ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol.35, No.3

EDUCATIONAL PORTAL DEVELOPMENT MODEL FOR IMPLEMENTING DESIGN FOR CHANGE

Vytautas Štuikys, Robertas Damaševičius, Marijus Montvilas, Virginija Limanauskienė, Giedrius Ziberkas

Software Engineering Department, Kaunas University of Technology
Studentų St. 50, LT–51368 Kaunas, Lithuania

Abstract. We analyze the development of educational (EDU) web portals from the perspective of meta-design. Here, we focus on the technical issues of meta-design only and consider design-for-change. Our contribution is a variability model based on the variability analysis, generic portal development processes, sub-processes and their relationships. We analyze the interface design problem in the context of the EDU Portal development and consider the implementation of the proposed model using meta-programming.

Keywords: Design for change, portal development, meta-modelling, meta-programming, variability model.

1. Introduction

For a long time the ability of end-users to effectively influence the IT development process was largely limited, despite of the innovative design methodologies such as *user-centred design* [1] or *participatory design* [2], where the needs and limitations of the IT end-user are given extensive attention at each stage of the design process. However, recently the situation is changing dramatically, especially in the end-user development for non-critical applications such as home appliances and learning-oriented ones. This change can be explained by: (1) higher degree of IT maturity and IT penetration into home environments, (2) higher motivation of end-users to be involved in system customization and adaptation, (3) complexity growth of IT systems and inability of designers and analysts to completely foresee and formulate requirements for future systems, (4) rapid growth of the end-user development community [3], (5) arrival of new design paradigms such as *meta-design* [4, 5].

Meta-design is actually a vision and strategy for the future end-user oriented IT development, in which design, learning, IT-based knowledge, and collaboration becomes a part of every day's working practice. The underlying concept of meta-design is about (1) how a system and its environment should be designed by experts, and (2) which way the end-user should be involved in the process that future changes and system evolution could be practically performed by efforts of the end-user. Although this new paradigm is not well understood yet and many issues are still open, it proposes great promises to seamlessly extend the system life cycle model from the development stage to the maintenance and evolution stage.

Meta-design should be especially useful in the domain of eLearning, where the interaction between experts, developers and end-users, and evolution of the teaching content plays an essential role in the development of educational (EDU) Portals [6, 7].

Our aim is to analyze the EDU Portal development process from the perspective of meta-design. In general, it includes two aspects: design for change and social-economic aspects [5]. In this paper, we focus on *design for change* and consider technical problems of meta-design only. Our contribution is a variability model based on the generic portal development processes, sub-processes and their relationships. We analyze the proposed model, the user interface design problems and implementation of the proposed model using *meta-programming* [8].

2. A framework of design for change: assumptions, context and principles

Design for change can be considered at different time and from different positions. Here we consider design for change in the framework of component-based design. We assume also that the architecture of a system is stable and only components of a system can be changed. For example, when *white-box* reuse is applied in system development, it includes the following stages at *design time*: to find appropriate SW components, to analyze and understand them, to modify or *change* them and to apply the modified components in a new context [9, p. 418]. The other example is maintenance and evolution of legacy systems at *use time*, when changes are managed according to very strict activities and procedures, such as the *impact analysis* prior to changes are actually performed [10].

Meta-design is aiming to plan, evaluate and incorporate at some extent the possible changes at design time in order to ease their implementation at use time based on the *evolutionary model* [4]. The latter deals with design processes and models enabling involvement of the end-users into the design process that he/she could become a *meta-designer* [5] and be able to perform changes at system evolution stage. As the issue of meta-design should be the creation of *solution spaces* [5] instead of specific solutions, which usually takes place in a traditional SW development, it is important to focus here on two topics: (1) *variability analysis* [11] in the given domain, (2) *interface design and modelling* [12].

Meta-design includes two major activities: meta-modelling and meta-programming. *Meta-modelling* [22] aims to derive an explicit description of how a domain-specific model is built. *Meta-programming* [8] aims to derive a generic solution (component, program, system) based on the results of meta-modelling and variability analysis.

The need for variability analysis can be motivated by the fact that designers increasingly spend their time for creating many similar IT systems (*program families* [13] or *product lines* [14]) with many variations. A systematic variability analysis and variability model can help designers (1) to identify and isolate *commonalties* in the domain, (2) to achieve higher design reuse and ease of change, (3) to predict the results of system's evolution, and (4) to identify opportunities for automating the creation of family members (instances).

For example, designers may develop IT systems using standardized, abstract interfaces to each subsystem (SW component, device, etc.), and encapsulate domain variability into separate modules accessed though the interface. In this case, the commonality is represented by the interface, and the variability – by the code. The other way is to develop common system building blocks, which communicate using different interfaces or *wrappers* [15], constructed (or generated automatically) on demand. In this case, the commonality is represented by the component, and the variability – by the interface.

The result of variability analysis is a *variability model.* For devising and implementing the variability model, we apply the following principles.

A. Emphasis On Program Change Ontology And Specificity Of The Application Domain, Development Methodology And Existing Standards

Principle A is about the analysis of ontology of S-, P- and E-programs and their external and internal environments within the framework of maintenance and evolution [16].

An S-program addresses a completely defined problem and provides an exact *specification* and correct solution. If however the environment of the problem changes, the result is a completely new problem that must be specified anew.

A P-program is based on a practical abstraction of the *problem* it addresses. The P-program's abstraction can be modified to reflect the changing requirements.

An E-program is *embedded* in the real world and changes as the world does.

Here, we focus on P-programs, because the EDU Portals belong to this category and, furthermore, it is a non-critical application.

B. Separation Of Concerns

Principle B is about the way of how a general design problem can be simplified and dealt with. We apply separation of concerns at different stages of the IT development process for separating: (1) the non-technical aspects of change from the technical ones at analysis stage, although they are overlapped and intertwined; (2) analysis from implementation; (3) problem context from model building, etc.

C. Selection/Adaptation Of The Appropriate Analysis Method

Principle C is about selection and adaptation of a systematic method for analysis. We have adopted the well-known method FODA (*Feature-Oriented Domain Analysis*) [17] and consider it within the so-called *twin life cycle model* that connects *design for reuse* and *design with reuse* [18].

D. Emphasis on Variability Analysis For Both Non-Technical And Technical Aspects

Principle D is about capturing, analysis and representation of variability in the domain. Here we focus on the technical aspects only. We consider Portal development as a domain of generic design processes, and sub-processes as sub-domains. By generic processes, we mean Content Management, Knowledge Management, User Relationship Management, Collaboration and Security-based ones, which in [19] are called "*the key elements*" of a Portal. We seek to obtain variability within the sub-processes of the generic processes as it will be explained in detail later.

E. Analysis Of The Approaches For Implementing The Variability Model To Support Design For Change

Principle E is about implementation technologies. We restrict ourselves in using *meta-programming* [8] and *pattern-based (or skeleton-based)* [20] approaches in implementing the proposed variability model.

3. Description of the variability model

Here, we represent the process of creating the model and model per se using Y-Chart [21], where two higher branches represent the problem domain and the solution domain.

System design is about implementing a solution to a given problem. It starts with a formulation of a problem and ends with its solution. To formulate a problem, we must, first, to analyze a problem domain. Problem domain concerns, basically, represent the concerns as seen from the end-user's perspective and focus on the functionality of an IT system as the client expects it. The analysis of the solution domain is a different matter. Solution domain concerns represent the concerns as defined by the solution techniques and focus on the implementation of a system from the designer's perspective: mapping problem domain onto solution domain using application development methods and models.

In this context, by the problem domain, we mean the EDU Portal development. By the solution domain, we mean the adopted FODA method. We consider the creation of the proposed variability model as a result of mapping the solution domain onto the problem domain (see Figure 1).

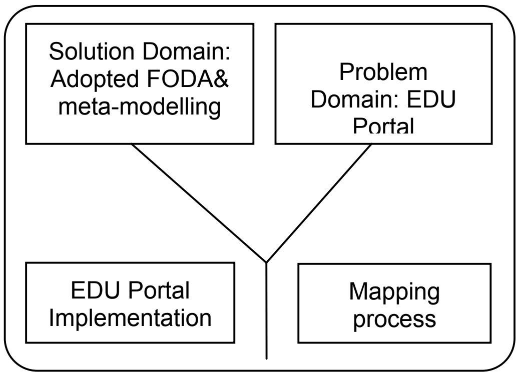
Solution Domain: Adopted FODA& meta-modelling
Problem Domain: EDU Portal
EDU Portal Implementation
Mapping process

Figure 1. Representation of the model using Y-Chart

Before mapping, we categorize the EDU Portal development process in the whole as consisting of 7 stages (see Table 1). Stages 1 and 2 deal with analysis, stages 3-5 – with mapping, and stages 6-7 – with implementation of the variability model, respectively.

Now, we consider the problem domain that is EDU Portal development. Because of the ever-increasing IT capabilities and continuous complexity growth of user requirements, we need to analyse and structure the given domain. We define the process as *meta-modelling* [22].

We consider the variability analysis of the domain (facts, knowledge) as a main task of meta-modelling. The issue of domain analysis and meta-modelling is a *set of parameters*, their *values* and *relationships* across parameter values. We represent the problem domain as a fixed set of parameters, which have well-defined values, and the solution domain as a set of design processes and sub-processes.

Table 1. Stages And Processes Of Variability Model

Model stage	Processes
Stage 1:	Analysis of EDU Portal environments and their key architectural elements
Stage 2:	Analysis of a generic 3-tiered architecture of EDU Portals
Stage 3:	Definition of generic processes and sub-processes and their analysis
Stage 4:	Building variability tables and obtaining parameter values through variability analysis
Stage 5:	Obtaining parameter dependency relations and building relationship graphs
Stage 6:	Obtaining components and their variants through application-level analysis
Stage 7:	Selection of approaches for implementing variability of the components

An example of the variability analysis is presented in Table 2 as a result of the usage of the proposed model. The underlying concepts are processes and sub-processes in our model. Here, we take the processes defined in [19]. The sub-processes of Content management processes are generation, assimilation, personalization, and presentation. These sub-processes can be applied for implementing a number of different criteria such as (1) *Presentation language* (English, German, local). (2) *Type of content* (textual, graphical, audio, video, etc.). (3) *User-oriented access levels* (Administrator, Publisher, Reader, etc.). (4) *Device type* (Desktop PC, PDA, mobile phone, etc.).

The introduction of the process concept in our model is much more than just renaming terms used in [19]. By defining attributes (such as parameter values) of a process, we can model the system behaviour. By obtaining relationships between processes (sub-processes), we can understand the system's structure.

Table 2. Results Of The Variability Analysis For The Content Management Process

Sub-processes	Criterion	Parameter values	Solutions for implementation
Generation	Presentation language	EN, DE, FR, RU, LT, …	Separation of presentation logic and page content
Assimilation	Type of content	Text, Picture, Audio, Video, …	Hierarchical parameterisation model
Personalization	User	Administrator, Publisher, Reader	Modes and security levels
Presentation	Device type	Desktop, PDA, mobile phone	User interface model

Different processes and sub-processes are related (see an example in Figure 2). The relationship is not static and can be changed depending upon the target system and end-user's requirements. In order to obtain a relationship, first we need to introduce some parameters and obtain their values though in-deep analysis and meta-modelling and create a variability document for expressing variability (a small part of it is shown in Table 2).

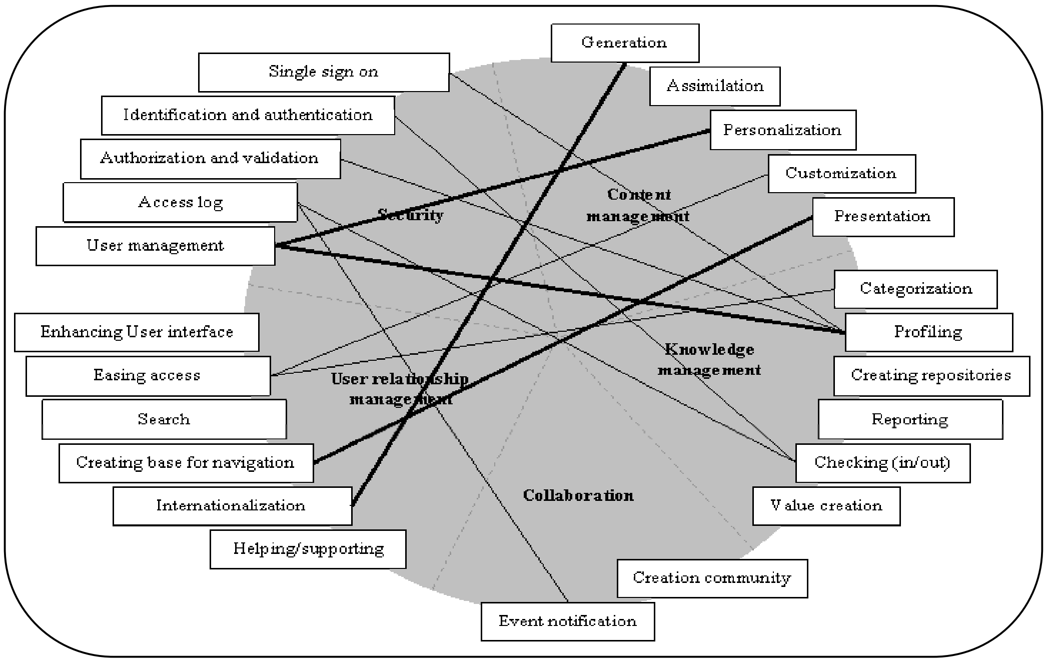

Figure 2. EDU Portal – related processes, sub-processes and "dynamic" relationships

4. Analysis of user interface problem in edu portal development

Interface modelling is an important problem in meta-design of web systems. Here, we have two problems: (1) modelling of interfaces between components and services, and (2) modelling of an interface between a system and an end-user, i.e., user interface. Here, the issues related with the user interface design in the context of design for change are considered only.

Currently, the main issue in user interface modelling [23] is the development of more human and intelligent forms of interfaces such as audio/speech interfaces, touch screens, handwriting interfaces, facial expression and gesture reading interfaces, special interfaces for disabled users, etc., in the context of *Ambient Intelligence* [24] environment in which people are surrounded with networks of embedded intelligent devices that provide ubiquitous information, communication, services and entertainment. The emphasis is given to the mobility of the end-user (or *nomad* [25]), wireless access to remote information and *intelligent user interfaces* [26].

Design of user interfaces has many technological challenges such as (1) diversity of devices, services and applications; (2) text input facilities; (3) screen size and resolution; (4) number of colours to properly display interface content; (5) different image, video and audio compression formats; (6) navigation, e.g., mouse-based, pen-based, etc.; (7) layout; etc.

Thus, interface designers are required to provide many different variants of interfaces for the same service or system. Designing user interfaces for a web service that must target many different types of devices, platforms, types of interfaces and user groups is a tedious and time consuming work. The designers have several alternatives:

(1) Design of an application-specific interface for each specific IT system and its customization for a specific user group. The disadvantage of this approach is that a designer has to be accustomed with many types of systems, devices and interfaces across the domain. Furthermore, there is an unnecessary repetition in implementing the same interface again and again. Creating multiple versions of interfaces for different devices increases development and maintenance costs and complicates the configuration management.

(2) Design of an interface for only one application and its adaptation for other applications. This approach can lead to the interfaces that are awkward to use or even not functional. Furthermore, it is hard to keep with consistency of a user interface when moving from one platform to another.

(3) Design of a high-level interface specification (model), which is open to changes and upgrading, and its refinement for each target device and user group based on the end-user's requirements. This approach, which basically deals with meta-design of interface, is the most promising in terms of addressing the design complexity problem and the end-users' needs.

However, it requires the development of the standard user interface specification methods, interface prototyping, design and validation tools.

Meta-design of user interfaces requires introduction of high-level abstract interface models. These models include high-level specification of the tasks that users need to perform, data models that capture the structure and relationships of the information that applications manipulate, specifications of the presentation and dialogue, user models etc, and automatically generate some parts or the complete user interface. Different types of models have been used in different systems including task models, dialogue models, user models, domain models, and application models [27], for example:

(1) *Platform model* describes IT systems that may run a user interface including features and constraints for each platform. It enables designers to generate a set of user-interfaces, one for each platform that is desired. The platform model can be sensitive to changing conditions of use at run-time.

(2) *Presentation model* describes visual appearance of user interface such as hierarchy of windows. Each window is modelled abstractly as a platform-independent interaction object.

(3) *Task model* is a structured representation of the tasks that a user may want to perform.

All these models represent the user interface at a higher level of abstraction than what is possible with a more concrete representation and requires thorough meta-modelling. The models can be implemented either automatically or semi-automatically to generate the user interfaces for a target system.

As we can see, the interface modelling of a web system already uses many of the principles of meta-design, such as thorough domain analysis, separation of concerns, complex modelling and meta-modelling of domain concepts and domain code generation. However, the concept of *change* is not sufficiently represented. We claim that the meta-design of user interfaces must include development of the user interface models that are *adaptable*, *evolutionary* and *open to change*. For this, each interface model must include its variability model that provides a framework for (semi-)automatic interface reuse and evolution.

5. Case study: Development of the EDU Portal

To illustrate the concepts presented here, we have implemented the above-described model by developing the experimental system for the generation of the eLearning-oriented Portal, partially described in [28].

The solution domain meta-model (see Figure 3) consists of two parts as follows: (1) Meta-program, which is developed using the meta-programming techniques. (2) EDU Portal, which consists of two parts: eLearning Content Management system (LCMS) and eLearning Management System (LMS).

Meta-program is the program generator for our problem domain implemented using the variability model described in Table 1. It represents variability in the domain (actually a family of the domain program instances) using the parameterisation mechanisms at a higher (meta) level of abstraction. To develop Meta-program, we need to map the problem domain parameters established via variability analysis (see Table 2) into the parameter space of EDU Portal. The component presented in Figure 3 has been implemented using relationships depicted by bold lines in Figure 2.

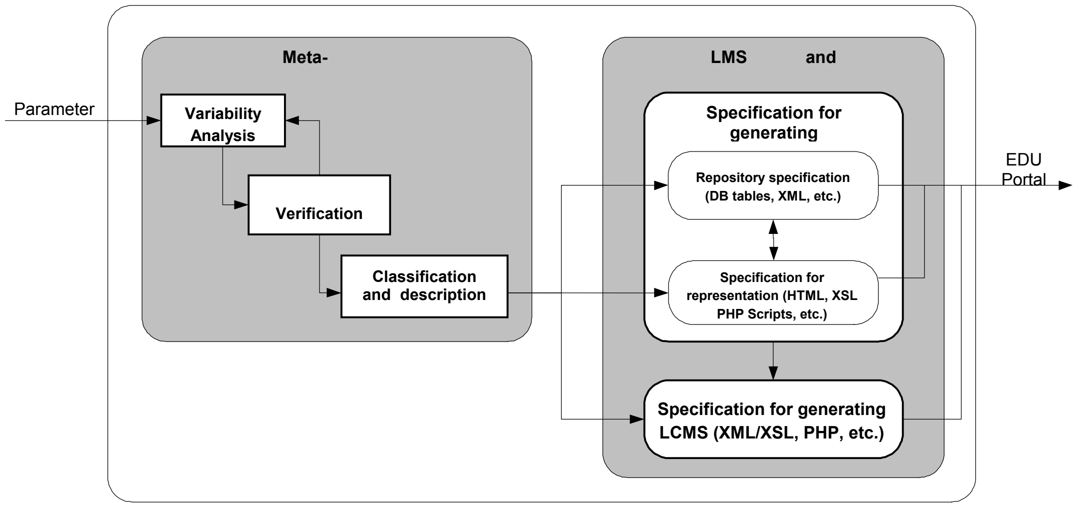
Meta-
Parameter
Variability Analysis
Verification
Classification and description
LMS and
Specification for generating
Repository specification (DB tables, XML, etc.)
Specification for representation (HTML, XSL PHP Scripts, etc.)
Specification for generating LCMS (XML/XSL, PHP, etc.)
EDU Portal

Figure 3. Detailed view of the solution domain meta-model

The skeleton of the target system (EDU Portal) is generated automatically from the Meta-program. It represents two aspects: (a) structuring and representation of the domain and (b) linking the domain structure with domain content.

From the usage viewpoint, the main features of the developed system are: capabilities for structuring and

]

representation, flexibility for design for change, and fully automatic generation of web pages. From the design for change viewpoint, the main feature is explicit separation of concerns, especially separation of domain structure from its content.

6. Evaluation and discussion

The IT-based learning community is, perhaps, one of the largest end-user communities in the world now. Various kinds of eLearning environments are centred on Portals with the involvement of end-users. As various studies show, EDU Portals are not so much different in structural and functional aspects from the commercial ones. Thus, the principle solutions can be and are actually based on the generic portal architecture and the generic processes within the architecture. But the end-user requirements for implementing the processes, the Portal content and environments vary extremely. Because of the architectural complexity, rapid evolution of the Internet technologies and dynamism (social, pedagogical, technical) within the learning community per se, the design for change paradigm in Portal development seems to be the most perspective and attractive technical solution for creating the environments for Knowledge-Based Society.

We have considered only some technical aspects of design for change in the context of meta-design and formulated the Portal development as a task in which the domain analysis for variability is the main focus. We have suggested a variability model and discussed a wide context of the processes in order to be able to discover the model and implement it. This context includes Portal Domain Analysis, Component-Based Design, and Reuse. The most crucial part of the proposed model is obtaining the relationships between different aspects of generic sub-processes.

The variability document is a roadmap for supporting the meta-designer in designing for future changes. As different aspects (parameters) of the sub-processes are overlapped and scattered, it is practically impossible to build a unified and precisely described the relationship document. Thus there is a great variability within the relationships, too. We call them the dynamic mappings. What aspects should be selected for inclusion into variability document, mostly depend on designer's view, stated requirements, choice, experience, strategy, given resources, etc.

The variability document systematizes the processes and allows foreseeing different variants for components, thus bringing a perspective for meta-design. As the domain is not mature enough, we have suggested focusing on most crucial aspects, such as interface variability, in Portal development.

The other side of the problem is model's implementation. The meta-programming technique seems suit well from the perspective of meta-design as it brings a generative technology easy to implement at use time. The "hard implementation" of variability, of course, is possible too but for the narrow spectrum of parameter values only. The generative technology is superior for large-scale variability as it takes place in meta-design.

7. Conclusions

We have formulated the design for change problem in the context of meta-design as a problem, which focuses on variability analysis in the given domain. For the educational portal design domain, we have proposed a variability model based on the generic processes and sub-processes known in the Portal developments. Processes and sub-processes describe functionality of the system to be built.

The most crucial part of the model is relationships between the sub-processes, which we obtain through analysis and meta-modelling. The relationships give understanding of the structure of the system.

The variability document, even not full and precise, systematizes the development process, brings parameter values for sub-processes and allows foreseeing different variants for components, thus bringing a perspective for meta-design. The generative technology using meta-programming is superior for large-scale variability as it takes place in meta-design.

References

[1] **A. Sutcliffe.** User-centred design for multimedia applications, Multimedia Computing and Systems, *IEEE International Conference on GUI*, *Vol.*1, 1999, pp. 116-123.

[2] **A.H. Namioka, C. Rao.** Introduction to participatory design. *D. Wixon and J. Ramey (Eds.), Field Methods Casebook for Software Design*, *NY: Wiley*, 1996, 283-299.

[3] **A. Sutcliffe, N. Mehandjiev.** End-user development, *Communications of the ACM*, *Sept*. 2004, 47(9), 31-32.

[4] **G. Fisher, E. Giaccardi.** Y. Ye, A.G. Sutcliffe, N. Mehandjiev, Meta-design: A manifesto for end-user development. *Communications of the ACM, Septemb*er 2004, 47(9), 33-37.

[5] **G. Fisher, E. Giaccardi.** Meta-Design: A Framework for the Future End-User Development. *H. Liereman, F. Paterno, V. Wulf, (Eds.). End User Development – Empowering People to Flexibly Employ Advanced Information and Communication Technology*, *Kluwer Academic Publishers*, 2005.

[6] **J. Reinhardt, H.F. Friedrich, J. Wedekind, B. Gaiser, S. Panke.** e-teaching. org: Qualifying academic teachers for the next decade. A pragmatic approach. *J. Cook (Ed.), Blue skies and pragmatism: Learning technologies for the next decade*, *Devon, UK: University of Exeter*, 2004, 36-53.

[7] **L. Nakayama, R. Vicari, H. Coelho.** An Information Retrieving Service for Distance Learning. *IPSI Transactions on Internet Research*, 1 (1), 2005, 49-56.

[8] **V. Štuikys, R. Damaševičius.** Metaprogramming Techniques for Designing Embedded Components for Ambient Intelligence. *T. Basten, M. Geilen, H. de Groot (eds.), Ambient Intelligence: Impact on Embedded System Design, Kluwer Academic Publishers, Boston,* 2003, 229-250.

[9] **W.C. Lim.** Managing Software Reuse. *Prentice Hall*, 1998.

[10] **T. M. Pigoski.** Software Maintenance, Guide to the Software Engineering Body of Knowledge. *IEEE-Trial Version* 1.00-*May* 2001.

[11] **J. Coplien, D. Hoffman, D. Weiss.** Commonality and Variability in Software Engineering. *IEEE Software* 15(6), 1998, 37-45.

[12] **M. Van Harmelen.** Object Modelling and User Interface Design: Designing Interactive Systems. *Addison Wesley Professional*, 2001.

[13] **D.L. Parnas.** On the Design and Development of Program Families. *IEEE Transactions on Software Engineering*, SE-5(2), 1976, 1-9.

[14] **D. Weiss, R. Lai.** Software Product Line Engineering. *Addison-Wesley Longman*, 1999.

[15] **M. Mecella, B. Pernici.** Designing wrapper components for e-services in integrating heterogeneous systems. *VLDB Journal* 10(1), 2001, 2-15.

[16] **S.L. Pfleeger.** The Nature of System Change. *IEEE Software*, May/June, 87-91.

[17] **K.-C. Kang.** Feature-Oriented Domain Analysis for Software Reuse. *Proc. of Joint Conference on Software Engineering (JCSE'93), November* 17-19, 1993, 389-395.

[18] **J. Sametinger.** Software Reuse with Reusable Components. *Springer Verlag*, 1997.

[19] **T.K. Hazra.** Building Enterprise Portals: Principles to Practice. *Proc. of International Conference on Software Engineering ICSE'02, May* 19-25, *Orlando, Florida, USA*, 2002, 623-633.

[20] **C. Alexander.** The Timeless Way of Building. *Oxford Univ. Press*, 1979.

[21] **M. Gries.** Methods for Evaluating and Covering the Design Space during Early Design Development. *Integration, the VLSI Journal*, *Elsevier Science*, 2004,

[22] **Z. Zhang, K. Lyytinen.** A Framework for Component Reuse in a Metamodeling Based Software Development. *Requirements Engineering Journal* 6 (2), 2001, 116 - 131.

[23] **P. Szekely.** Retrospective and Challenges for Model-Based Interface Development. *Proc. of the 2nd International Workshop on Computer-Aided Design of User Interfaces. Namur Univ. Press, Namur*, 1996.

[24] **E. Aarts, E. Roovers.** Embedded System Design Issues in Ambient Intelligence. *In T. Basten, M. Geilen, H. de Groot (eds.). Ambient Intelligence: Impact on Embedded System Design. Kluwer Academic Publishers*, 2003.

[25] **T. Makimoto, D. Manners.** Digital Nomad. *John Wiley and Sons*, 1997.

[26] **W.E. Hefley, D. Murray.** Intelligent user interfaces. *Proc. of the International Workshop on Intelligent User Interfaces*, *January* 4-7, 1993, *Orlando, Florida, USA*, pp. 3-10.

[27] **J. Eisenstein, J. Vanderdonckt, A. Puerta.** Applying Model-Based Techniques to the Development of UIs for Mobile Computers. *Proc. International Conference on Intelligent User Interfaces: IUI 2001*. Santa Fe, *NM: ACM Press., January* 14-17, 2001, 69-76.

[28] **V. Štuikys, R. Damaševičius, M. Montvilas.** A Metaprogramming-Based model for Generation of the eLearning-Oriented WEB Pages. *T. Boyle, P. Oriogun, A. Pakstas (Eds.), Proc. of the 2nd Int. Conf. on Information Technology: Research and Education (ITRE* 2004*), June* 28-July 1, 2004, *London, England*, 64-68.

Received April 2006.