Software Product Line Technologies for Web-based Applications

Noor Afshan Fathima
Graduate Student: Dept. of Computer Science (India Online)
Illinois Institute of Technology, Chicago

***Abstract*— A research paper that presents a comparative analysis of different approaches in utilizing Software Product Line Technologies for building a Web-based Applications.**

Keywords—Software Product Line, Web, Component based Approach, Web technologies, Domain Engineering, Application Engineering

I. Introduction

Software product line engineering is a strategic approach to developing software. Software product lines represent most exciting paradigm shift in software development since the advent of high-level programming languages. Nowhere else in software engineering there has been such breathtaking improvements in cost, quality, time to market, and developer productivity, often registering in the order-of-magnitude range. Software systems are built for around one-tenth the cost. With around one-tenth the faults. Delivered in around one-tenth the time.

The improvement of costs and time to market are strongly correlated in software product line engineering: the approach supports large-scale reuse during software development. As opposed to traditional reuse approaches this can be as much as 90% of the overall software. Reuse is more cost-effective than development by orders of magnitude. Thus, both development costs and time to market can be dramatically reduced by product line engineering. Unfortunately, this improvement does not come for free, but requires some extra up-front investment. This is required for building reusable assets, transforming the organisation, etc.

The Web is nowadays omnipresent: we use it at home for private reasons, and we use it at work for professional reasons; we use it for fun (e.g., gaming) and for serious interactions (e.g., home banking), via fixed stations and via mobile devices, and these are just few of the motivations for and the contexts in which we exploit such a powerful medium. The Web has indeed probably become the number one reason for private PCs at home, and the most important kind of "business card" for companies and institutions. Very likely, each of us has already tried at least once online applications such as Amazon.com for buying books or CDs, Ikea.com for buying furniture, and, of course, Google.com for searching Web sites. Similarly, most of us can no longer imagine a travel planning without the flight booking and hotel reservation systems that are accessible over the Web.

While the potential, contents, and features offered via the Web are fascinating and attracting an ever growing number of people, there is also a steadily increasing number of people who are interested in developing applications for the Web. If one likes the Web, there is nothing better than developing an own Web site or Web application. Yet, depending on the result one aims to achieve, writing a good application for the Web might be an intricate and complex endeavor that typically requires profound knowledge of the way the
Web works.

Several technologies have enriched the scenario and the Web has progressively become a multi-domain platform, offering support not only for information delivery, but also for application execution. Nowadays, complex Web applications, such as eCommerce systems, large-scale corporate platforms, collaborative distributed environments, social networks, and mobile
services, are almost omnipresent.

Developing Web applications involves several intrinsic challenges, which imply the adoption of adequate technologies and methodologies. Sound methodologies, forming the baseline for rigorous and repeatable development processes, are especially needed to cope with the complexity of current Web applications and to ensure their quality. Web

applications are accessed by users from different cultures and locations, and with different personal profiles and preferences, work contexts, and (dis)abilities. Therefore, an engineering team needs to handle and manage the variability imposed by different contexts as part of a Web product's engineering. On the other hand, it is also important to identify, manage, and reuse those Web product assets which are common across different contexts to make the engineering process less costly and more effective. This is directly related to the field of software product line engineering, where a family of software products is engineered based on the identification of a set of commonalities and variabilities. The focus here lies on the development of reusable and configurable software components, thereby reducing development time and cost for individual products.

II. Features of web applications

There are some features that characterize Web applications and distinguish them from traditional software systems:

- *Higher accessibility of information and services:* compared to closed intranets or desktop systems, the World Wide Web enables access to information and services for far more users simultaneously. Different modalities and views on data and services need to be designed to support different user needs.
- *Document-centric hypertext interface:* the offered information and services have to be mapped onto a hypertext document. Interconnections between various views on information and pages require peculiar design abstractions to understand and represent the resulting hypertext structures and their traversals.
- *Variable technologies for data management:* data is distributed on the Web in various formats, schemas, and technologies, such as XML, RDF, and traditional databases. Designers need to pay attention to the design of data structures, of the access to external data sources, and of the mapping between them
- *Variable presentation technologies and engines:* different presentation formats must be addressed to accommodate the characteristics and display capabilities of different browsers and different devices.
- *Architecture complexity:* the higher level of accessibility and the lighter nature of clients (the Web browsers) require distributed, multi-tier architectures for the access to information and services.

III. Software product line engineering framework

The framework divides product line engineering into two life-cycles : domain engineering and application engineering. Domain engineering results in the common assets that together constitute the product line's platform. Application engineering results in delivered products. Within the two life-cycles, there are nine sub-processes. Eight of the engineering processes form four pairs: requirements engineering, design, realisation and testing are done in both domain and application engineering. These pairs of processes are strongly connected. The domain engineering sub-processes result in common assets that are used in their application engineering counterparts to create products. In return, the sub-processes in application engineering generate feedback that is in use in domain engineering to improve the common assets. This feedback loop is essential to ensure that the platform remains suitable for the efficient production of end products. In some cases, application-specific assets are reused in domain engineering to become part of the platform, but that is a side effect not a goal – of application engineering.

Domain engineering is the life-cycle that results in the common assets that together form the product line's platform. It is further responsible for scoping the product line, and ensuring that the platform has the variability that is needed to support the desired scope of products.

Application engineering takes the common assets of the product line and uses them to create products. Application engineers bind variability in the common assets to create instances that are fit for the products that they are developing. They combine these instances with application-specific assets that they develop themselves.

IV. Adaptive Web Applications and Software Product Lines

The basic principle of adaptation is to select appropriate variants of particular product features or a combination of product features (either by a human or a system) to satisfy user needs. Features which are adaptable are the ones which vary, and thus we consider these variable features of the application. The features which are not varying can be considered as common features of the application. From the point of view of

software product lines, the engineering of adaptive (Web) applications has the following point in common: the application should be ready for customization for different customers, but still retain some parts in common. However, the customization idea should be taken beyond the static customization done by a development team. True adaptive Web applications adapt to changed environments, user features, and other parameters on the fly (i.e. at runtime) according to knowledge gathered by their sensors.

Domain analysis for Web-based applications involves application, environment, user domain conceptual and feature models where:

- Conceptual models: are used to model concepts and their mutual relationships in a particular domain and serve as vocabularies for later feature models and domain designs.

- Feature models: are used to encode configuration knowledge, i.e., to maintain common and variable features of concepts and their dependencies, such as a company's stored experience. While the configuration specification is user-regulated in application and environment domains, it is regulated by adaptation requirements in user domains.

The purpose of the conceptual models is to document domain and environment vocabulary used in all other models. For example, the domain/content presented in a training suite of a Java lecture implies that the domain conceptual model will refer to concepts from the Java programming language.
As the lecture is accessible in a course environment, the course structure and some other concepts will be depicted in the environment conceptual model. Feature models are on the other hand used to document configuration relations between concepts from the conceptual models. This means that in
Web applications they usually document which content concepts are used to articulate a particular concept in a presentation on a Web page, e.g., Java objects in case of a page on Java objects. The user domain feature models document configuration aspects according to the requirements of adaptation functionalities, i.e., which features and which combinations of features are required for certain adaptation strategies. Domain design for Web-based applications involves the navigation design, the user design, the application domain design, and the environment design.
The navigation domain design produces an architecture which enables the hypertext solution domain to generate HTML documents for particular environments with relevant content. The user domain design produces an architecture for user models to be used with applications in the domain. The
application domain design produces an architecture to access content as an instance of application domain concepts and features. The architecture can be domain-specific, i.e., based on vocabulary defined in an application domain conceptual model from domain analysis, or it can be generic in order to access any content. The environment design defines an architecture for accessing and manipulating the environment.
Further refinements of domain design concentrate on content composition and navigation between content components. The domain design elements are used to bind the concerned domains together to produce reusable units of Web applications. The application and environment domains are bound together to create content. An environment called story collaboration models serves to create content components.

The domain design also incorporates mappings of the collaboration models to navigation trails which specify the sequences of content components to be presented to a user as presented in section on navigation behavior design. The state diagram approach is adopted for this purpose. The access to content components and links between them are annotated by
guards. The guards consist of constraints specified by conditions, functions updating user profiles, information presentation, and access environment appearance represented as side-effect actions.

Domain implementation includes construction of parameterized implementation components with their mutual dependencies. Parameterization of implementation components can be realized, for example, as HTML templates, active server templates, WAP templates, or components in other implementation languages. Domain implementation should also incorporate domain-specific languages such as query languages or languages for selecting and integrating components for a given application from the application family. All the mentioned models and their parameterizations are transformed into these implementation components.

Application engineering is the activity that follows the domain engineering activities, and defines a particular contract for an application. The Web-based application is built according to requirements specific to a particular application. Similarly to software systems, the requirements are split into those which can be satisfied by the components from the application family framework created during the domain engineering process. should be satisfied by a custom development. The results of

the framework generation and custom development are integrated into the final product.

The domain analysis is governed by conceptual and feature models. These models are used to manage the common and variable features of the applications. The conceptual and feature modeling is explained by use of examples.

Features which can be denoted as adaptable are the ones which vary and are thus regarded as the variable features of the application. The features which stay unadapted are common and are thus regarded as the common features of the application. The commonalities and variabilities of software families are the main concerns in domain engineering methods.

The process of such information modeling can be summarized in the following steps:
1. Define a concept model for the application, user, and environment (e.g.,concepts used to teach the Java programming language in a course as for the application concept model, concepts for a learner's performance as for the user model, and concepts for course framework as for the environment model).
2. Define a feature model for all concepts from the application, user, and environment concept model;
3. Update concept and feature models of the domain, the user, and the environment if new concepts and/or features have been developed.

Concept models represent abstracted knowledge of the real world. Entities of interests are mapped one-to-one to concepts in these conceptual models. Similarly, the relations of interest between the entities are mapped to associations (relations) in the conceptual models. In Web applications, the conceptual models are usually concerned with the information (or its representation) being served, including information chunk representation and software artifacts which help to support access to information.
Information is usually comprised of one or more concepts from a domain where general conceptual models, taxonomies, or ontologies may already exist.
For example, a Java tutorial serves information which belongs to the domain of computer science. There are several taxonomies which are used to classify computer science literature (ACM CCS 17) or to describe a body of computing knowledge and curricula. 18 Companies also use their own conceptual models to communicate terminology used in their information systems. Information delivery environments can also feature different concepts which are related to each other.
In the Unified Modeling Language (UML) was used to model the application domain and environment domain models. A concept is modeled by a class, which is stereotyped by the Concept stereotype. Concepts can be connected by one of the relationship types, association, generalization specialization, or aggregation in order to support the known abstractions of model domains.

Furthermore, some domains provide information which is more of a "procedural knowledge" character. For those domains, we employed activity diagrams to model activity concepts with control flow relations between them.

Content concepts refer to an application domain as a domain of information (or content) which is to be served by a Web application. Conceptual modeling of an application domain models the domain in terms of concepts which are relevant to the content (or are described by the content) served by a Web application.
The app modeled by the UML class diagram with concepts annotated by the Concept stereotype and their mutual relationships. One possible view on relationships between Object, Class, and the object's State and Behaviour. Methods and Variables are used as additional concepts to describe the relationships, later realized by content fragments.

The process of application domain modeling differs from application to application and from organization to organization. It may well be that the conceptual models are created according to content which already exists in some applications (having already been supplied by a particular organization). In this case, the models are created for the purpose of reuse and customization, to document what has been already developed. But on the other hand it may well be that an organization foresees that its applications will be partially reused, and already uses models from the beginning of the development process.
The following are the main activities in application domain modeling:
• Collecting information sources on the application domain
• Analyzing the information sources.
• Extracting instances of concepts of interests from the information sources.
• Classifying/categorizing the instances into concepts.
• Creating relationships of interests between the concepts.
• Refining the conceptual models.

Information environment concepts

This refers to an information or environment domain as a domain for representation and organization of information in the context of a delivery platform. This may involve the existing content or references to be used as a source for new content. An example of an environment is a course with its lectures or modules. Another example is an e-book with its chapters. In a customer support domain, an environment can be a problem ticket with its subproblems, activities, contracts, and so on.

An Environment Domain Model is a set of models with concepts interconnected by association, aggregation, and generalization/specialization relationships. The concepts and the relationships originate from a domain which is used to organize the content for presentation, delivery, or management purposes in a Web application.

An example of an environment conceptual model in a training suite. As the training suite can be provided with several possible virtual environments, a development company team needs to communicate how the environments are structured. An example of such an environment suitable for a Java tutorial can be a virtual course. Concepts such as Course, Lecture, Module, Learning Object, and Person in various association roles such as Lecturer, Garant, and Provider would then appear in a similar UML class diagram for the environment conceptual model.

A process of information/environment modeling generally consists of the following activities:

• Identifying existing and planned environments within a Web product portfolio.
• Analyzing organization patterns for such environments.
• Identifying typical instances of concepts for such environments.
• Classifying/categorizing the instances into concepts.
• Creating relationships between the concepts.
• Refining the environment conceptual models.

User concepts

In this approach, a user domain model is defined as a model which characterizes the users, their behavior and features, their knowledge, and so on. Concepts are selected according to whether they are used in current Web applications or will be used in future Web applications to parameterize adaptation processes. A user model is created in a similar way to the application domain and environment model. The main characteristic of a user model then concerns the concepts and features which will be used as parameters for adaptation.

An example of such a user model is the e-learning domain. A Learner is a subclass of the generic User. The user usually belongs to one of several role groups. The learner's learning performance is represented by the Performance class. The learning performance is sometimes certified by Certificates. Each learner has own Preferences. These can be specialized into specific subclasses.

Feature modeling Adaptation components in adaptive Web applications usually recommend one of the options for links, content fragments in a content composition, or information items which are configurable in the Web application based on a user profile or the possibilities offered by a specific environment. This means that there are parts of the content, environment, and software components which are stable or common for any user or customer and parts which are variable depending on certain factors (mostly the values of user features).

To plan such an application, a designer should be able to think and reason about the common and variable parts. There is a similarity between domain engineering approaches concentrating on reuse and adaptive application engineering. In domain engineering, feature model ing plays a prominent role. The main reason for employing feature modeling in current domain engineering approaches is to handle variability in and dependencies between concept features of a system family resulting from different requirements of stakeholders using the applications from the family.

In adaptive applications, the customization has an even broader scope; i.e., in addition to the customization based on the explicit requirements of stakeholders, there is still some variability which is left until runtime to be exploited according to an evolving user profile or other factors.

From the system point of view, variability has been studied in the context of the software configuration management community. In such cases, variability is handled at the systems component level by means of versions which contribute to different system releases. Version control in the document-oriented hypermedia domain has been studied in several works. All the works mentioned provide a model of versions and a model of configurations, which define how the versions contribute to the final configuration.

The variability considered in feature models has a broader sense when considered at the application level. At this level, it is taken into consideration in several modeling aspects of Web applications, and not just in the context of source code and

changes to be made to the source code. Variability in product lines is defined as a measure of how members of a
product family may differ from each other. Variability can occur in all the significant aspects (products) of the Web application engineering process;ni.e., in the application domain, in navigation, and in presentation.

In the application domain, different content can be used to communicate the same information to people with different backgrounds and characteristics. Moreover, the same content can be represented by different media and this content can evolve in time. The content can also be presented in different
environments using different media, e.g., as a book, a lecture, or an article. Also overall access to the content can be managed through different patterns, such as a digital library, an e-course (virtual university), or online help.
Each user group may require a different information fragment to browse a different composition of the presented information (local navigation) and
a different order of and interconnections between information chunks (global navigation). Also, different navigation styles can be determined according to the target environment where the information is served to a user.
Similarly, it may be appropriate to supply different user types with specific display designs, layouts, and organization of the information to be read.
The target environment can also restrict presentation possibilities. Thus, it is important to take account of this kind of variability as well

A feature model is a set of models which represent configuration aspects of concepts from domains analyzed in Web application engineering. Each feature model has one concept and its respective features. The concept and features are connected to each other by a composition relationship. Configuration relations between features and the concept are represented as variation points.

V. Conclusion

In this study we have looked into some design concerns that are increasingly becoming mandatory features of modern Web applications, i.e., localization, internationalization, personalization and adaptation, and accessibility. These three concerns are peculiar characteristics of Web applications. While in traditional software products, localization and internationalization mostly meant translating all the menu items, commands, information, help instruction, and so on into multiple languages, localization and interna tional in Web engineering typically goes beyond the mere linguistic problems. For instance, big companies or associations that have affiliations in multiple countries and country-specific merchandising or assistance facilities (e.g., call centers) need a corporate Web application that is not only translated in multiple languages, but that also takes into account country-specific information, regulations, laws and cultures. Here we described the problems leading to the demand for localization, the anthropological research under-
pinning localization support, and how localization is addressed in Web design methods.
Given the wide variety of potential visitors, and the different contexts Web applications can be used in, adaptation is a feature that is of growing importance. Adapting a Web page is the general process of changing it to fulfill a certain requirement. It comes in different forms. Personalization is the adaptation for the specific user, based on his identity, his characteristics and his individual needs. Adaptivity is the ability of the Web application to adapt according to the user's browsing history. Context-awareness is a form of adaptation that allows a Web application to adapt according to the changing context. All these adaptations occur at runtime, i.e., in the running Web application. Software Product line engineering is one of the techniques to implement the previous features, and to manage their complexity.

References

[1] A Product-Line Architecture for Web Service-based Visual Composition of Web Applications. Marcel Karam, Sergiu Dascalu, Haidar Safa, Rami Santina, Zeina Koteich.

[2] A Reuse-Oriented Product-Line Method for Enterprise Web Applications Neil Mather and Samia Oussena School of Computing and Technology University of West London, London, UK, W5 5RF

[3] SUPPORTING WEB APPLICATIONS DEVELOPMENT WITH A PRODUCT LINE ARCHITECTURE. LUCA BALZERANI, DAVIDE DI RUSCIO, ALFONSO PIERANTONIO, GUGLIELMO DE ANGELIS.

[4] A Product Domain Model based Software Product Line Engineering for Web Application. Takashi Nerome. Masayuki Numao