Component Integration for Web Based Applications

RICHARD A. WASNIOWSKI
Computer Science Department
California State University Dominguez Hills
Carson, CA 90747, USA

Abstract

Abstract: - We describe a component-based infrastructure for Web-based applications that reduces the need for custom integration. We assume that to construct an application, users will give a specification of three kinds of components: data source, business analysis, and visualization unit. From this specification, our infrastructure will generate the desired application. To support this vision, the infrastructure contains an intelligent component integration system that automates component retrieval, adaptation, and integration.

Key-Words: - Component, Integration, Web Based Applications

1 Introduction

The types of problems that web application developers are solving today require the use of a diverse set of tools that operate in many domains. There is obviously a subtle distinction between a web application and a web site. For the purpose of this paper a web application uses a web site as the front end to a more complex application.

In order to provide flexible tool integration platform for component integration must allow tool developers to target different levels of integration based on the desired level of problem complexity, time/space limitations and specific tool needed. The Unified Modeling Language emerged in response to a need for today's interactive computing systems and that can guide in constructing them. In the UML framework, software design entails building an object-oriented representation of a system, as well as of its environment, e.g. its users. Interactive systems such as modeled with UML represent a new paradigm in computation that inherently cannot be modeled using traditional, or algorithmic, tools. The essence of this computing paradigm is the notion that a system's job is not to transform input to an output, but rather to provide service. When a system is viewed as a service provider, the interaction between the system and its environment becomes an integral part of the computing process. UML presents a uniform domain-independent framework for modeling the different interactions present in today's systems: interactions among objects or software components, interactions between users and applications, interactions over networks and multilevel integration of various components. Each integration level determines what end users can expect as a result. This paper analyzes the different levels of tool integration and gives an overview of how they work. Numerous software toolkits and libraries are available to support the construction of web applications [7,17,19,20]. However, due to differences in data formats and system implementation styles, using these systems often requires additional custom programming that is time-consuming, expensive and usually not reusable. Web application software is commonly developed from prototype systems and evolves through experimentation. As the software is expanded and generalized, it becomes difficult to modify and maintain due to this ad-hoc development style.

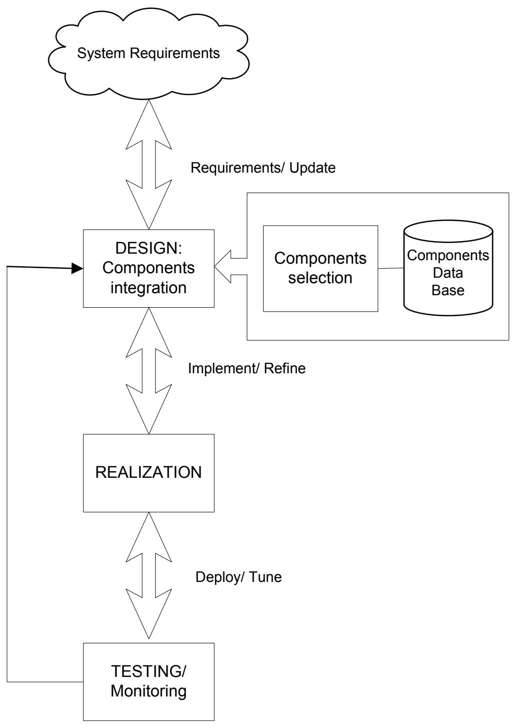
Our approach to deal with these problems is based on component based software development. Component based software development is based on the concept of building software systems by selecting, adapting and integrating a set of pre-engineered and pre-tested reusable software components. Component based software development has the potential to: reduce cost and development by allowing systems to be built from reusable components, enhance software reliability as components undergo evaluation during each use, improve maintainability and extensibility by allowing plug-and-play component replacement , and enhance the quality of applications by allowing application-domain experts to develop components. To provide this capability, the infrastructure contains an intelligent component integration subsystem that automates component retrieval, adaptation, and integration. It takes the user's request, locates the corresponding components in a database and uses knowledge about the components to generate adapters required for component integration.

In the next section, we describe the framework for component-based software development that is the foundation for the intelligent component integration system. We then describe how the intelligent component integration system is integrated with client/server architecture to support scalability and concurrent generation of multiple applications in a heterogeneous and distributed computing environment.

2 Component integration framework

Web based application is a combination of distributed and heterogeneous components. These applications include modeling and simulation for solving complex scientific and engineering problems. In web based application there are different layers in the architecture that represent application. Each layer consists of components that reflect the functionality of the layer. In the following section, we introduce the general architecture of web application in the web environment.

The process of design has some fundamental characteristics that cut across the different focus areas, and in fact, across the design of all aspects of a complex system, going beyond the Information Systems aspects alone, as illustrated in the following simplified process diagram:



The results of specification retrieval and matching are used to guide the selection of adapters that are used to integrate components. The system contains specifications of standard adaptation strategies that are used in the system. A strategy determines which adapter to generate and provides heuristics describing how to specialize it to the application being generated. Adapters are associated with each of the matching relationships that could exist between user request and library components. These adapters can be mapped into Java components using event-based component integration. The component retrieval and adaptation approach outlined above supports a bottom-up development style. Web based applications are applications that make heavy use of a database, document management systems, web-based repositories and search engines, and workflow systems that capture and support data-oriented business processes. Design of a complex web based application system is an ongoing exercise in finding the right configuration in a high-dimensional space of design choices. We use the term component integration broadly to denote design models and theories, design methodologies, and tools that build upon them. They are especially valuable in dynamic environments where the system must constantly react to changes in the environment, or be re-configured frequently to respond to changes in the objectives of the enterprise. This paper dealt with the problem of focused crawling using component integration intelligent crawling agent based on agent based learning. A crawler must be able to recognize patterns within the Web graph and make the right choices in order to be efficient and cost-effective. Agent based learning was chosen because it is a method that allows an agent to accumulate knowledge by experimenting with the environment, without using direct supervision. It seems to be appropriate for the task of Focused Crawling where success can be recognized but detailed guidance to this success cannot be provided, as would be required by a supervised learning approach.

3 Integration example

Focused crawling is a relatively new, promising approach to improve search on the Web [1,13,19,20]. A Focused crawler searches the Web for relevant documents, starting with a base set of pages. Each of these pages contains usually many outgoing hyperlinks and a crucial procedure for the crawler is to follow the hyperlinks that are more probable to lead to a relevant page in the future. Therefore, the crawler must include a component that evaluates the hyperlinks, usually by assigning a numerical score to each one of them. The highest the score is, the more probable it is that this hyperlink will lead to a relevant page in the future. This component is implemented here by a classifier/distiller learning agent. This agent recognizes different states of the environment and for each of these states it is able to choose an action from a set of actions. The choice of the action that the agent will perform is represented simply as a look-up table.

Another important factor of this system is the environment. The environment judges each of the agent's choices (actions) by providing a numerical reward. The reward is indicative of what we want the agent to perform, but not how it will perform it. When a reward is given, the course of actions that the agent has followed so far gets credit. This is a promising solution to the central problem of focused crawling, which is to assign credit to all the pages of the path that leads to a relevant document. Our aim is to construct a focused crawler that uses an agent to train the link scoring component. This crawler should have increased ability to identify good links, because of the agent based scheme, and therefore become more efficient and faster than a baseline crawler.

Focused crawling starts from a user-specific or group-specific set of topics along with a training set of documents and crawls the Web with focus on these specific topics of interest. The key components of a focused crawler are classifier used to test whether a visited document fits into one specific topic of interest, and distiller to identify the best URLs for the crawl. The quality of the training data is obviously the most critical issue for both components. The focused crawling process can either build a personalized, hierarchical ontology whose tree nodes are populated with relevant documents, or it can be initiated to process a query. Our approach myCrawler integrates the following components: the crawler, document analyzer, the SVM classifier, the feature selection filter and the training module for the classifier. The support vector method was developed to construct separating hyperplanes for pattern recognition problems. In the 1990s it was generalized for constructing nonlinear separating functions and for estimating real-valued functions. Applications of SVMs include text categorization, character recognition and face detection. The main idea of the SVM approach is to map the training data into a high dimensional feature space in which a decision boundary is determined by constructing the optimal separating hyperplane. Computations in the feature space are avoided by using a kernel function. The formal goal is to estimate the function $f: R \rightarrow \{+1,-1\}$ using input/output training data such that f will correctly classify examples. Simply minimizing the training error does not necessarily result in good generalization. Support

Vector classifiers are based on the class of hyperplanes and corresponding to the decision function f. The unique hyperplane with maximal margin of separation between the two classes is called the optimal hyperplane. The optimization problem thus is to find the optimal hyperplane. If f is a nonlinear function one possible approach is to use a neural network, which consists of a network of simple linear classifiers. Problems with this approach include many parameters and the existence of local minima. The SVM approach is to map the input data into a high, possibly infinite dimensional feature space, via a nonlinear. This high dimensionality leads to a practical computational problem in feature space.

The system is implemented in Java and runs under Web application server. Some components are also implemented as stored procedures under mySQL and other components are Java servlets under the Apache web server. All documents that the crawler fetches are stored in mySQL database. myCrawler uses an open-source implementation of a support-vector-machine (SVM) classifier. The classification proceeds in a top-down manner starting from the root of the ontology tree. For each node the classifier returns a yes/no decision and a confidence measure of the decision. The document is assigned to the node with the highest confidence in a yes subtree. Next, the classification proceeds with the children of this node, until eventually a leaf node is reached. myCrawler periodically re-trains the classifier to improve the effectiveness of classifications. It is clear from experiments that the accuracy of the ontology's documents improves as the crawl proceeds and undergoes re-training.

4 Conclusion

Application development, and in particular Web Clients are demanding high-quality applications in Web time to support critical processes. Modern Web applications consist of many loosely coupled technologies requiring diverse, often distributed teams, to create highly linked resources that are seldom collectively managed. Addressing this "software paradox" and managing application development complexity requires best tools that can be selected and integrated to provide a roles appropriate, end-to-end development environment tailored to individual development processes. But integrated tools require a platform of services, frameworks, and standards that allow vendors to focus on their value-add while reusing common infrastructure. The platform must include a workbench that provides a common view of the whole application across all resource types and the entire team. And the platform must be accessible to tool vendors under an acceptable license. We have outlined the architecture of component integration system that supports the component based development of web applications. The system relies on specifications to describe components, wrappers, and architectures and uses automated reasoning techniques to retrieve, adapt, and integrate components.

The results of the experiments based on focused crawling show that agent based is a good choice for this task. Indeed, in most of the cases only a small number of steps were required in order to retrieve all the relevant pages.

Further work includes further experimentations and potential extension of the method, incorporating features of the method used in myCrawler.

Our approach not only provides such a platform, but its architecture also provides flexibility in how tool integrate and at what level.

References

[1] Aggarwal C., Al-Garawi F. and Yu P. *Intelligent Crawling on the World Wide Web with Arbitrary Predicates.* In Proceedings of the 10th International WWW Conference, pp. 96-105, Hong Kong, May 2001.

[2] Brin S. and Page L. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In the Proceedings of the Seventh International WWW Conference, pp. 107-117, Brisbane, April 1998.

[3] Bosak, J. and Bray, T., "XML and the Second-Generation Web", Scientific American, May 1999.

[4] C.J.C. Burges: A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Vol.2, No.2, 1998.

[5] B. Fischer, J. Whittle: An Integration of Deductive Retrieval into Deductive Synthesis. Proceedings of the 14th International Conference on Automated Software Engineering (ASE-99, October 1999.

[6] Chakrabarti S., van den Berg M. and Dom B. *Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery.* In Proceedings of the 8th International WWW Conference, pp. 545-562, Toronto, Canada, May 1999.

[7] CROSS-lingual Multi Agent Retail Comparison. http://www.iit.demokritos.gr/skel/crossmarc.

[8] Hersovici M., Jacovi M.,Maarek Y., Pelleg D., Shtalhaim M. and Sigalit U. *The Shark-Search Algorithm - An Application: Tailored Web Site Mapping.* In Proceedings of the Seventh International WWW Conference, Brisbane, Australia, April 1998.

[9] Karkaletsis V., Paliouras G., Stamatakis K., Pazienza M.-T., Stellato A., Vindigni, M., Grover C., Horlock J., Curran J., Dingare S. *Report on the techniques used for the collection of product descriptions*, CROSSMARC Project Deliverable D1.3, 2003.

[10] Orfali, B. and Harkey, D., *Client/Server Programming with Java and CORBA*, Wiley, 1998.

[11] Pour, G., "Component Technologies: Expanding the Possibilities for Web-Based Enterprise Application Development, "Chapter in, Internet Technologies and Applications: Advanced Research, Theory, and Practice, To appear, 2000.

[12] Dorin Petriu, Murray Woodside, "Software Performance Models from System Scenarios in Use Case Maps", Proc. 12 Int. Conf. on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (Performance TOOLS 2002), London, April 2002.

[13]Grigoriadis A, Paliouras G., "Focused Crawling using Temporal Difference-Learning". Proceedings of the Panhellenic Conference in Artificial Intelligence (SETN), Lecture Notes in Artificial Intelligence, Springer Verlag, 2004.

[14] Khalid H. Siddiqui, C.M. Woodside "Performance aware software development (PASD) using resource demand budgets" In the Proceedings of the third international workshop on Software and performance, pp.275 – 285, July 2003

[15] McCallum A., Nigam K., Rennie J. and Seymore K. Building Domain-Specific Search Engines with Machine Learning Techniques. In AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford University, USA, March 1999.

[16] Murali Sitaraman, Greg Kulczycki, Joan Krone, William F. Ogden, A.L.N. Reddy "Performance Specification of Software Components", Proceedings of SSR '01, pp. 310. ACM/SIGSOFT, May 2001

[17] Cervantes Hector, Senior project,2004

[18] Stamatakis K., Karkaletsis V., Paliouras G., Horlock J., Grover C., Curran J.R. and Dingare S. Domain-specific Web Site Identification: The CROSSMARC Focused Web Crawler. In Proceedings of the Second International Workshop on Web Document Analysis (WDA2003), p.75-78. 3-6 August 2003, Edinburgh, Scotland.

[19]The BINGO! Focused Crawler at the TREC WebTrack (Poster). Martin Theobald, Sergej Sizov. Text Retrieval Conference (TREC), Gaithersburg, Maryland, 2003

[20] Wasniowski R., "Improving the Web Search Performance," RAW-94- 32, June, 2004.

[21] V. Vapnik: Statistical Learning Theory. Wiley, New York, 1998