Advanced quality prediction model for software architectural knowledge sharing

Peng Liang [a,*], Anton Jansen [c], Paris Avgeriou [b], Antony Tang [d], Lai Xu [e]

[a] Wuhan University, State Key Lab of Software Engineering, School of Computer, 430072 Wuhan, China
[b] University of Groningen, Department of Mathematics and Computing Science, Nijenborgh 9, 9700 AK Groningen, The Netherlands
[c] ABB Corporate Research, Industrial Software Systems, Forskargränd 8, SE-72178 Västerås, Sweden
[d] Swinburne University of Technology, Faculty of Information and Communication Technologies, VIC 3122, Melbourne, Australia
[e] SAP Research Switzerland, Blumenbergplatz 9, 9000 St. Gallen, Switzerland

ARTICLE INFO

ABSTRACT

In the field of software architecture, a paradigm shift is occurring from describing the outcome of architecting process to describing the Architectural Knowledge (AK) created and used during architecting. Many AK models have been defined to represent domain concepts and their relationships, and they can be used for sharing and reusing AK across organizations, especially in geographically distributed contexts. However, different AK domain models can represent concepts that are different, thereby making effective AK sharing challenging. In order to understand the mapping quality from one AK model to another when more than one AK model coexists, AK sharing quality prediction based on the concept differences across AK models is necessary. Previous works in this area lack validation in the actual practice of AK sharing. In this paper, we carry out validation using four AK sharing case studies. We also improve the previous prediction models. We developed a new advanced mapping quality prediction model, this model (i) improves the prediction accuracy of the recall rate of AK sharing quality; (ii) provides a better balance between prediction effort and accuracy for AK sharing quality.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

A paradigm shift is occurring in the field of software architecture (Avgeriou et al., 2007, 2009). The products of the software architecting process are no longer limited to architectural models and views, but it has a broader notion of Architectural Knowledge (AK) (Kruchten et al., 2006): the architecture design as well as the design decisions, rationale, assumptions, context, and other factors that together determine architecture solutions. Architectural (design) decisions are an important type of AK, as they form the basis underlying a software architecture (Jansen and Bosch, 2005). Other types of AK include concepts from architectural design (e.g., components, connectors), requirements engineering (e.g., risks, concerns, requirements), people (e.g., stakeholders, organization structures, roles), and development process (e.g., activities) (De Boer et al., 2007).

The entire set of AK needs to be iteratively produced, shared, and consumed during the whole architecture lifecycle by a number of different stakeholders as effectively as possible (Liang et al., 2010). These stakeholders may belong to the same or different organiza-

tion and include roles such as: architects, requirement engineers, developers, maintainers, testers, end users, and managers, etc. Each of the stakeholders, who are also knowledge workers, has her/his own area of expertise and a set of concerns in a system being developed, maintained or evolved. The architect needs to facilitate the collaboration between the stakeholders, by providing AK through a common language for communication and negotiation, and eventually makes the necessary design decisions and trade-offs, potentially in a collaborative architecting context (Liang et al., 2010).

However, in practice, there are several issues that hinder effective stakeholders' collaboration during the architecting process, which diminishes the quality of the resulting product. One of the fundamental problems is the lack of effective ways to share AK between stakeholders; this is not a common practice at present (Tang et al., 2006; Lago et al., 2008). The cause of this problem is that different stakeholders typically have different backgrounds, and use their own AK domain models (ontologies) and set of preferred AK tools. The result is a mosaic of activities and artifacts rather than a uniform process and a solid product (Liang et al., 2010). Consequently the stakeholders speak a different AK language, and the translation from one AK language to another AK language may be lost.

One can share AK through the use of concept mappings between different AK domain models, which allows users to perform automatic translation of AK instances in these models based on AK

* Corresponding author. Tel.: +86 27 68776056; fax: +86 27 68776027.
E-mail addresses: liangp@sklse.org, pliangeng@gmail.com
(P. Liang), anton.jansen@se.abb.com (A. Jansen), paris@cs.rug.nl (P. Avgeriou),
atang@swin.edu.au (A. Tang), lai.xu@sap.com (L. Xu).

concept mappings. The issue caused by automatic AK translation (i.e., AK sharing) is that part of AK instances can be translated into instances of a concept that does not fully represent the original concept or a wrong concept. Therefore, it is necessary to consider the quality and cost that such translation brings by evaluating prediction methods of AK mapping. The evaluation is used to predict how well the knowledge that is contained in one AK model can be mapped to another model. This is important because organizations with different AK models would want to predict the quality of AK sharing.

In our previous work, we proposed two Mapping Quality Prediction Models (MQPMs) to do so: The Simple MQPM (SMQPM) (Liang et al., 2009a) and the Random MQPM (RMQPM) (Liang et al., 2008). MQPMs predict the accuracy[1] of AK sharing between AK domain models given a mapping between them. In (Liang et al., 2009a), we compare two different mapping approaches using SMQPM to measure their cost-effectiveness. In (Liang et al., 2008), RMQPM has been used to select the most appropriate AK domain model as a standard model to translate AK between specific AK domain models. The problem of these two prediction models is twofold. Firstly, they are based on restrictive assumptions that are not realistic in AK sharing practices. Secondly, they have not been validated in industrial practices. We need to improve these aspects to provide better AK mapping prediction.

In this paper, we present the Advanced MQPM (AMQPM), which is based on a refinement of our earlier prediction models (SMQPM and RMQPM). To evaluate the accuracy and cost-effectiveness of AMQPM, we compare its predictions with the outcomes of the SMQPM and RMQPM models using four AK case studies. The results of manual AK mappings are further evaluated by domain experts.

The state-of-the-art of AK sharing practice and the related challenges are introduced and discussed in Section 2. The detailed description of three mapping quality prediction models (MQPMs) with their specific assumptions are presented in Section 3. The refined AMQPM and related calculation method are described in Section 4. To validate AMQPM, four experimental case studies and their results are presented in Section 5. The results are evaluated and discussed in Section 6. In Section 7, we present related work on knowledge sharing methods. The limitations of our work are discussed in Section 8. Our conclusions and future work are outlined in Section 9.

2. Software architectural knowledge sharing

Software developers are knowledge workers. They usually do not operate in isolation, but are typically part of one or more social networks, and communities of people they interact with (Lago, 2009). There is a need for them to share knowledge for taking decisions on design issues, applying patterns, negotiating solutions, and so on. With the increasing trend of distributed software development, e.g., Global Software Development (GSD), sharing and reusing AK across organizations becomes a critical factor for project success (Jansen and Bosch, 2005). Sharing AK is essential for effective communication between distributed teams that are responsible for different software development activities, e.g., requirement analysis, architecture design, and detailed design, etc. It is also an important part of all architecting activities like modifying past design decisions, performing architecture reviews, and trading off quality attribute requirements.

AK can be classified in several types. In knowledge management, a distinction is often made between two types of knowledge:

implicit and explicit knowledge (Nonaka and Takeuchi, 1995). Implicit (or tacit) knowledge is knowledge residing in people's heads, whereas explicit knowledge is knowledge that has been codified in some form (e.g., a document or a model). Two forms of explicit knowledge can be discerned: documented and formal knowledge. Documented knowledge is explicit knowledge that is expressed in natural language or images in documents. Typical examples of documented AK are Word and Excel documents that contain architecture description and analysis models. Formal knowledge is explicit knowledge codified using a formal language or model of which the exact semantics are defined. Typical examples of formal AK models include architectural (design) decisions ontology (Kruchten, 2004) and AK domain models (Capilla et al., 2006; Jansen et al., 2009; Tang et al., 2007; Zimmermann et al., 2007; Ali-Babar et al., 2006) that formally define concepts and relationships, and aim at providing a common language for unambiguous interpretation by stakeholders. In AK, much work has been done in mining existing AK bases or bodies of knowledge to generalize and codify AK instances in formal ontologies (Liang and Avgeriou, 2009). The focus of this paper is on formal AK sharing which is based on the AK instances annotated by using various AK domain models and ontologies.

Most of the existing AK management tools, which have an AK sharing function, only support documented AK sharing or formal AK sharing among the organizations, who employ the same AK domain model (Liang and Avgeriou, 2009). Sharing formal AK between different organizations or even between departments of a single organization, who have adopted different AK domain models, poses a great challenge: the domain models of AK are not standardized. On the contrary, they tend to vary enormously. In fact, various researchers and practitioners have proposed their own AK domain models or ontologies as mentioned before to document AK concepts and their relationships. Some of these concepts and relationships are different, while others are largely overlapping (Tang et al., 2010). These discrepancies among the AK domain models hampers 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 not specific in the field of AK, but is quite common in other fields, e.g., knowledge sharing in gene data (Camon et al., 2004) and geographic information systems (Fonseca et al., 2000).

In our work, we look at the problem of AK sharing through a knowledge grid perspective (Jansen et al., 2007; Zhuge, 2004). In this envisioned AK grid, AK is captured (annotated) in various AK domain models. For instance, a multi-site software development organization sharing different AK models across sites. Using the concept mappings between these models, all AK instances annotated by specific AK domain models, are transparently and automatically shared (mapped) from one AK domain model to the other among the interested stakeholders based on the AK concept mappings. The AK sharing activity raises the issue of the cost and quality of AK sharing, which is not only dependent on the AK domain models and concept mappings involved, but also on the actual AK instances related to these models. Only with these AK instances, the real cost and quality of AK sharing can be determined. However, annotating and mapping AK instances takes considerable effort compared to the effort of AK concept mappings, as a huge number of instances are involved and human intervention is required. To make matters worse, these efforts need to be continuous, as the AK domain models or concept mappings continue to evolve. Hence, we would like to predict the quality of AK sharing in advance before effort is spent on annotating AK instances. The other motivation for AK sharing quality prediction is that it can be used to predict the quality of automatic AK sharing (i.e., AK instance mapping) compared to a manual approach. As mentioned in Sec-

---

[1] The term accuracy refers how close the prediction of precision and recall rate, and *F*-measure of AK instances mapping results to the real values (precision and recall rate) of AK sharing as presented in Section 4.1.
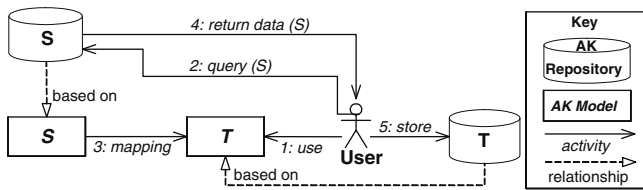
Fig. 1. Query-based scenario for architectural knowledge sharing.

tion 1, two Mapping Quality Prediction Models (MQPMs) have been proposed for the prediction of AK sharing quality: Simple MQPM (SMQPM) (Liang et al., 2009a) and the Random MQPM (RMQPM) (Liang et al., 2008). In Section 3, we elaborate on these prediction models and AK sharing.

3. Mapping Quality Prediction Model (MQPM)

3.1. AK sharing by concept mapping

Formal AK sharing takes place at two levels of abstraction: the conceptual level and the instance level. At the conceptual level, an AK domain model defines the *concepts* and *relationships* that a particular organization, department, project, or person uses. At the instance level, the actual AK *instances* of the aforementioned concepts and relationships are stored in an AK repository. The sharing of AK instances based on different AK domain models depends on the mutual understanding of the underlying AK models, i.e., one concept in one AK domain model can be *translated* (*mapped*) into a concept in the other AK domain model. Thus this mutual understanding can be specified by a set of mapping relationships between concepts from different AK domain models.

As mentioned in the previous section, we envision AK sharing in an AK grid, i.e., a heterogeneous AK repository that is comprised of different local AK repositories. Each local repository contains the definition of an AK domain model and its instances. A user can retrieve AK from all participating AK repositories transparently without being conscious of the underlying AK model differences. To quantify the AK sharing quality, we use a specific scenario in the form of a user query. Such a query is a typical activity performed during knowledge sharing. The query is a precise request for information retrieval, typically expressed as keywords combined with boolean operators and other modifiers. The query-based scenario is shown in Fig. 1. The numbers (1–5) in the figure denote the execution sequence of each activity (steps) in the scenario. A user who understands only AK domain model T queries the repository of AK domain model S (Step 2) using concepts from model T as query keywords (Step 1). The conceptual difference between AK domain model S and T poses a problem for AK sharing. The queried concepts from model T do not exist (or exist, but have a different meaning) in model S. Thus, the AK repository of model S cannot return any data (AK instances). Using concept mappings from model S to T (Step 3), the AK repository of model S can return partial data to the user (Step 4), and finally the returned data can be stored in the AK repository of model T for further usage (Step 5).

In AK domain models, concepts are defined as classes. The mapping relationships between concepts of AK domain models are therefore defined as relationships between classes. We use the following concept mapping relationships to relate AK domain 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.

- noMatchingPair, denotes that a concept cannot be mapped to another AK domain model.

Note that there are also many other concept mapping relationships besides the aforementioned four, such as disjointWith, compositionOf, and partOf, etc. The four relationships were selected based on two reasons: they can represent most of mapping semantics between AK concepts by a detailed analysis of a series of AK domain models and concept mappings between them (Liang et al., 2007); they can be readily represented in RDF Schema (Brickley and Guha, 2004) or OWL (Bechhofer et al., 2004), which are the most widely used languages for formal knowledge management (description, annotation, and sharing) in the semantic web (Shadbolt et al., 2006). The MQPMs are heavily based on the mapping semantics of these four concept mapping relationships. An AK domain model is composed of AK concepts and the relationships between them. Thus, the mapping quality (for a detailed description of mapping quality, see Section 4.1) between AK domain models can be represented by the aggregation of the mapping quality between their compositional AK concepts, as determined by their concept mapping relationships.

3.2. Theoretical background of AK sharing quality

As described in Section 3.1, AK sharing can be viewed as a combination of an information publication task with an Information Retrieval (IR) task that involves a query to share knowledge. The quality of this sharing can be quantified in terms of precision and recall rate (Liang et al., 2009a). The precision and recall rate originated from IR theory for the quality evaluation of IR results (Cleverdon, 1967). The precision rate is the proportion of *retrieved data* that is *relevant*. The recall rate is the proportion of *relevant data* that is *retrieved*. In an AK sharing context, the precision and recall rate can be reinterpreted as follows: the precision rate is the percentage of the amount of AK instances which can be correctly mapped (retrieved) from the source to the destination AK model compared to the amount of AK instances being mapped; the recall rate is the percentage of the amount of AK instances which can be correctly mapped (retrieved) from the source to the destination AK model compared to the amount of (relevant) AK instances belonging to the source AK model. Precision and recall rate address the AK sharing quality from different perspectives. There is a trade-off relationship between precision and recall rate, where it is possible to increase one at the cost of reducing the other (Buckland and Gey, 1994). Consequently, the *F*-measure is proposed to represent the overall quality of an IR task (i.e., AK sharing) taking both of precision and recall rate into considerations (Baeza-Yates et al., 1999). The detailed definition and calculation of the precision, recall rate, and *F*-measure in advanced mapping quality prediction model (AMQPM) for AK sharing are further presented in Section 4.1.

3.3. Three mapping quality prediction models

In practice, a domain expert can perform the AK instance mapping manually by following the AK concept mapping relationships[2] defined at the conceptual level, and then calculate the precision and recall rate using *relevant data*, *retrieved data*, and *relevant retrieved data* obtained from AK sharing as an IR task. To avoid the cumber-

---

[2] In the context of AK sharing, the AK concept mapping relationships between two AK domain models are defined by domain experts (e.g., software architects), and these concept mapping relationships are fixed (not changeable) when performing AK instance mappings in this paper, since we assume that domain experts can decide the best way of AK concept mapping for AK instance sharing.

some instance mapping work at the AK instance level, we try to map the AK instances automatically based on the concept mapping relationships, and predict the precision and recall rate of automatic instance mapping based on a MQPM (mapping quality prediction model). In our previous work (Liang et al., 2009a, 2008), two MQPMs, namely SMQPM and RMQPM, for the calculation of precision and recall rate were proposed based on the mapping relationships at the conceptual level, and different assumptions were assigned to these MQPMs.

In these models, two fundamental assumptions are considered:

1. whether the AK instances are evenly distributed over AK model concepts or not. Even distribution means that each AK concept in an AK domain model has the same number of AK instances in an AK repository. For example, *Design Decision* and *Alternative* are two AK concepts in an AK domain model, and they have the same number of instances of *Design Decision* and *Alternative* if this assumption is true. However, this assumption is not realistic in practice. For example, the number of instances of *Alternative* is always greater than that of *Design Decision*. The reason why we make this assumption is to simplify the calculation of precision and recall rate (see Section 4.1 for a detailed description) since it takes substantial effort to acquire the number of AK instances in an AK repository; and
2. whether the instance classifier employed is intelligent (maps AK instances to instances of correct AK concepts) or random (maps AK instances to instances of possible concepts randomly). For example, the AK concept *Requirement* is a superClassOf the concepts *Functional Requirement* and *Non-functional Requirement*. The *Requirement* has 10 instances, which is composed of 8 instances of *Functional Requirement* and 2 instances of *Non-functional Requirement*. With an intelligent instance classifier, the 8 functional requirements and 2 non-functional requirements can be mapped correctly to the concepts *Functional Requirement* and *Non-functional Requirement*, respectively. On the other hand, with a random classifier, all the 10 *Requirement* instances with a replicating approach (or every 5 instances with a splitting approach) will be randomly mapped to the two concepts *Functional Requirement* and *Non-functional Requirement* without considering whether these instance mappings are correct or not.[3] This kind of random instance mapping will inevitably cause some wrong instance mappings. For example, an instance of *Functional Requirement* is mapped to an instance of *Non-functional Requirement*, and consequently reduce the precision and recall rate[4] of AK instance mapping results.

The three mapping quality prediction models (SMQPM, RMQPM, and advanced MQPM (AMQPM)) with their respective assumptions are presented in Table 1 and compared with the associated assumptions of a real AK sharing case (i.e., the manual mapping by domain experts).

The detailed concept mapping relationships and related calculation methods for mapping quality prediction are presented in Section 4.2.

The SMQPM (simple mapping quality prediction model) was adopted in (Liang et al., 2009a) to predict AK sharing quality and cost. Of these three MQPMs, the SMQPM has the most optimistic assumptions: the AK instances are evenly distributed in the AK

Table 1
Assumptions of three MQPMs and real AK sharing case.

	Even distribution of AK instances	AK instance classifier
SMQPM	Yes	Intelligent
RMQPM	Yes	Random
AMQPM	No	Random
Real Case	No	Intelligent

repository, and the AK instance classifier always perfectly maps AK instances into the correct AK concepts. In this case, the prediction of a theoretical maximum precision (P_{SMQPM}) and recall (R_{SMQPM}) rate can be achieved. The disadvantage of SMQPM is that it is overly optimistic in practice. RMQPM (random mapping quality prediction model) was employed in (Liang et al., 2008) to calculate the semantic distances (reciprocal of *F*-measure) for selecting the most appropriate AK domain model as the core model for AK sharing in an indirect mapping approach. RMQPM also takes the assumption of even distribution of AK instances, but has the most pessimistic assumptions about the AK instance classifier (but more realistic than SMQPM) in which the AK instance classifier maps the AK instances to possible concepts randomly. In that case, the prediction of a theoretical minimum precision P_{RMQPM} and recall (R_{RMQPM}) rate can be achieved. The AMQPM has more realistic (neither optimistic nor pessimistic) assumptions compared with those for SMQPM and RMQPM. In the case of AMQPM, the prediction of precision (P_{AMQPM}) and recall (R_{AMQPM}) rate should be intuitively between the predictions of RMQPM and SMQPM. The conjectures on the relationship of the comparison among the predictions (precision (P) and recall (R) rate) of using the three MQPMs are presented in 1 and 2 respectively. Conjecture 3 is derived from Conjectures 1 and 2 to present the comparison relationship of the *F*-measure (*F*) prediction, calculated by Formula 9. The *F*-measure represents the overall quality of AK sharing including precision and recall rate. The *F*-measure increases when the precision or recall rate increases. Note that, the purpose of these three prediction models is not to achieve high *F*-measure, but to achieve accurate prediction results that are close to the *F*-measure, recall and precision rate of real AK sharing case. The real case acts as a benchmark for AK sharing quality between certain AK domain models, because we assume that the result of manual AK instance annotation and mapping by domain experts is perfect (intelligent) as specified in Table 1. In this paper, all three MQPMs are employed to predict the AK sharing quality in order to validate these conjectures.

$$P_{RMQPM} \leq P_{AMQPM} \leq P_{SMQPM} \tag{1}$$

$$R_{RMQPM} \leq R_{AMQPM} \leq R_{SMQPM} \tag{2}$$

$$F_{RMQPM} \leq F_{AMQPM} \leq F_{SMQPM} \tag{3}$$

In conclusion, two major issues still exist in our previous work on the SMQPM and RMQPM, which affect the wide applications of MQPM in AK sharing practices: (1) the SMQPM and RMQPM are based on quite restrictive assumptions that are not realistic for practical AK sharing activities; (2) the two MQPMs are only proved at the AK conceptual level without any validation in industrial AK sharing practices at the AK instances level to establish its credibility. In Section 4, the formulae and methods of AMQPM are presented in detail.

4. Advanced Mapping Quality Prediction Model (AMQPM)

In the SMQPM and RMQPM, the *weight* of each AK concept, namely the amount of AK instances belonging to each AK concept is defined as a constant *C*, which is feasible due to the assumption of even distribution of AK instances. However, for AMQPM, this assumption can be relaxed. The *weight* of each AK concept should

---

[3] For easier understanding, the dummy concept and the replicating and splitting approach for AK instance mapping in superClassOf mapping relationship is not mentioned here, which will be discussed in Section 4.2.

[4] Refer to Section 4.1 for descriptions of the extended meaning of precision, recall rate, and *F*-measure in AMQPM.
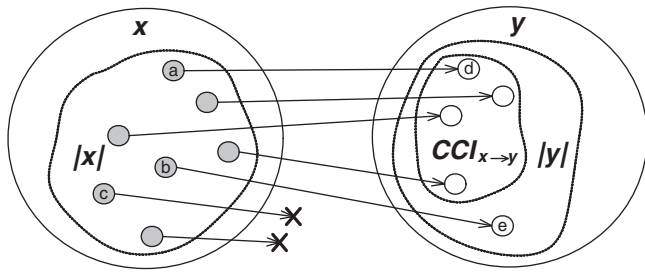
Fig. 2. Instance mapping scenarios for AK sharing.

be treated as a variable in the calculation methods of AMQPM, making AMQPM more realistic. In this section, we present the formulae and the associated methods that make up AMQPM.

4.1. Precision, recall rate, and F-measure in AMQPM

As mentioned in Section 3.1, AK sharing can be viewed as an IR task in which knowledge is shared by querying AK repositories. An AK repository is composed of AK instances belonging to certain AK concepts represented in a corresponding AK domain model. Fig. 2 illustrates AK sharing from the perspective of AK instance mapping based on AK concept mappings. The two bigger circles x and y represent two AK concepts in different AK domain models, and the small dots inside each concept circle represent the AK instances belonging to them. The directed arrows between the small dots are AK instance mappings. If concept x (e.g., *Human*[5]) is a superClassOf y (e.g., *Man*), three instance mapping scenarios may exist:

- an instance of concept x is mapped correctly as an instance of concept y, e.g., instance a (*John* is a *Human*(a) and he is also a *Man*(d));
- an instance of concept x is mapped as instance of concept y, but it is not a correct mapping, e.g., instance b (*Tom* is a *Human*(b), but he is not a *Man*, he is a *Boy*(e)). This instance mapping scenario is possible in practice by an instance classifier since concept *Man* and *Boy* are similar;
- an instance of concept x cannot be mapped as an instance of concept y, e.g., instance c (*Mary* is a *Human*(c), but she is not a *Man*, she is a *Woman*).

According to the query-based knowledge sharing scenario described in Section 3.1, three AK instance sets, originating from the IR theory, can be retrieved for the calculation of precision and recall rate (see Section 3.3):

- $|x|$: all the AK instances to be mapped in concept x regardless whether they are mappable or not, e.g., a, b, $c \in |x|$. This is the *Relevant data* in IR theory;
- $|y|$: all the AK instances mapped to concept y regardless whether they are correctly mapped or not, e.g., d, $e \in |y|$. This is the *Retrieved data* in IR theory;
- $CCI_{x \rightarrow y}$: all correctly mapped AK instances from concept x to y, e.g., $d \in CCI_{x \rightarrow y}$. This is the *Relevant retrieved data* in IR theory.

[5] For better understanding, we take common concepts as an example instead of the specific domain concepts from AK models. A practical AK concept mapping example between two AK domain models can be found in Table A.6 (see Appendix A).

Then the precision (P) and recall (R) rate based on AK concept mapping from x to y ($x \rightarrow y$) can be defined as follows:

$$P_{x \rightarrow y} = \frac{\text{relevant retrieved data}}{\text{retrieved data}} = \frac{CCI_{x \rightarrow y}}{|y|} \tag{4}$$

$$R_{x \rightarrow y} = \frac{\text{relevant retrieved data}}{\text{relevant data}} = \frac{CCI_{x \rightarrow y}}{|x|} \tag{5}$$

As proposed in our previous work (Liang et al., 2008), the precision (MP) and recall (MR) rate of the AK model mapping from S to T (see Fig. 1) can be calculated based on the aggregation of precision and recall rate for individual concept mappings. Some symbols for the calculation of MP and MR are defined: x_i denotes an AK concept of AK model S; y_i denotes a set of AK concepts of AK model T due to the 1:n mapping relationships from x_i to y_i; W_{x_i} denotes the *weight* of AK concept x_i in AK model S (i.e., the percentage of amount of AK instances in AK concept x_i denoted by $|x_i|$ in relation to the whole amount of AK instances in AK model S denoted by $|S|$). $NoC(S)$ is a function to get the number of AK concepts in AK model S. The formulae for the calculation of MP and MR are defined as follows in Formula 7 and 8. Note that although AK repositories based on different AK domain models have different number of AK instances, the number of AK instances in various AK domain models is not a variable in Formula 7 and 8, only the *weight* (calculated by Formula 6) of each AK concept counts, and since this is a unidirectional mapping, only AK concept mappings from model S to T are taken into account in calculation (i.e., $NoC(S)$), and the number of AK concepts in model T has nothing to do with the calculation.

$$W_{x_i} = \frac{|x_i|}{|S|} \tag{6}$$

$$MP_{S \rightarrow T} = \sum_{i=1}^{NoC(S)} (P_{x_i \rightarrow y_i} \times W_{x_i})(x_i \in S, y_i \subset T) \tag{7}$$

$$MR_{S \rightarrow T} = \sum_{i=1}^{NoC(S)} (R_{x_i \rightarrow y_i} \times W_{x_i})(x_i \in S, y_i \subset T) \tag{8}$$

The F-measure (F), the integration of precision and recall rate in IR theory cf. (Baeza-Yates et al., 1999) is defined as:

$$F_{S \rightarrow T} = \frac{2 \times MP_{S \rightarrow T} \times MR_{S \rightarrow T}}{MP_{S \rightarrow T} + MR_{S \rightarrow T}} \tag{9}$$

The meaning of these symbols ($W_x, P, R, MP, MR, F \in [0, 1]$) in the context of AK sharing is described as follows:

- W_x is the percentage of instances belonging to concept x in model S;
- $P_{x \rightarrow y}$ is the percentage of correctly mapped AK instances of concept x relative to all the mapped AK instances in concept y;
- $R_{x \rightarrow y}$ is the percentage of correctly mapped AK instances of concept x relative to all the AK instances of concept x;
- $MP_{S \rightarrow T}$ is the percentage of correctly mapped AK instances in model S to all the mapped AK instances in model T;
- $MR_{S \rightarrow T}$ is the percentage of correctly mapped AK instances in model S to all the AK instances of model S;
- $F_{S \rightarrow T}$ is an integrated criterion to quantify the AK sharing result: the greater the F value is, the better quality AK is shared in from AK repository S to T.

4.2. Concept mappings

In Section 3.1, four general concept mapping relationships (subClassOf, superClassOf, equivalentClass, and noMatchingPair) are defined between AK domain models. The calculation method of AMQPM for the precision (P) and recall (R) rate is based on the semantics of the four concept mapping relationships. As presented

Fig. 3. Instance mapping with equivalentClass relationship.

in Section 4.1, the amount of AK instances belonging to concept x is denoted as $|x|$, i.e., all the AK instances to be mapped in concept x. The calculation method for each concept mapping relationships is presented in the next subsections.

4.2.1. equivalentClass

If concept x (e.g., *Decision Topic*) is the equivalentClass of y (e.g., *Design Issue*), then all the AK instances of concept x (*Decision Topic*) are also the instances of y (*Design Issue*) as shown in Fig. 3. The precision (P) and recall (R) rate can therefore be calculated as:

$$P_{x\,\text{equivalentClass}\,y} = \frac{CCI_{x \to y}}{|y|} = \frac{|x|}{|x|} = 1 \tag{10}$$

$$R_{x\,\text{equivalentClass}\,y} = \frac{CCI_{x \to y}}{|x|} = \frac{|x|}{|x|} = 1 \tag{11}$$

4.2.2. superClassOf

One AK concept x (e.g., *Requirement*) can have multiple superClassOf mapping relationships with a set of AK concepts y ($y = \{y_1, y_2, ..., y_{NoS(x)}\}$, e.g., y_1 is *Functional Requirement*, y_2 is *Non-functional Requirement*, in which $NoS(x)$ is a function to get the number of subclass concepts of x.). AMQPM assumes a random instance classifier, thus this classifier is unable to recognize the correct AK concept to be mapped from multiple candidate subclass concepts for an AK instance, i.e., multiple y plus a *dummy* concept that represents the subclass concept of x that is not covered by y. In such a situation, two instance mapping approaches can be employed for the random instance mapping: a *replicating* or a *splitting* approach. The classifier either maps all the AK instances of x to all the candidate subclass concepts (e.g., y_1, y_2 and *dummy* concept) by *replicating* the instances of x as shown in Fig. 4, or maps part of the AK instances of x to all the candidate subclass concepts by evenly *splitting* the instances of x as illustrated in Fig. 5 (in this example, x is the superclass of two concepts including y_1 and plus a *dummy* concept, so the instances of x are evenly splitted into three parts, and mapped respectively into the three subclasses). Both the *replicating* and *splitting* approaches have their value with respect to the instance mapping quality and mapping results. The *replicating* approach achieves a higher recall rate since more correct AK instances are mapped, while the *splitting* approach returns less incorrect AK instance mappings since less AK instances are mapped.

In AMQPM, we make one implicit assumption both for the *splitting* and the *replicating* approach. We assume that the instances of one AK concept x are evenly distributed over the set of AK concepts y[6] (i.e., the subclasses of concept x) and the dummy concept. This assumption removes the need to assess for each AK concept how the AK instances distribution of x over y looks like, thereby trading off the accuracy of AMQPM with the effort to make an assessment.

The precision (P) and recall (R) rate with superClassOf mapping relationship using *replicating* or *splitting* can be calculated as follows:

4.2.2.1. Replicating approach.

$$\begin{aligned} P_{x\,\text{superClassOf}\,y} &= \frac{CCI_{x \to y}}{|y|} = \frac{(|x|/NoS(x) + 1) \times NoS(x)}{NoS(x) \times |x|} \\ &= \frac{1}{NoS(x) + 1} \end{aligned} \tag{12}$$

$$\begin{aligned} R_{x\,\text{superClassOf}\,y} &= \frac{CCI_{x \to y}}{|x|} = \frac{(|x|/NoS(x) + 1) \times NoS(x)}{|x|} \\ &= \frac{NoS(x)}{NoS(x) + 1} \end{aligned} \tag{13}$$

With the *replicating* approach, $|y|$ (all the AK instances mapped to concept y) is calculated by the product of $Nos(x)$ and $|x|$, since the AK instances to be mapped in concept x (i.e., $|x|$) are replicated $Nos(x)$ times for a full instance mapping to each subclass concept of x. $CCI_{x \to y}$ (the correctly mapped AK instances from x to y) is calculated by the product of $NoS(x)$ and $|x|/(NoS(x)+1)$, since for each subclass concept of x (i.e., y_i, $i = 1 \ldots NoS(x)$), there are only $|x|/(NoS(x)+1)$ AK instances that are correctly mapped (the 1 represents the *dummy* concept) when a random instance classifier is employed. The detailed calculations of $|y|$ and $CCI_{x \to y}$ with $|x|$ itself lead to the calculation of P and R using the *replicating* approach in Formula 12 and 13.

An example of instance mapping of one AK concept x with two subclass concepts (y_1 and y_2, $NoS(x) = 2$) using the *replicating* approach is shown in Fig. 6. The $|x|$ outside the concepts box of y denotes the amount of mapped AK instances, which contributes to the $|y|$. The $1/3 \times |x|$ inside the concepts box of y denotes the amount of correctly mapped AK instances from x to y ($CCI_{x \to y}$). Note, that the correctly mapped AK instances to the *dummy* concept (in dashed box) are not taken into account in correctly mapped AK instances from x to y ($CCI_{x \to y}$) due to its irrelevance to any y concepts.

4.2.2.2. Splitting approach.

The precision (P) and recall (R) rate with superClassOf mapping relationship using *splitting* can be calculated as follows:

$$\begin{aligned} P_{x\,\text{superClassOf}\,y} &= \frac{CCI_{x \to y}}{|y|} = \frac{(|x|/(NoS(x) + 1)^2) \times NoS(x)}{(|x|/NoS(x) + 1) \times NoS(x)} \\ &= \frac{1}{NoS(x) + 1} \end{aligned} \tag{14}$$

$$\begin{aligned} R_{x\,\text{superClassOf}\,y} &= \frac{CCI_{x \to y}}{|x|} = \frac{(|x|/(NoS(x) + 1)^2) \times NoS(x)}{|x|} \\ &= \frac{NoS(x)}{(NoS(x) + 1)^2} \end{aligned} \tag{15}$$

With the *splitting* approach, $|y|$ (all the AK instances mapped to concept y) is calculated by the product of $NoS(x)$ and $|x|/(NoS(x)+1)$, since the AK instances to be mapped in concept x (i.e., $|x|$) are evenly splitted into $NoS(x)+1$ parts (the 1 represents the *dummy* concept) for a partial instance mapping to each subclass concept of x. $CCI_{x \to y}$ (the correctly mapped AK instances from x to y) is calculated by the product of $NoS(x)$ and $|x|/(NoS(x)+1)^2$ since for each subclass concept of x (i.e., y_i, $i = 1 \ldots NoS(x)$), there are only $|x|/(NoS(x)+1)$ AK instances that are mapped, and only $1/(NoS(x)+1)$ of these mapped instances that are correctly mapped when a random instance classifier is employed. The detailed calculations of $|y|$ and $CCI_{x \to y}$ with $|x|$ itself lead to the calculation of P and R using the *splitting* approach in Formula 14 and 15.

An example of instance mapping of one AK concept x with two subclass concepts (y_1 and y_2, $NoS(x) = 2$) using *splitting* approach is shown in Fig. 7. The $1/3 \times |x|$ outside the concepts box of y denotes

---

[6] This is an assumption about the even distribution of AK instances of subclasses y (in destination AK model), which does not violate the explicit AMQPM assumption of uneven distribution of AK instances of x (in source AK model) presented in Table 1.
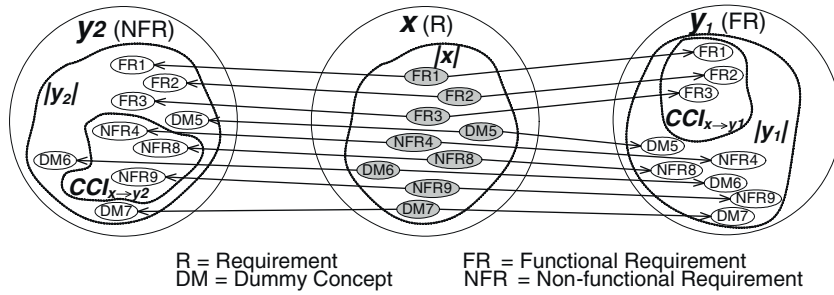
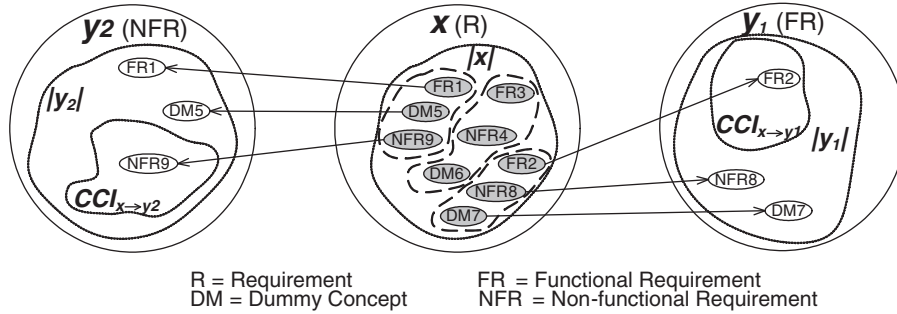**Fig. 4.** Instance mapping with replicating approach.

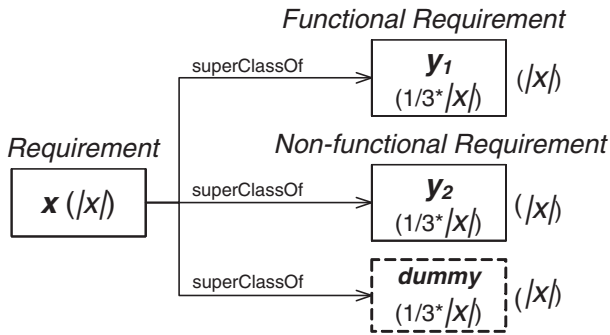**Fig. 5.** Instance mapping with splitting approach.

Fig. 6. Instance mapping with superClassOf relationship using replicating approach.

the amount of mapped AK instances, which contributes to the $|y|$. The $1/3 \times 1/3 \times |x|$ inside the concepts box of y denotes the amount of correctly mapped AK instances from x to y ($CCI_{x \to y}$) since only 1/3 of the mapped AK instances ($1/3 \times |x|$) are correctly mapped AK instances. For the same reason as in the *replicating* approach, the correctly mapped AK instances to *dummy* concept are not taken into account in the correctly mapped AK instances from x to y ($CCI_{x \to y}$) due to its irrelevance to any y concepts.
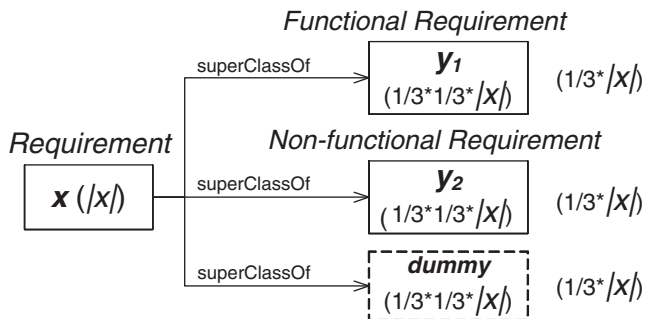

Fig. 7. Instance mapping with superClassOf relationship using splitting approach.

When calculating the precision (P) and recall (R) rate of a concept mapping with the superClassOf relationship, the calculation results based on the two approaches (*replicating* and *splitting*) are combined to get an average value since the two instance mapping approaches are both employed in AK sharing practice. Remark, that the formulae for precision calculation using the *replicating* (i.e., Formula 12) and *splitting* (i.e., Formula 14) approach are the same, which is counter intuitive. This is due to the two assumptions made in AMQPM: (1) the random AK instance classifier, and (2) even distribution of AK instances over the subclasses and dummy concept.

4.2.3. subClassOf

If an AK concept x (e.g., *Data Model*) is a subClassOf y (e.g., *Artifact*), then all the AK instances of concept x (*Data Model*) are also the instances of concept y (*Artifact*) as shown in Fig. 8. The precision (P) and recall (R) rate can be calculated as:

$$P_{x \, \text{subClassOf} \, y} = \frac{CCI_{x \to y}}{|y|} = \frac{|x|}{|x|} = 1 \tag{16}$$

$$R_{x \, \text{subClassOf} \, y} = \frac{CCI_{x \to y}}{|x|} = \frac{|x|}{|x|} = 1 \tag{17}$$

4.2.4. noMatchingPair

If an AK concept x (e.g., *Author*) has noMatchingPair concept in another AK model T, then all the AK instances of concept x (*Author*) cannot be mapped as the AK instances of model T as shown in Fig. 9. The precision (P) rate is not taken into account since no AK instances are mapped ($|y| = 0$), and the recall (R) rate can be calculated as:

$$R_{x \, \text{noMatchingPair} \, y} = \frac{CCI_{x \to y}}{|x|} = \frac{0}{|x|} = 0 \tag{18}$$
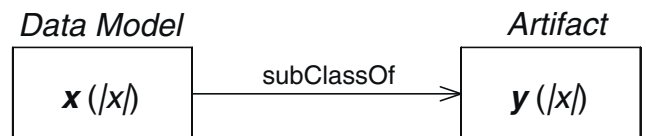

Fig. 8. Instance mapping with subClassOf relationship.

4.3. Using AMQPM

With the calculation formulae and method of AMPQM presented in this section, the predictions of the precision (*P*) and recall (*R*) rate of AK sharing can be calculated by using the AK concept mapping relationships and the *weight* of each AK concept as inputs. The predictions of the *P* and *R* of AK sharing between AK domain models is composed of the *P* and *R* of individual AK concept mappings. With the four concept mapping relationships, detailed calculations of *P* and *R* between individual AK concept mappings are achieved. Note that, one-to-many concept mapping with different mapping relationships is allowed for AK concept mapping. If AK concept *A* has a subClassOf or equivalentClass mapping relationship to AK concept *B*, then all the other concept mapping relationships of *A* will be ignored, since either of these two mapping relationship can achieve a full mapping (i.e., $P = R = 1$) for all the instances belonging to AK concept *A*. In the next section, we present four case studies (three from industry

Fig. 9. Instance mapping with noMatchingPair relationship.
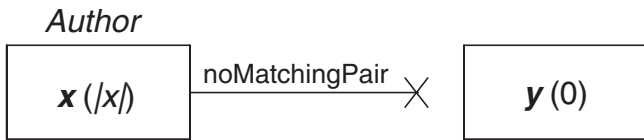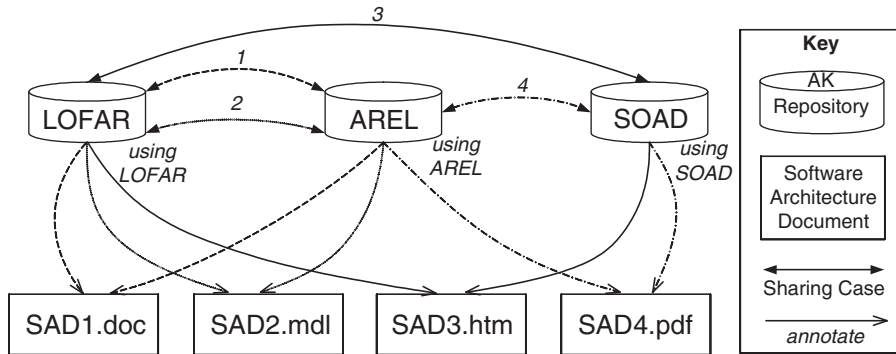
Fig. 10. Relationships between the architecture documents and the AK domain models for AK sharing case studies.
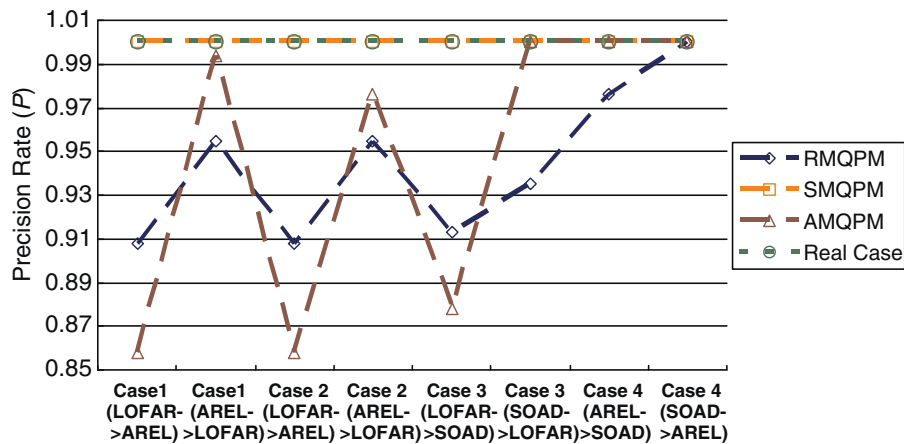
Fig. 11. Comparison among the predictions of precision (*P*) rate of MQPM and real case results based on AK sharing cases.
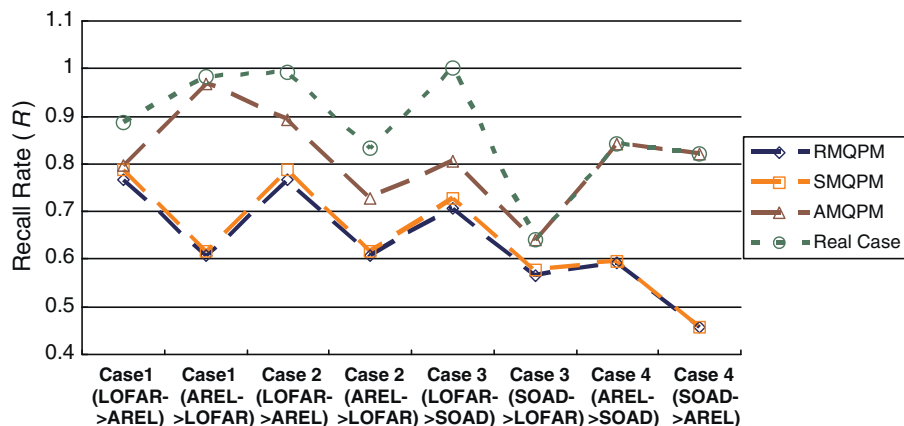
Fig. 12. Comparison among the predictions of recall (*R*) rate of MQPM and real case results based on AK sharing cases.
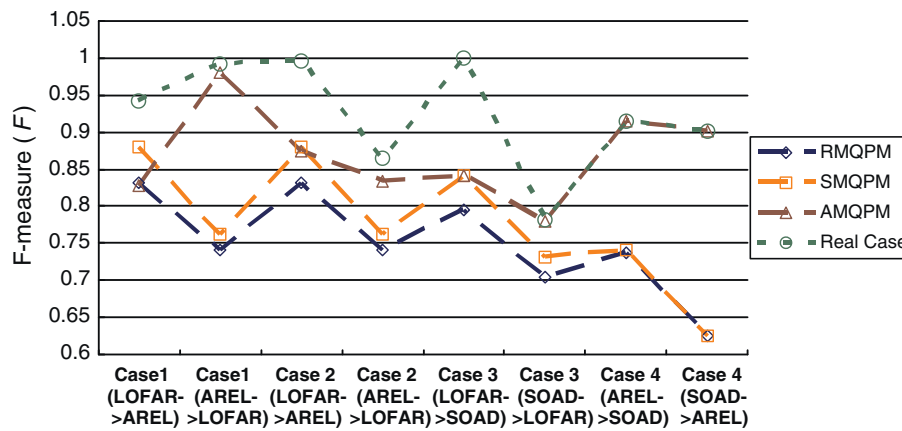
Fig. 13. Comparison among the predictions of *F*-measure (*F*) of MQPM and real case results based on AK sharing cases.

and one from research project), in which the AMQPM has been applied.

5. The experiments

In this section, we experiment with four cases. The predictions with the SMQPM and the RMQPM are calculated based on the formulae and methods presented in our previous work (Liang et al., 2009a, 2008). The AK sharing results of the real AK sharing cases (also using the four cases) are calculated through a manual mapping by domain experts. The prediction by AMQPM is further evaluated (compared) with the predictions by SMQPM, RMQPM, and the real AK sharing results in Section 6.

5.1. Experimental setup

In this experiment, we select three AK domain models: LOFAR (Jansen et al., 2009), AREL (Tang et al., 2007), and SOAD (Zimmermann et al., 2007). They act as the underlying concept models for AK annotation and sharing. In addition, four architecture documents are used to provide the content for AK annotation and sharing. The connection between the AK domain models and architecture documents will be described further on in this section.

The reason for selecting the three AK domain domains is that these AK domain models are mostly proposed and created by ourselves (SOAD is an exception, but we are familiar with this AK domain model). The same holds for the selected software architecture documents. This ensures that we, as domain experts, always make correct AK instance annotations and mappings, which is one of the assumptions specified in Table 1, since we have a good understanding of them. If we had selected other AK domain models and architecture documents, we might make mistakes in AK instance annotations and mappings. The other reason for using the four different documents is that they cover four representative application areas in software design (a mature information system, an emerging service-oriented system, a prototype system in a research project, and a scientific calculation system). We briefly introduce below the basic information of the four architecture documents and related AK sharing case studies.

The first architecture document is from the LOFAR software system. 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. The development of the LOFAR project is undertaken by a consortium of multiple international partners with a long development time of nearly a decade and operational lifetime of at least 20 years. We select an architecture document for a subsystem from the LOFAR project, which is documented using Microsoft Word with 33 pages, 7090 words, and 19 figures. The LOFAR architecture document is denoted as SAD1.doc in this paper. The second architecture document is about a system which was built to monitor car fleet for a major car manufacturer in Australia. The development took 2 calendar years and approximately 10 man years of development efforts to complete. We use the architecture design models in UML and the design document (70 pages) as the basis for this study, and this architecture document is denoted as SAD2.mdl. The third architecture document is from SOAD samples of ADkwik (Schuster and Zimmermann, 2008), which targets to the SOA (service-oriented architectures) decision modeling support developed by IBM Zurich Research. These samples are collected from IBM SOA projects and are documented in a html page, and this architecture document is denoted as SAD3.htm. The fourth case is about an architecture document of a research prototype developed in an large EU integration project with a duration of three years by a five-person team. The document describes architecture design of a process execution engine which orchestrates the execution of the semantic activities according to specified control and data flows. Given the nature of a research prototype, the architecture design includes new features for the process execution engine, but is less concerned with issues such as maintenance. This architecture document is in a PDF format, and denoted as SAD4.pdf. The different characteristics of these four software architecture documents enrich our experiments.

In the four case studies, the domain experts first annotate the same architecture document using two AK domain models (e.g., annotate SAD1.doc using LOFAR and AREL AK domain models). The collection of annotated AK instances belonging to certain AK model acts as an AK repository to be shared. The domain experts then manually map the AK instances from one AK repository to another (e.g., from LOFAR to AREL repository by manually linking the AK instance which has been annotated, and vice verse). The prediction results from SMQPM, RMQPM, AMQPM, and results of manual mapping by domain experts are compared. The manual mapping results by domain experts are used as a benchmark.[7] The relationships between the four architecture documents and the three AK domain models are shown in Fig. 10. For example, the AK sharing case 1 takes place between LOFAR and AREL AK repository which are both constructed by annotating SAD1.doc using LOFAR and AREL AK models respectively. The four AK sharing cases are

---

[7] We assume that domain experts always correctly annotate and map the AK instances. The human factor (different domain experts may have different understanding for annotating and mapping AK instances) is out of the scope of this paper, and will be further investigated in our future work.

represented in the figure by different Sharing Case arrows (e.g., the dotted arrow and the solid arrow).

The detailed experimental steps, as performed by domain experts, are described below:

- Step 1: annotate the same architecture document (e.g., SAD1.doc) into AK instances based on the two AK domain models, which are the underlying conceptual models of AK repositories. The collection of annotated AK instances belonging to certain AK model acts as an AK repository to be shared (mapped). The byproduct of this step is the *weight* of each AK concept (i.e., the percentage of amount of AK instances of an AK concept to the whole amount of AK instances in an AK model). This step can be assisted by the Document Knowledge Client of the Knowledge Architect tool suite for annotating the architecture documents in Microsoft Word (Liang et al., 2009b).
- Step 2: define the concept mapping relationship between the two AK domain models using the four concept mapping relationships presented in Section 3.1. This step can be assisted by the Knowledge Translator of the Knowledge Architect tool suite, which is used to define the concept mapping relationships in OWL (Liang et al., 2009b).
- Step 3: calculate the predictions, i.e., the precision (P), recall (R) rate, and F-measure (F) using SMQPM and RMQPM based on the concept mapping relationships defined in Step 2. The calculation formulae and methods are presented in (Liang et al., 2008, 2009a).
- Step 4: calculate the predictions, i.e., the precision (P), recall (R) rate, and F-measure (F) using the AMQPM that takes as input the concept mapping relationships defined in Step 2 and the *weight* of each AK concept obtained in Step 1. The calculation formulae and methods are presented in Section 4. The calculation of SMQPM, RMQPM, and AMQPM can be done by the functions provided in the Knowledge Translator mentioned in Step 2.
- Step 5: manually map the AK instances annotated in Step 1 from one AK domain model to another, and vice verse, by domain experts. This step can also be assisted by the Document Knowledge Client for mapping (linking) the AK instances between two annotated architecture documents in Microsoft Word (Liang et al., 2009b).
- Step 6: calculate the recall (R) rate of manual mapping between the two AK repositories constructed in Step 1 using Formula 5 from the IR theory (see Section 3.2). This recall rate is realistic since it is based on manual AK sharing, and we assume that the domain experts always correctly annotate and map the AK instances. Based on this assumption, the precision (P) rate of manual mapping is always 1 (100% correct).
- Step 7: compare the predictions (i.e., precision (P), recall (R) rate, and F-measure (F)) by SMQPM, RMQPM, and AMQPM obtained from Step 3 and Step 4, with the manual mapping results by domain experts obtained in Step 6.

Table 2
Predictions and manual mapping results of AK sharing case 1.

			P	R	F
LOFAR → AREL					
Annotatd Document	SAD1.doc	SMQPM	1.000	0.788	0.881
Model Mappings	Table A.6	RMQPM	0.908	0.768	0.832
Instance Annotation	Table A.8	AMQPM	0.858	0.797	0.827
Mapped Instances No.	182	Real Case	1.000	0.888	0.941
AREL → LOFAR					
Annotated Document	SAD1.doc	SMQPM	1.000	0.615	0.762
Model Mappings	Table A.7	RMQPM	0.955	0.606	0.741
Instance Annotation	Table A.9	AMQPM	0.994	0.968	0.981
Mapped Instances No.	182	Real Case	1.000	0.984	0.992

Table 3
Predictions and manual mapping results of AK sharing case 2.

			P	R	F
LOFAR → AREL					
Annotated Document	SAD2.mdl	SMQPM	1.000	0.788	0.881
Model Mappings	Table A.6	RMQPM	0.908	0.768	0.832
Instance Annotation	Table A.10	AMQPM	0.858	0.893	0.875
Mapped Instances No.	219	Real Case	1.000	0.991	0.995
AREL → LOFAR					
Annotated Document	SAD2.mdl	SMQPM	1.000	0.615	0.762
Model Mappings	Table A.7	RMQPM	0.955	0.606	0.741
Instance Annotation	Table A.11	AMQPM	0.976	0.728	0.834
Mapped Instances No.	288	Real Case	1.000	0.760	0.864

5.2. Comparisons of predictions and real sharing results

In this section, we present the predictions with real AK sharing results from four case studies. Due to space limitations, we have selected to present the values of key calculation parameters (e.g., the *weight* of each AK concept) without providing detailed steps of the calculation itself. All the experimental data related to certain AK sharing case is shown in one table, e.g., Table 2 for AK sharing case 1.

5.2.1. AK sharing case 1

As we can see, the AK sharing case 1 (see Table 2) takes place between the LOFAR and AREL AK repository, which are constructed by annotating SAD1.doc using LOFAR and AREL AK domain models. The AK *concept mapping relationships* from LOFAR to AREL and from AREL to LOFAR model are shown in Tables A.6 and A.7 (see Appendix A). The *number of AK instances* and the *weight* of each AK concept in the two AK repository (LOFAR and AREL) obtained by AK instance annotation are presented in Tables A.8 and A.9 (see Appendix A). For the manual mapping of AK sharing case 1 by domain experts, 182 AK instances (i.e., *relevant retrieved data* in the IR theory) are mapped from the LOFAR to AREL AK repository, and the same number of AK instances are mappable from the AREL to LOFAR repository. Based on the concept mapping relationships and the values of the calculation parameters presented in related tables, the *predictions* (precision (P), recall (R) rate, and F-measure (F)) and *manual mapping results* (real case) are calculated and shown in the right part of the Table 2.

5.2.2. AK sharing case 2

The experimental data for the second AK sharing case is presented in Table 3. This sharing case also takes place between the LOFAR and AREL AK repository, which are constructed by annotating the SAD2.mdl document. The difference between the number of AK instances mapped from the LOFAR to AREL AK repository (219) and from the AREL to LOFAR repository (288) is that domain experts annotate the architecture document in different granularity. For example, one AK instance annotated in LOFAR domain model can be annotated as several AK instances in the AREL domain model.

5.2.3. AK sharing case 3

The experimental data for the third AK sharing case is presented in Table 4. Note that the number of AK instances (216) mapped from LOFAR to SOAD AK repository is the same as the number of AK instances in LOFAR repository (216), which means that all the AK instances in LOFAR repository can be manually mapped to SOAD repository. The reason is that the SAD3.htm is documented based on the SOAD AK model, and there is no AK instances in this LOFAR repository (constructed from SAD3.htm) which cannot be mapped into (recognized by) SOAD repository. Consequently, the manual mapping recall rate is 1 (100% retrieved).

Table 4
Predictions and manual mapping results of AK sharing case 3.

			P	R	F
LOFAR → SOAD					
Annotated Document	SAD3.htm	SMQPM	1.000	0.727	0.842
Model Mappings	Table A.12	RMQPM	0.913	0.705	0.795
Instance Annotation	Table A.14	AMQPM	0.878	0.807	0.841
Mapped Instances No.	216	Real Case	1.000	1.000	1.000
SOAD → LOFAR					
Annotated Document	SAD3.htm	SMQPM	1.000	0.576	0.731
Model Mappings	Table A.13	RMQPM	0.935	0.566	0.705
Instance Annotation	Table A.15	AMQPM	1.000	0.640	0.780
Mapped Instances No.	216	Real Case	1.000	0.641	0.781

Table 5
Predictions and manual mapping results of AK sharing case 4.

			P	R	F
AREL → SOAD					
Annotated Document	SAD4.pdf	SMQPM	1.000	0.596	0.747
Model Mappings	Table A.16	RMQPM	0.976	0.591	0.737
Instance Annotation	Table A.18	AMQPM	1.000	0.841	0.914
Mapped Instances No.	127	Real Case	1.000	0.841	0.914
SOAD → AREL					
Annotated Document	SAD4.pdf	SMQPM	1.000	0.456	0.625
Model Mappings	Table A.17	RMQPM	1.000	0.456	0.625
Instance Annotation	Table A.19	AMQPM	1.000	0.820	0.901
Mapped Instances No.	159	Real Case	1.000	0.820	0.901

5.2.4. AK sharing case 4

The experimental data of the fourth AK sharing case is presented in Table 5. This sharing case takes place between the AREL and SOAD AK repository, which are constructed by annotating the SAD4.pdf document. Similar to AK sharing case 2, domain experts annotate the architecture document in different granularity, which results in the difference between the number of AK instances mapped from the AREL to SOAD AK repository (127) and from the SOAD to AREL repository (159). The recall rate prediction of this sharing case (SOAD → AREL) is a bit low (<0.500) since a considerable part of AK concepts from SOAD model have a noMatchingPair mapping relationship to the AK concept from AREL model, which reduces the prediction value of recall rate.

Table A.6
AK concept mappings from LOFAR to AREL AK domain model.

LOFAR	AREL	Mapping relationship
Risk	Risks and Non-risks	subClassOf
Requirement	Functional Requirement	superClassOf
Requirement	Non-functional Requirement	superClassOf
Concern	Motivational Reason	equivalentClass
Concern	Business Environment	superClassOf
Concern	Information System Environment	superClassOf
Concern	Technology Environment	superClassOf
Decision Topic	Design Issue	equivalentClass
Alternative	Alternative Architectual Rationale	subClassOf
Quick Decision	Decision	subClassOf
Specification	–	noMatchingPair
Decision	Decision	equivalentClass
Decision	Architectual Rationale	superClassOf
Decision	Qualitative Rationale	superClassOf
Decision	Quantitative Rationale	superClassOf
Artifact	Design Outcome	equivalentClass
Artifact	Data Model	superClassOf
Artifact	Application Model	superClassOf
Artifact	Technology Model	superClassOf
Artifact Fragment	Design Outcome	subClassOf
Author	–	noMatchingPair

Table A.7
AK concept mappings from AREL to LOFAR AK domain model.

AREL	LOFAR	Mapping relationship
Motivational Reason	Concern	equivalentClass
Functional Requirements	Requirement	subClassOf
Non-functional Requirements	Requirement	subClassOf
Business Environment	Concern	subClassOf
Information System Environment	Concern	subClassOf
Technology Environment	Concern	subClassOf
Design Outcome	Artifact	equivalentClass
Design Outcome	Artifact Fragment	superClassOf
Data Model	Artifact	subClassOf
Application Model	Artifact	subClassOf
Technology Model	Artifact	subClassOf
Decision	Decision	equivalentClass
Decision	Quick Decision	superClassOf
Architectual Rationale	Decision	subClassOf
Qualitative Rationale	Decision	subClassOf
Quantitative Rationale	Decision	subClassOf
Alternative Architectual Rationale	Alternative	superClassOf
Alternative Design	–	noMatchingPair
Alternative Behavior	–	noMatchingPair
Design Issue	Decision Topic	equivalentClass
Design Strengths and Weaknesses	–	noMatchingPair
Tradeoffs	–	noMatchingPair
Risks and Non-risks	Risk	superClassOf
Supporting Information	–	noMatchingPair
Data Viewpoints	–	noMatchingPair
Application Viewpoints	–	noMatchingPair
Technology Viewpoints	–	noMatchingPair
Business Viewpoint	–	noMatchingPair

6. Evaluation and discussions

6.1. Evaluation

In this section, we compare the predictions of AK sharing quality presented in Section 5. The objective is to find out whether the Conjecture 1, 2, and 3 presented in Section 3 still hold or not. The underlying reason of the findings is also discussed to guide users in selecting the appropriate MQPM to predict AK sharing quality.

We visualize the relationships of the comparison among the predictions of MQPMs and the manual mapping results in three line charts. Each line chart depicts one property of the AK sharing quality: Fig. 11 presents the precision (P) rate; Fig. 12 illustrates the recall (R) rate; and Fig. 13 visualizes the F-measure (F). In each figure, the x-coordinate denotes the four AK sharing cases in both mapping directions, and the y-coordinate denotes the value of the predictions. Due to the use of color schema to represent and differentiate the predictions of various MQPMs and the real case, we suggest reading these graphs on-screen or using a color print out.

With these comparison figures, we get three findings:

Table A.8
Manual annotation results of SAD1.doc by LOFAR AK domain model.

LOFAR concept	Number of AK instances	Weight of concept
Risk	3	0.015
Requirement	32	0.156
Concern	57	0.278
Decision Topic	4	0.020
Alternative	0	0.000
Quick Decision	0	0.000
Specification	23	0.112
Decision	73	0.356
Artifact	12	0.059
Artifact Fragment	0	0.000
Author	1	0.005
Sum	205	1.000

Table A.9
Manual annotation results of SAD1.doc by AREL AK domain model.

AREL concept	Number of AK instances	Weight of concept
Motivational Reason	25	0.135
Functional Requirements	28	0.151
Non-functional Requirements	4	0.022
Business Environment	15	0.081
Information System Environment	0	0.000
Technology Environment	17	0.092
Design Outcome	0	0.000
Data Model	0	0.000
Application Model	6	0.032
Technology Model	6	0.032
Architectual Rationale	0	0.000
Qualitative Rationale	0	0.000
Quantitative Rationale	0	0.000
Alternative Architectual Rationale	0	0.000
Alternative Design	0	0.000
Alternative Behavior	0	0.000
Design Issue	4	0.022
Design Strengths and Weaknesses	1	0.005
Tradeoffs	0	0.000
Risks and Non-risks	3	0.016
Decision	73	0.395
Supporting Information	2	0.011
Data Viewpoints	1	0.005
Application Viewpoints	0	0.000
Technology Viewpoints	0	0.000
Business Viewpoint	0	0.000
Sum	185	1.000

Table A.11
Manual annotation results of SAD2.mdl by AREL AK domain model.

AREL concept	Number of AK instances	Weight of concept
Motivational Reason	0	0.000
Functional Requirements	18	0.047
Non-functional Requirements	21	0.055
Business Environment	13	0.034
Information System Environment	0	0.000
Technology Environment	13	0.034
Design Outcome	0	0.000
Data Model	6	0.016
Application Model	33	0.087
Technology Model	18	0.047
Architectual Rationale	38	0.100
Qualitative Rationale	36	0.095
Quantitative Rationale	0	0.000
Alternative Architectual Rationale	16	0.042
Alternative Design	16	0.042
Alternative Behavior	0	0.000
Design Issue	36	0.095
Design Strengths and Weaknesses	38	0.100
Tradeoffs	1	0.003
Risks and Non-risks	2	0.005
Decision	38	0.100
Supporting Information	22	0.058
Data Viewpoints	0	0.000
Application Viewpoints	9	0.024
Technology Viewpoints	1	0.003
Business Viewpoint	4	0.011
Sum	379	1.000

1. AMQPM provides the best prediction of recall (R) rate, which is closer to the R of manual mapping, than SMQPM and RMQPM as shown in Fig. 12.
2. AMQPM provides better F-measure (F) (the integrated AK sharing quality) than SMQPM and RMQPM most of time. The only exceptional case is the AK sharing case 1 from the LOFAR to AREL repository as shown in Fig. 13. According to our experiments, AMQPM is a prediction model which can better trades off the prediction accuracy and effort on practical AK sharing.
3. The relationship of the comparison among the precision (P) rate predictions with different MQPM varies from case to case as shown in Fig. 11. For example, the P of AMQPM is lower than that of RMQPM in AK sharing case 1 (LOFAR → AREL), while in AK sharing case 1 (AREL → LOFAR), the P of AMQPM is greater than that of RMQPM. The P of SMQPM is always 1 (100% correct, same as the P of the real case) due to the assumption of using an intelligent AK instance classifier, something which is not realistic in an automated AK sharing practice.

Based on these three findings, we draw the following conclusions about the conjectures introduced in Section 3. Part of Conjecture 2 for the prediction of recall (R) rate holds according

Table A.12
AK concept mappings from LOFAR to SOAD AK domain model.

LOFAR	SOAD	Mapping relationship
Risk	–	noMatchingPair
Requirement	–	noMatchingPair
Concern	DecisionDrivers	superClassOf
Decision Topic	ProblemStatement	equivalentClass
Alternative	ADAlternative	equivalentClass
Quick Decision	ArchitecturalDecision (AD)	subClassOf
Specification	–	noMatchingPair
Decision	ArchitecturalDecision (AD)	equivalentClass
Artifact	ADOutcome	equivalentClass
Artifact Fragment	ADOutcome	subClassOf
Author	EditorialInfo	superClassOf

to the experimental results ($R_{RMQPM} \leq R_{AMQPM}$). The Conjecture 1 for the prediction of precision (P) rate and the Conjecture 3 for the prediction of F-measure (F) do not hold true since the relationship of the comparison among the P and F predictions with different MQPM varies from case to case. Hence, in a strict sense, no conjectures presented in Section 3 still hold. The revised conjecture for the recall (R) rate based on the experimental results is (no consistent

Table A.10
Manual annotation results of SAD2.mdl by LOFAR AK domain model.

LOFAR concept	Number of AK instances	Weight of concept
Risk	2	0.009
Requirement	39	0.176
Concern	26	0.118
Decision Topic	38	0.172
Alternative	16	0.072
Quick Decision	0	0.000
Specification	2	0.009
Decision	38	0.172
Artifact	3	0.014
Artifact Fragment	57	0.258
Author	0	0.000
Sum	221	1.000

Table A.13
AK concept mappings from SOAD to LOFAR AK domain model.

SOAD	LOFAR	Mapping relationship
Role	–	noMatchingPair
DecisionDriver	Concern	subClassOf
ProblemStatement	Decision Topic	equivalentClass
ADLevel	–	noMatchingPair
ADTopic	–	noMatchingPair
ArchitecturalDecision (AD)	Decision	equivalentClass
Scope	–	noMatchingPair
EditorialInfo	Author	subClassOf
ADAlternative	Alternative	equivalentClass
ADOutcome	Artifact	equivalentClass
ADOutcome	Artifact Fragment	superClassOf
Recommendation	–	noMatchingPair

Table A.14
Manual annotation results of SAD3.htm by LOFAR AK domain model.

LOFAR concept	Number of AK instances	Weight of concept
Risk	0	0.000
Requirement	0	0.000
Concern	43	0.199
Decision Topic	23	0.106
Alternative	81	0.375
Quick Decision	0	0.000
Specification	0	0.000
Decision	23	0.106
Artifact	23	0.106
Artifact Fragment	0	0.000
Author	23	0.106
Sum	216	1.000

Table A.15
Manual annotation results of SAD3.htm by SOAD AK domain model.

SOAD concept	Number of AK instances	Weight of concept
Role	23	0.068
DecisionDriver	43	0.128
ProblemStatement	23	0.068
ADLevel	23	0.068
ADTopic	29	0.086
ArchitecturalDecision (AD)	23	0.068
Scope	23	0.068
EditorialInfo	23	0.068
ADAlternative	81	0.240
Recommendation	23	0.068
ADOutcome	23	0.068
Sum	337	1.000

Table A.16
AK concept mappings from AREL to SOAD AK domain model.

AREL	SOAD	Mapping relationship
Motivational Reason	DecisionDriver	equivalentClass
Functional Requirements	DecisionDriver	subClassOf
Non-functional Requirements	DecisionDriver	subClassOf
Business Environment	DecisionDriver	subClassOf
Information System Environment	DecisionDriver	subClassOf
Technology Environment	DecisionDriver	subClassOf
Design Outcome	ADOutcome	equivalentClass
Data Model	ADOutcome	subClassOf
Application Model	ADOutcome	subClassOf
Technology Model	ADOutcome	subClassOf
Decision	ArchitecturalDecision (AD)	equivalentClass
Architectual Rationale	–	noMatchingPair
Qualitative Rationale	–	noMatchingPair
Quantitative Rationale	–	noMatchingPair
Alternative Architectual Rationale	ADAlternative	superClassOf
Alternative Design	ADAlternative	subClassOf
Alternative Behavior	ADAlternative	subClassOf
Design Issue	ProblemStatement	equivalentClass
Design Strengths and Weaknesses	ADAlternative	subClassOf
Tradeoffs	–	noMatchingPair
Risks and Non-risks	–	noMatchingPair
Supporting Information	–	noMatchingPair
Data Viewpoints	–	noMatchingPair
Application Viewpoints	–	noMatchingPair
Technology Viewpoints	–	noMatchingPair
Business Viewpoint	–	noMatchingPair

Table A.17
AK concept mappings from SOAD to AREL AK domain model.

SOAD	AREL	Mapping relationship
Role	–	noMatchingPair
DecisionDriver	Motivational Reason	equivalentClass
DecisionDriver	Functional Requirements	superClassOf
DecisionDriver	Non-functional Requirements	superClassOf
DecisionDriver	Business Environment	superClassOf
DecisionDriver	Information System Environment	superClassOf
DecisionDriver	Technology Environment	superClassOf
ProblemStatement	Design Issue	equivalentClass
ADLevel	–	noMatchingPair
ADTopic	–	noMatchingPair
ArchitecturalDecision (AD)	Decision	equivalentClass
Scope	–	noMatchingPair
EditorialInfo	–	noMatchingPair
ADAlternative	Alternative Architectual Rationale	subClassOf
ADAlternative	Alternative Design	superClassOf
ADAlternative	Alternative Behavior	superClassOf
ADAlternative	Design Strengths and Weaknesses	superClassOf
ADOutcome	Design Outcome	equivalentClass
ADOutcome	Data Model	superClassOf
ADOutcome	Application Model	superClassOf
ADOutcome	Technology Model	superClassOf
Recommendation	–	noMatchingPair

Table A.18
Manual annotation results of SAD4.pdf by AREL AK domain model.

AREL concept	Number of AK instances	Weight of concept
Motivational Reason	7	0.046
Functional Requirements	17	0.113
Non-functional Requirements	2	0.013
Business Environment	4	0.026
Information System Environment	1	0.007
Technology Environment	14	0.093
Design Outcome	46	0.305
Data Model	1	0.007
Application Model	0	0.000
Technology Model	0	0.000
Architectual Rationale	3	0.020
Qualitative Rationale	0	0.000
Quantitative Rationale	0	0.000
Alternative Architectual Rationale	0	0.000
Alternative Design	5	0.033
Alternative Behavior	1	0.007
Design Issue	17	0.113
Design Strengths and Weaknesses	1	0.007
Tradeoffs	0	0.000
Risks and Non-risks	0	0.000
Decision	11	0.073
Supporting Information	9	0.060
Data Viewpoints	3	0.020
Application Viewpoints	2	0.013
Technology Viewpoints	4	0.026
Business Viewpoint	3	0.020
Sum	151	1.000

Table A.19
Manual annotation results of SAD4.pdf by SOAD AK domain model.

SOAD concept	Number of AK instances	Weight of concept
Role	0	0.000
DecisionDriver	32	0.165
ProblemStatement	1	0.005
ADLevel	1	0.005
ADTopic	9	0.046
ArchitecturalDecision (AD)	24	0.124
Scope	1	0.005
EditorialInfo	23	0.119
ADAlternative	5	0.026
Recommendation	1	0.005
ADOutcome	97	0.500
Sum	194	1.000

conjectures hold for P and F):

$$R_{RMQPM} \leq R_{SMQPM} \leq R_{AMQPM} \leq R_{RealCase} \qquad (19)$$

6.2. Discussions

The underlying reason on the findings presented in Section 6.1 is discussed below in the same numbering as the findings being presented from (1) to (3):

1. The finding (1) is within our expectations since the assumptions of non-even distribution of AK instances in AMQPM is close to the situation in manual AK sharing, in which the *weight* of each AK concept are quite different (e.g., normally there are more *Alternative* AK instances than *Decision* AK instances in a LOFAR AK repository). We can get more accurate recall (R) rate by introducing the *weight* of each AK concept as a variable in the calculation of R.
2. The finding (2) is unexpected since we had thought that the SMQPM would determine the maximum F-measure (F), the integrated AK sharing quality, due to the assumption of using an intelligent AK instance classifier. Actually in practice, the sharing quality contribution to the precision (P) rate by the assumption of intelligent instance classifier does not always pay off to the detriment to the recall (R) rate due to the assumption of even distribution of AK instances (which counteracts to recall (R) rate in certain situations[8]). Consequently, in our experiments, most of the F predictions with AMQPM are greater (better) than those with SMQPM. The only exceptional case, i.e., SMQPM provides better F-measure (F) than AMQPM in AK sharing case 1 (LOFAR → AREL), is due to the domination role of precision (P) rate, which has a positive factor in calculating F-measure (F).
3. In the finding (3), the precision (P) rate does not hold the same relationship of comparison as the recall (R) rate. We think that the P prediction mainly depends on the AK concept mapping relationships from the source to the target model and mapping approaches employed (i.e., *replicating* or *splitting* approach), and the *weight* of each AK concept has little impact to P in prediction. For example, the P predictions with AMQPM from LOFAR to AREL (case 1 and 2) are always lower than the P predictions with RMQPM as shown in Fig. 11 (same situation for the predictions of P from AREL to LOFAR (case 1 and 2), in which the P predictions of RMQPM are always greater than those of AMQPM), and these sharing cases (e.g., LOFAR to AREL case 1 and 2) use the same concept mapping relationships (i.e., from LOFAR to AREL) with different *weight* of each AK concept (case

1 and 2 use different software architecture documents). The implicit assumption (even distribution of AK instances over the subclasses and dummy concept) made for the *splitting* and *replicating* approach in AMQPM may also affect the accuracy of the P prediction with AMQPM as we mentioned in Section 4.2.2. It needs more experiments to investigate above issues, including the impact of the assumption to the prediction accuracy, and why the P predictions with various MQPMs mainly depend on the mappings at the conceptual level, but not the *weight* of each concept at the instance level.

7. Related work

The three mapping quality prediction models, including AMQPM, presented and compared in this paper target to the sharing of formal AK which comes from the architecture documents annotation by AK domain models. Similar approaches exist in the context of knowledge management in the semantic web and data integration in the databases. In this section, we present related work and discuss their implications to our work.

The goal of ontologies is to facilitate knowledge sharing. Many practitioners and researchers from the knowledge engineering area argue that ontology is a key technology for explicit knowledge representation, management, reasoning, and sharing activities. An ontology can be in various formats varying from categories to formalized concept specifications, including domain models, e.g., AK domain models. Ontology mapping provides a common layer from which different ontologies could be accessed and therefore could enable knowledge sharing in semantically sound manners (Kalfoglou and Schorlemmer, 2003). Euzenat has researched on the quality evaluation of the ontology mapping results by introducing two criteria: the precision and recall rate (Euzenat, 2007), which also originate from the IR theory. However, this work focuses on the quality evaluation of the mappings between ontologies at the conceptual level without considering the effect to the instance level. To the best of our knowledge, there is no research work on predicting the mapping/sharing quality of knowledge instances.

Formal knowledge sharing is also based on semantic integration, which provides a common framework for knowledge integration. Noy made a survey on addressing the semantic integration problem (e.g., databases and information integration) in an ontology mapping perspective (Noy, 2004). Three dimensions of ontology mapping are classified: mapping discovery, formal representations of mappings, and reasoning with mappings. The survey focuses on the mapping at the conceptual (ontology) level with (semi-) automatic mapping support, which is suitable for large scale ontology mapping tasks (e.g., there are over 10^6 concepts in the system). On the contrary, in the AK sharing context, there are few AK concepts (around 10–30) in the AK domain models (e.g., LOFAR, AREL and SOAD), but large amount of AK instances to be shared. In our approach, we try to predict the mapping (sharing) quality at the AK instances level based on the manual AK concept mapping by domain experts at the conceptual level, in order to save the effort for mapping AK instances from one model to another.

In our approach, we assume that the mapping at the conceptual level dominates the mapping at the instances level. Our prediction models are based on this assumption. The instance level can also influence the mapping at the conceptual level. (Su and Gulla, 2006) propose an IR approach to using instance information of the concept to enrich the original ontology and calculate the similarities between concepts in two ontologies, to support the semi-automatic mapping of an ontology. This approach is useful for the AK domain model mapping in the formal AK sharing context, in which large

---

[8] The accurate impact caused by the assumption of even distribution of AK instances to recall (R) rate needs to be further investigated in our future work.

amount of AK instances exist. The manual AK instances annotation and mappings by domain experts can improve the mappings at the AK conceptual level.

Integration of heterogeneous databases through the mapping between the local name constants (e.g., value of personal names, company names, etc.) of a database record is also a problem of semantic integration, which requires domain knowledge of the world and the purpose of the user's query. Cohen proposes a logic WHIRL (Cohen, 1998), which can reason about the similarity between name constants by the query results (records in a database could be viewed as being similar to AK instances), as measured using the vector-space model commonly adopted in statistical IR. In this approach, the measures of precision and recall rate from IR theory are also employed as the evaluation criteria. The evaluation results are calculated based on concrete database query results by name constants and used to determine the similarity between the name constants. This approach focuses on the database integration at the record (instance) level without paying attention to the schema (conceptual) level.

In the field of knowledge sharing on heterogeneous web resources, Zhuge proposes a knowledge grid model in three dimensions (location, category, and level) for sharing and managing distributed knowledge resources on the Internet (Zhuge, 2002). This model enables people to conveniently create, represent, store, edit, locate knowledge (instances) and share knowledge with each other in the knowledge grid by a grid operation language called KGOL. The knowledge in the grid is a kind of formal knowledge which has been annotated by the coordinates of the three grid dimensions. This model introduces IS-SIMILAR-TO keyword in KGOL to establish a similar relationship between knowledge instances for the knowledge sharing (e.g., using the existing solution to a problem that is similar to the new problem). However, this knowledge grid model does not provide any criteria nor a method for the evaluation of the knowledge sharing quality.

8. Limitations

The evaluation results we get from the case studies of using SMQPM, RMQPM, and AMQPM are not without limitations. In this section, the foremost limitations of our work are outlined together with potential strategies how they could be addressed.

Weight of AK concept: In the calculation of the prediction by AMQPM, the *weight* of each AK concept is a key parameter, which can be quite different from case to case depending on the AK repository to be shared. In our experiment, we obtain the *weight* of each AK concept through manual annotation of architecture documents by domain experts. However, this is not realistic when the number of AK instances in the AK repository is huge, as this is the primary reason for predicting the AK sharing quality. So how to predict the *weight* of each AK concept in various sharing contexts is a challenge in performing AMQPM. We see several approaches to achieving this. For example, we can randomly select a collection of sample data from the AK repository to be shared, and predict the *weight* of each AK concept with the sample data, since we assume that the *weight* of an AK concept is relatively stable for the software architecture documents from certain domain (e.g., banking system, mission critical system, etc.). Alternatively, we can refer to the values of the AK concept *weight* from previous AK sharing cases, which sharing context (e.g., employed AK domain models and system domain) is similar to the current one, as the empirical data.

User concerns on AK sharing: Different AK consumers (people who query the AK from an AK repository) may have their own preference for a particular part of AK. For example, an AK consumer (scientist) may have special interest on retrieving *Requirements* and *Risks* rather than *Specifications* from the LOFAR repository. In such case, the AK consumer (user) concerns should be taken into account and users should be allowed to customize the AMQPM in order to predict the AK sharing quality based on their own preferences.

Human factors on manual AK annotation and mapping: In the experiments, we assume that domain experts always correctly annotate and map the AK instances from one AK domain model to another, which implies that different domain experts will get the same manual mapping results for AK sharing. However, this is not true in practice since different domain experts may have different understanding for annotating and mapping an AK instance. Different annotations and mappings by architects to the same architecture document can be used to investigate this issue.

Quality factor of architecture documents: When the quality of architecture documents is high (understandability, clarity, etc.), it would be easier to understand the architecture design, and get more AK instances being annotated/captured, which could consequently increase the prediction accuracy based on the *weight* of each AK concept. The four architecture documents used in our case studies are produced by experienced architects with relatively good quality. The architecture documents with different qualities can be used to evaluate this aspect.

9. Conclusions and future work

In this paper, we propose the AMQPM, which is based on our previous work on predicting the formal AK sharing quality. The approach addresses, to some extent, two critical problems that current AK sharing approaches face: the accuracy of AK sharing, and the validity of sharing quality prediction. We illustrate and validate the AMQPM through four AK sharing case studies from industry and research projects, by following the evaluation criteria from IR theory in a query-based scenario for AK sharing. The evaluation results indicate that AMQPM is a prediction model which provides a balance between the integrated AK sharing quality (combining the precision and recall rate of AK sharing) and practical sharing conditions (i.e., variability of AK instances distribution, and non-intelligent AK instance classifier). The distinguishing contribution of AMQPM is that it provides more accurate prediction results (compared to the SMQPM and RMQPM) with acceptable prediction effort. AK practitioners can employ AMQPM model to better predict the AK sharing quality between two AK models in advance before effort is spent on creating AK instances. Note that, although AMQPM provides a better solution to predict AK sharing quality with acceptable prediction effort, the other two prediction models (SMQPM and RMQPM) still have their value, e.g., less effort for prediction, and AK practitioners can select an appropriate prediction model by trading off prediction accuracy and effort in their own AK sharing context (e.g., motivations, domain, and organization, etc.).

Based on our experience from the AK sharing case studies and the limitations of the evaluation results, we see several directions for future work on AK sharing quality prediction: (1) investigate the accuracy of AK sharing quality prediction through estimating the *weights* or reusing the *weights* from empirical data; (2) evaluate the effort involved in performing AMQPM (i.e., annotating the sample architecture documents and calculating the *weight* of each AK concept) as compared to performing AK instance annotation and mapping in practical AK sharing cases; (3) investigate the user concerns on AK sharing quality prediction; (4) investigate the human factors on manual AK annotation to see how and to what extent the human factor affects the AK sharing results; (5) investigate ways to improve the accuracy of AK sharing quality prediction by introducing more comprehensive concept mapping relationships, such as disjointWith, compositionOf, and partOf as we discussed in Section 3.1; (6) investigate and evaluate the quality factor of architecture documents to the prediction results of AK sharing quality.

Acknowledgements

Appendix A.

Tables A.6–A.19

References

Ali-Babar, M., Gorton, I., Kitchenham, B., 2006. A framework for supporting architecture knowledge and rationale management. In: Rationale Management in Software Engineering. Springer, pp. 237–254.

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

Avgeriou, P., Lago, P., Kruchten, P., 2009. Towards using architectural knowledge. ACM SIGSOFT Software Engineering Notes 34 (2), 27–30.

Baeza-Yates, R., Ribeiro-Neto, B., et al., 1999. Modern Information Retrieval. Addison-Wesley Harlow.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L., 2004. OWL web ontology language reference. W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Brickley, D., Guha, R., 2004. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation. http://www.w3.org/TR/rdf-schema/.

Buckland, M., Gey, F., 1994. The relationship between recall and precision. Journal of the American Society for Information Science 45 (1), 12–19.

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., 2006. A web-based tool for managing architectural design decisions. In: Proceedings of the 1st Workshop on SHAring and Reusing architectural Knowledge (SHARK). ACM, pp. 42–49.

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

Cohen, W., 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD). ACM, pp. 201–212.

De Boer, R., 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). Springer LNCS, pp. 197–214.

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

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

Jansen, A., Avgeriou, P., van der Ven, J., 2009. Enriching software architecture documentation. The Journal of Systems & Software 82 (8), 1232–1248.

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). IEEE Computer Society, pp. 109–120.

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). IEEE Computer Society, pp. 44–53.

Kalfoglou, Y., Schorlemmer, M., 2003. Ontology mapping: the state of the art. The 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) , pp. 54–61.

Kruchten, P., Lago, P., van Vliet, H., 2006. Building up and reasoning about architectural knowledge. In: Proceedings of the 2nd International Conference on the Quality of Software Architectures (QoSA). Springer LNCS, pp. 43–58.

Lago, P., 2009. Establishing and managing knowledge sharing networks. In: Software Architecture Knowledge Management: Theory and Practice. Springer, pp. 113–131.

Lago, P., Avgeriou, P., Capilla, R., Kruchten, P., 2008. Wishes and boundaries for a software architecture knowledge community. In: Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE Computer Society, pp. 271–274.

Liang, P., Avgeriou, P., 2009. Tools and technologies for architectural knowledge management. In: Software Architecture Knowledge Management: Theory and Practice. Springer, pp. 91–111.

Liang, P., Jansen, A., Avgeriou, P., 2007. Refinement to griffin core model and model mapping for architectural knowledge sharing. Tech. Rep. RUG-SEARCH-07-L01, University of Groningen, http://www.cs.rug.nl/search/uploads/Publications/liang2007rgc.pdf.

Liang, P., Jansen, A., Avgeriou, P., 2008. Selecting a high-quality central model for sharing architectural knowledge. In: Proceedings of the 8th International Conference on Quality Software (QSIC). IEEE Computer Society, pp. 357–365.

Liang, P., Jansen, A., Avgeriou, P., 2009a. Sharing architecture knowledge through models: quality and cost. The Knowledge Engineering Review 24 (3), 221–240.

Liang, P., Jansen, A., Avgeriou, P., 2010. Collaborative software architecting through knowledge sharing. In: Collaborative Software Engineering. Springer, pp. 343–368.

Liang, P., Jansen, A., Avgeriou, P., 2009b. Knowledge Architect: A tool suite for managing software architecture knowledge. Tech. Rep. RUG-SEARCH-09-L01, University of Groningen, http://www.cs.rug.nl/search/uploads/Publications/liang2009kat.pdf.

Nonaka, I., Takeuchi, H., 1995. The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press.

Noy, N., 2004. Semantic integration: a survey of ontology-based approaches. ACM SIGMOD Record 33 (4), 65–70.

Schuster, N., Zimmermann, O., 2008. ADkwik: Architectural Decision Knowledge Wiki Version 1.2. http://www.alphaworks.ibm.com/tech/adkwik.

Shadbolt, N., Hall, W., Berners-Lee, T., 2006. The semantic web revisited. IEEE Intelligent Systems 21 (3), 96–101.

Su, X., Gulla, J., 2006. An information retrieval approach to ontology mapping. Data & Knowledge Engineering 58 (1), 47–69.

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Babar, M.A., 2010. A comparative study of architecture knowledge management tools. The Journal of Systems & Software 83 (3), 352–370.

Tang, A., Babar, M., 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.

Zhuge, H., 2002. A knowledge grid model and platform for global knowledge sharing. Expert Systems with Applications 22 (4), 313–320.

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

Zimmermann, O., Gschwind, T., Kuster, J., Leymann, F., Schuster, N., 2007. Reusable architectural decision models for enterprise application development. In: Proceedings of the 3rd International Conference on the Quality of Software Architectures (QoSA). Springer LNCS, pp. 157–166.

Dr. Peng Liang is an associate professor in the State Key Lab of Software Engineering (SKLSE), School of Computer at the Wuhan University, China. Between 2007 and 2009, he was a post-doctoral researcher at the Software Engineering and Architecture (SEARCH) research group at the University of Groningen, the Netherlands. His research interests concern the area of software architecture and requirements engineering. He has published more than 40 articles in peer-reviewed international journals, conference and workshop proceedings, and books.

Dr. Anton Jansen is a scientist at ABB corporate research in the software architecture & usability (SARU) group in Västerås, Sweden since 2009. Between 2002 and 2009, he was a member of the Software Engineering and Architecture (SEARCH) research group at the University of Groningen, the Netherlands. He received a master of science degree in computing science in 2002, as well as a Ph.D. in Software Architecture in 2008 from the University of Groningen, the Netherlands. He has worked as a Ph.D. associate (2002–2006) on architectural decisions under supervision of Jan Bosch, and as a post-doctoral researcher (2006–2008) on the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) project GRIFFIN. His research interests are software architecture and architectural knowledge. In his spare time, he likes to play multiplayer computer games, cycling, and reading books.

Dr. Paris Avgeriou is Professor of Software Engineering in the Department of Mathematics and Computing Science, University of Groningen, the Netherlands where he has led the Software Engineering research group since September 2006. Before joining Groningen, he was a post-doctoral Fellow of the European Research Consortium for Informatics and Mathematics (ERCIM). He has participated in a number of national and European research projects directly related to the European industry of Software-intensive systems. He has co-organized several international workshops, mainly at the International Conference on Software Engineering (ICSE). He sits on the editorial board of Springer Transactions on Pattern Languages of Programming. He has published more than 90 peer-reviewed articles in international journals, conference proceedings and books. His research interests lie in the area of software architecture, with strong emphasis on architecture modeling, knowledge, evolution, and patterns.

Dr. Antony Tang is a senior researcher at the VU University Amsterdam, the Netherlands. He spent 3 years as a senior lecturer/senior researcher at Swinburne University of Technology, Australia. Prior to that, he worked in the IT industry as software developer, designer, architect, consultant, and project manager for 24 years. His research interests are in the areas of architecture design reasoning and software development process. He received his double bachelor degrees from the University of Melbourne in 1983 and his Ph.D. degree in IT from Swinburne University of Technology in 2007. He has published over 25 articles in peer-reviewed international journals, conferences and workshops.

Dr. Lai Xu is a lecturer in the software systems research centre at Bournemouth University, UK. Previously she was a senior researcher at SAP research, Switzerland, a research leader of data management group and a senior research scientist at CSIRO ICT Centre, Australia, a post-doctoral researcher at the Institute of Information and computing sciences of the Utrecht University, the Netherlands and the department of computer science of the Free University Amsterdam, the Netherlands. She received her Ph.D. in Computerized Information Systems from the Tilburg University, the Netherlands in 2004. Her research interests include software architecture, service-oriented computing, enterprise systems, and business process management.