Web based legacy system componentization

Research plan

Josip Maras

1.10.2009
1. Research Area

My research will be performed in the areas of Component Based Software Engineering (CBSE) and Web Engineering. It will deal with component-based system refactoring in order to increase maintainability.

Today’s software development is characterized by continuous evolution – software products are continuously updated during their usage. In most cases systems grow organically by adding new functionalities or by changing the system architecture due to new technologies or business strategies. This organic growth makes the systems more difficult to maintain and improve. Systems often lose their integrity and dependencies between different parts become uncontrollable. This is especially true for Web applications, since their development, contrary to some other domains, is not characterized by rigorous software engineering methods. The consequence is that many Web applications are poorly structured and are not adequately documented, which leads to difficult maintenance. One of the proven ways of increasing system maintainability is by software system componentization [1], i.e. by breaking down the system into smaller, more easily identifiable pieces (components) with well defined interfaces in a specific way (by following a certain component model). Software engineering discipline that deals with components, component models and componentization is called Component based Software Engineering (CBSE).

CBSE is concerned with assembling applications from pre-existing components. A component is usually defined as in [2]: “A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third party”. Since applications are built from pre-existing components, CBSE increases system’s modularity which in turn improves maintainability, system behavior predictability and enables reusability. Even though there are strong benefits of component-based approaches in all domains, so far CBSE was not in the focus of Web Application development.

In the last two decades, Web applications have made a tremendous leap forward: from simple static Web pages developed only in HTML to complex dynamic Web applications developed using server-side technologies (e.g. PHP, ASP.NET, Java) that extensively use Web services, databases and client-side technologies (e.g. JavaScript, Flash, Silverlight). Unfortunately, the development of Web applications is often seen as a one-time event and is usually based on the knowledge, experiences, and development practices of individual developers. Because of the ever growing importance of Web applications, a more systematic, rigorous approach is necessary. Traditional software engineering is usually not enough, due to particularities of Web applications. A process performed throughout the whole lifecycle of a Web application is necessary. In [3], Web Engineering is defined as:

- **Web Engineering is the application of systematic and quantifiable approaches (concepts, methods, tools) to cost-effective requirements analysis, design,
implementation, testing, operation, and maintenance of high-quality Web applications.

- Web Engineering is also the scientific discipline concerned with the study of these approaches.

2. Research overview

The goal of this research is to study and provide solutions for the transformation of Web based legacy systems to component-based systems, while increasing system maintainability and preserving extra-functional properties characteristic for Web based systems (bandwidth, response time, performance, memory and CPU usage, etc.). In this context a Web based system is a distributed application consisting of at least two separate applications: (i) a Web application used for data access and changing the business state of the system and (ii) a standard, non-Web application that performs core system functionalities.

This research will be based on a case study application – iForestFire [4]. iForestFire is Web based legacy system developed at the University of Split, Croatia, used for forest fire monitoring. It is composed of two important parts:

- Fire detection application is an application responsible for forest fire detection by image/video analysis. It is a standard multi-threaded C application that gathers data from different embedded devices (cameras, meteorological stations), analyzes that data and decides whether or not there is any sign of fire.
- Web application acting as a user interface to the system. It enables data access and system configuration. It is developed using php, JavaScript and HTML.

In order to be able to componentize a certain Web based legacy system a suitable component model has to be defined. This component model has to capture all the important characteristics of Web based systems and has to model a system on the appropriate level of abstraction. This leads to the first research question:

**Q1:** What is an appropriate component model for distributed Web based systems?

After the definition of a component model, the current legacy system (iForestFire) has to be componentized. This is used as a basis for the second research question:

**Q2:** How can we perform componentization of legacy distributed Web based systems according to some component model?

The goal of this componentization is to increase maintainability while keeping (or even improving) overall system performance. So in order to validate the results, we have to be able to compare the two systems. This goal leads to the third research question:

**Q3:** How can we compare two functionally equal Web based systems in terms of their maintainability and performance?
3. Research methodology

This research will be performed in the following phases:

3. Defining a new or extending an existing component model suitable for component based development of Web based systems.
4. Refactoring the case study application according the component model defined in step 3.
5. Defining the necessary metrics in order to be able to compare the legacy case study application with the refactored one.

4. Related work

4.1. Some of the existing component models

Currently, there are many component models, some of them originating in the industry, while others in the scientific community. They can also be divided into component models targeting specific application domains (e.g. web applications, automotive industry, etc.), and general purpose component models.

Some of the general purpose component models are the following:

- **Component Object Model (COM)** [5] is a platform independent, distributed object-oriented software architecture for creating and connecting binary software components, developed by Microsoft. It was one of the first attempts to increase program independence and allow programming language heterogeneity. It was extended to support distribution (DCOM) and enterprise level features (COM+). COM is superseded by the .NET platform.

- **CORBA Component Model (CCM)** [6] is a component model developed in order to reduce the effort of developing and deploying CORBA applications. Components in the CCM are programming language and platform independent as long as they use the CORBA middleware. A component package in the CCM consists of compiled program code, and a CORBA component descriptor (a XML file containing information about the interfaces and the services that the component supports).

- **JavaBeans** technology [7] is a portable software component model for the Java platform. It is widely accepted and used, mostly in the desktop and Web application domains. A Java bean is a Java class that complies with conditions stated in the JavaBeans specification. The specification defines Java beans as reusable software components that can be manipulated visually in a builder tool. This is especially suitable for building graphical user interfaces (GUIs), where beans are most commonly employed.
• *Enterprise JavaBeans* (EJB) [8] is a component model that extends the *JavaBeans* component model to support server components. EJB is primarily used for a client – server model of distributed computing. The model simplifies the development of middleware by providing server support for a set of services, such as transactions, security, persistence, concurrency and interoperability.

More information about these and other component models can be found in [9].

So far, Web based systems were not in the focus of CBSE academic community. Some work has been done, but the majority of component based approaches come from the industry. Some of those approaches are:

• Portlets [10] in Java world are distributed Web components that encapsulate Web applications. Portlets can be viewed as presentation-oriented Web Services which are packed to be delivered through third-party Web applications.
• ASP.NET Web Controls [11] are Web components that run on the server and that encapsulate user-interface and other related functionality. They can be composed into ASP.NET Web pages.
• Web Parts [12] in Microsoft SharePoint are similar to portlets, with the difference that Web Parts are built using Microsoft ASP.NET technology.
• Google gadgets [13] are client side Web components developed in HTML and JavaScript. They can be composed into Web pages.
• Microsoft Pop Fly [14] is a discontinued Microsoft mashup technology that provided a Web tool for visually combining data sources (usually Web services) and visual Web components that would display that data. It was built using the Microsoft Silverlight technology [15].

There have also been some approaches in the scientific community:

• WebComposition [16] is the first work trying to introduce component based ideas into Web engineering. WebComposition is based on an object-oriented model in which an application is hierarchically composed of components. Components can be either primitive (single HTML elements) or composite (created from other components). At the highest point in the hierarchy components can be entire pages, while they are usually related to certain reusable parts of the pages.
• AMACONT [17] allows the development of Web applications by aggregating and linking configurable document components. The components are documents or document fragments that represent adaptable content on different levels of abstraction. Component interfaces are described by metadata specifying their properties and adaptive behavior.
• AlCoWeb [18] is a hierarchical component model for the presentation tier of Rich Web Applications. The component model includes the following elements: components, interfaces, ports and connectors. *Components* are Web elements that can be either atomic (basic HTML elements) or hierarchical (composed of multiple
atomic components). *Interfaces* represent public services defined by components. Interfaces are divided into provided, required, checking or event interfaces and are realized as JavaScript functions. *Ports* represent a set of interfaces of the same kind that are related to the same functionality. *Connectors* link interfaces of different components and encapsulate interaction protocols. A tool AlCoWeb-Builder, based on Eclipse, was developed in order to support Web application development based on the AlCoWeb component model.

### 4.2. Application metrics

In my research I will be looking for ways of increasing system maintainability while keeping system performance. So, from my point of view, the most important application metrics are: maintainability and performance.

#### 4.2.1. Maintainability

IEEE standard for software maintenance [19] defines *maintenance* as: “The modification of a software product after delivery to correct faults, to improve performance or other attributes or to adapt the product to a modified environment”. The term *maintainability* is used to describe the ease with which the software can be corrected, adapted or perfected [20]. So far, there is no clear way of directly measuring maintainability, but it can be evaluated by measuring some other, more tangible system properties like system modularity, system complexity, function points, etc.

Maintainability has been quantified by the Maintainability index (\(M\)) in [21] as a function of directly measurable attributes. These attributes can be related to management (management practices), operational environment (in terms of hardware and software) and target software system (source code attributes, documentation attributes, software system maturity, stability, and etc.).

The Maintainability Index is more related to standard (non Web-based) applications, but there have been attempts of estimating Web application maintainability based on traditional, object-oriented metrics like LOC – lines of code, LCOM – lack of cohesion in Methods, RFC – response for classes, and etc. in [22]. This study showed that OO metrics are not suitable for estimating maintenance effort for Web applications, even dough they should be taken into account when estimating maintainability of OO Web applications.

In order for the Maintenance Index to be usable in the context of Web applications, due to their inherent differences from traditional software systems, it was extended in [23] by Di Lucca et al. This extension accounted for the particularities of Web applications by including Web application attributes such as: total number of Web pages, server and client scripts, connections, Web objects and etc. Similar research has been presented in [24], where attributes used for maintainability measurements (coupling, complexity and reusability) are derived from Web application UML diagrams.
4.2.2. Performance

Web application performance metrics [25] are typically expressed in terms of response time (R), throughput (X) and utilization (U). The response time is typically defined as the time between sending a request to the Web server and receiving a response from that Web server. Throughput states the number of requests that the server can respond to in a time unit, and utilization shows the time percentage in which the system was being used. The goal of performance analysis is aimed at determining the relationship between performance metrics, the work load (number of requests that arrive per time unit) and the performance-relevant system characteristics. Performance analysis is also often used to determine bottlenecks (system component that first reaches 100% utilization as the work load increases).

5. Community

5.1. Conferences

A part of my research will be concerned with Web application component based development and Web application metrics so more specific Web Engineering conferences should be targeted. Most important ones are:

- **ICWE** (International Conference on Web Engineering) is, in my knowledge, the most important conference in Web Engineering that among other things deals with areas directly concerned with my research, like: Component based Web Engineering, Web architectures and application frameworks, Web quality and Web metrics, Performance modeling, monitoring and evaluation.

- **WISE** (International Conference on Web Information Systems Engineering) is another conference whose area is closely related to Web Engineering. From my point of view the most interesting parts are dealing with Web Metrics and Web applications in general.

- **WWW** conference, the most prestigious conference on the World Wide Web, contains a track on Web Engineering.

Since my research area is in general concerned with CBSE and software metrics (especially maintainability), the following conferences are also of interest:

- **CBSE** (Component Based Software Engineering)
- **QoSA**(Quality of Software Architecture)
- **ICSM** (International Conference on Software Maintenance)
- **CSMR** (European Conference on Software Maintenance and Reengineering)
- **WoSQ** (Workshop on Software Quality) is a workshop on software quality held as a part of ICSE.
Also, more general software engineering and software architecture conferences are also of importance:

- **ICSE** (International Conference on Software Engineering)
- **ESEC/FSE** (European Software Engineering Conference/Foundations of Software engineering)
- **ICSEA** (International Conference on Software Engineering Advances)
- **ASEA** (Advanced Software Engineering and its Applications)
- **WICSA** (Working IEEE/IFIP Conference on Software Architecture)

### 5.2. Research groups

As far as I was able to find, there is only one research group actively involved in the development of Web application component models, even though there are several researches that have published articles in that domain. Those are:

- **ONEKIN** [26] is a research group on the University of the Basque Country, Spain dealing mainly with portlet engineering issues, but also with Web Application Engineering and Component Based Software Engineering.
- **Kadri R.** from the University of Alkante, France that has created a hierarchical component model for the presentation tier of Rich Web Applications (AlCoWeb).
- **Jeewani A. Ginige** from the University of Western Sydney has worked on a component model that would enable end users to develop Web applications.
- **Florian Daniel** from the University of Trento, Italy focuses on Mashups and lightweight composition on the Web.

Research group focused on Web application metrics (with emphasis on maintainability) are:

- **GIDIS** group, from the National University of La Pampa, Argentina. They are active in research concerning: Quantitative Evaluation Methods for Web Sites and Applications and Web Metrics and Cataloguing.
- **RCOST** group, from the University of Sannio, especially **Giuseppe A. Di Lucca**. They have performed research in the areas of Web application Reverse Engineering, Maintainability and Metrics.
- **VSR** group, from the Chemnitz University of Technology, Germany. They are focused on Web engineering, especially on evolution, federation, and reuse as principles for building distributed systems, in the context of Internet and Web.
6. References

1. Maarten Koning. [Online] [Cited: 9 30, 2009.]


