Ur/Web: A Simple Model for Programming the Web

Matthew Campbell

COP 4813 – Internet Programming

Jim Littleton

April 18, 2020

**Introduction.** For this assignment, the instructor had assigned the task of a paper review. This paper will introduce the topic, present the major ideas, and discuss the impact the article might have made in the field of internet programming, The topic of the paper is Ur/Web, a programming language that provides a simple framework for developing web applications. The article has been broken up to five sections, which are the introduction, a Tutorial Introduction to Ur/Web, Related Work, and Conclusion.

**Article Introduction.** The very beginning of the article goes through the brief history of web development. Back a couple of decades ago, there were not strict standards on the methods used to create web pages. As a result, there are so many things that are quintessential in order to build a web page, which include HTML, JavaScript, and CSS. More recently, other languages were added to keep track of the data that is being utilized, like SQL and JSON. The author has proposed a language that will help the programmer. One way that the language will help programmers is that the language uses encapsulation, which privatizes the essential processes to a web application. Another way that the programming language will help the programmer is that it will provide simple concurrency. Ur/Web also uses an easy-to understand programming model to keep track of all of the standards that are used to developing web applications.

**A Tutorial Introduction to Ur/Web.** The author has decided to present the programming language in a form of a tutorial. Going on to the major concepts of the paper, one of the first ones that struck out to me is encapsulation. In the very first coding example, the author had left the database exposed to other parts of the application. This makes it easier for hackers to access the application and cause havoc. The author’s solution is to encapsulate the database into a module so that way it is only accessed by the methods that need to access it. In addition, the results from encapsulation emulates the Model, View, Controller setup.

**Client-side GUI Scripting.** The second major concept that I would like to mention is the client side-GUI scripting. There are two aspects of client-side scripting that are mentioned by the author, one of which is the reactive GUI. The GUI style of the Ur/Web programming language is that user inputs provide the raw data for the environment to change into what the user would see on the screen. Ur/Web uses the raw data and puts them into streams, which are the input for the data sources. The streams can be used for various tasks, such as for online transactions for shopping carts and messages in a chat room. Moving on, I will talk about the second aspect of client-side GUI scripting, which is about remote procedure calls. The author mentioned that there needs to be a way for the client to contact the server. One of Ur/Web’s solutions is for the client to run callable methods, imitating the server in the process. All that needs to take place is for the programmer to insert the characters rpc to indicate that data needs to be sent to the server.

**Sending a Message From A Server to the Client.** The last major concept that the tutorial goes through is sending the message to the client. Ur/Web uses some abstraction that will allow the server to send the message to any client(s). The communication is done via channels that go in a single direction. For each channel, there is a client and type that is partnered with it. Once the client is hooked up to a channel, the server may send values of a certain type to the client. Sometime after the message has been sent, the clients receive the messages from the channels that they are taking part of. As for threading, the clients can run on either no threads or one threads. With the server, it can run a whole bunch of threads at once. As a result, the main program can choose which channel it can send messages to.

**Related Works.** The author compared other research languages to Ur/Web. One of the most notable comparisons is that Ur/Web is the only language that can encapsulate the database into a module. The comparison that the author made here is that Ur/Web can send different parts of the input stream to separate parts of the program. Moving on, the author had stated a popular tool for Web Development, which is Meteor. One of the features that Meteor has is that the database changes on the server automatically goes into the client’s files. However, when a reactive webpage is experiencing an update, Ur/Web has a simpler API than Meteor’s. Afterwards, the author states that Ur/Web’s five lines of APIs is much simpler than the taxing approaches of the other languages.

**Article Conclusion/Future Impact.** The final statements that the author makes is that Ur/Web is used in the workplace today and that Ur/Web performed very well in benchmarking tests against other languages. The main purpose of the article was to advocate for the usage of encapsulation in specific aspects and multithreading. As for future impacts, the programmers will implement encapsulation and multithreading into their websites. In addition, programmers will start getting familiar with function-oriented languages since the author highly endorses Ur/Web, which is a function-oriented language. Another future impact is that programmers will start to critically evaluate to see if their typical language(s) can do the functionality that was discussed in the article. To conclude the review, the author did a good job of explaining the concepts. However, I would recommend that a person gets familiar with functional-oriented languages before he/she reads this article because I got the impression that function-oriented programming is different from object-oriented programming.

References

Chippala, Adam (2016). Ur/Web: A Simple Model for Programming the Web. Retrieved March 13, 2020.