Towards a Methodical Development of Electronic Catalogues*

Nora Koch† Andreas Turk‡

Introduction

The technologies currently used to develop and deliver multimedia systems like electronic catalogues are still far from being elaborate and easy to apply. Due to the distinct difficulties arising during development, production, and maintenance of sophisticated multimedia software, it is necessary to get the job done by a multi-disciplinary team of programmers, graphic designers, media-experts, and quality control specialists. The results of their work could be significantly improved by easy-to-use tools which achieve the following goals:

- help to determine the requirements,
- reduce the design efforts,
- increment the quality testing speed,
- reduce the costs of producing and updating multimedia systems, and
- simplify the system maintenance.

Electronic product catalogues (EPCs) are information systems, that put emphasis on the multimedia presentation of products (or services) and which contain some standard functionality of searching, selection, and ordering of products. In this paper we will concentrate on this particular sort of information systems. However, we expect most of the presented topics to be useful for the development of information systems in general.

We present some of the results of the EPK-fix[1] project, which deal with the systematic construction of EPCs. Within the scope of this project a methodology of EPC-engineering has been developed and an integrated set of powerful software tools has been designed and implemented to cover the whole life cycle of electronic product catalogues on CD-ROM. A SGML-based specification language permits a declarative description of catalogues, enabling the tools to support the requirements analysis and the specification of EPCs. The essential EPC-functionality is provided in advance as predefined services. An exhaustive automated validation of the resulting catalogue becomes feasible.

The project EPK-fix is supported by the german *Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie* (BMBF). The partners are the Bavarian Research Center for Knowledge-Based Systems (FORWISS) in Erlangen, the Ludwig-Maximilians-University of Munich, the Technical University of Darmstadt, the Technical University of Dresden, and the mediatec GmbH in Nuremberg.

In the first section we describe the software development process for EPCs. The second section presents the architecture of the EPK-fix system including brief descriptions of the specification language EpkMl and the four assistance systems. In the last section some conclusions are delineated.

*This work was supported by the BMBF, Förderkennzeichen 01 IS 520.

†Nora Koch (kochn@informatik.uni-muenchen.de) is a research assistant of the Institute of Computer Science at the University of Munich. Her main interests are in the fields of Multimedia and Electronic Publishing with a special focus on user modeling.

‡Andreas Turk (turk@forwiss.de) works at the Bavarian Research Center for Knowledge-Based Systems on strategies of system development. In the EPK-fix project he focuses on requirements and system analysis for EPCs.

[1] http://www.forwiss.uni-erlangen.de/fg-we/epkfix/

1 The Development Process

The development process for EPCs is derived from general object-oriented software engineering and thus contains the main phases *requirements analysis*, *system analysis*, *specification*, *automated software generation*, and *testing* (fig. 1) [7, 1].
The requirements analysis serves to elicit the primary customer expectations in order to decide about the feasibility of a project and the adequacy of the EPK-fix methods and tools. The resulting *requirements analysis document* may be part of a concrete offer. During system analysis, which introduces the actual development cycle, a model of the EPC has to be described informally. The resulting *analysis model* is used immediately as a guidance in the following graphical specification in order to generate a formal catalogue description (*specification*). During generation phase this formal description is transformed automatically into program code (the *EPC*). At last the EPC has to be tested against the specification and the analysis model (*test report*). Every iteration of the development cycle leads to a new prototype.
The development model shows an evolutionary nature: in the course of several iterations a prototype will approximate the customer's requirements, which in turn also have to be elaborated and detailed. Mutual adjustment belongs to the *prototype analysis*, which is part of the system analysis.
Each of the development phases (analysis, specification, generation, and testing) considers the orthogonal aspects of EPCs — *layout*, *structure*, *product data*, *navigation and direction*, and *special services*. These aspects are described in detail in [5, 4].

2 The System Architecture

The EPK-fix methodology and software tools support the complete production cycle of EPCs starting with the catalogue providers *requirements*, continuing with the catalogue *design* up to the functional *tests*. These tools have to be sophisticated and handy at the same time to reduce the time expended in EPC development and thus permit a low-cost production of catalogues — necessary conditions to employ EPK-fix in small and medium sized enterprises.

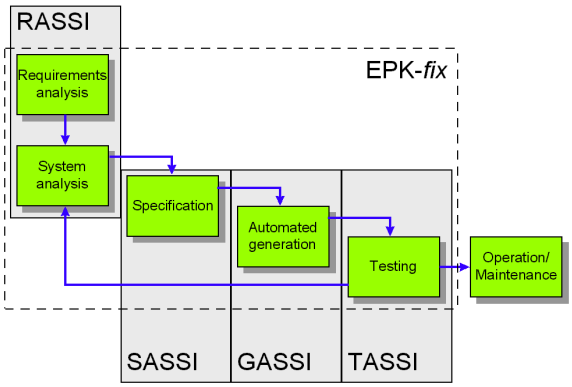

Figure 1: The EPC development model (overview)

The EPK-fix system components comprise the formal description language for electronic product catalogues (EpkMl) and the following software tools: the Requirements Analysis Assistant (RASSI), the Specification Assistant (SASSI), the Generation Assistant (GASSI), and the Test Assistant (TASSI), which will be explained in the following. An integration is provided by data exchange via World Wide Web (WWW) thus enabling distributed workgroups to act efficiently.

2.1 The EPKML Language

EpkMl is a specification language that enables the description of the static and dynamic aspects of the electronic product catalogue. EpkMl is a high-level HtMl-like language. It is defined as an instance of SgMl (ISO-standard 8879, [2]). Here we enumerate only some of the characteristics of the language, for more details see [3, 4, 6].
The Standard Query Language (SQL) is embedded into the language for easy access to relational *databases*. EpkMl has primitives for *navigation* flow and permits connection to external languages like Java-applets. The language enables a conceptual view of EPCs comparable to a structured, annotated, multimedia, and highly automatic database front-end.

Products are organized in *hierarchical structures*, which are the base for automatic navigation. The visual (layout) features of EpkMl are a superset of those in HtMl. EpkMl is *window-oriented* instead of screen-oriented as HtMl. *Multimedia* integration is achieved adding constructs expressing time-dependency while user interaction is specified with buttons and menus. Some features of the language are shown in the following example.

```
<theme name=all>
   <window style=introduction>
      <video name=presentation src=video.avi>
         <stop-button>
            <on-click>
               <close name=presentation>
            </on-click>
      </video>
      <open name=presentation-video>
   </window>
   <extension result=city>
      <sql>
         SELECT name,description,image
         FROM database
      </sql>
      <template name=cities>
         <page>
            <frame name=details>
                <img src=$city.image$>
                <p>$city.name$</p>
                <previous-button>
                <next-button>
                <back-button>
            </frame>
         </page>
      </template>
   </extension>
</theme>
```

2.2 RASSI

In our opinion, human dialogue is central to the analysis process in software engineering. Requirements analysis for EPCs can be viewed as *information acquisition*, compared to the acquisition of knowledge in knowledge-based system development. Thus, the fundamental analysis activity is to carry out *structured interviews* [8].

The *Requirements Analysis Assistant* is a software tool that supports the task of interviewing. It contains a couple of integrated submodules, that serve to manipulate and convert the basic object types *checklists*, *questionnaires*, *protocols*, and the final *analysis document*. Checklists enumerate all important aspects that have to be addressed during EPC design and development. They are to be extended to situation specific questionnaires for the interview. Interviews result in protocols, which in turn can be combined and edited to receive the analysis document.

The following are the RASSI submodules: a *questionnaire editor* is used to assemble a questionnaire from a checklist by formulating questions and adding explanatory documents; the *interview assistant* performs a complete audio recording of an interview, allows for synchronously written notes, and links arbitrary multimedia material together; a *presentation assistant* supports the revisiting and combining of multiple interviews and the generation of the analysis document.

As an example, fig. 2 sketches a single, unmodified topic concerning the arrangement of elements in the

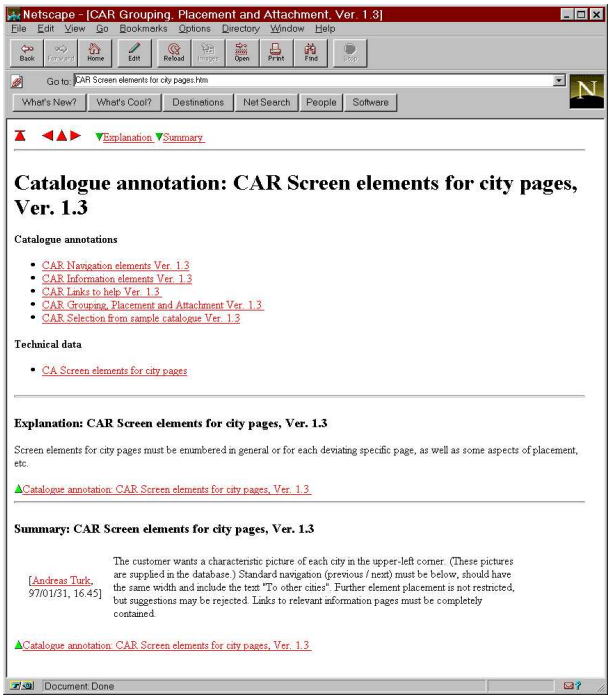

Figure 2: Part of a RASSI analysis document

user interface of a service catalogue (page design). It is part of the resulting documentation.

2.3 SASSI

The Specification Assistant supports the transformation of an informal description of an EPC to a formal specification. A qualified developer uses SASSI's set of powerful editors to compose an EPC according to the information gained by the requirements analysis. SASSI is based on an intuitive graphical user interface and implemented in Smalltalk. SASSI includes a three-window interface presenting the relevant part of the analysis document, one of the following editors: *structure editor*, *layout editor*, or *database editor*, and the corresponding part of the generated specification. Main features of the SASSI component are to support a complete specification of the analysis information, to protocol which analysis objects led to which specification objects, to make sensible assumptions where analysis information is incomplete, to provide version information in order to make recovery possible, and to generate syntactically and static-semantically correct EpkMl output.

2.4 GASSI

The purpose of the Generation Assistant is to generate catalogues based on their formal specification in EpkMl. These catalogues are in fact Java programs. GASSI is a fully automated tool, which needs no user interaction. Details and backgrounds of software generation must be regarded as laying beyond the scope of this brief article. However, a main concept of GASSI is the application of a generic software repository containing reusable components for EPC standard functionality, called *standard services* (online help, ordering facilities, search, etc.). Assembling software by predefined and well-tested components leads to improvement in quality.

2.5 TASSI

The Test Assistant ensures a high level of software quality in the development process. TASSI enables a test engineer to systematically and mostly automatically check static and dynamic aspects of an EPC. Where no automated testing is possible, strong support for *manual* tests is given. Some features of TASSI are: fully graphical user interface (fig. **??**), rule-based declarative specification of automatic tests, automated execution of static tests of media-objects, support of dynamic tests by a test agent, error classification, and automatic test document generation, besides others.

3 Conclusions

The EPK-fix project comes out with an integrated package of tools that support the whole life cycle of the production process of EPCs. These tools aim at the efficient production of low cost electronic product catalogues. Catalogues are described in a declarative form using the language EpkMl specially designed for that purpose. The developer is assisted from the beginning of the requirements analysis phase through the design and catalogue generation up to the testing process. Manual and automatic tests assure the correctness of the catalogue.
An EPC is in our approach the result of cooperating experts working at different places. In order to keep track of the correct versions of the documents and programs produced so far, version management is required. We are currently implementing a WWW-based project server, which permits to manage projects and which handles developers' access. Furthermore the server will control revision and release management.
Additional features (e.g. secure electronic online ordering and payment) are planned for the future commercial version of EPK-fix. Further *user modeling* aspects may be incorporated to the development process of EPCs, thus producing adaptive catalogues.

References

[1] B. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):61–72, May 1988.

[2] C. Goldfarb. *The SGML Handbook*. Clarendon Press, Oxford, 1994.

[3] A. Knapp, N. Koch, and L. Mandel. The Language EPKML. Technical report 9605, LMU München, November 1996.

[4] A. Knapp, N. Koch, M. Wirsing, J. Duckeck, R. Lutze, H. Fritzsche, D. Timm, P. Closhen, M. Frisch, H.-J. Hoffmann, B. Gaede, J. Schneeberger, H. Stoyan, and A. Turk. EPK-fix: Methods and Tools for Engineering Electronic Product Catalogues. In R. Steinmetz and L. Wolf, editors, *Interactive Distributed Multimedia Systems and Telecommunication Services*, LNCS 1309, pages 199–209. Springer-Verlag Berlin-Heidelberg, September 1997.

[5] N. Koch and L. Mandel. Catalogues on CD–ROM: The State of the Art. Technical report 9610, Ludwig–Maximilians–Universität München, December 1996.

[6] J. Schneeberger, N. Koch, A. Turk, R. Lutze, M. Wirsing, H. Fritzsche, and P. Closhen. EPK-fix: Software-Engineering und Werkzeuge für elektronische Produktkataloge. In M. Jarke, K. Pasedach, and K. Pohl, editors, *Informatik'97, Informatik als Innovationsmotor, 27. Jahrestagung der Gesellschaft für Informatik*, Informatik aktuell. Springer Verlag, September 1997.

[7] I. Sommerville. *Software Engineering.* Addison-Wesley, 4th edition, 1992.

[8] A. Turk and H. Stoyan. Erfassung, Verarbeitung und Dokumentation natürlichsprachlicher Äußerungen in der Anforderungsanalyse. In E. Ortner, B. Schienmann, and H. Thoma, editors, *Natürlichsprachlicher Entwurf von Informationssystemen*, pages 32–46. Universitätsverlag Konstanz, May 1996.