Towards Meta-Design for E-Business: Experiences & Challenges

Claudio Muscogiuri, Claudia Niederée, Matthias Hemmje, Michael Fuchs

FhG - IPSI
D-64293 Darmstadt, Germany

[muscogiuri, niederée, hemmje, fuchs]@ipsi.fhg.de

Abstract

One way to address the challenge of application evolution management in the highly competitive and quickly changing area of e-business is user empowerment through meta-design support: By integrating user-friendly and, task-oriented tools for Web application design and evolution into the Web application itself, users are empowered to act as application co-designers and to directly control the creation and the evolution of the Web application.

The development and evolution of Web applications requires a coupling of entities from at least two different domains: 1) the Web application development domain 2) the application domain, which includes business objects, relationships and business processes of the application domain under consideration.

A systematic approach to a flexible coupling of the two domains and their entities is based on the explicit modeling of the different domains. In a second step, a mapping from objects of the Web application user interface to business objects or business process subtasks of the application domain has to be enabled. Therefore, a third model is needed that systematically specifies the process of mapping between the two domains. This user-centered approach to application system evolution has been evaluated in first trials of the FAIRWIS project implementing an e-business solution with user empowerment for the trade fair business. Exploiting heuristics from the Web application design as well as from the application domain task-oriented authoring tools for Web application components combined with business process subtask packages form a powerful and comfortable domain oriented design environment for user empowerment in the FAIRWIS system.

Evaluation of the FAIRWIS system by different fair organizers showed significant viability and users' acceptance of the approach, but also showed up challenges for the next generation of such user-oriented design empowerment support.

1. Introduction

In order to support modern business companies to take part in strong competition in a rapidly developing global market, a new generation of software systems have been developed over the last few years. Such systems are referred as e-business solutions. Technologically, they can be classified as Web applications. Developing a Web application for migrating a traditional business process into a new e-business process poses the main challenge of how the information, communication, and knowledge management functionalities of such a Web application should be designed and efficiently implemented.

Due to the particularly dynamic global market in which the e-business solution is positioned, the requirements toward the system change as frequently as the market's tastes and trends do.

It is our goal to investigate how far this constant Web application adaptation task can be shifted from a programming task, for which a software expert is necessary, to a design task, which can be performed by a so-called *power user* [1] of the respective application domain. We believe, that a wide range of evolutionary changes can be supported in the Web application itself by providing adaptation support mechanisms, avoiding in this way time-consuming software re-implementation cycles and the associated communication overhead.

For this purpose Web application *meta-design* [2]) is required, i.e., Web applications have to provide integrated support for participating in their own design process. To achieve this goal, powerful Web application authoring support is necessary. It empowers special users of the Web application, the power users, not only to use the application but also to adapt it to changing requirements. The design of a well-founded Web application authoring support has to be based on a clear conceptual understanding of the supported evolution process including the evolution process of the underlying domains.

In the focus of our approach there are, thus, conceptual models for the involved domains. In addition to the application domain model capturing the typical entities, relationships and processes of the application domain we also need a model of the domain of Web application design construction. A third model is needed to express the mapping relationship between entities from the first two models that have to be established in Web application development for a specific domain.

In FAIRWIS, an R&D project founded by the European Union currently under development, we are applying our meta-design approach by providing an integrated Web-based advanced information and communication system for supporting the trade fair business sector. During the FAIRWIS project and related projects we investigated

- the exploitation of heuristics in the involved domains to provide high-level, best practise packages for frequent domain-specific sub-tasks and
- the role of domain modelling for system evolution support

In the current phase of the iterative software development process, the FAIRWIS project platform already includes Web application authoring support based on the results of the aforementioned investigations. In fact, within the context of the FAIRWIS project, trade fair organizer enterprises are considered as communities of designers in a particular application domain. The computational environment hosting the Web application that supports the on-line trade fair processes has been enriched by an ad-hoc authoring environment in which the fairs organizer's stakeholders can participate in the activities of defining and evolving the Web application, as they need it for supporting their business-process with respect to

- basic user interface layout and navigation support
- coupling of user interface and business objects
- infrastructure service support
- application domain specific support packages

The rest of this paper is structured as follows: Our approach to Web application meta-design and user empowerment is presented in section 2. The support for user empowerment in the FAIRWIS is described in section 3. An overview of related work is given in section 4. Since this is ongoing research, section 5 summarizes the lessons we learned from this and similar projects concerning Web application evolution and discusses our plans for the next steps on the way to improved meta-design for e-business Web applications..

2. Dynamic Evolution Support through User Empowerment

2.1. Iterative Application Evolution

The development of a larger software system is a complex process that in general requires several iterations until the produced system fulfils the requirements of the intended user community [9]. This is especially true for Web applications implementing e-business solutions, for which the requirement analysis and design phase necessitate the involvement of a variety of stakeholders [3], including, but not necessarily limited to marketing people, consultants, creative designers, customer supports, vendors, lawyers, business executives, system designers and developers.

Even when the developed system has reached a state upon which all stakeholders agree and is finally deployed, the system change process does not stop. Business process redesign due to new end user requirements, user feedback, new legal regulations, competitive reasons, etc. may require an adaptation of the associated Web application to the new process.

Designing and implementing a new system also contributes to the creation of a shared understanding between all the stakeholders. Once the users find themselves working together using the same system and exploiting this shared understanding, this can lead to "new insights, new ideas, and new artefacts" [2]. This provides an important additional source for system evolution requests.

System evolution, thus, is an integral part of the operational phase of a Web application's lifecycle and has to be supported efficiently. The traditional software development cycle delays the integration of change requirements coming up in the operational phase to the next software release. This delay can be unacceptable in the highly competitive market of e-business.

As an alternative or complement to this software evolution approach the user can be empowered to implement at least certain classes of changes by himself.

2.2. User Empowerment

If we want to convert at least part of the users from normal users into power users that act as co-designers and developers in system customisation and evolution, adequate system support is required for this task. Integrating mechanisms for system evolution steered by the user into the system leads to the concept of meta-design [2]: User empowerment services have to be defined that are themselves intended for modifying the design of the system.

For real user empowerment we are interested in system modification that goes beyond user interface customisation: The user shall be able to manipulate system functionality adapting it to the underlying business process. Such modifications are normally reserved to skilled software developers. Since the power users are members of the application domain, we may not assume programming skills. Tackling this virtual conflict user empowerment is based on the following meta-design concepts in our approach:

- domain modelling and knowledge management for all involved domains
- the exploitation of domain heuristics

The impact of domain heuristics and domain modelling for a user empowerment that is flexible, but also task-oriented and well-rooted in the application domain is discussed in this section. The use of task specific design support [1] instead of general programming languages provides user-friendly interfaces for the development task.

It has to be noted here, that we do not expect the power user to develop an entire e-business solution from scratch on his own. The focus is on small incremental changes and adaptations in an already operational system

2.2.1. Domain Modeling and System Evolution

The development and evolution of Web applications require a coupling of entities from at least two different domains:

- the domain of Web application development,
- the application domain, which includes business objects, relationships and business processes of the application domain under consideration.

A systematic approach to a flexible coupling is the modelling of the different domains. In a second step, a model that systematically specifies the process of mapping between from objects of the Web application user interface to business objects or business process subtasks of the

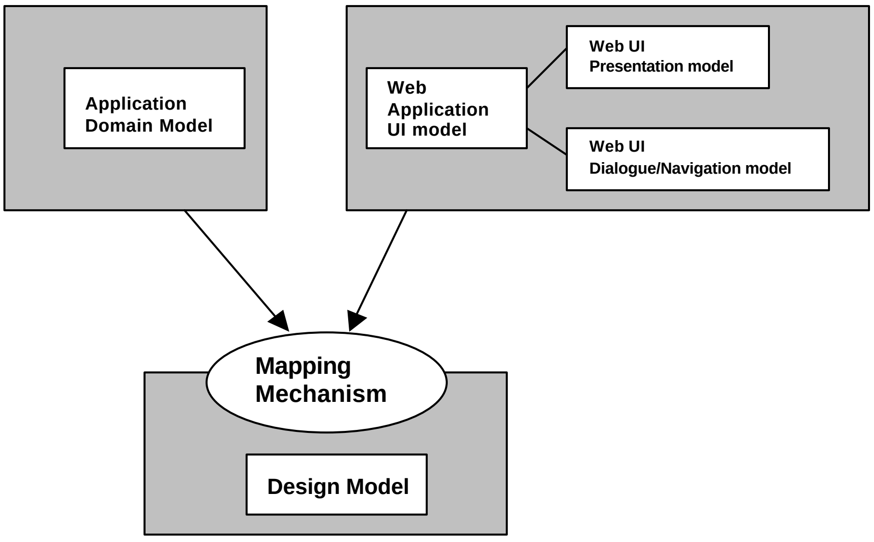

Figure 1: *a model that systematically specifies the process of mapping between from objects of the Web application user interface to business objects or business process subtasks of the application domain has to be enabled.*

application domain has to be enabled (see Figure 1).

In addition to enabling the discussed forms of user empowerment, modelling of the application domain and making it available to the users (power users as well as normal users) also contributes improved communication between all community members and a deeper domain understanding.

2.2.2. The Role of Domain Heuristics

Processes in apparently all domains are driven by heuristics and best practices, which are based on individual and community experiences with the same or similar processes. These heuristics can be made explicit, as in the case of published guidelines, or implicit, as in the case of individual experience.

In the case of Web application meta-design heuristics from different domains are available. In the domain of Web interface design heuristics are collected in guidelines for "good" Web page design. In e-business application domains heuristics for the efficient design of the central business processes are available. In the trade fair domain, for example, processes like booking of stand services are well-understood subtasks of organizing a fair.

In our approach we plan to exploit heuristics from the involved domains to enrich the power user's work environment in different ways:

- Customisable subtask packages, that implement heuristics like best practises, frequently used patterns, and standardized business process steps provide high level building blocks for Web application construction and evolution.
- Consulting, wizards and critiquing systems can be improved by evaluating construction heuristics and proposing adequate process steps and design decisions to the user.
- Processing and structuring templates like navigation menus that are based on usage heuristics are offered as frameworks that can be customized and instantiated.

3. The FAIRWIS Experience

3.1. The FAIRWIS Project

FAIRWIS is an EU funded project that aims at offering on-line innovative services to support business processes of both real and virtual trade fairs. An application design environment has been integrated into the FAIRWIS system to provide support for the user empowerment approach outlined in the previous section enabling trade fair organizers to adapt and evolve their own systems.

This publication focuses on the meta-design components for user empowerment in the FAIRWIS system and the experience gained with these components in an evaluation phase. A more detailed description of the FAIRWIS project can be found in [6].

3.2. The FAIRWIS Application Design Environment

FAIRWIS system's integrated design environment consists

- of a set of authoring tools for the design and evolution of the components of an e-business Web application with a multilingual user interface and
- of a set of subtask packages for typical business processes in the trade fair context.

The authoring tools are called *managers* and are intended for system developers as well as for power users.

The managers can be organized in four groups:

- Managers for basic user interface layout and navigation support,
- Managers for the coupling of user interface elements with business object,
- Managers for infrastructure service support, and
- Managers for application domain specific support packages.

3.2.1. Basic User Interface Layout and Navigation Support

Referring to the model in Figure 1 introduced in section 2.2.1, this set of authoring tools implements the fundamental Web UI Navigation model [7], [8] and part of the Web UI Presentation model [7], [8]:

Menu Manager: The Menu Manager is used to create hierarchies of navigational, multilingual menu and sub-menu entries. For each considered language, a separate hierarchy of menu entries can be defined. For each menu entry entity, power users can define its visibility to user groups, the name and type of the Web page to which the menu entry links and other menu entry properties like e.g. onMouse events, pre or post menu entry text, etc.

Page Manager: The Page Manager can be used for the creation and editing of static Web pages providing a versioning mechanism. It supports the upload of Web pages produced using other off-the-shelf tools and the application of templates.

Image Manager: The Image Manager enables the upload of images and makes them available for the other Managers.

Color Manager: Using the Color Manager, the power users can define the basic look and feel of the Web application. The user interface has been conceptually structured into six main components (e.g. header, footer, menu entities), for which layout properties can be defined by the Color Manager.

3.2.2. Coupling of User Interface and Business Objects

Forms, as they are known from HTML pages, are an important way of user interaction in Web applications. If they are used as user interfaces in Web application a coupling between components of the Web page and business objects of the underlying application domain have to be established.

The Form Manager enables to define a mapping between the user interface components and application domain by defining the structure and basic layout of dynamic Web pages (forms) (see Figure 1). The class diagram of the model underlying the Form Manager is given in .Figure 2.

Dynamic Web pages are pages displaying application (database) state, enabling on-line users to execute business workflow steps through input, confirmation and data processing. Using the Form Manager, power users can create the dynamic Web pages for the Web application under design, specifying the static and dynamic properties of their components (field elements, buttons, checkboxes…). For each component the power user can specify:

- Element type: For each element composing the page, predefined templates can be used and new ones can be created (e.g. checkbox, check box list, checkbox list with default, select list, radio, radio list, etc.). Each template defines both static layout properties from the Presentation model (see Figure 1) and specific business logic that implements a coupling mechanism with application specific code. The element type is an instance of the Element class (see Figure 2)
- Field content: the data objects to which a Web design element is related, and on which it will carry out the business logic defined for the assigned "element type". For example, given the element type is a select list for an input field, the element content property will be a list of settings the power user specifies for defining the available default values in the list (e.g. list of countries), and how the input value will be used in the business logic

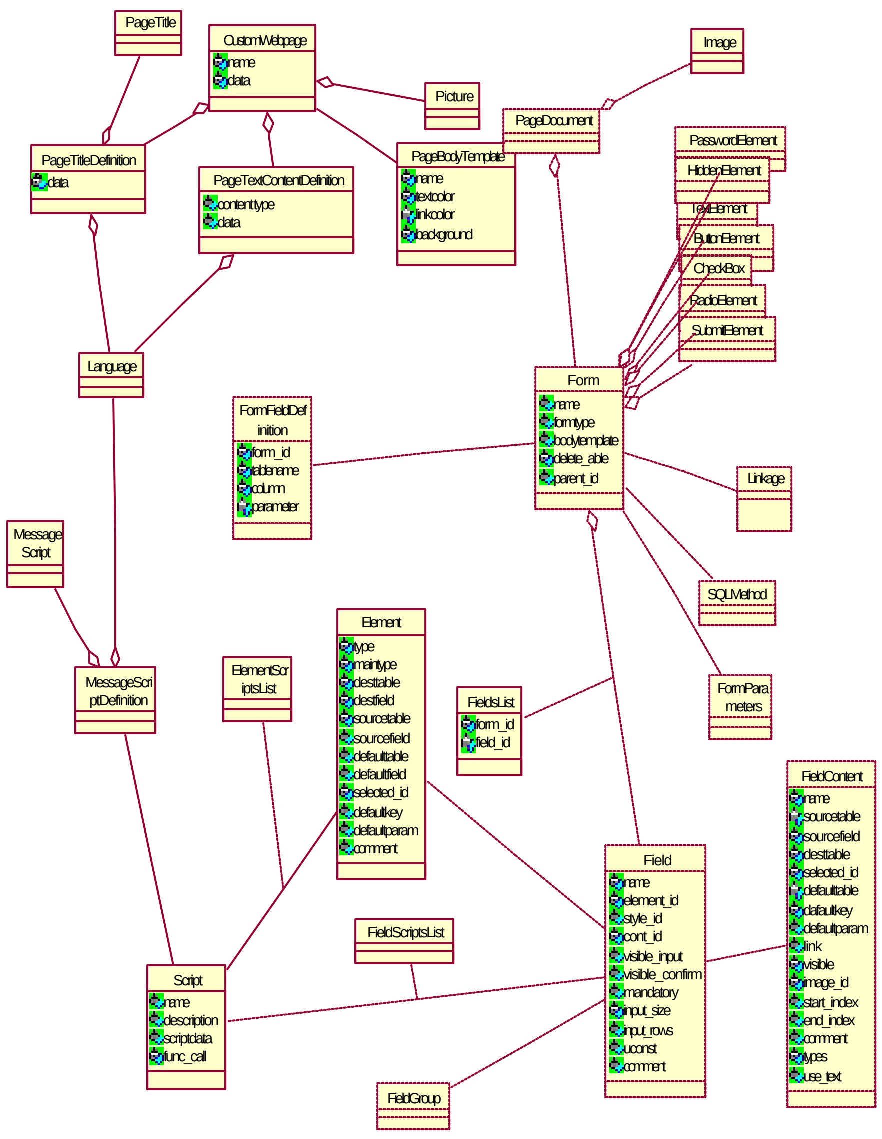

Figure 2: *The model for the Form Manager*

(e.g. set the city value in the user's personal data). The field content is an instance of the FieldContent class (see Figure 2)

- Related "script" object: a set of script objects is available, implementing additional business logics. Some examples, among others are: print, submit, check mandatory fields before submitting, change user group after a business transaction, dynamic link, print special content areas. The related script object is an instance of the Script class (see Figure 2)

3.2.3. Infrastructure service support

User Manager, Communication Manager (Mail Manager): Power users can manage user data and all related data object from the Application Domain model (see Figure 1). Power users can query the business data, can create and manage predefined dynamic queries. A user profile editor is integrated in this Manager, in which power users can manage and edit user data (e.g. name, age, address) and related business objects' data, such as stands data, account data (login, password), and usergroup in the users hierarchy. The Mailing lists can be set up by means of dynamic queries on the user database, segmenting parts according to their profiles and interests. Mails can be delivered using regular e-mail. Alternatively, messages can be displayed individually to on-line users registered to the Web Application.

3.2.4. Application domain specific support packages

The booking of service products and the management of trade fair events have been identified as important business process subtasks in the trade fair domain during the FAIRWIS project. Therefore, customisable support packages have been implemented for these subtasks.

Booking Manager: Power users can use the Booking Manager to set the peculiar characteristics (such as description, fees, availability, and currencies) for each family of services offered and that on-line users can book on-line and list in the shopping cart pages. For each family of services a corresponding Complex Field is created.

The Complex Fields are a special kind of dynamic page elements. Such fields realize the part of relationship between special entities in the fair domain (the services) and Web page elements providing the right interface for the on-line booking of services. The behavior of the complex field, implementing the business logic, is automatically created in an associated *FieldScript* object (instance of the Field class in Figure 2).

The services booking Web page for a family of services is created, like the other Web pages, by power users using the Form Manager.

Program Manager: Within this Manager, power users can design dynamic catalogue pages, as in the case of the services booking Web pages, by editing the properties of ad hoc multidimensional hierarchy of business object, like exhibitors catalogue information, and event information (e.g. conferences and workshops). Visitors can examine the events on-line and add them to their personal program in their personal area, like the exhibitors add booked services to the shopping cart.

4. Related Works

An interesting form of task-specific design support are DODEs (Domain-oriented Design Environments) [4]. Essentially, a domain oriented design environment includes the following components

a) a construction kit providing a palette of domain building blocks

b) an argumentative support containing issues, answer, and arguments about the design domain and the design rationale

c) a catalogue consisting of a collection of pre-stored designs

d) a specification component supporting the interaction among stakeholder

e) a simulation component (carrying out "what-if" games)

Typically, domain oriented design environments provide tools for creating design representations, information repositories for storing domain information knowledge, and knowledge based mechanisms that link the design representations and the stored domain information.

An extended analysis of the designers' activities and their tools has been performed also in [5]. Such analysis confirms that design communities gradually construct their domain by defining domain objects, creating and evolving multiple representation of the domain objects, and establishing complex relationships between objects through their representation (domain construction process); thus, it confirms the necessity to support domain modelling support in design environments.

It is our aim to build a design environment that is not restricted to one application domain but usable for an entire class of systems, namely e-business solutions with community support in different application domain like trade fair business, e-learning, etc. For this purpose, we plan to generalize the DODE approach by factoring domain-specific knowledge into a domain model and developing a meta-model for the interaction of the domain model with the components of the design environment.

As explained in the previous sections of the paper, all our efforts are in the direction of investigating on how to support the user in modelling objects in different domain and in defining the mapping properties among objects from the different domains in a declarative way. The validity of such approach is confirmed by analysing best practice in the UI designer community [7]. For instance, following the so-called Model Based Interface Development paradigm [8]**;** model-based UI tools support the design and the development of Uis in a professional and systematic way allowing designers/developers to specify UI design by a) managing entities and relationship in the following domains: Domain model, User model, Presentation model, Dialog model, Task model, and b) setting mapping properties among entities from the different domains. The study of such design environments confirm that the model based paradigm is a good fit for Internet-based UI that rely on the use of modular components

5. Conclusion and Future Work

5.1. Lessons learned from FAIRWIS

The FAIRWIS system is already productive in a public trial site, and real trade fair organisers are using the design environment illustrated in section 3 to set up Web applications supporting events they organise.

As the project partners expected system acceptance is high, because of the deep involvement of trade fair organizers and associations into the process of the requirement analysis and design. In addition, trial site evaluation showed encouraging synergies between the stakeholders. Members of the stakeholder communities that have developed expertise in specific off-the-shelf tools (e.g. CAD people, marketing people) flexibly bring their expertise and working results into the distributed common design environment, sharing in this way design products and knowledge. At the same time, the FAIRWIS prototype is permitting domain professionals to easily acquire contextualized IT knowledge.

The evaluation of the FAIRWIS system also pointed out some shortcomings of the prototype functionality. Driven by variances in the underlying trade fair business process and inspired by the exp erience with the FAIRWIS system modifications of the domain model underlying the system like the insertion of new user types were desired by some of the evaluating trade fair organizers. Making changes to tools and the underlying domain models requires technical knowledge and actual programming efforts. This limitation still ends up in situation where even power users are in need of collaboration with system developers to assist them in redesign activities.

The design and evolution of a Web application is a complex project involving different stakeholders. Therefore, the need for tools supporting the management of the Web application construction workflow has been identified as a further system requirement during the evaluation.

In summary the claim of Fisher [2], that system involvement leads to "new insights, new ideas, and new artefacts" is confirmed by the evaluation results. Becoming acquainted with the system, the power users discover new evolution ideas that can be covered only by a new generation of more flexible authoring tools.

5.2. User Empowerment - Next Generation

As confirmed from the users' evaluation in FAIRWIS, explicit domain modelling and domain construction support is needed by the user in order to allow them to able to create and change entities and relationships in the domain (domain construction frameworks [5]). The currently implemented mapping mechanism between Application Domain model and Web Application UI model (see Figure 1) allows the generation of the UI element by matching data object with element types as explained in section 3.2.2. However, the users don't think in term data model but in term of business object [10]; so an explicit support for the application domain modelling could allow user to effectively express the element that must be displayed in the interface in term of domain objects.

Introducing a specific support for the domain modelling implies the introduction of a Domain Adaptor Layer to map business object at the UI level with data object on the business layer [10]. This additional layer would also help to more effectively integrate information sources from the users' back office system.

Besides, for a large Web application design practice, typically the designers produce one navigation map per actor, one for each use case identifying pages to which the users will navigate. As the pages are identified, the designers then describe the information the pages handle [3]. Analogously, the Manager tools in the FAIRWIS design environment should be extended allowing power users to define the business process workflow, the business objects affected in each step of the business workflow, one navigation map per user group for each step in the business process, identifying the pages to which the user group will navigate while performing the step in the business workflow.

6. Acknowledgements

The support of European Commission through grant FAIRWIS IST-1999-12641 is acknowledged. We are also grateful to the work carried out by the other partners of the FAIRWIS project.

7. References

[1] Nardi B.A., *A Small Matter of Programming,* MIT Press, Cambridge, Mass., 1993

[2] Fischer G., *Social Creativity, Symmetry of Ignorance and Meta-Design*, Knowledge-Based Systems J., vol. 13, nos. 7-8, Dec. 2000, pp. 527-537

[3] A rational Software and Context Integration white paper, *Building Web solution with the Rational Unified Process: Unifying the Creative Design Process and the Software Engineering process*, http://www.rational.com

[4] Fischer, G. (1994), *Domain-Oriented Design Environments*, Automated Software Engineering, 1(2), 1994 pp. 177-203

[5] Sumner T. R. (1995), *Designers and their tools: Computer Support for Domain Construction*, University of Colorado at Boulder, Ph.D. Dissertation, Dept. of Computer Science, 1995

[6] Muscogiuri C., Jaeschke G., Paradiso A., Hemmje M. (2002), *FAIRWIS: An Integrated System offering Trade Fair Web-based Information Services – A R&D Case Study*, in: Proceedings of the 35th Hawaii International Conference on System Sciences, Hawaii, Big Island, IEEE Press

[7] Pinheiro da Silva P. (2000), *User Interface Declarative Models and Development Environments: A Survey* Proceedings of DSV-IS2000

[8] Puerta A. R.. (1997), *A Model-based Interface Development Environment*, IEEE Software, pages 40--47, July/August 1997

[9] Kruchten P, *The Rational Unified Process: An Introduction*, The Addison-Wesley Object Technology Series, 2000

[10] Anderson, D.J., *TUPIS2000 – Notes on Interaction spaces from UML2000*, Conference TUPIS Notes Report: uidesign.net, http://www.uidesign.net/2000/conference/TUPISreport.html