SE 339 - Assignment 5
Architectural Drivers

11/21/19

Authors:

Stamatios Morellas (*morellas@iastate.edu*)
Richard Smith (*gsmith7@iastate.edu*)

2.2 Questions

1.)

Yes. Since the server logs data from the vehicles into the database in a .txt format every time the service is run, it essentially means that the server is “modular” in the sense that the structure could be modified to support different variations of logging data.

2.)

The short answer is yes. Each data collector in the vehicles could be configured to record errors in addition to the data of the vehicle itself that it is meant to record. It could then send the error information to the server, which could be configured to hold a log of errors sent from the data collectors in the fleet of vehicles.

3.)

No. Based on the description of the fleet management system, there isn’t anything explicit about handling exceptions. However, it could be modified to allow for exception handling.

2.3 Exercise 1

1.)

A plug-in pattern is a form of architecture that calls external code throughout execution without knowing the function of the code in advance.

2.)

With a plug-in architecture, the main problem it addresses is the ability to dynamically add new features to an application, which it allows for. It can also allow for parallel development, so features can be implemented as separate components.

3.)

Scripting can be used to develop higher-level components in the application, first. By doing this, the application is more modifiable and future-proof in later stages of development. The scripting language also allows for the manipulation of lower-level libraries, which frees them from any specific interface.

4.)

As stated above, one way to go about using libraries in a plug-in architecture is to use them in the development of lower-level components, which then can be manipulated by higher-level scripts to work as one.

2.4 Exercise 2

1.)

Eugene Letuchy chose to use HTTP GET requests with iframe's and Javascript, which enables a persistent connection that doesn't return until the server has data from the client. If the request is interrupted or timed out, it conveniently reestablishes itself, which isn't a new way to implement this engineering problem by any means. GET requests are super popular and have been around for awhile now. I dealt with GET and POST requests in many of my own software projects in the past, as have many other software developers.

2.)

The most common way is to use the Model View Controller architecture pattern, which does a good job at separating what is actually going on with the program from the user interface. If there is a bug in the system or something is wrong, it will try to solve it on it's own in the background and not give you a frozen screen. Keeping the Model separate from the view makes for a lot smoother responsive application and makes the user think that your app is nice and isn't a piece of junk. Model View Controller is a very helpful way to look at how to go about all types of application development.

3.)

It solves the miscommunication between different programming languages, let's say you're trying to put together with PHP, Javascript, C++, and Erlang all together into one program like the facebook chat does. Thrift allows you to translate service description into "RPC glue code" allowing you to make calls over different languages. Very awesome and very helpful.

4.)

A better solution than thrift would be just to simply not use so many different languages in one program, stick to 2 or 3 and it should be smooth sailing. Any more than that and you will end up with lots of errors and method calls not being able to go through from language to language.