>HasKOMaturityLevel</td>
<td>MaturityLevel</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.4: Activity Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasActivityName</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasActivityDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasActivityInput</td>
<td>String</td>
<td>0..*</td>
</tr>
<tr>
<td>HasActivityOutput</td>
<td>String</td>
<td>0..*</td>
</tr>
</tbody>
</table>

Table 2.5: LearningTrailStep Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Mult.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasLearningTrailStepPredecessor</td>
<td>LearningTrailStep</td>
<td>0..1</td>
</tr>
<tr>
<td>HasLearningTrailStepSuccessor</td>
<td>LearningTrailStep</td>
<td>0..1</td>
</tr>
<tr>
<td>HasLearningTrailStepDescription</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>HasLearningTrailStepReferenceToKO</td>
<td>Knowledge-Object</td>
<td>0..*</td>
</tr>
</tbody>
</table>
further restricts the previous example query by considering only methods that support the Common Criteria\textsuperscript{3} standard. The other use cases rely on the technical realization of the CBK. Especially the use cases 4 and 7 (compare and identify research and practice gaps) have to be implemented using special technical means (see Sect. 2.3).

After the presentation of the ontology, we continue with the technical realization of the CBK.

2.3 Technology

After defining use cases and conceptualizing structures for the CBK, we looked for adequate platform types. We wanted to achieve the greatest possible match of built-in functionality of a certain product with our requirements for a CBK derived from the use cases. A major requirement was to minimize efforts to program missing functionality. We describe the conducted platform evaluation and the criteria on which we took our final decision in Sect. 2.3.1. We then briefly present the chosen platform in Sect. 2.3.2 and describe in Sect. 2.3.3 how our concrete knowledge base structure presented in Sect. 2.2 is realized with our platform choice.

2.3.1 Platform Evaluation

Since our CBK has its origin in a NoE with the specific task of building a long-lasting research community and integrating different research efforts (see Sect. 2.1.1), major themes under which we chose our platform types was not only knowledge management, but also collaboration and sustainability. We therefore looked at platforms from the platform types knowledge/content/document management systems, repositories and portals, but also at the more collaboration-centric platform types computer supported collaborative work (CSCW), eLearning, social networks, and wikis. We then preselected five platforms from different platform types in a first phase by considering mandatory requirements such as open source licensing, collaboration features, and easy creation and management of KAs, KOs, and relationships between them. But, also past experiences of our team members concerning the use and development of the considered systems played a role. The five candidates were Liferay (portal), Elgg (social network), DSpace and MyCoRe (both document management systems), and the Semantic MediaWiki (wiki). All underwent a more detailed investigation with respect to the following requirements:

1. easy and intuitive user interface with respect to the creation and linking of KOs, but also with respect to finding relevant information
2. support of collaborative content creation (versioning of content, commenting, rating, discussing, abuse reporting, etc.)
3. support of access rights management
4. elaborate search functionality, at least full text search, better faceted search, allowing an exploration of information by filtering
5. possibility to generate documents from the CBK content (e.g., security engineering catalogue, CBK as a book) rather than having to rewrite everything from scratch
6. support of an easily extendable data model to be used with an evolving ontology
7. mechanisms to create KAs and learning trails to group KOs
8. mechanisms to build up a common terminology
9. support of an easy creation or generation of forms to allow a convenient input of data
10. open interfaces to be able to integrate the CBK with development tools and other platforms

The Liferay\textsuperscript{4} portal system fulfills most of our requirements. Originally, Liferay as a portal system intends to integrate different information sources to be presented to the user within one or more portal pages. Furthermore, it serves as a runtime environment for mini applications written in a standardized way (portlet specification), which can be integrated into portal pages as well. While on the one hand Liferay is a platform for different information sources and standardized mini-applications, on the other hand it ships with an enormous amount of prepackaged mini-applications such as a wiki, a forum, a blog,

\textsuperscript{3}http://www.commoncriteriaportal.org/
\textsuperscript{4}http://www.liferay.com
a messaging module, etc., which can be integrated into a portal installation on demand. All content from these prepackaged mini-applications is searchable, and the access rights management is quite elaborate. The two major drawbacks of Liferay are the complexity and difficulty to extend the user interface and data model to be used with our CBK concepts. Without major programming efforts it is almost impossible to implement requirements 4 to 8.

Elgg\textsuperscript{5} is an open source social network system. From a collaboration perspective, this platform seemed to be perfect. It provides all social media features one can think of, such as linking friends, group discussions, rating, commenting and so on. From a knowledge management perspective, however this platform lacks of a proper document management, and extending the data model with respect to our CBK concepts literally means programming.

DSpace\textsuperscript{6} and MyCoRe\textsuperscript{7} are both document management systems. Whereas DSpace is the more popular one especially within research institutions world-wide, MyCoRe has its roots and is quite present in the German academic library landscape. Both tools focus on handling and making available to the public documents of any kind. Their strengths lie in the integration possibilities into existing (library) systems, categorization options and meta-data handling along many known standards and elaborate searching through meta-data and document contents. Both systems mostly rely on the management of existing documents but do not support collaborative document writing. Extending the data model is only possible by programming a plugin.

At last we realized that the Semantic MediaWiki\textsuperscript{8} offers almost all functionality we were looking for with almost no need for programming. In the next chapter, we take a closer look on the Semantic MediaWiki, and how it will serve our needs as a platform for the CBK.

2.3.2 Our Choice

The Semantic MediaWiki itself is an extension of the MediaWiki platform, which is quite renowned empowering the popular Wikipedia encyclopedia. The Semantic MediaWiki has been funded in part by projects of the EU Framework Programmes (FP6 & FP7), SEKT\textsuperscript{9} and ACTIVE\textsuperscript{10}, and by the project Halo\textsuperscript{11}. The Halo core extension in turn is an extension to the Semantic MediaWiki featuring better usability options. The whole conglomerate of the MediaWiki, the Semantic MediaWiki extension, and the Halo core extension is called SMW+. In the following, when we speak about SMW+ we will refer to the whole package consisting of the MediaWiki, the Semantic MediaWiki extension and the Halo core extension.

SMW+ is an open source semantic enterprise wiki for creating and sharing knowledge. Since SMW+ is a wiki using the MediaWiki platform as its basis, the fundamental concept underlying SMW+ is the web-based collaboration and quick authoring and provision of content. This means that each web page is in principle editable by anyone. It also includes the possibility to version and to discuss the content. Quality is ensured by the masses of people reading and working with the content. The wiki principle enables them to instantly correct or report wrong information or vandalism. Smaller organizations lack of this mass effect, but can ensure quality by organizational means or access control assignable to groups or individuals. Access control functionality is not part of the MediaWiki but is a feature of SMW+. By this, we see the requirements 1 to 3 fulfilled.

The major feature of the SMW+ extension is the possibility to add semantic annotations to wiki pages. Semantics are meta-data that allow better processing, searching and re-using of knowledge. The semantic annotations are formalized by an ontology. This allows the system to give answers to questions such as “What are the 10 largest capitals located in the European Union?” without the need to look at each relevant wiki page and calculate the result manually. Thus, semantically enriched wiki pages make it possible to query the wiki like a database. Aggregated data is kept up-to-date, can be explored interactively and can be visualized. It even becomes realistic to create a good starting point for a book with an adequate and probably quite complex query. By this, we see the requirements 4 and 5 fulfilled.

\begin{itemize}
\item \textsuperscript{5}http://www.elgg.org
\item \textsuperscript{6}http://www.dspace.org
\item \textsuperscript{7}http://www.mycore.de
\item \textsuperscript{8}http://www.semantic-mediawiki.org
\item \textsuperscript{9}http://www.sekt-project.com
\item \textsuperscript{10}http://www.active-project.eu
\item \textsuperscript{11}http://www.projecthalo.com
\end{itemize}
The concept of semantic forms helps to create pages with predefined annotated data, because the content of form fields is automatically annotated in the resulting wiki page. Forms are generated with the help of wizards. By choosing a class and defining a subset of attributes from the ontology, a form is generated for a specific class with input fields according to the selected attributes. The ontology itself is editable and extensible by a comfortable ontology browser. By this, we see the requirements 6 to 9 fulfilled.

The SMW+ provides web services that can be used by external applications to access the ontology as well as the actual contents. Additionally, an export module can be used to extract the ontology as an OWL file. By this, we see the requirement 10 fulfilled.

2.3.3 Realization

In SMW+, every part of the ontology is represented by a wiki page. A wiki page exists for each class, each property, and each individual. Creating our CBK ontology in SMW+ literally means creating wiki pages for each element presented in Sect. 2.2. After creating classes and properties, class individuals have to be provided.

KO individuals (see Sect. 2.2.1), the actual contents of the CBK, such as concrete methods, tools, techniques, and notations are provided by community members. Semantic forms are supposed to support the creation of such KO individuals for a certain class. Semantic forms are initially generated along the properties of a class. This means that for each property a corresponding input element for filling in the property value is placed on this form. The generated form can be adapted afterwards to implement user interface specialties such as grouping the generated input elements into areas or tabs. We have therefore generated and adapted semantic forms for each KO subclass so that community members are able to contribute KO individuals more comfortably through these forms.

All KA individuals were created along the KAs defined in Sect. 2.2.1. On the one hand, KA individuals can be referenced during the KO individual creation; on the other hand, all KO individuals referencing a KA are queried and presented, when browsing a certain KA individual wiki page.

To create a learning trail individual (see Sect. 2.2.1), the following information must be provided for each learning trail step:

1. a description of the learning trail step
2. a reference to the next learning trail step if applicable
3. a reference to the previous learning trail step if applicable
4. references to KOs if applicable

In conjunction with SMW+, we use the open-source tool Protégé to edit our CBK ontology. Since both products are able to handle the OWL ontology format, we established a round trip workflow in which individuals are provided through the SMW+, which are then imported into Protégé. We are then able to fine tune and test the ontology and queries in Protégé. Adaptations to the ontology or queries are then imported back into the SMW+, where semantic forms and individuals have to be regenerated and customized again depending on the adaptations made.

2.4 Establishing Processes

Formulating a body of knowledge for a new discipline is not a task which is accomplished by an individual. It is a highly collaborative effort with many people involved comprising many activities, such as having discussions about what the core of the discipline is, what common terminology to agree upon, and what the state-of-the-art is constituted by, to name just a few. It should also be realized collaboratively since codifying the knowledge into words and sentences or at least referencing existing knowledge like books and papers means a lot of work. Since the work is never finished, thinking of all new research results contributing to the body of knowledge every day, collaboration is the only feasible way, to keep the CBK up-to-date. We acknowledge this by choosing a collaborative approach backed by SMW+ (see Sect. 2.3) to build up a CBK for engineering secure software and services relying explicitly on user participation.

http://protege.stanford.edu/
A CBK has the greatest benefit, if it is complete, up-to-date, and valid. Especially in the beginning of such a project this is not the case, leading to low acceptance and low user participation, if launched for the public too early. We therefore conceived three phases, each with a different focus and participation style in order to work against this effect. Furthermore, the CBK content has to be revised on a regular basis to ensure a high quality, which can be summed up by the question: How is content provisioning and quality assurance supported best while relying on user participation?

We present the three phases in Sect. 2.4.1, and we introduce quality assurance means in Sect. 2.4.2.

2.4.1 Three Phases

The first phase is a planning phase, in which all discussed aspects are considered while preparing the initial CBK structure and planning.

During an inception phase, content is provided by a closed user group, consisting of experts from different areas within the secure software development field. These experts are mainly researchers from NESSoS, where we profit from the opportunity of having so many researchers linked together through the NoE. The writing process is managed by a central coordinator, who creates the initial CBK structure, defines clear writing responsibilities, watches deadlines, and ensures quality (see Sect. 2.4.2). At the end of this phase, the result is a sound CBK content base providing a complete, up-to-date, and validated state-of-the-art of this interdisciplinary research field. A high benefit of this work is expected for researchers from service, security, and software engineering. But also practitioners will find it interesting to get a glimpse on what current research has to offer.

The run phase is marked by the launch of the CBK for the general public in terms of reading and writing. At this point in time, the CBK should provide a complete overview of the research field of secure software development. To launch the CBK with a sound content base, which has mostly been created by the community itself, increases the attraction of the CBK for other people that we considered in the use cases (see Sect. 2.1.2). Especially for practitioners and for stakeholders other than researchers, learning trails will guide through the vast amount of research results, with respect to their expert level (see Sect. 2.2.1).

2.4.2 Quality Assurance

The SMW+ supports quality assurance tasks in different ways. Authors are notified via e-mail, when other people have modified their KO. In the case of vandalism or wrong information, it is possible to revert the changes back to a previous state, making use of the versioning functionality of SMW+. If provided information is controversial, the system allows users to have discussions for each knowledge object on the same page. If new attributes are introduced, it is usually the case that these attributes lack of values for existing individuals. SMW+ provides a mechanism to gather information about missing attribute values and makes it possible to notify the respective author. Furthermore, SMW+ provides an elaborate access control mechanism, which makes it possible to define groups and assign read and write access rights. We make use of this mechanism in order to introduce roles, each with different access rights for, e.g., KOs, KAs or administrative functions of SMW+.

Depending on the project phase, quality is assured in different ways.

In the inception phase, quality is assured by a restrictive access control, allowing only partners of the network to have full access to the CBK. Additionally, a central quality assurance (QA) team will start their work having a regular qualitative review on the contents of the CBK, flagging them with a marker indicating, when a KO needs to be revised due to a low content quality. But not only the QA team is able to flag KOs. Everyone is allowed to flag an article if vandalism is detected.

While the inception phase is characterized by a controlled environment through a closed user group, the run phase takes a more decentralized and community-driven approach. Since we assume that we will reach a critical mass in users during a short period after going public, content contribution will increase and self-regulation will become realistic. Thus, quality assurance is incrementally shifted over to the user, because the QA task is no more feasible to be exercised by a few experts. Instead of this, experts will rather be assigned responsibilities along the knowledge areas, taking a more moderating role.

As already mentioned, the underlying SMW+ platform supports both approaches, providing adequate collaboration functionality like feedback and access control mechanisms.
2.5 Related Work

The concept of a codified BOK is not new and can be found in many different disciplines. Compared to our CBK they all differ in how they were created and in how knowledge is codified.

All of the BOKs presented in the following were created top-down. By this we mean that an expert team was formed or authors were chosen to write articles. Our approach comprises a top-down phase, but also a bottom-up phase in which the CBK is opened to the public in terms of reading and writing (see Sect. 2.4). This is comparable to the shift from the creation of the Encyclopedia Britannica to the creation of Wikipedia, acknowledging the fact that new knowledge is generated very fast and by many people these days.

A BOK mentioned before is the “Software Engineering Body of Knowledge” aka SWEBOK [8], the most prominent among all other BOKs within the SE discipline. The Computer Engineering Body of Knowledge (Computing Curricula 2005) [1] has a more educational program focus, which covers undergraduate degree programs in computer engineering, computer science, information systems, information technology, and SE. A special focus on SE education has the Software Engineering Education Knowledge (SEEK) (part of [18]), which covers SE knowledge for undergraduate SE education. Compared with this the SWEBOK covers knowledge a software engineer should have after four years of experience. The Project Management Body of Knowledge (PMBOK) [13] is also well-known and covers project management knowledge in general. It is not limited to software project management. This book has influenced many subsequent BOK efforts in the computing disciplines.

In the security field, BOKs do exist with different content-wise focal points promoted by both industry and governments. Examples are the (ISC)² Common Body of Knowledge [20] created by the non-profit organization “International Information Systems Security Certification Consortium, Inc.” and the Information Technology Security Essential Body of Knowledge [21]. Both serve the purpose of defining a basis for curricula and profession profiles, as well as certification programs for training IT security personnel. Concerning industry best practices for secure software engineering the non-profit organization “The Software Assurance Forum for Excellence in Code” (SAFECode) should be mentioned. Many software companies participate in SAFECode, which publishes documents on a regular basis describing principles and best practices [5]. The BOK Software Assurance - A Curriculum [17] is another U.S. Department of Homeland Security effort to collect knowledge in one place about developing secure software.

Planned but not yet realized efforts towards BOKs in the SE field are the Body of Knowledge for Software Quality Measurement [15], the Common Body of Knowledge for Information Security [19], the Requirements Engineering Body of Knowledge (REBOK) [3, 4], and the Computer Engineering Technology Body of Knowledge [9].

A more collaborative approach is taken by the two BOK projects Usability BOK13 and Build Security In14 by the U.S. Department of Homeland Security, both also fostering user participation to provide content following a bottom-up approach.

All of the BOKs presented so far cannot be queried elaborately. Many BOKs only exist as a book with access possibilities given by the table of contents or a key word index, while others also provide a hypertext system, allowing to browse content along links, such as the online version of the SWEBOK15, the IEEE Body of Knowledge on Services Computing16 or the Guide to the Systems Engineering Body of Knowledge (G2SEBoK)17. We go one step further and allow more elaborate queries (see Sect. 2.2.2). This is realized through our CBK ontology with which we are able to semantically enrich all CBK contents.

The need for defining a common terminology for different interdisciplinary communities led to a large number of publications in this area. These include the work by Firesmith [11] for safety and security and by Fabian et al. [10] for SE and security, to name only two examples. Similar to our common terminology concept, their approaches define taxonomies relating fundamental notions across the different disciplines, and they specify to what extent notions from one discipline can be translated into notions of the other discipline. The main difference to our work is that Firesmith and Fabian et al. do not complement their results by further concepts such as KAs, learning trails, etc. to create a CBK.

---

13http://www.usabilitybok.org
14https://buildsecurityin.us-cert.gov
15http://www.computer.org/portal/web/swebok
16http://www.servicescomputing.tv
17http://g2sebok.incose.org
3 Content of the Common Body of Knowledge

In this chapter, we give an overview of the current state of the CBK.\(^1\)

The overview in this chapter refers to the state of the website [http://www.nessos-project.eu/cbk/](http://www.nessos-project.eu/cbk/) of August 31st, 2011.

Table 3.1 gives an overview of all 68 KOs currently available in the CBK. They are grouped according to the four KO types.

As stated in the previous chapters, the CBK’s structure and its content are constantly evolving. In the following, we graphically present the structural as well as the contentual development over time. All three graphs start at the beginning of the project and end at the current state of month 12.

Figure 3.1 shows how the overall amount of KOs (68 in total at month 12) are divided into the four currently available KO types. Most of the KOs (28) are techniques closely followed by the number of tools (24). The other KOs are split into notations (9) and methods (7).

![Figure 3.1: Development of KOs according to KO types over time](image)

Figure 3.2 shows the mapping of KOs to different KAs. Not all KOs are already mapped to a KA. Also, authors can map a KO to more than one KA. In this figure, only the first chosen KA is considered. In total, this leads to a number of only 32 classified KOs out of 68 KOs available in the CBK at month 12. The KAs with the most KOs are Secure architecture and design (11), Software Engineering Process (9) and Risk and cost aware SDLC (8). All other KAs only have less than three KOs. To be able to draw sound conclusions, we first have to increase the amount of mappings of KOs to KAs.

Figure 3.3 gives an overview on how mature the provided KOs are. The maturity level properties have been added after month 6. Therefore, the figure does not refer to the time before month 6. Only 31 KOs out of 68 KOs available in the CBK at month 12 provide data about their maturity. Most of these KOs are considered as Proof-of-concept/Unvalidated (26). Four KOs are considered Stable for production and only one as Used in some organisations. No KO is considered Widely adopted in practice which represents the final level concerning the KO’s maturity lifecycle. Since most of the state-of-the-art considered in the first year is research, the number of KOs drops rapidly in the higher maturity levels.

In the following four Figs. 3.4, 3.7, 3.9, 3.12, screenshots from the CBK website are depicted. Each represents an example of one out of the four KO types.

---

\(^1\) A special account for reviewers has been created so that access is possible to the currently still password-protected content of the CBK: User: UDE2; Password: 47FunU

NESSoS - 256980
| Method | CORAS Method  
Security Engineering Process Using Patterns  
SecMER - Requirement Evolution Management  
XESB |
|-------------------------|
| Notation | UMLsec  
SecureUML  
SECTET  
A uml-based pattern specification technique  
An Expressive Aspect Composition Language for UML State Diagrams  
Misuse Case  
UWE |
| Tool | MASTER Design Workbench  
X-CREATE  
Jalapa  
WS-TAXI  
EOS  
SSG: Smart and Secure GUI Builder  
Avantssar-atse  
Avantssar Orchestrator  
VeriFast  
MagicUWE  
Service Development Environment  
CORAS Tool  
CORAS Risk Monitor  
UML4PF |
| Technique | An approach for model composition and verification  
An aspect-oriented methodology for designing secure applications  
A guidance for model composition  
Software Architecture Evolution  
Model-based development of dynamically adaptive software  
Structured service composition execution for mobile web applications  
A security adaptation reference monitor (sarm) for highly dynamic wireless environments  
Modeling aspect-oriented web service compositions at shared join points  
Modeling and Verification of Web Services Composition based on CPN  
Verification and Trade-Off Analysis of Security Properties in UML System Models  
Providing support for model composition in metamodels  
Aspect Oriented Modeling of Component Architectures Using AADL  
An Aspect-Oriented and Model-Driven Approach for Managing Dynamic Variability  
Security-driven Model-based Dynamic Adaptation  
Model-Based Software Design and Adaptation  
Introducing variability into aspect-oriented modeling approaches  
Verification of Access Control Requirements in Web Services Choreography  
Trust Evolution Policies for Security in Collaborative Ad Hoc Applications  
Semi-automated adaptation of service interactions  
Adaptation of Service Protocols using Process Algebra and On-the-Fly Reduction Techniques (cont’d)  
A Formal Approach to Component Adaptation  
Context-Aware Composition and Adaptation Based on Model Transformation  
A Formal Approach to Component Adaptation  
SYNTHESIS A Tool for Automatically Assembling Correct and Distributed Component-Based Systems (cont’d)  
Automated Generation of BPEL Adapters  
Automatic Generation of Security Controller  
Event Tree Analysis  
Fault Tree Analysis |

**Table 3.1: Overview of all KOs ordered by KO types**
Figure 3.2: Development of KAs over time

Figure 3.3: Development of KO’s maturity over time
UWE is a software engineering approach for the Web domain aiming to cover the whole life-cycle of Web application development. It makes the integration of security aspects in web engineering possible.

**Contents**

<table>
<thead>
<tr>
<th>[hide]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Context</td>
</tr>
<tr>
<td>2 Problem (and motivation)</td>
</tr>
<tr>
<td>3 Solution</td>
</tr>
<tr>
<td>4 Consequences</td>
</tr>
<tr>
<td>5 Knowledge Area</td>
</tr>
<tr>
<td>6 Relation to Common Terminology</td>
</tr>
<tr>
<td>7 Image Gallery</td>
</tr>
<tr>
<td>8 Usage Example(s)</td>
</tr>
<tr>
<td>8.1 HospInfo</td>
</tr>
<tr>
<td>8.2 Secure Address Book</td>
</tr>
<tr>
<td>9 Publications</td>
</tr>
<tr>
<td>10 Relations</td>
</tr>
<tr>
<td>11 External Links</td>
</tr>
</tbody>
</table>

**Context**

Secure web applications are becoming increasingly important due to rising cybersecurity as well as the growing awareness of data privacy. Since adding security features to already existing applications can be a very time-consuming task, it is important to take security aspects into account while planning and modeling a web application.

**Problem (and motivation)**

The security aspects of UWE tackle the problem of insecure web applications. The earlier approaches for security engineering were too formal and complex for the web and no pragmatic approaches existed so far.

**Solution**

With the security extension of UWE, web developers are enabled to model security aspects as authentication, access control and secure connections efficiently. For this purpose, the UML-based Web Engineering (UWE) approach is enhanced. Despite its seamless integration into UWE, the security aspects of UWE can also be utilized independently for other UML-based modeling techniques.

**Consequences**

For web engineers it is now easy to consider security features at an early stage, without the necessity to switch between two separate modeling techniques.

**Knowledge Area**

| Belongs to Knowledge Area | Secure architecture and design |

**Relation to Common Terminology**

Currently no relations defined. Feature planned for 01/2012.

Figure 3.4: Example method: UWE (part 1 of 3)
**Image Gallery**

Overview of UWE models and security aspects

**Usage Example(s)**

**HospInfo**

Authentication, authorization and secure connections are basic and important requirements for Hospital Information Systems (HISs). The sooner security aspects are incorporated into the development process, the sooner errors and inappropriate concepts can be revealed that otherwise would have required costly fault analysis and patching cycles. Our case study, called HospInfo (Hospital Information), is a web-based HIS.

https://www.fmi.mw.tum.de/exampleHospInfo.html

**Secure Address Book**

A web application of a secure address book should allow registered users to create several address books and to add new contacts.

Non-registered visitors can only read an introduction and the terms of service until they register or authenticate themselves. Administrators cannot use the address book functionality, but they are allowed to search for users and to delete their accounts including all address books and contacts. For registered visitors the address books are shown in a column on the left of the page. On the right the contact details of the currently selected address book are displayed. Every address book can be deleted and besides it is possible to create additional ones. The contacts can be created/removed and the user may read or update the contact details, e.g. the name, picture, postal addresses, email address or phone numbers. The latter three elements are tagged, i.e. the user can specify an arbitrary named tag for each address to distinguish between them, for example between home address and business address.

https://www.fmi.mw.tum.de/exampleSecureAddressBook.html

---

**Figure 3.5: Example method: UWE (part 2 of 3)**
Figure 3.6: Example method: UWE (part 3 of 3)
SecureUML

SecureUML provides a language for specifying access control policies for actions on protected resources. It leaves open what the protected resources are and which actions they offer to clients. These are specified in a so-called dialect and depend on the primitives for constructing models in the system design modeling language.

Contents

[Hide]

- 1 Context
- 2 Problem (and motivation)
- 3 Solution
- 4 Consequences
- 5 Knowledge Area
- 6 Image Gallery
- 7 Usage Example(s)
- 8 Publications
- 9 Relations
- 10 External Links

Context

Model driven development holds the promise of reducing system development time and improving the quality of the resulting products. Recent investigations have shown that security policies can be integrated into system design models and that the resulting security-design models can be used as a basis for generating systems along with their security infrastructures.

Problem (and motivation)

Engineering security into system design is usually neglected so security requirements are typically integrated in the system at implementation phases in an orthogonal manner. This “ad hoc” integration has a negative impact on security leading to security flaws in the system.

Solution

We propose the development of modeling languages, methods and tool support to model secure designs and automatically transform models into secure systems.

In particular, SecureUML is a modeling language for formalizing access control requirements that is based on RBAC. In RBAC, permissions specify which roles are allowed to perform given operations. These roles typically represent job functions within an organization. Users are granted permissions by being assigned to the appropriate roles, based on their competencies and responsibilities in the organization. RBAC additionally allows one to organize the roles in a hierarchy, where roles can inherit permissions along the hierarchy. In this way, the security policy can be described in terms of the hierarchical structure of an organization. However, it is not possible to specify policies that depend on dynamic properties of the system state, for example, to allow an operation only during weekdays. SecureUML extends RBAC with authorization constraints to overcome this limitation. It formalizes access control decisions that depend on two kinds of information: (i) Declarative access control decisions that depend on static information, namely the assignments of users and permissions to roles, which we designate as an RBAC configuration. (ii) Programmatic access control decisions that depend on dynamic information, namely the satisfaction of authorization constraints in the current system state.

Consequences

Model driven security development should reduce complexity of secure applications development and improve the quality of the resulting applications by enabling the analysis of system designs to detect and remove errors at earlier stages of the development process.

Figure 3.7: Example notation: SecureUML (part 1 of 2)
### MagicUWE

The CASE tool MagicUWE has been created to support the development of web applications. It focuses on the modeling phase and uses the UML-based Web Engineering (UWE) methodology. UWE provides among others a UML extension (a so called UML profile) based on stereotypes, tagged values and OCL constraints. The tool is built as a plugin for MagicDraw v.16.2. The aim is to augment usability providing additional support in the use of the web specific elements in the design, automating certain steps and providing shortcuts.

### Context

MagicUWE is implemented as a MagicDraw plugin. It was created for Web engineers who want to model secure web applications using the UML-based Web Engineering (UWE) profile and the MagicDraw CASE tool.

### Problem (and motivation)

Whenever UWE models are created, some tasks have to be repeated over and over, such for example how the navigation menu structure is built. Furthermore, some consistency checks and transformations are very time consuming if executed manually.

### Solution

The plugin MagicUWE provides features like inserting UWE's stereotyped elements and copying stereotypes and their tags. Furthermore, MagicUWE supports RIA patterns, transformations between UWE models and a consistency check for secure connection redefinitions in substrates or substrate machines.

### Consequences

MagicUWE facilitates the modeling of web applications. In particular, it provides specific elements for the modeling of security aspects, such as role-based access. The advantage of UWE is to stick to the standard UML, which allows using UML CASE tools such as MagicDraw.

---

**Figure 3.9: Example tool: MagicUWE (part 1 of 3)**
Knowledge Area

Belongs to Knowledge Area Secure architecture and design

Belongs to Knowledge Area Software Quality

Belongs to Knowledge Area Security Requirements

Suggestion of new Knowledge Area(s) Secure Web architecture and design

Image Gallery

Figure 3.10: Example tool: MagicUWE (part 2 of 3)
**Technical Details**

<table>
<thead>
<tr>
<th>Technical Requirements</th>
<th>MagicDraw v. 16.8 (at least Parema) has to be installed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Common Public License Version 1.0 (see <a href="http://www.opensource.org/licenses/gpl-0.php">http://www.opensource.org/licenses/gpl-0.php</a>)</td>
</tr>
<tr>
<td>Eclipse Update Site</td>
<td>-</td>
</tr>
</tbody>
</table>

**Installation Guide**

Installer version: java -jar MagicUWE16.802 installer jar (or double-click on the file in many operating systems)

Zip-version for MagicDraw Resource Manager: If you prefer to use the MagicDraw Resource Manager, you can import the zipfile.

(Please delete the folder MagicDraw/plugins/MagicUWE before re-installing MagicUWE)

SDE based tool installation: To use this tool within the SDE, not only MagicDraw has to be installed, but also MagicUWE (see instructions above). The SDE just launches MagicDraw

**Tool input type(s)** (e.g. Java
function, JSON, SOAP XML)

- MagicDraw 16.8 project

**Tool output type(s)** (e.g. Java
function, JSON, SOAP XML)

- MagicDraw 16.8 project

**Function signature**

void editUMLFile(String file) throws MagicDrawException

**Description of parameter(s)**

The file has to be an MagicDraw file, e.g. a *.mbo* file.

**Description of return value(s)**

- Opens the UML file at the specified location in MagicDraw.

**Description of function**

If a file within the SDE has to be included in an eclipse project.

Options: MAGICDRAW_EXECUTABLE_LOCATION Absolute path to MagicDraw executable in local file system

**Function signature**

void openMagicDraw() throws MagicDrawException

**Description of parameter(s)**

-

**Description of return value(s)**

-

**Description of function**

Opens MagicDraw.

Options: MAGICDRAW_EXECUTABLE_LOCATION Absolute path to MagicDraw executable in local file system

**Usage Example(s)**

**Publications**

Host: Currently BibTeX is displayed as source only.

**Relations**

<table>
<thead>
<tr>
<th>Returns to tool/method</th>
<th>UWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation Type</td>
<td>SupportsBy</td>
</tr>
</tbody>
</table>

**Known uses**

Model examples can be found here:

[http://www.post.itimu.de/examples.html](http://www.post.itimu.de/examples.html)

**External Links**

- [1](#)

This page was last modified on 20 August 2011, at 12:10. This page has been accessed 55 times.

Figure 3.11: Example tool: MagicUWE (part 3 of 3)
Bayesian belief network

A Bayesian network is a directed, acyclic graph where a relation between two nodes means that the source node represents a cause or contributing factor to the target node. A Bayesian network can be utilized both quantitatively and qualitatively. If the Bayesian network is analysed qualitatively, it provides relations between causes and effects. When a Bayesian network is analysed quantitatively, each node holds a table with a probability distribution reflecting its parent nodes. For any manipulation of the probabilities of the nodes, the effects both forwards (towards child nodes and the top node) and backwards (towards parent nodes) can be computed using Bayesian probability reasoning.
4 Conclusions

We give a brief summary of this deliverable in Sect. 4.1, and we present a number of plans on how to further develop and exploit the CBK in Sect. 4.2.

4.1 Summary

We presented a conceptual approach to support bringing together disciplines based on a CBK. Our approach comprises the following main contributions to consolidate interdisciplinary communities:

- KOs allow users to *structure knowledge* such as best practices and research results according to their type. Provided content is semantically enriched in an automated manner. This allows users to browse, compare, and run complex queries on the CBK.
- The CBK introduces a mechanism to *group knowledge* into KAs. This provides access to the CBK via a hierarchical taxonomy and represents a valuable instrument to discover gaps in practice and research.
- The common terminology helps the community to find a common language of the different disciplines, and to define and use translations.
- *Learning trails* provide access to the CBK for a broader audience, practitioners and researchers in particular.
- The ontology which underlies the whole CBK enables several representations of the CBK, including known ones like books and hypertexts. It gives the flexibility to create customized representations.
- User participation is supported by adequate processes and by the chosen SMW+ framework, which might lead to a more up-to-date, a more comprehensive, and a sustained CBK.
- The realization of the CBK through SMW+ provides a smart means to allow collaborative creation and editing since SMW+ is fully integrated with the design of the ontology using Protégé.

4.2 Further Development and Exploitation of the CBK

All of the endeavors presented in this deliverable and also our plans for future work have one major goal: The research and practice communities to be integrated by the NESSoS project eventually use the CBK. The CBK should become kind of a lighthouse, which researchers and practitioners see and use as the main reference for knowledge in the field of engineering secure software and services. Therefore, we plan to focus the future developments of the CBK concerning its implementation, processes and QA, administration, and dissemination on achieving this goal.

Technically, we plan to fine tune our CBK ontology based on the collaborative work with LMU on deliverable D2.1 [7] – the first results of the method and tool evaluation. There, we recognized that some ontology properties might be more specific, and that further properties might ease the comparison of different tools and methods. Moreover, we plan to elaborate more on using the ontology properties for specific use cases such as identifying gaps in practice and research areas and comparing different tools or methods. In particular, we want to implement a “faceted” search interface for the CBK, which allows the user to graphically construct complex CBK queries based on the ontology structure and properties. A combination of several ontology properties serve as criteria or “facets” for a search activity, e.g., searching KOs of type “Method” (first facet) that belong to the KA “Software Design” (second facet) and that support the notation “UML” (second facet, via the ontology property HasMethodNotationFor. So far, the common terminology and the learning trails as parts of the CBK’s conceptual framework exist as design sketches only. Therefore, we plan to technically realize first versions of these two basic CBK concepts within the next year.

From the processes and QA point of view, we plan to establish additional means to evaluate the CBK content. For example, we want to include user rating mechanisms in our CBK concept.

From the administration perspective, we believe that it is instrumental to be able to measure who added or retrieved what content from the CBK, an who executed which queries on the CBK. This will help us to evaluate whether, how, and by whom the CBK is actually used. The measurement results will enable us to react rapidly, e.g., if they show that the CBK is used by a limited community only. We plan to support
the intended measurements by make the CBK available to public (not only the NESSoS partners) as early as possible.

Ultimately, we plan to focus our dissemination activities, e.g., presence on relevant conferences and industry events, publications, and CBK demos, on increasing the popularity of the CBK as a helpful tool and handbook for researchers and practitioners in the field of engineering secure software and services.
Bibliography


[12] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides. Design Patterns - Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.


A Ontology Properties

This appendix gives a complete overview of all properties currently realized in the NESSoS CBK.

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasKOExecutiveSummary</td>
<td>String</td>
<td>1</td>
<td>short summary of the KO</td>
</tr>
<tr>
<td>HasKOAlias</td>
<td>String</td>
<td>0..*</td>
<td>alternative name</td>
</tr>
<tr>
<td>HasKOContextDescription</td>
<td>String</td>
<td>1</td>
<td>according to design pattern format [12]</td>
</tr>
<tr>
<td>HasKOProblemDescription</td>
<td>String</td>
<td>1</td>
<td>do.</td>
</tr>
<tr>
<td>HasKOSolutionDescription</td>
<td>String</td>
<td>1</td>
<td>do.</td>
</tr>
<tr>
<td>HasKOConsequencesDescription</td>
<td>String</td>
<td>1</td>
<td>do.</td>
</tr>
<tr>
<td>HasKOUsageExample</td>
<td>UsageExample</td>
<td>0..*</td>
<td>to present case studies</td>
</tr>
<tr>
<td>HasKOPublication</td>
<td>Publication</td>
<td>0..*</td>
<td>for referencing scientific and technical publications</td>
</tr>
<tr>
<td>HasKOAddress</td>
<td>String</td>
<td>1</td>
<td>postal address of contact</td>
</tr>
<tr>
<td>HasKOContactEmailAddress</td>
<td>string</td>
<td>1</td>
<td>email address of contact</td>
</tr>
<tr>
<td>HasKOWebsite</td>
<td>String</td>
<td>0..1</td>
<td>website of KO</td>
</tr>
<tr>
<td>HasKOTags</td>
<td>String</td>
<td>0..*</td>
<td>keywords (comma-separated)</td>
</tr>
<tr>
<td>HasKOKA</td>
<td>KnowledgeArea</td>
<td>1..*</td>
<td>to categorize a KO into a specific KA</td>
</tr>
<tr>
<td>HasKOKASuggestions</td>
<td>String</td>
<td>0..*</td>
<td>to suggest a new KA if KO does not fit into existing KAs</td>
</tr>
<tr>
<td>HasKOSynonymTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
<td>see Sect. 2.2.1</td>
</tr>
<tr>
<td>HasKOBroaderTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
<td>do.</td>
</tr>
<tr>
<td>HasKONarrowerTerm</td>
<td>CommonTerm</td>
<td>0..*</td>
<td>do.</td>
</tr>
<tr>
<td>HasKOImage</td>
<td>Image</td>
<td>0..*</td>
<td>for screenshots, illustrations, etc.</td>
</tr>
<tr>
<td>HasKOKnownUses</td>
<td>String</td>
<td>0..*</td>
<td>to present known uses of the KO</td>
</tr>
<tr>
<td>HasKOAdditionalInfo</td>
<td>String</td>
<td>0..*</td>
<td>additional information (e.g., formal aspects, standards, patents, etc.)</td>
</tr>
<tr>
<td>HasKORelevanceForStandards</td>
<td>String</td>
<td>0..*</td>
<td>information about relevance for national or international standards</td>
</tr>
<tr>
<td>HasKORelevanceForAuditingAndCertification</td>
<td>String</td>
<td>0..*</td>
<td>information about relevance for auditing and certification</td>
</tr>
<tr>
<td>HasKONotation</td>
<td>Notation</td>
<td>0..*</td>
<td>specify notations a KO is based on</td>
</tr>
<tr>
<td>HasKOToolSupport</td>
<td>Tool</td>
<td>0..*</td>
<td>specify tools which support this KO</td>
</tr>
<tr>
<td>HasKOMaturityLevel</td>
<td>MaturityLevel</td>
<td>1</td>
<td>choose one out of four levels</td>
</tr>
</tbody>
</table>

Table A.1: KnowledgeObject Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasMethodSupportFor</td>
<td>Tool</td>
<td>0..*</td>
<td>to specify tools a method supports</td>
</tr>
<tr>
<td>HasMethodSubMethod</td>
<td>Method</td>
<td>0..*</td>
<td>to specify sub-methods</td>
</tr>
<tr>
<td>HasMethodPredecessor</td>
<td>Method</td>
<td>0..*</td>
<td>to specify method predecessors</td>
</tr>
<tr>
<td>HasMethodActivity</td>
<td>Activity</td>
<td>1..*</td>
<td>activity(ies) the method consist(s) of</td>
</tr>
<tr>
<td>HasMethodRole</td>
<td>Role</td>
<td>0..1</td>
<td>to assign roles</td>
</tr>
<tr>
<td>HasMethodRelation-ToSDLCPhase</td>
<td>SDLCPhase</td>
<td>0..*</td>
<td>relation to phase from the SDLC</td>
</tr>
</tbody>
</table>

Table A.2: Method Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasActivityName</td>
<td>String</td>
<td>1</td>
<td>activity name</td>
</tr>
<tr>
<td>HasActivityDescription</td>
<td>String</td>
<td>1</td>
<td>activity description</td>
</tr>
<tr>
<td>HasActivityInput</td>
<td>String</td>
<td>0..*</td>
<td>activity input artifacts</td>
</tr>
<tr>
<td>HasActivityOutput</td>
<td>String</td>
<td>0..*</td>
<td>activity output artifacts</td>
</tr>
</tbody>
</table>

Table A.3: Activity Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasToolUses</td>
<td>Tool</td>
<td>0..*</td>
<td>to specify tools the tool is based on</td>
</tr>
<tr>
<td>HasToolTechnicalRequirements</td>
<td>String</td>
<td>0..1</td>
<td>to specify software and hardware restrictions</td>
</tr>
<tr>
<td>HasToolLicense</td>
<td>String</td>
<td>0..1</td>
<td>to specify licensing issues</td>
</tr>
<tr>
<td>HasToolEclipseUpdateSite</td>
<td>String</td>
<td>0..1</td>
<td>NESSoS specific</td>
</tr>
<tr>
<td>HasToolEclipseSDEUpdateSite</td>
<td>String</td>
<td>0..1</td>
<td>NESSoS specific</td>
</tr>
<tr>
<td>HasToolInput</td>
<td>String</td>
<td>0..*</td>
<td>NESSoS specific tool input types like XML, JSON, etc.</td>
</tr>
<tr>
<td>HasToolOutput</td>
<td>String</td>
<td>0..*</td>
<td>NESSoS specific tool output types like XML, JSON, etc.</td>
</tr>
<tr>
<td>HasToolInstallationGuide</td>
<td>String</td>
<td>0..*</td>
<td>to refer to installation guides and howtos</td>
</tr>
<tr>
<td>HasFunction</td>
<td>Function</td>
<td>0..*</td>
<td>NESSoS specific</td>
</tr>
</tbody>
</table>

Table A.4: Tool Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasSignature</td>
<td>String</td>
<td>1</td>
<td>to specify the signature of the function</td>
</tr>
<tr>
<td>HasParameters</td>
<td>String</td>
<td>0..1</td>
<td>to specify the parameters of the function</td>
</tr>
<tr>
<td>HasReturnType</td>
<td>String</td>
<td>1</td>
<td>to specify the return type of the function</td>
</tr>
<tr>
<td>HasDescription</td>
<td>String</td>
<td>1</td>
<td>to describe the function</td>
</tr>
</tbody>
</table>

Table A.5: Notation Property

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasUsageExampleTitle</td>
<td>String</td>
<td>1</td>
<td>usage example name</td>
</tr>
<tr>
<td>HasUsageExampleDescription</td>
<td>String</td>
<td>1</td>
<td>usage example description</td>
</tr>
<tr>
<td>HasUsageExampleCode</td>
<td>String</td>
<td>0..*</td>
<td>usage example sample code</td>
</tr>
<tr>
<td>HasUsageExampleImage</td>
<td>Image</td>
<td>0..*</td>
<td>usage example images</td>
</tr>
</tbody>
</table>

Table A.6: Function Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasSignature</td>
<td>String</td>
<td>1</td>
<td>to specify the signature of the function</td>
</tr>
<tr>
<td>HasParameters</td>
<td>String</td>
<td>0..1</td>
<td>to specify the parameters of the function</td>
</tr>
<tr>
<td>HasReturnType</td>
<td>String</td>
<td>1</td>
<td>to specify the return type of the function</td>
</tr>
<tr>
<td>HasDescription</td>
<td>String</td>
<td>1</td>
<td>to describe the function</td>
</tr>
</tbody>
</table>

Table A.7: UsageExample Properties
### Table A.8: LearningTrail Property

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasLearningTrailStep</td>
<td>LearningTrailStep</td>
<td>1..*</td>
<td>reference to learning trail step(s)</td>
</tr>
</tbody>
</table>

### Table A.9: LearningTrailStep Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasLearningTrailStepPredecessor</td>
<td>LearningTrailStep</td>
<td>0..1</td>
<td>reference to predecessor step</td>
</tr>
<tr>
<td>HasLearningTrailStepSuccessor</td>
<td>LearningTrailStep</td>
<td>0..1</td>
<td>reference to successor step</td>
</tr>
<tr>
<td>HasLearningTrailStepDescription</td>
<td>String</td>
<td>1</td>
<td>description of this step</td>
</tr>
<tr>
<td>HasLearningTrailStepReference-ToKo</td>
<td>KnowledgeObject</td>
<td>0..*</td>
<td>KOs that are described in this step</td>
</tr>
<tr>
<td>HasLearningTrailStepKA</td>
<td>KnowledgeArea</td>
<td>1</td>
<td>to categorize a trail according to a specific field</td>
</tr>
<tr>
<td>HasLearningTrailStepTargetGroup</td>
<td>TargetGroup</td>
<td>1</td>
<td>specifies for which target group this step has been created</td>
</tr>
</tbody>
</table>

### Table A.10: Link Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasLinkURL</td>
<td>String</td>
<td>1</td>
<td>URL</td>
</tr>
<tr>
<td>HasLinkURLShortDescription</td>
<td>String</td>
<td>1</td>
<td>URL description</td>
</tr>
</tbody>
</table>

### Table A.11: Image Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasImageCaption</td>
<td>String</td>
<td>1</td>
<td>image caption</td>
</tr>
<tr>
<td>HasImageURL</td>
<td>String</td>
<td>1</td>
<td>image URL</td>
</tr>
</tbody>
</table>

### Table A.12: Publication Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>M.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HasPublicationReference</td>
<td>String</td>
<td>1</td>
<td>BibTeX-String or link to DBLP</td>
</tr>
<tr>
<td>IsPublicationReference-Prime</td>
<td>Boolean</td>
<td>1</td>
<td>flag to identify primary publications</td>
</tr>
</tbody>
</table>

NESSoS - 256980 43
B Published Paper [16]
TOWARDS A COMMON BODY OF KNOWLEDGE 
FOR ENGINEERING SECURE SOFTWARE AND SERVICES∗

Widura Schwittek, Holger Schmidt, Stefan Eicker and Maritta Heisel

paluno - The Ruhr Institute for Software Technology – University of Duisburg-Essen, Germany
{Widura.Schwittek, Holger.Schmidt, Stefan.Eicker, Maritta.Heisel}@paluno.uni-due.de

Keywords: common body of knowledge, knowledge management, software engineering, security engineering, services computing, interdisciplinary.

Abstract: Interdisciplinary communities involve people and knowledge from different disciplines in addressing a common challenge. Differing perspectives, processes, methods, tools, vocabularies, and standards are problems that arise in this context. We present an approach to support bringing together disciplines based on a common body of knowledge (CBK), in which knowledge from different disciplines is collected, integrated, and structured. The novelty of our approach is twofold: first, it introduces a CBK ontology, which allows one to semantically enrich contents in order to be able to query the CBK in a more elaborate way afterwards. Second, it heavily relies on user participation in building up a CBK, making use of the Semantic MediaWiki as a platform to support collaborative writing. The CBK ontology is backed by a conceptual framework, consisting of concepts to structure the knowledge, to provide access options to it, and to build up a common terminology. To ensure a high quality of the provided contents and to sustain the community’s commitment, we further present organizational means as part of our approach. We demonstrate our work using the example of a Network of Excellence EU project, which aims at bringing together researchers and practitioners from services computing, security and software engineering.

1 INTRODUCTION

Software engineering (SE) can be considered as an “umbrella discipline”: typical SE tasks involve interdisciplinary knowledge about processes, methods, tools, and standards. Consequently, new types of SE sub-disciplines have emerged bringing together SE and other disciplines. For example, the field of security engineering, which “is about building systems to remain dependable in the face of malice, error, or miscalculation” (Anderson, 2001), has been combined with SE, and is referred to as secure software engineering.

Bringing together different disciplines harbors a number of problems, such as bringing together differing perspectives, vocabularies, and approaches. Moreover, these problems have to be considered with respect to multiple dimensions such as research and practice, which further complicates the situation.

We present in this paper an approach to overcome the aforementioned problems based on a common body of knowledge (CBK). While existing bodies of knowledge (BOKs) like the Software Engineering Body of Knowledge (SWEBOK) (Div. Auth., 2005) solely rely on books or hypertext systems as a medium, our CBK provides several advantages such as improved flexibility and access possibilities for its users. In fact, the CBK introduces an ontology that allows users to semantically enrich content. The CBK ontology is backed by a conceptual framework consisting of three main pillars: The structuring of knowledge from different disciplines that the CBK collects and integrates, such as specific tools, methods, and notations constitutes the first pillar. To consolidate an interdisciplinary community, we need a common understanding of the key concepts as well as a common vocabulary of the different disciplines. The CBK introduces a common terminology, i.e., necessary basic notions and relations between them. The common terminology is the second pillar, and it allows us to create a mapping between discipline-specific terminology and the notions of the common terminology. The last pillar comprises means to group knowledge in order to provide a variety of access options to the knowledge for a wide range of different
target groups. Since the CBK’s content is semantically enriched, it can be precisely queried in order to, e.g., find appropriate methods to analyze and solve a given problem. Moreover, the CBK can be used to contribute to identifying possible research gaps, weaknesses, and interesting directions for future research.

Another difference to existing BOKs is that our CBK heavily relies on user participation supported by the wiki platform Semantic MediaWiki\(^1\) with additional extensions (SMW+). Consequently, the CBK supports collaborative writing and provides mechanisms to build up and update the CBK. Since the CBK will be opened for the public, our approach is complemented by organizational means considering aspects such as quality assurance to ensure a high quality of content.

We demonstrate our work using the example of the EU project Network of Excellence (NoE) on Engineering Secure Future Internet Software Services and Systems (NESSoS)\(^2\), which aims at bringing together researchers and practitioners from security engineering, service computing, and SE. One of the main goals of the NESSoS project is to create a long-lasting research community on engineering secure software services and systems. Our approach for creating a CBK presented in this paper is our contribution to this goal.

The paper is organized as follows: we briefly present our case study, the NESSoS EU project, and outline use cases for the CBK in Sect. 2. In Sect. 3, we present the concepts and an ontology underlying our CBK. Section 4 introduces organizational measures of the CBK. We present related work in Sect. 5. Finally, we conclude and raise ideas for future work in Sect. 6.

2 SCOPE AND FUNCTIONALITY

The major goal of NESSoS is to lay the foundation for a long-lasting research community on engineering secure software-based Future Internet services and systems within a funding period of 42 months. Thus, partners from different fields coming from both academia and industry are re-addressing, harmonizing and integrating research activities. This interdisciplinary and international research setting has a high demand in transferring knowledge from research into practice and triggering research from practical challenges. An impact on training and education activities in Europe is expected as well. Within this overall effort of building a long-lasting research community the CBK plays an integral part. It supports the community to integrate and structure overlapping knowledge areas (e.g. SE, security engineering, services computing). Having in mind that the CBK should serve as a flexible computer-based handbook, we identified several roles and use cases in order to sketch the functionality and the scope of the CBK. We identified the researcher, practitioner, administrator and quality agent as typical roles. Each role has different aims when using the CBK, which have to be considered when defining the use cases. We finally came up with two groups of use cases. In the first group, use cases are defined concerning the management of contents such as adding and editing. In the second group, use cases define different views on the same contents for different target groups and purposes. Two examples for the second group of use cases are “Overview of a specific knowledge area” and “Comparison of different knowledge entities of the same type”. Based on the uses cases sketched in this section, we have identified four key concepts for a CBK, which we present in the next section.

3 KNOWLEDGE BASE

STRUCTURE

The basic idea behind the structural concept of the CBK is to be able to link arbitrary content classes with each other and to allow users to browse content along the links. Furthermore, the aim is to provide several access possibilities to the CBK, each customized to the target audiences and use cases the CBK addresses. In the following, we introduce a conceptual framework that consists of four basic concepts: knowledge objects, knowledge areas, learning trails, and the common terminology. All these concepts can be considered as the building blocks of the CBK.

We formalize these CBK concepts using a special CBK ontology. Ontologies are used to capture knowledge about some domain of interest. We use the OWL (Web Ontology Language)\(^3\) terminology in the following. An ontology describes concepts and relations between them. In OWL, a concept is specified in terms of a class, i.e., a set of individuals. An individual represents a concrete object in the domain in which we are interested. In general, in OWL a relation is specified as a property, which represents a binary relation between individuals.

We partially present the current ontology under-

\(^1\)http://www.semantic-mediaWiki.org
\(^2\)http://www.annessos-project.eu/
\(^3\)http://www.w3.org/TR/owl2-overview/
lying the NESSoS CBK in Fig. 1. We describe the CBK concepts as well as their representation as part of the CBK ontology in more detail in the following subsections.

3.1 Knowledge Objects

A knowledge object (KO) is a fundamental entity of the CBK. The content of a KO is structured around the problem and solution description (see exemplary properties of KO class in Fig. 1) based on the pattern approach prevalent in SE, such as design and architectural patterns). Each KO can be linked to other KOs, resulting in a network of KOs, which as a whole can be considered as a representation of a body of knowledge of a certain discipline. For the initial version of the CBK, we derived four KO types, which we consider as typical types of contributions to a body of knowledge of an engineering discipline, engineering secure software and services in particular. These KO types are methods, tools, patterns, and notations (see Fig. 1). We consider them as a starting point, open for extensions in the future. Methods define a set of activities, which in combination with a notation or a number of notations are used to tackle problems in engineering secure software and services in a systematic way. Tools support a software engineer in achieving a development goal in an (at least partially) automated way. Patterns provide a form through which knowledge about recurring development tasks is codified. A notation defines symbols, a syntax, and semantics to express relevant artifacts.

3.2 Knowledge Areas

We adopt the concept of knowledge areas (KA) from the SWEBOK (Div. Auth., 2005) for our CBK. KAs span the research field as a whole, dividing it into smaller parts and providing an easier access to subjects of interest. The SWEBOK was created in a long process from 1998 to 2003, involving approximately 500 reviewers from 42 countries in a first phase and over 120 reviewers from 21 countries in a second phase. One main result is the worldwide accepted common understanding of what is today viewed as SE. This includes the differentiation of the field into a number of KAs on which we want to base our KAs, e.g., software requirements and software design (see exemplary properties of KA class in Fig. 1). We took this decision because we regard the field of engineering secure software and services as a supplement of SE and therefore concerning all SE KAs. In addition, we introduce KAs specific to the fields of security and services based on standard literature. For example, we introduce the KAs risk analysis and privacy as presented in Anderson’s Security Engineering book (Anderson, 2001).

Each KA consists of a description providing an overview of the KA and its scope, as well as relationships to other KAs. KAs are detailed further into sub-areas, topics and sub-topics. Each topic or sub-topic contains the following three items: A short state-of-the-art description of the topic/sub-topic, links to KOs supporting the topic/sub-topic and a list of the most relevant publications for further reading.
3.3 Learning Trails

Learning trails are a structuring element meant to provide access to the common body of knowledge on engineering secure software and services for different target groups. This idea is based on the fact that content has to be prepared in accordance with the background of the reader. An expert in a topic area expects more detailed information, whereas a non-expert needs more contextual information in order to be able to understand. Learning trails are therefore written and categorized along different expert levels indicating, e.g., what prior knowledge is required to understand the content. Another differentiation is made concerning the reader’s background: if (s)he is from research or from practice. Learning trails are realized by moderated tours, which guide the reader through a set of KOs, which are considered to be part of a certain topic. Each step builds upon the previous step and gives a successive introduction into a topic with respect to the reader’s expert level and background (see classes LearningTrail and LearningTrail-Step in Fig. 1). The overall aim of this approach is to provide access to the CBK for a broad spectrum of people, regardless of whether the reader is a student, an experienced expert, a practitioner, or a researcher.

3.4 Common Terminology

The aim of a common terminology is to enable a community to speak the same language; or at least to simplify the translation of a term to another domain with the help of a common reference or common term as we want to call it in the following (see Fig. 1). A common term is a term with a meaning on which an agreement was reached within the community. With the common terminology, we therefore introduce an instrument for defining a common term with a certain meaning and for relating different terms with the same or a similar meaning to this common term. In the opposite direction, the common terminology serves the purpose of a dictionary from which synonyms and translations can be queried. A term does not always have the same exact meaning of another similar term, so that deviations to the meaning of the common term must be made explicit. In the CBK, this is realized by three different relationship types. A term’s meaning is either synonymous, broader or narrower in relation to another term’s meaning (see relationships between CommonTerm and KOTerm in Fig. 1).

The core CBK team initially creates an ontology of terms of the domain “engineering secure software and services” on basis of the existing CBK content and term usage after a certain period of time. It is then proposed to the community and refined within regular feedback cycles.

4 ORGANIZATIONAL MEASURES

Formulating a body of knowledge for a new discipline is not a task which is accomplished by an individual. It is a highly collaborative effort with many people involved comprising many activities, such as having discussions about what the core of the discipline is, what common terminology to agree upon, and what the state-of-the-art is constituted by, to name just a few. It should also be realized collaboratively, because codifying the knowledge into words and sentences or at least referencing existing knowledge like books and papers means a lot of work. Since the work is never finished, regarding of all new research results contributing to the body of knowledge every day, collaboration is the only feasible way to keep the CBK up-to-date. We acknowledge this by choosing a collaborative approach backed by SMW+ to build up a CBK for engineering secure software and services relying explicitly on user participation.

A CBK has the greatest benefit, if it is complete, up-to-date, and valid. Especially in the beginning of such a project this is not the case, leading to low acceptance and low user participation, if launched for the public too early. We therefore conceived three phases, each with a different focus and participation style in order to work against this effect. Furthermore, the CBK content has to be revised on a regular basis to ensure a high quality, which can be summed up by the question: How is content provisioning and quality assurance supported best while relying on user participation?

We present the three phases in Sect. 4.1, and we introduce quality assurance means in Sect. 4.2.

4.1 Three Phases

The first phase is a planning phase, in which all discussed aspects are considered while preparing the initial CBK structure and planning.

During an inception phase, content is provided by a closed user group, consisting of experts from different areas within the secure software development field. These experts are mainly researchers from NESSoS, where we profit from the opportunity of having so many researchers linked together through the NoE. The writing process is managed by a central coordinator, who creates the initial CBK structure, defines clear writing responsibilities, watches deadlines,
and ensures quality (see Sect. 4.2). At the end of this phase, the result is a sound CBK content base providing a complete, up-to-date, and validated state-of-the-art of this interdisciplinary research field. A high benefit of this work is expected for researchers from service, security, and software engineering. But also practitioners will find it interesting to get a glimpse on what current research has to offer.

The run phase is marked by the launch of the CBK for the general public in terms of reading and writing. At this point in time, the CBK should provide a complete overview of the research field of secure software development. To launch the CBK with a sound content base, which has mostly been created by the community itself, increases the attraction of the CBK for other people that we considered in the use cases (see Sect. 2). Especially for practitioners and for stakeholders other than researchers, learning trails will guide through the vast amount of research results, with respect to their expert level (see Sect. 3.3).

4.2 Quality Assurance

The SMW+ supports quality assurance tasks in different ways. Authors are notified via e-mail, when other people have modified their KO. In the case of vandalism or wrong information, it is possible to revert the changes back to a previous state, making use of the versioning functionality of SMW+. If provided information is controversial, the system allows users to have discussions for each knowledge object on the same page. If new attributes are introduced into the ontology, it is usually the case that these attributes lack of values for existing individuals. SMW+ provides a mechanism to gather information about missing attribute values and allows us to notify the respective author. Furthermore, SMW+ provides an elaborate access control mechanism, which makes it possible to define groups and assign read and write access rights. We make use of this mechanism in order to introduce roles, each with different access rights for, e.g., KOs, KAs or administrative functions of SMW+.

Depending on the project phase, quality is assured in different ways.

In the inception phase, quality is assured by a restrictive access control, allowing only partners of the network to have full access to the CBK. Additionally, a central quality assurance (QA) team will start their work having a regular qualitative review on the contents of the CBK, flagging them with a marker indicating when a KO needs to be revised due to a low content quality. But not only the QA team is able to flag KOs. Everyone is allowed to flag an article if vandalism is detected.

While the inception phase is characterized by a controlled environment through a closed user group, the run phase takes a more decentralized and community-driven approach. Since we assume that we will reach a critical mass of users during a short period after going public, content contribution will increase and self-regulation will become realistic. Thus, quality assurance is incrementally shifted over to the user, because the QA task is no more feasible to be exercised by a few experts. Instead, experts will rather be assigned responsibilities along the knowledge areas, taking a more moderating role.

As already mentioned, the underlying SMW+ platform supports both approaches, providing adequate collaboration functionality such as feedback and access control mechanisms.

5 RELATED WORK

The concept of a codified BOK is not new and can be found in many different disciplines. Compared to our CBK, they all differ in how they were created and in how knowledge is codified.

All of the BOKs presented in the following were created top-down. By this we mean that an expert team was formed or authors were chosen to write articles. Our approach comprises a top-down phase, but also a bottom-up phase in which the CBK is opened to the public in terms of reading and writing (see Sect. 4). This is comparable to the shift from the creation of the Encyclopedia Britannica to the creation of Wikipedia, acknowledging the fact that new knowledge is generated very fast and by many people these days.

A BOK mentioned before is the “Software Engineering Body of Knowledge” aka SWEBOK (Div. Auth., 2005), the most prominent among all other BOKs within the SE discipline. The Computer Engineering Body of Knowledge (Computing Curricula 2005) (Div. Auth., 2006) and the Software Engineering Education Knowledge (SEEK) (part of (Div. Auth., 2004)) have a special focus on SE education. The Project Management Body of Knowledge (PMBOK) (Project Management Institute, 2008) is also well-known and covers project management knowledge in general. In the security field, BOKs do exist with different focuses promoted by both industry and governments such as the Information Technology Security Essential Body of Knowledge (U.S. Department of Homeland Security Office of Cybersecurity and Communications National Cyber Security Division, 2008).

A more collaborative approach is taken by the two
BOK projects *Usability BOK*\(^4\) and *Build Security In*\(^5\) by the U.S. Department of Homeland Security, both also fostering user participation to provide content following a bottom-up approach.

All of the BOKs presented so far do not allow to be queried elaborately. Many BOKs only exist as a book with access possibilities given by the table of contents or a key word index, while others also provide a hypertext system, allowing one to browse content along links, such as the online version of the SWEBOK\(^6\), the *IEEE Body of Knowledge on Services Computing*\(^7\) or the *Guide to the Systems Engineering Body of Knowledge (G2SEBoK)*\(^8\). We go one step further and allow more elaborate queries through our CBK ontology with which we are able to semantically enrich all CBK contents.

The need for defining a common terminology for different interdisciplinary communities led to a large number of publications in this area, e.g., the work by Fabian et al. (Fabian et al., 2010) for SE and security. Similar to our common terminology concept, their approach defines a taxonomy relating fundamental notions across the different disciplines, and they specify to what extent notions from one discipline can be translated into notions of the other discipline. The main difference to our work is that Fabian et al. do not complement their results by further concepts such as KAs, learning trails, etc. to create a CBK.

The CBK introduces a mechanism to consolidate interdisciplinary communities:

- KOs allows users to *structure knowledge* such as best practices and research results according to their type. Provided content is semantically enriched in an automated manner. This allows users to browse, compare, and run complex queries on the CBK.
- The CBK introduces a mechanism to *group knowledge* into KAs. This provides access to the CBK via a hierarchical taxonomy and represents a valuable instrument to discover gaps in practice and research.
  - The *common terminology* helps the community to find a common language of the different disciplines, and to define and use translations.
  - *Learning trails* provide access to the CBK for a broader audience, practitioners and researchers in particular.
  - The *ontology* which underlies the whole CBK supports several representations of the CBK, including known ones like books and hyper texts. It provides the flexibility to create customized representations.
  - *User participation* is supported by adequate processes and by the chosen SMW+, which might lead to a more up-to-date, a more comprehensive, and a sustainable CBK.
  - The realization of the CBK through SMW+ provides a smart means to allow collaborative creation and editing since SMW+ is fully integrated with the design of the ontology.

In the future, we plan to elaborate more on using the ontology properties for specific use cases such as identifying gaps in research areas. Moreover, we want to include user rating mechanisms in our CBK concept.

## 6 CONCLUSIONS

In this paper, we presented a conceptual approach to support bringing together disciplines based on a CBK. We demonstrated our approach using the NESSoS EU project and the interdisciplinary field of engineering secure software and services.

Our approach comprises the following main contributions to **consolidate interdisciplinary communities:**

- KOs allows users to *structure knowledge* such as best practices and research results according to their type. Provided content is semantically enriched in an automated manner. This allows users to browse, compare, and run complex queries on the CBK.
- The CBK introduces a mechanism to *group knowledge* into KAs. This provides access to the CBK via a hierarchical taxonomy and represents a valuable instrument to discover gaps in practice and research.

### REFERENCES


---

\(^4\) [http://www.usabilitybok.org](http://www.usabilitybok.org)

\(^5\) [https://buildsecurityin.us-cert.gov](https://buildsecurityin.us-cert.gov)

\(^6\) [http://www.computer.org/portal/web/swebok](http://www.computer.org/portal/web/swebok)

\(^7\) [http://www.servicescomputing.tv](http://www.servicescomputing.tv)

\(^8\) [http://g2sebok.incose.org](http://g2sebok.incose.org)