Fuzzy Evaluation of Domain Knowledge

Bedir Tekinerdoğan & Mehmet Aksit

TRESE Software Engineering group,
Dept. of Computer Science, University of Twente,
P.O. Box 217, 7500 AE, Enschede, The Netherlands
{bedir| aksit| @cs.utwente.nl

Abstract: Software engineering utilizes selected knowledge sources from which the fundamental concepts for the software solution can be extracted. The quality of the adopted knowledge sources intrinsically defines the quality of the software solution. This quality of knowledge sources is, on the one hand, determined by its objectivity value, and on the other hand, by its relevance value for the given problem. Since the relevance and objectivity values may change due to newly generated knowledge or evolving requirements, crisp decisions in accepting or rejecting the knowledge sources will result in an inappropriate evaluation. We propose to apply fuzzy reasoning in which knowledge sources are assigned fuzzy linguistic quality values to express the quality degrees. This provides a more precise evaluation and can better cope with the evolution of the knowledge sources and the corresponding software requirements. We describe the validation of our proposal with an experimental case study on the evaluation of domain knowledge for the design of transaction systems.

1 Introduction

Software engineering can be essentially seen as a problem solving process that aims to find software solutions for a given problem. The problem is typically initiated by the client's requirement specification and the solution is defined as a software program. Providing a solution for a given problem is not trivial and involves the accumulation and utilization of a huge amount of knowledge. This process of identifying the solution domain knowledge and extracting this knowledge to produce solutions is defined as solution domain analysis [3].

One of the core activities of the solution domain analysis process is the identification of the knowledge sources from which the necessary solution domain concepts will be extracted. To provide quality software it is necessary to elicit the important knowledge sources for a given problem, so that suitable solution abstractions can be identified. As a matter of fact, the quality of the adopted knowledge sources intrinsically defines the quality of the software solution. The corresponding domain knowledge space, though, may be very large and evaluating the knowledge sources may as such be complicated.

This quality of knowledge sources is, on the one hand, determined by its objectivity value, and on the other hand, by its relevance value for the given problem. In practice, evaluation of knowledge domains is uncertain, vague and very often based on the subjective interpretation of the domain engineer. Moreover, the relevance and objectivity values may change due to newly generated knowledge or evolving requirements. Two-valued logic based decisions that either result in accepting or rejecting the knowledge source do not conform with the conceptual evaluation of the human engineer and may easily result in information loss and inappropriate evaluation of the knowledge sources.

We propose to apply fuzzy logic techniques in which knowledge sources are assigned fuzzy linguistic quality values to express the quality degrees. This provides a more precise evaluation and can better cope with the evolution of the knowledge sources and the corresponding problems that are expressed as software requirements. For this purpose we have developed a fuzzy controller to evaluate the relevance and objectivity quality factors of the knowledge sources and determine the abstraction qualities. The fuzzy controller is validated with an experimental case study in which a set of knowledge sources need to be evaluated by a transaction domain expert and a group of novice transaction designers, for two distinct problems.

The remainder of the paper is organized as follows: In section 2, we define the background knowledge on solution domain analysis. Section 3 defines the problem statement in evaluating knowledge domains. Section 4 describes the fuzzy control approach that we have adopted to evaluate knowledge domains. Section 5 provides a case study for the design of atomic transaction systems. Finally, section 6 provides the related work and finally section 7 that provides the conclusions.

2 Solution Domain Analysis

Solution domain analysis aims to identify the right solution domains for the given problems and extract the relevant knowledge from these domains to come up with a feasible solution [3][5]. Fig. 1 represents a conceptual model for illustrating the solution domain analysis process that we apply.

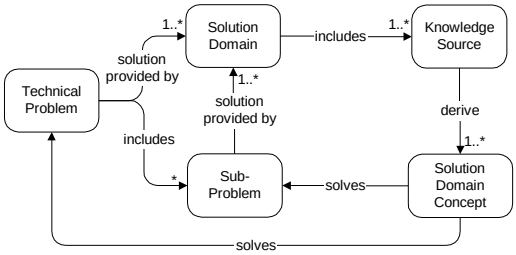


Fig. 1. The basic concepts in solution domain analysis

Hereby, the rounded rectangles represent the concepts and the directed arrows represent the associations between these concepts. The figure typically illustrates the relations between the given problem, the solution domain analysis process and the extracted solution domain concepts that forms the output of the solution domain analysis process. The concept *Technical Problem* represents the problem that needs to be solved and likewise forms an input to the solution domain analysis process. For every *Technical Problem* a solution is provided by one or more *Solution Domains*. The concept *Technical Problem* includes zero or more *Sub-Problems*. The concept *Solution Domain* represents the set of *Knowledge Sources* that provide the concepts for solving the problem. From every *Knowledge Source* one or more *Solution Domain Concepts* can be derived.

For the overall problem and each sub-problem we search for the solution domains that provide the solution abstractions to solve the technical problem. The solution domains for the overall problem are more general than the solution domains for the sub-problems. In addition, each sub-problem may be recursively structured into sub-problems requiring more concrete solution domains on their turn.

Each identified solution domain may cover a wide range of solution domain knowledge sources. These knowledge sources may not all be suitable and vary in quality. For distinguishing and validating the solution domain knowledge sources we basically consider the quality factors of *objectivity* and *relevance*. The objectivity quality factor refers to the solution domain knowledge sources itself, and defines the general acceptance of the knowledge source. Solution domain knowledge that is based on a consensus of a community of experts has a higher objectivity degree than solution domain knowledge that is just under development. The relevance factor refers to the relevance of the solution domain knowledge for solving the identified technical problem.

The relevance of the solution domain knowledge is different from the objectivity quality. A solution domain knowledge entity may have a high degree of objective quality because it is very precisely defined and supported by a community of experts, though, it may not be relevant for solving the identified problem because it addresses different concerns. To be suitable for solving a problem it is required that the solution domain knowledge is both objective and relevant.

The evaluation of a knowledge source based on the quality factors of relevance and objectivity will result in the quality factor that we term *abstraction quality* [1]. The abstraction quality defines the importance of the corresponding knowledge source for extracting solution concepts. The highest abstraction quality is achieved when both the relevance and the objectivity of the knowledge source are high. A high abstraction quality of the knowledge source means that it provides the fundamental concepts for producing a solution with high quality, that is, a solution that fully meets the requirements and which is stable. The relation between the three quality factors may be given in the following empirical formula:

Abstraction Quality (ks) = (Objectivity(ks), (Relevance(ks))

Hereby *Abstraction Quality(), Objectivity()* and *Relevance()* represent functions that define the corresponding quality factors of the argument ks, that stands for solution domain knowledge source. For solving the problem, first the solution domain knowledge with the higher abstraction qualities is utilized. The measure of the objectivity degree can be determined from general knowledge and experiences. The measure for the relevance factor can be determined by considering whether the identified solution domain source matches the goal of the problem. Note, however, that this formula should not be interpreted too strictly and rather be considered as an intuitive and practical aid for prioritizing the identified solution domain knowledge sources.

3 Problem Statement

A simple approach to evaluate the available knowledge sources for a given problem is to provide Boolean variables *objective, relevant*, and *abstraction quality* for each knowledge source and as such assign either the values true or false to it. Typically we could express the corresponding heuristic rule as follows:

> **IF** knowledge source is *RELEVANT* for the problem **AND** *OBJECTIVE*
> **THEN** knowledge source has *ABSTRACTION QUALITY*

If we apply traditional two-valued logic for this rule, then a knowledge source either completely possesses the qualities of relevancy, objectivity and abstraction quality, or it does not. This implies that a knowledge source possesses the abstraction quality only in case it is both considered relevant and objective.

In practice, the process of domain analysis, though, is complex and often related to subjective evaluations, vagueness and uncertainty. Therefore, for a more practical and precise evaluation of the knowledge sources we state that the following three requirements are necessary:

1. Expressing the degree of quality

The evaluation of the objectivity and the relevance value of knowledge domains are basically dependent on the background and expertise of the domain engineer. For a given knowledge source it may be hard to decide whether it completely possesses the quality factors or not. Rather, the domain engineer may decide that it partially possesses the objectivity, relevance and the abstraction quality. Formally, this means that a knowledge source ks of a solution domain SD is mapped to a number in [0,1]. This holds for all the three quality factors:

$$Relevance(ks) : SD \rightarrow [0, 1]$$
$$Objectivity(ks) : SD \rightarrow [0,1]$$
$$Abstraction\ Quality(ks) : (Objectivity(ks),(Relevance(ks)) \rightarrow [0,1]$$

2. Need for linguistic evaluation of knowledge sources

Knowledge sources may be evaluated by assigning numbers to their corresponding quality factors. In practice, however, this is counter to the intuition of the domain engineer, which is rather based on linguistic evaluations, such as fairly, substantially, possibly etc. To cope with this, the domain knowledge evaluation approach must therefore provide means to express quality factors using natural linguistic terms to facilitate the communication about their decisions.

3. Providing means to cope with evolution of knowledge and problems

As a matter of fact, knowledge domains are not static but evolve over time. On the one hand, knowledge domains may become obsolete and less useful for solving a problem. On the other hand, they may become more important, for example, after one has better understood the problem. In addition to the evolution of knowledge, the requirements may evolve as well, and as such the corresponding problem that needs to be solved may change in parallel. Both cases, that is, evolution of knowledge and evolution of problems, may impact the value of the quality factors of the knowledge sources. This implies that the evaluation of the quality of knowledge sources should not be absolute but adaptable to the changing context. Adopting two-valued logic inherently leads to the absolute elimination or acceptance of the knowledge sources and as such fails to cope with this evolution of knowledge and problems appropriately. To address this evolution properly, it is required that knowledge sources are preserved and their quality is adapted to the changing context.

4 Fuzzy Knowledge Source Evaluator

We believe that the evaluation of knowledge domains may be more effectively supported by the use of fuzzy logic and in particular fuzzy control techniques. For this purpose we have designed a fuzzy control system for evaluating domain knowledge, which is illustrated in Fig. 2.

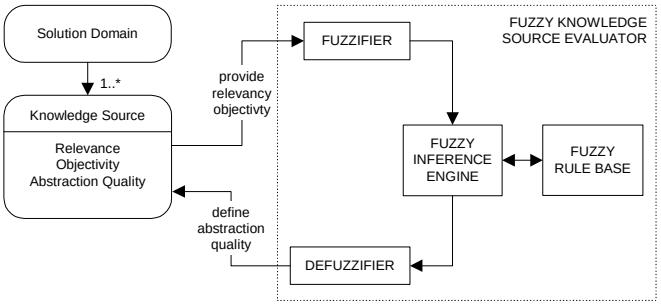


Fig. 2. Fuzzy Controller for defining Abstraction Quality of Knowledge Sources

Note that Fig.2 is an elaboration on the model of Fig.1 in that it provides a fuzzy controller, called *Fuzzy Knowledge Source Evaluator (FKSE)*. The FKSE follows the general structure of fuzzy controllers [8] and consists of the four modules *Fuzzifier*, *Fuzzy Inference Engine*, *Fuzzy Rule Base*, and *Defuzzifier*. The basic inputs for FKSE are the values for the *relevance* and the *objectivity* quality factors of the knowledge source, which are used to compute the value for *abstraction quality* of the knowledge source.

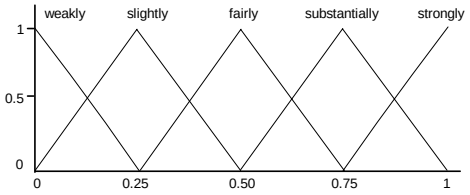


Fig. 3. Shape of membership functions of the linguistic input variables Relevance, Objectivity and the linguistic output variable Abstraction Quality

The membership functions for the input linguistic variables, *relevance* and *objectivity*, as well as the output linguistic variable *abstraction quality* are given in Fig. 3. From an experimental perspective we have applied the triangular membership functions, though, other membership functions may be adopted as well. The optimal membership functions may be determined, for example, after a set of comparative experiments. We will not elaborate on this topic in this paper. For the triangular membership functions of the quality factors we have adopted five linguistic values: *weakly*, *slightly*, *fairly*, *substantially*, and *strongly*.

The evaluation of the relevance and objectivity quality factors are provided to the module *Fuzzifier*. The evaluation may be both expressed numerically, for example as a crisp value in [0..1] or linguistically using one of the five linguistic values. In the first case, the module *Fuzzifier* takes these crisp values as input and maps these into their membership functions and truth-values. The resulted fuzzy set is then provided to the module *Fuzzy Inference Engine*. In the latter case, the *Fuzzifier* provides the fuzzy set directly to the module *Fuzzy Inference Engine*. The module *Fuzzy Inference Engine* uses the fuzzified values to evaluate the control rules that are stored in the *Fuzzy Rule Base*. The adopted meta-rule in this fuzzy rule base is as follows:

IF knowledge source is <relevance value> *RELEVANT* **and** <objectivity value> *OBJECTIVE*
THEN knowledge source has <abstraction quality value> *ABSTRACTION QUALITY*.

This meta-rule can be parameterized with one of the five linguistic values for the quality factors, to define sub-rules. Since, both the *relevance* and *objectivity* quality factors have five linguistic values, we can derive 25 rules from this meta-rule. These rules are represented in Table 1.

Table 1. Rules for inferring abstraction quality of knowledge sources with equal weighting factors for the input variables relevance and objectivity

		Objectivity				
	Abstr. Quality	WE	SL	FA	SU	ST
Relevance	WE	WE	WE	SL	SL	FA
	SL	WE	SL	SL	FA	FA
	FA	SL	SL	FA	FA	SU
	SU	SL	FA	FA	SU	SU
	ST	FA	FA	SU	SU	ST

The first cell of the table, for example, represents the following fuzzy sub-rule:

IF ks is <weakly> *RELEVANT* **and** <weakly> *OBJECTIVE*
THEN ks has <abstraction quality value> *ABSTRACTION QUALITY*.

In Table 1, the relevance quality and objectivity quality have equal weight and as such the table is symmetric along the upper left to right bottom diagonal axis. Alternatively, one may consider the *relevance* quality factor more important than the *objectivity* quality factor. In that case, the *relevance* quality factor must have a larger weight than the *objectivity* quality factor. Based on this assumption we developed the fuzzy rules as given in Table 2. Note that this table is not symmetric anymore.

The module *Fuzzy Inference Engine* executes the fuzzy rules given the input fuzzy set. In our FKSE we have defined the max-min inferencing method although, other inferencing methods may be adopted equally. We do not further discuss this with respect to the scope of the paper.

Table 2. Rules for inferring abstraction quality of knowledge sources with higher weighting factor for relevance factor

		Objectivity				
	Abstr. Quality	WE	SL	FA	SU	ST
Relevance	WE	WE	WE	WE	WE	WE
	SL	WE	SL	SL	FA	FA
	FA	WE	FA	FA	FA	SU
	SU	SL	FA	FA	SU	SU
	ST	FA	FA	SU	SU	ST

After the module *Fuzzy Inference* has executed the fuzzy rules it generates a fuzzy output set, which is then provided to the module *Defuzzifier*. The module *Defuzzifier* converts the provided fuzzy set into a crisp value for the *abstraction quality*. For the defuzzification the centroid method is applied.

Once the knowledge sources have been assigned values for *abstraction quality*, the domain engineer can do an explicit trade-off analysis to decide whether the corresponding knowledge source fulfills the required quality to extract the solution domain concepts. An appropriate evaluation will as such improve the quality of the solution abstractions that are derived from selected knowledge sources.

5 Case Study: Evaluating Transaction Domain Knowledge

In this section, we will illustrate the application of the fuzzy controller using an experimental case study on the evaluation of the knowledge domains for the design of atomic transaction systems. The goal of the case study is to validate the applicability of our knowledge evaluation approach to real design cases, both for domain experts, and designers who are inexperienced in the corresponding domain.

Section 5.1 presents the set of selected knowledge sources that need to be evaluated against two distinct transaction design problems. Section 5.2 illustrates the application of the fuzzy controller by adopting the evaluation of the given knowledge sources by a transaction domain expert. Section 5.3 adopts and discusses the evaluation of the knowledge sources by novice transaction system designers. Finally, in section 5.4 we provide the conclusions of this experimental case study with respect to the earlier defined requirements in section 3.

5.1 Selection of Problem and Knowledge Sources

We can derive a large number of publications on the theory of transaction systems. Informally atomic transactions are characterized by two properties: serializability and recoverability [16]. Serializability means that the concurrent execution of a group of transactions is equivalent to some serial execution of the same set of transactions. Recoverability means that each execution appears to be all or nothing; either it executes successfully to completion or it has no effect on data shared with other transactions.

Many different transaction systems can be designed by extracting various concepts from the transaction literature. Although a common architecture for transaction systems can be derived, transaction systems may nevertheless differ in the selected transaction protocols, such as transaction management, concurrency control protocols, recovery protocols, and data management techniques. Obviously, every transaction system design may need its own dedicated knowledge. The kind of transaction system is basically defined by the corresponding problems. In our experimental case study we have adopted the following two distinct design problems.

Example Problem 1:
Design a transaction system for a distributed system which is based on a two-phase locking concurrency control scheme and a recovery scheme that handles transaction failures based on image logging. The transaction management needs to consider only flat transactions.

Example Problem 2:
Design an advanced transaction system with adaptable transaction properties. In addition to flat transactions, the system must be able to compose nested transactions as well. The choice of the concurrency control and the recovery protocols must be made adaptable according to the performance characteristics of the system.

Note that the first design problem requires the design of a transaction system with fixed properties while the second design problem requires adaptable transaction protocols. To provide a solution for the first problem requires knowledge on the specific protocols, whereas the second problem requires knowledge on a wide range of transaction protocols and additionally it requires knowledge on performance of transaction protocols, and adaptation protocols to switch between the various protocols.

The knowledge sources that needed to be evaluated are presented in Table 3. We have deliberately selected knowledge sources that differ from each other to capture the impact of the fuzzy reasoning process. Differences can be observed with respect to the form of the knowledge source, the date of publication, the location of the publication, the generality of the publication etc.

Table 3. A selected set of knowledge sources for the overall solution domain

KS	*KNOWLEDGE SOURCE*
KS1	Concurrency Control & Recovery in Database Systems [17]
KS2	Atomic Transactions [21]
KS3	An Introduction to Database Systems [18]
KS4	Database Transaction Models for Advanced Applications [19]
KS5	The design and implementation of a distributed transaction system based on atomic data types [25]
KS6	Concurrency Control Performance Modeling: Alternatives and Implications [13]
KS7	Principles of Transaction Processing [16]
KS8	Course Notes of Transaction Design
KS9	Concurrency Control in Advanced Database Applications [15]
KS10	Conference Proceedings on Advanced Transaction Systems and Applications
KS11	On-Line Transactions Tutorial [24]
KS12	Transaction Domain Expert with 15 years of experience
KS13	Design of Adaptable Transaction Systems [23]
KS14	Nested Transactions [22]
KS15	Adaptable Concurrency Control for Atomic Data Types [14]
KS16	A survey of techniques for Synchronization and Recovery in Decentralized Computer Systems [20]

5.2 Evaluation by a Domain Expert

Table 4 represents the evaluation of *relevance* and the *objectivity* of the knowledge sources by a transaction domain expert who has experience in the theory, design and implementation of a wide range of transaction systems.

Table 4. The evaluation of the relevance and objectivity of knowledge sources by a transaction domain expert

KS	RELEVANCE P1	RELEVANCE P2	OBJECTIVITY
1.	ST	SU	SU
2.	FA	FA	ST
3.	SL	WE	FA
4.	WE	ST	SU
5.	SU	SL	ST
6.	WE	ST	ST
7.	SL	WE	FA
8.	SL	WE	SL
9.	FA	ST	SU
10.	WE	FA	SU
11.	WE	WE	FA
12.	ST	ST	ST
13.	FA	SU	FA
14.	FA	SU	ST
15.	WE	ST	SU
16.	SU	WE	WE

During the evaluation all the knowledge sources were actually made available and could be analyzed. The knowledge sources have been evaluated for both problem 1 and problem 2. Since *relevance* is dependent on the problem, the table provides one row for the relevance of each of the two problems. In contrast, *objectivity* is problem independent and as such includes only one row.

Using the input values for the relevance and objectivity quality factors, the fuzzy knowledge evaluator can infer the abstraction quality for each knowledge source. To illustrate the inference mechanism we consider, for example, the inference of the abstraction quality for knowledge source KS4 for problem 1. For this knowledge source the fuzzy relevance value is *weakly*, for problem 1, and its objectivity value is *substantially*. Considering the membership functions as illustrated in Fig. 3, we can derive that for the linguistic variable *relevance*, the fuzzy value *weakly* overlaps with fuzzy value *slightly*. For the linguistic variable *objectivity*, the input value *substantially* overlaps with the values *fairly* and *strongly*. The inference engine will fire all rules but in the end only those rules for which *relevance* is *weakly* or *slightly*, and *objectivity* is *fairly*, *substantially* or *strongly*, will have an impact on the final result. For the experimental case study we adopted the rules as defined in Table 1, that is, *relevance* and *objectivity* of knowledge sources have equal weight. As such, for determining the fuzzy set of the abstraction quality of KS4, the following six rules will have a impact:

1. **IF** ks is <weakly> *RELEVANT* **and** <fairly> *OBJECTIVE*
   THEN ks has <slightly> *ABSTRACTION QUALITY*.
2. **IF** ks is <weakly> *RELEVANT* **and** <substantially> *OBJECTIVE*
   THEN ks has <slightly > *ABSTRACTION QUALITY*.
3. **IF** ks is <weakly> *RELEVANT* **and** <strongly> *OBJECTIVE*
   THEN ks has <substantially > *ABSTRACTION QUALITY*.
4. **IF** ks is <slightly> *RELEVANT* **and** <fairly> *OBJECTIVE*
   THEN ks has <slightly> *ABSTRACTION QUALITY*.
5. **IF** ks is <slightly > *RELEVANT* **and** <substantially> *OBJECTIVE*
   THEN ks has <fairly> *ABSTRACTION QUALITY*.
6. **IF** ks is <slightly> *RELEVANT* **and** <strongly> *OBJECTIVE*
   THEN ks has <fairly> *ABSTRACTION QUALITY*.

The execution of these rules results in the fuzzy set that is illustrated in Fig. 4a.

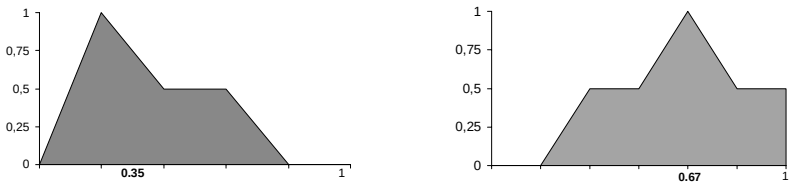


Fig. 4. Fuzzy set of the inferred abstraction qualities for knowledge source 4 for a) problem 1 and b) problem 2

The execution of the fuzzy rules for problem 2 will yield the fuzzy set as given in Fig4b. Note that for problem 2 the set of rules that provide impact on the final result will be different than for problem 1 because of the different value for *relevance*. Using the centroid method the module *Defuzzifier* computes a crisp value of the fuzzy sets, which are 0.35 and 0.67 for problem 1 and problem 2, respectively. These numbers give the software

engineer a practical indication of the quality of the knowledge source. In this case, KS4 has a clearly higher abstraction quality for problem 1 than for problem 2.

Similar to KS4, the fuzzy knowledge evaluator yields the fuzzy sets for the other knowledge sources. Fig 5. shows the defuzzified values of the abstraction qualities of the sixteen knowledge sources, both for problem 1 and problem 2. This figure gives already an hint of the quality of the various knowledge sources and this information is valuable for the software engineer who needs to extract the abstractions from the knowledge sources to develop the solution. The lines in the figure do not have a specific meaning but have been only included for visibility. Let us now take a closer look at Fig. 5 and interpret the resulted abstraction qualities.

A global look at the figure shows, that knowledge sources that only deal with advanced transactions (4, 9, 10), performance modeling (6) and dynamic adaptation (13,15) have got a low abstraction quality for problem 1, that is, the design of a flat transaction system with fixed properties.

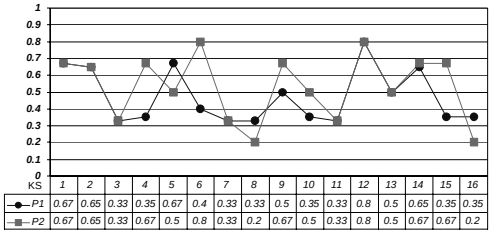
Knowledge source 3 has got a low abstraction quality for both problems. This may be because it is not directly or explicitly related to the subject of transaction systems.

Knowledge source 5 has the highest abstraction quality for problem 1 but a lower value for problem 2. This may be attributed to the fact that it is both a journal paper, leading to a strong objectivity value, and because it is directly related to the design of flat transaction systems.

Knowledge source 6, the journal paper on performance modeling, has the highest abstraction quality for problem 2 but a rather low abstraction quality for problem 1. This may be due to its irrelevance to problem 1. It is not the lowest abstraction quality for problem 1 because it has been evaluated as strongly objective, which indirectly increases the abstraction quality.

Knowledge source 12, the transaction domain expert, has the highest abstraction quality for both problems. This may be explained from the fact that the expert knows both problems well and possesses recent and strongly objective knowledge.

Knowledge source 13, a MSc thesis on the design of adaptable transaction systems, although strongly relevant for problem 2, has not got the highest abstraction value. This may be because it is not recent and the fact that it is a MSc thesis.



KS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P1	0.67	0.65	0.33	0.35	0.67	0.4	0.33	0.33	0.5	0.35	0.33	0.8	0.5	0.65	0.35	0.35
P2	0.67	0.65	0.33	0.67	0.5	0.8	0.33	0.2	0.67	0.5	0.33	0.8	0.5	0.67	0.67	0.2

Fig. 5. Inferred (defuzzified) abstraction qualities of the knowledge sources after the evaluation by a domain expert

The above observations show that the evaluation of the knowledge sources using fuzzy linguistic rules is not ad hoc and reasonably match the intuition. In addition, we can observe that the abstraction qualities have indeed been evaluated differently for both problems, and this difference also provides a sound conceptual interpretation. The knowledge sources have now been more precisely evaluated than in case of a two-valued logic evaluation approach, because every evaluation closely matches the intuition and as such no information is lost.

5.3 Evaluation by Novice Transaction System Designers

To determine the validity and the applicability of the heuristic fuzzy rules for inexperienced domain engineers, we have presented the 16 knowledge sources in Table 3, also to novice transaction system designers. This group consisted of 25 fourth year students of Computer Science at the University of Twente, in the lecture on object-oriented software analysis and design. The students got, a week before, a one and half-hour lecture on solution domain analysis and evaluation of knowledge sources based on the relevance and objectivity quality factors. Similar to the evaluation by the transaction domain expert, the students got the actual knowledge sources during the experimental case study, and evaluated these one by one, separately and independently. The students did not know in advance, though, that they were involved in an experimental case study, but considered the evaluation process more as a practical assignment. The evaluation of the knowledge source took about two hours and consisted again of assigning the five fuzzy values of *weakly*, *slightly*, *fairly*, *substantially*, and *strongly* to the corresponding knowledge sources. We have collected and processed all the results of all students and did not eliminate any.

For every knowledge source we have counted the frequency of the five linguistic fuzzy values and represented these in histograms as illustrated in the Appendix. For example, for the relevance of knowledge source 1

for problem 1, none student assigned the values *weakly* and *slightly*, 2 students assigned the value *fairly*, 12 students thought that it was *substantially*, and 11 students assigned the value *strongly*.

We have implemented an algorithm for deriving the fuzzy sets of a given histogram. Hereby, the average of the 25 fuzzy values in each histogram is taken to yield a new fuzzy set. Consider for example the computation of the average fuzzy set for the relevance of knowledge source 1 (KS1) for problem 1. The histogram of KS1 can be fuzzified by taking the average of 2 *fairly*, 12 *substantially*, and 11 *strongly* fuzzy values. The resulted fuzzy set is shown in Fig. 6. Fuzzy set for the relevance of knowledge source 1 for problem 1, derived from the histogram of the evaluation of novice transaction system designers

.

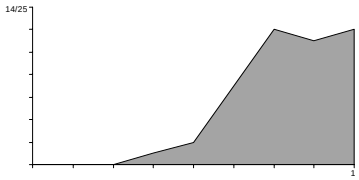


Fig. 6. Fuzzy set for the relevance of knowledge source 1 for problem 1, derived from the histogram of the evaluation of novice transaction system designers

We have done this for each histogram and derived the fuzzy sets of the relevance and objectivity quality factors of each knowledge source. The fuzzy sets have then been provided as an input to the fuzzy knowledge evaluator to provide the abstraction qualities. The results of the defuzzified sets of the abstraction qualities are shown in Table 5.

Table 5. Inferred (Defuzzified) Abstraction Quality of a group of novice transaction designers

KS	DEFUZZIFIED ABSTR. QUAL P1	DEFUZZIFIED ABSTR. QUAL P2
1.	0.67	0.52
2.	0.65	0.62
3.	0.33	0.42
4.	0.35	0.50
5.	0.67	0.50
6.	0.40	0.53
7.	0.33	0.58
8.	0.33	0.44
9.	0.50	0.51
10.	0.35	0.62
11.	0.33	0.49
12.	0.80	0.58
13.	0.50	0.55
14.	0.65	0.53
15.	0.35	0.55
16.	0.35	0.43

To interpret and validate these results we will provide a comparison with the evaluation of the transaction domain expert. Fig. 7 shows a comparison between the evaluation of the inferred evaluation values for the domain expert and the novice group.

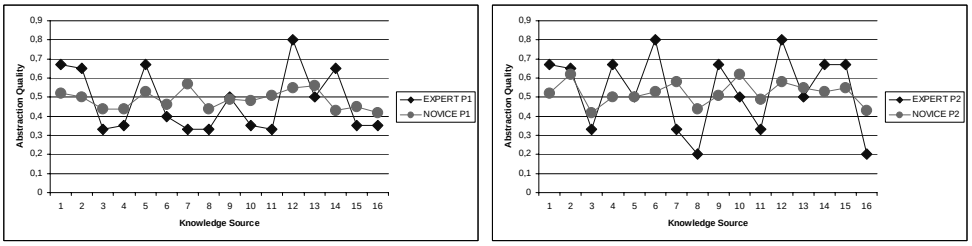


Fig. 7. Comparison of defuzzified abstraction qualities of domain expert and novice transaction designers for (a) problem 1 and (b) problem 2

A first glimpse at Fig. 7 makes clear that the novice group is more careful in their evaluation than the transaction domain expert, because for both problems the difference among the abstraction qualities of the various knowledge sources is less than in the evaluation by the domain expert. This may be attributed to the self-

confidence of the domain expert who has a complete and objective overview of the domain and as such can make more sharp decisions. The students on the other hand may be more careful if they are not sure about their decisions.

If we take a look at the histograms we can observe that there is a reasonable consensus among the 25 students for the evaluation of the knowledge sources. For some knowledge sources (such as 1, 6, 12, 14 and 15) this is more obvious, than others (such as 2, 5, 11, 16). The lack of consensus may show the difficulty of the evaluation, which is influenced by the background and experience of the individual students.

The experience factor of the novice group is obvious for the evaluation of knowledge source 2, a textbook including many formal algorithms and proofs on basically nested transactions. For the relevance quality for problem 1 we can observe that the students do not form a common opinion. This may be attributed due to their lack of experience in interpreting and applying formal algorithms. The values for the objectivity, however, are commonly assigned high values. The reason for this may be that although they cannot precisely evaluate the relevance they have confidence in its scientific objectivity because of the many formulas.

The students were able to distinguish the relevancy of the knowledge sources for both problems. If we compare the defuzzified abstraction qualities with that of the abstraction qualities inferred from the domain expert's evaluation we can observe that the difference in the quality values are reasonably similar. For example, for KS6 the evaluation of the domain expert resulted in the values of 0.4 and 0.8 (Fig. 5), for problem 1 and problem 2, respectively. The evaluation by the novice group resulted in the same increasing order for the abstraction quality of 0.4 and 0.53 (Table 5), for problem 1 and problem 2, respectively. Although, the domain expert's evaluations are more sharp, the application of the fuzzy heuristic rules resulted generally in the same ordering of the abstraction qualities of the knowledge sources. As a matter of fact, this ordering of the knowledge sources is of higher importance than the assigned values.

We may infer many more conclusions from this experimental case study, though, due to space limitations we will not elaborate on it.

5.4 Conclusions of the Overall Case Study

The goal of the case study was to validate the applicability of our knowledge evaluation approach with respect to the identified requirements. In the previous two sections we have respectively discussed the evaluations of the domain expert and the novice group individually. For both cases we have shown that the fuzzy knowledge evaluator is useful and applicable.

First of all, the use of linguistic values for the assessment of the knowledge sources was applied quite straightforward; the evaluation of knowledge sources was done in a rather short period (about 2 hours) and every student could evaluate the knowledge sources independently.

Instead of a classification of accepted and rejected knowledge source as it would result from using two-valued logic, there is a clear graded ordering of the knowledge sources, which likewise provide a more precise evaluation. This is valuable for selecting the knowledge sources for extracting the solution abstractions.

The evaluation of both the domain expert and the novice group illustrates the shift in the values for the abstraction qualities for problem 1 and problem 2. Our approach does not need to eliminate knowledge sources and as such the abstraction qualities may be changed according to the evolution of the context, that is, problems and knowledge.

The reasonable ordering of the inferred abstraction qualities from the domain expert and the novice group show that the fuzzy rules may be successfully applied. The novice group was more careful and had more problems in the evaluation, possibly due to the lack of general scientific experience.

6 Related Work

The solution domain analysis process basically forms the core of our earlier work on the *synthesis-based design process* [11]. Synthesis is a problem solving approach that is applied in mature engineering disciplines such as electrical engineering, mechanical engineering and chemical engineering. The synthesis-based design process consists basically of the sub-processes of technical problem analysis, solution domain analysis and alternative design space analysis. The technical problem analysis phase aims to define the technical problems that have been initiated by the requirement specification. The solution domain analysis aims to find the corresponding solution domains for the given problem and extracts solution domain concepts from these domains. The alternative design space analysis aims to define the space of the alternatives for the given problem and evaluates these against quality criteria. This paper focuses on the solution domain analysis process of the synthesis-based design process. We aim to formalize the whole synthesis process and apply fuzzy control were necessary. The next step

to integrating fuzzy control in the synthesis-based design process will be the evaluation of the extracted solution abstractions.

Fuzzy control has been applied in many different fields but very few of them in the area of software engineering. In [2] fuzzy control is adopted to enhance object-oriented methods by modeling and controlling the design alternatives. The authors maintain that design alternatives during the software development process needs to be preserved to allow further refinements. Two-valued logic is not able to meet this requirement and eliminates alternatives too early.

Several domain analysis processes have been published, e.g. [7], [9], [10] and [5]. Two surveys of various domain analysis can be found in [3] and [12]. In [5], a more recent and extensive up-to-date overview of domain engineering methods is provided. The fuzzy evaluation approach in this paper can be applied to all of these domain analysis methods.

Solving a problem requires first identifying the corresponding solution domains. This may a difficult task if the domain knowledge space is very large. To support the search for the right solution domains we may categorize the domain knowledge [6]. Software engineering applies knowledge of a wide range of application domains, one of them especially is the Computer Science domain, such as programming languages, operating systems, analysis and design methods etc. This type of knowledge has been recently compiled in the so-called Software Engineering Body of Knowledge [4].

7 Conclusion

Software engineering is a problem solving process in which domain knowledge needs to be applied to provide software solutions. The quality of the produced software solution is intrinsically related to the selected knowledge sources. It is therefore necessary to elicit the important knowledge sources for a given problem, so that suitable solution abstractions can be identified. To be useful for solving a problem, the knowledge source must be relevant to the problem and have objective quality, which together define the so-called abstraction quality. A simplistic approach to evaluate domain knowledge is by using two-valued logic, whereby a knowledge source inherently either has the abstraction quality or not. In practice, however, the software engineer may decide that it partially possesses the abstraction quality and need to express these in natural language. In addition, the evaluation of the knowledge sources cannot be absolute but need to change along with the evolution of the available knowledge and the given technical problems. Adopting two-valued logic leads either to the absolute elimination or preservation of the identified knowledge sources.

To cope with these requirements, we have provided a model and an approach for evaluating domain knowledge using fuzzy logic techniques, the so-called fuzzy knowledge source evaluator (FKSE). The FKSE takes as input the relevance and the objectivity (fuzzy) values of a knowledge source and computes the abstraction quality. The inference engine adopts 25 fuzzy heuristic rules.

We have illustrated the application of FKSE in an experimental case study on evaluating domain knowledge for the design of atomic transactions. Thereby, we have provided two distinct problems to a domain expert and a group of novice transaction system designers. From the case study we concluded that applying fuzzy logic techniques in evaluating domain knowledge is of practical use and can support the solution domain analysis process.

Acknowledgements

We would like to thank Pim van den Broek for providing us his Java implementation of the fuzzy algorithms, and Lodewijk Bergmans for helping to extend the Java implementations for deriving fuzzy sets from frequency histograms. Further, we thank the students who have cooperated in the experimental case study on evaluating knowledge sources for the transaction domain. This research is supported by the Dutch Scientific Organization (NWO).

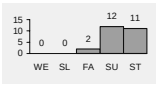
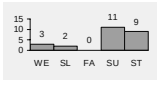
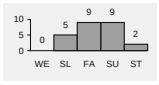
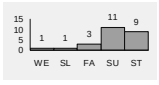
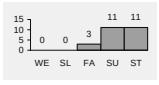
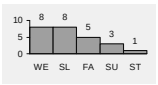
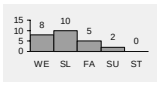
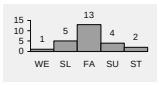
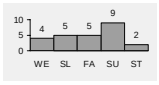
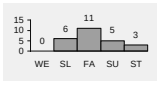
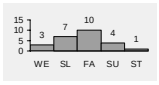
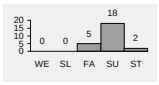
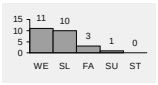
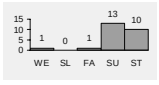
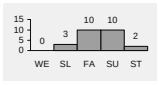
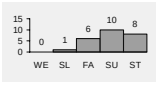
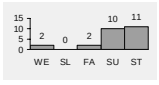
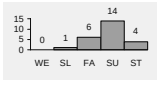
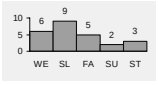
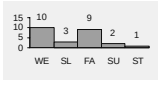
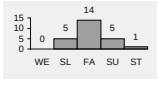
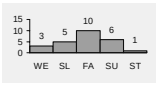
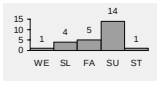
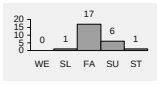
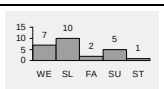
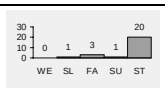
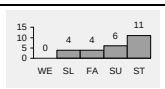
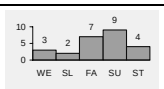
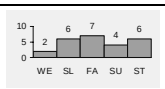
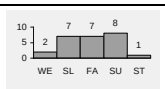
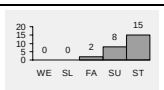
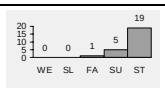
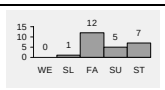
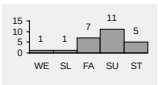
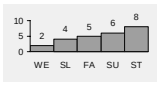
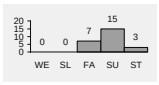
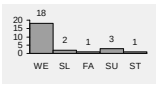
References

1. M. Akşit, Course Notes: Designing Software Architectures, Post-Academic Organization, 2000.
2. M. Aksit & F. Marcelloni. *Deferring Elimination of Design Alternatives in Object-Oriented Methods*, Concurrency Practice and Experience, 2000.
3. G. Arrango. *Domain Analysis Methods*, in: Software Reusability, R. Schäfer, R. Prieto-Diaz & M. Matsumoto (eds.), Ellis Horwood, New York, 1994.
4. P. Bourque, R. Dupuis, A. Abran, J.W. Moore, & L. Tripp. *The Guide to the Software Engineering Body of Knowledge*, Vol. 16, No. 6, pp. 35-45, November/December, 1999.
5. Czarnecki & U. Eisenecker, *Generative Programming*, Addison-Wesley, 2000.
6. R.L. Glass & I. Vessey. *Contemporary Application-Domain Taxonomies*, IEEE Software, Vol. 12, No. 4, July 1995.
7. K. Kang, S.Cohen, J. Hess, W. Nowak & S. Peterson. *Feature-Oriented Domain Analysis (FODA)*, Technical report, Software Engineering Institute, Carnegie Mellon University, 1990.
8. G.J. Klir & B. Yuan. *Fuzzy Sets and Fuzzy Logic:: Theory and Applications*. PrenticeHall, 1995.
9. R. Prieto-Diaz & G. Arrango, G. (Eds.). *Domain Analysis and Software Systems Modeling*. IEEE Computer Society Press, Los Alamitos, California, 1991.
10. M. Simos, D. Creps, C. Klinger, L. Levine & D. Allemang. *Organization Domain Modeling (ODM) Guidebook*, 1996.
11. B.Tekinerdoğan. *Synthesis-Based Software Architecture Design*, PhD thesis, Dept. Of Computer Science, University of Twente, March, 2000.
12. S. Wartik, & R. Prieto-Díaz. *Criteria for Comparing Domain Analysis Approaches*. In International Journal of Software Engineering and Knowledge Engineering, vol. 2, no. 3, pp. 403-431, 1992.

References of the knowledge sources of the case study

13. R. Agrawal, M. Carey & M. Livney. *Concurrency control performance modelling: Alternatives and implications.* ACM Transactions on Database Systems, Vol. 12, No. 4, pp. 609-654, December 1987.
14. M.S. Atkins & M.Y. Coady. *Adaptable Concurrency Control for Atomic Data Types*. ACM Transactions on Computer Systems, Vol. 10, No. 3, pp. 190-225, August 1992.
15. N.S. Barghouti & G.E. Kaiser. *Concurrency Control in Advanced Database Applications*, ACM Computing Surveys, Vol. 23, No. 3, September, 1991.
16. P.A.Bernstein & E. Newcomer. *Principles of Transaction Processing*, Morgan Kaufman Publishers, 1997.
17. P.A. Bernstein, V. Hadzilacos & N. Goodman. *Concurrency Control & Recovery in Database Systems*, Addison Wesley, 1987.
18. C.J. Date. *An Introduction to Database Systems*, Vol. 3, Addison Wesley, 1990.
19. A.K. Elmagarmid (Ed.). *Database Transaction Models for AdvancedApplications Transaction Management in Database Systems*, Morgan Kaufmann Publishers, 1992.
20. Kohler, *A survey of techniques for Synchronization and Recovery in Decentralized Computer Systems*, ACM Computing Surveys, Vol. 13, No. 2, 1981.
21. N. Lynch, M. Merrit, W. Weihl, & A. Fekete, A. *Atomic Transactions*. Morgan Kaufmann Publishers, 1994.
22. J.E.B. Moss. *Nested Transactions : an approach to reliable distributed computing*, Cambridge, MA: MIT Press, 1985.
23. B.Tekinerdoğan. *Design of an Object-Oriented Framework for Atomic Transactions*, MSc. Thesis, Dept. of Computer Science, University of Twente, 1994.
24. B. Tekinerdoğan, Dept. of Computer Science, University of Twente, *On-line transactions Tutorial*, web-document: www.cs.utwente.nl/~bedir/ TransactionsTutorial.
25. Z. Wu, R.J. Stroud, K. Moody & J. Bacon. *The design and implementation of a distributed transaction system based on atomic data types*, Distributed Syst, Engineering, 2, pp. 50-64, 1995.

Appendix - Evaluation of Knowledge Sources by Novice Group

KS	Relevance DP1	Relevance DP2	Objectivity
KS1			
KS2			
KS3			
KS4			
KS5			
KS6			
KS7			
KS8			
KS9			
KS10			
KS11			
KS12			
KS13			
KS14			
KS15			
KS16			