Fusebox for reuse and ease of maintenance in a large web application

Wayne Gray **Satoko Mowbray** **Tony Clear**

School of Computer & Information Sciences
Auckland University of Technology, NZ
tony.clear@aut.ac.nz

1. Introduction

The capstone project we are undertaking as part of our degree is a continuation of a previous project (Charkova, Clear, Lin & Lomax, 2003) that focused on creating an online membership register for Te Runanga A Iwi O Ngapuhi (TRAION). As a result of that project a large robust object oriented web application was created using PHP & MySQL, it consisted of around 250 PHP files in 2 folders. When the current project started it was intended that we take the software created previously, merge it with TRAION's existing web site & implement the system in the production environment. As the project team progressed trying to understand such a large application, we found it hard as we found the documentation to be at too high a level. This made it hard to find the files that were required for a specific piece of functionality. We also found it hard to change the user interface, as there were functions throughout many of the files that were responsible for building the output to send to the user's browser. Because of these problems the team decided to investigate PHP Fusebox 3.0 as a solution (Available from phpfusebox.sourceforge.net).

2. What is fusebox?

Fusebox is an open source web application framework that has been designed to break an application into manageable units of functionality known as circuits (similar to classes in OO). On disk a circuit appears as a folder with all code files to implement its functions stored either directly in the folder or sub-folders in a large circuit to separate the different types of code file (dsp, qry, act etc). A circuit can contain a number of fuseactions (similar to methods in OO) that will perform the tasks required of the circuit. Each fuseaction is implemented in files called fuses and fuses can be used for more than one fuseaction. There are different types of fuses that handle different functions such as output to the user & interacting with a database etc. Another useful feature we have taken advantage of is the concept of exit fuseactions (XFAs). XFAs allow us to create banks of registered fuseactions that can be made available outside the circuit where they are implemented. For a full description and resources related to fusebox visit www.fusebox.org.

3. Our Experience

Through the process of refactoring the web application to fusebox we managed to remedy some of the difficulties we encountered at the start of the project. The application is now more modular as code files have been separated into separate folders (circuits) for each piece of functionality ensuring all code related to a function is easy to find. It also means the application can be configured to a certain degree by adding or removing circuits as required. Because everything is implemented in fuse files we have created a library of code for reuse in different circuits. Output to the user's browser is now controlled through a single layout fuse that controls the overall layout of a page, and display fuses that control the output from circuits. This allows us to create a new interface for the system without having to change any of the backend code. We have also setup XFA banks to register fuseactions within & outside of a circuit to aid in the future maintenance of the application, if a fuseaction name is changed the update need only be applied to the fuse that sets up the XFA bank, not all files that call the fuseaction.

Charkova, R., Clear, T., Lin,A., & Lomax, T. (2003, 6-9 Jul). Nga Iwi o Ngapuhi Membership System: Relationship Management and Relational Design. NACCQ Conference, Palmerston North.