Software Design Document for Visualization Tool for Composition of Cloud Computing Services (VTCCS)  

Preliminary Version

Approved by:

Prepared by: Zolangi Ramirez  
Hongsuk Choi  
Sudip Baral  
Gonzalo Serrano  
Johnson Truong

Sponsored by: NASA DIRECT-STEM/Faculty

22 November 2017
## Revision History

Rows are updated when the document is revised

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update File</td>
<td>12/4/2017</td>
<td>Add references</td>
<td>1.0</td>
</tr>
<tr>
<td>Edit Section 6</td>
<td>12/5/2017</td>
<td>Add Components</td>
<td>1.0</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose
The purpose of this document is to outline in detail the software architecture and define the design for the Visualization Tool for Composition of Cloud Computing Services (VTCCCS). This document will provide several views of the VTCCCS’s design in order to facilitate understanding of the tool.

This document will show how the design of the VTCCCS will accomplish all the software requirements specified in the Software Requirements Specification (SRS) document.

1.2 Document Conventions
The organization of this document begins by describing design considerations made, followed by the architectural strategies used to design the web application. Then the document will mention the policies and tactics used for the VTCCCS. After the document will describe the detailed system design. Once the document describes the system, it will show the User Interface. Then the document will show the Database design and finally the document will show the Requirements Validation and Verification part.

1.3 Intended Audience and Reading Suggestions
This document is intended for project managers and project members working on the VTCCCS. If the reader is only looking for a brief overview of the project, they should focus on reading section 1 (Introduction) of this document.

If the reader is looking for a detailed description of the project, they should focus on sections 4, 5, and 6. These sections provide an in depth description of the project and policies used.

If the reader is looking for requirements that have been verified and validated, they should refer to section 10 of this document.

If the reader is looking for any definitions and/or abbreviation meanings, they should refer to the Glossary in section 11.

If the reader wants more information regarding this document, they should refer to section 12 of this document.

1.4 System Overview
The VTCCCS will have two types of web applications. One application will be used for users to create their own web services and be able to visualize the architecture. The other application will be for the administrator to be able to manage users who have created accounts.
2. Design Considerations

2.1 Assumptions and Dependencies

This project connected some of different tools for:

- Web Application
- Connect Angular Node.js and AngularCLI
- Related Java with Apache Tomcat or MySQL
- Later connect D3.

2.2 General Constraints

This Web application is based on Angular and it has to connect with Node.js and AngularCLI. It also uses Java and MySQL for the back-end, for the user to be able to call data.

Such constraints may be imposed by any of the following:

- Angular or Node.js environment.
- Java or MySQL environment.
- Availability or volatility of resources
- Standards compliance
- Data repository and distribution requirements
- MySQL memory and other capacity limitations
- Network communications
- Communications Angular with Java.

2.3 Goals and Guidelines

For the VTCCCS, the goal is to have all of the tools be connected with each other and be able to call suitable data.

- Visualization on the web
  - Network communication is important.
  - Angular communication with network.
- Connect the web application with MySQL
- Save information from user input

2.4 Development Methods

Using Angular we connected more than 30 different Components and each component has their own sub-component. Also each Component has their own work, like the ‘demo-app’ Component shows result of what the user did on the web. Also some Components it is possible to make them connect with MySQL. It saves user input and uses queries to the database and shows the information from the data returned. The problem here is that D3 has yet to be used to show this visualization of the interaction between the visuals and the data returned from the database.
3. Architectural Strategies
The VTCCCS web application is combination of web services. The users can use this tool to visualize the interaction between services. Each tools is connected with each other for visualization and each show a different result.

- Tools used in this project
  - Angular
  - MySQL
  - JAVA

- Save Data
  - user input
  - user information

- Error detection and recovery
  - check error and shows solution guideline.
  - Communication mechanisms

- later we will use and finish D3 part.
- Show external databases to user and data storage persistence
- Distributed data or control over a network
- Guideline for generalized approaches to control
- Concurrency and synchronization
  - user control to different tools on same time.
4. System Architecture

Cloud computing architecture is composed of front-end and back-end platforms. Both components interact with each other to output value product. Visualizing the service is one of the key roles our software does. The front-end for the VTCCCS is responsible for enabling the user to handle the nodes which are the composition of single or multiple services. Managing users and filtering inputs are other functions front-end is devoted to. On the other end of the spectrum, back-end manages the storage, by running the server and maintaining the session.

The VTCCCS web application is responsible for managing web services. The entry point for the service is the main app component. On the basis of the functionality and nature of job a branch is created which we call component here. Some important subdivided components can be listed as: demo-app, projects, main-nav, etc. The application being built is equipped with the tool which allows us to visually manage services and control parameters.
The Projects menu lets us create a new project. Every new project is assigned a unique id with some default name. Composition of the service is injected by the user. Project can be edited through graphically or through menu options. Debug component allows us to see the nodes in JSON format. It includes information of the node along with its neighbors and child nodes. Demo menu has options to get all the users in the database and delete or modify user if it is needed. This component also lets us register a new user by an administrator.
5. Policies and Tactics

For accomplishing what we have planned, we are using Angular. It is a platform which allows us to build a web component. It is based of of JavaScript and TypeScript in a super level.

5.1 Choice of which specific products used

The IDE being used to develop this software is Intellij IDEA Ultimate version. It is capable to handling both front-end and back-end.

5.2 Plans for ensuring requirements traceability

The requirement laid out by the client is the basis of our milestone.

5.3 Plans for testing the software

Initial step for testing is done by the mock-up data. In near future, when the software is completely developed will be deployed in the server.

5.# Engineering trade-offs
    …Describe…

5.# Coding guidelines and conventions
    …Describe…

5.# The protocol of one or more subsystems, modules, or subroutines
    …Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality
    …Describe…

5.# Plans for maintaining the software
    …Describe…

5.# Interfaces for end-users, software, hardware, and communications
    …Describe…

5.# Hierarchical organization of the source code into its physical components (files and directories).
    …Describe…

5.# How to build and/or generate the system's deliverables (how to compile, link, load, etc.)
    …Describe…

5.# Describe tactics such as abstracting out a generic DatabaseInterface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the DatabaseInterface class.
For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.
6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD’s will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.1 AppComponent - Module

6.1.1 Responsibilities

The primary responsibility of the AppComponent is to display all components in the web application in one component. The AppComponent’s role acts as the main component of the application.

The implementation of the shell AppComponent is distributed over three files:

1. app.component.ts
2. app.component.html
3. app.component.css

6.1.2 Constraints

The component class code is written in TypeScript in the file named ‘app.component.ts’
The component template is written in HTML in the file named ‘app.component.html’
The component’s private styles are written in CSS in the file named ‘app.component.css’

6.1.3 Composition

No subcomponents are needed for the AppComponent

6.1.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass’s, and metaclasses.

6.1.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers,
databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.x.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

6.2 Home Component - Module

6.2.1 Responsibilities

The primary responsibility of the HomeComponent displays the home page of the web application. The HomeComponent’s role acts as the main component of the application. The implementation of the shell HomeComponent is distributed over three files:

1. home.component.ts
2. home.component.html
3. home.component.css

6.2.2 Constraints

The component class code is written in TypeScript in the file named ‘home.component.ts’

The component template is written in HTML in the file named ‘home.component.html’

The component’s private styles are written in CSS in the file named ‘home.component.css’

6.2.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.2.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented
designs should include a description of any known or anticipated subclasses, superclass’s, and metaclasses.

6.2.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.2.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

6.x Template - Module

6.x.1 Responsibilities

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.x.2 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.x.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.x.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)?
concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass’s, and metaclasses.

6.x.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.x.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.
7. Detailed Lower level Component Design

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”). Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.x Name of Class or File

7.x.1 Classification
   The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)
   A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy
   Class inheritance: parent or child classes.

7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation
8. Database Design

Include details about any databases used by the software. Include tables and descriptions.
9. User Interface
This software makes use of a few form components.

9.1 Overview of User Interface
Once the user clicks the link login/signup, he/she will be directed to the Login Form as shown above. This modal will allow the user to input email, password, first name and last name if the user is registering as a new user or can sign in if account has already been created.

9.2 Screen Frameworks or Images
Here are some screenshots from the web service application.
# DIRECT STEM

## Web Service Composition

Build, Import, Export, Test, View, Analyse

[Start Here]

<table>
<thead>
<tr>
<th>Composition</th>
<th>Service</th>
<th>Select</th>
<th>Configure Composition</th>
<th>Configure Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>382448535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>NODE_382448535</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ADD PARAMETER**
9.3 User Interface Flow Model

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.
10. Requirements Validation and Verification

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Module that Satisfies Requirement</th>
<th>Method for Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the requirement has been met.
11. Glossary

- VTTCS - Visualization Tool for Composition of Cloud Computing Services
- SRS - Software Requirements Specification
- HTML - Hypertext Markup Language
- CSS - Cascading Style Sheets
- JSON - JavaScript Object Notation
- MySQL - My Structured Query Language

12. References

https://csns.calstatela.edu/department/cs/project/view?id=6059778

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc