

Emergent Web Server Architecture – Experiments Replication

May 23rd, 2016

1 Introduction

Welcome and thank you for using our emergent software framework prototype. This prototype was described in the paper “*Losing Control: The Case for Emergent Software Systems using Autonomous Assembly, Perception and Learning*” submitted to **SASO 2016**. This manual contains instructions to execute the framework with the available web server. This document also describes the steps to replicate the results described in the paper.

Please, download and install Dana programming language¹. Feel free to drop us a line in the forum if you have any questions, comments or suggestions. Also, consult Dana’s Programming Guide², for instructions to properly install Dana’s toolchain. This project can be executed in all supported Operating Systems (currently it supports Windows, Ubuntu, Mac OS X and Raspberry Pi).

2 Project Structure

The folder structure of this project follows that of all Dana projects. The *resources* folder contains all of the interfaces for the components. The implementation of the interfaces are provided in the folders outside of *resources*. For example, the interface that defines Composer’s functionality is in *resources/composition*, in *MetaComposer* interface. The components are located in the folders with the same name as the interface’s folders outside *resources*, in this example *composition/MetaCompos – er.dn*.

The framework and web server code are located in different folders. The web server main components are located in *http*, *cache* and *compression* inside of *web_server* folder. The main framework components (composer, monitor and learner) are presented in *composition*, *monitoring* and *learning* inside of *meta_systems* folder.

¹<http://www.projectdana.com>

²<http://www.projectdana.com/dana/guide/doku.php>

3 Execution

The first step to execute the code is to compile. In order to compile the entire project it is necessary to execute the command “`dnc .`” in *meta_system* folder, and execute “`dnc . -sp ../meta_system/`” in *web_server* folder. Please, make sure Dana’s toolchain is properly installed and working.

To properly execute the prototype some additional programs have to be installed. The information generated in the components are stored in a database, so make sure MySQL³ database is installed. Also, to have access to the Perl scripts to generate HTML files for the variation pattern⁴ and enable Dana components access to the database, please, install Perl’s⁵ toolchain.

3.1 Web Server

In order to execute only one configuration of the web server, without monitoring and learning, compile all components in *web_server* folder (as described above) and execute `dana -sp ../meta_system/ WebServer` in a command prompt in the *web_server* folder. To verify if it is running, go to a web browser and connect to ‘`http://localhost:2012/`’. The *index.html* file (located in *htdocs*) should be loaded and presented as result.

By doing this the primary web server configuration will be executed. To change this configuration to any other web server configuration available go to *http* and change the *.manifest* file. This file contains, in a JSON syntax format, the component, in case an interface has multiple implementations, that Dana VM will load. To choose any other configuration, change the name of the component to any other component name present in the *http* folder. For example, if the `HTTPHandlerCGZ` is selected as primary and you want to change it to the component that only implements cache but not compression, change it to `HTTPHandlerC`, and so on.

All available web server’s content is located in *htdocs* in *web_server* folder. Any file dropped in *htdocs* will be found and sent to the client upon request by any executing configuration of the web server.

3.2 Monitoring System

The Monitoring System role is to assemble a configuration of the web server, collect information generated by its monitored components and save them into the database.

In order to use it, compile the entire project (as described above). Then, open two command prompt window. In one of the windows execute the Perl script to enable Dana components access to the database (see Sec. 5 for information on how to setup and execute the database). Make sure to change the *.manifest* file in *meta_system/learning* folder to use

³<http://www.mysql.com/downloads/>

⁴The variation pattern is an artificially created request pattern. We generated 30000 HTML files and for each request the client requests a different file from the 30000 files (more information on that, please refer to the paper).

⁵<https://www.perl.org/get.html>

Explorer instead of MetaLearner, by editing the `.manifest` file and changing the component field. In the other prompt window execute `dana Main ../web_server/WebServer` to start monitoring the system and collecting information. Every 10 seconds, the monitor will collect metrics and events from the executing components and change the web server configuration.

After start system execution in that setting, make sure you run any client request pattern located in `ws_clients` directory from the client machine. Otherwise, the components will not produce any metrics. You can run the client by compiling (`dnc .`) and executing (`dana Client /`) the Client component from `ws_clients` directory. The `ws_clients/Client.dn` is a Dana component that simulates the small text request pattern. This pattern consists of a continuous series of request for the `small_text.html` file, located at `htdocs`. Other types of request patterns will be described in Sec. 4.

3.3 Learning System

The Learning System uses the monitoring system to collect information from the executing components. It then establishes correlation between the recognised pattern and the best performing architecture (a fully description of this component is provided in the paper – including the real-time learning technique and related details).

In order to execute the Learning System, compile the entire project (as previously described). Again, open two prompt windows. In the first one execute the Perl script to enable access to the database (refer to Sec. 5 for further information on how to setup and execute the database). Make sure to edit the `.manifest` file located in the `learning` folder and change the component field from Explorer to MetaLearner (in case you executed the monitoring system in previous section and changed the `.manifest` file configuration). In the second window execute `dana Main ../web_server\WebServer` in the command prompt inside `meta_system` folder.

Also, similarly to the monitoring system, it is important to execute a client request pattern from a client machine to watch the learning system in action; for example by executing: `dana Client /` from `ws_clients` folder.

4 Tests Replication

The evaluation was conducted using two machines. The configuration of these machines are described below.

“ The server was a rackmount server system with an Intel Xeon E3-1280 v2 Quad Core 3.60 GHz CPU, 16 GB of RAM, running Ubuntu 14.04. The client was a desktop machine with an i7-4770 3.40 GHz CPU, 6 GB of RAM, running Ubuntu 14.04. The server and client reside on two different subnets of our campus network, in different buildings.”

The paper presents two results, the first one referenced as ‘Divergent Optimality’, and the second, ‘Online Learning’. The steps to replicate the findings described in the paper is presented in the subsequent sub-sections.

4.1 Divergent Optimality

The Architecture Optimality Divergence presented three graphs demonstrating different architectures performing differently when subjected to varied request patterns – in the paper, this is illustrated by the graphs in Fig. 4, Fig. 5 and Fig. 6 in the paper.

The graphs were generated by executing the `dana Main ../web_server/WebServer` component and collecting the information directly from the database.

The request patterns evaluated were small TEXT files, small IMAGE files and multiple small HTML files in case of variation pattern. All requests are made from the client machine running a client component located in `ws_client` folder in the project.

For each client component used to generate request patterns, make sure it issues requests to the right server machine IP address. You can change the IP address in the `Client` component by opening its source code (`.dn`) in any text editor and changing the IP address in the first line of the code, for each client component.

To run the tests follow the steps below.

1. Change `.manifest` file in `learning`, replace the value in ‘component’ field to Collector;
2. Execute the Main component `dana Main ../web_server/WebServer`.
3. Execute the client in a prompt with the command `dana Client \small_text.html`, from the `ws_clients` folder on the client machine.

The `Collector.dn` component will run each available architecture one hundred times with 10 seconds exposure. At the end of the execution, the database will hold all monitoring information in the monitoring table. The graphs were produced based on the result returned from the execution of the SQL query: “SELECT * FROM monitoring;”.

The instructions above describes the steps to run the small text file request pattern. To run the small image file pattern change `dana Client \small_text.html` to `dana Client \small_image.jpg`. There is also the NASA trace, which can be reproduced by executing `dana NasaClientRequest`.

4.1.1 Variation Pattern

In order to generate the requests for the variation pattern, you will need to generate 30000 variations of the small text in `htocs` folder. To generate the files simply execute the Perl script `script_to_create_variations.pl` using the command `perl script_to_create_variations.pl`.

To issue the client requests, instead of using `Client` component, uses `VariationClient` component with the command `dana VariationClient`. At the end of the execution, run the query “SELECT * FROM monitoring;” for the results.

Note that if you want to delete all of the 30000 files, run the script `script_to_delete_variation.pl` with the command `perl script_to_delete_variation.pl` located in `htdocs`.

4.2 Online learning

The online learning result shows the benefits of using our framework to find the best configuration every time it recognises a new pattern. The results described in the paper were obtained by following the steps (result such as in Fig. 7 in the paper):

1. Change `.manifest` file in *learning*, replace the value in ‘component’ field to `MetaLearner`;
2. Execute the Main component `dana Main ../web_server/WebServer`.
3. Execute the client in a prompt with the command `dana NasaClientRequest`, from *ws_clients* on the client machine.

Note that it is possible to limit the components that will be available for experimentations. In order to do so, copy *.ignore.example* file and rename it to *.ignore*, this file contains a list of components that will be ignored by the composer (so, please, only use the *.ignore* file when trying to limit the number of architectures that will be exposed to the requests, otherwise, there is no need to create the *.ignore* file).

5 Database Configuration

To configure the database you have to create a database named `webservice` in a MySQL database on the server machine. After that, create the database schema by executing the file named `webservice_schema.sql`, located at `database_external/database_schema` folder. Remember to set the user name and password to access the database in the `database.config` file located in `database_external` directory. To enable Dana components access to the database, open a new command prompt and execute the Perl script `database_broker.pl`, also in `database_external` folder.

To execute the script run the command `perl database_broker.pl`. It is imperative that `database_broker.pl` script remains running while all tests are running.

6 Final Considerations

Thank you for your interest in our research. We hope you find this manual helpful to get you started with our current version of the emergent software framework. All questions, comments and feedback is greatly appreciated. Contact us on Project Dana’s forum (<http://www.projectdana.com/fora>) or over email (r.rodriquesfilho@lancaster.ac.uk).