The Knowledge Engineering Review, Vol. 24:3, 225–244. © 2009, Cambridge University Press
doi:10.1017/S0269888909990038 Printed in the United Kingdom

Sharing architecture knowledge through models: quality and cost

PENG LIANG, ANTON JANSEN and PARIS AVGERIOU

Department of Mathematics and Computing Science, University of Groningen, 9700 AK, Groningen, The Netherlands; e-mail: liangp@cs.rug.nl, anton@cs.rug.nl, paris@cs.rug.nl

Abstract

In the field of software architecture, there has been a paradigm shift from describing structural information, such as components and connectors, to documenting architectural knowledge (AK), such as design decisions and rationale. To this end, a series of industrial and academic domain models have been proposed for defining the concepts and their relationships in the field of AK. To a large extent the merit of this new paradigm is to share and reuse AK across organizations, especially in geographically distributed settings. However, the employment of different AK domain models by different parties makes effective AK sharing challenging, as it needs to be mapped from one domain model to another. In this paper, we investigate two different approaches for sharing AK, based on either direct or indirect mapping between different AK domain models. We compare the cost and quality of these two approaches, with respect to the processing of large amounts of AK instances. To predict the quality and costs of this processing in advance, a prediction model is proposed and validated with a concrete AK sharing case. Based on the comparison results, stakeholders involved with AK sharing can select an appropriate approach by trading off quality and cost in their own context.

1 Introduction

Software architecture is considered of paramount importance to the software development life cycle (Bass *et al*., 2003). It is a key artifact for the early analysis of the system, as it facilitates stakeholders' communication and understanding, and drives both system construction and evolution. Current research trends in software architecture focus on the treatment of architectural decisions (Kruchten, 2004; Kruchten *et al*., 2005; Tyree & Akerman, 2005) as first-class entities (Bosch, 2004) with explicit representation in the architectural documentation (Jansen & Bosch, 2005). Kruchten *et al*. (2005) define architectural knowledge (AK) as: **AK = design decisions + design**, in which four types of design decisions are classified. Generally speaking, AK encompasses not only decisions and rationale, but also other architecturally significant information. For example, the AK core model proposed by de Boer *et al*. (2007) suggests that AK is a set of relationships between decisions, stakeholders, architectural design, and processes. AK may contain alternative solutions, significant entities from the problem space (e.g. key stakeholder concerns), technology constraints, business information, general knowledge (e.g. design patterns) etc. (Avgeriou *et al*., 2007).

The main value of a software company is its intellectual capital and AK is deemed a valuable part of this (Kruchten *et al*., 2005). It is in the interest of companies to share AK, for example, by recording the knowledge from the architects' minds on paper, or further formalizing it in a knowledge repository, so that individual expertise and collective decisions on AK can be shared among stakeholders. Although sharing and reusing of AK is far from common practice at present

(Tang *et al*., 2006), it is considered as a dominant factor in a software architecting process and for eventual project success (Jansen & Bosch, 2005). AK sharing may serve significant architecting activities like modifying past design decisions, or performing architecture reviews and trading off quality attribute requirements. It may also play a simpler role, for example, in a cross-team software development project, all the stakeholders must have access to information on who is responsible for which part of the whole system.

Sharing AK between different organizations, or even between the departments of a single organization, poses a great challenge: the domain models of AK are not standardized. On the contrary, they tend to vary enormously. In fact, various authors (Kruchten, 2004; Tyree & Akerman, 2005; Ali-Babar *et al*., 2006; Capilla *et al*., 2006; Tang *et al*., 2007; Jansen *et al*., 2008) have proposed their own AK domain models to document AK concepts and their relationships. Some of these concepts and relationships are different, while others are largely overlapping. These discrepancies between the AK domain models can hamper the effective sharing of AK, which in turn results in misunderstandings among stakeholders, expensive system evolution, and limited reusability of architectural artifacts (Jansen *et al*., 2007). This problem is of course not specific in the field of AK but it is very common in other fields, such as knowledge sharing in gene data (Camon *et al*., 2004) and geographic information systems (Fonseca *et al*., 2000), etc.

In this paper, we look at the problem of AK sharing through a knowledge grid (Zhuge, 2004; Jansen *et al*., 2007). In this envisioned AK grid, AK is captured in domain-specific models. The use of such models allows different organizations, departments, or even persons to express their AK using their own concepts in the AK grid. Using the mappings between these models, all AK is transparently shared among the interested stakeholders as the AK is expressed for each person in terms of his or her own domain model(s). Two different approaches for implementing this AK sharing strategy are investigated in this paper. The first approach is a direct mapping approach, in which all models are directly mapped onto each other. The second approach is an indirect mapping approach, in which all the models map onto one central model. This central model acts a mediator between the different domain models. The topic of this paper is to find out the quality of both AK sharing approaches and the cost associated with them.

Both the cost and the quality of AK sharing are not only dependent on the models and mappings involved, but also on the actual AK instances of these models. Only with these instances the real cost and quality of AK sharing can be determined. However, creating these instances entails considerable effort as human intervention is required. To make matters worse, most of this effort needs to continuously evolve when, due to further insight, domain models or mappings are changed. Hence, we would like to predict the cost and quality of AK sharing in advance before effort is spent on creating instances. This paper contributes such a prediction model for both the direct and indirect mapping approaches.

The rest of this paper is organized as follows. In Section 2, we present related work about software architecture, AK, and knowledge sharing methods. The problem statement of evaluating AK sharing approaches with respect to cost and quality is refined in Section 3. In Section 4, a cost model for AK sharing is presented. Section 5 proposes three models for predicting the AK sharing quality. A concrete AK instance mapping experiment is presented in Section 6 to validate these prediction models. A comparison is made on the difference in quality and cost between the two approaches for sharing AK in Section 7. We wrap up with conclusions and future work in Section 8.

2 Related work

2.1 Software architecture

In the early 1990s, Perry and Wolf (1992) formed the starting point for an evolving community that actively studied the notion and practical application of software architecture. In the years to follow, software architecture has been broadly adopted in the industry as well as in the software engineering research community. A generally accepted definition of the notion of software

architecture is captured in the IEEE 1471-2000 standard (Institute of Electrical and Electronics Engineers, 2000): 'software architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution'. This definition places component and their connectors as the central concepts of software architecture. Architectural Description Languages (ADLs) (Medvidovic & Taylor, 2000) use these concepts as first-class entities to formally describe software architectures.

2.2 Architectural knowledge and architectural knowledge sharing

The traditional approach of components and connectors has failed to address one of the major shortcomings in the field of software architecture: *Architectural Knowledge Vaporization* (Bosch, 2004; van der Ven *et al*., 2006), that is, the architecturally significant information being lost during system evolution. To this end, Jansen and Bosch (2005) suggested to treat software architecture as a result of a set of architectural design decisions, which is one of the most significant forms of AK (Kruchten *et al*., 2005; van der Ven *et al*., 2006). The SHARK (SHAring and Reusing architectural Knowledge) series of workshops (Lago & Avgeriou, 2006; Avgeriou *et al*., 2007) demonstrate the academic and industrial interests and advancement in AK sharing and reusing.

Various specific AK domain models have been proposed by industrial organizations (see the LOFAR AK model (Jansen *et al*., 2008) from the LOFAR[1] project) and domain experts (see Kruchten's ontology (Kruchten, 2004), Tyree's template (Tyree & Akerman, 2005), Architecture Rationale and Elements Linkage (Tang *et al*., 2007), Process-centric Architecture Knowledge Management Environment (Ali-Babar *et al*., 2006), Architecture Design Decision Support System (Capilla *et al*., 2006) and Archium (Jansen *et al*., 2007) AK models). There are also generic AK models, such as the IEEE 1471-2000 standard (IEEE, 2000), which define a domain model for architecture concepts that can be used for AK management. All these AK models are composed of a set of concepts and relationships between the concepts, and therefore ontologies can be used to model these AK models (Akerman & Tyree, 2005).

Preliminary work on investigating AK sharing in a knowledge grid has been performed by de Boer *et al*. (2007). They investigate the issues of AK sharing using an indirect mapping approach. To this end they construct a core model that covers most of the concepts from different AK domain models. The core model acts as a mediator for mapping between different AK models. Their work mainly focuses on a model and a conceptual level sharing solution that uses concept (i.e. model element) mappings. However, the issue of evaluating this core model with respect to AK sharing quality and cost has not been addressed. We extend this work, by looking in detail, into the quality and cost that such an approach brings.

2.3 Knowledge sharing approaches and tools

Wikis are an emerging Web-based tool for cooperative knowledge management and sharing. The most famous and successful example of a wiki is Wikipedia. Wikis are very popular in cases that demand effective collaboration and knowledge sharing at low costs. Some concrete research effort has recently been made on using wiki for documentation of software requirements (Silveira *et al*., 2005), software architecture (Bachmann & Merson, 2005), software artifacts (Aguiar & David, 2005), and software development processes (Louridas, 2006). The limitation of a wiki is that it shares knowledge based on common topics (e.g. tags). A wiki therefore suffers from the same problem as AK sharing through models, as it requires stakeholders to share a common understanding and adopt common terminology. In addition, a wiki allows users to make different explanations for one topic, which may confuse the stakeholders who want to share the knowledge.

Ontology is a key technology for knowledge representation, management, and sharing, and has been widely used in some emerging fields, such as the semantic Web. In the knowledge engineering

[1] LOFAR is the abbreviation of Low Frequency Array project undertaken by Astron, the Dutch Astronomy Institute, which is involved in the development of large software-intensive systems used for astronomy research.

domain, an ontology can be regarded as a conceptual model. This is confirmed by the commonly accepted definition in Gruber (1993): 'An ontology is a specification of a conceptualization'. In the emergent semantic Web, search, interpretation, and aggregation can be addressed by ontology-based semantic annotation (Uren *et al.*, 2006), which can also be applied in document annotation for structuring, interpreting, and sharing documents. This approach has standard annotation formats (e.g. RDF, Resource Description Framework; OWL, Web Ontology Language) as a prerequisite for sharing annotations, thus facilitating knowledge sharing and collaboration. The limitation of this approach is that all stakeholders involved with the knowledge sharing should agree on the ontology being used for the semantic annotation. This is desirable but highly unlikely to occur in practice.

Due to the existence of multiple ontologies in one application domain, there needs to be a reconciliation of heterogeneous ontologies for effective knowledge sharing. Ontology mapping is an activity to relate semantically two ontologies, for example, O_1 and O_2. Mapping one ontology onto another means that for each concept C in ontology O_1, we try to find a corresponding concept, which has the same intended meaning in ontology O_2 (Ehrig & Staab, 2004). Many research efforts on ontology mapping methods (automatic or semi-automatic) have been taken to achieve good mapping result with less cost (Kalfoglou & Schorlemmer, 2003; Choi *et al.*, 2006). However, ontology mapping still remains an expensive, time-consuming, and error-prone activity.

2.4 Ontology mapping quality and cost evaluation

As mentioned in Section 2.3, ontology mapping is a promising approach for information exchange in a semantically sound manner (Kalfoglou & Schorlemmer, 2003), in which the meaning of information is taken into account. Ehrig and Euzenat (2005) have performed research on the evaluation of ontology mapping by introducing two criteria: the precision and recall rate. These metrics originate from information retrieval (IR) theory, and are revised as relaxed precision and recall rate, which is more appropriate for ontology mapping evaluation. The criteria are relaxed in the sense that they are not as strict as those for IR. The reason is that a mapping between similar ontologies is better in relevant data retrieval than a mapping between totally different ontologies, but these two cases are both regarded as the non-relevant data in IR. Euzenat (2007) extends this work by proposing semantic measures in order to get maximal precision and recall for correct ontology mappings. However, this work is limited in evaluating the quality of the mappings between ontologies in the conceptual level. For the ontology mapping cost, Ehrig and Staab (2004) proposed a cost-effective way QOM (Quick Ontology Mapping) for ontology mapping by introducing a restricted range of costly features, but they provide no quantified cost analysis for this mapping method. To the best of our knowledge, there is no research work on the quality and cost of instance mapping for knowledge sharing.

3 Analysis

3.1 Introduction

As mentioned in Section 1, the research focus of this paper is on the prediction of AK sharing quality and cost. For AK sharing, two approaches can be employed: direct and indirect mapping. Section 3.2 presents how AK can be shared using concept mappings. Section 3.3 outlines how the cost of AK sharing is evaluated, whereas Section 3.4 does this for quality.

3.2 Sharing architectural knowledge using concept mappings

For the purpose of clarity, we generalize AK sharing in two levels: the conceptual level and the instance level. At the conceptual level, an AK *model* defines the *concepts* and *relationships* that a particular organization, department, project, or person uses. At the instance level, the *instances* of the aforementioned concepts and relationships exist. The sharing of AK instances based on different AK models depends on the mutual understanding of the underlying AK models, that is,

one concept in one AK model can be translated or *mapped* into a concept in the other model. Thus, this mutual understanding can be specified by a set of mapping relationships between concepts from different AK models.

For example, domain experts can define a concept A in one AK model to have the same meaning as a concept B in another AK model. Once the equality mapping relationship is defined between concepts A and B, the instances (e.g. pieces of text) of concept A are considered to be instances of concept B as well.

To achieve AK sharing, we use the mappings defined between concepts of different AK models to translate the instances of one model into another. For the required model mappings, two approaches can be employed: a direct or an indirect mapping approach. With the direct mapping approach, one defines mapping relationships from a source AK model to a target AK model directly. The source AK model is the model in which the instances to be translated are defined. The target AK model is the language in which one would like to use the AK. An indirect mapping approach defines mapping relationships from a source AK model to a target AK model through a central model, which acts as a mediator.

Both methods have their pros and cons. In most cases, a direct mapping approach achieves a better AK sharing quality than the indirect one, as every translation is bound to lose some information. On the other hand, a direct mapping approach requires more effort to realize the mappings and is harder to maintain than an indirect mapping approach: with the direct approach, the number of mapping relationships increases exponentially with the number of AK models involved, whereas with the indirect approach the number of mapping relationships increases linearly. In other words, there is a tradeoff between the cost and the quality of the AK sharing when using these two approaches.

3.3 Evaluating the cost of sharing architectural knowledge using model mappings

The cost of sharing AK using model mappings has many aspects. In this paper, we concentrate on those costs that come from human effort, thereby leaving out hardware and software costs. The cost for sharing AK using model mappings consists of the following different aspects:

- **Modeling costs.** Effort is needed to find the concepts and relationships that are relevant to AK and express them in AK models.
- **Capturing costs.** The aforementioned AK models need to be filled in with relevant data (AK instances), which is the most costly operation of all.
- **Mapping costs.** To share AK, mapping definitions among the AK models are needed.
- **Evolution costs.** Further insights into a domain can cause changes to the used AK models.
- **Extension costs.** An organization, department, or person might want to join the knowledge grid and extend it with their own AK model.

The challenge is to find ways to quantify these aspects of the cost. Such quantifications do not have to provide absolute numbers, as relative numbers or even the order magnitude of these numbers is useful enough in making coarse-grained comparisons between different approaches.

3.4 Evaluating the quality of sharing architectural knowledge using model mappings

3.4.1 Query-based architectural knowledge-sharing scenario

We envision AK sharing in an AK grid, that is, a heterogeneous AK repository that is comprised of different local repositories. Each local repository contains one AK model and its instances. A user can retrieve AK from all involved AK repositories transparently without being conscious of the underlying model differences. To quantify the quality, we use a specific user scenario in the form of a query, which is a typical activity for knowledge sharing. The query is a precise request for IR, typically expressed as keywords combined with Boolean operators and other modifiers.

The query-based scenario is shown in Figure 1. A user who understands only AK model T queries the repository of AK model S using concepts from AK model T as query keywords.

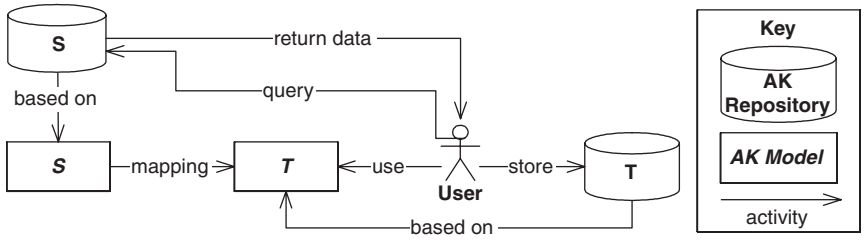

Figure 1 Query-based scenario for architectural knowledge sharing

The conceptual difference between AK models S and T poses a problem for AK sharing. The concepts from model T queried do not exist (or exist, but have a different meaning) in model S. Thus, the repository of model S cannot return any data (AK instances). Using concept mappings from model S to T, the repository of model S could return partial data to the user.

3.4.2 Model mappings

To construct AK models, we use ontologies (see Section 2.3). In ontologies, concepts are defined as classes. The mapping relationships between the concepts of the AK models are therefore defined as relationships between classes in an ontology. We use the following mapping relationships to relate AK models with each other:

- `subClassOf`, denotes one concept to be a specialization of another.
- `superClassOf`, denotes one concept to be a generalization of another.
- `equivalentClass`, denotes two concepts to be the same.
- `disjointWith`, denotes that the instances of two concepts can never belong to both concepts.
- `noMatchingPair`, denotes that a concept cannot be mapped to another AK model.

3.4.3 Problematic mapping scenarios

Both mapping approaches suffer from two problematic mapping scenarios that result in **lost data** and **garbage data**. Hence, these scenarios influence the mapping quality in a negative way. **Lost data** are created when the instances cannot be classified to the target model, as either the necessary concepts are missing in the target model or the defined concept mapping is incomplete. Consequently, queries are able to return less relevant data; hence data is lost. **Garbage data** are the instances that are wrongfully classified, thereby contaminating the relevant data in the query results. The instances can be wrongfully classified due to the following reasons:

- **Instance classification problems,** which sometimes occur with the `superClassOf` mapping relationship. If a concept is mapped as a `superClassOf` a concept in a target model, some of the instances of this concept might be an instance of this target concept, whereas others are not. Hence, there is a classification problem for instances of a concept mapped as a generalization of another concept.
- **Faulty mappings,** which are due to human errors in defining the mappings. Often this is caused by an expert not understanding the involved AK models correctly. The indirect mapping approach is more vulnerable to this problem than the direct mapping approach, as it uses two mappings instead of one.

Both **garbage data** and **lost data** are best illustrated using an indirect mapping approach, although they also occur in the direct mapping approach. Figure 2 presents an example of this. In the figure, the three circles represent three different AK models (i.e. S, T, and central model C). Inside the circles reside x_S, y_S, x_C, y_C and x_T, y_T, which are concepts from models S, C, and T, respectively. The mappings are indicated with arrows: the dotted arrow lines correspond to an indirect mapping, while the others to a direct mapping.

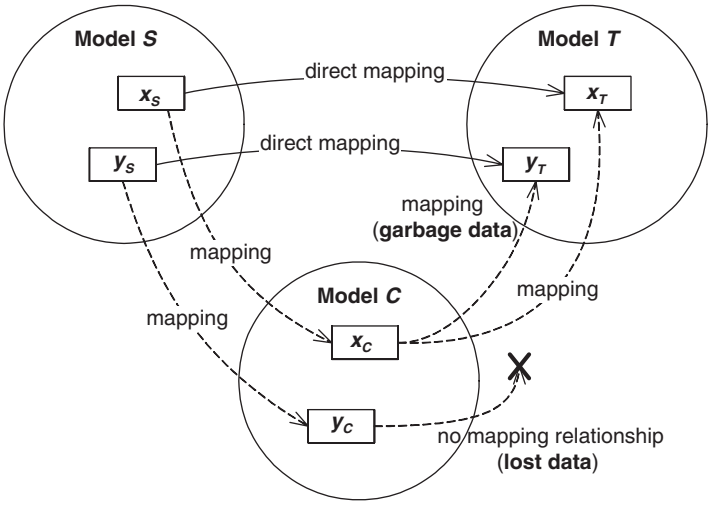

Figure 2 Scenarios of lost data and garbage data with indirect mapping approach

In the figure, **lost data** are created due the inability to map concept y_C to a concept of T. In this case, we know through the direct mapping that y_C should have been mapped to concept y_T. However, we only know this holds for instances of y_S and this might not be the case for y_C. Thus, the amount of relevant data of model S that can be retrieved is reduced. **Garbage data** are created in this example by the 'faulty' mapping from x_C to y_T. Although this mapping makes perfect sense in the context of models C and T, we know from the direct mapping between models S and T that the instances of x_S have nothing to do with the concept y_T. Hence, garbage data are created, which contaminate the relevant data in the query result.

3.4.4 Definition of architectural knowledge sharing quality

The query-based scenario in Section 3.4.1 is a typical activity in the field of IR. To define the quality of such queries, IR theory defines the concepts of **recall** and **precision**. These concepts could, therefore, also be used to quantify the AK sharing quality. The **recall** rate is the proportion of *relevant data* that is *retrieved*. The **precision** rate is the proportion of *retrieved data* that is *relevant* (Cleverdon, 1967). For the query-based scenario, the recall rate can be calculated by looking at the number of correctly classified instances to model T compared to the total number of the instances in model S, that is, **recall** = $|$*correctly classified instances of* $T|/|S|$, in which the notation $|Set|$ denotes the number of instances in a *Set*. The precision rate can be calculated by dividing the number of correctly classified instances to model T to the total number of classified instances to model T, that is, **precision** = $|$*correctly mapped instances of* $T|/|T|$.

The goal of a good IR system (and therefore also the AK grid) is to retrieve as much relevant data as possible (i.e. have a high **recall**), and very few non-relevant data (i.e. have a high **precision**). Unfortunately, these two goals have proven to be quite contradictory. Techniques that tend to improve recall tend to hurt precision and vice versa.

3.4.5 Prediction of precision and recall

A problem in using the recall and precision rate as quantifications of the AK sharing quality is that it requires knowledge of the instances of the involved AK models. This is problematic, as capturing AK is the most expensive activity for sharing AK (see Section 3.3). Instead, we would like to predict the recall and precision rate before AK is being captured. Hence, such a prediction may only use limited information, such as concepts and concept mappings.

As we want to compare a direct mapping approach with an indirect one, as mentioned in Section 3.2, we would like to use a **relative** precision and recall definition. We can assume that a direct mapping gives the best mapping result. Therefore, we normalize the precision and recall rate

Table 1 Redefined terms for relative precision and recall calculation of an indirect mapping approach

Set	IR theory	Query-based scenario	Set symbol
Relevant data	The data that the user is querying for	The instances classified from S to T with a direct mapping	DM
Retrieved data	The data retrieved by an IR system to the user query using different retrieval methods	The instances classified from S to T with an indirect mapping	IM
Relevant retrieved data	The subset of the retrieved data that match with the user query	The instances that have been classified both by the direct and indirect mapping approaches	$DM \cap IM$

IR = information retrieval.

to the direct mapping. To do so, we redefine the concepts of *relevant data*, *retrieved data*, and *relevant retrieved data* introduced in IR theory, and represent them in terms of the query-based scenario. Table 1 presents an overview of these redefined terms.

Using the definitions in Table 1, the relative precision, P, and recall, R, can be calculated as follows:

$$P = \frac{|DM \cap IM|}{|IM|} \text{ and } R = \frac{|DM \cap IM|}{|DM|}. \tag{1}$$

When predicting the relative precision and recall, the problem is how many instances these three sets (i.e. DM, IM, and $DM \cap IM$) contain. We introduce the concept of set distribution, D, to address this problem. A **set distribution** is defined as the fraction of the number of instances a set contains to the number of instances of a fixed superset $|S|$ (S is the source model in the query-based scenario). Let D_{Set} be the set distribution of the Set. We do not know the exact value of set distribution defined in Table 1, as we do not have the instances. Therefore, we predict the set distribution. Let D'_{Set} be the prediction of a set distribution D_{Set}, then the prediction of precision, P', and recall, R', can be calculated as follows based on Equation (1):

$$D_{Set} = \frac{|Set|}{|S|},\ D'_{Set} = \frac{|Set|'}{|S|'},\ P' = \frac{D'_{DM \cap IM}}{D'_{IM}} \text{ and } R' = \frac{D'_{DM \cap IM}}{D'_{DM}}. \tag{2}$$

4 Cost prediction

The cost of AK sharing is composed of five aspects: modeling, capturing, mapping, evolution, and extension costs (see Section 3.3). Predicting the costs is relatively easy compared to quality prediction. To quantify these costs, we use the parameters defined as follows:

- n, the number of AK models involved with AK sharing in the knowledge grid.
- m, the average number of concepts per AK model.
- k, the average number of instances per AK model.
- c, the number of concepts in the central model for the indirect mapping approach. Note that $c \approx m$, as the central model will be similar in complexity to most AK models.

Using these parameters, we can predict the order of magnitude of the cost for AK sharing for the five different aspects:

- **Modeling costs.** The modeling costs will predominantly dependent on the number of models (n) and the number of concepts these models have (m). In addition, the familiarity with the domain being modeled has also an influence. Assuming this is constant for domain experts, modeling costs are $O(m \times n)$ for the direct mapping approach. For the indirect mapping approach also,

the central model needs modeling. Consequently, the costs become $O(m \times n + c) \approx O(m \times n + m) = O(m \times (n + 1)) = O(m \times n)$. Thus, there is no significant difference in modeling costs between both mapping approaches.

- **Capturing costs.** The capturing costs are the most dominant costs for sharing AK using model mappings, as the number of instances is typically much bigger than the number of models and concepts involved, that is, $k \gg m, n$. The order of magnitude of the capturing costs are directly related to the number of instances, which is the number of models times the average number of instances per model, that is, $O(n \times k)$. This holds for both mapping approaches.
- **Mapping costs.** The effort required for the mappings depends on the amount of concept mappings that should be considered. For the direct mapping approach, this is $O(n \times m \times (n - 1)) = O(n^2 \times m)$, as every concept of every model should be considered whether it maps to another concept of all other models. In the indirect mapping approach, every concept is mapped onto the central model and vice versa. No other models have to be considered in this mapping. The costs of this mapping is therefore $O(n \times (m + c)) \approx O(n \times m \times 2) = O(n \times m)$.
- **Evolution/Extension costs.** The evolution costs can be quantified considering a scenario in which one AK model is changed. The extension costs can be quantified in a scenario of extending the AK grid with a new AK model. In both the direct and the indirect approach, all the instances and mappings of the new or updated model need to be created or verified. The costs at the instance level have the same order of magnitude in both approaches, as the cost of the individual operation does not matter, only the number of operations counts. If this is the case, all the instances need to be verified and the created mapping to and from the model should be reconsidered. Thus, the evolution costs depend on the mapping approach taken. For the direct mapping, the evolution cost is $O(k + 2 \times (n - 1) \times m) = O(k + n \times m)$. Since every model in the grid maps directly back to the changed model, all these mappings need to be verified. For the indirect mapping approach, only the mappings to and from the central model need to be examined. The costs for an indirect mapping approach is $O(k + m \times 2) = O(k + m)$, assuming that the central model does not have to change. This is reasonable, as the other models already had a reasonable mapping to the central model. However, if the central model needs to be changed, the cost can be up to a maximum of $O(k + c \times n + m \times n) \approx O(k + 2 \times (m \times n)) = O(k + m \times n)$, which is equal to the costs of a direct mapping approach.

5 Instance mapping quality prediction

In this section, a closer look is taken at how the quality of AK sharing could be predicted. In Section 5.1, three different approaches are presented for this purpose. For one of these approaches, Sections 5.2 and 5.3 present the calculation methods and formulas.

5.1 Mapping quality prediction models (MQPM)

In Section 3.4.5, it was argued that the quality of AK sharing should be assessed by the precision and recall rate using predictions of set distributions (see Equation (2)). The value of these set distributions (i.e. D'_{DM}, D'_{IM}, and $D'_{DM \cap IM}$) depends on the instance classification results. This in turn is based on the concept mapping relationships between AK models. Consequently, the prediction of set distribution can be calculated using the set distribution prediction of all individual concept-mapping relationships.

The individual concept-mapping relationships are influenced by several aspects. To make fair predictions about them and their associated set distributions, a prediction model should either take them into account or make assumptions about them. In short, these aspects are the following:

- The relative importance of a concept in an AK model. For certain use-cases, some concepts are more important than others. Thus these concepts have a bigger impact on the perceived AK sharing quality.

- The instance distribution of each concept in the AK model, that is, some concepts have many more instances than other concepts. Hence, good mappings for the instances of these concepts greatly influences the overall sharing quality.
- The type of instance classification employed: manual or automated classification. In principle, a manual classification can always make correct classifications. For an automated classification, that is, a classification tool, the result doesn't have to be correct for every instance mapping. Consequently, instances are incorrectly classified and/or are lost during the mapping process.
- The quality of the automatic classification. If an automatic instance classification tool is employed, the quality of this tool directly affects the AK sharing quality: the better the tool is, more instances will be mapped and classified correctly.

The more of these aspects we take into account, the better the prediction model we can come up with. However, this comes at the cost of additional complexity. To deal with these aspects, we have defined three different prediction models each with their own set of assumptions. In order of complexity, these models are as follows:

Simple MQPM (SMQPM)

- All concepts are equally important;
- All instances in a AK repository are evenly distributed over the AK concepts;
- We use a perfect instance classification tool for the instance classification by which all instances will be smartly classified into correct concepts.

Random instance classification MQPM (RMQPM)

- All concepts are equally important;
- All instances in a AK repository are evenly distributed over the AK concepts;
- We use a random instance classification tool for the instance classification by which all instances will be classified into possible concepts randomly.

Advanced MQPM (AMQPM)

- Not all concepts are equally important;
- The domain expert predicts a more realistic instance distribution in a AK repository over the AK concepts;
- We use a random instance classification tool for the instance classification by which all instances will be classified into possible concepts randomly.

Of these three prediction models, the SMQPM has the most optimistic assumptions in which the instance classification tool classifies perfectly the instances into correct concepts. In this case the prediction of a theoretical maximum precision (P'_{SMQPM}) and recall (R'_{SMQPM}) can be achieved. The RMQPM has the most pessimistic assumptions in which the instance classification tool classifies the instances into possible concepts randomly. In that case the prediction of a theoretical minimum precision (P'_{RMQPM}) and recall (R'_{RMQPM}) can be achieved. The AMQPM has the most realistic assumptions compared with those in SMQPM and RMQPM. In that case the prediction of precision (P'_{AMQPM}) and recall (R'_{AMQPM}) is theoretically closer to the real case, and $P'_{RMQPM} \leqslant P'_{AMQPM} \leqslant P'_{SMQPM}$, $R'_{RMQPM} \leqslant R'_{AMQPM} \leqslant R'_{SMQPM}$.

In this paper, the focus is on the SMQPM. The other two prediction models, that is, RMQPM and AMQPM, will be investigated in the future.

5.2 SMQPM calculation rules

The calculation rules for SMQPM are based on the aforementioned assumptions and the effect these assumptions have on the set distributions. To be more precise, the predicted set

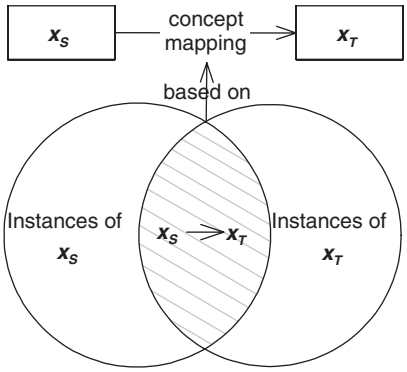

Figure 3 A concept mapping set distribution based on a concept mapping relationship from x_S to x_T

distributions (D'_{DM}, D'_{IM}, and $D'_{DM \cap IM}$) are calculated using the set distribution prediction of all the individual concept mapping relationships. To be concise and to differentiate from the **set distribution** concept defined in Section 3.4.5, the set distribution of individual concept mapping relationships is named the **concept mapping set distribution**. The calculation rules for the prediction of this **concept mapping set distribution** is based on different concept mapping relationships. The following symbols are used to describe the effect of these mapping relationships:

- x_S and x_T are concepts from two AK models S and T. A concept mapping relationship from x_S to x_T is normalized as a triple $\langle x_S, m, x_T \rangle$, in which m is the mapping relationship from x_S to x_T. The concept mapping relationships include `equivalentClass`, `subClassOf`, `superClassOf` (`inverseOf subClassOf`), `disjointWith` and `noMatchingPair`, which can be readily represented in RDF Schema (Brickley & Guha, 2004) or OWL (Dean *et al*., 2004). To keep things simples, other mapping relationships like `partOf`, `compositionOf` are not considered in this model.
- $NoC(x)$ denotes the Number of Concepts of x. For the triple $\langle x_S, m, x_T \rangle$,
  - if $m \neq$ `noMatchingPair`, then $NoC(x_S) = 1$ and $NoC(x_T) \geqslant 1$, which means that the concept mapping relationship maps a concept 1 to 1 or 1 to many;
  - if $m =$ `noMatchingPair`, then $NoC(x_S) = 1$ and $NoC(x_T) = 0$, which means that there is no mappable concept for x_S.
- $D'(\langle x_S, m, x_T \rangle)$ denotes the prediction of the **concept mapping set distribution** of an individual concept mapping relationship represented by the triple $\langle x_S, m, x_T \rangle$. Its real value, $D(\langle x_S, m, x_T \rangle)$, is the fraction of the number of instances classified from concept x_S to x_T to the number of instances of x_S, that is, $D(\langle x_S, m, x_T \rangle) = \frac{|x_S \rightarrow x_T|}{|x_S|}$. Figure 3 illustrates this with x_S, x_T, and $x_S \rightarrow x_T$ being sets of instances. With the assumption of an even distribution of instances over the concepts in SMQPM, we can assign a constant C as number of instances of all concepts, that is, $|x_S| = C$. Thus for the prediction of $D'(\langle x_S, m, x_T \rangle)$, we get:
  - $0 \leqslant D'(\langle x_S, m, x_T \rangle) \leqslant 1$;
  - $D'(\langle x_S, m, x_T \rangle) = 0$, if $m =$ `noMatchingPair`;
  - $D'(\langle x_S, m, x_T \rangle) = 1$, if $m \neq$ `noMatchingPair` and all the instances of concept x_S can be classified as the instances of concepts x_T.

In the remainder of this section, the calculation rules are presented to predict the **concept mapping set distribution** (i.e. $D'(\langle x_S, m, x_T \rangle)$) and the **side-effect concept mapping set distribution**. The latter distribution describes how the mapping indirectly affects the instance distribution of other concepts. For each different type of concept mapping relationship, that is, `equivalentClass`, `subClassof`, `superClassOf`, `noMatchingPair`, different calculation rules exist. We explain which rules should be used in which situation.

5.2.1 equivalentClass

Rule 1. `equivalentClass` **concept mapping relationship**

- **Concept mapping set distribution**
  - *Calculation*: $D'(\langle x_S, m, x_T \rangle) = 1$ ($m =$ `equivalentClass`)
  - *Reason*: Since x_S is `equivalentClass` of x_T, any instance of x_S is also the instance of x_T, that is, $|x_S \rightarrow x_T| = |x_S| = C$, then $D'(\langle x_S, m, x_T \rangle) = \frac{|x_S \rightarrow x_T|}{|x_S|} = 1$. An example of this is presented in Figure 4.
- **Side-effect concept mapping set distribution**
  - *Condition*: y_T is a concept in model T and is a direct `subClassOf` x_T. All the concepts y_T are `disjointWith` each other.
  - *Calculation*: $D'(\langle x_S, m, y_T \rangle) = \frac{1}{NoC(y_T)+1}$ ($m =$ `Rule 1`), in which $NoC(y_T)$ denotes the number of concepts as y_T.
  - *Reason*: With the assumption of even distribution of instances in SMQPM, all the instances of x_T will be distributed evenly in its direct subclasses (as y_T) plus one dummy subclass, which represents the concept of instances not covered by all the explicit direct subclasses of x_T. An example of this is presented in Figure 5. All the concepts y_T are `disjointWith` each other, so there are no instances in the intersection of different y_T. With $|x_S \rightarrow x_T| = |x_S| = C$, $|x_S \rightarrow y_T| = \frac{|x_S \rightarrow x_T|}{NoC(y_T)+1} = \frac{C}{NoC(y_T)+1}$, then $D'(\langle x_S, m, y_T \rangle) = \frac{|x_S \rightarrow y_T|}{|x_S|} = \frac{1}{NoC(y_T)+1}$.

5.2.2 subClassOf

Rule 2. `subClassOf` **with** `disjointWith` **concept mapping relationship**

- **Concept mapping set distribution**
  - *Calculation*: $D'(\langle x_S, m, x_T \rangle) = 1$ ($m =$ `subClassOf`)
  - *Reason*: Since x_S is `subClassOf` x_T, any instance of x_S is also an instance of x_T, that is, $|x_S \rightarrow x_T| = |x_S| = C$, then $D'(\langle x_S, m, x_T \rangle) = \frac{|x_S \rightarrow x_T|}{|x_S|} = 1$. An example of this is presented in Figure 6.
- **Side-effect concept mapping set distribution**
  - *Condition*: y_T is a concept in model T, and is a direct `subClassOf` x_T, and x_S is `disjointWith` y_T.
  - *Calculation*: $D'(\langle x_S, m, x_T \rangle) = 0$ ($m =$ `Rule 2`)
  - *Reason*: Since x_S is `disjointWith` y_T, there are no instances in the intersection between x_S and y_T. With $|x_S \rightarrow y_T| = 0$, then $D'(\langle x_S, m, y_T \rangle) = \frac{|x_S \rightarrow y_T|}{|x_S|} = 0$.

Rule 3. `subClassOf` **without** `disjointWith` **concept mapping relationship**

- **Concept mapping set distribution**
  - *Calculation*: $D'(\langle x_S, m, x_T \rangle) = 1$ ($m =$ `subClassOf`)

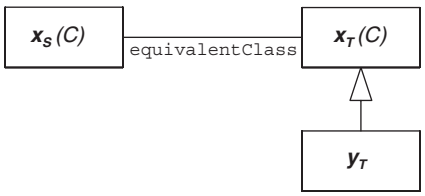

Figure 4 equivalentClass concept mapping relationship from x_S to x_T

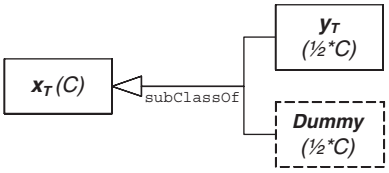

Figure 5 Instances classification example of internal subClassOf relationship with one subclass y_T

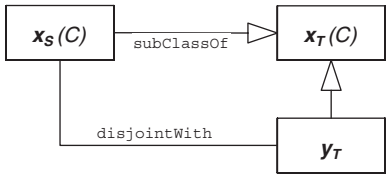

Figure 6 subClassOf concept mapping relationship from x_S to x_T with x_S disjointWith y_T

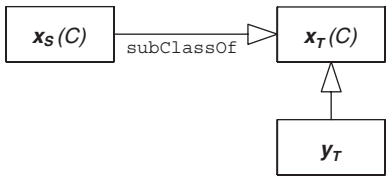

Figure 7 subClassOf concept mapping relationship from x_S to x_T without x_S disjointWith y_T

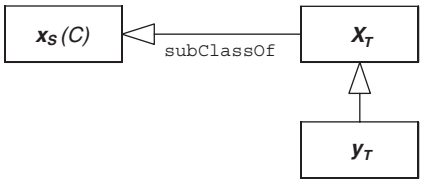

Figure 8 superClassOf concept mapping relationship from x_S to x_T

- *Reason*: Here the same reason applies as for the concept mapping set distribution in `Rule 2`. An example of this is presented in Figure 7.

- **Side-effect concept mapping set distribution**
  - *Condition*: y_T is a concept in model T and is a direct `subClassOf` x_T. x_S is not `disjointWith` y_T, which is the default situation as no mapping relationship is defined between them. All concepts y_T are `disjointWith` each other.
  - *Calculation*: $D'(\langle x_S, m, y_T\rangle) = \frac{1}{NoC(y_T)+1}$ ($m =$ `Rule 3`)
  - *Reason*: The same reason applies as for the side-effect concept mapping set distribution of `Rule 1`

5.2.3 superClassOf

Rule 4. `superClassOf` **concept mapping relationship**

- **Concept mapping set distribution**
  - *Calculation*: $D'(\langle x_S, m, x_T\rangle) = \frac{1}{NoC(x_T)+1}$ ($m =$ `superClassOf`)
  - *Reason*: Since x_S is `superClassOf` x_T, then x_T is `subClassOf` x_S. The same reason as that for side-effect concept mapping set distribution in `Rule 1` applies. An example of this is presented in Figure 8.

- **Side-effect concept mapping set distribution**
  - *Condition*: y_T is a concept in model T, and is a direct `subClassOf` x_T, and x_S is not `disjointWith` y_T, which is a default situation between x_S and y_T if there is no mapping relationship defined between them. All concepts as x_T are `disjointWith` each other, and all concepts as y_T are also `disjointWith` each other.
  - *Calculation*: $D'(\langle x_S, m, y_T\rangle) = \frac{1}{(NoC(y_T)+1)\times(NoC(x_T)+1)}$ ($m =$ `Rule 4`)

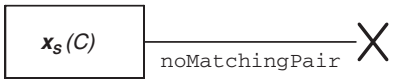

Figure 9 noMatchingPair concept mapping relationship from x_S

– *Reason*: The same reason as that for side-effect concept mapping set distribution of `Rule 1`. With $|x_S| = C$, $|x_S \rightarrow x_T| = \frac{C}{NoC(x_T)+1}$, $|x_S \rightarrow y_T| = \frac{|x_S \rightarrow x_T|}{NoC(y_T)+1}$, then $D'(\langle x_S, m, y_T\rangle) = \frac{|x_S \rightarrow y_T|}{|x_S|} = \frac{1}{(NoC(y_T)+1)\times(NoC(x_T)+1)}$.

5.2.4 *noMatchingPair*

Rule 5. `noMatchingPair` **concept mapping relationship**

- **Concept mapping set distribution**
  - *Calculation*: $D'(\langle x_S, m, x_T\rangle) = 0$ ($m =$ `noMatchingPair`)
  - *Reason*: Since x_S has `noMatchingPair` of x_T, any instance of x_S is not the instance of x_T, that is, $|x_S \rightarrow x_T| = 0$, then $D'(\langle x_S, m, x_T\rangle) = \frac{|x_S \rightarrow x_T|}{|x_S|} = 0$. An example of this is presented in Figure 9.

5.3 *Prediction of set distribution for precision and recall*

In this section, the calculation formulas are presented for the prediction of the set distribution, that is, D'_{DM}, D'_{IM}, and $D'_{DM\cap IM}$. The calculation formulas make use of the earlier presented calculation rules of Section 5.2.

5.3.1 D'_{DM} *calculation*

We use the definition from Section 3.4.5: $D'_{DM} = \frac{|DM|'}{|S|'}$. With the assumption of even instance distribution over concepts in SMQPM, the prediction value of the set $|S|'$ is the number of concepts in AK model S times the constant C which is the number of instances of each concept. The set $|DM|'$ is calculated by the sum of the **concept mapping set distributions** and the **side-effect concept mapping set distributions** of all the concept mappings from model S to T multiplied by the constant C. The concept mapping relationship from one concept x_S in model S to concepts in model T can be a 1 to 1 or 1 to multiple concepts mapping, so we use x_T to represent the set of concepts mapped from x_S. Detailed calculation formulas are presented in Equation (3). In this equation, n is the number of mapping relationships (including direct mapping or side-effect mapping caused by calculation rules) from x_S to T and $NoC(S)$ is the number of concepts in AK model S:

$$D'(\langle x_S, m, x_T\rangle) = \sum_{j=1}^{n} D'(\langle x_S, m, {x_T}^j\rangle), (x_T = \{{x_T}^1, \ldots {x_T}^j \ldots, {x_T}^n\} \subset T);$$

$$D'_{DM} = \frac{|DM|'}{|S|'} = \frac{C \times \sum_{i=1}^{NoC(S)} D'(\langle {x_S}^i, m, {x_T}^i\rangle)}{C \times NoC(S)} = \frac{\sum_{i=1}^{NoC(S)} D'(\langle {x_S}^i, m, {x_T}^i\rangle)}{NoC(S)}. \quad (3)$$

5.3.2 D'_{IM} *calculation*

D'_{IM} is the prediction of set distribution based on the concept mappings from S to T with the indirect mapping approach, in which concept mapping relationships from concepts of model S to central model C, and from the mapped concepts in C to T will occur. D'_{IM} can be calculated in a similar way as D'_{DM}. The only difference is that we use $D'_C(\langle x_S, m, x_T\rangle)$ to represent a prediction of the set distribution of individual concept mapping relationships through model C based on the two mapping relationships represented by triples $\langle x_S, m, x_C\rangle$ and $\langle x_C, m, x_T\rangle$. In these triples, x_C represents the set of concepts of the central model C mapped from x_S. x_T represents the set of concepts mapped from x_C. To distinguish it from other kinds of **concept mapping set distributions**, $D'_C(\langle x_S, m, x_T\rangle)$ will be called as the **combined concept mapping set distribution**. Its calculation

formula is described in two steps. In the first step, the **concept mapping set distribution** for each $x_C{}^j$ (concept mapped from x_S to central model C) is calculated by the sum of the product of the **concept mapping set distribution** and the **side-effect concept mapping set distribution** from x_S to $x_C{}^j$ and $x_C{}^j$ to T. In the second step, the **combined concept mapping set distribution** for x_S is calculated by the sum of the **concept mapping set distribution** for each $x_C{}^j$ mapped from x_S to C. Detailed calculation formulas are presented in Equation (4). In this equation, n is the number of mapping relationships (including direct mapping or side-effect mapping caused by calculation rules) from x_S to C. $l(j)$ is a function of parameter j and calculates the number of mapping relationships, including direct mappings and side-effect mappings caused by the calculation rules, from $x_C{}^j$ to T:

$$D'_C(\langle x_S,m,x_\mathrm{T}\rangle)=\sum_{j=1}^{n}\left(\sum_{k=1}^{l(j)}D'(\langle x_S,m,x_C{}^j\rangle)\times D'(\langle x_C{}^j,m,x_T{}^k\rangle)\right),$$

$$(x_\mathrm{C}=\{x_C{}^1,\ldots x_C{}^j\ldots,x_C{}^n\}\subset C,$$

$$x_\mathrm{T}=\{\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(1)}\}\cup\ldots\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(j)}\}\ldots\cup\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(n)}\}\}\subset T);$$

$$D'_{IM}=\frac{|IM|'}{|S|'}=\frac{\sum_{i=1}^{NoC(S)}D'_C(\langle x_S{}^i,m,x_\mathrm{T}{}^i\rangle)}{NoC(S)}. \tag{4}$$

5.3.3 $D'_{DM\cap IM}$ calculation

$D'_{DM\cap IM}$ is the prediction of the set distribution of the instances that belong to both the DM and IM sets. It is calculated in a similar fashion as D'_{IM}. The only difference is that part of the **combined concept mapping set distribution** in D'_{IM}, whose combined concept mapping relationships (indirect mapping caused by two concept mappings) do not belong to the direct concept mapping relationships from S to T, should be taken out. The reason is that this part of the **combined concept mapping set distribution** is not relevant to the **concept mapping set distribution** in D'_{DM}. We use $D'_{RC}(\langle x_S, m, x_\mathrm{T}\rangle)$ to represent the **relevant combined concept mapping set distribution** in D'_{IM}. Its calculation is the same as that for $D'_C(\langle x_S, m, x_\mathrm{T}\rangle)$, except for an additional parameter r. $r=1$ when the **combined concept mapping set distribution** is relevant, and $r=0$ when it is not. Detailed calculation formulas are given below in Equation (5), in which all the parameters, except for r, have the same meaning as those in Section 5.3.2:

$$D'_{RC}(\langle x_S,m,x_\mathrm{T}\rangle)=\sum_{j=1}^{n}\left(\sum_{k=1}^{l(j)}D'(\langle x_S,m,x_C{}^j\rangle)\times D'(\langle x_C{}^j,m,x_T{}^k\rangle)\times r\right),$$

$$(x_\mathrm{C}=\{x_C{}^1,\ldots x_C{}^j\ldots,x_C{}^n\}\subset C,$$

$$x_\mathrm{T}=\{\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(1)}\}\cup\ldots\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(j)}\}\ldots\cup\{x_T{}^1,\ldots x_T{}^k\ldots,x_T{}^{l(n)}\}\}\subset T);$$

$$D'_{DM\cap IM}=\frac{|DM\cap IM|'}{|S|'}=\frac{\sum_{i=1}^{NoC(S)}D'_{RC}(\langle x_S{}^i,m,x_\mathrm{T}{}^i\rangle)}{NoC(S)}. \tag{5}$$

5.4 Example of instance classification quality prediction

In this section, we exemplify the instance classification prediction. In this example, the LOFAR AK model (Jansen *et al.*, 2008) is used as the source model S and Kruchten's ontology (Kruchten, 2004) as the target model T. The core model proposed by de Boer *et al.* (2007) acts as the central model C. The LOFAR AK model is used to document the AK in the LOFAR project; this knowledge needs to be shared and reused over more than 25 years. Kruchten's ontology is an ontology of architectural design decisions which comprise a major part of AK, as defined by Kruchten *et al.* (2005). The core model is a first attempt to cover all the concepts from different AK models, and is thus a good candidate for a central model. Due to space limitations, the detailed concept mapping relationships and calculation of set distribution (i.e. D'_{DM}, D'_{IM}, and

$D'_{DM\cap IM}$) of this example can be found in Liang *et al.* (2008). Note, that the calculation result of $D'(\langle x_S, m, x_T\rangle) > 1$ is regarded as $D'(\langle x_S, m, x_T\rangle) = 1$ for the sum calculation in Equations (3)–(5). Since $D'(\langle x_S, m, x_T\rangle) \in [0,1]$, as defined in Section 5.2. We simply present the calculation results as follows: with $D'_{DM} = 0.835$, $D'_{IM} = 0.835$, and $D'_{DM\cap IM} = 0.680$, then,

$$P' = \frac{D'_{DM\cap IM}}{D'_{IM}} = \frac{0.680}{0.835} = 0.814 \text{ and } R' = \frac{D'_{DM\cap IM}}{D'_{DM}} = \frac{0.680}{0.835} = 0.814$$

6 Instance mapping experiment

In this section, a mapping experiment at the instance level for AK sharing is presented to validate the SMQPM. The experiment process is divided into two steps as illustrated in Figures 10 and 11. We use the same AK models, that is, *S*, *T*, and *C*, as defined in Section 5.4 for the instance mapping quality prediction. Note, that the 'instance mapping' in this section is performed manually by domain experts, while the 'instance classification' in the previous sections is performed automatically by an instance classification tool. The sample input for this experiment is a software architecture document (33 pages) of the LOFAR software system for data processing, denoted as LOFAR.DOC.

The first step is the instance mapping using a direct mapping approach. In this step, we manually annotate the LOFAR.DOC using the LOFAR AK model and Kruchten's ontology to construct two AK repositories. Both originate from the same architecture document (LOFAR.DOC) but use different AK models. After this, we map the AK instances from the LOFAR AK repository to the Kruchten AK repository based on the direct mapping relationship from the LOFAR AK model to Kruchten's ontology. In the instance mappings, only one mapping

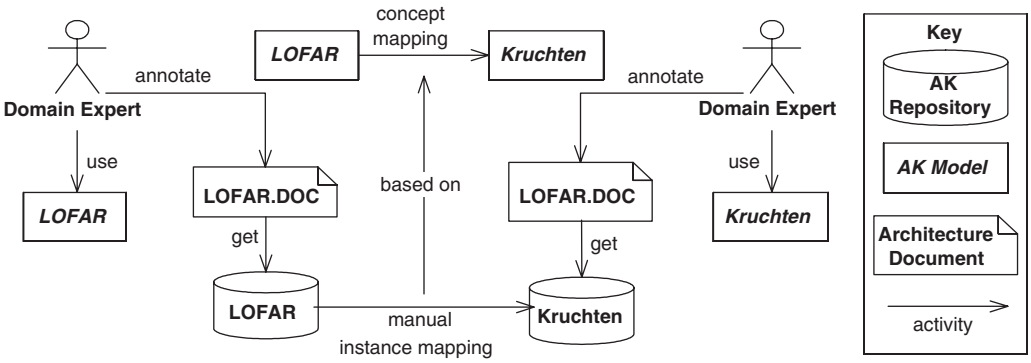

Figure 10 Instance mapping with direct mapping approach

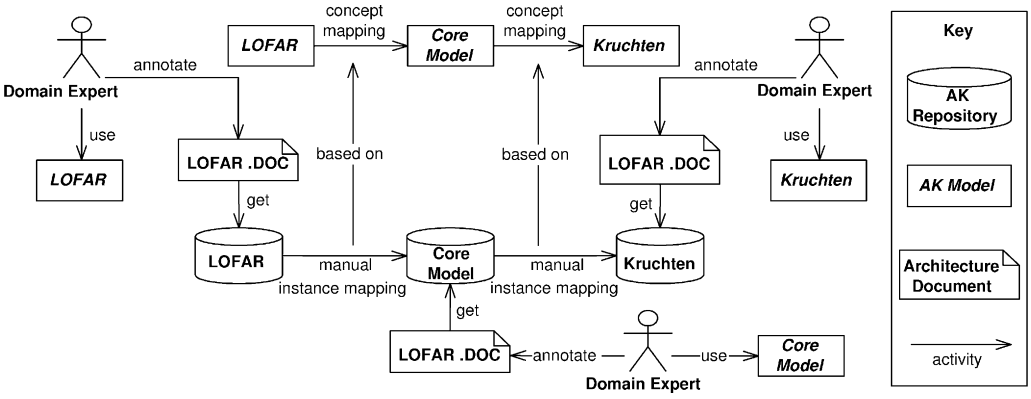

Figure 11 Instance mapping with indirect mapping approach

relationship is defined between instances: `sameAs`. For example, the text 'In the data factory architectural view the focus is on the configuration' (*instanceA*) is an instance of concept *Concern* in the LOFAR AK repository, while it is an instance of the concept *Requirement* of the Kruchten AK repository. With the direct mapping relationship: *Concern* is `superClassOf` *Requirement* and by the domain expert analysis: *instanceA* is an instance of concept *Requirement* we create the instance mapping: *instanceA* (`type`:*Concern*) `sameAs` *instanceA* (`type`:*Requirement*). The result of the first step is the set of instances mapped from the LOFAR to the Kruchten AK repository, that is, the set *DM*. With the set of instances in the source LOFAR AK repository, we can calculate the **set distribution** of set *DM* using: $D_{DM} = \frac{|DM|}{|LOFAR|}$.

The second step is the instance mapping using the indirect mapping approach. In this step, we manually annotate the LOFAR.DOC using the core model. The two AK repositories constructed in the previous step are reused to construct a total of three AK repositories that originate from the same architecture document (LOFAR.DOC), but use different AK models. Next, we map the AK instances from the LOFAR AK repository to the core model AK repository and subsequently to the Kruchten AK repository using the direct mapping relationships for both instance mappings, as presented in Figure 11. In both mappings, the instance mapping relationship `sameAs` is used similar to the first step. The result of the second step is the set of instances mapped from the LOFAR to the Kruchten AK repository through the core model, that is, the set *IM*. With the set of instances in the source LOFAR AK repository, we can calculate the **set distribution** of set *IM*: $D_{IM} = \frac{|IM|}{|LOFAR|}$.

Finally, we compare the instance mappings between the two sets *DM* and *IM* to determine which instance mapping are relevant and which are not in the set *IM*, then we can get the intersection of sets $DM \cap IM$. Then we can calculate the **set distribution** of set $DM \cap IM$: $D_{DM \cap IM} = \frac{|DM \cap IM|}{|LOFAR|}$. With the value of set distribution (D'_{DM}, D'_{IM}, and $D'_{DM \cap IM}$) and calculation Equation (2) defined in Section 3.4.5, we can get the precision (*P*) and recall (*R*) for this instance mapping experiment: $P = 0.958$ and $R = 0.912$.

7 Comparison analysis

In this section, we compare the direct and indirect mapping approaches with respect to their quality and cost. In addition, the prediction of the sharing quality is compared with the real value. An overview of the results is presented in Table 2. For both the precision and the recall, **normalized numbers** are shown to ease the comparison between the two mapping approaches. The values of the direct mapping ($P' = R' = P = R = 1$) form the basis for this normalization. The prediction and real value of the precision and recall of the indirect mapping approach come from the values (P', R', P and R) calculated in Section 5.4 and 6. The cost of AK sharing consists of five aspects: modeling (MOD), capturing (CAP), mapping (MAP), evolution (EVO), and extension (EXT) costs, as presented in Section 4.

Looking at the table, we can conclude that for the sharing quality our estimation was too pessimistic. This is reasonable, as the LOFAR AK model, the core model, and Kruchten's

Table 2 Comparison of the direct and indirect mapping approach

		Quality		Cost			
AK sharing approach	Prediction or Real	Precision	Recall	MOD	CAP	MAP	EVO/EXT
Direct Mapping	Prediction	1	1	$O(m \times n)$	$O(n \times k)$	$O(n^2 \times m)$	$O(k + m \times n)$
	Real	1	1	34	224	34	146
Indirect Mapping	Prediction	0.814	0.814	$O(m \times n)$	$O(n \times k)$	$O(n \times m)$	$O(k \times m)$
	Real	0.958	0.912	34	224	58	124

AK = architectural knowledge; MOD = modeling; CAP = capturing; MAP = mapping; EVO/EXT = evolution/extension.

ontology all place stress on architectural design decisions, and therefore have many overlapping concepts in AK documentation. Comparing the direct and indirect mapping approaches with each other, we find, quite surprising, that for both the precision and recall we only lose 5% and 10% in quality, respectively, when using an indirect mapping approach. This is partially due to the fact that the employed central model (the core model) has been influenced by the existence of the LOFAR AK model, thereby covering many of LOFAR model concepts to some extent. If we look at the cost, then an indirect mapping approach starts paying off when more AK models are involved in the knowledge grid. In that case, the mapping, evolution, and extension costs become considerably lower, as compared to the costs of a direct mapping approach.

Concluding, it seems that the indirect mapping approach loses some quality, although not much. The cost benefit is however huge, if more than three AK models are being used. However, these are very preliminary results, as only two AK models, one central, and one architectural document have been tested so far. Further experimentation on several AK models and architecture documents is needed to draw more firm general conclusions.

8 Conclusions and future work

In this paper, we presented a mapping quality prediction model (SMQPM) that is composed of the following: (1) It makes specific assumptions for the quality prediction of AK sharing; (2) it defines evaluation criteria for AK sharing quality by introducting the precision and recall from the IR theory; and (3) it specifies a calculation method and rules for the prediction of these criteria. The direct and indirect mapping approaches are evaluated in terms of the quality and cost for AK sharing. According to the comparison analysis result, stakeholders involved with AK sharing can select an appropriate approach by trading off quality and cost in their own context. In order to do so, one needs to consider the number of AK repositories involved with AK sharing, and the number of instances in those AK repositories.

We outline our future work in several points: (1) More AK models should be covered and AK repositories should be included for the validation of mapping quality prediction models. (2) The mapping quality prediction models RMQPM and AMQPM, which contain more realistic assumptions as specified in Section 5.1, should be investigated. (3) The relationships between AK instances as specified in Kruchten (2004) and de Boer *et al*. (2007) are lost in the currently proposed AK sharing scenarios, which result in traceability problems. For example, a relationship exists between the AK instances of concept *Alternative* and concept *Decision Topic* in that *instanceA* (`type`: *Alternative*) `isProposedFor` *instanceB* (`type`: *Decision Topic*) in an architecture design. A solution that retains the relationships between AK instances for AK sharing needs to be investigated. (4) Tool support for AK model mapping and the quality and cost prediction calculation needs to be implemented in order to automate the central model evaluation.

Acknowledgements

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge, and Peng Liang is funded by the project Hefboom 641.000.405. The authors would like to thank Astron for their support and access to the LOFAR software architecture documents.

References

Aguiar, A. & David G. 2005. WikiWiki weaving heterogeneous software artifacts. In *Proceedings of the 1st International Symposium on Wikis (WikiSym)*, San Diego, CA, USA, 67–74.

Akerman, A. & Tyree, J. 2005. Position on ontology-based architecture. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Pittsburgh, PA, USA, 289–290.

Ali-Babar, M., Gorton, I. & Kitchenham, B. 2006. A framework for supporting architecture knowledge and rationale management. In *Rationale Management in Software Engineering*, Dutoit, A. H., McCall, R., Mistrik, I. & Paech, B. (eds). Springer, 237–254.

Avgeriou, P., Kruchten, P., Lago, P., Grisham, P. & Perry, D. 2007. Architectural knowledge and rationale: issues, trends, challenges. *ACM SIGSOFT Software Engineering Notes* **32**(4), 41–46.

Bachmann, F. & Merson, P. 2005. Experience using the Web-based tool wiki for architecture documentation. *Technical note SEI-2005-TN-041*, Carnegie Mellon University, Pittsburgh, PA, USA.

Bass, L., Clements, P. & Kazman, R. 2003. *Software Architecture in Practice*, 2nd edn. Addison-Wesley.

Bosch, J. 2004. Software architecture: the next step. In *Proceedings of the 1st European Workshop on Software Architecture (EWSA)*, St Andrews, UK, 194–199.

Brickley, D. & Guha, R. V. 2004. *RDF Vocabulary Description Language 1.0: RDF Schema*. Recommendation, W3C.

Camon, E., Magrane, M., Barrell, D., Lee, V., Dimmer, E., Maslen, J., Binns, D., Harte, N., Lopez, R. & Apweiler, R. 2004. The gene ontology annotation (GOA) database: sharing knowledge in UniProt with gene ontology. *Nucleic Acids Research* **32**(90001), W313–W317.

Capilla, R., Nava, F., Pérez, S. & Dueñas, J. C. 2006. A Web-based tool for managing architectural design decisions. *ACM SIGSOFT Software Engineering Notes* **31**(5), 4–11.

Choi, N., Song, I. Y. & Han, H. 2006. A survey on ontology mapping. *ACM SIGMOD Record* **35**(3), 34–41.

Cleverdon, C. W. 1967. The Cranfield tests on index language devices. *Aslib Proceedings* **19**(6), 173–193.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Schneider, P. F. & Stein, L. A. 2004. *OWL Web Ontology Language Reference*. Recommendation, W3C.

de Boer, R. C., Farenhorst, R., Lago, P., van Vliet, H., Clerc, V. & Jansen, A. 2007. Architectural knowledge: getting to the core. In *Proceedings of the 3rd International Conference on the Quality of Software-Architectures (QoSA)*, Boston, USA, 197–214.

Ehrig, M. & Euzenat J. 2005. Relaxed precision and recall for ontology matching. In *Proceedings of the K-CAP Workshop on Integrating Ontologies (IntOnt)*, Banff, Canada, 25–32.

Ehrig, M. & Staab S. 2004. QOM-quick ontology mapping. In *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, Hiroshima, Japan, 683–697.

Euzenat, J. 2007. Semantic precision and recall for ontology alignment evaluation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 348–353.

Fonseca, F. T., Egenhofer, M. J., Davis, C. A. & Borges, K. A. V. 2000. Ontologies and knowledge sharing in urban GIS. *Computers, Environment and Urban Systems* **24**(3), 251–272.

Gruber, T. R. 1993. A translation approach to portable ontologies. *Knowledge Acquisition* **5**(2), 199–220.

Institute of Electrical and Electronics Engineers. 2000. IEEE recommended practice for architecture description of software intensive system, IEEE Std 1471-2000, IEEE.

Jansen, A. & Bosch J. 2005. Software architecture as a set of architectural design decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Pittsburgh, PA, USA, 109–119.

Jansen, A., de Vries, T., Avgeriou, P. & van Veelen M. 2008. Sharing the architectural knowledge of quantitative analysis. In *Proceedings of the 4th International Conference on the Quality of Software-Architectures (QoSA)*, Karlsruhe, Germany.

Jansen, A., van der Ven, J., Avgeriou, P. & Hammer D. K. 2007. Tool support for architectural decisions. In *Proceedings of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Mumbai, India, 44–53.

Kalfoglou, Y. & Schorlemmer, M. 2003. Ontology mapping: the state of the art. *Knowledge Engineering Review* **18**(1), 1–31.

Kruchten, P. 2004. An ontology of architectural design decisions in software intensive systems. In *Proceedings of the 2nd Groningen Workshop on Software Variability Management (SVM)*, Groningen, The Netherlands, 54–61.

Kruchten, P., Lago, P., van Vliet, H. & Wolf T. 2005. Building up and exploiting architectural knowledge. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Pittsburgh, PA, USA. 291–292.

Lago, P. & Avgeriou, P. 2006. First workshop on sharing and reusing architectural knowledge. *ACM SIGSOFT Software Engineering Notes* **31**(5), 32–36.

Liang, P., Jansen, A. & Avgeriou, P. 2008. *A Case of Quality Prediction of Architecture Knowledge Sharing Through Model Mapping*. Technical report RUG-SEARCH-08-L01, University of Groningen, Groningen, The Netherlands.

Louridas, P. 2006. Using wikis in software development. *IEEE Software* **23**(2), 88–91.

Medvidovic, N. & Taylor, R. N. 2000. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering* **26**(1), 70–93.

Perry, D. E. & Wolf, A. L. 1992. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes* **17**(4), 40–52.

Silveira, C., Faria, J. P., Aguiar, A. & Vidal R. 2005. Wiki based requirements documentation of generic software products. In *Proceedings of the 10th Australian Workshop on Requirements Engineering (AWRE)*, Melbourne, Australia, 42–51.

Tang, A., Babar, M. A., Gorton, I. & Han, J. 2006. A survey of architecture design rationale. *The Journal of Systems & Software* **79**(12), 1792–1804.

Tang, A., Jin, Y. & Han, J. 2007. A rationale-based architecture model for design traceability and reasoning. *The Journal of Systems & Software* **80**(6), 918–934.

Tyree, J. & Akerman, A. 2005. Architecture decisions: demystifying architecture. *IEEE Software* **22**(2), 19–27.

Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E. & Ciravegna F. 2006. Semantic annotation for knowledge management: requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web* **4**(1), 14–28.

van der Ven, J., Jansen, A., Nijhuis, J. & Bosch, J. 2006. Design decisions: The bridge between rationale and architecture. In *Rationale Management in Software Engineering*, Dutoit, A. H., McCall, R., Mistrik, I. & Paech, B. (eds). Springer, 329–346.

Zhuge, H. 2004. *The Knowledge Grid*. World Scientific.