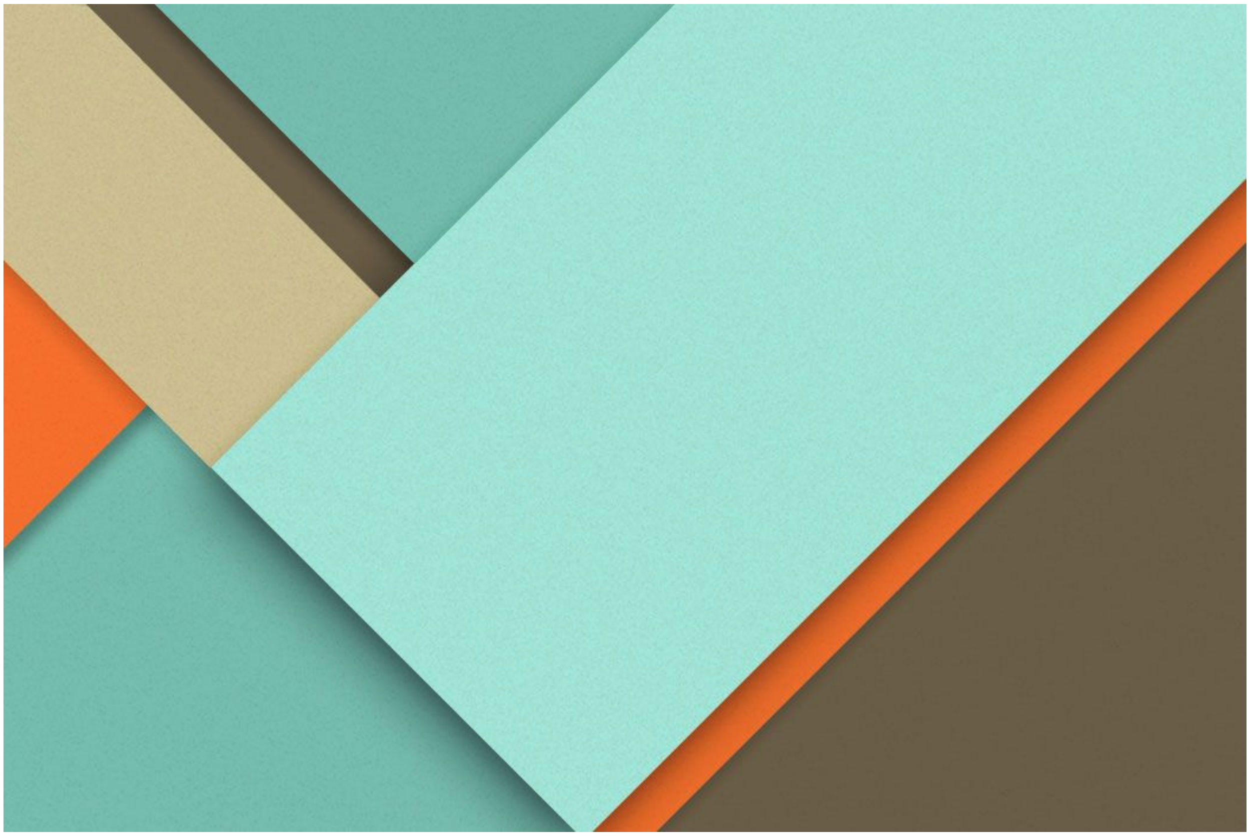

SupplyChain2Map

May 2016

Mirco Sprung

Omid Najaei Nezhad

Mohammad Ali Ghasemi

Sina Mahmoodi

Introduction

Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the functionalities of the SupplyChain2Map system. This document will cover each of the system's intended features, as well as offer a preliminary glimpse of the software application's User Interface (UI). The document will also cover hardware, software, and various other technical dependencies.

Scope

Enterprises often have to deal with the management of their supply chain, to coordinate the flow of goods and services, from the producer to the consumer. Albeit, if done right, this task can help integrate different enterprises together better and improve efficiency, it is usually done manually, which is a complex and costly task. SupplyChain2Map, as a part of SCIApp, tries to facilitate the automation of this task by visualizing the supply chain of enterprises, embedded with some analytical data about the flow, such as the delivery rate of routes.

Definitions, Acronyms and Abbreviations

- **SC2M** SupplyChain2Map, which is being described in this document
- **SCOR** The Supply Chain Operations Reference model (SCOR) is the world's leading supply chain framework, linking business processes, performance metrics, practices and people skills into a unified structure.
- **SCORVoc** An OWL vocabulary which fully formalizes the latest SCOR standard, while overcoming identified limitations of existing formalizations.
- **W3C** The World Wide Web Consortium is the main international standards organization for the World Wide Web.
- **HTTP** The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.
- **SPARQL** The SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF.
- **RDF** The Resource Description Framework (RDF) is a general-purpose language for representing information in the Web.

- **ES6 (ES2015)** ECMAScript (or ES) is a trademarked scripting-language specification standardized by Ecma International, and ES6 is the 6th version of this specification. One of the most popular implementations of this standard is Javascript.
- **Babel** Babel is a compiler which has support for the latest version of JavaScript through syntax transformers. This comes into play as not all the features of ES6 are implemented in current browser versions. Babel fills that gap.

References

1. http://www.apics.org/sites/apics-supply-chain-council/frameworks/scor
2. https://github.com/vocol/scor
3. https://www.w3.org/TR/REC-rdf-syntax/
4. https://www.w3.org/TR/sparql11-protocol/
5. https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Overall Description

Product Perspective

At the core of SupplyChain2Map a Javascript library resides which fetches necessary data from databases (triplestores) and after doing some transformations shows it on a map for the user to see.

In this sense, the project is divided into a client-side and a server-side. The server-side keeps the supply chain data for enterprises adhering to a standard ontology and exposes it to the client. The ownership of the server might be in the hands of the enterprise itself. In turn, the client fetches the necessary parts of data, caches it in the browser, and executes queries on it upon need.

Specific Requirements

External Interface Requirements

Hardware Interfaces

SC2M is intended as a JS library, therefore the hardware which the client can run on is not very limited, as various hardware devices support a display screen and can be made to run a browser and therefore use the library. Examples are PCs and handheld devices. The data fetching part requires a network card connected to the the internet.

The server-side requires a server, preferably geared towards being a data store with features such as high speed disks and large amounts of memory. The server requires a network card connected to the internet.

Communication Interfaces

Most of the communication happens between the client and the server(s), through client requesting enterprise data from the data stores. The two communication protocols which are used, are HTTP and SPARQL, which itself uses HTTP. HTTP works on top of the TCP/IP stack.

The data transferred using these protocols are either SPARQL queries and results as defined in the SPARQL Protocol Document by W3C, or JSON documents for every other request.

Software Interfaces

The JS library depends on the API provided by the browsers (or Babel) for normal functions such as accessing the user's screen size or accessing the LocalStorage to cache RDF triples. For visualizing the supply chain of an enterprise, SC2M depends on the interface provided by Leaflet, which in turn is a JS library. Furthermore, the SPARQL Protocol and Query language are used to access and modify the data.

Ontology

The SCOR framework provides many different definitions for expressing enterprise data, including various processes which might occur in a supply chain. Since the data SC2M will use is in form of RDF triples, it depends upon SCORVoc to shape the SCOR framework in terms of concept hierarchies in RDF.

Functional Requirements

ID	Requirement Statement	Must/Want	Comments
FR001	The user shall import data	Must	The user should be able to either import a set of data, or choose an enterprise from the existing data set.
FR002	The website shall have a world map.	Must	

FR003	The map shall be interactable.	Must	Providing general map interaction like zooming or dragging.
FR004	SC data shall be visualized on the map.	Must	
FR005	The nodes on the map shall be selectable.	Must	Selection in order to get additional information about the node.
FR006	The website shall provide analysis functionality over the SC data.	Must	Given already existing algorithms and metrics.

Non-functional Requirements

ID	Requirement Statement	Must/want	Comments
NFR001	Modularity	Must	Different segments of the the product should be architected in a way to interact with each other in a standard and independent fashion.
NFR002	Maintainability	Want	The library is to be architected in a developer-friendly approach in mind. Is crucial for the library to be easily extendable by everyone, not only the initial developers group..
NFR003	Performance	Want	The product is to be reasonably operating on the inputs, regardless of size and complexity.
NFR004	Usability	Want	Due to the focus importance of graphical interface in this project, the app should be as easy and intuitive as possible for the user to interact and learn
NFR005	Accessibility	Must	The product should be possible to run on the latest version of modern

			browsers (i.e. Google Chrome, Firefox, Edge)
NFR006	Security	Want	Enterprises might want to own their own data, which brings about the question of authentication and secure communication to prevent enterprise data leak.
NFR007	Testability	Want	The functionalities within the source-code should be written in a way that is comprehensively verifiable by an automatic testing suit with a high code test coverage.
NFR008	Documentation Requirement	Must	The documentation for each of the elements of the library should be written and compiled in an easy to use manner. The remarks should be clear and well-organized. The intention of the documentation should be to enable third-party developers easily understand the way to work with the library by reading the documentation.
NFR009	Acceptance requirements	Must	The library should display the location of the supply chain elements accurately and clearly.

Design Constraints

Memory Usage

The final product being a Javascript library which runs on the browser, there are limitations on the amount of memory which could be consumed by the library. These limitations may be due to browser policies, or performance. Since web applications which consume a large chunk of memory tend to perform slower. Besides, since there's no installation of web application, the library can't determine the total or free amount of memory on the user's computer. Analysing the available web applications led to an estimated cap of 200 MB.

Disk Usage

For the aforementioned reasons, and the fact that web applications do not have direct access to the client file system and can only use disk through the LocalStorage API provided by the browser, put a restrictive limit upon disk usage. Disk I/O is usually problematic because it's non-deterministic for a multitude of reasons. A simple disk operation can take anywhere from zero milliseconds to a few seconds. Browsers cope with this by preloading the entire Local Storage key/value store into memory on first request, which makes the limitation for applications even more sensitive. Therefore, the library must use lazy loading methods to fetch the data, only when it is needed, and discarding it directly afterwards to save disk and memory.

Software Licensing

The project depends upon a variety of open source softwares which are under different licenses. As an example, the GNU GPLv3 license requires anyone who distributes a code or a derivative work to make the source available under the same terms. Therefore, the license of the final library, must adhere to and be compatible with the licenses of the tools and libraries being used.

Immutability

The aim is to reduce the amount of mutability in the source code. This means that by declaring variables in a way which must be. For instance declaring constants, public and private variables. We should also focus on conventions in coders community which make our software capable to be interested by aforementioned community.