A KNOWLEDGE-BASED SYSTEM FOR RISKS EVALUATION ON SOFTWARE PROJECTS VIABILITY

Javier Andrade, Juan Ares, Rafael García, Santiago Rodríguez and Sonia Suárez
Software Engineering Laboratory, University of A Coruña, Campus de Elviña s/n,15071, A Coruña, Spain

Keywords: Clips, CommonKADS, Knowledge-Based System, Project Management, Risk Management, Viability.

Abstract: In software development, an adequate risks management increases the quality of the final product. However, the importance of this activity is not always acknowledged, and since it requires a high level of experience, it is often not carried out. This article presents a Knowledge-Based System that allows software developers and, or, project managers to evaluate the viability of a project on the basis of its risks: it offers an initial estimation of the risks that must be taken into account and of their impact on the software development project. The proposed system was designed according to the CommonKADS methodology and implemented by means of the Clips tool.

1 INTRODUCTION

Risks are inherent to every activity and, software development, as a particular case, is not an exception. However, whereas some risks can be assumed, other ones must be prevented to avoid their effect on the final purposes of a project.

Software development is a complex process in which many factors can fail. Countless software projects surpass the initial budget and are delivered late or not at all (Somerville, 2006). In order to avoid or alleviate as much as possible this situation, we must identify the existing risks, classify them, and take proactive action to either avoid or manage them to the maximum extent. As such, risk management is a key practice in a successful software projects management (Pressman, 2006).

Determining the viability (or not) of a software development project on the basis of its inherent risks, is a very important task. Nevertheless, it is highly dependent on experience: it is fairly simple with experience, but such it becomes very difficult for the researcher to calibrate the impact of the risk factors on the project without experience (Putnam and Myers, 1997). This is why risk evaluation is often not, or insufficiently, done and the development process, as well as the final product, is affected.

This paper proposes a Knowledge-Based System (KBS) for risks evaluation on software project viability. The system reduces the need for experienced staff and simplifies the estimation of a project's risks based on its characteristics. Section 2 presents the design of this system according to CommonKADS methodology (CommonKADS, 2008), and Section 3 presents its implementation by means of Clips Tools (Riley, 2008). Section 4 presents the main conclusion to be drawn.

2 DESIGN OF THE PROPOSED SYSTEM

The quality of KBS design depends on the knowledge engineer's programming skills, and on his ability to devise, remember, and dynamically update a design specification. This is a difficult task for all but the smallest KBSs.

Difficulties like these can be alleviated by producing representations of the expert's knowledge and of the design specification in the shape of text or diagrams. The best known approach towards the production of such documents is the CommonKADS methodology (CommonKADS, 2008) (Schreiber et al., 2000) (Kingston, 1998) (Valente et al., 1998). It now is the European *de facto* standard for knowledge analysis and knowledge-intensive systems development, and it has been adopted as a whole or has been partly incorporated in existing methods by many major companies in Europe, as well as in the US and Japan (CommonKADS, 2008).

By CommonKADS we elaborate a list of potential components of the model for the KBS, select the adequate template for the task, and construct the initial domain scheme. The last stage is a complete specification of the knowledge model. The following sections describe how each of these activities was carried out.

2.1 List of Potential Model Components

The first task that is tackled by the present KBS belongs to a highly specialized domain within Software Engineering, and is therefore sustained by empirically proven information on how project viability can be considered on the basis of its possible risks.

This task takes into account the fact that a given organisation faces certain risks that have an influence on the viability of a project. These risks are identified a priori and grouped by their original causes, whose presence can be extracted from a series of questions related to the project characteristics. In this way, the presence or absence of certain causes determines the exposition to a given risk (Somerville, 2006) (Pressman, 2006) (Pritchard, 2001).

Also, not all the risks have the same impact in a project, even though this impact can be considered predefined in a concrete organisation (based on the history and experience of the organisation in projects of a similar nature to that of the project that is being considered). By taking into account the impact of the considered risks, as well as the probability that they appear according to the present original causes, we can determine the viability of a project (Putnam and Myers, 1997).

The previous considerations are represented schematically in the ontology of Figure 1, which constitutes a first approach to the domain model. Each section represents the possible risks, and for each risk there is a series of causes that are represented in the shape of questions.

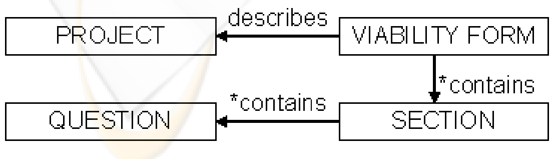


Figure 1: Initial relationships structure.

2.2 Selection of the Task Template

The final purpose of the proposed system is for an organisation to have the possibility to fill out a form with the characteristics of a development project, to inquire as to the project viability, and to obtain a summary of the risks that must be controlled.

In this context, and from the point of view of the task, this is an activity that fits into the category of assessment. These activities are provided with various templates, from which we have selected the one mentioned in (Schreiber et al., 2000).

The main motive for this choice is that the associated inferential structure matches the purpose of the application. A good technique to establish this adequacy to the problem consists in building an annotated inferential structure in which the dynamic roles are annotated or made to correspond with specific elements of the domain. This inferential structure is shown in Figure 2.

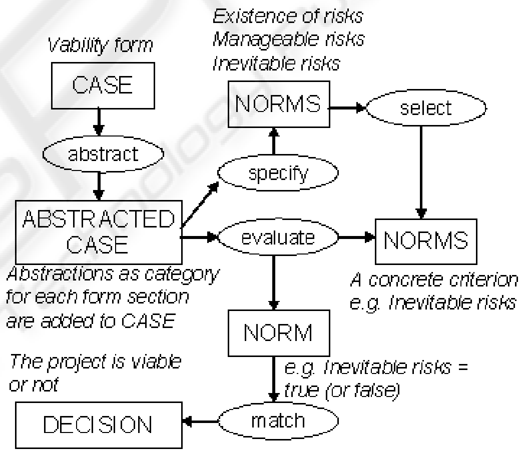

Vability form
CASE
Existence of risks
Manageable risks
Inevitable risks
NORMS
select
abstract
specify
ABSTRACTED CASE
evaluate
NORMS
Abstractions as category for each form section are added to CASE
A concrete criterion e.g. Inevitable risks
NORM
The project is viable or not
e.g. Inevitable risks = true (or false)
DECISION
match

Figure 2: Annotated inferential structure.

2.3 Construction of the Initial Domain Scheme

As recommended in (Schreiber et al., 2000), this activity was carried out in parallel to the previous one. As a result, we obtain a set of domain-specific conceptualizations—shown in Figure 3—and a set of method-specific conceptualizations—shown in Figure 4.

In the problem domain, we have detected three main concept types: Project, Form, and Section. Form represents the initial reasoning case, describes a given Project—which contains an attribute that refers to the viability—, and is composed by a series of sections. Each Section refers to a predetermined risk.

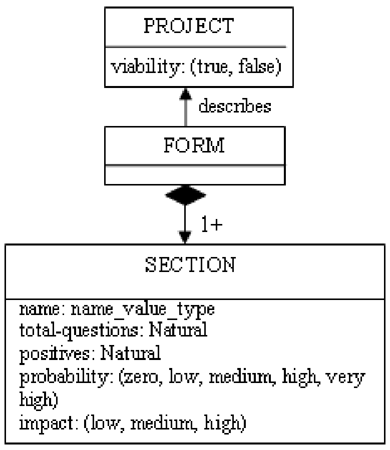


Figure 3: Domain-specific conceptualizations.

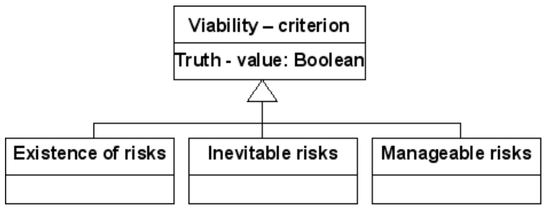


Figure 4: Method-specific conceptualizations.

To reflect this relationship, we use an aggregation between both concepts.

Finally, the Section concept presents five attributes. The first attribute, name, refers to the name of the section, i.e. the name of each risk. The second attribute, total-questions, represents the total number of questions of the Section, i.e. the number of questions that refer to the project and allow us to extract the probability that the considered risk may appear. The third attribute, positives, determines how many of the above questions received a positive response in order to extract the presence level of the risk. The fourth attribute, probability, refers to the probability of appearance of the risk; and, finally, the attribute impact pre-establishes the impact of the risk on a project developed by the organisation that wishes to implant the KBS.

We must keep in mind that, even though a form presents a set of questions and answers, the only relevant data for the KBS' purpose are the attributes of each Section: in order to take the viability decision, the KBS considers the number of questions that received affirmative answers against the total of presented questions, and calculates on that basis the risk probability. This probability is specified as follows:

- If the positive answers represent less than 15%
- of the total, the probability that the risk appears is considered zero.
- If the positive answers represent greater than 15% or equal to 15% and less than 30% of the total, the probability that the risk appears is considered low.
- If the positive answers represent greater than 30% or equal to 30% and less than 70% of the total, the probability that the risk appears is considered medium.
- If the positive answers represent greater than 70% or equal to 70% and less than 85% of the total, the probability that the risk appears is considered high.
- If the positive answers represent between 85% and 100% of the total, the probability that the risk appears is considered very high.

Once it is determined how the domain concepts will be used, we must establish the criteria that will be applied to the data in order to determine the viability of the project. In this concrete case, we have considered three different criteria, each one with a truth-value attribute that determines whether or not the criterion was fulfilled:

- Existence of risks: This criterion determines whether or not risks exist. Even though certain risks may exist, the criterion may respond false if its probability (and hence the project exposure to risks) is small; this would mean that the risks are not relevant to the project viability.
- Inevitable risks: This criterion determines whether there are risks that, due to their characteristics, require an important change of orientation in the project.
- Manageable risks: This criterion determines whether the existing risks, given their exposition, can be manageable with a cost that can be assumed in the project.

The project is considered viable in any of the following situations:

- There are no risks: the "Existence of risks" criterion must take the truth-value false.
- There are risks, but none of them is inevitable and they can all be managed. The "Existence of risks" criterion must take the value true, the "Inevitable risks" criterion must be false, and the "Manageable risks" criterion value must be true.

On the contrary, the project will be considered not viable if:

- There are inevitable risks
- There are risks that are not inevitable but cannot be managed.

The risk existence criterion automatically determines the viability of the project if its value is false; likewise, if the inevitable risk criterion is true, the project is automatically determined to be not viable.

We have established these three criteria to simplify a posterior explanation of the KBS decision.

2.4 Complete Specification of the Knowledge Model

As explained before, the task to be modelled is an instance of the assessment task type. The chosen template shows an inferential structure that is adequate for the purpose of the application, where the inferences present a sufficient level of detail.

The task that must be carried out is decomposed into two subtasks, which means that the "task method" structures the reasoning process in two steps:

- Abstraction: the purpose of this step is to determine the probability that a risk appears in a given project. As mentioned before, this probability can be zero, low, medium, high, or very high.
- Compliance or no compliance with the established criteria by matching the abstractions.

Figure 5 shows the template that was chosen for the modelling.

On the other hand, the knowledge scheme that was finally obtained is shown in figure 6. We can observe that the final domain scheme incorporates three rule types:

- form abstraction: this rule type refers to obtaining the probability of appearance of a risk by using the attributes total-questions and positives.
- viability requirement: the purpose of this rule type is to offer real values to the criteria existence of risks, manageable risks, and inevitable risks.
- project decision rule: this rule expresses the relationship between the different criteria and the final decision taken by the KBS.

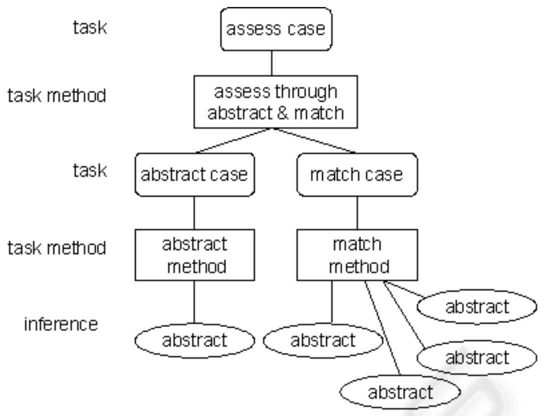


Figure 5: Decomposition of the task.

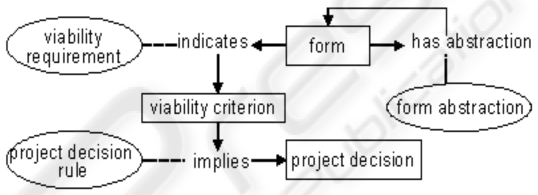


Figure 6: Final knowledge scheme.

3 IMPLEMENTATION OF THE PROPOSED SYSTEM

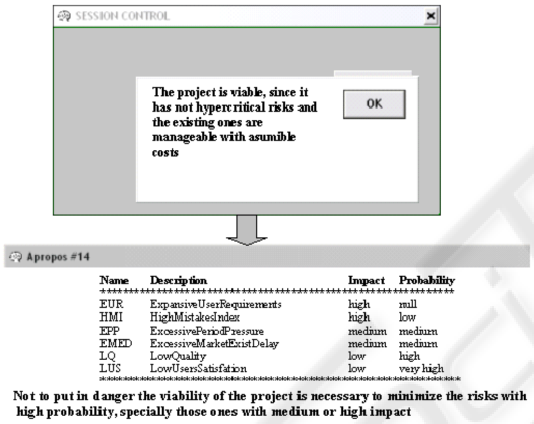
The system was implemented according to the design presented previously and by means of the Clips tool (Riley, 2008). In order to provide the application with modularity and make the development and depuration processes easier, we have defined the following five knowledge bases:

- elements.kb: This knowledge base contains all the definitions of classes, objects, and properties. Since it also contains the operative knowledge of the system, it is the base that must be loaded first.
- rSection.kb: This knowledge base is the first of the rules bases. Its purpose is the dynamic creation of the objects of the Section class on the basis of the corresponding text file.
- rAbs.kb: This knowledge base contains the abstraction rules needed to obtain the probability of a certain risk to appear in a project. As explained in the knowledge model, each risk entails a series of questions that the organisation must answer. Relevant to the system are not the questions themselves but rather the number of affirmative answers with respect to their

total amount. The abstraction of the incidence probability is then calculated.

- rVblty.kb: This knowledge base contains the necessary rules to evaluate the criteria that were established and specified above (existence of risks, manageable risks, and inevitable risks), and uses them to determine whether or not these criteria are fulfilled.
- rDcson: The rules contained in this knowledge base refer to the final assessment decision according to the values of the criteria specified above.

The Clips inference engine is started and the corresponding knowledge bases are loaded. Once the graphic interface is initiated, the inferential process begins. Figure 7 shows the result of the execution of the proposed KBS for a project that is evaluated as viable: there are certain risks, but these can be managed and financially assumed by the project.



The project is viable, since it has not hypercritical risks and the existing ones are manageable with asumible costs

```
Name   Description                     Impact  Probability
*************************************************************
EUR    ExpansiveUserRequirements       high    null
HMI    HighMistakesIndex               high    low
EPP    ExcessivePeriodPressure         medium  medium
EMED   ExcessiveMarketExistDelay       medium  medium
LQ     LowQuality                      low     high
LUS    LowUsersSatisfation             low     very high
*************************************************************
```

Not to put in danger the viability of the project is necessary to minimize the risks with high probability, specially those ones with medium or high impact

Figure 7: An execution example.

The developed KBS is currently being tested in three subjects at the Computer Science School of the University of A Coruña. These subjects are related to Software Engineering concepts, including Software Risks Management. The students are using the system to train the risk management methodology studied in the classroom.

4 CONCLUSIONS

When a software project assumes risks, it is necessary to carry out a risks analysis. Most projects however do this only informally and superficially, if they do. The time that would be invested in such an analysis is worth the while for many reasons: less incidents in the course of the project, increased control of the project evolution, and procurement of solutions before risks actually occur (i.e., problems). Risk analysis can absorb a significant part of the scheduled time, and the proposed KBS gives an automatic support to evaluate the project viability according to its risks, their possible impact on the project, and their probability of occurrence.

REFERENCES

CommonKADS, 2008. *CommonKADS home page*. URL: http://www.commonkads.uva.nl/frameset-commonkads.html. Last access: 18/06/2008

Kingston, J., 1998. *Designing Knowledge Based Systems: The CommonKADS Design Model*. Knowledge-Based Systems, Vol. 11(5-6). Elsevier. 311-319.

Pressman, R.S., 2006. *Process Improvement Software Engineering: A Practicioner's Approach*, McGraw Hill. International Edition.

Pritchard, C., 2001. *Risk Management: Concepts and Guidance*. ESI International.

Putnam, L., Myers, W., 1997. *Implementing Industrial Strength Software*, IEEE Computer Society Press.

Riley, G., 2008. *Clips. A Tool for Building Expert Systems*. URL: http://clipsrules.sourceforge.net/. Last access: 15/07/2008

Schreiber, G., de Hoog, R., Akkermans, H., Anjewierden, A., Shadbolt, N., de Velde, W.V., 2000. *Knowledge Engineering and Management: The CommonKADS Methodology*. The MIT Press.

Sommerville, I., 2006. *Software Engineering*, Addison Wesley.

Valente, A., Breuker, J, van de Velde, W., 1998. *The CommonKADS Library in Perspective*. International Journal of Human-Computer Studies, Vol. 49(4). Elsevier. 391-416.