Requirements and Tools for Reasonable End User Development

Marcus Elzenheimer, Jörg Lonthoff, Erich Ortner

Business Informatics – Development of Application Systems
Institute of Business Administration, Technische Universität Darmstadt
Hochschulstraße 1; 64289 Darmstadt
{elzenheimer, lonthoff, ortner}@winf.tu-darmstadt.de

Abstract: The concept of end user development (in the sense of application development done by the end user) is not suitable for all situations. Therefore, this paper describes the situations and conditions where end user development can be reasonable applied and mention interesting tools for supporting the application development processes. In the near future these tools will enable the development of application systems using reconstructed natural languages. So an entirely new concept of application development for end users will become thinkable.

1 Motivation

Developing application systems means that software development must be perceived as a whole and, in addition to software development, further aspects must be considered: e.g. the context-providing aspects such as related processes or the (business) organizational structure.

If you look at different application system designs one will find that there are multitudes of (expert) information's available and in use. That information can, and should, later be retrieved directly by the end user. [Or05] End users have explicit expert knowledge and are able to provide a large number of feasible and useful ideas for design in their field of application (that is to be reconstructed). We can obtain some of this knowledge quite easily by asking people directly. A little more effort is required if this knowledge is available in an implicit form. In this case we can recognize it only by studying the people during their everyday actions in the field we want to reconstruct.

This will result in condensed information, in form of a number of expert statements about the field of application that is to be supported. It is now our aim to submit this collection of statements to a linguistic reconstruction process while phase of expert design and, to keep it consistent. If we do this successfully, it is quite possible to further formalize these statements and examine them for certain patterns. If we further develop flexible transformations between these patterns and the diagrammatic patterns of the system design's modelling language, there will be no obstacle to the vision of an end user-driven software development.

2 Software Development

2.1 The Main Players in the Process

Looking at application development over the past decades we find a number of methodology changes. This is due partly to the requirements of software systems that are becoming more and more complex and, on the other hand, to the aspect that end users are the bearers of most relevant information.

In the early days of software development the programmer was responsible for the implementation and that it was the person who integrated the major part of the expert knowledge into the application. The end user was not involved but merely confronted with the results. This procedure proved to be unsuited (as described) for developing complex application systems, which is why software designers were integrated into the process. Quality and pragmatism of the software design has been shifted to their responsibility. In parallel, it was acknowledged that the end user had the competence (in the sense of ability) to contribute its expert knowledge to the desired solution.

2.2 The Ideal Software Development Process

Currently, software is developed using a functional specification that is based on the given facts of language composed "reality". This specification describes the requirements the software to be developed must meet. In a structured development process it will form the basis for the subsequent expert and system designs. This may cause some problems, because the functional specifications were often composed within a different intention. Therefore this functional specification may not be suitable as a basis for the expert and system design. Therefore we suggest creating and maintaining a collection of statements in addition to the functional specification. This collection should contain all statements and requirements the software to be developed must satisfy, in a way that is independent of technology and methods. This base of statements must be standardized in the phase of expert design and, later provide the basis for the system design. Standardized signifies that all terms of the expert language that is used are uniquely defined, and that their use is specified clearly and unambiguously. The definitions refer to the number of possible sentence constructions and, therefore, also to the possible combinations of one term with others or, with structuring words. But there is also a demand for a stabilized solution. Ideally the attempt is made to integrate the software in the application system. To integrate means for example, to synchronize processes, to adapt organizational structures, as well as to provide training for end users.

3 Favourable Conditions for Effective End User Computing

This paper makes the case that end user development is only to be seen in the context of business software and, particularly in the development of ERP (Enterprise Resource Planning) software. Once this problem can be handled scientifically, economically and technically, the attempt can be made to go on some further steps and include end users substantially in the software development. Then it might be possible, to step beyond their often contribution of naming functional demands while of the creation of a marketing mix or

marketing concept (e.g. in the case of product policy). Focusing on ERP Software can be explained by the fact that this type of software reconstructs a significantly large part of generally available knowledge (processes, data structures, organizational structure), which isn't done in some kind of Office-Software. In fields of software development that are very specific and/or very technical (for e.g. software for optimization a round trip problem) it is less easy to integrate end users. The same is true for software that is produced for an anonymous market. Here would it be also difficult to formulate general requirements and reconstruct an implicitly existing structure. The role of end users developing in several types of software still needs more research.

3.1 Prerequisites

The ideal software development process can only be implemented if the developer is able to put himself in the user's position when specifying the desired solution. Even if designers and developers communicate to a large extend diagrammatically, this is not at all recommended for communicating with end users. An exception to this rule might be a situation where the diagrams that are used are simple and clear (e.g. Use Case diagrams). Another exception might be allowed if the circumstances to be reconstructed are extremely complicated and full of variation [Oe05] p. 305. It is our stated objective and strong advice to communicate with the user using a standardized natural language.

3.2 The Right End User

Even if the vision of software development described above may one day come true, there is no doubt that this procedure is not applicable to all situations and, in particular, not suited equally well for all users. This strongly depends (as described) on the software systems to be specified. Our assumptions are based on business end users who are professionally interested in their business field and are motivated by their daily work to specify a requirement in such a way that the later application system can be derived from it. We assume additionally that, these end users are interested in influencing their environment. We believe that this kind of motivation cannot always be found with end users in the private sector. Derived from this assumption one could state that the integration of private users in the software development process may be much more difficult. In our point of view private users can only be integrated in the process by considering demands for the future application system. We anticipate, however, that these demands will be too inconsistent, unspecific and incomplete.

3.3 The Capability of Schematization

If the question of user integration in the development process arises, it is important to determine beforehand the abilities these users must have to allow efficient integration. Generally, linguistic competence (that is, the ability to express themselves) and linguistic performance (that is, the ability to use language pragmatically and accurately) would be of advantage, but it is no prerequisite for obtaining information or knowledge about the domain to be supported. The information and knowledge lie within the users' artefacts. It is the developer's responsibility to compensate for the users' possible lack of linguistic competence or performance.

Knowledge is gained from information by means of abstraction or schematization, thus identifying invariant connections. These schemas can then be used in the context of the later application system by “instantiating” them in any number and make so use of them. It is obvious that user integration will be even more efficient if end users themselves are able to schematize. It ought to be a developer's stated objective to identify these in particular. On the other hand the process of obtaining knowledge will be expensive.

4 Tools for Supporting the Software Development Process

The university department "Development of Application Systems", computer and Business Informatics, in Darmstadt has performed several preliminary studies on the subject over the past few years with the intention of proving the general feasibility of the described approach. Among these studies were the following:

- **OTMAR** [Vo94] is an acronym that stands for Object Type Model Translator. The program enables the automatic translation of statements of a standardized language into the data model of a possible DBMS.
- **NELA** [GKR05] (Normative Expert Language Application) aims at collecting and maintaining users' field-specific statements directly in a regulated language (standardized language).
- **IDT (**"Intelligent Development Tools") aim supporting the comprehensive approach of the software development described. Statements in a strongly simplified German or English could be transferred into a diagrammatic notation. At the moment, the functional scope is limited (transferring a subset of the UML class diagrams is possible), but tool development will be continued.

5 Conclusion

Transforming natural-language statements into diagrammatic statements may become possible as the studies confirm. With help of MDA and generators it is even possible to transform these diagrams into source code. Based on this, in the future we will examine what kind of user knowledge can be used for the specification of an application. We would also like to thank our students for their contributions on IDT, which is based on the preliminary work in form of several studies and dissertations (OTMAR and NELA).

6 Literature

[GKR05] Gerhardt, Florian; Klunker, Gerald; Roos, Christoph: Sprachkritische Systementwicklung - Normative Expert Language Application (NELA), Seminararbeit mit Prototypischer Umsetzung, Darmstadt, 2005.

[Oe05] Oestereich, Bernd: Analyse und Design mit UML2, 7. akt. Aufl., München, 2005.

[Or05] Ortner, Erich: Sprachbasierte Informatik, Leipzig, 2005.

[Vo94] Vogler, Martin: OTMAR - Ein automatischer Übersetzer zur Konstruktion von Objekttypen und Beziehungen aus Aussagen, Diplomarbeit, Konstanz, 1994.