Development of a Horticultural Soil Management Information System (SMIS)

SMIS: System architecture

V1.0

Revision History

Version	Description	Date	Author
1.0	Main functionalities and initial requirements	24/03/2017	Fady Mohareb

Development of a Horticultural Soil Management Information System (SMIS)

SMIS: System architecture

1. Executive summary

Soil degradation affects yield and yield quality directly, as does the timing of tillage, planting and harvesting operations. Poor soil quality (*e.g.* compacted soil) leads to gaps in production continuity and critically to pinch points in product delivery. Such continuity gaps can exert significant financial impact on growers and increase the reliance on imports to meet customer requirements and to maintain national food security. Furthermore, uneven labour requirements and increased food prices have direct effects on society. A key industry-wide challenge is therefore to identify relevant metrics of 'soil' both in crop-specific and rotational contexts, and to use these both in mitigating soil degradation and in driving sustainable soil management.

The Horticultural Soil Management Information System (SMIS) Project was established in response to the soil degradation challenges mentioned above. The main objective is to develop a central data repository for growers' datasets, allowing software users to store, manipulate and represent a wide range of available sources of data pertaining to the specific effects of soil management practices on horticultural crop productivity and environmental protection. Ultimately, the outcome of SMIS will be an e-Guide toolkit able to provide AHDB users with a set of robust, empirically-based, best-practice guidelines, and the likely consequences of applying them on crop production and environmental protection.

The purpose of this document is to give an overall view of the SMIS software architecture, the rationale behind choosing the outlined programming environments, as well as presenting our vision for the main functionalities of the software framework

2. System architecture

SMIS as a Web-based interactive application

The first decision to be made in terms of software design architecture is whether to design SMIS as a desktop standalone or a Web-based application. While a standalone application provides the client with full-control over the computational resources needed to perform specific tasks, the advantages of a Web-based application clearly outweighs these of standalone applications, some of these advantages are represented below:

1. **Applications with database backend:** Applications where a database back-end is connected to the main framework UI are usually developed as web applications, as this substantially reduces the effort needed to maintain and update the data repository.
2. **Maintenance:** Future maintenance of web based application is more efficient, since the version upgrade is needed to performed only once (on the server side) for the upgrade to be reflected for all clients accessing the system. Furthermore, desktop applications are limited by the physical location and hence oppose a usability constraint. Web applications on the other hand makes it convenient for the users to access the application through access to a secure network (Intranet, VPN etc.).
3. **Visualisation:** Since data visualisation represents a main functionality in the SMIS system, the plethora of web based visualisation libraries, such as D3.js, Java Server Faces and HTML5 makes the decision to implement SMIS as a web framework an attractive choice. The Web currently is the most common method of information exchange. Dynamic and interactive web applications have become more popular as the Web has evolved. New technologies like HTML5 enable software developers to port desktop applications to the Web, which provides advantages such as instant availability with no installation requirement. HTML5 and other related Web standards adopted recently consist of many types of technologies. Some of them are purely graphics and display related (Canvas, CSS3, SVG, WebGL), some bring new semantic support (new semantic tags, old display tags deprecated), some bring new computational capabilities (Web Workers) and some are data integration related (CORS, FileReader).

SMIS overall architecture

The overall architecture of the SMIS system is outlined in **Error! Reference source not found.Error! Reference source not found.**. Briefly, the software will include three main modules as follows:

a. Storage layer:

This layer will be developed based on a MongoDB framework. MongoDB is a flexible NoSQL database. NoSQL stands for Not Only SQL (Structured Query Language) and stores data according to different types of models: wide column store, document store, key value store and graph models. Their main advantage is that they are not based on relational models like with a SQL database. The main differences are the semi structured data against the predefined structured data in SQL, the great performance

Figure 1The SMIS overall architecture.

with large datasets and the integrity of the database managed in the programming part instead of the SQL database system. The database will include three main storage modules: ***i) Fields data, ii) evidence data, and iii) experimental data.*** Details about each module will be provided later.

b. Application Layer:

The application layer will hold all frameworks required to query, visualise and analyse the data from within the storage layer. Node.js was chosen as the IDE of choice for providing the application layer. Node.js is a runtime environment for JavaScript built on top of Google's V8 engine, providing a context for various interactive display and visualisation JavaScript libraries. The V8 engine brings in an efficient model here, where the JavaScript code will be compiled into machine-level code and the executions will happen on the compiled code instead of interpreting the JavaScript.

Node.js will provide the main foundation for the application layer, supporting the following environments:

1. **Express.js:** Express is Model View Controller (MVC) based web application framework. MVC is just an architectural design pattern:
    a. **Models** are used to represent the data or entities of the web application. They lie more close to the instances, which store the data of the application, typically a database or a web service.
    b. **Views** are responsible for how the application gets presented to the end user. So, a view can be considered as the presentation layer of the application.
    c. **Controllers** are basically the middleware between models and their respective views and to take care of the request from the user for a particular web page in our application end to end.

2. **Mongoose:** Mongoose is a wrapper for the MongoDB database, allowing mapping the table schema to the Node.js IDE, and facilitating the interaction with the storage layer.

3. **R:** R is an open source programming language and software environment for statistical computing and is the most commonly used language for machine learning and data science within our group. R (together with Python) will be used to develop the decision-support system and prediction models developed based on the experimental data collected within this project.

4. **LaTeX:** LaTeX is a document preparation system, and will be used as a framework for generating PDF reports based on the analysis performed on the growers data.

c. Presentation Layer:

D3.js: D3 is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards. D3 will be the library of choice for creating interactive plots based on user queries.

AngularJS and Twitter Bootstrap are the most famous JavaScript frameworks and AngularJS is also involved in the MEAN stack architecture. All of these aforementioned frameworks were extended with multiple open source libraries to quickly embed functionalities. **Cytoscape** will be used to visualise the interaction and relationships between the "Cause" and "Problem", and the "Problem" and "Solution" in a node-edge model based on graph theory. Cytoscape is an open source software platform for visualizing complex networks and integrating these with any type of attribute data. Cytoscape was originally developed to visualise molecular and genetic interaction, however its application is nowadays extended to include almost any type of interaction withdrawn from a complex data structure such as social and data mining network. Cytoscape.js is the javascript version of the standalone Java application, and can be used as network data visualization engine for a web application. It is a pure JavaScript library and no plugin is required for the web browsers.

2.1 Data repository

As stated above, MongoDB was chosen as the framework for choice for implementing the SMIS database backend. The database will include three main storage modules:

2.1.1. Fields data module:

The collection schema for the fields data storage module is outlined in Figure 2. The document oriented type of MongoDB has semi structured data and each element is considered as a document with a unique identifier (ID) which matches the needs of SMIS. The database stores data coming from multiple resources (Gatekeeps data will be used as model input dataset for designing the framework); imported as CSV files. The Growers collection has a one-to-many relationship (**1-N**) with the fields collections, and each field can have one or many datasets (**1-N** - Field's year). The individual gatekeeper dataset will be stored in a Variable-Value relationship file.

Thanks to MongoDB, in a collection, all the documents have a JSON (JavaScript Object Notation) structure and are easily manipulated by the client and the server. For both sides, JavaScript was chosen for multiple reasons. First of all, employing JavaScript on the client and server side avoids code redundancy and allows focusing

and mastering only one language. On the client side, it is executed by users' web browsers instead of waiting for web server responses. On the server side, JavaScript engines are present to optimise its execution.

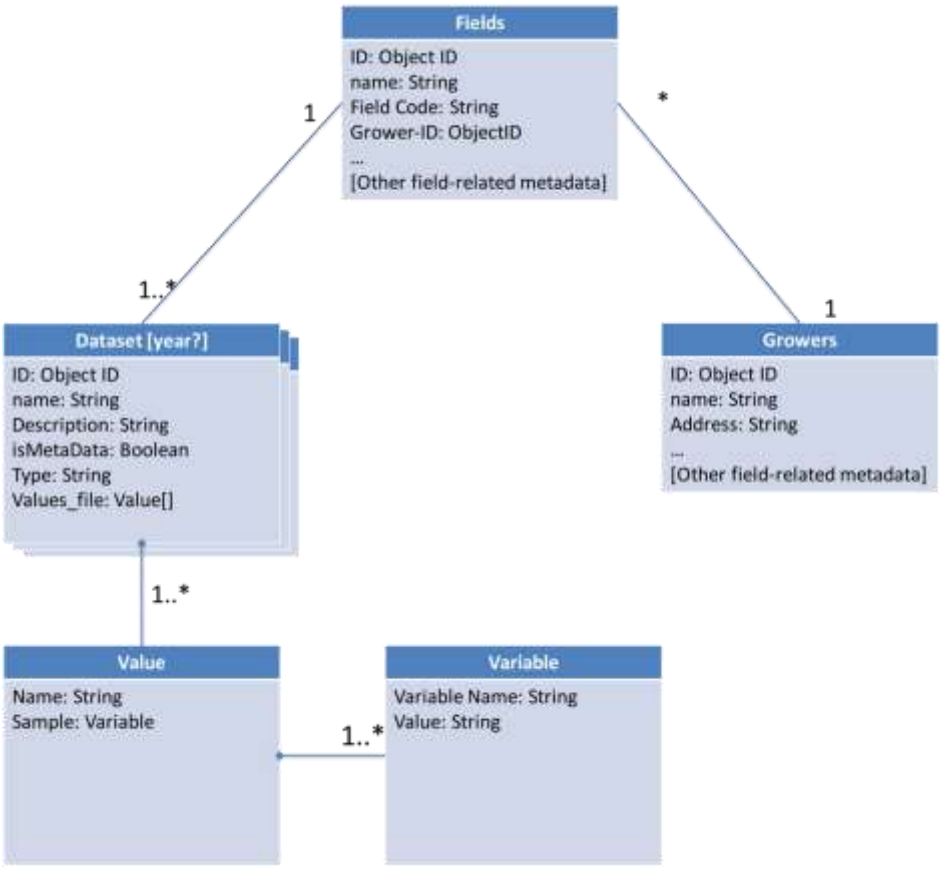

Figure 2. The fields and experimental evidence storage module, which will be the core storage module for SMIS.

2.1.2 Evidence data

Evidence data represents a straight forward storage module, as it will be stored as a singular collection (single table) which will include the established relationship between cause and problem from one side, and problem and solution from the other as a one-to-one relationship. These relationships will be withdrawn from the literature review study conducted during the first year of this project. This one-to-one relationship is required in order to visualise the interaction network for the visualisation module as described later on. A sample table describing the **"cause"-"problem"** relationship is presented below in Table 1.

Problem	Cause	Evidence	Weight
Compaction	Working wet soil	Literature	0.4
Compaction	Excess traffic	Literature	0.4
Drainage	High water table	experimental	1.2
Drainage	Poor soil structure	experimental	1.2
Soil life	Low organic matter	experimental	3
Soil life	Low residues	experimental	2
Soil life	Excess pesticides or fertilizers	Emperical	2
Soil life	Excess tillage	Emperical	2
Soil life	Poor aeration	Emperical	2
Salinity	Saline seeps	Emperical	2
Salinity	Saline irrigation water/well	Emperical	2
Salinity	Shallow water table	Emperical	2
Salinity	Poor drainage	Emperical	2
Salinity	Excess evaporation	Literature	2
Erosion	Lack of cover and residue	Literature	0.4
Erosion	Low organic matter	Literature	0.4
Erosion	Poor aggregation	Literature	0.4
Erosion	Tillage pan or compacted layer	Literature	0.4
Erosion	Tillage practices that move soil down slope	Literature	0.4
Erosion	Excessive tillage	Literature	0.4
Erosion	Intensive crop rotation	Literature	0.4
Infiltration	Lack of cover and residue	Literature	0.4
Organic matter/residue	Low residue crops	Experimental	0.4
Organic matter/residue	Too much fallow	Experimental	0.4
Organic matter/residue	Insufficient additions of crop residue	Experimental	0.4
Soil pH	Use of ammonium fertilizers	Experimental	0.4
Soil pH	No liming	Experimental	0.4

Table 1 Evidence data table (For illustration purposes only).

2.1.3 Experimental data

Experimental data will be stored in the same storage module as the fields data, thanks to the MongoDB semi-structured design. Soil conditions and/or impact of the field practices will be stored as an additional dataset within the datasets collection table as metadata. The Boolean field “is metadata” will be used to differ between experimental measures and impact which are both stored as “dataset” entities within a “field”

instance. The impact dataset will be used in order to train, optimise and validate the decision-support predictive models.

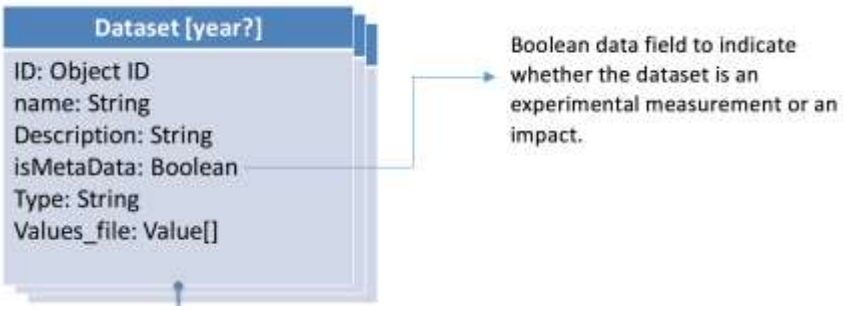

Figure 3. The dataset MongoDB collection table.

2.2 SMIS interface

2.2.2 Data browsing

As stated earlier Node.js will be chosen as the application server for the SMIS UI. The server will be extended with ExpressJS, a web application framework to simplify building the web platform. The choice of Express was based on its popularity and its involvement in the MEAN (MongoDB ExpressJS AngularJS NodeJS) stack architecture. The back end also interacts with R libraries to perform classification and regression of based on the experimental data storage module.

On the front end, the visualisation represents an important side of this software. That's why D3.js, a library for creating interactive and neat plots, will be widely applied. Fields and experimental data will be displayed as JQuery tables with multiple-columns compound search functionalities (See Figure 3 for table display example). This allows rapid allocation for a given experimental and/or fields entity within the database.

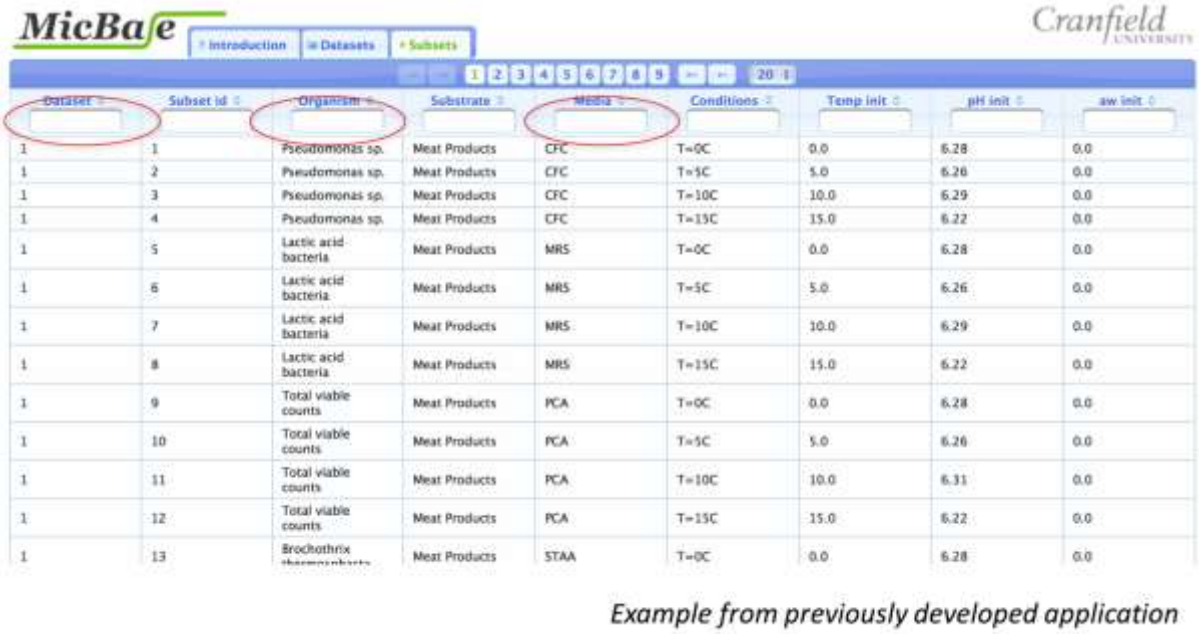

Figure 4 Fields and experimental data will be displayed in a tabular view. Interactive search. Example from a previously developed application is showed in this figure **for illustration purposes only**.

To visualise data, the majority of computations will be performed on the client and server side. Three main functionalities imply calculations: displaying the raw data, performing dimension reduction methods and executing classification modelling and machine learning. For these three actions, the same process is applied to retrieve the datasets (Figure 5). For instance, upon a user's query to browse an experiment, the experiment ID is added to the Uniform Resource Locator (URL). Then, the web browser sends this request containing the ID to the server. The server communicates with the database and retrieves the data related to the selected experiment thanks to its ID. Then, the datasets are transformed from JSON objects to two dimensional arrays.

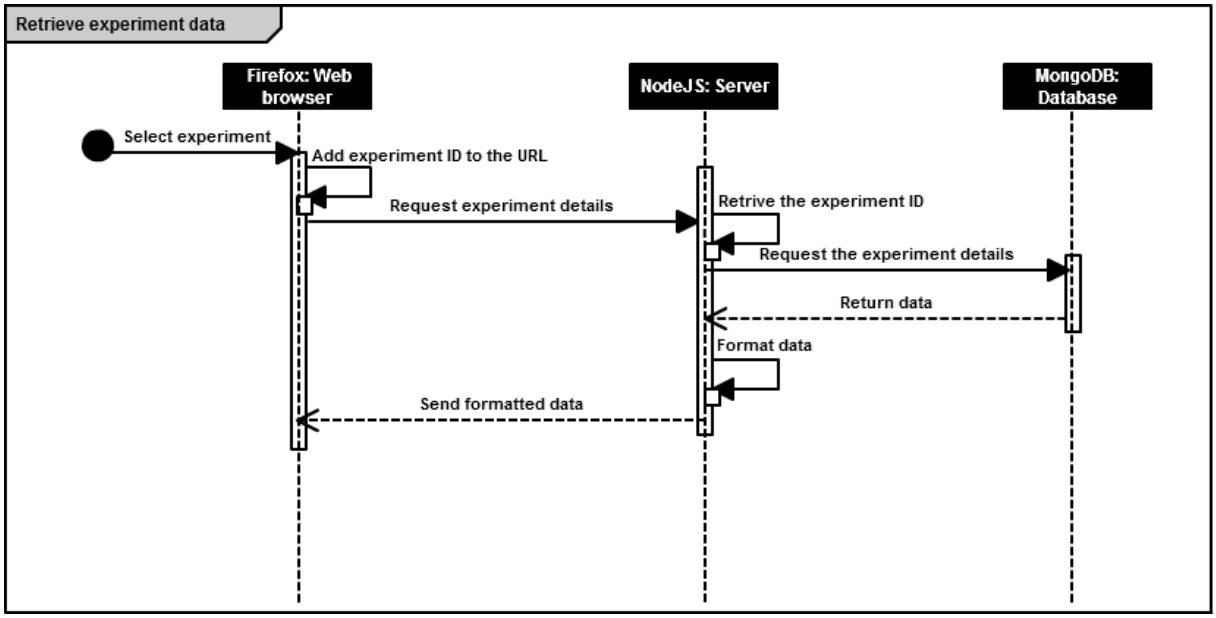

Figure 5: Sequence Diagram for retrieving experiment data on SMIS

To display data, after retrieving the datasets, if the data comes from an experimental platform, the web browser reveals the data in an array and the dataset type is displayed to know which dataset the array refers to. For the metadata, the type influences the way the data is represented. The NVD3 library will be employed to create interactive plots for each type. NVD3 provides neat and easily adapted examples. NVD3 provides neat and easily adapted examples, which could be adapted to provide an alternative view to the fields and experimental datasets in addition to the tabular view (See Figure 6 for an illustrative example).

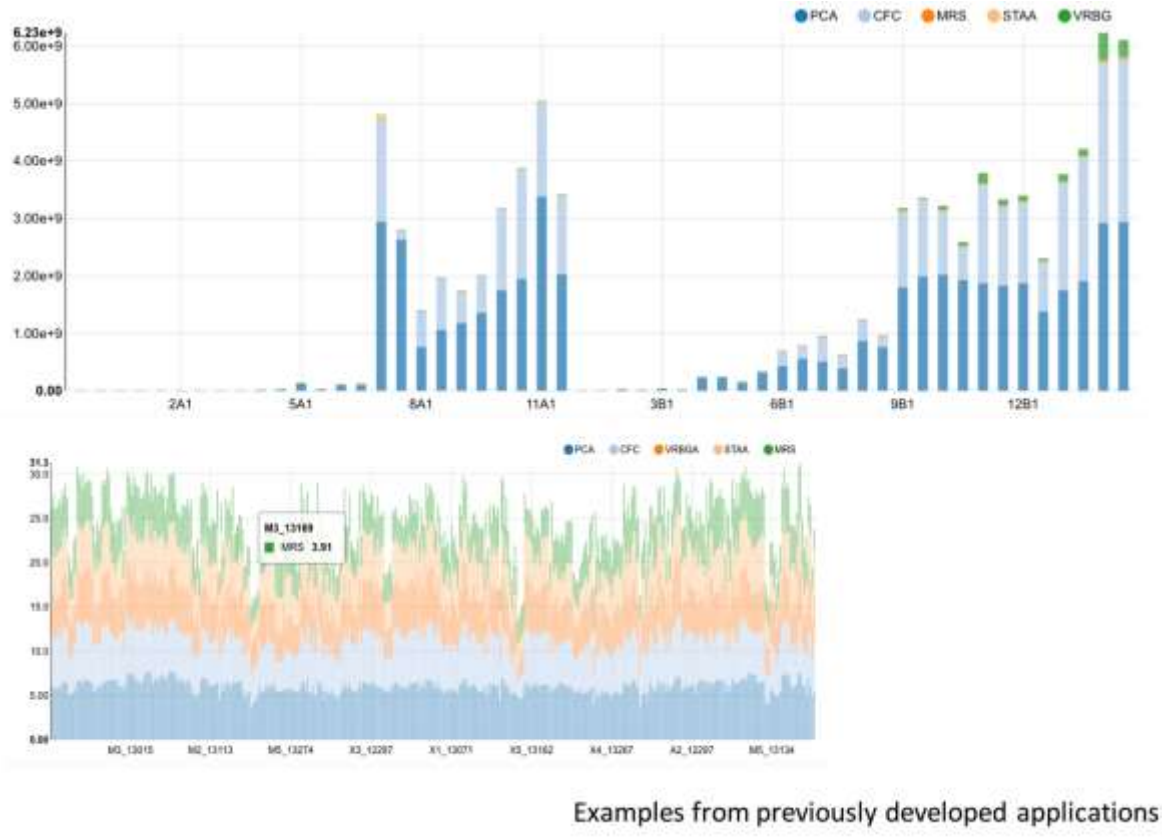

Figure 6. Interactive plots developed using the NVD3 library. Figure is adapted from previously developed application and is for illustrative purposes only.

2.2.3 Evidence-based data visualisation

Evidence data as described in Section 2.2.1 will be used in order to generate an interactive network visualisation graph, linking different problems to causes as supported by the literature, experimental and empirical evidences (See Figure 7). Evidence data network will be visualized using the Cytoscape.js library using nodes and edges. “Problems” and “causes” are presented as polygon and circle nodes respectively. The weight of a given problem (visualized by the thickness/colour of the edge vector) will be calculated via a mathematical formula depending on number and type of evidences supporting the corresponding relationship. The plotted network will be interactive, such as upon clicking on a particular problem, a zoomed-in view network will be displayed highlighted the corresponding solution(s). “Solutions” and “Problems” will also be visualized in a network view, where the edges weight (i.e. edge thickness/colour) corresponds to the confidence value assigned to the solution through experimental, empirical and literature findings.

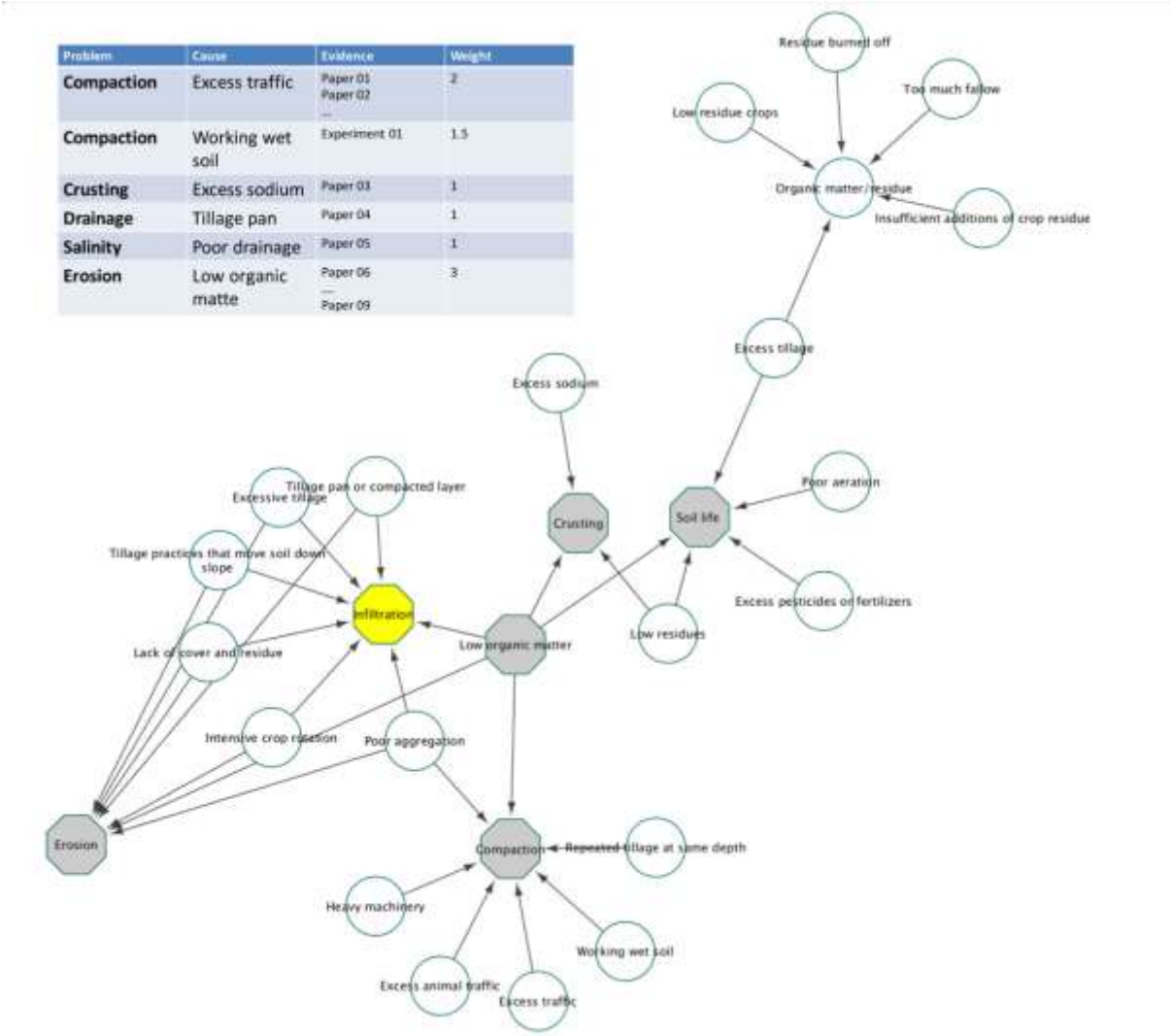

Figure 7. Evidence data network visualisation using the Cytoscape.js library. In this figure, problems and causes are presented as polygon and circle nodes respectively. The weight of a given problem will be calculated via a mathematical formula depending on number of supporting evidences for the corresponding relationship.