Partitioning the Navigational Model: A Component-Driven Approach

Stephen Kerr and Daniel M. German

Department of Computer Science,
University of Victoria,
E-mail {skerr,dmgerman}@cs.uvic.ca

Abstract. This paper proposes using a Component Approach to model navigation in a hypertext application. It proposes Navigational Semantic Units (NSUs), which are component-oriented units that describe some meaning of navigation from the user's perspective. The NSUs are an abstraction of a navigational model and are the basis for a methodology to design hypermedia applications based on the Model View Controller Architecture.

1 Introduction

Component-oriented software design and implementation is an established school within the software engineering community. Partitioning of software applications into components is left to the designer who uses experience, domain knowledge, and judgement in order to define the responsibilities, tasks, and interaction of components within a software system. Hypermedia applications can benefit from component-oriented techniques to develop, test, and maintain complex systems with greater ease. In models such as OOHDM and RMM, the hyper-model is generally viewed as a single unit. A large hyper-model becomes complex as the navigational paths of associative links, structural links, and navigational design patterns are employed to encompass all possible user navigational requirements. The relationships between pages must be understood in order for maintenance operations to be successful. Unfortunately, it is difficult to track all the incoming links to a page and unless aggressive management is used, the implementation of complex models diverges from documentation over time. Navigational paths may change, associative links become invalid, and content becomes obsolete. Complexity of model and implementation is an issue of large hypermedia systems. By partitioning a hyper-model, the partitions can be modelled individually. The internal models are less complex because the number of inter-related pages (or nodes) is smaller within a partition than the number of inter-related pages in the whole hyper-model. These caracteristics will make maintenance easier.

2 Partitioning the Hypermedia Navigational Space

A problem with not partitioning the hypermedia model is that the information space of a hypermedia application in terms of the number of entities, data elements, and data instances can become very large. A partitioning of the hypermedia system can alleviate

the complexity by endorsing the ”divide and conquer” philosophy. If we are to introduce a partitioning system, it must still be able to provide associative and referential links between data. The partitioned system will also need to have some navigational structure that will allow the user to traverse the partitions or the areas of important data within an application. Most importantly, the partitioning must allow a designer to create and a coder to implement a large complex system with greater ease than an equivalent system that does not have a partitioning system. The number of components will depend on the scope of the application, its overall size, and the number of natural partitions. It is always possible to partition a given hypermedia application. The partitioning might occur on a site, server, directory, or domain name level. In many design models the notion of a view over the hyper-base can be considered as a partition. OOHDM Abstract Data Views (ADVs), RMM slices or Abstract Design Perspectives (ADP) [1] are groupings, partitions, or collections of data for various different purposes and they enable a designer to deal with conceptual design issues in sections smaller than the entire hypermedia application.

Hypermedia applications are both information presentation applications and functional applications. Many systems employ some sort of dialogue with the user beyond allowing the user to choose navigational paths. Users can also modify, delete, or append information. A partitioning system must be able to encompass both the functions and information presentation needs of the application. The hypermedia space needs to be partitioned into units that are large enough to be useful but not too large to be unwieldy. The elements within a partition need to have some commonality. This commonality can be units of instantiation, units of fault containment, or units of locality.

2.1 Partitioning the Hypermedia Navigational Space Using a Granularity of Abstraction

Using a granularity of abstraction, the hyperapplication can be partitioned according to the concept of a Navigational Semantic Units (NSU). The NSU is centered on the user’s intention when navigating through a hypermedia application. The NSU can be an entity, or represent a task, or be a combination of tasks and entities. How to separate intentions becomes a difficult task. The person who designs the application will have to decide the extent of each component. Cheeseman [2] partitions components by using use case models and business concept models, then separating system services from business services. The business concept model is refined into a business type model, and this is used to develop a set of business interfaces. The components are then implemented to support the interfaces. The “core” business types have independent existence and are characterized as business identifiers that have no mandatory associations except to a categorizing type (i.e. a pointer to another type that is not an aggregation or composition association) [2]. Hypermedia applications can be similarly partitioned, with an emphasis on using Navigational Semantic Units as the level of granularity of a component. The NSUs can be considered as “core” business types. Employing navigational use cases and a Navigational Class Model to initially define the components, similar to the methodology suggested by Cheeseman, greatly increases the likelihood of the designer identifying successful partitions. An NSU represents a user’s intention. NSUs

themselves, however, can be viewed as abstracted entities. These entities are not necessarily concepts related to the underlying domain information structure [3]. The NSU's are an abstraction that are a combination of the elements of the Business Concept Model and elements of the use case model, with a further "intuitive" aspect of user intention. The following are the three main types of components that constitute a hypermedia application:

NSU Components The majority of the components in a hypermedia application are expected to be NSU components. These components will consist of a session bean that contains all the dynamic data to be displayed, the dynamic links, the actions that constitute the interface, and the views to display the data. The NSU components can provide associative paths to other parts of the same NSU or to specific information in other NSUs.

Structural Components The Structural Components are those that provide navigational paths between the components. The Structural Components render the menus needed to facilitate the structural navigation. The extensibility of the component framework is dependent upon the navigational structure of the application.

Business Process components The Business Process Components are those logic components that reside in the presentation layer. These components may have some sort of presentation (views), but are generally used by NSUs or Structural Components in order to affect some aspect of the presentation. Such aspects might be user tracking, user profiling, user configuration, etc. Business Process Components will typically handle user login and authorization, and any other business processing needed that falls outside the NSU components or is applicable to the entire application.

3 A Component-Based Hypermedia Development Model

We propose the following steps for a component-based hypermedia development model:
Step 1. DOM Design–Navigational Model All the information that is needed in the hyperapplication might not be known, but an initial DOM can be built. The designer must determine the data and relationships that exist in the underlying system that are needed in the hypermedia application. The data and relationships can be: 1) a *Class diagram of the underlying application* that describes the information that is needed by the application; 2) *Use Case Diagrams* that describe all the tasks that encompass the Web site; and 3) a *Model Class Diagram* of the DOM that will describe the information that is to be displayed in the Web site. The DOM contains the entities and their operations as viewed from a presentation or user perspective.
Step 2: Conceptual Grouping. The designer must determine the conceptual grouping in the same manner as components are determined. The groupings can consist of Navigational Nodes that are strongly related to each other in terms of the user's intentions or functionality. The Nodes are grouped into NSU units. The NSU can contain information from all the classes in the DOM.
Step 3: Navigation Requirements. Determine navigation needs: 1) Determine structural navigational needs (navigation between NSUs) and create structural components; 2) determine associative needs (calls to information in NSUs, outgoing links to other NSUs); and 3) determine functional needs (within components).

Step 4: Component Requirements. Refine the conceptual boundaries of the component. This will drive the data and display requirements: 1) determine page templates; 2) for functional processes, determine the HTML forms (use case diagrams of functional processes will give insight about the needed pages /steps); 3) determine navigation within components.
Step 5: Component Interface Design The interface contains the operation specifications for what the component needs to display. For each page that is meant to be accessible by incoming links, assign an action and determine the parameters needed to gather the information for the view. The interface services calls from within the component and from other components
Step 6: Session Data Design Determine what data needs to be available for each view in terms of data from the underlying application, titles, and labels. Determine calls or other data retrieval procedures needed to get the required data. Each data element will be represented in a session bean. This includes data structures to hold the results of searches and collections of information.
Step 7: Construction of the Presentation Construct view frameworks (template pages) to display the data. These would typically be either JSP pages or programs to generate HTML. A framework can be reused to display different data. The view should contain no processing of data, only code to drive the display such as "mode" sensitive branching (i.e. toggle attributes).
Step 8: Displaying the Data Determine specific display details. A page can be considered as an observed composite that has a set of perspectives. The data can be displayed differently for different users depending on their access to data, their preferences, or even the previously viewed component.

4 Conclusion

This paper proposes a conceptual paradigm for partitioning a hypermedia application. Using a granularity of abstraction based on Navigational Semantic Units, a designer can construct a set of hypermedia components that can be individually constructed and maintained. Three types of components are introduced: the NSU, the Structural Component, and the Business Process component. Each of these components reside in the presentation layer. A methodology to construct the components is suggested. The methodology is similar to a typical component construction methodology. Use case diagrams, component dependency diagrams, and class diagrams are used as examples to guide the specification of the components.

References

1. German, D.: Hadez, a Framework for the Specification and Verification of Hypermedia Applications. PhD thesis, University of Waterloo (2000)
2. Cheeseman, J., Daniels, J.: UML Components: A Simple Process for Specifying Component-Based Software. Addison-Wesley (2001)
3. Cachero, C., Koch, N., Gomez, J., Pastor, O.: Conceptual navigation analysis: a device and platform independent navigation specification. In: 2nd International Workshop on Web Oriented Software Technology. (2002)