In *Science of Computer Programming 20*, 1993, 3-50.

[10] L. Chung: Representation and utilization of non-functional requirements for information system design. In *Advanced Information Systems Engineering*, R. Andersen et al., eds. LNCS 498, Springer, Berlin, 1991, 5-30.

[11] J. Mylopoulos, L. Chung, and B. Nixon: Representing and using non-functional requirements: a process-oriented approach. In *IEEE Transactions on Software Engineering 18(6)*, 1992, 483-497.

[12] L. Chung, P. Katalagarianos, M. Marakakis, M. Mertikas, J. Mylopoulos, and Y. Vassiliou: Mapping information systems requirements to designs. In *Database Applications Engineering with DAIDA*, M. Jarke, ed. Research Reports ESPRIT Project 892 DAIDA Vol. 1, Springer, Berlin, 1993, 243-280.

[13] A. Shaw: Reasoning about time in higher level language software. In *IEEE Transactions on Software Engineering 15(7)*, 1989, 875-889.

[14] C.U. Smith and L.G. Williams: Software performance engineering: a case study including performance comparison with design alternatives. In *IEEE Transactions on Software Engineering 19(7)*, 1993, 720-741.

[15] D.N. Card with R.L. Glass: *Measuring Software Design Quality*. Prentice Hall, Englewood Cliffs, 1990.

[16] C. Potts and G. Bruns: Recording the reasons for design decisions. In *Proceedings of the 10th International Conference on Software Engineering* (Singapore, April 11-15), 1988, 418-427.

[17] J. Lee: Extending the Potts and Bruns model for recording design rationale. In *Proceedings of the 13th International Conference on Software Engineering* (Austin, Texas, May 13-17), 1991, 114-125.

[18] K. Poeck, D. Fensel, D. Landes, and J. Angele: Combining KARL and configurable role limiting methods for configuring elevator systems. In *Proceedings of the 8th Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'94* (Banff, Canada, January 30 - February 4), 1994.

evaluation of factors contributing to the satisfaction of non-functional requirements, while, e.g., [11] focus on qualitative aspects which, in our opinion, are more prone to subjective bias. The problem with quantitative measures lies in the difficulty to identify or develop reliable metrics, i.e. metrics which actually allow to predict the effect on the associated requirement. Furthermore, in the MIKE framework, metrics are required which allow to rate requirements on the basis of the design in contrast to metrics which estimate the final code. The importance of such measures is substantiated by the insight that "the greatest potential leverage for software measurement lies in design, not code, analysis" ([15], p. 3).

The model adopted in MIKE for describing design rationale is based on earlier work by [16] and [17] which promote an issue-based style, basically consisting of setting up questions and providing potential answers. In MIKE, a more result-oriented stance is taken and design decisions are linked to requirements more directly. It is not attempted to capture the discourse leading to the preference of one possible solution over others in order to avoid putting too much additional overhead for documentation on the designer.

Currently, work is in progress which aims at gaining more experience with the design framework in MIKE and, particularly, its treatment of non-functional requirements by applying it to complex problems such as, e.g., the configuration of elevator systems [18].

6 References

[1] D. Landes and R. Studer: The design process in MIKE. In *Proceedings of the 8th Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'94* (Banff, Canada, January 30 - February 4), 1994.

[2] D. Landes: Development of knowledge-based systems on the basis of an executable specification. In *Expertensysteme '93*, F. Puppe and A. Günter, eds. Springer, Berlin, 1993, 139-152 (in german).

[3] J. Angele, D. Fensel, D. Landes, S. Neubert, and R. Studer: Model-Based and Incremental Knowledge Engineering: The MIKE Approach. In *Knowledge Oriented Software Design*, J. Cuena, ed. IFIP Transactions A-27, Elsevier, Amsterdam, 1993, 139-168.

[4] D.O. Williams, C. Tomlinson, C.K. Bright, and T. Rajan: The CommonKADS quality viewpoint. Technical report KADSII/T2.2/TR/LR/0040/1.0, Lloyd's Register, London, 1992.

[5] S.E. Keller, L.G. Kahn, and R.B. Panara: Specifying software quality requirements with metrics. In *System and Software Requirements Engineering*, R.H. Thayer and M. Dorfman, eds. IEEE Computer Society Press, Los Alamitos, 1990, 145-163.

[6] G. Guida and G. Mauri: Evaluating performance and quality of knowledge-based systems: foundation and methodology. In *IEEE Transactions on Knowledge and Data Engineering 5(2)*, 1993, 204-224.

[7] B.W. Boehm: A spiral model of software development and enhancement. In *IEEE Computer 21*, 1988, 61-72.

[8] D. Landes: DesignKARL - A language for the design of knowledge-based systems. In *Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering SEKE'94* (Jurmala, Latvia, June 20-23), 1994.

[9] A. Dardenne, A. van Lamsweerde, and S. Fickas: Goal-directed requirements acquisition.

product which are affected by the design decision to be evaluated.

The question whether an analytical evaluation of design decisions or an evaluation by prototyping should be preferred in a particular context depends on the nature of the involved requirements: some of the requirements are amenable to quantitative measurement while others can be evaluated more easily by means of prototyping. Among the non-functional requirements that are particularly addressed in MIKE, maintainability and portability are members of the first category, while understandability, reliability and environmental requirements belong to the second. Both approaches are possible to evaluate efficiency aspects. Clearly, the decision whether the selection of a design alternative should be based on the evaluation of a prototype or on a quantitative evaluation also depends on factors, such as, e.g., the availability of useful measures, the required effort for the computation of measures vs. the required effort for constructing a prototype, etc. If neither type of evaluation can be used, the selection of a design alternative has to be based on qualitative considerations similar as proposed in, e.g., [11].

5 Discussion

The main contribution of the MIKE design approach is the explicit integration of the treatment of non-functional requirements into KBS development which has several benefits. First of all, the design process itself is made more transparent since the record of design decisions and their rationale helps the designer to avoid repeating erroneous design decisions as well as inadvertently undoing earlier design decisions simply because the reasons have got lost why they initially had been made. Furthermore, design decisions and rationale are described in a formalism [8] which is amenable to automated processing in order to provide intelligent support to the designer in a similar fashion as envisioned in the mapping assistant in [12] (this type of support has not been addressed in MIKE so far). Furthermore, the explicit connection between requirements and affected portions of the design product ensures traceability of requirements. Similarly, this also holds for functional requirements since design decisions link two versions of the design product such that parts of the final design can be traced back to corresponding sections of the model of expertise and even further back to parts of a semiformal description of expertise [3]. The explicit description of design decisions and their conjunction to requirements is also indispensable if not only parts of the specification (i.e. generic, implementation-independent descriptions of problem-solving methods), but also corresponding designs are to be reused since it is then much easier to determine which design decisions are still applicable in a new context and which others must be treated differently.

The ideas concerning the treatment of non-functional requirements are considerably influenced by similar work in the context of information systems design, notably the DAIDA project ([10], [11], [12]). The major difference between this work and the approach taken in MIKE lies in the fact that different types of requirements are considered most relevant in the two domains and that MIKE puts more emphasis on the quantitative

ment, i.e. the portion of the design product to which it refers, may be made more specific, for instance, the aim for efficiency of the complete system may be reduced to efficiency of a crucial subtask (or several crucial subtasks). Usually, goals can be decomposed in several ways, i.e. there are several ways to reach a goal. Therefore, decomposition of goals generally results in an AND/OR tree.

Besides the decomposition relationship, additional types of relationships may exist between goals. Since goals may be decomposed in various ways, the designer has to select one of the available alternatives which seems to be most suitable in the given context. The motivation for preferring one alternative over another can be expressed by means of *preferences*. Preferences indicate, e.g., according to which criteria an alternative is preferred over another. In some cases, the selection of an alternative may be due to the fact that some potential alternatives are excluded because they are incompatible with previous design activities or, conversely, implied by earlier activities. These circumstances can be expressed as *implications* or *exclusions* between subgoals. Furthermore, the basic quality of interdependencies between requirements can be described by means of *correlations*. Correlations indicate if actions taken to satisfy one requirement positively or negatively affect the fulfilment of another requirement. For instance, efficiency with respect to processing time and efficiency with respect to storage space are in general inversely *correlated*. Often, a design alternative is chosen tentatively and has to be withdrawn at a later stage of the design process when additional information has been gained. This can be expressed as a *revision* to indicate which preferences are now superseded.

4 Evaluation of Non-Functional Requirements

The choices between alternative ways to reach a goal are crucial steps in the design process which should be grounded on a firm basis. Ideally, the selection of a design alternative is based on a reliable quantitative estimate of what can be achieved with each of the available alternatives. To that end, quantitative measures for the involved non-functional requirements are necessary. In the context of evaluating, e.g., time efficiency, estimates of the algorithmic complexity of the algorithms employed or execution time estimates (cf., e.g., [13], [14]) may give an indication which alternative to choose. Maintainability can be linked, e.g., to complexity: [15] show in a case study how complexity may be used to predict, e.g., the effort required for maintaining a software system. The complexity of a modular software design is determined by the interconnectivity of modules and the internal complexity of individual modules, i.e. the average number of “decisions” in the modules. Individual measures may then be combined using a scheme similar to the one presented in [6].

Alternatively, a tentative decision for one of the alternatives may be substantiated by running the current description of the design product as a hybrid prototype. The prototype then combines parts of the (executable) model of expertise with parts which have already been mapped to the target language, in particular those portions of the design

requirements while the other one concentrates on non-functional requirements. Usually, acquisition of functional requirements precedes acquisition of non-functional requirements. The identification or clarification of non-functional requirements can be facilitated by running a prototype. Such a prototype is available in MIKE due to the fact that the model of expertise as the result of knowledge acquisition, i.e. the "document" capturing the functional requirements, is described using an executable formalism. During knowledge acquisition, this prototype first of all serves the purpose of evaluating the developed model with respect to functional requirements whereas focus shifts to non-functional requirements in the design phase.

3 Treatment of Non-Functional Requirements

Since the formalism for specifying the model of expertise is executable, an operational, though usually inefficient, solution to the considered task is already available after each cycle through the knowledge acquisition phase (a life-cycle model similar to Boehm's spiral model [7] is adopted in MIKE). Further development is driven by non-functional requirements since these have not been addressed during knowledge acquisition. Notably, the efficiency of the solution has to be improved by developing appropriate algorithms and data structures or the solution has to be integrated in a previously determined hardware or software environment.

To that end, four basic types of design decisions have been identified ([1], [8]): *realize*, which refines parts of the model by introducing algorithms and data structures, *structure*, which indicates the application of structuring primitives to decompose the overall model to smaller, largely self-contained portions or externally visible modifications of such portions, *introduce*, which refer to portions of the model which appear without being a refinement of previously existing parts of the model, and *abandon*, which indicates that portions of the model are no longer needed and, thus, removed.

The motivation for performing an activity belonging to one of the four categories of design decisions lies in the aim to contribute to the satisfaction of a particular non-functional requirement. Thus, we take a goal-oriented viewpoint on the design of KBS in a similar way as it is done in [9] for requirements engineering and in [10], [11], and [12] for information systems design, i.e. non-functional requirements are viewed as goals to be achieved. In general, top-level requirements will be fairly unspecific, e.g., as a top level goal it might be expressed that "the system should be efficient enough to respond to a query within less than 10 seconds". Unspecific means that it is not immediately clear how such a requirement can be met or which portion of the system is affected by the requirement in the first place. Therefore, goals are gradually decomposed into elementary subgoals which can be met by performing a collection of design decisions. Two aspects may be used for achieving a decomposition. On the one hand, a requirement (i.e. a goal) may be reduced to a collection of more basic requirements. The global aim for efficiency may, e.g., be reduced to efficiency with respect to processing time or efficiency with respect to storage needs. On the other hand, the scope of the require-

In the following, non-functional requirements that turned out to be relevant for the development of knowledge-based systems will be briefly characterised before their treatment in the MIKE framework is sketched. In particular, the model for describing the rationale behind design decisions and the role of non-functional requirements in this model will be outlined. Finally, the approach taken in MIKE is discussed and put in relation to similar work.

2 Non-Functional Requirements and KBS

The quality of KBS has become a research topic only recently. [4] address this issue in the context of the CommonKADS framework for building KBS and provide a taxonomy of factors which determine the overall quality of the system. The top levels of this taxonomy very much resemble the ones that have been proposed for "conventional" systems (cf., e.g., [5]). A different taxonomy proposed by [6] is basically characterised by the distinction of factors affecting the behaviour of the system from factors associated with its ontology. In both proposals, no explicit distinction between functional and non-functional requirements is made. These taxonomies indicate which types of requirements must in principle be considered for being able to build a high-quality KBS. On the basis of these taxonomies, important non-functional requirements to be addressed in MIKE currently include the following (the list is open for extension in a later stage of the MIKE project):

- *efficiency*, i.e. the aspect whether the system can fulfil its task with the available resources of processing time and storage space,
- *maintainability*, i.e. the question whether the system can easily be adapted to changes in the environment or in case of detected insufficiencies,
- *understandability*, i.e. the ease of grasping how the system arrives at a solution,
- *reliability* in the sense of how robustly the system reacts to wrong or incomplete case descriptions, missing knowledge, etc.,
- *portability*, i.e. how easily parts of the system can be transferred to other hardware or software environments,
- requirements resulting from a fixed hardware / software *environment* or system *architecture*.

As these requirements refer to the realisation of the KBS, they are in many cases not yet clear in an early stage of the project, but evolve gradually when the functionality of the system has been clarified. Furthermore, experts, i.e. persons that know how to solve the task under consideration, are deeply involved in the formulation and evaluation of functional requirements. However, these experts are experts in their domain, but usually not in computer issues. Therefore, non-functional requirements usually cannot be acquired from domain experts together with functional requirements, but rather must be acquired separately at a different stage of development. Thus, there are basically two development steps concerned with requirements acquisition, namely one focusing on functional

Addressing Non-Functional Requirements in the Development of Knowledge-Based Systems

Dieter Landes

Institut für Angewandte Informatik und Formale Beschreibungsverfahren
Universität Karlsruhe, D-76128 Karlsruhe, Germany
e-mail: landes@aifb.uni-karlsruhe.de

We argue that non-functional requirements are the driving force behind the design decisions in the MIKE approach to the development of knowledge-based systems. We outline the model adopted in MIKE to record the rationale of design decisions by explicitly linking them to non-functional requirements and indicate how the evaluation of the design wrt. the posed requirements can be integrated into the development process.

1 Introduction

The main goal of a knowledge-based system (KBS) or, specifically, an expert system is the ability to adequately perform a task which normally requires expert knowledge to be carried out. For the type of problems considered, expertise does not only comprise knowledge about the application domain, but also knowledge about how the problem in question can be solved. Therefore, the specification of the system must, in contrast to specifications in conventional software engineering, not only address *what* functionality the system must provide, but also has to pay attention to *how* the required functionality can be exhibited, i.e. which steps have to be performed in order to solve the given problem. Yet, aspects concerning the *realization* of the required functionality are still at a different level. Therefore, issues of realization can be neglected as long as the focus is on identifying the required functionality and may rather be addressed at a different stage of development (cf. [1], [2]).

MIKE (Model-based Incremental Knowledge Engineering) [3] is a framework for the development of knowledge-based systems which tries to integrate the benefits of life-cycle models, prototyping, and formal specification techniques. In particular, the above-mentioned types of considerations are carried out in different phases of the MIKE life-cycle: while functional requirements are addressed in the knowledge acquisition phase, aspects concerning the appropriate computational realization of the required functionality are emphasized in the design phase. This means in turn that decisions in the design phase are primarily motivated by non-functional requirements, or conversely, non-functional requirements constitute the justifications of design decisions. In order to improve the transparency of the design process, a model for describing the design rationale has been integrated in MIKE [1]. By the explicit consideration of non-functional requirements, MIKE differs from many other knowledge engineering approaches which mostly focus on functional aspects while non-functional issues only play a minor, if any, role. Furthermore, MIKE explicitly ties non-functional requirements to those portions of the system design they affect.