Web Engineering with the Visual Software Circuit Board

Hovhannes Avoyan
Computer and Information Science Master Program
American University of Armenia
40 Baghramian Ave.,
Yerevan, Armenia 375019
+37 41 512045
havoyan@computer.org

Barry Levine
Computer Science Department
San Francisco State University
San Francisco, CA 94132
415-338-1661
levine@sfsu.edu

ABSTRACT
The Visual Software Circuit Board (VSCB) platform supports a *component based development methodology* towards the development of software systems. The circuit board design techniques and methodologies have evolved for electronic device and component engineering for decades. The circuit board approach, now applied for software systems and applications, makes the component based development process easy to visualize and comprehend. This paper describes the VSCB based design methodology with a specific focus on usage of VSCB for web application engineering.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**] - *Computer-aided software engineering (CASE), Evolutionary prototyping, Flow charts, Modules and interfaces, Object-oriented design methods, Software libraries.*

D.2.11 [**Software Architectures**] - *Domain-specific architectures, Patterns (e.g., client/server, pipeline, blackboard).*

D.2.6 [**Programming Environments**] - *Graphical environments, Integrated environments, Interactive environments, Programmer workbench.*

General Terms: Design, Economics, Reliability, Experimentation, Standardization.

Keywords
Rapid application development, visual programming, web application development, web engineering, component based development, circuit based software development

1.0 INTRODUCTION
We developed the VSCB methodology as an attempt to adapt a *circuit board component based development methodology* towards the construction of software systems. Electronic circuit board engineers use the sophisticated and rigorous techniques and methodologies that allow them to predictably build complex electronic devices. In a construction process similar to that of using microchips to build circuit boards, as well as composing more basic designs to build more complex devices, the VSCB methodology is used to construct software components and applications.

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

Software development utilizing the VSCB approach is quite intuitive. It was used in several case studies to build both interactive web and Java-based standalone applications. The design of interactive applications can be made easy and intuitive by visually drawing a flow of screens. The developer can use the VSCB approach to represent a flow of screens as components and connections. In this fashion, the web application logic is rendered transparently. Furthermore, re-engineering is simplified and, perhaps, even trivial. VSCB *diagrams are executable,* facilitating an effective debugging process. The usage of libraries of prefabricated components allows rapid prototyping and early feedback, thus ensuring high relevancy of the final product to the customer needs. Finally, we note that the VSCB approach allows the construction of highly reusable components and improvement of productivity and quality. Summarizing the benefits:

- VSCB provides support for a visual, drag and drop based web application development.
- VSCB makes Web application logic visible and shareable.
- VSCB makes Web Application easy to re-engineer.
- VSCB allows quick prototyping.
- VSCB improves development productivity and product quality via high re-use of components.

2.0 VSCB CONCEPTS
The building elements of VSCB are *parts, devices and wires*. Parts denote software components, while wires are the component connectors. We use different terminology here because the terms components, interfaces and connectors have widely varying interpretations.

Parts consume, produce and/or transform data. Parts are *anonymous collaborators*. That is, there is no coupling between different parts. Part interfaces are also standardized. The interfaces provide ports to input and output data. Each port is denoted as either an *inpin* (consumes data) or *outpin (*produces data).

Parts can be atomic or composed into *devices (*composed, or assembled, from atomic parts and/or simpler devices*)*. Atomic parts are implemented in an underlying programming language, such as Java.

Wires connect parts and forward data from one part to another. A wire connects part outpins with part inpins. Adding and removing wires do not affect parts. If a part outpin is connected to more than one inpin then the VSCB framework takes care of multicasting the output data.

VSCB users can use both *top-down* and *bottom-up* engineering approaches. Utilizing the *top-down* approach, the application

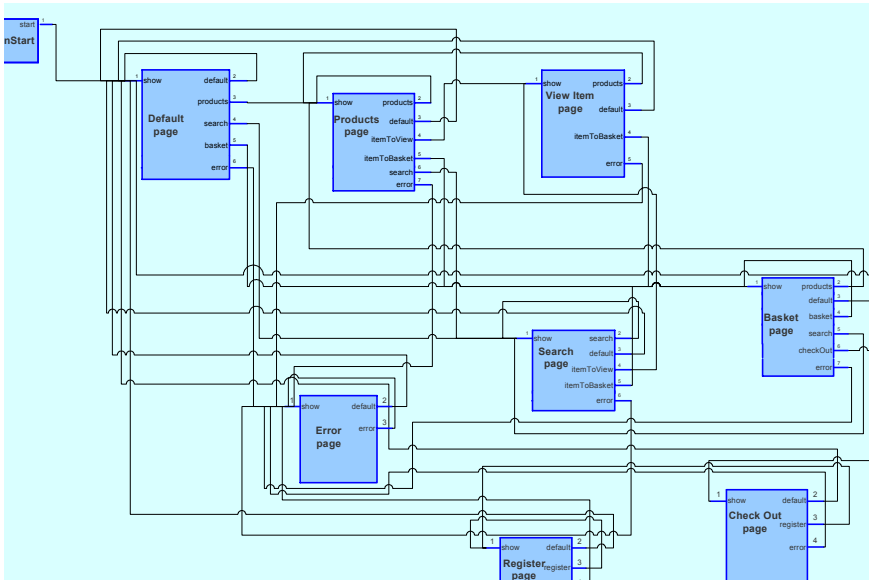

Figure 0. E-commerce VSCB diagram

developer initially creates abstract boxes in the high-level circuit board diagram. Subsequently, the developer constructs finer-grained devices via visual diagrams until reaching either an atomic or library part. In the *bottom-up* approach, initially, developers use library parts and, subsequently, utilize aggregation to build higher-level abstractions (devices). The combination of both approaches can be used.

The use of part libraries facilitates the rapid construction of basic applications. In this fashion, collections of standard parts will be provided to developers such as filters, parsers, iterators, aggregators, converters etc.

The Diagram Editor is a central and critical part of VSCB. It is used to develop VSCB configurations visually. The developer can drag and drop parts from libraries, run simulations and perform visual tracings.

3.0 VSCB BASED WEB ENGINEERING

VSCB can be used to draw the web application flow visually and execute the resulting diagrams. VSCB uses a part-and-wires circuit-board-like diagram to represent flows of web pages. On the high-level diagram, parts represent web pages and wires link the pages. In Figure 2, we provide a simple e-commerce application diagram. Parts denote web pages such as *Products* and *Search* pages. Outpins represent the actions which could be triggered by a user clicking links to submit forms.

The most important VSCB feature associated with this diagram is that the wires (rather than, e.g., URLs in html pages) dictate the flow. This means that we need only re-wire the diagram to modify the web application.

Users trigger actions by clicking links or *submit* buttons on the web pages. The VSCB diagram represents actions by outpins at each web page device. There is one action associated with each submit, as well as important (in a context of flow) links. Inpins for the web page device trigger the page generation process. On the VSCB diagram, outpins are connected to inpins. Thus, every action of a user yields another screen. Within each page device there is a controller part, which interprets the user action and fires an appropriate outpin. To do this, the controller parses the request URL produced **after** the user receives the page produced by the current device. This scenario describes the only place where the URL really matters; it is within the scope of the page, which produces the request. The framework, which accepts the fired data, will navigate further to the next page according to the wires.

As mentioned above, the web page parts on the high-level diagram are devices because they are composed from other parts. These web page devices in turn contain the model, presentation and controller parts. The model parts are responsible for content generation, while the presentation parts are responsible for html rendering. The controllers are responsible for handling user input. Model category parts are usually database parts, which query databases and generate the results as XML output. Presentation parts usually implement XSLT transformations and controllers are usually some filters which check the user-initiated submit or link actions and fire appropriate outpins. We will not elaborate on the details of the aforementioned scenario in this paper. We note the dispatcher part is a servlet, which behaves like a state machine. It 'remembers' from which pin it was fired for sending a response and then fires the appropriate pin when it gets a request

4.0 CURRENT STATUS

The VSCB project began during spring 2003 as a project in the graduate level course *Advanced Software Engineering: Software Architecture* taught at the American University of Armenia [1]. The project continued as an open source project under Open Source Armenia [2].

Our ideas were inspired by Jack Harich's VCB project [3]. Unfortunately, the VCB project was discontinued; we used similar terminology with generous permission from Jack Harich.

During 2003 five masters projects were developed based on VSCB. These projects included E-Commerce and Web Applications, UI applications and Web services orchestration. Some modules are available for download. At this point, these prototype modules have been downloaded 3500 times.

We expect that VSCB will have an important impact on the information/computing industry. Component Based Development (CBD) and Visual Programming (VP) are increasing in their importance with respect to improvements in productivity, predictability of software development and quality of products.

5.0 REFERENCES

[1] American University of Armenia: http://www.aua-mirror.com

[2] Open Source Armenia: http://vcb.opensourcearmenia.com

[3] Visual Circuit Board: http://www.jcon.org/projects/vcb