

FEATURE ARTICLE

Armin Eberlein, Dale Fukami,
& Wonh Chieh Lam

Using an Expert System

These guys show us the benefit of using an expert system for applying advanced techniques to system design and the steps they took in discovering the advantage of using computer-aided software tools to manage complex system requirements.



The development of increasingly complex systems has become a major challenge for today's high-tech companies. Developers admit that the most problematic task lies in understanding the requirements and correctly transforming them into code. [1] The objective of this article is to apply advanced techniques of software engineering, requirements engineering, and artificial intelligence to system design. The expert system is a computer-aided software engineering (CASE) tool designed to aid in the gathering, analysis, formalization, documentation, and management of requirements for complex systems.

The importance of requirements engineering to the overall software design process should not be underestimated. Getting the requirements correct from the beginning is crucial to ensure that customers get what they want and to minimize defects. Errors not found during requirements analysis will lead to expensive defect

ogy in addition to guidance of lower levels such as individual requirements and documents.

METHOD

The expert system has been implemented in three layers—interface, development, and domain (see Figure 1). The latter two contain various conceptual models (intelligence models, development models, and various domain models) to guide and constrain the user in the requirements gathering and specification phases.

The domain models mainly provide domain-specific information, and the development models guide the designer through the methodology. The intelligence models provide the most comprehensive guidance, telling the designer the next development steps. The methodology is based on a three-dimensional framework for system design—completeness, refinement, and formality (see Figure 2). The aim of the methodology is a completely refined, formal specification of the system under development.

Also, there are documents, such as the Initial Customer Description (ICD), the Brainstorming List (BL), and the System Definition (SD), in which the customer plays an integral role. The ICD is a paragraph that briefly describes the end product that the user requires. The Brainstorming List is the initial list of possible requirements that have been thought of during the early stages of requirement engineering. Finally, the System Definition is a document that contains the description of each requirement. Completing these documents marks the completion of the requirements engineering phase.

Each requirement has various attributes that facilitate traceability and annotate the development. Traceability, which is an essential part of requirements management, is achieved

with three different types of links—structure, logic, and history. By maintaining these attributes, it is easy for a developer to discover the effect that making a change to one requirement may have on the overall system. It also provides a history of the requirements, so future developers may understand the reasons for certain features and the dependencies among the requirements.

Development methodology leads the user through the various stages of completeness, refinement, and formality. Completeness refers to the proper description of the requirement, ensuring that nothing is left out, so each requirement describes exactly what the customer requests. A complete system or component description specifies normal behavior, parallel behavior, exceptional behavior, and the overall behavior of the system. Normal behavior refers to the standard use of the system, parallel behavior allows the user to select among various options, and exceptional behavior specifies the system actions during errors.

Refinement refers to the decomposition of high-level system characteristics into lower-level components. This dimension currently has three stages—system level, high-level components, and low-level components. Depending on the domain, these different component levels can take on a more domain-specific meaning. Refer to [5] for an example of customizing the methodology to the telecommunications domain, in particular to the intelligent network (IN) architecture. [6]

Formality refers to the descriptive quality of the requirement. Beginning with textual descriptions of the requirement, formality increases towards a complete system specification in a formal language (such as SDL [7]). This transition has been simplified by introducing several semi-formal steps, in particular a three-stage use-case design process. [8] After the final level of formality is achieved, it should be easy for you to implement the requirements using commercial code-generation tools. The tools guides the user through these three dimensions in such a way that no area gets too far

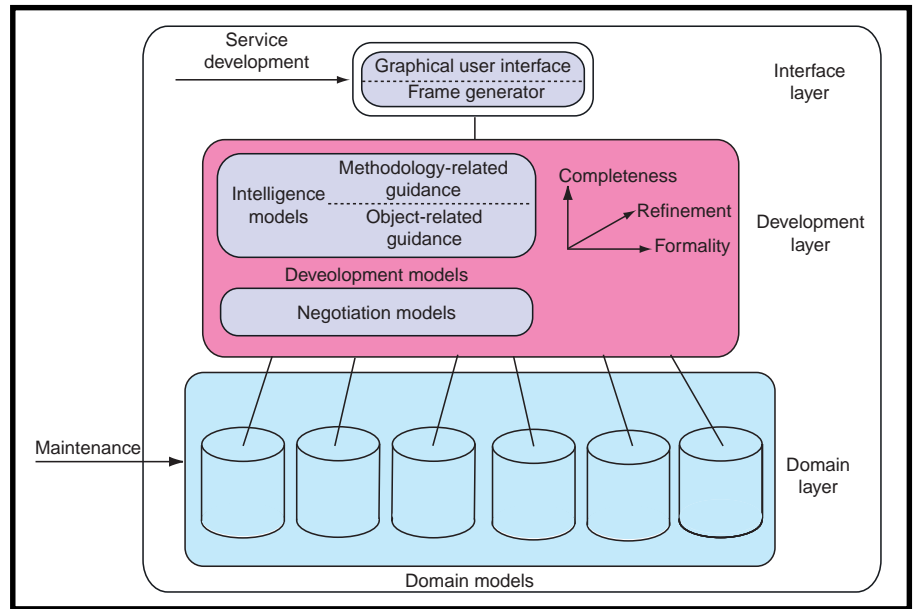


Figure 1—Here you can see the architectural design of the expert system.

ahead of another. Figure 2 shows a graphical representation of these three dimensions.

The domain layer contains information about the system domain. It provides you with information about certain standards the requirements have to meet, system constraints, and information about the customer's environment, such as the technology in place and how the software is used.

IMPLEMENTATION

The expert system is implemented using a knowledge-representation language called Telos [9] and a database system called ConceptBase. [10] The expert system is designed in a layered approach with the different layers providing different kinds of intelligence. Currently, the development layer and domain layer are implemented in the Telos language, and the interface layer is still being developed using Java.

The interface layer consists of two components, the graphical user interface (GUI) and the frame generator (FG). By implementing these two components separately, the functionality of the interface layer can be changed without having to change the frame generator. The interface layer also is the client of the expert system and runs on the user's computer, while the ConceptBase server, which contains the development and domain

layers, runs on a remote machine. As long as the expert system user is allowed to connect to the server, it does not matter where the server is located on the network, even if the connection is made through the Internet. This allows for distributed system development.

After the connection to the server is established, the interface layer of the client processes the user's requests in the GUI and then passes them to the frame generator to perform the necessary database accesses. The GUI contains three major areas—the document view, the network view, and the messaging/guidance windows (see Photo 1).

The document view has three tabbed panes, one for each of the three different kinds of documents mentioned earlier. It is in this area where the requirements engineer will perform the tasks of gathering and refining the requirements.

The network view shows a graphical representation of the overall requirements structure by displaying how the requirements are related and which type of link (e.g., logical or historical dependencies) exists between the requirements. On the right, there are three messaging/guidance windows. The task of these windows is to provide feedback in the form of error/success messages from the ConceptBase database, guidance from

the tool about how to proceed in the refinement process, and error/success messages from the tool.

The frame generator currently acts as a buffer between the GUI and server for error messages. The FG catches the exception thrown by the calls to the server and throws a generic FGException for the GUI to catch instead. Operation success messages are generated by the expert system and displayed to the user.

User inputs are processed by the GUI and changed to function calls to the frame generator, which provides the interface layer with a means of communication with the database. These functions perform all of the insertions into and deletions from the database and the connection to the server. These functions have been optimized with the GUI to provide as few database accesses as possible while performing certain tasks. An example of this can be seen when the user creates a new requirement.

The GUI opens a new window to collect the necessary fields, asks the FG to retrieve all instances of the class agent, and displays them in a box in this window. After the data is gathered, the GUI calls the necessary functions in the FG to create the appropriate Telos frame. This portion is done without accessing the database until the new requirement is inserted into the database. For every new re-

quirement, there is a minimum of two database accesses, one to retrieve the agent list and one to insert the requirement.

While the requirement is being inserted into the database, several actions take place. The various modules in the development layer and domain layer contain numerous rules and constraints, which ensure that the requirements that are inserted follow the three-dimensional framework for increasing formality, refinement, and completeness on each level (see Figure 2). Any guidance from these modules will be relayed back to the FG and then passed to the GUI to be displayed in one of the messaging windows. While the constraints prevent the insertion of incorrect information into the database (which relates to passive guidance), the rules enable the provision of active guidance to the system developer.

There is a set of "intelligent objects," which contain information concerning the next development step. These objects can relate to individual requirements, as well as the overall development process. Such an object is linked to a requirement, depending on the state of the requirement. The intelligence rules check the development state of each requirement or requirements document and are triggered if certain conditions apply. The rules then link the appro-

priate intelligence objects to the particular requirement, so they can be displayed to the developer.

RESULTS

The current results are still preliminary because the interface layer is being refined. However, initial results are promising. The methodology is completely implemented in the Telos language and, together with the intelligence models, provides comprehensive guidance to the developer. The expert system can be asked what further work is needed for a certain requirement or a requirements document and responds to the user with instructions for the tasks that are necessary to further develop the system specification according to the implemented methodology. The user also is informed about any actions that violate constraints specified in the database, preventing incorrect use of the development methodology. This shows that the initial objective of the research is met and intelligent support for the traditionally difficult first phase of the system development life cycle can be provided using an expert system.

Requirements management issues are addressed by the links that are created among the requirements. This allows the tracing of a requirement back to its initial source and caters to the impact analysis of a change request.

EVALUATION

In order to evaluate the approach, a case study is planned. Connections to the telecommunications industry are already established, and the experimental development of an intelligent network service is intended. The framework for requirements engineering has already been adapted to the intelligent network (IN) architecture by mapping the system level of the framework to the service level of the architecture, the high-level components to the service features, and the low-level components to the service-independent building blocks. This customization offers more possibilities for intelligent support.

The case study is expected to reveal any shortcomings of the overall approach of using an expert system for

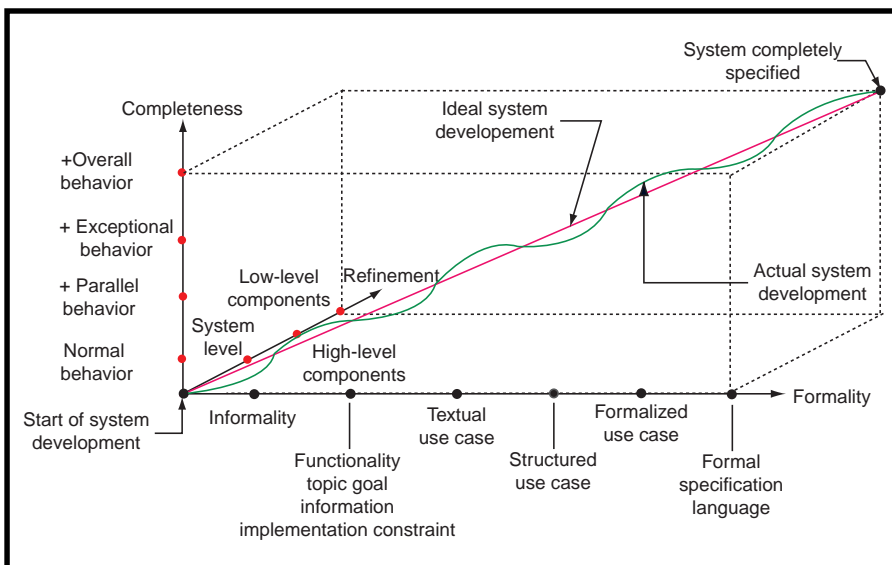


Figure 2—The methodology is based on a three-dimensional framework (completeness, refinement, and formality) for system design.

the early system development life cycle and offers the possibility to mature the methodology. Scalability is also of concern and modifications might become necessary to allow the development of large-scale systems using this expert system. Currently, this is one of the major issues. The use of the ConceptBase tool for requirements engineering can easily result in a number of objects that drastically slow down database queries. Although the current experimentation still performs well, a large-scale case study will require a high-performance workstation to ensure acceptable response times and usability.

WHAT WE'VE LEARNED

The requirements engineering methodology and its implementation in a tool aptly portray the use of an expert system to facilitate the gathering of requirements information. The tool provides active and passive guidance at various levels to the system developer, resulting in a high-quality specification of the new system. This will result in faster time-to-market and increased reliability and customer satisfaction.

I encourage the expert system community to use expert systems not only for specific domains, but to apply them to software development as

well. The software community is still struggling with many difficulties and would welcome any support for developing reliable software that meets customers' needs. Widening the scope, the general use of artificial intelligence during system development is an area of research that requires much more work, but has great potential for success. ▣

Dr. Armin P.-G. Eberlein is an Associate Professor at the University of Calgary, Canada. He graduated from the Mannheim University of Applied Sciences in Germany with a Dipl.-Ing.(FH) in Telecommunications Engineering. After his degree he spent some time working as a hardware and software developer at Siemens in Munich, Germany. This was followed by a postgraduate course in Communication Systems at the University of Wales, Swansea in the UK leading to a MSc and PhD. Dr. Eberlein's research interests focus on the improvement of requirements engineering practices and techniques.

Dale Fukami and Wonh Chieh Lam were undergraduate students at the University of Calgary where they obtained a BSc in Electrical & Computer Engineering.

You can reach the team at eberlein@enel.ucalgary.ca

REFERENCES

- [1] I. Sommerville, *Software Engineering*, Addison-Wesley, 1995.
- [2] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, NY, 1981.
- [3] "Can Artificial Intelligence help during Requirements Engineering?," *Proceedings of the 28th Argentine Meeting on Informatics and O.R., Workshop on Requirements Engineering (WER '99)*, Buenos Aires, Argentina, 1999.
- [4] R. Reddy, "The Challenge of Artificial Intelligence", *Computer*, 29, (10), pp. 86-98, 1996.
- [5] A. Eberlein and F. Halsall, "Telecommunications Service Development: A Design Methodology and its Intelligent Support", *Journal of Engineering Applications of Artificial Intelligence*, 10, (6), pp. 647-663, 1997.
- [6] K. Terplan and P. Morreale, *The Telecommunications Handbook*, IEEE Press, 1999.
- [7] ITU-T Z.100 Recommendation, *Specification and Description Language (SDL)*, International Telecommunication Union, Geneva, Switzerland, 1999.
- [8] A. Eberlein, "Bridging the Gap Between Informality and Formality in Telecommunication Service Design," *Proceedings of the 8th International Conference on Telecommunication Systems (ICTS2000)*, Nashville, TN, 2000.
- [9] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, "Telos: A Language for Representing Knowledge about Information Systems", *ACM Transactions on Information Systems*, 8, (4), pp. 325-362, 1990.
- [10] M. Jarke, M. Jeusfeld, and C. Quix, *ConceptBase V5.1 User Manual*, University of Aachen, Germany, 1999.

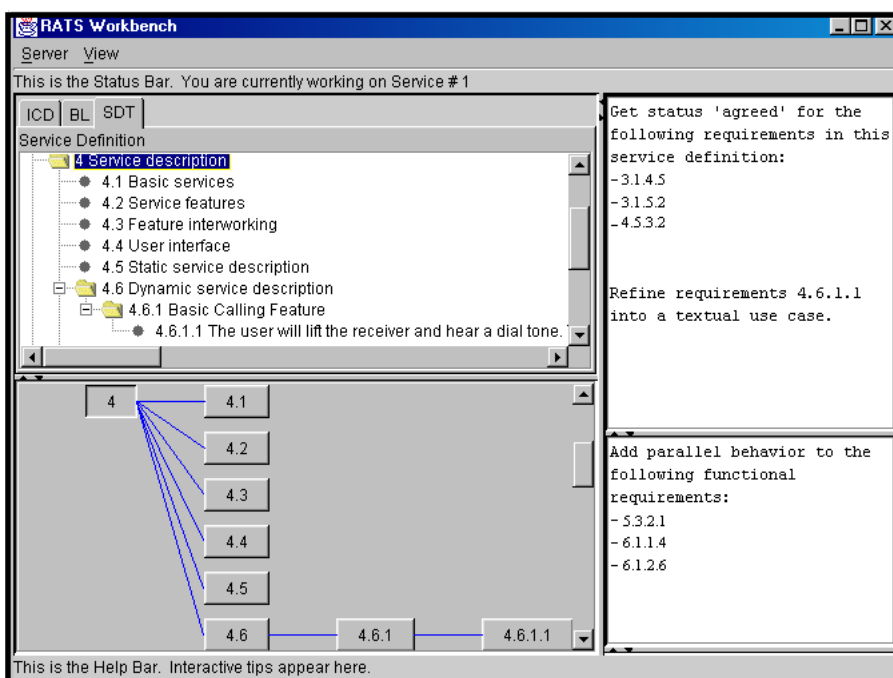


Photo 1—The document view, the network view, and the messaging/guidance windows are all accessible from the GUI.

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuircell.com or www.circuircell.com/subscribe.htm.

