ARA FSP Collaborative Intelligence Software README

POC: Chris Argenta | cargenta@ara.com | 919.582.3443

Overview

The CI contribution to FSP includes demonstrator software that implements a Partial FSP Run-Time implementation that includes: a feature/entity typing system for missioning the system [see com.ara.fsp.runtime.DemoRunTime.java for example code for building out features and entities]; FSP state space based on ARA's design, and web-based; and web-based user interfaces for FSP Explorer, a public-oriented crowd-sourcing site, and a crowd-sourcing management interface. We have also implemented a crowd-sourcing projector (already built into DemoRunTime) and a "random" projector for demonstration purposes (as a separate stand-alone executable in com.ara.fsp.runtime.DemoRandomProjector). Our example features and state space include examples of aggregation at the feature level – including our innovative modality detection based crowd-projection of conditioning events, and state space level. The overall architecture is shown in **Error! Reference source not found.**.

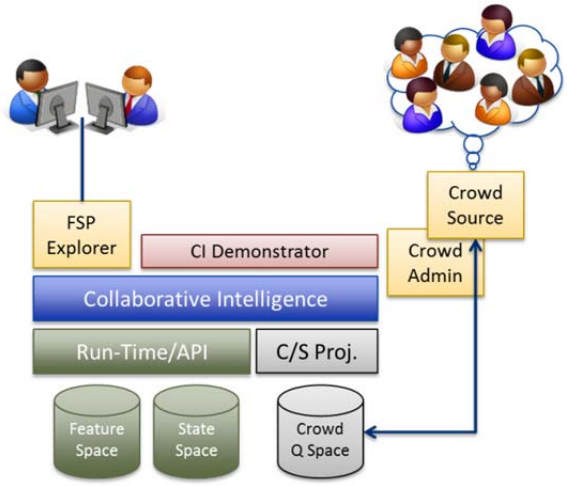

Figure 1. Overview showing logical architecture

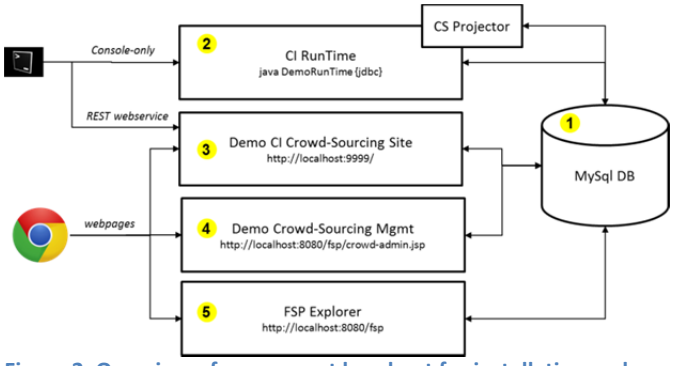

Figure 2. Overview of component breakout for installation and demonstration purposes

Directory Structure

Name	Description
README.pdf	This document, click HERE to go to it.
/api	The Java interfaces used to communicate with the runtime components and state-space. There should be nothing to set here.
/crowdsourcing	This is the software that implements the public crowd-sourcing site and webservices for allowing Node-Red to pull question results. The key configuration file is ".env" – be sure to set the user and pass for your database. You will need to use the following PHP commands to setup the login tables for in your database: cd crowdsourcing php artisan migrate php artisan db:seed These will create the tables and default accounts (user:fsp@ara.com pass:ara123) needed for the crowd-sourcing public site. If you want create others you can edit /crowdsourcing/database/seeds/UserTableSeeder.php. If you have an email server connected you should be able to register new users too.
/database	This file contains the .sql files to setup the db schema for database "fsp" and clear out tables (to reset db as needed). The defaults in the code are all configured for user:root with pass:dogstar.
/explorer	Java Servlet and JavaScript for deployment in Tomcat. A copy of fsp.war is in the top-level directory for reuse. If you want to recompile and deploy it, use the following sequence: cd explorer/support vi build.properties // edit to your setup, particular tomcat ant deployWeb If everything goes well this should be ready to go.
/runtime	Java code for RunTime. A .project file for Eclipse is included. The key executable classes are in the com.ara.fsp.runtime package and include: • **DemoRunTime.java** – which accepts a JDBC connect string and starts up the state-space and run-time in a manner suitable for demonstration. The default mission is setup for the energy sector and it will create a new mission (2010-2016) if one does not exist (you need a mission to do anything interesting). This implementation of the run-time automatically expands out the future space 1 year whever all the questions are answered (until it reaches the mission horizon). • **DemoRandomProjector.java** – this is a class for automatically filling in the answers to all open feature questions with random values. This is helpful when the state-space gets large.

Installation Information

The following sections are broken out to describe the configuration of various components of the system. The break out is shown in Figure 2.

1. MySQL Database

This demonstrator is built on a MySql Database (version 5.6.24, but anything recent should work fine, all development was done using a version packaged with XAMPP) which is started as a service on port 3306 (configurable in code/startup).

2. Demonstrator CI Run-Time

The Run-Time is a java application that runs on the console and logs to stdout. The only parameter is a JDBC connection string (see main function for example). In general, we executed this in the debugger so that we could put break points to control the flow for interactive demonstration. The delivered version is set up to be more robust that the live demo and can run without the debugger fine. The run-time will automatically use the crowd-source projector for all features and conditioning. You can run the DemoRandomProjector.java file to automatically scan the open feature questions and fill them in with random values.

3. Demo CI Crowd-Sourcing Site

This is a PHP-based website built on the Laval framework that provides the following capabilities:

A. Login and Registration – if the email server is configured people can register for an account and answer questions. The following accounts are setup in the Artisan configuration:

User Id	Password
demo2015@ara.com	demo2015
cargenta@ara.com	ara123
fsp@ara.com	ara123

B. List of open questions (left side). You will need to manually refresh the screen, and unfortunately, it is not well suited to the large number of questions that get generated in the demo run-time.

C. Question forms for both Feature type and Conditioning type crowd-sourcing questions. When submitting answers there are a few things to note:
    a. There is not input checking for the feature types in the demonstrator
    b. Each response counts as a single input – the demonstrator does not differentiate input by user (so the demo user can answer as much as they want without their responses replacing each other). Feature questions require >20 response before the Crowd-Sourcing Projector will integrate the responses. Conditioning explanations are aggregated by voting, the top answer much have at least 3 votes before it is recognized.

3. Demo CI Crowd-Sourcing Mgmt Site

This site is hosted along with the FSP Explorer and run via Tomcat. It's primary purpose is to allow a demonstrator to generate a set of random crowd responses to a specific question. It can be accessed via: http://localhost:8080/fsp/crowd-admin.jsp in the default setup.

3. Demo FSP Explorer

The FSP Explorer is a demonstrator visualization of the FSP state space that allows some interaction/editing. It is run via Tomcat and can be accessed via: http://localhost:8080/fsp/ in the default setup. Note that the initial screen provides for the selection of a Mission. The demonstrator run-time is configured to default to the more last mission in the database – so to avoid conflicts, it is recommended that each install focus on one mission until some aspect of mission management is implemented in the CI run-time. Notice that the crowd-sourcing site and projector is Mission-Independent. Note, this is just a proof-of-concept, so some features are only partially implemented for the demonstrator.