L. Fritsch et al. (Eds.): Open Identity Summit 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 167

A Semantic Data Model for the Development of Secure and Valuable Software

Andrea Horch[1], Uwe Laufs[2], Rachelle Sellung[3]

Abstract: IT security is a crucial non-functional software requirement. Nevertheless, there are several other aspects that a softwares' market success depends on. Therefore, it is vital that during the development process software developers consider different disciplines needs that essentially add value when going to market such as usability and socio-economics. The project CUES addresses these aspects by developing an interdisciplinary and integrated guidance tool, called the Wizard. The Wizard is designed to support software developers with interdisciplinary knowledge during the software development processes. The core of the Wizard builds a knowledge base, which is based on a semantic data model. While the semantic data model is finished, the Wizard is still undergoing development in the CUES project and is not yet complete. This paper presents the semantic data model as a first project result and as the core element of the Wizard. The proposed data model stores knowledge about software development processes, methods and tools in order to derive problems and corresponding solutions which may occur in real software development processes.

Keywords: Data Model, Software Development, IT Security, Usability, Socio-economics.

1 Introduction

The consideration of different sustainability dimensions as security, usability and socio-economic aspects during the software development process is crucial for the success of the software on the market [KAB15], [ZR11a], [ZR11b]. Security is a very important non-functional software requirement, however the usability of software is essential for user acceptance [FMS07]. There is a need for novel approaches that support the software development process by considering interdisciplinary software requirements in order to create secure and usable software, which can achieve sustained and long-term success on the market [FMS07]. The project CUES[4] addresses the problem of "protecting the wrong things, or protecting the right things in the wrong way" [An01] as well as the issue of building software not needed by the market. The project develops an integrated guidance tool, the Wizard, which supports software developers during the whole software development process in order to create secure and usable software with a value for the market [HS16]. To achieve this goal the Wizard recommends suitable methods of different disciplines as

[1] Fraunhofer IAO, Competence Team Identitätsmanagement, Nobelstr. 12, 70569 Stuttgart, andrea.horch@iao.fraunhofer.de
[2] Fraunhofer IAO, Competence Team Identitätsmanagement, Nobelstr. 12, 70569 Stuttgart, uwe.laufs@iao.fraunhofer.de
[3] IAT der Universität Stuttgart, Competence Team Identitätsmanagement, Nobelstr. 12, 70569 Stuttgart, rachelle.sellung@iat.uni-stuttgart.de
[4] http://www.cues-projekt.de/

security, usability and socio-economics to the developers within every phase of the development process. To manage this task the system needs knowledge about the process and its single phases, the methods and their quality. The details about the project and the resources are provided by the users, whereas the knowledge about the methods comes from domain experts. The knowledge is stored as a semantic graph within the system's graph database. The development of the Wizard is still on-going work. This paper presents the semantic data model behind the graph, on which the recommendations for the phases of the software development process will be based.

2 Related Work

There is previous work on the implementation, structuring and documentation of software development processes based on semantic technologies to optimize software development. The authors of [G01] describe a framework for the description of software development processes in the form of a process ontology defining persons, roles, activities, problems and process patterns as well as the relationships between these concepts. The main concept of the framework is given by process patterns which define solutions for particular problems which may occur during the software development process. The framework supports the documentation of knowledge about the development to improve and tailor adequate software development processes within companies.

ODE (Ontology-based software Development Environment), an ontology-based approach to integrate tools, processes and knowledge into software engineering environments (SEEs), is described in [F03]. The ALIGEND project[5] has developed a set of ontologies, which build the ALIGNED generic metamodel for software and data engineering. The metamodel supports the documentation of the development and design processes of the software development and it supports to define processes as the software life cycle. Additionally, it provides mechanisms and vocabularies for reporting quality measures and to describe the data within the software development process. There are also low level approaches for semantic-enabled software development to automatically generate software code from data models based on semantic technologies such as the approaches found in [BR06] and [Zy09]. Even though these approaches do not improve the development process itself, they can be useful within the implementation phase. Additionally, there are models and ontologies developed in the areas of Model-Driven Software Development (MDSD) and Ontology-Driven Software Development (ODSD), which support the software developers in automatically generating executable software from formal models Pa12.

In contrast to the approaches mentioned above, the data model introduced in this paper is not designed to automatically generate software code directly from the structural or semantic models nor it will store only knowledge in a model as a documentation or report. Instead, it supports software developers during the whole software development process by identifying potential weaknesses or problems, by providing adequate solution suggestions as well as by animating them to think about important socio-economic and usability aspects which affect the software development.

[5] http://aligned-project.eu/

3 Semantic Data Model

The proposed data model is shown in Fig. 1. The model was created by following the On-To-Knowledge Methodology (OTKM) presented in [SSS04] as it simplifies the creation process by dividing the processes of building the meta model of the data model and the acquisition and management of knowledge into separate parts. The resulting data model comprises 9 entity classes which are interconnected by several relationships. The resources *User* and *Project* as well as their outgoing connections represent user data given by the users of the Wizard, whereas the other entity classes *Question*, *Problem*, *Solution*, *Approach*, *Tool*, *Process*, *Phase* and their outgoing connections depict expert knowledge provided by domain experts. The expert knowledge is the core element of the data model which enables the retrieval of solutions in the form of recommendations for the use of adequate methods and tools depending on the actual process situation (e.g. a problem) of a user, which is derived from the user data.

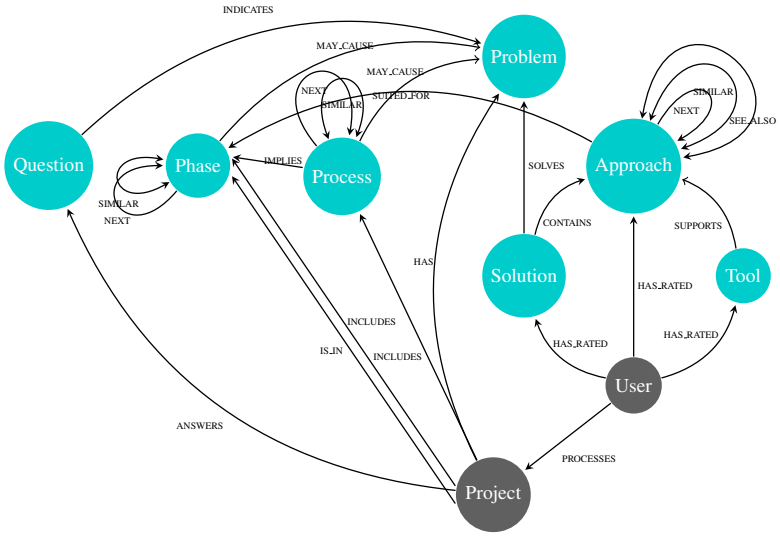

Fig. 1: Semantic Data Model of Software Development Projects

An instance of the entity class *User* contains the information about a certain user. The *User* works in a sector of a company and has contact details. The *User* processes one or more *Projects* and can give ratings for *Solutions*, *Approaches* and *Tools*. *Project* instances store information about a project, which include the name of the *Project* as well as the start date and the end date of the *Project*. A *Project* includes one or more *Processes* as well as the *Phases* of the *Processes*. In accordance with its time schedule a *Project* is in an actual *Phase*, which is defined by an *IS_IN* relationship.

```
Resource User:
    string first_name
    string last_name
    string email
```

```
    string sector
    string company

Resource Project: string name, date start_date, date end_date
```

As a *Project* follows a process model defined by a *Process* instance, as e.g. *V-Model*, a *Process* is included in a *Project* by an *INCLUDES* relationship. Moreover, the *Project* instance itself has a name, a textual description, a list of links to external information sources and a list of links to internally stored files as pictures or other documents.

```
Resource Process:
    string name
    string description
    list   [ext_link_1, ext_link_2, ...]
    list   [int_link_to_file_1, int_link_to_file_2, ...]
```

The *INCLUDES* relationship contains the information when a *Process* starts and when it ends for a defined *Project*. A *Project* can include several *Processes*. The temporal sequence of the *Processes* is ordered by *NEXT* relationships between the *Process* instances.

```
Relationship INCLUDES: date start_date, date end_date
```

A *Process* instance implies several *Phases* in a specific temporal order, which is defined by *NEXT* relationships between the different *Phases*. *Phase* instances consist of the same attributes as *Process* instances.

```
Resource Phase:
    string name
    string description
    list   [ext_link_1, ext_link_2, ...]
    list   [int_link_to_file_1, int_link_to_file_2, ...]
```

Within a *Process* or a *Phase* instance may occur different problems. These *Problem* instances are connected to the respective *Process* or *Phase* instance through a *MAY_CAUSE* relationship. A *Problem* instance has a name, a textual description and a score which states how serious the *Problem* is regarding to the success of the *Project*.

```
Resource Problem: string name, string description, float score
```

In the best case, for a certain *Problem* instance there exist one or more *Solution* instances in order to solve the problem. A *Solution* instance consists of a name, a textual description and lists of external and internal links, which describe the *Solution* instance in more detail.

```
Resource Solution:
    string  name
    string  description
    integer estimated_duration
    list    [ext_link_1, ext_link_2, ...]
    list    [int_link_to_file_1, int_link_to_file_2, ...]
```

Solution instances contain one or more *Approach* instances, which represent methods or best practices do be done in a certain *Phase* of a *Process* or in order to solve a specific

Problem. *Approach* instances include all necessary attributes to describe the respective instance as e.g. name or type (e.g. method or best practice).

```
Resource Approach:
   string  name
   string  type
   string  description
   link    interal_link_to_key_visual
   integer estimated_duration
   list    [tag1, tag2, ...]
   boolean includes_end_user
   float   difficulty
   string  motivation
   string  example
   list    [link_to_reference1, link_to_reference2, ...]
   list    [ext_link_1, ext_link_2, ...]
   list    [int_link_to_file_1, int_link_to_file_2, ...]
```

Approach instances may be supported by one or more *Tool* instances which are connected to the respective *Approach* instance via a *SUPPORTS* relationship.

```
Resource Tool:
   string name
   string type
   string description
   list   [ext_link_1, ext_link_2, ...]
   list   [int_link_to_file_1, int_link_to_file_2, ...]
```

Another core element of the data model is the *Question* entity. *Question* instances depict real-world questions, which support users to identify existing or possible problems within the process phases of a project. Therefore, *Question* instances are related to *Problem* instances through a *INDICATES* relationship as the *Question* instance indicates the occurrence of a certain *Problem* instance. In order to be able to know if a *Question* instance is applicable for a specific *Project* instance the user needs to answer the question with regard to the certain *Project* instance processed by the user represented by a *User* instance. The assessment of a user regarding a certain question is realized as an *ANSWERS* relationship between a *Project* instance and an instance of a *Question*.

```
Resource Question:
   string question_text
   list [answer_option_1, answer_option_2, ...]
   float threshold

Relationship ANSWERS: float answer, boolean ignore

Relationship INDICATES: float score
```

Process, *Phase* or *Approach* instances, which are similar to another instance of the same entity class are connected through a *SIMILAR* relationship including a similarity value defining the degree of similarity. The *SIMILAR* relationship and the *SEE_ALSO* relationship are the only bidirectional relationships of the data model. The *SEE_ALSO* relationship links from one *Approach* instances to another to declare that for users who are interested in the first *Approach* instance may also be interested in the other.

```
Relationship SIMILAR: float similarity
```

Users are allowed to rate *Solution*, *Approach* and *Tool* instances, which is realized by a *HAS_RATED* relationship between the *User* instance which represents the rating user and the rated instance.

```
Relationship HAS_RATED: float user_rating
```

4 Evaluation

The methodology presented in [SSS04], which we have selected to create the semantic data model, includes three different strategies for the evaluation of the data model: (1) a technology-focused evaluation, (2) a user-focused evaluation and (3) an data model-focused evaluation. For the technology-focused evaluation we created a test data set based on the data model and wrote some Cypher queries to check the consistency.

Approach	Problem Identification	Solution Provision	Process Control	Process Tailoring	Process Documentation
Framework of Gnatz et al. [G01]	○	●	●	●	●
ODE [F03]	○	◐	●	●	●
ALIGNED Metamodel	○	◐	●	●	●
Proposed Data Model	●	●	●	●	●

Legend: ○ not supported, ◐ partially supported, ● supported

Tab. 1: Comparison to other data models

For the user-focused evaluation we have extensively discussed the initial design of the data model together with a domain expert. Additionally, we have compared the proposed data model to existing solutions as shown in Table 1. Further experts will be consulted in the near future. The criteria to compare the different data models derived by the existing work and enriched with aspects to be able to identify problems and to provide adequate solutions were the following: Problem Identification, Solution Provision, Process Control, Process Tailoring and Process Documentation.

As shown in Table 1 all data models allow process control, process tailoring and process documentation, but they leak to identify possible problems.

For the data model-focused evaluation the suitability of the data model to depict all required use cases of the Wizard has been checked. These use cases are: (1) The browsing function in order to find adequate solutions and hints for software development through filtering by process step, expert level or discipline. The use cases of the browsing function can be handled by the data model as all necessary information is directly stored in the *Approach* instances. (2) The guiding function, where the most complex use case of the *problem identification* is shown in Fig. 2, which exemplary shows how problems within

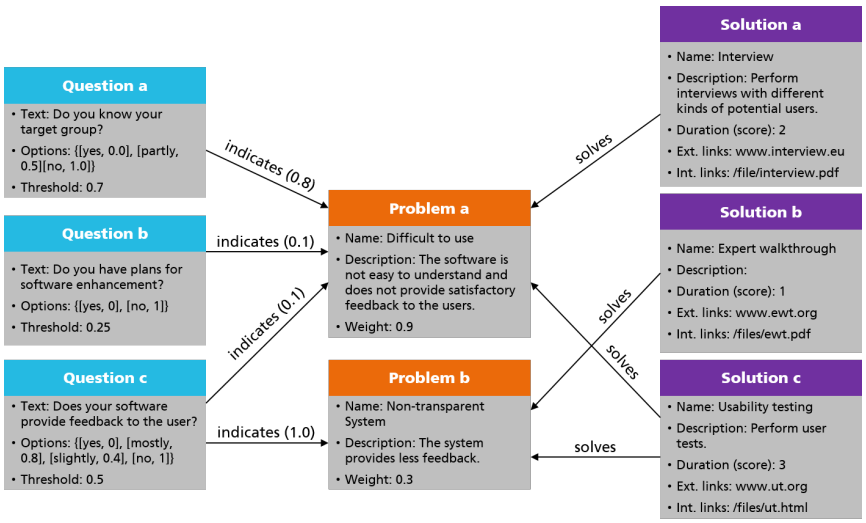

Fig. 2: Problem identification through the proposed data model.

the software development can be identified by using the data model: Within a defined process step (e.g. analysis step) the users are ask the questions a, b and c. The users answer the questions by e.g. using a slider in order to state the degree of fulfillment according to the ask subject of the question. A threshold defined in the *Question* instance determines if the *Question* may indicate *Problem* related to the *Question* instance. Additionally, a score defined by the experts defines how strong the *Question* instance is an indicator for the related *Problem* instance (weight factor). Given all that information and weighting the answers with the defined weighting factors the data model can derive the problems which may exist within a certain process step.

5 Conclusions and Future Work

The contribution of this paper is a data model to support software developers by providing relevant knowledge as well as by indicating possible problems during the whole software development process. We especially aim to assist developers of IT security solutions to use an interdisciplinary approach for software development in order to create software which is comfortable to use and which can compete successfully on the market. The data model is part of on-going work with the goal to develop an integrated guidance tool, the Wizard, which supports the software developers by providing recommendations for adequate methods and other practices to handle their work and deal with the problems in each process phase. The data model was assessed by three different types of evaluation: (1) a technology-focused evaluation, (2) a user-focused evaluation and (3) an data model-focused evaluation. As the implementation of the Wizard is still in progress, it was not possible at this time to conduct an empirical evaluation, but this aspect will be addressed in future work.

ACKNOWLEDGEMENTS We thank and acknowledge the Baden-Württemberg Stiftung for financing the CUES project. For more information please visit: `https://www.bwstiftung.de/`.

References

[An01] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st. Auflage, 2001.

[BR06] B. Bauer; S. Roser. Semantic-enabled Software Engineering and Development. In *INFORMATIK 2006 - Informatik für Menschen, Lecture Notes in Informatics (LNI)*, Seiten 293–296, 2006.

[F03] R. A. Falbo et al. ODE: Ontology-based software Development Environment. In *Proceedings of the IX Congreso Argentino de Ciencias de la Computación (CACIC 2003)*, CACIC 2003, Seiten 1124–1135, Argentinien, 2003. SEDICI.

[FMS07] I. Flechais; C. Mascolo; M. A. Sasse. Integrating Security and Usability into the Requirements and Design Process. *Int. J. Electron. Secur. Digit. Forensic*, 1(1):12–26, Mai 2007.

[G01] M. Gnatz et al. Towards a Living Software Development Process Based on Process Patterns. In *Proceedings of the 8th European Workshop on Software Process Technology*, EWSPT '01, Seiten 182–202, London, UK, UK, 2001. Springer-Verlag.

[HS16] J. Hofer; R. Sellung. An interdisciplinary approach to develop secure, usable and economically successful software. In *Open Identity Summit 2016, 13.-14. October 2016, Rome, Italy*, Seiten 153–158, 2016.

[KAB15] S. A. Koçak; G. I. Alptekin; A. B. Bener. Integrating Environmental Sustainability in Software Product Quality. In *Proceedings of the Fourth International Workshop on Requirements Engineering for Sustainable Systems, RE4SuSy 2015, co-located with the 23rd IEEE International Requirements Engineering Conference (RE 2015), Ottawa, Canada, August 24, 2015.*, Seiten 17–24, 2015.

[SSS04] Y. Sure; S. Staab; R. Studer. On-To-Knowledge Methodology (OTKM). In *Handbook on Ontologies*, Seiten 117–132. Springer, 2004.

[ZR11a] J. Zibuschka; H. Roßnagel. A framework for designing viable security solutions. In *Workshop on Information Security & Privacy (WISP)*, 2011.

[ZR11b] J. Zibuschka; H. Roßnagel. A structured approach to the design of viable security systems. In *Proceedings of the Information Security Solutions Europe Conference (ISSE)*, Seiten 246–255, 2011.

[Zy09] S. Zykov. ConceptModeller: a Graph-Based Semantic Modeling Tool for Building Enterprise Applications. In *Supplementary Proceedings of the 17th International Conference on Conceptual Structures (2009)*, Jgg. 483, 2009.