e-Informatica Software Engineering Journal, Volume 7, Issue 1, 2013, pages: 15–24, DOI 10.5277/e-Inf130102

Efficient Adoption and Assessment of Multiple Process Improvement Reference Models

Simona Jeners*, Horst Lichter*, Carlos Gomez Rosenkranz*
*Research Group Software Construction, RWTH Aachen University
simona.jeners@swc.rwth-aachen.de, lichter@swc.rwth-aachen.de,
rosenkranz@swc.rwth-aachen.de

Abstract

A variety of reference models such as CMMI, COBIT or ITIL support IT organizations to improve their processes. These process improvement reference models (IRMs) cover different domains such as IT development, IT Services or IT Governance but also share some similarities. As there are organizations that address multiple domains and need to coordinate their processes in their improvement we present MoSaIC, an approach to support organizations to efficiently adopt and conform to multiple IRMs. Our solution realizes a semantic integration of IRMs based on common meta-models. The resulting IRM integration model enables organizations to efficiently implement and asses multiple IRMs and to benefit from synergy effects.

1. Introduction

Nowadays, the software market is expanding and clients are requesting better, faster, and cheaper software products. However, the Standish Group regularly reports that the failure rate of IT-projects is still too high: 68% of IT-projects do not meet their deadlines nor achieve the requested quality or are cancelled [1]. One important impact factor to project success is the quality of the applied IT-processes because software quality heavily depends on these processes. Hence, more and more organizations are obligated to identify, structure, and improve their processes systematically. Because the process improvement road is quite long and expensive it needs to be guided. To support process improvement different IT reference models such as CMMI [2], ISO/IEC 15504 [3], COBIT [4] or Functional Safety [5] may be considered and applied. Improvement Reference models (IRMs) are collections of best practices based on the experience and knowledge of many organizations. We call these best practices procedures. The IRMs are published as maturity-, procedure- or quality-models as well as standards or norms. Although IRMs exist for different IT areas, such as Software and System Development, IT Governance, Software Safety or IT Services, they may address similar topics. For example, project or risk management is addressed in almost all IRMs. The adoption of multiple IRMs allows an organization to exploit synergy effects between them. On the one hand organizations can address coordinately different and common areas. On the other hand the weaknesses of a single IRM can be overcome by the strengths of others.

Although there is free information available about each single IRM, there is no integrated solution that makes a collection of IRMs more transparent and supports organizations in the adoption and assessment of IRMs. This lack of transparency makes the effort for process management and assessment of multiple (evolving) IRMs unnecessary high. The main problems that hamper organizations to use the experience and knowledge reflected by IRMs are:

- **IRMs exist in different shapes, for multiple domains and cover many areas**. As already mentioned there are many types of IRMs, such as norms, quality-models or standards, which vary in certain areas but also share some ones in multiple IT domains. Organizations usually do not understand the similarities and differences between the IRMs in this collection.
- **IRMs are based on different structure and terminology**. Because IRMs are developed for different IT domains and by different institutions, each single IRM defines its own specific structure and uses a specific set of terms. Hence, different terms are used for the same semantic concept. Both, the different structure and different terminology hamper to understand and adopt IRMs.
- **IRMs may address similar topics**. Although IRMs exist for different IT areas, they may address similar topics. For example, project or risk management is addressed in almost all IRMs. To efficiently adopt multiple IRMs the organization must be able to easily compare the selected IRMs and identify their similarities and differences. Procedures of IRMs can be described either very generally or more concretely. The organizations should recognize similar procedures to better understand the abstract requirements of the more general one. Furthermore, organizations should be aware of the essence of similar procedures for a better overview and understanding. The specific details of each IRM should also be easy to identify if necessary.
- **IRMs are changing**. Since IRMs are updated continuously and new IRMs are developed organizations must keep pace with their evolution and must be able to understand and apply the changes.

1.1. Goals and Solution Approach

In order to solve the problems mentioned above we propose a new approach called MoSaIC (Model based Selection of Applied Improvement Concepts) aiming to achieve transparency and to support organizations in an effective adoption and assessment of multiple IRMs. The goals associated with MoSaIC are:

- **G1**: Facilitate the understanding and avoid misinterpretations of IRMs.
- **G2**: Identify similar procedures and extract their essence as abstract practices.
- **G3**: Provide traceability between abstract practices and their instances in the IRMs.
- **G4**: Allow an easy identification of the dependencies between procedures or process areas of different IRMs.
- **G5**: Support different levels of abstraction of IRMs.
- **G6**: Support an easy update of changed IRMs and an easy integration of new ones.

For short, we achieve transparency by a seamless and semantic integration of different IRMs. Although the integration of IRMs is a central issue of MoSaIC, it addresses further challenges as well, e.g. the systematic selection of IRMs or parts of IRMs that are best suited for an organization. To address this problem, not only IRMs but also other information (e.g. business goals or constraints) is integrated into MoSaIC. We focus here on MoSaIC's model based integration approach of IRMs only.

1.2. Related Work

The need of a process architecture in a multi-model context is mentioned in a series of articles from SEI [6]. This raises the awareness to define a generic and integrated model which allows describing different IRMs as well as organizational processes. This model should make IRMs more transparent and support organizations to find similarities between different IRMs. Basic elements mentioned in [7], [6] or [8] such as inputs, outputs, roles, their relations are part of our integration model as well.

Ferreira et al. [9] present an approach to achieve transparency of IRMs by comparing IRMs. The problems mentioned above, the different abstraction levels of IRMs, their overlapping, and their complexity are also mentioned. This approach tries to solve these problems by defining metrics to manually compare IRMs. A manually comparison of the IRMs

is performed in [10] to identify similar practices of different IRMs and their essence. We also provide support for identifying abstract practices but based on an automatic comparison. Our best practices can be easily traced to their instances in the different IRMs. The traceability is also addressed in a case study for the integration of large aerospace IRMs [11]. Here, different types of traceability between activity, input or output elements are defined. However, our approach extends one traceability type by considering the semantic relations between these elements. There are many contributions in the literature to the integration of IRMs and their comparison. For example in [12], [13] or [7], the authors define a common structure to link IRMs and reveal their similarities. For this purpose similar procedures of IRMs are connected manually. In contrast to the first two approaches we model on a more fine grained level and identify procedure' elements to support an automatic comparison. The third approach also addresses this fine granularity, it connects similar elements of different procedures but does not define what does similarity means. We differentiate between different similarityrelations to get a more accurate degree of similarity between IRMs. Soto and Münch [14] formalize and automatically compare the IRMs and internal processes but also by only considering the semantic equivalence between the basic elements (activities, stakeholder, and products). Their approach also addresses the problem of IRMs' evolution. Their comparison approach is used to support the changes of the internal processes or IRMs and to assure a continuously compliance.

The remaining of this paper is organized as follows. In the second section we describe the elements and relations of the MoSaIC's IRM integration approach. In section 3 we present excerpts of a MoSaIC case study applied to model parts of CMMI, COBIT and Functional Safety. Based on this case study we finally discuss the results of our evaluation and give an overview to future work. Conclusions and a summary conclude this paper in the last section.

2. The MoSaIC IRM Integration Approach

In the following we describe the MoSaIC way to integrate IRMs. First, we motivate and give a short overview of our integration approach. Then we present in detail the two meta-models of MoSaIC that provide the basis for a model based IRM integration approach.

The main idea of MoSaIC's IRM integration approach is to normalize IRMs based on a joint structure and on a common set of terms. According to mega modeling theory [15], we can normalize by defining appropriate meta-models. We have analyzed published IRM meta-models, e.g. the one of CMMI, extracted and added only elements that are sufficient to achieve the goals defined at the beginning of this paper.

To model different IRMs the same way we have developed the so called *Integration Structure Meta-Model* (IS Meta-Model). It defines core and additional IRM element types introduced by different IRMs as well as fine grained IRM concept element types, such as activities, artifacts or roles. While the core and additional element types allow providing a rough overview of the most important aspects of IRMs, the conceptual elements types allow the integration of concrete and abstract IRMs and a detailed comparison of IRMs. A *IRM concept* (concept for short) is a word or the smallest combination of words that has a unique meaning in the context of IRMs. For example "project plan" or "work breakdown structure" are concepts used in IRMs. Concepts can be derived from activities, roles, inputs and outputs of IRMs.

For each IRM, such as CMMI, SPICE, COBIT or ITIL, we have extracted the core, additional and conceptual information and created respective *IRM Integration Structure Models* (IRM-ISMs). Mappings from the single IRM-ISMs to the IRMs' original structures provide more information if needed.

Furthermore, IRMs should be modeled using the same terminology. We introduce a mechanism to translate and map the terms/concepts used by each single IRM to a common normative set of terms/concepts. For this purpose we

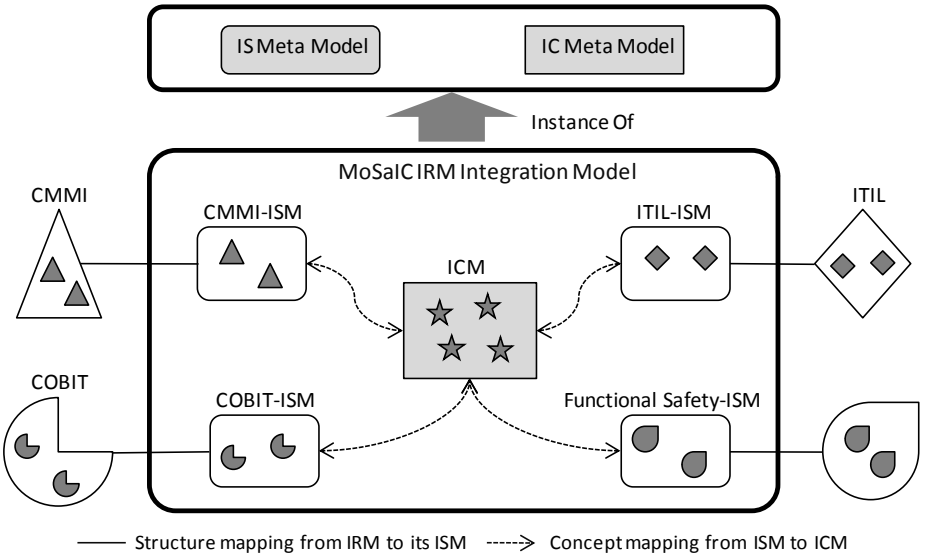

Figure 1. Architecture of the MoSaIC IRM integration approach

have created a model containing the closure of all IRM concepts, called *Integration Concept Model* (ICM). For each specific conceptual element in a IRM-ISM there is a corresponding general ICM concept. A general concept defined in the ICM can be mapped to several finer conceptual elements of one single IRM or of several different IRMs if these together are semantically equivalent to the general concept. The ICM concepts can be seen as dictionary entries having synonyms and explanations in the different IRM-ISMs. This conforms to what is called *Linguistic concordance* (list of words with their immediate contexts) and supports a better understanding and avoidance of misinterpretations of the IRMs' content. Obviously, the ICM is the sole instance of its meta-model, the *Integration Concept Meta-Model* (IC Meta-Model) and links all IRM-ISMs.

In order to model the semantic of IRMs appropriately and to further improve their comprehension we have enhanced our meta-models by attributes and semantic relations (e.g. to model similarity between concepts). For example, an ICM concept has a *definition* attribute (usually given by an expert) or ICM concepts may be related by a *generalizationOf* relation. To summarize, the ICM specifies the common concept language for IRMs and facilitates the understanding of their content.

Figure 1 schematically depicts the purpose and application of both meta-models and their respective concrete models IRM-ISMs and ICM. The different structures of IRMs are represented by different geometrical shapes while the different used terminology is symbolized by different small geometrical internal shapes. For each IRM a corresponding IRM-ISM is shown (e.g. CMMI-ISM) being part of the overall MoSaIC IRM Integration Model. All ISMs are instances of the IS Meta-Model. Hence, all ISMs use the same set of element types which makes them analyzable and comparable. ICM (the only instance of its IC Meta-Model) is part of MoSaIC's IRM Integration Model as well. It defines all concepts and semantically links all IRM-ISMs by connecting related concepts across the borders of single IRMs.

Figure 2 shows the most important elements of the ***Integration Structure Meta-Model***. For the representation we use a notation similar to UML class diagrams. We have grouped the elements in three packages:

1. **Core** contains elements mostly defined by meta-models of existing IRMs.
2. **Add-Ons** offers elements that are not always present in all IRMs.
3. **Concepts** contains elements to model concept information of IRMs on a fine grained level.

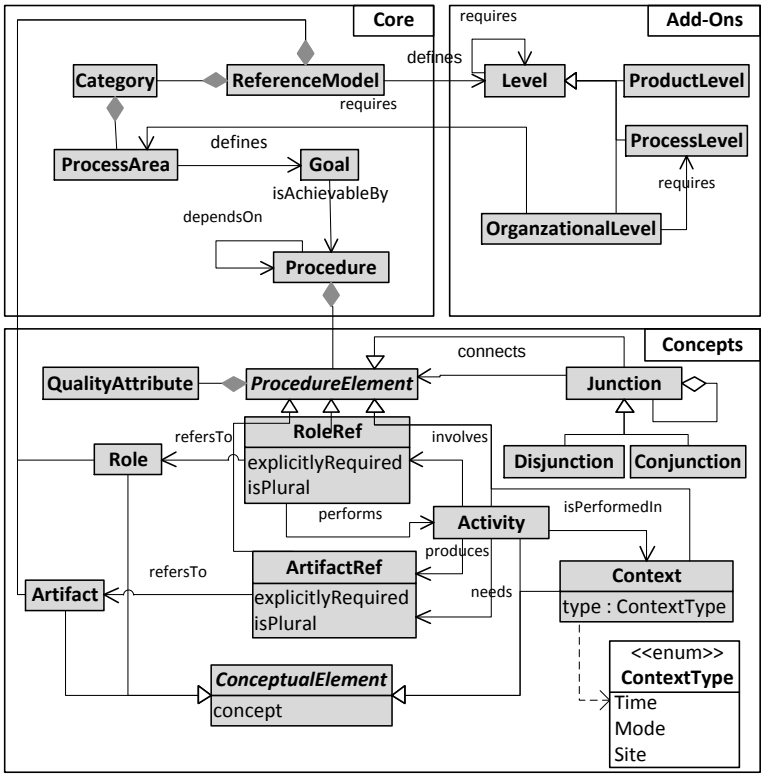

Figure 2. The Integration Structure Meta-Model

Package **Core**. The top element of this package is called *ReferenceModel*. It represents a certain IRM and is structured by means of *Categories*. A *Category* defines a certain topic that is addressed in one or more processes defined by the IRM. A *ProcessArea* addresses a topic to be improved and is part of exactly one *Category*. Each *ProcessArea* defines one or more *Goals*. The requirements to achieve the goals are described by *Procedures*. A *Procedure* defines one or more *Activities* with their *Roles*, *Inputs* and *Outputs* (called *Artifacts*). By means of the *dependsOn* relation dependencies between Procedures are modeled (e.g. if a procedure needs as input the output of another procedure).

Package **Add-Ons**. *ReferenceModels* may define Levels. A *Level* represents a degree that an organization can reach by applying the IRM. By means of the relation requires a hierarchy of levels can be modeled. Three special kinds of levels are defined: *OrganizationalLevel*, *ProcessLevel* (i.e. a level of a *ProcessArea*) and *ProductLevel*. An *OrganizationLevel* may require a certain *ProcessLevel* and may also require that certain *ProcessAreas* are established in the organization.

Package **Concepts**. Our approach to integrate different IRMs is centrally based on the notion of *Concepts*. Therefore, we model the specific conceptual elements of each single IRM in the respective ISM as well as their corresponding general concepts in the ICM. This enables to link similar specific concepts of different IRMs and to compare IRMs.

Activities, *Roles*, and *Artifacts* of IRMs are concrete *ConceptualElements*. An *Activity* may involve *Roles* and is performed by one or more *Roles*; it usually needs and produces *Artifacts*. Because a certain *Role* or *Artifact* can be used in different *Procedures* of a IRM, only their references are associated with *Activities*. Hence, *Activities*, *RoleRefs* and *ArtifactRefs* are the central aspects of a *Procedure*, abstractly modeled by class *ProcedureElement*. *ProcedureElements* have additional information that specifies their usage in *Procedures*. For example, they may be characterized by *QualityAttributes* (e.g. “formally approve the project plan”). Furthermore, they may

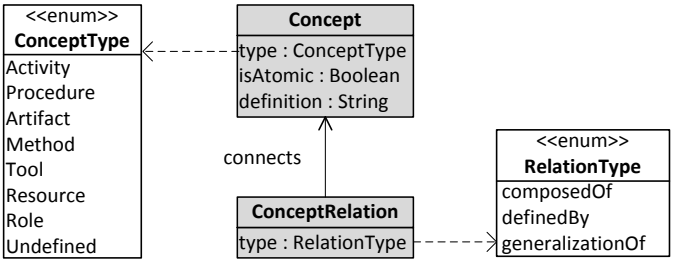

Figure 3. The Integration Concept Meta-Model

be connected by the logical relations *Conjunction* or *Disjunction*, because a procedure may require multiple *ProcedureElements* of the same type (e.g. "eliminate or minimize project risks"). By means of the *Junction's* self composition relation each combination of logical relations can be modeled (e.g. "carrying out the applicable overall, the E/EPS and software lifecycle phases").

However, there are some differences between the concrete *ProcedureElements*. One the one hand, *Roles* and *Artifacts* mentioned in a procedure can be *explicitlyRequired* or not. For example, in "define the project plan" the artifact "project plan" is explicitly required whereas "eliminate the faults in the software" may lead to model the artifact "software without faults" which is not explicitly required. We also model the multiplicity of an artifact or role (*isPlural*) to allow a precise comparison. On the other hand, *Activities* may be performedIn different *Contexts* (e.g. "approve the plan before project initiation"). As context information is conceptual information as well, *Contexts* are special *ConceptualElements* and as they are elements of procedures they are also special *ProcedureElements*. A *Context* usually explains its *Activity* (e.g. "maintain the programme by controlling the projects"), but it may also specify a temporal relation (e.g. "approve the plan before project initiation") or specify a local relation (e.g. "review requirements specification in the IT department"). The different context types are modeled by an attribute of type *ContextType*.

Figure 3 depicts the elements of MoSaIC's ***Integration Concept Meta-Model***. Although it has a pretty simple structure it is sufficient to model the world of IRM concepts with their relations. Obviously, a *Concept* (which is a term or a combination of terms from a IRM) is the main element. A *Concept* always has a *ConceptType* and may be related to other *Concepts* by so called *ConceptRelations* which are typed as well. The *ConceptType* determines the role of a *Concept* in a certain context (e.g. "work breakdown structure" may be an *Artifact* but also a *Method* depending on the context). *ConceptRelations* are used to model similarities between concepts. A *Concept* may be *composedOf* other *Concepts*. For example "requirements" is *composedOf* "functional requirements" and "non-functional requirements". Furthermore, a *Concept* may be a *generalizationOf* a more concrete *Concept* (e.g. the concept "stakeholder" is more general than "project manager"). In addition, a *Concept* may be *definedBy* other *Concepts*. For example, the concept "plan the involvement of stakeholder" is defined based on the concept "stakeholder". If a *Concept* is self-contained, it is called atomic (attribute *isAtomic*). Atomic concepts are usually defined by experts. Currently, initial sets of concept and concept relation types are offered. This architecture is flexible and open to introduce new concept and concept relation types if needed.

In the next section we describe the application of the both meta-models to ease their understanding and to show their modeling abilities.

3. Application of the Meta-Models

In order to evaluate the appropriateness of the developed meta-models we have performed a mid-size case study considering some parts of COBIT, CMMI and FS.

Although each IRM focuses on a specific IT domain they are similar in certain aspects. Be-

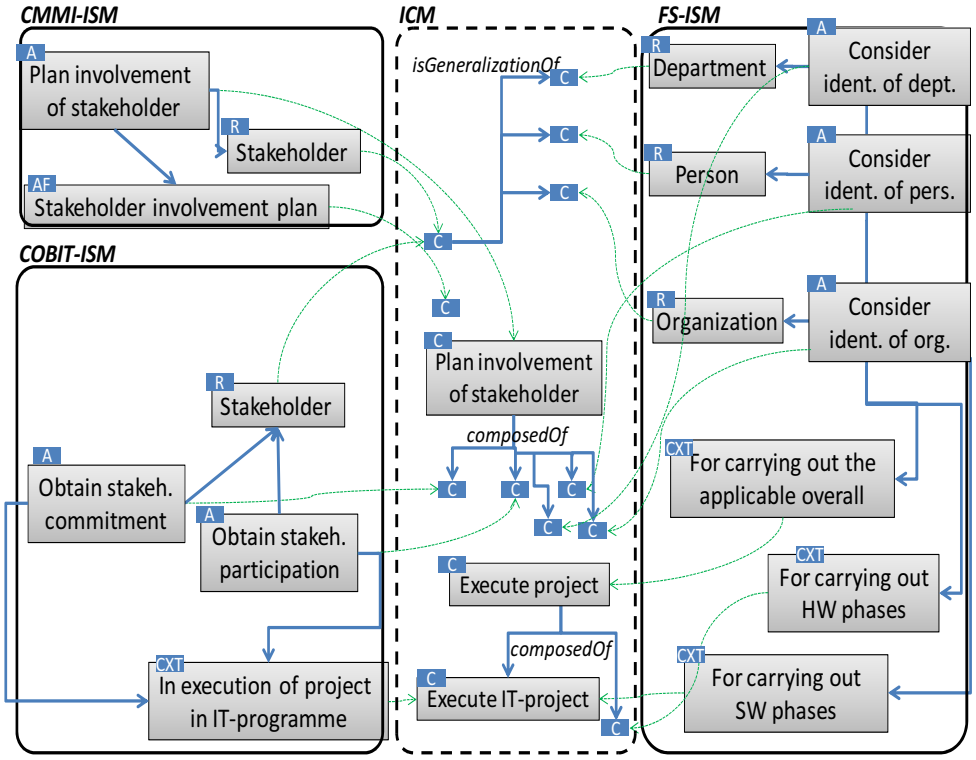

Figure 4. Excerpt of a MoSaIC IRM Integration Model

cause the aim of our case study was to explicitly show the similarities between the considered IRMs and therewith their integration, we selected designated procedures that contain similar conceptual elements: we consider procedures the following IRMs-elements: CMMI practices, the sentences of COBIT control objectives and Functional Safety requirements (CMMI PP SP2.6: "Plan the involvement of identified stakeholder" with the typical work product "stakeholder involvement plan"; COBIT PO10.4: "Obtain commitment and participation from the affected stakeholders in the (..) execution of the project within the context of the overall IT-enabled investment programme"; Functional Safety, Part 1, 6.2.1b: "Consider the identification of the persons, departments and organizations which are responsible for carrying out (..) the applicable overall, E/E/PES or software safety lifecycle phases").

Figure 4 depicts the developed models. Each ISM contains conceptual elements, such as activities (A), artifacts (AF), or roles (R). The COBIT- and FS-activities are described by additional context information, contexts (CXT) are added and linked to the respective activities. The associations of ISM conceptual elements with their correspondent concepts (C) in the ICM integrate the procedures of the different IRMs. For example, the CMMI-ISM activity "Plan involvement of stakeholder" is composed of the COBIT- and FS- activities. Another example of similarity between the modeled procedures is given by the role "Stakeholder". While CMMI and COBIT use this term, it is represented in FS by the three roles "Departments", "Persons" and "Organizations". Therefore, we can easily identify the similarities between these procedures. Furthermore, we can easily extract the essence by identifying only the general concepts: perform the activity "plan the involvement of the stakeholder" for the "execution of the project", involve the "stakeholder" and produce the output "stakeholder involvement plan". The ICM allows identifying the relations of these general concepts to the more concrete concepts in the ISM of the corresponding IRMs: the organization can easily identify the details (e.g. in FS the stakeholder are represented by persons, departments, organizations). Therefore, the traceability between the abstract practice

and the more detailed instances in the IRMs is supported.

The dependencies between the procedures of a IRM or different IRMs can also be identified. The procedures that share similar artifacts are interdependent: one procedure uses the output of another procedure as input. For example the CMMI procedure PP 2.7 “Establish and maintain the overall project plan” produces the output “project plan” that is used as input in the COBIT procedure PO10.7 ”The project plan (. . .) should be approved in line with the programme and project governance framework”.

4. Experiences and Future Work

In the following we describe the experiences gained with the developed IRM integration approach. Furthermore, we propose some ideas towards further research and ideas concerning the application of the MoSaIC IRM integration approach.

4.1. First Experiences

The modeling of selected parts of CMMI, COBIT and Functional Safety showed that the IS and IC meta-models offer a stable structure for integrating the chosen IRMs. However, modifications of the meta-models may be possible after gaining more experience in modeling further IRMs.

The IRM specific ISMs as well as the central ICM were created manually. Because our approach is based on fine grained IRM information, a large number of conceptual elements had to be modeled. Thereby, redundant definitions of semantically equivalent concepts in the ICM had to be avoided. Furthermore, we always tried to relate new concepts to existing ones in case of similarity (by “composedBy”, “generalizationOf” or “definedBy” relations). However, the manual modeling was not always easy and sometimes it was difficult to model concepts consistently.

Based on our experience we evaluate our IRM integration approach in relation to the goals listed in section 1.1 as follows.

The created ISMs and the ICM allow a semantic integration of IRMs. Because each IRM-ISM contains the most important IRM information it provides a condensed overview for organizations (G1). Furthermore, the ICM eases the understanding and avoids misinterpretations of terms and concepts used in IRMs, because the concepts are associated with their synonyms and contexts in the IRM-ISMs. Every new IRM may enrich the description of a concept and ease its comprehension. Therefore, the more IRMs are integrated, the smarter the ICM will become. This approach is a kind of crowdsourcing where organizations work together to improve the ICM by model-ing more and more IRMs. Furthermore, the created ICM connects the different terms used in the IRMs and connects the procedures of different IRMs. Therefore, the organizations can easily identify the dependencies between these different IRMs (G4).

Our approach supports a detailed comparison of IRM procedures. The fine grained model elements, the concepts connecting the procedures, and the semantic concept relations allow identifying similar procedures and their essence (G2). The extracted abstract practices help organizations to avoid redundancies and to reduce the effort in the adoption and assessment of multiple IRMs. The implementation of the abstract practices of certain IRMs assures the adoption of these IRMs. If necessary, the organizations can easily identify the details in the IRMs with the help of the traces (G3). These traces are useful also in the assessments. By considering the abstract practices of certain IRMs, it can be roughly assessed if these are implemented in the organizations. Finally, the details of the IRMs can be traced and be assessed.

The fine granularity of the models (ISMs and ICM) enables to model IRMs on different levels of abstraction: abstract, concrete IRMs and even internal processes can be easily integrated in the MoSaIC Integration Model (G5). This integration supports a better understanding of a certain area that is addressed by both of them (G1). Finally, changes on existing IRMs or new IRMs can be integrated in the MoSaIC Integration Model (G6). This is done by creating or updat-

ing the respective ISM and adding or updating the connections between the changed or newly created ISM and the central mediator, the ICM. The integration of a new IRM implies adding new concepts to the ICM in case they are not already defined.

4.2. Future Work

The proposed approach realizes a sound basis to integrate IRMs. However, more research has to be done to fully achieve all defined goals and to facilitate the application of MoSaIC's IRM integration approach.

Because the manual modeling of IRMs is a time consuming and error-prone process, a dedicated tool box is needed. Currently we have developed only sparse tool support based on the implementation of the meta-models as Ecore models in the Eclipse Modeling Framework [16]. By means of the respective generated tree-based editors we are able to manually create and maintain the ISMs and the ICM. A more sophisticated tool box should support the IRM expert e.g. to cope with the consistency problem and to model the fine grained conceptual elements. Furthermore, a semiau-tomatic tool performing a syntactical and linguistic analysis of the IRM documents may generate recommendations to model the conceptual elements properly. For example, prepositions like "based on" or "in line with" may require modeling a respective artifact. Because the IRM documents are written very differently, modeling recommendations can not only rely on syntactic rules. For example, in some IRM documents the activities of procedures are written by nouns while in others verbs are used. Hence, a plain syntactical analysis of the documents is not sufficient. Further rules are needed to transform the language used in the original documents in a "normalized" language. We will investigate if rules used to precisely write requirements ([17]) could be adopted to transform the original text in a "normalized" language (e.g. the passive or noun form of a verb is transformed in its active form). This may allow a semiautomatic extraction of conceptual elements and their modeling in the Integration Structure and Concept Model. Furthermore, the tool box should be able to adapt and improve its generated recommendations according to modeling decisions done by IRM experts.

Currently we are developing a new approach to compare IRMs based on the information stored in our models. At the moment, the identification of similarities and the comparison has to be done manually be an expert. Our new approach should enable a tool supported automatic comparison of IRMs and the determination of coverage degrees of procedures, respectively of considered IRMs.

5. Conclusions

In this paper we have presented MoSaIC, a model based approach to integrate reference models. The core idea is to represent each IRM in a dedicated Integration Structure Model (ISM) and the common concepts in one central Integration Concept Model (ICM).

The IRMs' integration achieves transparency and reduces complexity of IRMs. As the IRM-ISMs are instances of their common meta-model they normalize the different structures of the IRMs and provide a rough overview. Due to the modeled fine granularity, the common structure enables to semantically integrate different IRMs. The ICM normalizes the different concept terminology of the IRMs and forms a landscape of IT terms and their relations. Through common concepts and their semantic relations, this model facilitates the understanding of IRMs. A better understanding is supported also by the integration of general and concrete IRMs. Furthermore, it supports a detailed comparison of IRMs that allows to identify similar and interdependent procedures and helps organizations to avoid redundancies in the adoption and assessment of multiple IRM. Therefore, they will be able to efficiently and asses adopt multiple IRMs.

First experience and results with the presented approach are promising; we were able to effectively model the integration of some process areas of three IRMs. We expect that the results of our future work will make the integration of IRMs more accurate and comfortable.

References

[1] "CHAOS summary 2009," The Standish Group, Boston, Tech. Rep., 2009. [Online]. http://www.statelibrary.state.pa.us/portal/server.pt/document/690719/chaos_summary_2009_pdf

[2] "CMMI for development, version 1.3," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI 2010-TR-033, ESC-TR-2010-033, 2010. [Online]. http://www.sei.cmu.edu/reports/10tr033.pdf

[3] *ISO/IEC 15504 Information Technology Process assessment. Part 1: Concepts and vocabulary, Part 2: Performing an assessment, Part 3: Guidance on performing an assessment, Part 4: Guidance on use for process improvement and process capability determination, Part 5: An exemplar Process Assessment Model*, ISO/IEC Std., 2007.

[4] "Control objectives for information and related technology version 4.1," ISACA, Tech. Rep., 2007. [Online]. https://www.isaca.org/bookstore/Pages/Product-Detail.aspx?Product_code=CB4.1

[5] *IEC 61608 or Functional safety of electrical/electronic/programmable electronic safety-related systems*, IEC Std., Rev. 2, 30 April 2010.

[6] J. Siviy, P. Kirwan, L. Marino, and J. Morley, "Process architecture in a multimodel environment. white paper," Software Engineering Institute, Tech. Rep., 2008. [Online]. http://www.sei.cmu.edu/library/assets/multimodelSeries_wp4_processArch_052008_v1.pdf

[7] D. Malzahn, "Assessing – learning – improving, an integrated approach for self assessment and process improvement systems," in *ICONS 09, the Fourth International Conference on Systems*. Gosier, Guadeloupe, France: IEEE Computer Society Conference Publishing Services, 2009, pp. 126–130.

[8] Y. Wang, G. King, A. Dorling, and H. Wickberg, "A unified framework for the software engineering process system standards and models," in *4th IEEE International Software Engineering Standards Symposium and Forum*, Curitiba , Brazil, 1999.

[9] A. Ferreira, L. Machado, and M. C. Paulk, "Quantitative analysis of best procedures models in the software domain," in *17th Asia Pacific Software Engineering Conference*, 2010, pp. 433–442.

[10] Y. Wang, G. King, and H. Duncan, "Deriving personal software processes from current software engineering process models," in *European software process improvement: proceedings of the EuroSPI 2000 Conference*, Copenhagen, Denmark, 2000.

[11] O. Armbrust, A. Ocampo, J. Münch, M. Katahira, Y. Koishi, and Y. Miyamoto, "Establishing and maintaining traceability between large aerospace process standards," in *5th International ICSE Workshop on Traceability in Emerging Forms of Software Engineering*, 18 May 2009, pp. 36–40.

[12] A. Ferchichi, M. Bigand, and H. Lefèbvre, "An ontology for quality standards integration in software collaborative projects," in *Model Driven Interoperability for Sustainable Information Systems*, J.-P. Bourey, Ed., Montpellier, FRANCE, 2008, pp. 17–30. [Online]. http://ceur-ws.org/Vol-340/paper02.pdf

[13] L. Liao, Y. Qu, and H. Leung, "A software process ontology and its application," in *International Workshop on Future Software Technology (IWFST-2005)*, Shanghai, 2005.

[14] M. Soto and J. Muench, "Using model comparison to maintain model to standard compliance," in *ICSE Workshop Comparison and Versioning of Software Models (CVSM 2008)*, Leipzig, Germany, 17 May 2008.

[15] J.-M. Favre, "Megamodelling and etymology," in *Transformation Techniques in Software Engineering*, ser. Dagstuhl Seminar Proceedings, J. R. Cordy, R. Lämmel, and A. Winter, Eds., No. 05161. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. [Online]. http://drops.dagstuhl.de/opus/volltexte/2006/427

[16] D. Steinberg, *EMF : Eclipse Modeling Framework*. Upper Saddle River, NJ: Addison-Wesley, 2009.

[17] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals A study Guide for the Certified Professional Requirements Engineering Exam Foundation Level IREB compliant*. Rocky Nook, Inc, Santa Barbara, 2011.