Abstract

Web applications continue to grow in number and importance in business, government, entertainment and education. Web Engineering deals with the development of high-quality Web applications with respect to quality factors such as flexibility, extensibility, cost, maintainability, and evolution. Existing Web engineering methodologies fall short in supporting new requirements such as native support for XML technologies, design-for-change, and parallel development.

In this research abstract we present XGuide, a model-driven methodology for XML-based Web engineering which supports the design, development, maintenance and evolution of Web applications using XML technology. The XGuide development process introduces the notion of a contract which forms the basis of a concurrent development process and reduces time-to-market. We identify several aspects involved in developing flexible and extensible Web applications and use a component-based approach to maximize reuse and extensibility. The goal of this Ph.D. work is to define the XGuide methodology and component model and evaluate XGuide on the basis of the proposed XML framework.

Keywords: Web engineering, XML, methodology, separation of concerns, aspect-oriented programming, contract-based development.

1. Introduction

The World-Wide Web (WWW) is the fastest growing part of the Internet with incredible growth rates every year [18]. Today’s Web applications are dominated by the Hypertext Markup Language (HTML) and several scripting and programming languages on the client- and server-side (e.g., JavaScript, PHP, Java servlets, ASP, JSP, etc.). These technologies focus on an easy-to-use implementation model but have considerable weaknesses in terms of flexibility, extensibility and reuse.

Even worse, the development of a Web application is often an ad-hoc process [8] which lacks systematic approaches and monitoring of various quality factors. Some claim that this situation has led to the Web crisis [8].

Web engineering deals with the development of high-quality Web applications with respect to quality factors and new requirements such as device-independence (access using devices with varying capabilities such as notebooks, PDAs or mobile phones), content-only Web services (services which are targeted at other machines, e.g., to incorporate third party content into a Web application), meta-data (information about the application and the content itself to improve rendering, searching, etc.), and personalization (adaptive selection of content, navigation or display characteristics based on user preferences and/or behavior).

When we started working with XML technology for Web engineering in early 2000, we extended the principle of separation of concerns [19] from the layout and the content to the business logic. With MyXML [12, 16] we developed a tool which generates Web applications from three separate concerns:

- The content is defined in XML files which can be enriched with tags from the MyXML namespace to embed content from relational databases, Web services, business processes or CGI parameters.
- The *layout* is specified in XSL stylesheets which facilitate reuse and easy maintenance of layout information.

- The *business logic* is encapsulated in an arbitrary programming language (we currently support Java) and transparently uses content and layout definitions.

We evaluated MyXML in two industrial case studies: the Vienna International Festival (VIF) [23] and the Austrian Academy of Science (AAS) [1]. Although the MyXML tool worked as expected, we identified several other issues - especially on the conceptual and methodological level [13, 15].

In XGuide, we apply ideas from aspect-oriented programming (AOP) [5, 14] and identify several cross-cutting concerns in Web engineering. A contract-based development process builds the foundation for the definition of aspects and concurrent development. Aspects are modeled as XML components using existing and emerging XML standards such as XSL, XML Schema, XLink or XForms. XGuide introduces an XML component model for Web applications which is platform, technology and language-independent and can be mapped to various implementation environments.

Although much work exists in the area of Web engineering, the popularization of XML technologies and recent developments in software engineering (including component software and aspect-orientation) have introduced new requirements and challenges. There is a need for methodologies that natively support XML and related technologies, that take parallel development into account, and that are designed to maximize reuse and flexibility.

The remainder of this research abstract is structured as follows: Section 2 presents the three phases of the XGuide methodology and discusses the key concepts of aspects, contracts and components. Section 3 presents related work. Section 4 gives an overview of the status and details on the contribution and evaluation of this Ph.D. work.

2. The XGuide Methodology

Developing an XML-based Web application is a task which requires many different skills [10] and involves different people such as content managers, programmers, design experts, and project managers. The first phase of the XGuide process, the *Setup Phase*, deals with requirements, technology and communication issues. The following *Implementation Phase* guides the creation of a conceptual model and the implementation in the chosen target technology. Finally, the *Evolution Phase* deals with maintenance and evolution issues. The following sections give a high-level overview of the three phases—a detailed description of each phase is beyond the scope of this paper.

2.1. Setup Phase

The setup phase starts with the requirements analysis and the definition of a set of characteristics the Web application shall provide. This ensures that all people involved understand what the ultimate goal of the project is—at least from a high-level point of view. Typical characteristics which should be discussed in this phase include, but are not limited to, the lifetime of the Web application, multi-lingual support, multi-device support, integration of legacy data, browser-less access to the application, external data sources, performance considerations, and evolution scenarios.

In the next step, the question whether XML is the appropriate technology for the project has to be answered. Often people implicitly assume that a new technology such as XML has to be better than existing ones. This is not necessarily the case. Especially, if the project deals with highly irregular and dynamic, small or short-lived Web sites, the overhead of using XML might be considerable [13]. Still, we believe that most Web projects lend themselves to XML implementations because they have a consistent layout and page structure which can be well modeled by XML technologies.

The final important task in the first phase is to establish a means of communication between the people involved. Mailing lists, Web sites, periodical meetings and a textual or graphical modeling notation are only some possibilities. The efficient and unambiguous communication is especially important for parallel development as introduced in the implementation phase.
2.2. Implementation Phase

The implementation phase covers traditional Web engineering tasks including the design of a conceptual model and the mapping of the conceptual model to an implementation technique. Since in many Web projects time-to-market is short, we support parallelism wherever possible. As a result, we push the concept of separation of concerns [19] to the extreme and do not only separate content, layout and business logic as in our previous MyXML approach, but define additional concerns such as navigation, security, device information, workflow, or meta-data. In terms of Web pages several of these concerns become cross-cutting aspects as defined in AOP (e.g., security, navigation, workflow, etc.). Even layout and content information can be viewed as cross-cutting if they are reused and applied to multiple pages.

To make parallel development of aspects possible, we define a contract for every type of page in the Web application. The contract defines how the separate aspects work together and how they can be combined. We currently use document type definitions (DTDs) or XML Schemas to form the contract. Such a contract describes the structure of the content which is sufficient for the content manager and the layout designers to develop the content and layout components (in XSL). From the contract, we can also automatically derive the interfaces to the business logic and thus express which constraints the business logic must satisfy. As a consequence of the loose coupling of aspects, each aspect can be developed independently of the others based on the contract. In the end, all aspects are combined and the final Web application is generated.

The introduced contracts work well for the separation of content, layout and business logic. For instance in the Vienna International Festival case study, the business logic was finished before either the content or the layout information was available.

Another level of complexity is introduced if aspects do not add to the definition of single pages but specify relationships between pages and/or components (e.g., navigation, workflow, etc.). We are aware of the fact that we need a more powerful contract structure to be able to integrate new concerns such as meta-data, workflow or security contract information.

Once the contracts are identified and specified, the various aspects can be developed concurrently. In XGuide all aspects are specified as XML components which again are built from reusable XML subcomponents. For instance, a content component can be composed from several other content components including the result of a database query, the document returned from a Web service and static XML content.

In XGuide we only support two kinds of composition: composition within the domain of a single concern is called vertical composition. Only at the page-level, the components representing different aspects are integrated (horizontal composition). Many other approaches support composition of already integrated page fragments (e.g., the embedding of formatted HTML fragments into another HTML page). Such an approach, however, significantly reduces the usability of the generated components. Consider the case where you automatically embed pre-formatted navigational structures into HTML pages. As long as HTML browsers are the only target device, this works well. Other devices such as personal digital assistants (PDAs) or mobile phones require a different formatting of the navigational structures and thus could not be easily supported.

When all aspects are fully implemented and tested against the contracts, they form the final model of the Web application. The last step is the code and document generation process which is realized as exchangeable module which only depends on the target technology to be used (e.g., Java servlets, JSP, ASP, etc.). The XML component model is also the unit of maintenance and evolution as described in the next phase.

2.3. Evolution Phase

When the initial release of the Web application is deployed, the XGuide process enters the evolution phase. In this phase, we distinguish between the maintenance and the evolution of the application. The term maintenance denotes all changes to the Web site which require the modification of a single concern. Updating the site’s content or changing the layout information are examples for maintenance tasks. As soon as an update directly or indirectly requires modifications of more than one concern, we talk about evolution.
According to our experience, maintenance tasks occur frequently (especially in the content and layout domains) but are easy to implement. This kind of updates significantly adds to the flexibility and extensibility of the Web application since they can be performed independently of and without changes to all other aspects.

Modifications which are classified as evolution, on the other hand, are much more sensible due to possible side-effects and far reaching influences. Especially since XGuide supports reuse of components within the same domain (vertical reuse), a modification usually affects not only one but several pages. In the maintenance case, this behavior is usually desired, i.e., consistent updates of concerns such as the content, navigation, layout or business logic are desirable. If several components from different concerns are updated, however, the final result of the modifications is much harder to identify and control.

3. Related Work

Many of today’s Web engineering approaches model Web applications using experience from software engineering such as component-based development or build on structured information stored in (relational) databases to achieve a certain degree of separation of concerns.

Widely-accepted methodologies such as HDM [6] or RMM [10] build on modeling techniques from database design and add hypertext constructs such as navigation. They are focused on structured, relational domains which usually cover only parts of a Web application. OOHDH [21], an object-oriented extension of HDM, uses a domain model to introduce a sophisticated set of navigation primitives. Some of the limitations of RMM are addressed in the Extended RMM [9]. A key concept of the Extended RMM are m-slices which support aggregation of information from different entities and support reuse of such artifacts.

WebML [3] is a notation for modeling Web applications at the conceptual level. WebML does not include a methodological approach for Web development but identifies four concerns which are represented in separate models: the structural (i.e., content), page composition, navigational and presentational models. WebML is based on an E-R model to retrieve the structural and navigational model and does not support parallel development.

WebComposition [7] describes a component-based approach for the support of the full life-cycle of a Web application. Components are collections of properties (key/value pairs) on a fine-grained level (e.g., single elements of a page) which can be composed to larger artifacts such as page fragments, pages or even sites. The underlying idea of WebComposition is component reuse; separation of concerns is not addressed.

The main goal of the World-Wide Web Design Technique (W3DT) [2] is to define a simple and comprehensive model for Web development. The concerns content, layout and navigation are separated on a conceptual level. The Extended W3DT [20] focuses on the application of W3DT in commercial systems and details on navigational and structural design of large sites. W3DT and eW3DT define themselves as complementary to methodologies such as the above-mentioned RMM or OOHDDM.

The Extensible Web Modeling Framework (XWMF) [17] is an application of the resource description framework (RDF) and defines an exchangeable model of Web applications. Such a model is intended to support the life-cycle of a Web application and provides a mechanism for customization and extension. The underlying model (WOCM - Web object composition model) is based on object-oriented concepts and does not support a strict separation of concerns or support for parallel development.

Initiatives such as Cocoon [22] or Barracuda [4] take a technology-centered approach and exploit XML or object technology to support Web development. They provide implementation frameworks but do not include a methodology or conceptual models.

4. Conclusion

XGuide tackles the problem of systematic and concurrent development of flexible and reusable Web applications. The work strongly builds on the principle of separation of concerns and loosely coupled aspects to support development of flexible, extensible and maintainable Web applications. Aspects are represented as XML components and compositional concepts implement reuse and consistency requirements. Unlike many existing methodologies, XGuide not only
uses XML as modeling language but natively supports XML and related technologies in the development process. Further, XGuide introduces the concept of contracts in the context of Web engineering to shorten development time through the explicit support of parallel development. XGuide models represent platform, technology and language-independent Web applications which are then mapped to the deployment environment.

The current version of XGuide supports contracts for the concerns layout, content, device information and business logic. It further transparently integrates a variety of data sources including databases and Web services and has basic support for navigational structures. We did a first evaluation of XGuide with a small Web project building a Web site for a course on XML technology [11]. We are now working on the full integration of other concerns such as more flexible navigation, security and personalization issues into the contract specification and XGuide model. In parallel, we develop XDesigner, a Java-based tool which supports creation of Web applications using XGuide. The final evaluation of XGuide will be performed with the Web site of the Vienna International Festival 2003.

The contribution of this Ph.D. work is a novel methodology for XML-based Web engineering combining the principle of separation of concerns, cross-cutting aspects and component-based development with the strengths of XML technology.

References


