Chapter 7

Conclusion and Future Work

"It is never a mistake to say good-bye."[1]

This dissertation dealt with the design and development process of context-adaptive Web applications. It reviewed the state of the art in the fields of adaptive hypermedia and Web engineering, identified main shortcomings of existing solutions, and proposed a component-based approach for engineering adaptive Web sites. For the systematic realization of adaptive Web presentations from reusable components, a model-based authoring process was designed and constructively validated based on a prototypically realized visual component authoring tool, an automatic transformation facility for the model-driven generation of adaptive component structures, as well as a number of example applications. Finally, it was shown how the lessons learned from engineering component-based adaptive Web sites can be generalized in order to add adaptation to existing (not component-based) Web applications.

This final chapter reflects on the results of this dissertation. First, Section 7.1 recapitulates the work presented in the previous chapters. Then, Section 7.2 discusses the dissertation's main scientific contributions and achievements, but also mentions its limitations and boundaries. Finally, Section 7.3 presents possible directions of future work to be carried out on the foundations of this thesis.

7.1 Summary of the Chapters and their Contributions

Chapter 1 - Introduction

Chapter 1 introduced the background and motivation of this work. It pointed out that personalization and device independence are crucial issues of today's Web development, but also identified a number of problems and shortcomings regarding the design, development, and delivery of such adaptive Web applications. The vision of the work, the main problems to be solved, basic research theses, as well as the goals to be achieved were presented. Moreover, a short outline of the dissertation and the structure of its chapters was given.

Chapter 2 - Adaptive Hypermedia and Web-based Systems

The goal of Chapter 2 was to provide necessary background information for the reader on the field of adaptive hypermedia and Web-based applications. It stated main definitions and described the most significant methods, techniques, and application areas of hypermedia adaptation. Furthermore, the most important reference models aimed at identifying the

[1] Kurt Vonnegut, Jr.: The Books of Bokonon (from Cat's Cradle, 1963)

most common features of adaptive hypermedia and Web applications were also summarized in detail.

The bases of this chapter were fundamental works and existing surveys on the field of hypermedia, adaptation, user modeling, and context-awareness. Its contribution is a compact introduction to the field of adaptive and context-aware hypermedia systems and Web applications.

Chapter 3 - Development of Adaptive Web Applications: State of the Art

After summarizing the fundamentals of adaptive hypermedia, Chapter 3 analyzed existing approaches aimed at engineering adaptive Web applications. The focus of investigations was on two different aspects: 1) component-based and document-centric approaches aimed at the implementation of Web applications, 2) and model-based methodologies focusing on their structured design and development. From both fields the most prominent approaches were summarized. While examining related work, a special focus was put on the question of how adaptation and personalization are supported.

The basis of this chapter was a thorough examination of the recent years' scientific literature on adaptive hypermedia, Web engineering, and multimedia engineering. It pointed out the importance of declarative, component- and document-centric approaches, yet identified the lacking support for the efficient creation of adaptive multimedia Web presentations from reusable and configurable implementation entities.

Chapter 4 - A Concern-Oriented Component Model for Adaptive Web Applications

Chapter 4 presented a concern-oriented component model for component-based adaptive Web documents. First, basic requirements towards the model were discussed and the concept of declarative document components was introduced. Then, a component-based format for adaptive dynamic Web documents and its XML-based description language were presented. The different abstraction levels of document components, their support for adaptation, as well as the concept of document component templates are explained by examples. Furthermore, a modular architecture for the on-the-fly publishing and adaptation of component-based Web documents was described. Finally, the description of the model's selected benefits rounded off this chapter. The contributions of Chapter 4 are the:

- design of a component-oriented XML-based document model for dynamic adaptive Web documents;
- compact introduction and overview of the model's language constructs and its XML-based description language.

The main research results of Chapter 4 were published in a number of international publications: [Fiala et al. 2003b, Fiala et al. 2003a, Fiala and Meissner 2003, Fiala et al. 2003c, Hinz and Fiala 2005]

Chapter 5 - The Authoring Process and its Tool Support

After presenting the component-based document model, Chapter 5 dealt with the authoring process of component-based adaptive Web applications and its tool support. It discussed different application scenarios, but put a main focus on the development of data-driven adaptive

Web presentations from reusable document components. First, a possible structured development process by adopting and extending the Hera design methodology to the context of component-based Web engineering was presented. Second, the AMACONTBuilder, a modular authoring tool for the intuitive graphical authoring of component-based Web documents was introduced. Selected editor plugins of the AMACONTBuilder were presented from the point of view of component authors. Moreover, it was also shown how the overall process of design and implementation can be even automated, by automatically generating a component-based adaptive Web application from high-level design specifications in a model-driven way. Finally, the different authoring approaches were discussed, and example applications were described. The chapter's contributions are the:

- adoption of a structured design methodology for the development process of component-based adaptive Web applications;
- design and prototypical implementation of a visual authoring tool for component-based adaptive Web applications;
- extension of the Hera methodology by design and RDF(S)-based formalization of an adaptive presentation model;
- design and implementation of a solution for automatically translating high-level model specifications (design artefacts) to a component-based implementation supporting different aspects of static and dynamic adaptation;
- validation of the different authoring approaches by the development of a number of component-based adaptive Web applications.

The research results described in Chapter 5 were previously described in several publications: [Fiala et al. 2004a, Fiala et al. 2004b, Frasincar et al. 2004, Hinz and Fiala 2004, Fiala et al. 2005]

Chapter 6 - A Generic Transcoding Tool for Making Web Applications Adaptive

While the combination of the component-based document format with a structured design and authoring process provides an efficient framework for developing adaptive Web-based systems, it assumes to develop adaptive Web applications from scratch. Therefore, Chapter 6 dealt with the research question of how the lessons learned from the preceding chapters can be generalized for adding adaptation to a broader range of Web presentations. It was recognized that adaptive Web applications can be reduced to a series of data transformations and that major parts of the adaptation-specific transformations can be separated from the rest of the applications. Thus, a flexible transcoding tool called the Generic Adaptation Component (GAC) was introduced.

First, the GAC's main architecture and most important application scenarios were described. Then, its RDF-based configuration language was introduced in detail, allowing to define both content adaptation and context data update rules. To prove the concept's feasibility, it was illustrated how GACs can be configured to add adaptation to an existing Web application. As the contributions of Chapter 6 we mention the:

- design of mechanisms for separating adaptation-specific transformations from adaptive Web applications;

- design and implementation of the Generic Adaptation Component (GAC), a generic Web transcoding component for making existing XML-based Web applications adaptable and adaptive;
- specification of an RDF-based declarative language for the application-independent definition of adaptation and interaction processing rules;
- illustration of the GAC's functionality based on a running example.

The Generic Adaptation Component and its application in different transcoding scenarios were published in a number of international publications: [Fiala and Houben 2005, Houben et al. 2005, Casteleyn et al. 2006a, Casteleyn et al. 2006b].

7.2 Discussion

The research theses formulated in the introduction of the dissertation served as its main guidelines and motivation. Nevertheless, note that some of them cannot be proven directly and can thus be only evaluated as a result of extensive long-term studies. This work provides a sound foundation for such investigations. On the other hand, the research goals derived from those theses (see Section 1.2) can be concretely compared with the results of the dissertation. As also described in the previous summary of chapters, those goals could be successfully accomplished.

The main goal to develop adaptive Web applications from reusable, configurable, and adaptable implementation artefacts was achieved by the design and development of a declarative, document-centric component model. Its underlying XML-based description language supports the definition of document components that encapsulate both separate application aspects (content, structure, semantics, navigation, presentation) and their corresponding adaptation issues on different abstraction levels. Consequently, its expressivity and reusability goes far beyond the possibilities of conventional Web document formats. Based on a number of examples, it could be shown that the component model is applicable for implementing the most important hypermedia adaptation techniques. Furthermore, it was illustrated how document components can be automatically transformed to traditional Web document formats, adapted to the appropriate user and his usage context.

Second, the applicability of the proposed component model in the overall engineering process of adaptive Web applications was also successfully demonstrated. Its combination with the model-based Hera-AMACONT methodology provides significant research benefits: 1) the thorough separation of concerns and the reuse of artefacts at both design and implementation, 2) the systematic consideration of different adaptation aspects at each development step, and 3) the design-time support for presentation layer adaptation, an issue that has not been sufficiently addressed by existing methodologies, previously. With the AMACONTBuilder a flexible visual tool was introduced to facilitate different authoring scenarios independent of any one specific methodology. Furthermore, the RDF(S)-based formalization of the Hera-AMACONT presentation model enabled even the automatic, model-driven generation of a component-based implementation. The main concepts and tools of the overall multi-stage authoring and document generation process could be presented and thus be constructively validated by a number of developed adaptive Web applications.

Finally, the research goal of extending existing XML-based Web applications with (additional) adaptation concerns was also achieved. Aided by the Generic Adaptation Component, Web developers have the possibility to decouple selected adaptation operations from the rest

of a Web applications and thus to specify them at a later stage, i.e. after the Web site has already been deployed. Furthermore, as this specification of adaptation is not intertwined with the regular Web application, it allows easy re-use of adaptation configurations for different Web sites. Again, the developed concepts could be demonstrated by an implementation based on example applications.

7.2.1 Scientific Contributions

Section 7.1 already summarized the contribution of each chapter of the dissertation. As mentioned there, the most important results were also published in a number of international publications. The following list recapitulates the most important scientific contributions of the overall work.

- Development of a novel, document-centric component model for context-adaptive Web presentations with a rigorous separation of different application and adaptation concerns on multiple component levels. Thereby, extensive use of XML standards for the homogeneous description of component properties, composition, interlinking, and adaptation.
- Design of a structured, model-based authoring process for component-based adaptive Web applications. Therefore, adoption and extension of existing Web design methods to the context of component-based Web engineering. Provision of design-time support for presentation-layer adaptation in Web site modeling.
- Model-driven generation of a component-based implementation based on an RDF(S)-based high-level Web design specification. Thus, automatic combination of the advantages of model-based Web design methodologies (e.g. high-level specification, thorough separation of concerns, etc.) with the benefits of component-based implementation techniques (such as reusability, configurability, or self-adaptation).
- Provision of a mechanism for decoupling selected adaptation operations from the rest of a Web application. Design of a declarative rule-based language for the application-independent description of adaptation operations. Development of a generic tool for the addition of (both static and dynamic) adaptation concerns to existing XML-based Web applications.

7.2.2 Limitations and Boundaries

The following list summarizes limitations and boundaries of this dissertation. Some of these limitations concern the proposed approach itself. However, there are also some issues, the thorough elaboration of which would go beyond the time scope of the thesis, and thus should be tackled in form of future work.

Limitations of the component-based document format: The adaptation facilities provided by the component-based document model cover the most important adaptation techniques (see Section 4.6.4), yet assume the author to predefine all possible adaptation variants (for content, navigation, presentation) already at authoring time. A more dynamic and transparent specification or automatic computation of adaptation variants (e.g. based on rules or behavior constraints to be evaluated at run-time) are not supported, as this was not the focus of this thesis.

Limitations of the supported adaptations: The adaptation addressed in this work focuses mainly on personalization, i.e. the adjustment of Web applications to individual users, their client devices, and context. Adaptation based on the behavior of all users (e.g. by examining common browsing patterns over a longer time scale) are not supported yet, and would necessitate the introduction of a "global user model" and appropriate modeling mechanisms. Furthermore, the supported adaptations are considered to be performed on the server, client-side adaptation possibilities (e.g. on top of the emerging AJAX technology [Gamperl 2006]) should be investigated within the scope of future work.

Limitations of the utilized context model: The CC/PP-based context model is perfectly suited for describing client capabilities and user preferences, but its flat two-level hierarchy does not support for more complex model descriptions utilizing concept hierarchies and/or relationships. In order to use more sophisticated user modeling mechanisms (e.g. based on semantic or probabilistic inferences), a more complex context model and an appropriate manipulation language based on Semantic Web technologies should be used.

Limitations of the presented Web engineering process: The Web engineering process described in Chapter 5 covers the phases of designing and implementing component-based adaptive Web applications. Yet, it does not consider important Web engineering issues, such as the maintenance, continuous updating (i.e. content management), or testing adaptive Web sites. These topics imply interesting research questions and should thus be more thoroughly investigated within the scope of future work.

Limitations of the AMACONTBuilder: The editor modules of the AMACONTBuilder support the most important steps of the authoring process, yet do not cover all facilities provided by the component-based format, and should thus be further developed, respectively.

Limitations of the example applications: The approach presented in this dissertation was proven by a number of example applications (see Section 5.4.2). However, even the largest example application (the Web Information System aimed presenting students' works) is rather middle-scaled in comparison to today's Web sites both in size and the amount of its visitors. For a more thorough evaluation of the authoring process, its tool support, and the run-time performance of the resulting applications further experiments on larger-scaled Web applications would be needed.

Limitations of the GAC: Currently, the RDF-based adaptation and update rules configuring the Generic Adaptation Component have to be edited by hand. The creation of a visual GAC configuration tool, the systematic authoring process of GAC-based adaptation rules on top of an underlying Web application, as well as the automatic generation of rules based on higher-level design specifications are not supported yet, as this was not the focus of this work.

7.3 Future Work

The work presented in this thesis provides different possibilities for further work. Whereas some of them concern straightforward extensions of the presented approach and its tool support, there are also possibilities to combine the results of the thesis with other research

areas. As the most important and interesting issues (parts of which are already addressed by ongoing work) the following can be mentioned:

Extensions to the Component-based Document Model

A possible extension of the component-based document model is the integration of additional adaptation facilities (e.g. automatic pagination or link/component sorting) into its repertoire of built-in language constructs. Furthermore, the support of streaming-based media content on the level of media components and its adaptive delivery by the document generation architecture should be also investigated. Finally, the development of transformation stylesheets for converting component-based adaptive Web presentations to alternative multimedia and print formats (such as SVG, Flash, MPEG-4, or PDF) or even fat-client user interfaces (e.g. Java Swing) is also an interesting extension issue.

Extensions to the AMACONTBuilder

The AMACONTBuilder introduced in Section 5.2 offers a number of graphical editor modules to demonstrate the authoring process of component-based adaptive Web presentations. Still, it does not lay claim to be a mature development environment covering all the facilities provided by the document model. Especially, editor modules for the intuitive authoring of adaptable interaction elements (e.g. form elements or other client-side interaction components) and for the visual configuration of the different user and context modeling facilities should be developed. Furthermore, tools aimed at the graphical specification of context models would be also desirable. Moreover, a thorough evaluation of the existing tools' correctness and usability by involving a number of test authors should be performed.

Furthermore, while the AMACONTBuilder is a graphical authoring tool oriented at the specifics of the concern-oriented component model, there is a need for an integrated development suit that covers all phases of the overall Web engineering process described in Chapter 5 (also shown in Figure 5.21), as well as other Web engineerings tasks that were not considered in this dissertation, such as requirement engineering, maintenance, or the test of Web applications. The vision is a modular development suit that enables designers, developers, content creators, etc. to choose from a repertoir of modeling, implementation, and content management tools that best fit the requirements of a given application. The basis of such a development suit could be a plug-in-oriented framework (e.g. based on the OSGI standard [OSG 2005]), allowing to integrate and combine different "tool components".

Support for Collaborative, Interdisciplinary Web Authoring

Chapter 5 introduced the authoring process of adaptive Web applications from a Web engineer's (model designer or application developer) point of view. Still, the overall process of designing, developing, maintaining, and evolving Web applications involves a number of experts from different domains (Web developers, graphics and layout designers, usability experts, content editors, etc.), whose work should be appropriately supported and coordinated.

This interdisciplinary authoring approach requires facilities for defining different author roles and the assignment of specific authoring workflows to those roles. First steps in this direction have been already undertaken by a prediploma thesis supervised by the author [Niederhausen 2005b], aimed at the definition and conducted execution of self-defined authoring workflows. Still, the interplay and coordination of different authoring roles should be investigated more thoroughly within the scope of a larger development project.

Support for Collaborative Adaptive Web Applications

An important characteristics of today's World Wide Web is its evolution to a communication and cooperation medium. Web applications are increasingly used in collaborative scenarios, supporting both asynchronous (e.g. Web annotations) and synchronous ways (e.g. shared Web browsing) of communication[2]. As the participants of such collaborative scenarios use typically different mobile devices, there is a need for a proper combination of Web collaboration and Web adaptation techniques. Still, even though there exist a number of Web collaboration solutions both from industry [Lin 2003, Ulbricht 2006] and the academic field [Greenberg and Roseman 1996, Esenther 2002], there are only very few approaches (such as [Han et al. 2000]) that explicitly address adaptation and device independence.

The combination of Web collaboration and Web adaptation techniques implies the investigation of a number of challenging research issues, among them the sharing and synchronization of different user and context models between parallel Web sessions, the adaptation of Web content to the varying interaction capabilities of cooperating mobile devices, multi-device Web browsing, or even the device dependent visualization of group awareness in collaborative Web sessions. Since the work presented in this thesis provides generic and reusable facilities for developing adaptive Web applications, it appears to be ideal for combination with existing Web collaboration techniques. We note that the industry project VCS (Virtual Consulting Services [@VCSProject]) aims at adopting the research results of this thesis to the field of ubiquitous personalized co-browsing.

Authoring Support for Adaptive Rich Media Web Applications

A main trend of today's WWW is the emergence of so-called Rich Internet Applications (RIAs [Duhl 2003]). RIAs (also often referred to as *rich media applications*) embed audio, video, 3D, and other highly interactive multimedia content, and can be seen as the fusion of the interactive and multimedia user interface functionality of desktop applications with Web applications. Their development necessitates to combine methods and tools of both the Web engineering and multimedia engineering [Bailey et al. 2001, Sauer and Engels 1999] disciplines. Furthermore, the device independent delivery of RIAs makes adaptation (of content, navigation, presentation, modality, interaction) to a crucial issue. Still, existing work on adaptation engineering mainly focuses on traditional hypermedia and Web content, there are only a few approaches (e.g. [Preciado et al. 2005, Bozzon et al. 2006]) addressing the specifics of RIAs.

The flexibility of the component-based document format introduced in this thesis allows for the easy integration of rich media elements into component-based adaptive Web applications. First steps in this research direction have been already undertaken by combining its adaptation facilities with the component-based 3D user interface description language of the CONTIGRA project in [Dachselt et al. 2006]. Furthermore, the genericity of the GAC approach promises to easily address adaptation in a large number of XML-based Web applications including rich media content. For this purpose an appropriate extension of the GAC's rule-based adaptation configuration language appears to be a feasible solution.

[2]This evolution of the WWW to a "second generation" of collaborative services is also often denoted as Web 2.0 [O'Reilly 2006].

Extensions to the Generic Adaptation Component

Chapter 6 introduced the GAC as a generic transcoding tool aimed at adding adaptation to XML-based Web applications. While it illustrated its application in a number of server-side transcoding scenarios, its utilization as a client-side component appears to be also an interesting future research direction. A client-side GAC could act as a portable private adaptation component providing a personalized view on Web applications for its users. As possible use cases we mention the adaptive management of personal bookmarks and links, the maintenance of users' private comments (annotations) attached to Web page fragments, or even the "portable" realization of a Web application's adaptive presentation and adaptation layer on a mobile device. For this purpose an intuitive, "easy-to-use" GAC configuration user interface should be developed, allowing users to set it up for adjusting selected Web applications to their personal preferences.

Another possible research issue is the development of a graphical GAC configuration module that can be plugged into existing XML-based authoring tools to visually create GAC rules. A promising approach seems to be the application of the so-called "transformation by example" (or programming by demonstration) paradigm [Koyanagi et al. 2000], that enables authors to visually define transformations on fragments of an exemplary XML document and to generalize the resulting rules, so that they can be applied to other documents slightly different from the original one. A similar solution was already introduced in [Ono et al. 2002], enabling to automatically generate XSLT stylesheets based on the WYSIWYG editing of HTML documents. Its adoption for the automatic derivation (generation) of GAC transformation rules appears to be a straightforward solution.

Third, the specification and development of high-level, domain-specific GAC rules (as well as corresponding authoring support) is also an interesting extension possibility. Even though generally applicable to arbitrary XML content, note that the current GAC rules are rather low-level, i.e. the usage of domain-specific extensions would provide adaptation engineers a more high-level, application-dependent view on the given adaptation scenario. Such rules could be either automatically mapped to one or more "original" GAC rules, or, for the sake of better performance, provided with their own DOM-based implementation. A possible solution would be to bundle domain-specific (language-specific) GAC rules to so-called GAC profiles (e.g. HTML GAC profile, X3D GAC profile, etc.). Thus, developers of an adaptive Web application could easily specify, implement, publish, and exchange predefined "adaptation rule packages" dedicated to the current application domain.

Finally, the application of the GAC for model-level adaptations is also a possible research issue. Even though it was originally developed to adjust XML document instances at the level of hypermedia presentation generation, the trend to formalize (Web) design models in XML (and to generate applications based on model transformations) allows to apply it even for model adaptation. Thereby, a conceivable scenario is the combination of GACs both at model-level and instance level. While the former ones could realize static adaptation (i.e. adaptation that has to be performed only once and is thus executable at model level), the latter could implement dynamic adaptation (i.e. adaptation that has to be performed for each requested document instance separately).

Application of AOP Principles to Web Application Design

The GAC-based implementation architecture illustrated in Chapter 6 allows to easily incorporate additional (independent) adaptation concerns into a Web application. Still, whereas the GAC supports powerful adaptation operations on XML input at instance level, the com-

plexity of Web applications, and the typical distribution of adaptation throughout the application, necessitates the high-level specification of such adaptations at design level. Currently, adaptation is in most design methods specified in the form of conditions that are embedded (intertwined) in the relevant design models. Extending a design with a particular context-dependency concern therefore requires that the designer embeds for the relevant design elements the new adaptation condition(s) that result in the desired context-dependency. Though these conditions can occur at one specific place in the design (e.g. to remove a link between two concrete pages), it more frequently happens that they cannot be pinpointed to one particular element (e.g. to hide for privacy reasons all sensitive data) and need to be applied at distributed places in the design (model) [Casteleyn et al. 2006a].

A similar observation was made in the programming community, when considering different design concerns of a software application: some concerns cannot be localized to a particular class or module; instead they are inherently distributed over the whole application. Such a concern is called a *cross-cutting concern*. To cleanly separate the programming code addressing this concern from the regular application code, Aspect-Oriented Programming [Kiczales et al. 1997] was introduced. An *aspect* captures the functionality of a cross-cutting concern and can be applied at different parts of the application.

Therefore, a challenging research issue for future work is the application of principles of Aspect-Oriented Programming to Web design, allowing to separate a given Web application design from the specification of additional context-dependency design concerns. If one can easily add such functionality, it becomes possible to cleanly separate additional design aspects and describe them independently from the base application. Furthermore, by translating high-level design aspect descriptions to appropriate GAC rules, it becomes also possible to automatically generate a component-based implementation. First steps towards the application of aspect-oriented principles to Web design and their GAC-based implementation were already successfully undertaken and published in [Casteleyn et al. 2006b, Casteleyn et al. 2006a].

