ISSN:0975-9646

A Task Oriented View of Web Software Visualization and Architecture

Laxmareddy.A[1], M.Ganesan[2], DR.E.Kannan[3], M.Dhilsath Fathima[4], M.S.Saravanan[4]

[1,3]*Dept. of Computer Science and Engineering*
[2,4]*Dept.of Information Technology*
Veltech Technical University, Chennai.

Abstract:Software visualization studies techniques and methods for graphically representing various ways of software. Its main purpose is to enhance, simplify and clarify the representation a software engineer has of a computer system. Software systems grow in size and complexity, so it difficulties to understand and maintaining the software.In this paper existing software visualization tools require too much time for end user developers to learn and make effective used. We are currently building a web software visualization application that allows end-user that will provide create, view, save, share visualizations and it will show the output like web visualization like histograms etc..,

Keywords – Software Visualization, Information Visualization, Software Comprehension, user – computer interaction and visualization of words.

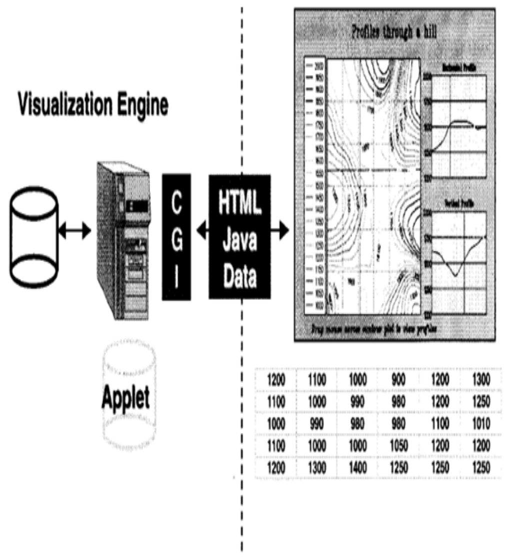
1. INTRODUCTION

Software visualization represents many things to many people. Price presents the following general
Definition of software visualization
“Software visualization is 2D or3 D Visual Representation of information about Software based on their structure with modern user-computer interaction and computer graphics technology to
Facilitate both the human understanding and effective use of computer software.”
Or “Visualization Software is a range of Computer Graphics and used to create graphical display and interfaces for software programs”.
Web services describes the information, existing software’s, and make them available internet to utilize standard interfaces and protocols such as Simple Object Access Protocol (SOAP), Extensible Markup Language (XML) etc.., Web services provides a implementation way of the loosely coupled Service-Oriented Architecture (SOA) which is widely accepted by the information industry as the future internet application architecture and ease the process of application integrations. However, the true potential of SOA and web services, in general, can be realized when they are used for creating new services by composing existing services together. In other words, when a requested functionality cannot be offered by a single service alone, some of the existing ones can be composed to provide the same functionality. This process of web service composition requires effective web service semantic description language and tool to facilitate quick and simple composition of web services, especially for end-users.
However, we focus on the theory part of the composition and formalization of web service. A lot of web service semantic description languages have been made available by many researchers and organizations to describe web service compositions. However, little progress has been made in the direction of describing the dynamic transformations among atomic components of web service compositions. Most of the research projects on web service composition have led to the development of web service semantic description language. Furthermore, these technologies handle only part of the problem and not deal with dynamic transformations. What is required to facilitate web service composition is an ontology-based web service semantic description language, which is capable to describe dynamic transformations among components of web service compositions, to ease the process of web service composition, thereby reducing end-users’ demands obtaining time, integration efforts and composition expenses. This web service semantic description language should be able to describe the dynamic transformations among the components of a composite service in a manner.
The Other Part of this Abstract looks at existing software visualization tools, characterizes web information visualization tools and discusses implications.
The Web introduces a new model in which the client GUI, based on HTML, is less functional and relies upon the data or application servers for visualization traditionally will be executed and that display in pixels.

1.1 *Thin Client*

The Web services in the world, the client are effectively reduced to a browser viewer of information supported by a server.
A true Web client is not capable of program execution unless the executables are downloaded to the client as either Plug-ins or Components. This client is normally referred to as the “thin” client.
A thin client, by definition, have minimal software requirements necessary to function as a user interface front-end for a Web enabled application. Local data manipulation, information drill-down technique,
Context sensitive menus, object picking and other interactive user interface functions that traditionally have been available on the client are now controlled by the visualization server. In the “thin” client model, nearly all functionality is delivered from the server side of the visualization engine while the client performs very simple display and querying functions.
The most appealing aspect of the “thin” visualization client to information visualization users is that the overall cost of software and maintenance can be dramatically reduced. The

"thin" client allows the application developers to eliminate the notion of software distribution at the client level, eliminate the notion of maintaining local software and supporting multiple operating systems on remote clients.
The concept of thin client, the issue of client and server data visualization will be standard Web browsers are "static" and do not permit any visual data manipulation at the client side. The user interaction is dependent upon the network bandwidth. Partitioning the visualization process between clients and servers is an effective way to distribute the computing resources. The most flexible visualization system allows the application developers to control the visualization partitioning.



1200	1100	1000	900	1200	1300
1100	1000	990	980	1200	1250
1000	990	980	980	1100	1010
1100	1000	1000	1050	1200	1200
1200	1300	1400	1250	1250	1250

Figure1: Overview of Web Server Architecture is describes 2D graphics of java 2D API Applet is transferred as to the client. A second applet performs the calculation of profile code and the data should be selected. The Horizontal and Vertical profiles are displayed and calculated as line graphs based moments of the user. Both Profile Calculation and Graphics rendering is done on the client – side. The 2D graphics was produced on server – side by visualization in java2D API class. The advantage of local data manipulation in client and improves the text quality in server architecture.

2. Evaluation Of X3D

X3D can support a range of 3D software visualization techniques to determine if the technology is viable for use in software visualization. More precisely we want to experiment with automatically creating X3D software visualizations, evaluate X3D's animation and test the integration capabilities, and analyze the performance display capabilities of X3D.
The UML diagram replicates a similar example by[McIntosh et al.2005].
Design: X3D is free open standards file format and run-time architecture to represent, communicate, and deploy 3D scenes and objects over the web using XML. The X3D specification is comprised of components which contain nodes (e.g. geometry) that are declared in a graph scene that content can be created using some text editors, X3D editors (e.g. X3D-Edit a Netbeans plug-in), digital content creation tools (3DS Max, Maya, Blender), or XSLT transformations. Here used some three of the main X3D browser free version implementations including (in order of preference): BS Contact VRML/X3D Player (6MB download), Octaga Player (5MB), and Flux Player (1.5MB). Each of these X3D browsers operate on Windows and can be plugged into Mozilla Firefox and Microsoft Internet Explorer or operate as stand alone. There is also the Web3D Consortium's stand-alone open source test-bed implementation Xj3D (12MB). X3D content can be rendered in either OpenGL or DirectX. Some of these X3D browsers do not implement all of the X3D specification nor do they make the Scene Access Interface (SAI) run-time API available. This makes it hard for developers to create consistent X3D software visualizations. Graphical Capability: A rich set of graphical elements exist to create high quality visual pictures required in visualization software .the points and lines are can be implemented using nodes from 2D. Shapes have size, height, radius, color, and transparency fields, and can be animated to change in software visualization. Shapes can be orientated in any order (e.g. translated then rotated) and change position during a software visualization. When nodes that are connected in graph visualization are moved in a scene, scripting is required to preserve the node-link relationships.

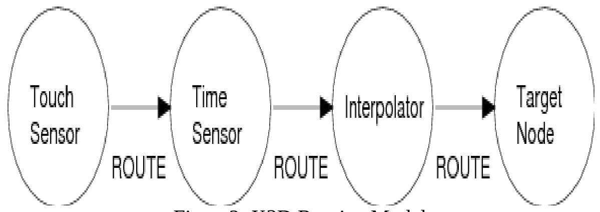


Figure2: X3D Routine Model

Visualization Techniques: There is no specific software visualization component or library. We replicated a range of software visualization techniques in X3D including algorithm animations, 3D UML diagrams (class, package, and sequence diagrams), documentation related visualizations (source code, API Java doc , and video visualizations), and execution trace visualizations (3D compound shapes and 3D information visualization metaphors). The data for our visualizations have been encoded using three different approaches: in the X3D scene, transformed from XML execution traces into X3D geometry primitives are will be node prototype ways and our visualizations can provide Java and C++ programs. X3D can be used to represent program synchronization. Multiple views can be accomplished by displaying the data in different positions in the scene or integrating external web pages.

2.1*Visualizing Software Architecture*

Visualizing Software Architectures As the size and complexity of software systems increase the design problems behind the algorithms and data structures of the computation: Designing and specifying the overall system structure emerges as a new kind of problem.
According to an IEEE standard [IEE00] "Architecture is the fundamental organization of a system in those components of relationships to everyone and to the environment and the principles guiding its design and evolution".In some keywords are in this definition are structure, environment and principle. As will become apparent in this chapter, visualization so far has

concentrated mostly on the structure. Visualizations of software architectures typically deal with the structure at various levels of abstraction. At a higher level, the architecture consists of components with ports, and ports are linked through connectors.

When it comes to actually describing architecture, there are many aspects, including its gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives.

Most of these aspects have both functional and nonfunctional properties and are often described in natural language illustrated with diagrams. Breaking a system into modules facilitates the design and development of software systems. As David Parnas put it, “The existence of the hierarchical structure assures us that we can ‘prune’ off the upper levels of the tree and start a new tree on the old trunk.”

It is widely accepted in software engineering that when one is designing software architectures there should be low coupling between modules and high cohesion within modules. Coupling is “a measure of the strength of the association established by a connection is between one module to another module while the degree of connectivity among the elements of a single module.

However, these original notions of coupling and cohesion do not take the direction of the dependencies into account. In particular, in relation to object oriented design, these original ideas have been extended and new principles and related metrics have been devised to guide the design of software systems. The first part of this chapter is about types of diagrams that show the structure of architecture. Then we shall look at some approaches that can be used to extract and visualize architectural information from the source code of a system. Finally, the use of 3D and dynamic software architecture visualization is discussed.

2.2Some Familiar Architecture

We first look at some diagrams of some basic architecture widely used in software systems:

Pipes and filters: Filters receive a stream of input data and produce a stream of output data. Pipes pass the output data of one filter as input data to the next one. The Info pipes notation extends the pipe metaphor with buffers, pumps, split and merge tees, etc. to design distributed streaming applications.

Layered Systems: The functionality of a system is organized into several layers. In a purely layered system, the functionality of one layer is implemented by the functionality provided by the layer directly below. Very often, layered systems also allow access to some of the other layers below. One can use both horizontal layers and an onion model. In the first case all layers have the same size, whereas the onion model emphasizes that the core is smaller than the outer layers.

Blackboards: In the blackboard architecture, there are multiple units that share data through a blackboard. Typically, the units can read and write to the blackboard. So for example, the blackboard might contain both tasks to be computed and results of previously computed tasks.

A Web search with a search engine such Google for images related to the term software architecture reveals a wealth of different styles for drawing architecture diagrams. Most of these use ad hoc visual representations, and the semantics of the colors, nodes, icons, lines, and arrows is often unclear. To remedy this situation somewhat, one can follow general rules for the use of connectors, icons, text, color, etc. On the basis of a study of software architecture diagrams found on the Web. Compiled a list of guidelines for drawing architecture diagrams but when it comes to building large systems with many developers, a common understanding of the architecture diagrams is key, and standardized graphical notations such as UML promise to be the solution.

3.WEB SERVICE FLOW PATTERNS

Web services are currently praised as the solution to the development of distributed applications by allowing one to compose services in a standardized .As more Web services become available, applications get larger and more complex. The Web Services Navigator is a visualization tool for understanding, debugging and analyzing the performance of these complex applications. To this end, execution traces in the form of Web service Transactions are collected. A Web service transaction is a tree of messages and invocations that is initiated by a client. Thus, a transaction captures the flow of one service invoking one or more other services, which in turn may invoke other services and so on.

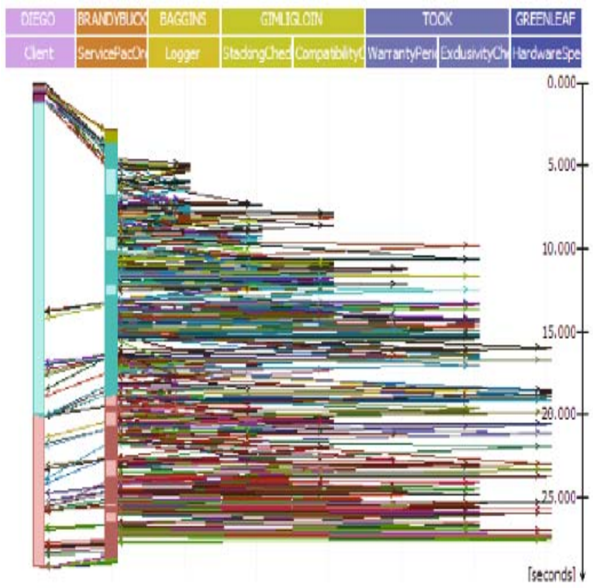
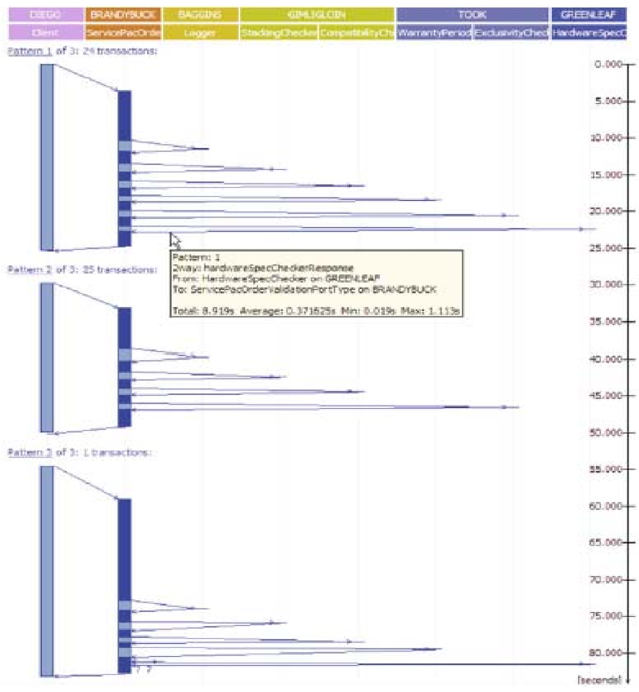


Figure3: Traces of several web service transactions

Transactions can be visualized using a sequence diagram-like representation, where vertical lines represent the different services involved. To test or tune the performance of a complex Web service, one typically needs to collect a large number of transactions. Simply drawing all transactions in a single diagram does not reveal any new insight other than revealing the fact that it looks like a mess. By partitioning the transactions into groups of isomorphic tree shapes, and then further subdividing these groups based on matching node and edge at Visual Testing.



Pattern: 1
2way: hardwareSpecCheckerResponse
From: HardwareSpecChecker on GREENLEAF
To: ServicePacOrderValidationPortType on BRANDYBUCK

Total: 8.919s Average: 0.371625s Min: 0.019s Max: 1.113s

Figure4: Flow patterns computed for many transactions

3.1Web Software Visualization

In a recent survey based on questionnaires completed by 111 researchers from software maintenance, reengineering and reverse engineering, 40% found software

Visualization absolutely necessary for their work and another 42% found it important but not critical. The majority of the researchers are primarily using or integrating existing software visualizations tools developed by others (33%).

We want software visualization to be an easy task for end-users without the need for downloading and installing separate applications. However, it is not clear what a good software visualization system looks like. We believe the web is an excellent platform for creating a software visualization application. Web based software visualization allows end-users to independently create, view, save, and share visualizations with others. We have explored the 43 software visualization tools

Listed by Diehl and found that only one of the tools was web-based, the SHriMP2 application for visualizing dependencies in hierarchically structured data as nested graphs.

The rest of the applications require a separate download, plug-in to an IDE such as Eclipse, or are proprietary software. Since there is a lack of freely available web software visualization tools we have decided to explore existing information visualization web tools. We want to see if any of these tools have useful features that we could incorporate into our software corpus visualization project.

There are some implications for our research. We need to determine what visualization types to implement. Depending on the visualization type different methods will be required to parse the source code. We intend to make our software visualization system public facing so we will need to consider how we handle

Proprietary software and how the system scales once there are lots of software uploaded and much visualization created. Once our application is in production we intend to conduct user evaluations (user testing and interviews) to see how effective the system is.

In summary, there is a lack of easy to use web software visualization systems. We are working towards a web based application that will help end-users to upload their Java software applications, create visualizations, and share their visualizations with other users.

REFERENCES

[1] Towards End-User Web Software Visualization 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).

[2] Anslow.Evaluating X3D for use in software visualization. Master's thesis, VUW, 2007

[3] Information Visualization on the Web Mikael Jem Advanced Visual Systems

15 Blokken, DK 3460, Birkeroed, Denmark and R. Koschke. Software visualization for reverse engineering.In Revised Lectures on Software Visualization, pages 138–150. Springer Verlag, 2002.

[4] Web Software Visualization Using Extensible 3D (X3D) GraphicsCraigAnslow, James Noble, Stuart Marshall_Victoria University of Wellington, New Zealand Robert Biddle†Carleton University, Canada

TEXTBOOK

[5] Stephan Diehl – Software Visualization Visualizing the Structure, Behaviour, and Evolution of Software .Springer

LINKS

[6] http://en.wikipedia.org/wiki/Software_visualization

[7] http://en.wikipedia.org/wiki/Visualization_software