

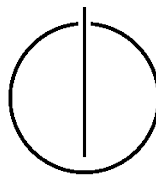
FAKULTÄT FÜR INFORMATIK

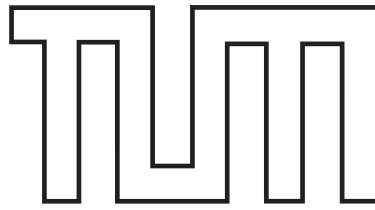
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

**Design and Prototypical Implementation
of a Web-based Spreadsheet System
for Managing and Analyzing Semi-structured Data**

Alexander Meissner





FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

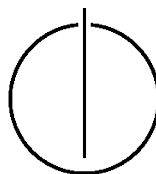
Design and Prototypical Implementation of a Web-based Spreadsheet System

for Managing and Analyzing Semi-structured Data

Konzeption und prototypische Umsetzung eines webbasierten Spreadsheet-Systems

zur Verwaltung und Analyse semi-strukturierter Daten

Author: Alexander Meissner
Supervisor: Prof. Dr. Florian Matthes
Advisor: Thomas Reschenhofer, M.Sc.
Date: October 15, 2015



Erklärung

Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I confirm that this master's thesis is my own work and that I have documented all sources and material used.

München, den 15. Oktober 2015, Alexander Meissner

“There’s something that happens with the collection of a large amount of data when it’s dumped into an Excel spreadsheet or put into a pie chart. You run the risk of completely missing what it’s about.” — Aaron Koblin

Acknowledgments

Dedicated to Family, Friends and the One.

Abstract

Today's enterprises are heavily reliant on the use of spreadsheets (e.g., Microsoft Excel) in order to enable end-users to interactively analyze and visualize certain data sets on their own. The intuitive UI of spreadsheets enables that even non-IT-affine users are able to design complex spreadsheets. This spreadsheet virtue has led to the rise of spreadsheet use in many critical functions of organizations. It has also led to the increased risk of organizational department failure due to the error-proneness of spreadsheets. Numerous studies have been conducted to explore different approaches to solving this issue.

Based on the current research conducted in the field of error prevention for spreadsheets and solutions that already exist, this thesis sets out to develop its own approach to the prevention of errors in the form of a prototype, specifically with the ability of handling complex linked data. This is attempted by making use of an existing prototype of a data-table control, a functional domain-specific language (DSL) as well as a Hybrid Wiki REST API. It is in solving the issue of how complex linked data types are managed by spreadsheets that one not only introduces new approaches to collecting and displaying data in this field of research but also advances the field of spreadsheet error prevention.

Contents

I. Introduction	1
1. Problem Statement	2
1.1. Motivation	2
1.2. Research Question	4
1.3. Research Method	5
1.4. Outline	7
2. Environment	9
2.1. Business Needs	9
2.2. Existing Technology	10
3. Knowledge Base	13
3.1. Terminology	13
3.2. Errors in Spreadsheets	14
3.3. Type Checking and Model Approaches	14
3.3.1. Inferring Templates from Spreadsheets	16
3.3.2. ClassSheets	16
3.3.3. Label and Unit Approaches	16
3.3.4. The Assertion Approach	17
3.3.5. Data Mapping Approach	18
3.3.6. The Visual Approach	18
3.3.7. Service-Orientated Architecture (SOA) to Control Information Flow	18
3.4. Hybrid Wikis	19
3.4.1. Current State	19
3.4.2. Hybrid Wikis and the Spreadsheet Prototype	21
3.4.3. The SocioCortex Architecture	22
3.5. Attributes of the SocioCortex	23
3.5.1. Data Types	23
3.5.2. Additional Properties	23

II. Conceptual Design	25
4. Conceptual Design	26
4.1. SocioCortex Sheet Smart Layer (SocioCortex SSL)	26
4.1.1. SC Angular Library	27
4.1.2. SCSSSL Architecture	28
4.2. SCSSSL Controller Functions	28
4.2.1. SCSSSL Number - Number Transformation Functions	30
4.2.2. SCSSSL Date - Date Transformation Functions	30
4.2.3. SCSSSL Enum - Enumerable Support Functions	31
4.2.4. SCSSSL Rich String - WYSIWYG Transformations	31
4.2.5. SCSSSL Derived Attribute Support Functions	31
4.2.6. SCSSSL Text - Text Countdown / Validation Function	32
4.2.7. SCSSSL Referenced Entity Support Functions	32
4.2.8. SCSSSL No Type Support Functions	32
4.3. SCSSSL UI	33
4.3.1. Number UI	36
4.3.2. Date UI	38
4.3.3. Enum UI	39
4.3.4. Entity Reference UI	40
4.3.5. No Type UI	41
4.3.6. Derived Attribute UI	41
4.3.7. Rich String UI (WYSIWYG), Image UI & Text UI	41
4.4. SCSSSL Multiplicity Handler	42
4.4.1. Multiplicity Transformation Function	42
4.4.2. Multiplicity User Interface	43
4.5. SCSSSL UI Overall Mockup	44
III. Software Design and Implementation	45
5. Software design	46
5.1. AngularJS Framework	46
5.2. Spreadsheet Framework	47
5.2.1. The Wijmo Framework	48
5.2.2. The Flexgrid from the Wijmo Framework	50
5.2.3. SC Angular Library in Detail	50

5.3.	The SC Sheet Directive	51
5.4.	Bootstrap Loading Process	53
5.4.1.	The scBaseObject	53
5.4.2.	Multiplicity Loading	53
5.4.3.	Linked Entity Data	55
5.5.	scEntityData, scAttributes and scSet Objects	55
5.6.	Editing Process for Controls	56
5.6.1.	The Editing Process for Controls (SCSSL)	56
5.6.2.	The No Type Process	58
5.6.3.	The Reference Type Process	58
5.7.	Widget Editing Process (SCSSL UI)	59
5.7.1.	Implementation of SCSSL Date UI	60
5.7.2.	Implementation of SCSSL Enum UI	60
5.7.3.	Implementation of the SCSSL Number UI	61
5.7.4.	Implementation of other UI elements	61
6.	Evaluation	62
6.1.	Prototype Issues	62
6.1.1.	Using a closed source framework	62
6.1.2.	Ongoing development of the SocioCortex Framework	63
6.1.3.	Insufficient AngularJS and Wijmo Knowledge	63
6.1.4.	Time-frame and Underestimation of Scope	63
6.2.	Critical Reflection	63
7.	Conclusion	65
7.1.	Future work	66

List of Figures

- 1.1. Left: Image of VisiCalc - The very first spreadsheet program. Right: Image of Microsoft Excel, spreadsheet market leader with a market share of 94% in the 2000s. [35]. 3
- 1.2. Adaptated Version of the Design Science Research Framework [40] 6
- 3.1. Type Checking and Model Approaches 15
- 3.2. The Hybrid Wiki data model by Matthes et al.[33] 21
- 3.3. The SocioCortex Architecture [1] 22
- 4.1. SocioCortex and Spreadsheet 2.0 Components 27
- 4.2. The Social Cortex Smart Sheet Layer Architecture 29
- 4.3. High-fidelity mockup of the a Spreadsheet 2.0 table. 33
- 4.4. Type Conversion & Widget Display Flowchart 34
- 4.5. Low-fidelity wireframes of a number stepper widget (Number UI) 36
- 4.6. High-fidelity mockup of the number widget 36
- 4.7. Low-fidelity wireframes of a number stepper multiplicity widget (Number UI) 37
- 4.8. Low-fidelity wireframe of a date picker widget (Date UI) 37
- 4.9. High-fidelity mockup of a date picker 38
- 4.10. Low-fidelity wireframe of a selector widget (Enum UI) 39
- 4.11. High-fidelity mockup of a selector widget (Enum UI) 39
- 4.12. Low-fidelity wireframe of the entity reference UI. 40
- 4.13. High-fidelity wireframe of the entity reference UI. 40
- 4.14. Low-Fidelity Mockup of the MxL derived attribute column 41
- 4.15. Low-fidelity wireframe of the multiplicity widget 43
- 4.16. Multiple Values Mockup 43
- 4.17. High Fidelity Mockup of SCSSL UI Overall Mockup 44
- 5.1. Bootstrap Loading Process 54
- 5.2. Example of \$scope.scBaseObject 55
- 5.3. Example of \$scope.scset 56

5.4. Steps involved in the editing process.	57
5.5. Prototypical implementation of the SCSSL Date UI	60
5.6. Prototypical implementation of the SCSSL Enum UI	61
5.7. Prototypical implementation of the SCSSL Enum UI	61

List of Tables

1.1. Design Science Research Guidelines in the context of the thesis.	6
2.1. Comparison of Existing Technologies	12
3.1. Attribute Types of SocioCortex	24
5.1. Comparison of different spreadsheet and grid tables	49
5.2. Parameters for the SC Sheet Directive	52
5.3. Overview of scsslCtrl functions	59

Part I.

Introduction

1. Problem Statement

1.1. Motivation

Since the advent of the first spreadsheet computer program (VisiCalc see: Figure 1.1) in 1978 [31], spreadsheets have assisted humanity in visualizing and manipulating various quantities of data. Although in the early days spreadsheets may have proven difficult to master, over the years the knowledge barrier to entry has decreased. This has resulted in even the laggard individuals to master the basic functionality of a spreadsheet. Because of this, from an information systems perspective, spreadsheets form a vital part of any business or organization as their use ranges from “financial reporting to workload planning to general administration” [32]. They have become such a commonplace in society that they are often misused for tasks that they are not suited for which often results in errors. This is commonly referred to as the golden hammer anti-pattern [12].

Initially Microsoft Excel and Lotus were among the most popular systems available, however, over the past 20 years. Microsoft Excel has become the dominant market leader with a market share of 94% in the 2000s [35]. Today 1 in 7 people on the planet use Microsoft Office (which includes Excel) [3] and spreadsheets are “used in a huge majority of firms in the US and Europe”[32]. However, the expansion of the web and the birth of Web 2.0 have led to the emergence of web-based spreadsheet applications like Google spreadsheets and Zoho Docs that offer functionality similar to Microsoft Excel. Differently to Excel in which one has to buy from Microsoft, Google and Zoho offer this software at no cost and at the convenience of just using a web browser. This has not only resulted in Microsoft having to reinvent its office package but also has led to the rise of many new innovations.

Even though all of these spreadsheet applications have been built on the same premise of simple data modeling and transformations, their structure has not been changed in many years. Over the years the interface has become more intuitive (e.g., Microsoft office introducing “Ribbon-style” user interface [8]), yet there are still problems that remain with this archaic concept. These problems range from erroneous data entries to usability.

1. Problem Statement

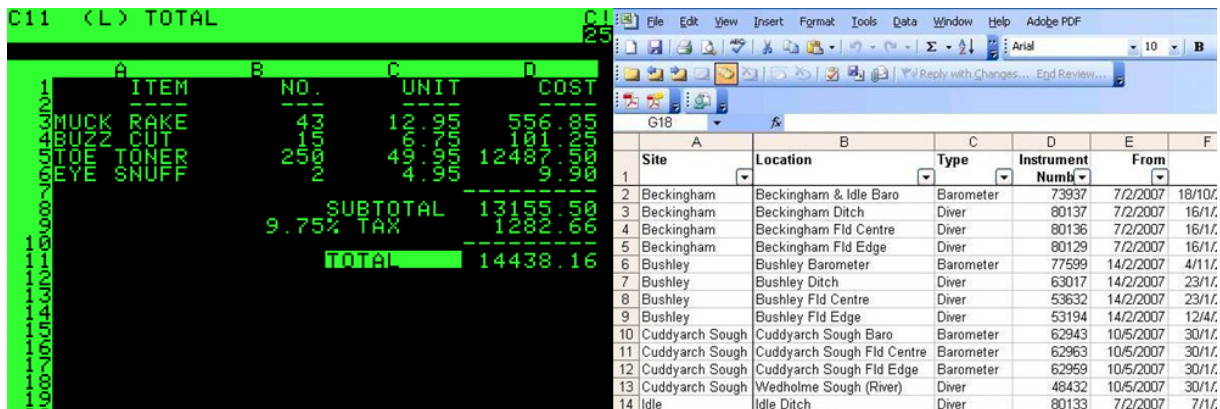


Figure 1.1.: Left: Image of VisiCalc - The very first spreadsheet program. Right: Image of Microsoft Excel, spreadsheet market leader with a market share of 94% in the 2000s. [35].

Specifically with regard to the erroneous data entry topic, there is an increasing number of research papers written about this topic and this is an indication for its severity. Panko [28] states that 94% of spreadsheets have errors and the average cell error rate is 5.2%. Although this number is refuted by another study (Powel et al. [30]) which claims that a more accurate cell error rate is 1.3%, it still shows that this issue is prevalent. Powel's study showed the following error distribution: 37.7% of errors are hard-code related[30], 32.9% are reference errors[30], 21.9% are logic errors[30] while the rest 7.3% percent are split among the errors of type copy/paste, omission and data input[30].

Numerous industry spreadsheet failure examples are a testament to how erroneous spreadsheets can

- cause share prices to fall as seen in the case of AstraZeneca which had confidential data embedded in template which ultimately lead to a loss of 0.4 in their share price[2].
- cause the lowering estimate of VaR in Basel II models by JP Morgan in 2013 by not reviewing the logic behind their spreadsheets and just resorting to manual copy and pasting[2].
- cause a \$6 million accounting error and \$12,500 audit fee because of a bad spreadsheet link in the Knox County Trustee's Office, US in 2011[2].

The issues of erroneous input data represent one of many other categories of spreadsheet issues. Matthes et al. [32] have categorized 20 of the most prevalent shortcomings as follows:

- Readability and understandability of the spreadsheet
- Extendability
- Manageability
- Collaboration and multi-user support
- Data
- Processes

The purpose of this thesis is to specifically focus on the specific problem area of “Data”. Specifically the implementation of a spreadsheet prototype that is capable of handling complex data types and assists users with input by the use of data modeling.

It is in solving the issue of how complex data types are managed by spreadsheets that one not only introduces new approaches to collecting and displaying data in the field of research but also advances the field of spreadsheet error prevention. Errors in spreadsheets are on the rise, and solving this issue will, from an Information Systems viewpoint, be essential to many organizations as they form such a critical part of them because of their increasing use in critical business applications[9].

1.2. Research Question

The research question that arises and that will be attempted to be answered is:

- *What is a model driven approach to managing complex linked data in order to prevent erroneous input in web-based spreadsheet systems?*

This consists of four main elements:

Web-Based Spreadsheets These are equivalent to traditional spreadsheet systems yet are accessible through web browsers which make them available from anywhere and allow for multi-user collaboration.

Model-Driven Approach This an approach that makes use of a model for the specification and classification of data.

Complex Linked Data As apposed to basic cell types in spreadsheets, complex linked data can contain multiple elements that consist of the primary type or other complex types. One could specify it is a composition of data types. An example of a complex

type is an address field. This can consist of a field for the address, a field for the city, a field for the suburb and a field for the postal code.

Erroneous Input Spreadsheets, stigmatized with the Golden Hammer [12] pattern, typically contain erroneous input. Erroneous input ranges from incorrectly formatted cells and number to invalid information.

These four elements establish the outset for this thesis, and the development of the prototype will be discussed in further detail.

1.3. Research Method

To accomplish the creation of a spreadsheet prototype the design science research methodology is used. This methodology can be used in Information Systems research to develop a prototype or artifact based on the existing knowledge of:

1. people, organizations and technology (“Relevance”): It is in the investigation of people, organizations and the technology that one can establish the business need.
2. research foundations and methodologies (“Rigor”): The current research leads to the collection of existing knowledge that can be applied in the development of the artifact.

The research methodology then proposes the development of an artifact which is then assessed, tested and refined in iterative cycles based on feedback from testing. Eventually, the refinement of the artifact leads to a prototype that can be applied to the business environment. Similarly, the knowledge which is gained in the iterative development can be added to the existing “Knowledge Base”. Hevner et al. [40] define Design-Science Research Guidelines which is followed in this thesis as follows in Table 1.1.

Design science research can be represented graphically as seen in the Figure 1.2, which defines the build up of this thesis which will be discussed in the outline.

1. Problem Statement

GUIDELINE	
1: Design as an Artifact	A viable artifact in the form of a prototype will be created which attempts to answer the research question.
2: Problem Relevance	We will sets out to solve problems as indicated in[32] which refers to business problems that occur with the use of spreadsheets.
3: Design Evaluation	It was planned that the quality, utility and efficacy of the design artifact (the prototype) would be carefully reviewed by third parties.
4: Research Contributions	This thesis will also set out to show the clear and verifiable contributions that arise through the prototype.
5: Research Rigor	To design and develop the prototype we will be looking at the current knowledge base. See Chapter 2.
6: Design as a Search Process	The prototype will be developed by making use of all available knowledge whilst satisfying laws in the problem environment.
7: Communication of Research	Even though the prototype is of a very technical nature this thesis will try to ensure that it is suitable for both “technology-orientated and management orientated audiences”. [40]

Table 1.1.: Design Science Research Guidelines in the context of the thesis.

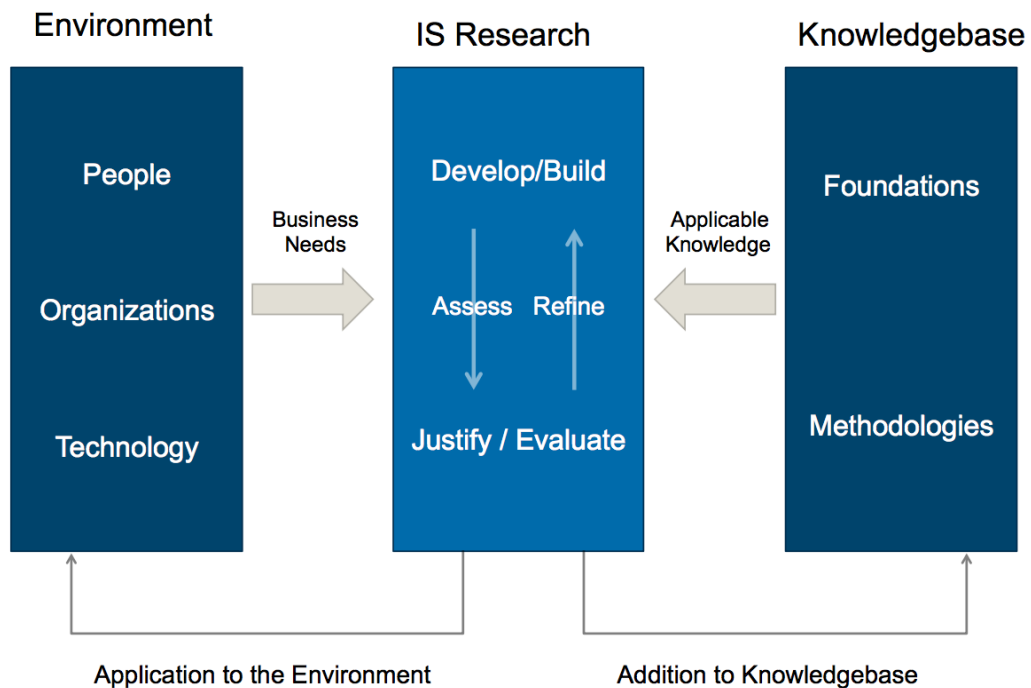


Figure 1.2.: Adaptated Version of the Design Science Research Framework [40]

1.4. Outline

This thesis is developed based on the design science research concept. Similar to artifacts that are developed on the two pillars of “Environment” and “Knowledge Base”, this thesis is from the onset dealing with these two pillars.

In the “Environment” (Chapter 2) section, existing scenarios are analyzed and business needs are determined. In addition to the needs mentioned in the Motivation chapter, additional ones are discussed. Here we will also be highlighting people and organizations that interact with spreadsheets and attempt to establish an overall stakeholder image.

Following this, existing technologies are examined and evaluated in detail. Depending on the business needs, solutions may already exist which cater toward the prevention of errors, and it is imperative that these are covered as they might be missing from the current research.

Succeeding the examination of the “Environment” is the “Knowledge Base” which is explored. The “Knowledge Base” for this topic is deeply rooted in research. Studies with similar approaches are assessed and discussed. Specifically model-driven approaches are examined. This thesis explores the existing foundations and methodologies that have been researched in the past and defines applicable knowledge and principles that can be used in the construction of a prototype.

After covering the “Knowledge Base” and the “Environment”, this thesis discusses the envisioned concept (Chapter 3). A detailed approach is presented which also encapsulates mockups that range from low fidelity and high fidelity. These mockups should provide a detailed picture of the final concept as well as provide a glimpse into its functionality.

This detailed concept is then be used in the implementation of the prototype (Chapter 4), and the implemented solution is discussed. While discussing the prototypical implementation, a detailed chronological plan, will accompany the discussion. Problem areas and issues that arose from the implementation of the described concept are also highlighted.

Unfortunately, because of a range of issues that occurred in the implementation phase, actual evaluation did not take place. According to design science research, this should take place in order to critically evaluate the proposed prototypical solution. However, as this thesis forms part of a dissertation on the topic of Spreadsheet 2.0, this evaluates a future prototype that will be built on this thesis. Therefore, in the last chapter (Chapter 6), we conclude with a summary that focuses on the research question and a discussion concerning the prototype. This will eventually be followed by the outlook in the field of

1. Problem Statement

the study of "static type checking" in spreadsheets and how this thesis has contributed to the current research.

2. Environment

Following the Design Research Framework that was mentioned in the previous chapter, this chapter sets out to analyze and define the business needs of organisations that are impacted by the research question. Herver et al. [40] stipulate that: “In it are the goals, tasks, problems, and opportunities that define business needs as they are perceived by people within the organization.” Thus, in exploring this premise, we can establish the environment. We briefly explore the stakeholders, organizational processes and strategies affected by errors in spreadsheets. Lastly, we take a look at the current technology and existing implementations that already exists, and that try to solve the problems that organizations have.

2.1. Business Needs

According to Baskara [10] spreadsheets are ubiquitous in large organizations as they heavily rely on them for data analysis, management reporting, and decision making. Many spreadsheet error horror stories[2] like the ones that are mentioned in the motivation, about AstraZeneca, JP Morgan and Knox County Trustee’s Office listed which are listed on the website of the European Spreadsheet Interest Group, reflect the notion that spreadsheet use not only is ubiquitous in every department of an organization but spans over all industries ranging from government to financial and academic. “Within certain large sectors, spreadsheets play a role of such critical importance that without them, companies and markets would not be able to operate as they do at present“ [17].

Thus, spreadsheet errors affect all levels of organizations and all business processes. Financial and reporting processes are however severely impacted by spreadsheet errors, and thus the need to resolve and fix these errors becomes an essential business need for an organization to function. And while a few market sectors control risks relating to spreadsheet use relatively well, the sectors of financial markets, fund management, investment research, and financial reporting remain of grave concern as spreadsheet risks are relatively uncontrolled[17], yet spreadsheets play such a critical role.

2.2. Existing Technology

After exploring the business needs, it is essential, for conciseness to also explore the existing technology. The reason for exploring existing technologies lies in the fact that even though much research is conducted in the field of erroneous input, some of the proposed solutions are already implemented in commercial software due to the fact that they solve a business need. In an investigation of similar technologies that tackle the problem of complex data and error handling, six have come to light worth mentioning:

Slate ¹Slate is a Microsoft Excel add-in that allows for users to evaluate complex spreadsheets by displaying a cells data through a structure interactive tree. By making use of the interactive tree users will not only be able to trace back how formulas are derived, but also allows for easy checking of inconsistencies. A problem with slate however still is that it is built on the primary structured data and therefore unable to handle complex types. This is a problem that many of the other software solutions have as well.

ZK Spreadsheet Server ² The ZK Spreadsheet Server is standalone server software that allows for real-time collaboration. It features a simple version of an online spreadsheet. It does however feature “3-D cell” which is its contiguous blocks of cells reference feature. Furthermore it boasts external book reference support, sorting, filtering, formulating and auto-filtering.

Google Sheets ³ This is one of the state of the art online spreadsheets available. As with many others it supports cell formulas typically found in most desktop spreadsheet packages. Additionally, even though it lacks support in the front end intuitiveness department, it does however feature many more unique integrations and support for querying types ranging from simple arrays to complex database models⁴. To solve its inability to supply widgets for input assistance, Google makes use of Google Forms⁵. Google Forms assists users when entering data and displays the result as a data set in Google Sheets.

Zoho Sheet ⁶ The Zoho Docs Suite features a whole range of online collaboration options, amongst these the Zoho Sheet. The Zoho sheet, similarly to Google Sheets features

¹<https://useslate.com/>

²<http://www.zkoss.org/>

³<http://www.google.com/sheets>

⁴<https://support.google.com/docs/table/25273?hl=en>

⁵<https://www.google.com/forms/about/>

⁶<https://www.zoho.com/docs/sheet.html>

basic spreadsheet functionality to handle formulas and numbers. Additionally this software allows users to categorize data by making use of filters and also supports data validation.

Excel Web App ⁷ Excel web app is part of Office 365. Microsoft with Excel is, as mentioned previously, playing catchup to other systems that have been available in web-based form. It does, however, have Sharepoint integration which allows for it to query from a Sharepoint Wiki site and does therefore resemble a similarity to the prototypical implementation with the Hybrid Wiki System SocioCortex.

Smartsheet ⁸ Smartsheet is a web based spreadsheet that contains all the standard functionality of a spreadsheet. Smartsheet allows for columns to be grouped and aggregated. Similarly to Google sheets, Smartsheet also features a web form editor to accompany users with entering data in a standardized way. In addition to the base functionality, Smartsheet allows for Excel importing and widgets like date picker or enumerable values. A critical look at Smartsheet shows that it contains elements like the widgets and cell grouping, that are essential for standardizing data and ensure consistency. However, it does lack the data linkage from other sources. It also has the problem, of being hosted on an external server and causing vendor lock-in issues to prevail.

Table 2.1 shows a summary of how these systems (and others) are compared to one another. The Excel web app is the front runner in this area based on its Sharepoint integration, yet it does still not entirely support Hybrid Wikis. Some cover the basic functionality, some even supply the option for widget support. All however do not support the complex linked data. Complex Linked Data is a type consisting of a schema that changes and can be referenced to and from other entities. This type will be elaborated on in the following chapter.

⁷<https://office.live.com/start/Excel.aspx?omkt=en-US>

⁸<https://www.smartsheet.com>

2. Environment

Existing Technology	Web Based	Real-time Collaboration	Widget Support	Complex Linked Data ⁹	Notes
LibreOffice ¹⁰	No	No	No	No	
MS Excel ¹¹	No	No	No	No	Uses “Flash Fill” to predict users final outcome
Apache OpenOffice Calc ¹²	No	No	No	No	Natural language formulas and uses wizards to guide users through different processes
ZK Spreadsheet Server ¹³	Yes	Yes	No	No	Own installation, not cloud hosted
Slate ¹⁴	No	No	No	No	Interactive tree
Google Sheets Google Sheets ¹⁵	Yes	Yes	No	No	Provides Google forms for input consistency
Zoho Sheet ¹⁶	Yes	Yes	No	No	
MS Excel Web App ¹⁷	Yes	Yes	Yes	No	Allows for linked data to be imported from Sharepoint
Smartsheet ¹⁸	Yes	Yes	Yes	No	Provides web forms for input consistency

Table 2.1.: Comparison of Existing Technologies

⁹Complex Linked Data is covered in Chapter 3 and refers to users being able to reference and edit data which is of a complex (composed of different types) type.

¹⁰<http://www.libreoffice.org>

¹¹<http://office.microsoft.com/en-gb/excel>

¹²<https://www.openoffice.org/product/calc.html>

¹³<http://www.zkoss.org/>

¹⁴<https://useslate.com/>

¹⁵<http://www.google.com/sheets>

¹⁶<https://www.zoho.com/docs/sheet.html>

¹⁷<https://office.live.com/start/Excel.aspx?omkt=en-US>

¹⁸<https://www.smartsheet.com>

3. Knowledge Base

In this Chapter we will examine the current knowledge base. This consists of all the research currently conducted in the field of error prevention and detection in spreadsheets. Initially, the base premise of developing solutions that reduce errors in spreadsheets is explored. Following the exploration of different approaches in academic literature, we focus at the proposed models in research specifically those of type systems. To conclude, as this thesis builds on the research conducted in the area of Hybrid Wikis and the SocioCortex from the sebis Chair at the Technical University of Munich, we explore these constructs and concentrate distinctly on their architecture and their attribute type systems.

3.1. Terminology

This thesis covers terms like spreadsheet, cell and value which are very commonly used and, due to this fact, might evoke the thought that no explanation or elaboration is needed. However, to prevent ambiguity, we will cover many of the terms in this section:

Spreadsheet A computer program that provides a user interface consisting of a collection of worksheets.

Worksheet A table which consists of columns and rows similar to an $N \times N$ matrix. The limit of the maximum number of columns or rows depends on the software. A unique cell exists at the intersection of a columns and a row.

Table A synonym for “worksheet”.

Cell A field for storing values, formulas and references to other cells.

Formula Formula are spreadsheet instructions which could be regarded as “programming code”. Formulas are often very basic as non programmers interact with them.

Value A representation of data in a cell. This value is normally entered in the spreadsheet manually.

Unit A secondary attribute or dimension assigned to a value which describes it. Examples of units: “cm, kg, inch”.

Label A label is an arbitrary category or type given to a value to describe it.

Schema A definition of a type that is applied to data.

3.2. Errors in Spreadsheets

One of the essential differences that one has to be aware of is that spreadsheets are developed in a different manner compared to conventional programs. While conventional programs are dictated by specifications (i.e. real-world-problems that are solved by specifying a solution and testing), a spreadsheet program is driven by trial and error. This means that one has to cater for an additional level of complexity within a spreadsheet and its ability to handle errors.

Current research in the domain of solving errors in spreadsheets can be split into two categories: “Testing and auditing” by using test case systems (e.g., the “What You See Is What You Test” approach [34]) and “static type checking.”

Testing and auditing is presently seen as an approach that not only is inaccurate [29] and ineffective but also involves the high overhead of requiring a user to invest additional effort. Users become strained because of the lack of automation[15] which could prevent this by speeding up the process.

A better approach is that of model-based systems and static type checking [15]. The model-driven approaches do not only allow for automation but also provide more definition of the data. Figure 3.1 represents a portion of research being done in this field which will be further elaborated on.

3.3. Type Checking and Model Approaches

In the next section, we explore different types of checking and model approaches. These approaches range from being proactive, that is, determining a set of framework or model before the data is being entered such as ClassSheets or the Assertion approach, to being reactive, that is, modeling data that has already been entered into a spreadsheet by inference or classifying label or units.

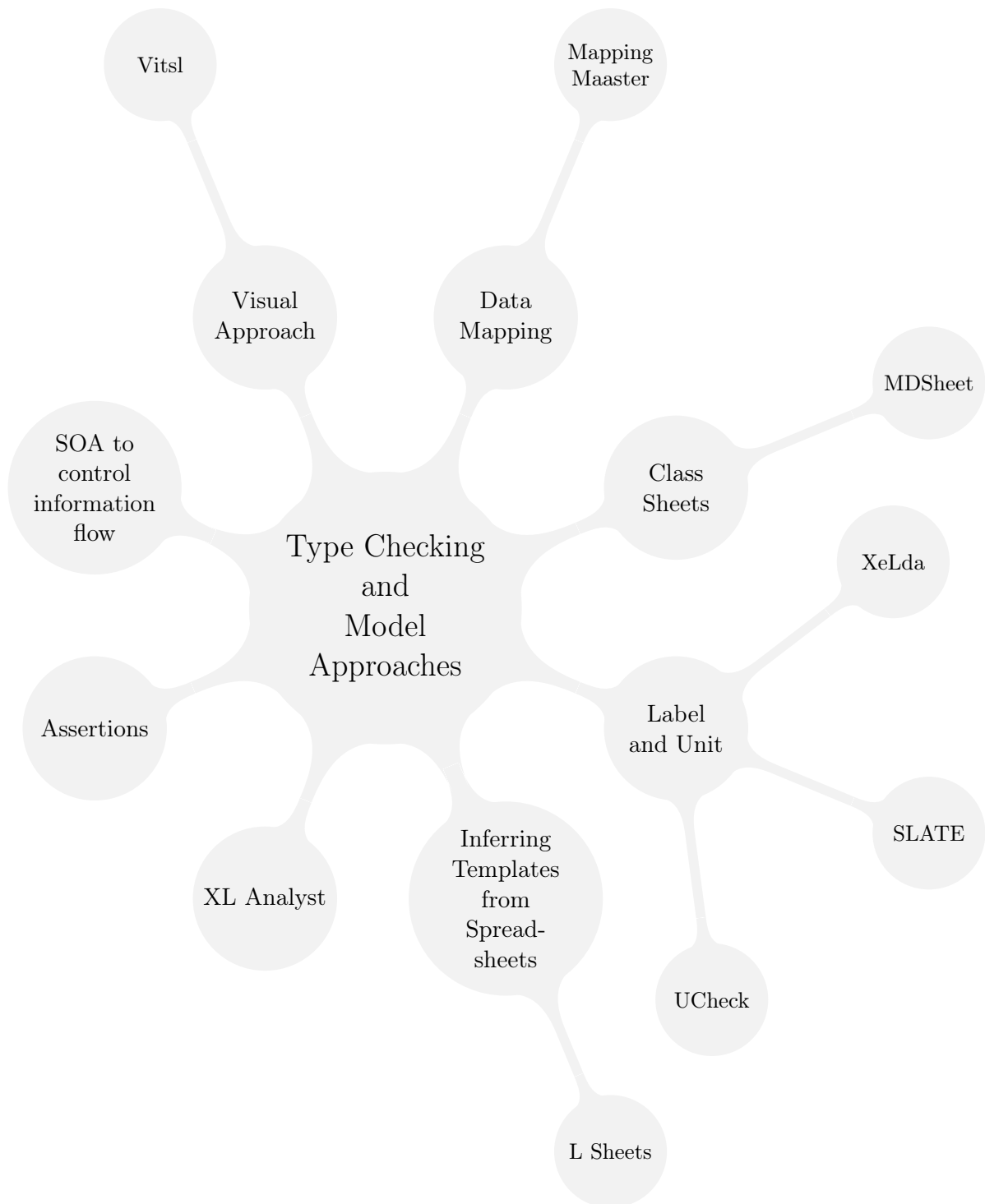


Figure 3.1.: Type Checking and Model Approaches

Note: the diagram above represents a map of all the approaches that will be discussed in the next part. Its purpose is to create an overview. Overlapping of certain concepts and models might occur.

3.3.1. Inferring Templates from Spreadsheets

Abrahams et. al. [4] proposed the solution, as data is not always modeled in a spreadsheet, to identify regions of “similarity” within worksheets. Then by grouping these, templates are inferred based on the grouped properties. This approach allows for data to be modeled retrospectively which is vital when dealing with legacy tables.

3.3.2. ClassSheets

Jorge Mendes [26] defines ClassSheets as “high level, object-oriented formalism to specify the business logic of spreadsheets.” With ClassSheets business logic can be modeled and expressed in business object structure which is similar to UML. In ClassSheets, it is possible to define types of values for the columns in a spreadsheet. Then with this information, a model is established that will form the foundation of the spreadsheet and its values. One of the issues that ClassSheets faces is that for every change that is made to the model, the resulting table would have to be regenerated.

3.3.2.1. MDSheet

MDSheets [18] allow for a solution to this problem. By making use of “semantic”, “combinator” and “layout” rules, the MDSheet adds forward and backward transformation capabilities to ClassSheets. This allows for increased legacy spreadsheet support and assists end users in the safe and correct input of data[18].

3.3.3. Label and Unit Approaches

By making use of labels [16] or units [6] input data receives context and additional information which can categorise it and prevent errors. One could say that the information here is modeled based on the label applied to it or the unit of measurement that it has. These models have resulted in the introduction of SLATE and UCheck.

3.3.3.1. SLATE - “A Spreadsheet Language for Accentuating Type Errors”

SLATE[16] (not to be confused with the Slate Microsoft Excel add-in highlighted in the existing technologies chapter) separates the unit from the object of measurement,

and defining new semantics for spreadsheets so that both the unit and the object of measurement are taken into consideration[16]. In SLATE every cell object is split up into three attributes: a value, a unit and a label. Slate for instance allows users to indicate that they have “25 kg of Copper.” Here 25 would be the value, “kg” the unit and copper the label. By adding dimensions of units and labels SLATE is able to display when values that are being added together are incorrect. This allows for users to add more specificity to data. By the use of a unit or label indications, cells can be categorized and become a “type”. A problem with slate however still is that it is built on the primary structured data (string, integer) and therefore unable to handle complex types such as dates and lists. Moreover, SLATE only transforms labels and dimensions and does not identify errors through additional logic.

3.3.3.2. UCheck

UCheck[4] is a system that provides unit checking for spreadsheet formulas. According to Abrahams and Erwig [4], UCheck uses a two-step process for error correction. The first step involves inferring labels automatically to cells, and the latter step then involves assigning units to those inferred cells based on the prior inference. The system then carries out automated checking on the formulas and identifies problems based on a defined unit system[20].

3.3.3.3. XeLda

XeLda[7], a tool for highlighting errors in cell input, is nearly identical to SLATE. On the basis of comparisons of labels to other labels and units to other units XeLda can pick up if there is a clash between the two and alerts the users to this issue. Its difference to SLATE is that it is a tool that can be utilized in MS Excel.

3.3.4. The Assertion Approach

Burnett et al. [14] propose a solution for the reduction of errors in spreadsheets by defining assertions prior to the users’ input which limit it to those assertions being made. By making use of a particular syntax users are able to predefine conditions such as ranges or certain sets of numbers that should be accepted as input. These would then apply to user input as post conditions when executing a cell or as preconditions to “cells further down stream”[14]. This can model a user’s input from the outset and efficiently allows

the user to debug more effectively. The studies also showed that end users did not only understand the assertion but like them as a concept.

3.3.5. Data Mapping Approach

While the Assertion Approach deals with defining rules, O'Connor et al. [27] explore the angle of data mapping. Specifically with regard to the Web Ontology Language (OWL). Their developed declarative OWL mapping language (called *Mapping Master* or M^2) allows users to define OWL entities from within a spreadsheet by making use of the correct syntax. It supports referencing, entity mapping, and missing value handling. One of the reasons for the development of M^2 is the matter that this DSL should support complex spreadsheets that do not conform to the entity-per-row assumption [27]. (This construct will be revisited later again in this thesis as the SocioCortex modeling language MxL.)

By making use of mappings one is able to bind table data displayed in cells to a certain data source. Making use of such concepts allows for the view of data to change yet allows for the source to stay constant. It also provides an increase in the amount of information. M^2 provides a range of benefits for spreadsheets ranging from financial spreadsheets to ontological ones to international disease classification.

3.3.6. The Visual Approach

The Visual Approach, which is similar to the ClassSheet approach, confers that spreadsheets can be conceptually split into two phases[5]. The first being the creation of a model or template that defines cell headers and formulas. The second is when data is entered into this template. On the basis of this reasoning Abraham et al. [5] developed a tool, called the Vitsl editor, which provides a GUI approach for the establishment of spreadsheet templates.

3.3.7. Service-Orientated Architecture (SOA) to Control Information Flow

As apposed to changing the actual spreadsheet software and with ever more organizations having to satisfy requirements of Section 404 of the Sarbanes-Oxley Act, this specifically stipulates that the company should be “maintaining an adequate internal control structure

and procedures for financial reporting” [39] which has resulted in research that covers the intersection between spreadsheets and the actual data that is inserted. Samar et al. [36] propose the implementation of service-orientated architecture after analyzing the bottlenecks and pitfalls of current systems. On the basis of their analysis many of the errors occur in the input process through either “copy pasting” being a “manual process, with high error and no validation” [36] or using importers such as ODBC, web query or simple csv. While simple csv, xls imports tend to break linkage to the original data, more complex importers are often too complicated and business users lack certain knowledge to use them.

The SOA solution developed by Samar et al. [36] adds a new layer to the spreadsheet paradigm which lets the spreadsheet still behave and do what it primarily does. However, by making use of Excel Plug-ins it is possible to add another layer to the spreadsheet paradigm that connects the data of a spreadsheet to services. This layer features data access, orchestration, transformation, security and resource management.

3.4. Hybrid Wikis

We have looked at many different approaches to solving the issue of erroneous input. A few of these concepts will be used and embedded in this thesis such as the assertion approach and the data mapping approach. Before we commence with the conceptual part of this thesis it is important to still address one additional topic: Hybrid Wikis. Hybrid Wikis are important to cover as this thesis build up on such a system, namely “SocioCortex” which forms the cornerstone of the Spreadsheet 2.0. Hence, it is essential to briefly explore this concept and the current state of its research.

3.4.1. Current State

Hybrid Wiki “*A lightweight approach for data and information management within enterprises facilitating structuring of content for business users.*”[25]

The hybrid wiki concept aims to solve the problem of unstructured wikis - specifically unstructured content. Prior to the hybrid wiki, numerous research was conducted in the area of templating wikis [21, 19] and semantic wikis [37, 38, 13]. The hybrid wiki builds on this knowledge and extends it with its own paradigm: As opposed to traditional wikis which only support simple pages with rudimentary methods of input (e.g., text

and tagging), hybrid wikis allow for users to extend and enhance basic page information, Matthes et. al. [25] defined hybrid wikis to allow for users to assign *attributes* and *type tags*.

Attributes Key value pairs that consist of an attribute name and a value. Attributes allow for additional content to be added to the page as page content.

Type Tags Allow for information on pages to be defined and categorized. By making use of type tags pages containing information, for example, about a team member or knowledge page become team member pages or knowledge pages.

The assumption of the Hybrid Wiki is that data evolves over time and it is the model which has to be adjusted for the data not vice versa. Both attributes and types can be related to data and allow for more structure, yet they are not necessary for the data to exist.

Figure 3.2 shows the Hybrid Wiki data model. The Hybrid Wiki allows for types to be defined, which may have several attributes also defined which create a model (also referred to as schema) that is then applied to the unstructured data[33]. According to Matthes et. al.,[33] unstructured data and schema structure is applied to one another in real-time to provide the requested output of information.

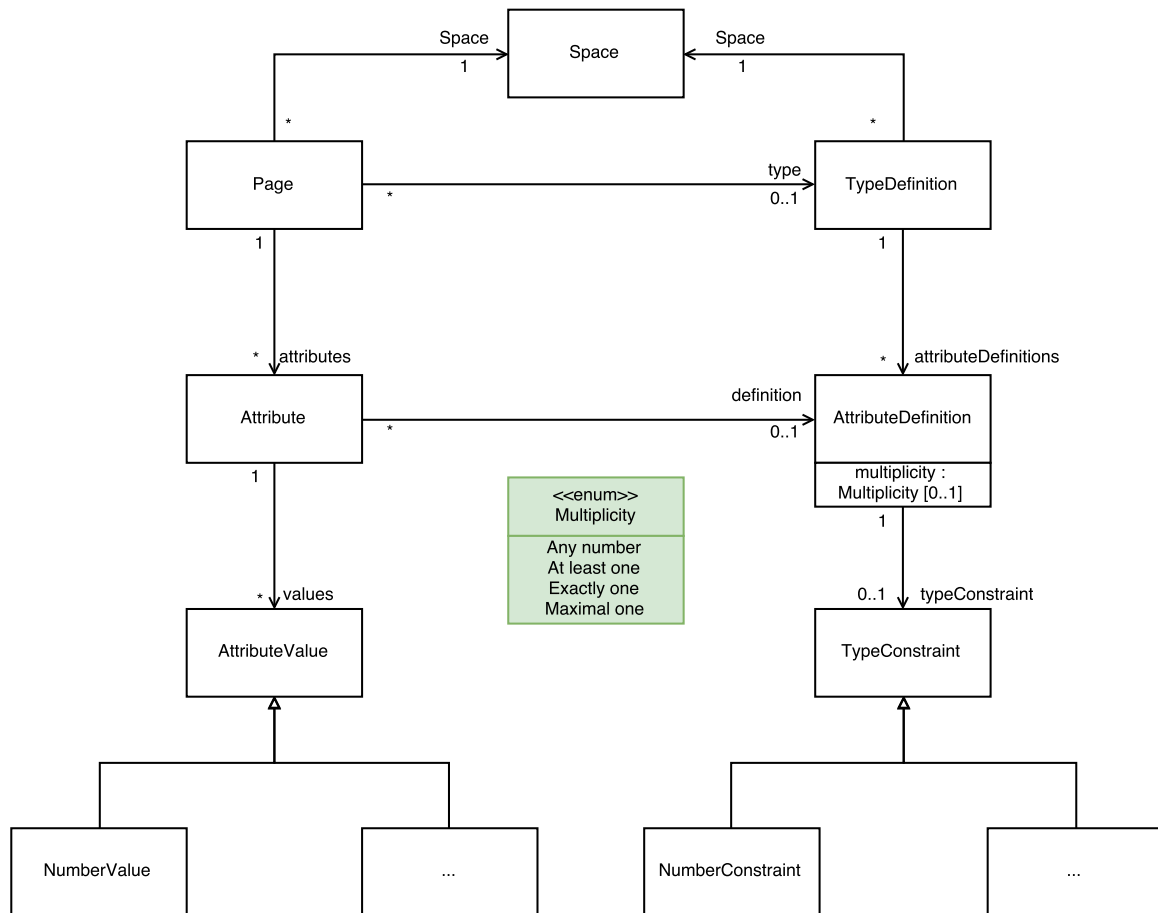


Figure 3.2.: The Hybrid Wiki data model by Matthes et al.[33]

3.4.2. Hybrid Wikis and the Spreadsheet Prototype

As mentioned in the paper by Matthes et al. the purpose of a hybrid wiki is to “enable all users to enter structured data, in contrast to a two-phase process where inexperienced users enter textual content that is later enriched with annotations by experts”[25]. This thesis aims to achieve this by making use of a widely used tool, like a spreadsheet, for enabling all users to interact with the hybrid wiki and its contents. The addition of a spreadsheet for the management of a hybrid wiki does not only mean that users do not have to learn new concepts, as experience spreadsheet users already possess the needed

skills and knowledge, but will also by making use of spreadsheet concepts will also allow for an increase in efficiency when populating data into a Hybrid Wiki.

The current version of the Hybrid Wiki developed at the SEBIS chair is called SocioCortex. Besides the Hybrid Wiki, SocioCortex brings with it Social integration, Semantic integration and Content integration [1] which add an enhanced sphere of collaborative activities.

3.4.3. The SocioCortex Architecture

Figure 3.2 depicts the current SocioCortex architecture. SocioCortex offers specific layers of functionality which is shown by the different colors. All of these layers interact with data and message connectors as well as functionality for bulk dumping and loading of data. The SocioCortex is accessible via a REST API which allows a range of web application to interact with the different layers.

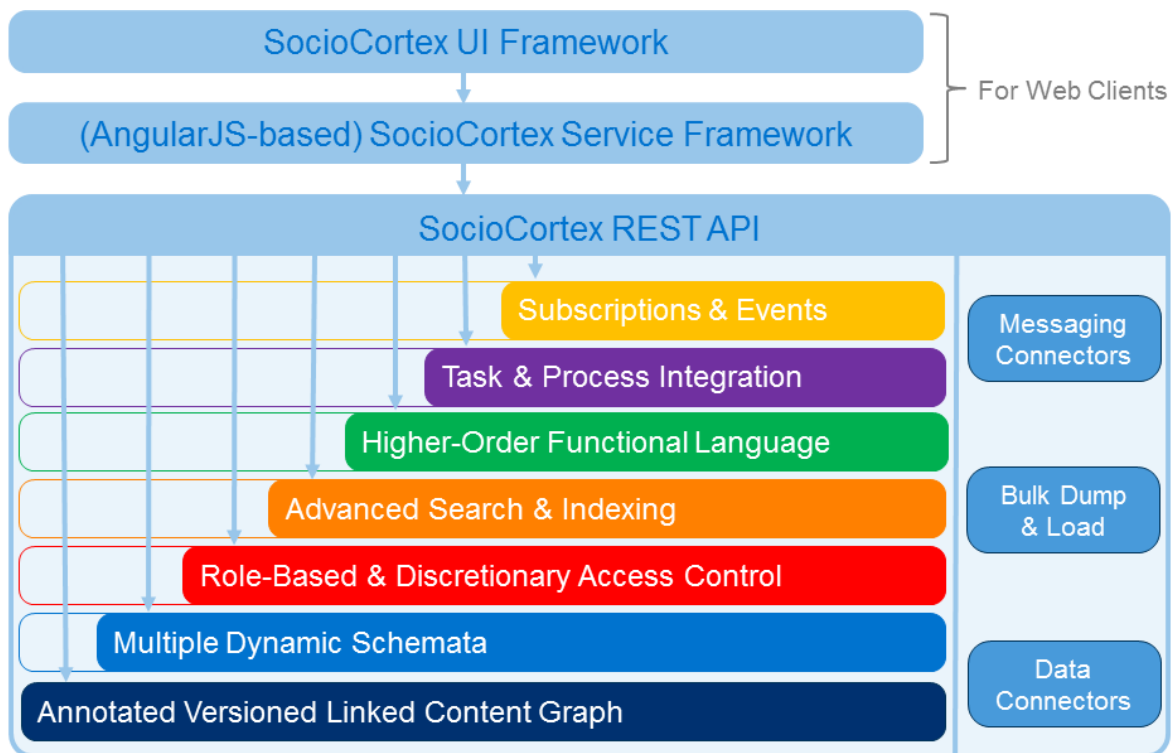


Figure 3.3.: The SocioCortex Architecture [1]

3.5. Attributes of the SocioCortex

3.5.1. Data Types

As this concept builds up on SocioCortex it will encompass the current data types that SocioCortex offers. However, this can be extended to encapsulate other types as well. The attributes of SocioCortex currently support the following types shown in Table 4.1

The attributes of SocioCortex are defined by specified attribute types which allow for change. If an attribute experiences a type change in its lifespan, the data it holds will not cease to exist or prevent this change. The rationale for this, is that a system like SocioCortex needs to adapt its structure based on its data and not the reverse. This leads to the system indicating which values need to be adjusted in order for them to adhere to the type structure and not forcing the user to have to migrate the data.

3.5.2. Additional Properties

In addition to the attribute type definitions, SocioCortex caters for the following additional properties:

Multiplicity A user is able to define the multiplicity of an attribute. The current choice consists of at least, at most and exactly one value. In addition to this, the option of selection any amount of values is also allowed.

Read Only Users are able to define whether objects are able to be edited or not.

Show in Tables By selecting this option users are able to select whether or not options should be shown in tables.

Default Values Every attribute can have default values defined that will show when a user initially does not fill anything in.

Type	Description	Examples
Text	Also referred to as a string value, the purpose of this type is to store short key values.	Names, Telephone Numbers
Boolean	This data type can only have two possible values.	true, false
Long Text	This data type can be seen as one of the primitive wiki types. The differentiation between Text and Long Text is its storage capacity.	Lone pieces of information, Articles, Biographies
Number	Also referred to as an integer value, the purpose of this type is to store numbers which can be used in calculations.	12; 99; 200
Date	Defines a data type of type date which can then be used as an attribute.	07.07.2015
Enumeration	Defines a list of options. End-Users select an option from a list of predefined options.	Option A, Option B, Option C
Image	Defines images that can be stored in the system.	Profile pictures, charts, graphic content.
Reference	This is a link to another SocioCortex object. Any object can be referenced.	Other content pieces or entities
Rich String	A rich string is usually used for a WYSIWYG editor. Rich Strings contain code that help with the formatting of the object. Currently only HTML is supported.	<i><i>This text consists of italics</i></i>
No Type	Entities can also have “no type” definitions. These can then at a later stage be more defined. The “No Type” type allows for any type of input to be entered, i.e. it can consist of date, references and text entities.	08.07.2015 22 Names

Table 3.1.: Attribute Types of SocioCortex

Part II.

Conceptual Design

4. Conceptual Design

In the following chapter we explore a conceptual design which sets out to solve the initial research question. We introduce the concept of the SocioCortex Sheet Smart Layer Framework which accomplishes the rendering of widgets and conversion of data between SocioCortex and a Spreadsheet framework. After exploring this concept, its architecture and components, we make use of mockups to demonstrate features, specifically widget design that interacts with the SocioCortex and the Spreadsheet framework.

4.1. SocioCortex Sheet Smart Layer (SocioCortex SSL)

This thesis is concerned with the front end experience of the SocialCortex which will be managed by making use of the spreadsheet paradigm. This resulted in the conception of the SocialCortex Smart Layer model (See Figure 4.1).

The SocialCortex Smart Layer or SCSSL sits between the SocialCortex and a spreadsheet framework, and is used for front end management. SCSSL provides the spreadsheet framework with widget support as well as SocioCortex type handling and mapping.

As mentioned previously, the SocioCortex attributes and their types are defined in a back end web interface, however when developing a prototype toward Spreadsheet 2.0 there is a need to supply additional logic in the front end which assists users with their input before it is being sent to the SocioCortex REST API: SCSSL provides this logic. For every type that exists in the SocialCortex a front end model is proposed to be developed. Additional front end logic is also required to assist the user when editing the spreadsheet.

SCSSL is built up similar to the Model-View-Controller concept. The “controller” is represented by the “SCSSL Controller Functions” and the “Multiplicity Handler”. The “SCSSL UI” could represent the “view” whilst the defined types from SocioCortex represent the “model”. A graphical representation of the role that SCSSL has in conjunction with the SocioCortex and Spreadsheet 2.0. paradigm is reflected in figure 4.1.

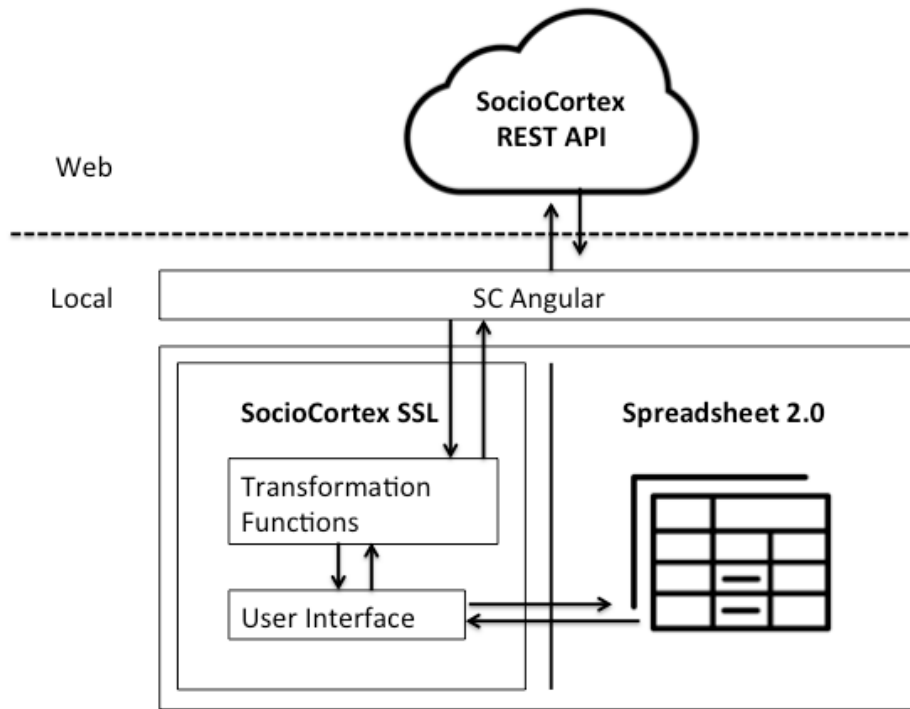


Figure 4.1.: SocioCortex and Spreadsheet 2.0 Components

On the web side: the SocioCortex Rest API and SocioCortex are positioned in the cloud and accessible on a predefined endpoint. Locally (or client side) - three main components will interact: the SC Angular Library, the spreadsheet framework used in Spreadsheet 2.0 and SocioCortex SSL (SCSSL). We will further on explore these in more detail, specifically SCSSL.

4.1.1. SC Angular Library

SC Angular is a framework used at the Sebis Chair of the Technical University of Munich. Initially programmed by Patrick Bürgin for his master thesis, SC Angular allows for front end components to authenticate and communicate with the SocioCortex REST API. Furthermore it provides wrappers, unwrappers and utility functions for fetching and handling multiple SocioCortex objects. The SC Angular Framework will be used as the main communication framework between SCSSL and SocioCortex. A more in-depth look at SC Angular and its integration is covered in the “Implementation” chapter (Chapter 5).

4.1.2. SCSSL Architecture

Figure 4.1 describes the SCSSL containing components of transformation functions and user interface. SCSSL transformation or controller functions are responsible for sanitizing and handling data between the SCSSL UI Widgets and the SC Angular library. The SCSSL UI is responsible for rendering widgets that handle input data from the Spreadsheet 2.0. For every SocioCortex type different handlers, widgets and transformation functions are defined which are shown in Figure 4.2 and which form the SCSSL library.

4.2. SCSSL Controller Functions

The rationale for having front-ended transformational (controller) functions is the fact that there might exist a discrepancy between user input shown in the spreadsheet table and the data coming from SocioCortex. It is important therefore to “sanitize” data both ways by making use of transformation functions. Furthermore transformation functions also specifically allow for a table raw edit mode:

Table Raw Edit Mode This mode allows for users to be able to edit table cell values directly without making use of widgets. It allows for keyboard only navigation and cell editing. I.e. Instead of displaying a date picker widget to select the date by tabbing through the options - Table Raw Edit Mode allows for users to edit the value of a date in a cell directly. This mode still makes use of transformation functions but bypasses the SCSSL UI. The rationale for the existence of raw table mode lies in assumption that keyboard only navigation and input is more efficient than using a mouse, especially when it comes to user interfaces[24].

Widget Edit Mode This mode allows for users to be able to edit table cell values by making use of SCSSL UI Widgets.

Transformation functions are the first step in a two prong implementation approach. They are necessary for text to object conversions which need to occur for some of the widgets that need to show. A date picker widget, for example, requires date objects as input and output while a number picker would need an integer or a float. Both of these would therefore require the initial data from SocioCortex to be converted from strings to their respective types.

Transformational functions also add a layer of error handling to the input as they perform the sanitization which will attempt to also interpret the data that is passed to it.

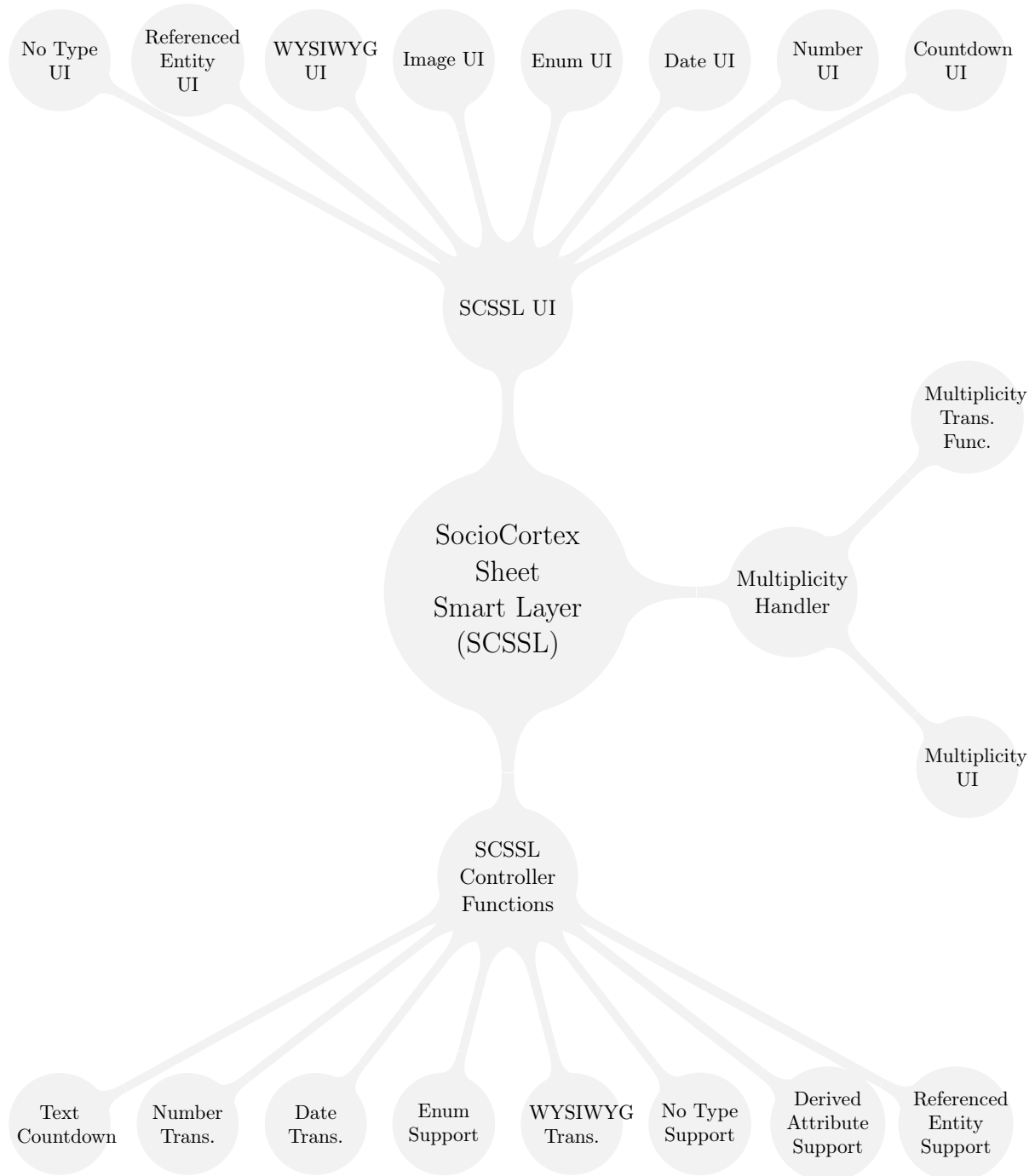


Figure 4.2.: The Social Cortex Smart Sheet Layer Architecture

We further explore the transformational functions in detail that are required for the interaction between the SocioCortex and the Spreadsheet 2.0.

4.2.1. SCSSL Number - Number Transformation Functions

The number transformations should be concerned with the SocioCortex attribute of type number. These number transformations are responsible for:

1. Internationalization - As some countries use “,” while others use “.” to separate decimals this can cause issues with formulas and input. These will be filtered and sanitized so that different formats are allowed.
2. Four digit and higher separators sanitization - Similarly to internationalization, four digit (1.000 for a thousand) and greater (1.000.000 for a million) numbers are separated by fullstops or spaces which can result in incorrect input and output. The number transformation will sanitize this and make sure that strings of this nature are convertible to integers or floats.
3. Natural language numbers - For numbers one to ten this transformation will also be responsible to convert numbers written in natural language to number format.
4. Conversion of SocioCortex Number to Integer or Float

The SCSSL number transformations should prevent erroneous input and ensure stability of the Spreadsheet 2.0. The default format for numbers in the spreadsheet should be float based.

4.2.2. SCSSL Date - Date Transformation Functions

Date attributes in SocialCortex take the unstandardized form of DD.MM.YYYY. This format can cause problems when being used in conjunction with a date widget which is one of the core purposes of the SCSSL Date. Furthermore SCSSL is responsible for:

1. Sanatization of different date formats: Whether DD/MM/YYYY or DD-MM-YYYY or DD.MM.YYYY the SCSSL makes sure that these can be converted from and to objects of Javascript type “date”.
2. Internationalization - These date transformations should be able to take into account MM-DD-YYYY and DD-MM-YYYY depending on the international setting that a user has set in the front end.

3. Natural language dates - Initial handling of natural language dates i.e. September 21st, 21 Sept should also be supported. The current year will be used if the year is not supplied.

The default format for dates in the spreadsheet should be MM/DD/YYYY.

4.2.3. SCSSL Enum - Enumerable Support Functions

The SocioCortex attributes of type “enumerable” should be able to take on values of a predefined list. These can be used to show status indicators for entities for instance. Support for these on the Spreadsheet 2.0 side is supported by the SCSSL Enum Controller. Not only should the SCSSL Enum functions support the UI elements with the list of possible defined values, but they should also provide validation support. The purpose of these functions should be to initially bind the model of enumerated attribute to the table cell.

4.2.4. SCSSL Rich String - WYSIWYG Transformations

WYSIWYG transformations should be similar to text transformations. The main purpose of these transformations should be to prevent security concerns and erroneous input that is parsed to the SocioCortex. These transformations will sanitize input received from a WYSIWYG editor. A further issue that exists is the one of incorrectly closed code tags which will also be handled by these functions.

4.2.5. SCSSL Derived Attribute Support Functions

In SocialCortex one can define derived attributes. Derived attributes are useful as they can hold aggregated information and formulas defined in the the DSL of the SocioCortex using a language called MxL. Currently it is planned the SCSSL supports MxL 2.0. The “Derived Attribute” support functions should assist in creating new columns from MxL queries.

4.2.6. SCSSL Text - Text Countdown / Validation Function

As the text field from the SocioCortex has a limited amount of characters it is important to have a function that checks the character limit of a plain text field before sending it to SocioCortex. SCSSL text functions assure us that text is correctly sanitized before being submitted to SocioCortex.

4.2.7. SCSSL Referenced Entity Support Functions

Entities should be able to reference each other in SocioCortex by utilizing attributes of type “reference”. In the sheet these values should be displayed wrapped with “{}” separators and be linked via a hot key to the page of the entity. The entity reference functions should support the user by querying the SocioCortex REST API and displaying the results via the UI widget. Furthermore, these support functions should validate incorrect input and prevent users from referencing incorrect content.

4.2.8. SCSSL No Type Support Functions

A SocioCortex “No Type” attribute can represent anything from date, number, text to enumerated and referenced.

The no type support functions are concerned with detecting the type of a cell input based on a cell analysis. If one for instance enters 12 - this should be detected as a number. However, if one then proceeds to enter 12-12-2015 - this should then be detected as a “date”. Based on the detected type, the Object sent to the SocioCortex should have the correct attached type and the correct widget. Moreover, it should display the second time a user tries to edit the value.

As the type detection of the No Type support works similar to a programming language “switch” statement, the order is important. The following order applies to the type detection (1 = high priority, 6 = low priority):

1. Enum Detection
2. Referenced Entity Detection
3. Date Detection
4. Number Detection

4. Conceptual Design

5. Rich String Detection

6. Text Detection

Furthermore, No Type support should integrate with many of the above functions and SCSSL UI elements to offer extended hybrid cell editing. If one for instance enters a date, this should get converted to the correct format by making use of the SCSSL date transformation functions.

4.3. SCSSL UI

Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Text)	Unit price (Number)	Notes (No Type)
2-XMC-DD	25	26.03.2015	Open	{4-XMMP};{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[4XMMP][4X][4M]	[8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed		[3P]	[9][18][47][8]	2.33	[22][23][24]
4-AMC	10	08.03.2015	Open	{1-YMC-XC}	[4A][4M]	[1][23][22]	1.40	{2-XMC-DD};{3MC-DD}
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.3.: High-fidelity mockup of the a Spreadsheet 2.0 table.

The next section is mainly concerned with the UI Widgets that show up when editing content in the sheet.

Depending on the type of column and the type of field different widgets may show up. If, for example, an attribute had initially the “no type” type and was populated with data - the type of data entered (and corresponding widget shown) should be determined and set by the SCSSL UI functions. However, if the attribute then changed from “no type” to date - the UI should force a date picker to be shown whenever editing any cell content.

The decision making process to determine which widget to display is shown in Figure 4.3. If the schema is of the type “no type”, then the text in the cell should be analyzed and a widget displayed based on the result of the interpretation. If a schema type does already exist and is not equal to the cell type then it should be attempted to converted the value of the cell to the parent schema type. Should this not be possible, then the value of the

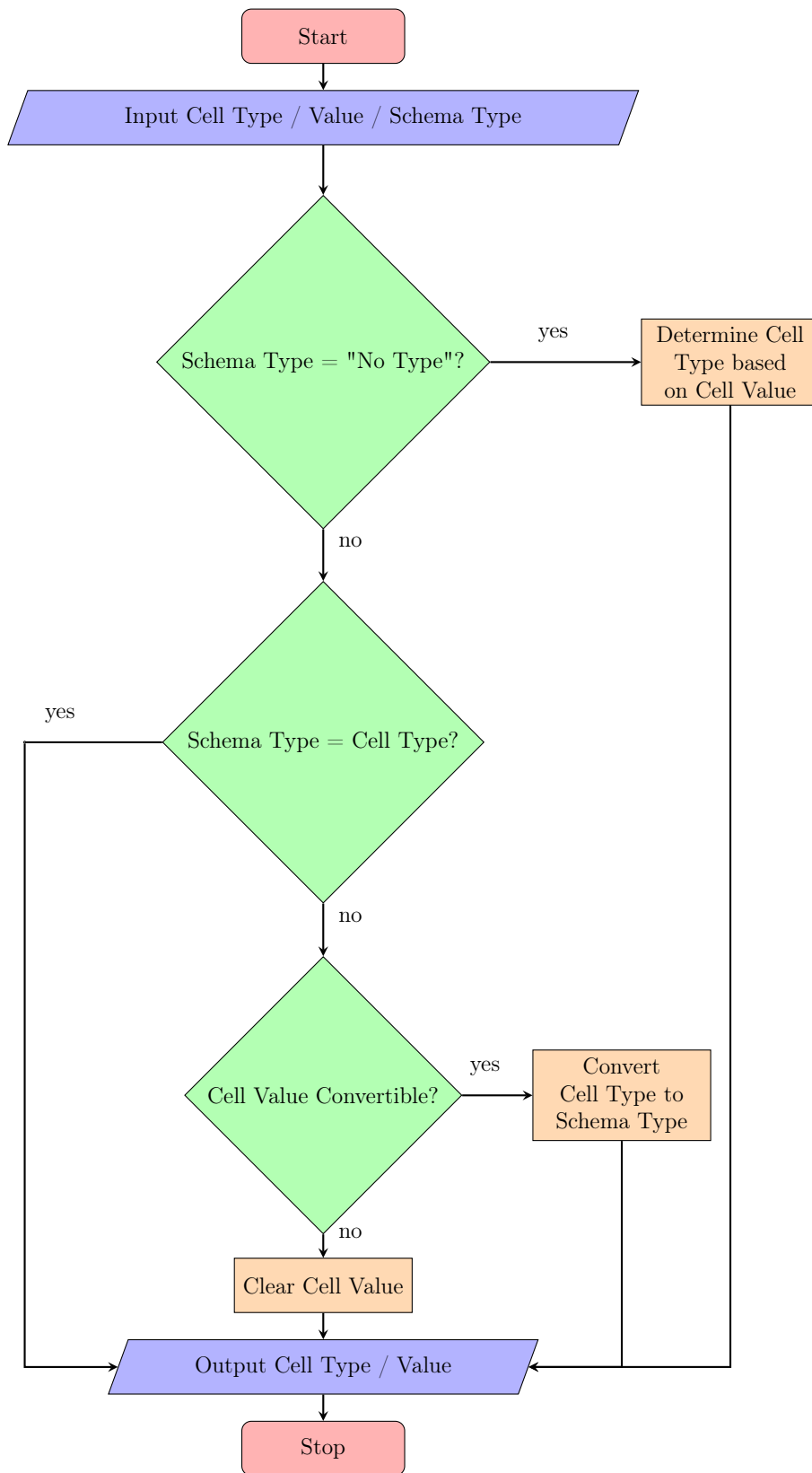


Figure 4.4.: Type Conversion & Widget Display Flowchart

4. Conceptual Design

cell should be cleared and the widget for the schema shown. For the best user experience, the system should still display the initial value somewhere when editing.

In the following we further explore different UI Widgets (specifically Number, Date, Enum, Reference & No Type). We also provide mockups for the individual widgets which based on the table found in Figure 4.4. It depicts a typical order/quantity spreadsheet that one might find in the industry. It does however also feature linked products that are displayed in the “Referenced Products” column.

4.3.1. Number UI

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[][4XMMP][4X][4M]	8[22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed		[][3P]	[][9][18][47][8]	2.33	Curabitur nibh...
4-AMC	10	08.03.2015	Open	{1-YMC-XC}	[][4A][4M]	[][1][23][22]	1.40	Praesent sed...
1-YMC-XC	10	29.11.2015	Open		[][1Y]	[][23][40]	2.35	Etiam urna velit...

Figure 4.5.: Low-fidelity wireframes of a number stepper widget (Number UI)

The Number UI should support users with entering integers and float based numbers into the table. Instead of allowing user to directly change a cells value by typing in a value, the Number UI should provide the user with the option of increasing or decreasing a number value by pressing buttons which increase or decrease the value. Figure 4.6 depicts a number stepper widget that allows users to increase and decrease the number based on which arrow is clicked. I.e. If the arrow pointing up is clicked, the number will increase and vice versa with the arrow pointing down.

The Number UI widget also filters for input and only allows numbers as input.

Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Text)	Unit price (Number)	Notes (No Type)
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[4XMMP][4X][4M]	[8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed		[3P]	[9][18][47][8]	2.33	[22][23][24]
4-AMC	10	08.03.2015	Open	{1-YMC-XC}	[4A][4M]	[1][23][22]	1.40	{2-XMC-DD}{3MC-DD}
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.6.: High-fidelity mockup of the number widget

In the case of an attribute allowing multiple numbers, a multiplicity widget should be shown with multiple number steppers as indicated in Figure 4.7. The SCSSL Multiplicity Handler and the UI that it offers is discussed in chapter 4.8.

4. Conceptual Design

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[][4XMMP][4X][4M]			Donec sit...
2-XMC-3P	44	07.07.2015	Closed		[][3P]			Curabitur nibh...
4-AMC	10	08.03.2015	Open	{1-YMC-XC}	[][4A][4M]			Praesent sed...
1-YMC-XC	10	29.11.2015	Open		[][1Y]			Etiam urna velit...

8	-	+
22	-	+
	-	+

Figure 4.7.: Low-fidelity wireframes of a number stepper multiplicity widget (Number UI)

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[][4XMMP][4X][4M]	8[22]	1.25	Donec sit...
2-XMC-3P	44	12/10/2015	Closed		[][3P]	[][9][18][47][8]	2.33	Curabitur nibh...
4-AMC	10			{1-YMC-XC}	[][4A][4M]	[][1][23][22]	1.40	Praesent sed...
1-YMC-XC	10				[][1Y]	[][23][40]	2.35	Etiam urna velit...

◀ OCTOBER 2015 ▶

S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Figure 4.8.: Low-fidelity wireframe of a date picker widget (Date UI)

4.3.2. Date UI

One of the many problems that spreadsheets have is the ability to support users with different input dates. Many different date formats and notations lead to wrong inputs and erroneous data. Even though these are being converted to a certain extent by the transformation functions to and from date objects, in the front end, there is also a need to assist the user when entering the correct date formats. In SCSSL, we propose to solve this problem by making use of input assistance in the form of a date picker. Date pickers are widgets that pop-up with a calendar from which a user can then select a date.

A rudimentary date picker concept is shown in Figure 4.8. As seen in this figure, the user can alternate between the months by making use of the arrows at the top next to the date. At the same time the days are shown from which the user can make a selection. Upon selecting the day the widget will disappear and cell below that hold the value of the selected date.

A high-fidelity version of the Date UI is show in Figure 4.9. The date picker here shows up upon being selected by the cell which is a drop-down.

Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Num.)	Unit price (Number)	Notes (No Type)																																										
2-XMC-DD	25	26.03.2015	Open	{4-XMMP};{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]																																										
4-XMMP	50	<div style="border: 1px solid gray; padding: 5px;"> <div style="display: flex; justify-content: space-between; align-items: center;"> < March 2015 > </div> <table border="1" style="width: 100%; text-align: center; font-size: 0.8em;"> <tr> <td>Su</td><td>Mo</td><td>Tu</td><td>We</td><td>Th</td><td>Fr</td><td>Sa</td> </tr> <tr> <td>26</td><td>29</td><td>30</td><td>01</td><td>02</td><td>03</td><td>04</td> </tr> <tr> <td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td> </tr> <tr> <td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td> </tr> <tr> <td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td> </tr> <tr> <td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>01</td> </tr> </table> </div>		Su	Mo	Tu	We	Th	Fr	Sa	26	29	30	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	01	{2-XMC-DD}	[4XMMP][4X][4M]	[8][22]	1.25	Donec sit...
Su	Mo	Tu	We	Th	Fr	Sa																																												
26	29	30	01	02	03	04																																												
05	06	07	08	09	10	11																																												
12	13	14	15	16	17	18																																												
19	20	21	22	23	24	25																																												
26	27	28	29	30	31	01																																												
2-XMC-3P	44				[3P]	[9][18][47][8]	2.33	[22][23][24]																																										
4-AMC	10			{1-YMC-XC}	[4A][4M]	[1][23][22]	1.40	{2-XMC-DD};{3MC-DD}																																										
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015																																										

Figure 4.9.: High-fidelity mockup of a date picker

4.3.3. Enum UI

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[][4XMMP][4X][4M]	8[22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed		[][3P]	[][9][18][47][8]	2.33	Curabitur nibh...
4-AMC	10	08.03.2015	Open Pending Closed	YMC-XC}	[][4A][4M]	[][1][23][22]	1.40	Praesent sed...
1-YMC-XC	10	29.11.2015	Open		[][1Y]	[][23][40]	2.35	Etiam urna velit...

Figure 4.10.: Low-fidelity wireframe of a selector widget (Enum UI)

The enumeration user interface (or Enum UI) should provide widgets that show the user with a restricted list of predefined options from which a selection should be made. This list is displayed in the form of a drop down widget. The predefined options are set when creating attributes in SocioCortex and are displayed on a column by column basis.

Figure 4.10 depicts a low-fidelity version of a drop-down list. As represented in the figure, the user should choose between “Open”, “Pending” and “Closed”. A user is also able to directly type in a value into the field which will then be filtered by the Enum UI to only contain the specified values.

Figure 4.11 is an improved version of the drop down designed based on material design. The drop down list shown has all the available options and highlights the current selected option in blue.

Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Num.)	Unit price (Number)	Notes (No Type)
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[4XMMP][4X][4M]	[8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Open Pending Closed		[3P]	[9][18][47][8]	2.33	[22][23][24]
4-AMC	10	08.03.2015		{1-YMC-XC}	[4A][4M]	[1][23][22]	1.40	{2-XMC-DD}{3MC-DD}
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.11.: High-fidelity mockup of a selector widget (Enum UI)

4.3.4. Entity Reference UI

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	2-XMC-DD	[][4XMMP][4X][4M]	[][8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed	2-XMC-3P	[][3P]	[][9][18][47][8]	2.33	Curabitur nibh...
4-AMC	10	08.03.2015	Open	4-AMC	[][4A][4M]	[][1][23][22]	1.40	Praesent sed...
1-YMC-XC	10	29.11.2015	Open	4-XMMP	[][1Y]	[][23][40]	2.35	Etiam urna velit...

Figure 4.12.: Low-fidelity wireframe of the entity reference UI.

As mentioned previously, it is possible to reference other entities in SocioCortex by using attributes which are of type “reference”. These references to linked data should make use of an auto-complete widget to allow the user to quickly find the referenced entity. The auto-complete widget should query the SocioCortex REST API via the SCSSL Referenced Entity Support Functions. SocioCortex would return a list of possible results for the user. If a user then selects a specified result, the entity ID of that result is saved in the cell. Figure 4.12 demonstrates what this should look like when the input cell is highlighted and presents a range of drop-down options sorted alphabetically.

A field containing multiple should display a list of auto-complete fields for users to select their referenced entities. Every referenced entity is linked as indicated in Figure 4.13 by the different font color. Therefore, once clicked (in conjunction with a hot-key), the link should open a page for the linked entity. Selecting the cell without any hot-key should open up the auto-complete editor or the multiplicity widget with the multiple auto-complete fields.

Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Num.)	Unit price (Number)	Notes (No Type)
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
4-XMMP	50	12.04.2015	Pending	2-XMC-DD	[4XMMP][4X][4M]	[8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed	2-XMC-3P 2-XYCC-5PU	[3P]	[9][18][47][8]	2.33	[22][23][24]
4-AMC	10	08.03.2015	Open		[4A][4M]	[1][23][22]	1.40	{2-XMC-DD}{3MC-DD}
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.13.: High-fidelity wireframe of the entity reference UI.

4.3.5. No Type UI

Figure 4.3 demonstrates what a column of “No Type” could resemble. As seen by the data entered into the last column, it could consist of any nature. Populated data can consist of a range of dates, numbers, text to references. With the No Type UI we set out to support the complex issue of supporting different data on a cell by cell basis.

4.3.6. Derived Attribute UI

In Figure 4.14 we see the process of a derived attribute being added to the table. The process is being handled by the Derived Attribute UI and should allow the user to open up the “Add derived attribute” dialog by clicking on the (+) sign. The user is then prompted to enter an MxL query which could then already be validated in the dialog box. If the query is valid and the user has entered a column name, the “Add” option will then be activated for the user to click. If the user proceeds a new read-only column will be added to the table with the result of the users query for every entity object.

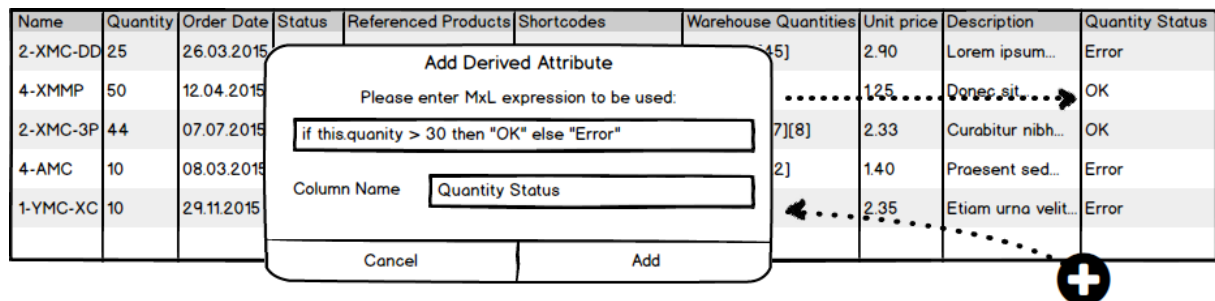


Figure 4.14.: Low-Fidelity Mockup of the MxL derived attribute column

4.3.7. Rich String UI (WYSIWYG), Image UI & Text UI

This thesis mainly focuses on the rendering of numbers, dates, enumerables and referenced linked data in a spreadsheet scenario and therefore will only highlight the functionality of the Rich String UI (WYSIWYG), Image UI and the Text UI. It will however not be concerned about the implementation of these as these either do not contribute to solving the research problem or they fall outside the scope of implementation. They are however mentioned for purposes of completeness and can be extended with future research.

4.4. SCSSL Multiplicity Handler

As SocioCortex attributes are not only single value but also allow for for multiplicity we have to add support for multiple entries in the SCSSL framework. This is handled by the multiplicity handler which consist of a transformation function and a UI handler.

4.4.1. Multiplicity Transformation Function

The initial functionality of the multiplicity handler is to allow for users to efficiently add, remove and edit multiple values. As the display of the spreadsheet is of a one dimensional nature (flat data format), it is necessary to have a transformation function which is responsible for mapping single value cells to multiple objects. This would allow for users, without the use of widgets, to navigate through the spreadsheet and change values at an individual cell level. It also handles the display of the cell values.

Conceptually proposed is using principles of flat data formats (CSV or tab separated) to map cell values to multiple values. For this we introduce the “[“ and “]” separator. Note: This separator should changeable if it does conflict with actual data. By making use of this seperator a multiple date object such as:

```
1 {  
2   1: Object  
3     name: "MultiDate"  
4     type: "date"  
5     values: Array[3]  
6       0: "10.07.2015"  
7       1: "31.07.2015"  
8       2: "02.06.2015"  
9 }
```

would be represented as [10.07.2015][31.07.2015][02.06.2015] in the cell of a table. Similarly numbers (e.g. [1,56][2,44][5,65]) or any other attribute types (eg. [text1][text2][text3]) would also be separated by the separator.

The purpose of the transformation function is to “watch” the table array for changes. Upon the table array changing it should update the original object and submit the changes to SocioCortex. In the example above if the cell containing “[10.07.2015][31.07.2015][02.06.2015]” changes the middle date to “[10.07.2015][31.07.2016][02.06.2015]” or (even removes a date

such as “[10.07.2015][31.07.2015]”) - these changes would reflect on the main object through the transformation function which would then submit the data.

4.4.2. Multiplicity User Interface

Name	Quantity	Order Date	Status	Referenced Products	Shortcodes	Warehouse Quantities	Unit price	Description
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[][2XMC][2X][2M]	[][1][20][45]	2.90	Lorem ipsum...
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}			1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed				2.33	Curabitur nibh...
4-AMC	10	08.03.2015	Open	{1-YMC-XC}			1.40	Praesent sed...
1-YMC-XC	10	29.11.2015	Open				2.35	Etiam urna velit...

Figure 4.15.: Low-fidelity wireframe of the multiplicity widget

The user interface component of the SCSSL Multiplicity Handler conceptually should make use of the drop-down list paradigm with input boxes. On the right of every row (-) signs could be placed to accommodate users with removing a value. In addition to the (-) signs, a (+) sign at the end of the list is envisioned to allow users to add more rows. A mockup can be seen in Figure 4.15.

The SCSSL Multiplicity Handler UI is responsible for displaying a UI widget that uses the flat data cell value to improve the user experience when using the sheet.

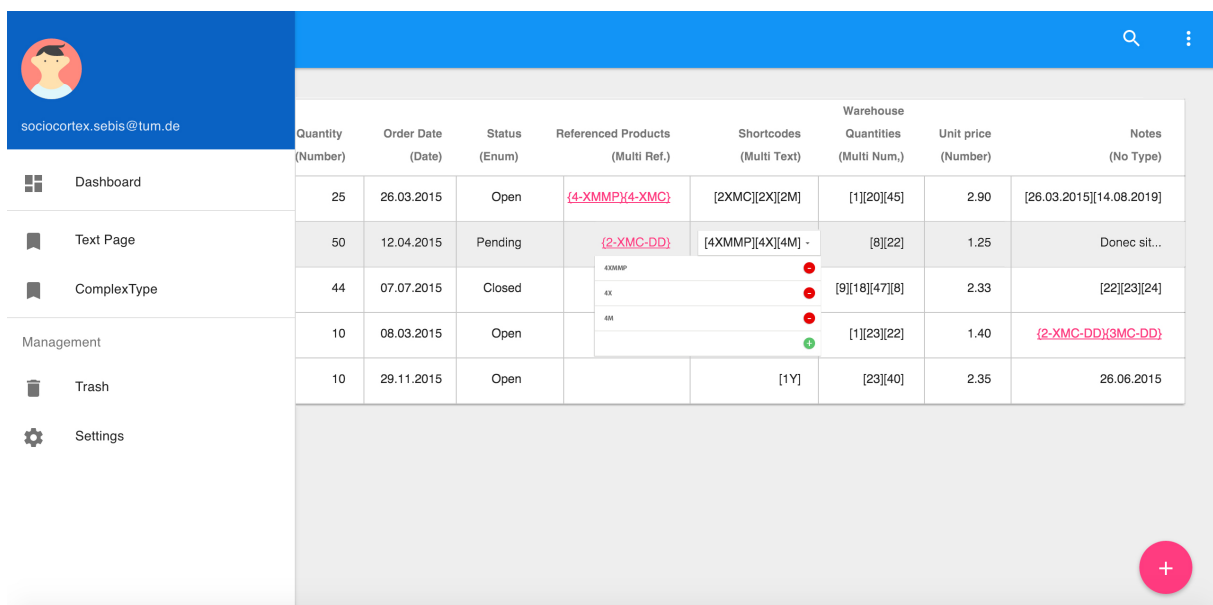
Name (Text)	Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Num.)	Unit price (Number)	Notes (No Type)
2-XMC-DD	25	26.03.2015	Open	{4-XMMP}{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
4-XMMP	50	12.04.2015	Pending	{2-XMC-DD}	[4XMMP][4X][4M] -	[8][22]	1.25	Donec sit...
2-XMC-3P	44	07.07.2015	Closed			[9][18][47][8]	2.33	[22][23][24]
4-AMC	10	08.03.2015	Open			[1][23][22]	1.40	{2-XMC-DD}{3MC-DD}
1-YMC-XC	10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.16.: Multiple Values Mockup

4.5. SCSSL UI Overall Mockup

Lastly for a better overall understanding of the implementation of the SC Sheet: Figure 4.17

This figure shows a high fidelity mockup of SCSSL when implemented with a user interface like Angular Material. The process for opening a table would involve selecting the type to be edited from the left menu. This would then load the table in the background.



The mockup shows a user interface with a blue header bar containing a search icon and a menu icon. On the left, there is a sidebar menu with a user profile at the top (sociocortex.sebis@tum.de) and several menu items: Dashboard, Text Page, ComplexType, Management, Trash, and Settings. The main area displays a table with the following data:

Quantity (Number)	Order Date (Date)	Status (Enum)	Referenced Products (Multi Ref.)	Shortcodes (Multi Text)	Warehouse Quantities (Multi Num.)	Unit price (Number)	Notes (No Type)
25	26.03.2015	Open	{4-XMMP};{4-XMC}	[2XMC][2X][2M]	[1][20][45]	2.90	[26.03.2015][14.08.2019]
50	12.04.2015	Pending	{2-XMC-DD}	[4XMMP][4X][4M] -	[8][22]	1.25	Donec sit...
44	07.07.2015	Closed	4XMMP 4X 4M		[9][18][47][8]	2.33	[22][23][24]
10	08.03.2015	Open			[1][23][22]	1.40	{2-XMC-DD};{3MC-DD}
10	29.11.2015	Open		[1Y]	[23][40]	2.35	26.06.2015

Figure 4.17.: High Fidelity Mockup of SCSSL UI Overall Mockup

Part III.

Software Design and Implementation

5. Software design

In the following Chapter, the approach that was taken to implement the proposed concept from Chapter 4, will be discussed. As the SocioCortex makes use of a REST based API, developing a JavaScript client-side application seemed like the most suitable approach to a solution. This solution was to be implemented in JavaScript by making use of existing web patterns and frameworks like AngularJS and Wijmo and design patterns like Google Material design. Google Material design is implemented for AngularJS by the AngularJS Material Library.

5.1. AngularJS Framework

AngularJS is a modern JavaScript Framework which is based on the concepts of MVC, however it is closer to an MVVM (Model-view Model-view) Architecture. This framework was chosen for the first prototypical implementation as it assists developers to rapidly develop state of the art front end applications by supplying:

1. Two way data binding
2. Templates
3. Dependency injection
4. Directives
5. Services

AngularJS will mainly assist on the client side with handling data from REST APIs (in our case SocioCortex API) and rendering data on the client side. This will be needed when implementing the SCSS UI to render widgets to collect input data.

As opposed to other frameworks like Cappuccino and Knockout, Angular is maintained by Google and its engineers[22]. This assures the framework to be of a certain code quality with projected long term support. It also ensures its adoption among developer

communities. Features [22] like those listed above make it the current cutting edge technology to make use of when designing a complex type spreadsheet. In conjunction with the spreadsheet, these features help with the display, rendering and mapping of data from SocioCortex to a spreadsheet framework.

5.2. Spreadsheet Framework

From the outset it was important to develop a solution which would not only support AngularJS and be browser based, but also be intuitive. Intuitive meaning that as we are replicating a spreadsheet system, spreadsheet functionality should exist that users are used to. This functionality is reflected in the spreadsheet use of keys (i.e. a spreadsheet should not only be navigable by clicking but also by making use of “up”, “down”, “left” and “right” arrows. Furthermore users are used to being able to drag and drop columns and rows; as well as sort columns which is one of the features that the spreadsheet framework should not lack. In addition to the normal features of a spreadsheet it should also support and display information in a new way and handle the SCSS UI widgets. To summarize - the solution should supply the following:

1. Direct cell editing
2. Navigation and edit support by the use of keyboard keys
3. Custom fields - Ability to display various content
4. Allow for the use of data-bindings

Initially the approach of developing a spreadsheet grid was tested. This consisted of a table being generated with input fields which were repeated with the ng-repeat directive in AngularJS. This construct did initially work, yet creating many bindings for every cell (input box) started to slow the program down and another solution was investigated. Trying to use, because of speed considerations, D3¹⁹ as the rendering engine for fields was a better solution. However, developing the entire spreadsheet logic and making the spreadsheet match the capabilities of real spreadsheet software seemed problematic, as the wheel would have to be reinvented which would involve many more hours of development. Therefore, it was decided to look at frameworks already available which could offer such functionality.

¹⁹<http://d3js.org/>

Non AngularJS frameworks were tested like SlickGrid²⁰, jqGrid²¹, BackgridJS²² and ClusteriseJS²³. The idea here was to use these on top of the AngularJS framework.

A collection of AngularJS spreadsheets frameworks were also investigated. Amongst other the following are quite notable: UI Grid²⁴, Wijmo FlexGrid²⁵, ngTable²⁶ and Smart table²⁷. Figure 5.1 shows a comparison between all of these frameworks.

Eventually the Wijmo framework was chosen as the framework of choice to use as the Spreadsheet 2.0 framework that would interact with SCSS. Not only did this framework fulfill all the conditions above but also catered for the use of widgets and provided copy paste from excel support.

5.2.1. The Wijmo Framework

The Wijmo5 is a package of over 40 UI JavaScript controls that are developed in HTML5 and support AngularJS. The company behind Wijmo5, GrapeCity, have been developing Wijmo for jQuery since 2010²⁸. The Wijmo package consists of 4 components: Core, DataGrid, Input and Data Visualization which contain the following²⁹:

Core Base Control, Globalization, CollectionView, Themes, Clipboard, Events and Validation

Data Grid FlexGrid and FlexGrid Extensions (Filter, GroupPanel, DetailRow)

Input ListBox, ComboBox, AutoComplete, Menu, Calendar, InputDate, InputTime, InputNumber, InputMask, ColorPicker and InputColor

Data Visualisation FlexChart, FlexPie LinearGauge, RadialGauge, BulletGraph and FlexChart Extensions (RangeSelector, Analytics, Annotations)

For the spreadsheet framework we made use of the core components for the data bindings, the DataGrid for the FlexGrid (which is our base spreadsheet control) and the input

²⁰<https://github.com/mleibman/SlickGrid>

²¹<http://www.guriddo.net/demo/guriddojs/>

²²<http://backgridjs.com/>

²³<http://nexts.github.io/Clusterize.js/>

²⁴<http://js-spreadsheets.com/ui-grid.html>

²⁵<http://wijmo.com/5/docs/topic/wijmo.grid.FlexGrid.Class.html>

²⁶<http://ng-table.com/>

²⁷<https://github.com/lorenzofox3/Smart-Table>

²⁸<http://wijmo.com/welcome-to-wijmo-5/>

²⁹<http://wijmo.com/products/wijmo-5/>

Framework	Version	Direct Cell Editing	Keyboard Navigation Supp.	Custom Fields	Widgets Supp.	AngularJS & Data Binding Supp.	Source	Excel Copy Paste Support	Updated	Notes
Slick Grid	?	Yes	Yes	No	No	No	Open	No	05-03-2014	Provides "MS Excel like" look and feel.
jqGrid	5	No	No	No	No	No	Open	No	03-08-2015	Free for non commercial purposes.
BackgridJS	0.3.5	Yes	No	No	No	No	Open	No	21-01-2014	Datagrid with "edit in place" cells.
ClusteriseJS	0.12.0	No	No	No	No	No	Open	No	14-08-2015	Optimized for large data sets.
ag-Grid	2.2.0	Yes	Yes	Yes	No	Yes	Open	No	03-10-2015	
Wijmo Flexgrid	5.201	Yes	Yes	Yes	Yes	Yes	Closed	Yes	28-08-2015	JavaScript based framework with AngularJS wrapper. Very good support.
ui-grid	3.0.6	Yes	No	Yes	Yes	Yes	Open	No	03-10-2015	Swaps out cell input boxes and widgets as cell receives focus. Uses no jQuery.
ngTable	1.0.0-beta.6	No	No	No	No	Yes	Open	No	03-10-2015	Limited support.
Smart table	2.1.3	No	No	No	No	Yes	Open	No	10-09-2015	

Table 5.1.: Comparison of different spreadsheet and grid tables

component for making use of some of the Wijmo widgets such as the InputNumber, InputDate and ListBox.

5.2.2. The Flexgrid from the Wijmo Framework

The FlexGrid forms part of Wijmo Data Grid component. It is a full-featured grid which supplies users with all the functionality and user experience found in the usual spreadsheet application. One is even able to copy and paste directly from Microsoft Excel into Wijmo FlexGrid.

5.2.3. SC Angular Library in Detail

Initially, in the early versions of the prototype, all of the connection handling was done as an SCSSSL utility function which made use of Angular `$http.get` and other functions for connection and data request handling. The issue however was that the endpoints and responses of SocioCortex seemed to be limited which would need to either be re-parsed or sanitized for use. Other issues were that multiple requests needed to be made for the querying of multiple data results.

Patrick Bürkin³⁰, then for his Thesis: “Design and Prototypical Implementation of a Dashboard System for Visualizing Semi-Structured Data in a Traceable Way” created the SC Angular library which dealt with many of the issues (e.g., connection management, wrapping and unwrapping of responses) encountered whilst developing for SocioCortex. Not only was this library of a superior quality to the previously implemented prototype SCSSSL solutions since it made use of testing frameworks such as Jasmine³¹ which help with the establishment of finding out when the SocioCortex API changes break an application; but Patrick Bürkin was also a month ahead with his implementation which meant that some of the functions needed in future for the SCSSSL had already been implemented. Also, due to its cleanly written nature, adding functionality to SC Angular is effortless. This led to the adoption to the SC Angular (v0.6.3) library as the communication library (layer) between SCSSSL and SocioCortex. (As seen in Chapter 4, Figure 4.1)

SC Angular comes with three services: `scCore`, `scCrud`, `scMxl` and `scUtils` which are further elaborated on as they are required by SCSSSL.

³⁰<https://www.matthes.in.tum.de/pages/16qwn6mc8kei2/Master-s-Thesis-Patrick-Buerkin>

³¹<http://jasmine.github.io/>

scCore This is the connection handler for SocioCortex. It deals with authentication, request and URL handling.

scCrud This is responsible for CRUD operations on the SocioCortex REST API. This allows users to find, create, update and remove as well as supplying users with the crud functions for entities, groups, users, type and workspaces.

scMXL This service is responsible for handling all MxL related communication.

scUtils This deals with wrapping and unwrapping objects as well as recursive requests.

SCSSL mainly makes use of scCore for the connection establishment; scCrud for fetching workspaces, types, attributes and entities from SocioCortex; and scUtils for its object handling functionality.

5.3. The SC Sheet Directive

Initially in the first prototype it was attempted to load SCSSL UI Widgets which were implemented on top of a Wijmo FlexGrid directive. The Flexgrid directive would allow for cell types and the data source to be defined. It would then bind the Flexgrid to the data and render the result. This is an example of the implementation code:

```
1 <!-- Example of Flexgrid implmentation with a binding on "data" -->
2 <wj-flex-grid items-source="data"></wj-flex-grid>
```

This meant that the SCSSL UI would need to load on top of the Flexgrid. This limited the functionality because the directive options and controller functions were defined by the Wijmo directive. Furthermore, it was experienced that the ability to add columns could only be implemented from the AngularJS controller, as recreating the grid would only be possible from the controller. This ability was needed by SCSSL to implement adding additional attribute types.

The solution to this problem was the SocioCortex Sheet (abbr. “scsheet”) directive. With scsheet all the functionality was moved into the controller which resulted in more control and better modularisation.

```
1 <!-- Example of of the scsheet directive -->
2 <scsheet ng-typeid="1clqy1q3yk2v3" ng-use-url="true" ng-mode="widget" ng-show-
  inconsistencies="true" ng-editable="true"></scsheet>
```

Table 5.2 further elaborates on the input parameters for the SC Sheet Directive.

Parameter	Required Input Parameter	Default Value	Property Description
ng-typeid	1clqy1q3yk2v3 (Type ID)	Required if URL is not set	Specifies the type identifier that should be used to load all the content entities from SocioCortex into the FlexGrid.
ng-use-url	true/false (Type ID parameter via URL)		This option allows for the passing of the last URL parameter to the grid. (ex. <i>http://127.0.0.1:54251/index.htm#/sheet/types/1clqy1q3yk2v3</i> uses <i>1clqy1q3yk2v3</i> as type identifier as it is the last parameter)
ng-mode	rawtext/widget	rawtext	This property allows for defining the table editing mode. “rawtext” will only make of the transformation functions whilst “widget” would show the SCSSL UI.
ng-show-inconsistencies	true/false	false	This property allows for setting of displaying inconsistency of types.
ng-editable	true/false	true	This property allows for one to make the SC Sheet editable or not.
ng-add-new-row	true/false	false	This property grant the ability to add new rows.
ng-allow-sort	true/false	false	This allows for sorting of table values.
ng-group-values	true/false	false	This allows for the grouping of similar cell values.

Table 5.2.: Parameters for the SC Sheet Directive

It should lastly be noted that the SCSSL, specifically SC Sheet requires SC Angular, Wijmo and Wijmo Angular to function.

5.4. Bootstrap Loading Process

To initially generate the SC Sheet with entities from the request type, involves a Bootstrap loading process. This process is show in Figure 5.1. We start with the type id as the first input requirement. This parameter is then passed to the *scInitialise* function which then triggers the following bootstrap process:

1. First all the `$scope` variables and objects are initialized (existing ones destroyed and recreated) with the *scInitialScope* function. This also destroys current controls that might exist as well as a Wijmo Flexgrid instance.
2. Then the *scGetColumns* function fetches the schema structure of the current type. This also determines the columns of the Wijmo Grid (e.g., type of column for every attribute) and will also preset enumerable drop-down widgets if the widget mode is switched on.
3. A new *wijmo.collections.CollectionView* object is then created from the *getData* function which fetches and parses the current entities. This object is bound to `$scope.data` object.
4. The *scBuildGrid* then builds the spreadsheet Grid from the `$scope.data` object.

5.4.1. The `scBaseObject`

In the *scGetColumns* function the schema of the current type is not only received and stored for the column definitions, but also for the `scBaseObject`. The `$scope.scBaseObject` is an object that reflects the schema for the type. One could look at it as an empty entity for the type. All entities should have this schema. It is then used as an initialization template object in the creation of `$scope.data` for all its entities as shown in the Figure 5.2. This is needed as SocioCortex does not send empty objects which causes Wijmo not to enable the empty cells for editing.

5.4.2. Multiplicity Loading

During the *getData* loading phase if an entity does have multiple objects for an attribute it will parse these objects to the *scsslMultiCtrl.serializeData* function. This will then serialize them into a string and add them to the `$scope.data` object. In the implementation the delimiter has changed from the initial double “[“]” notation to a single “[|” notation.

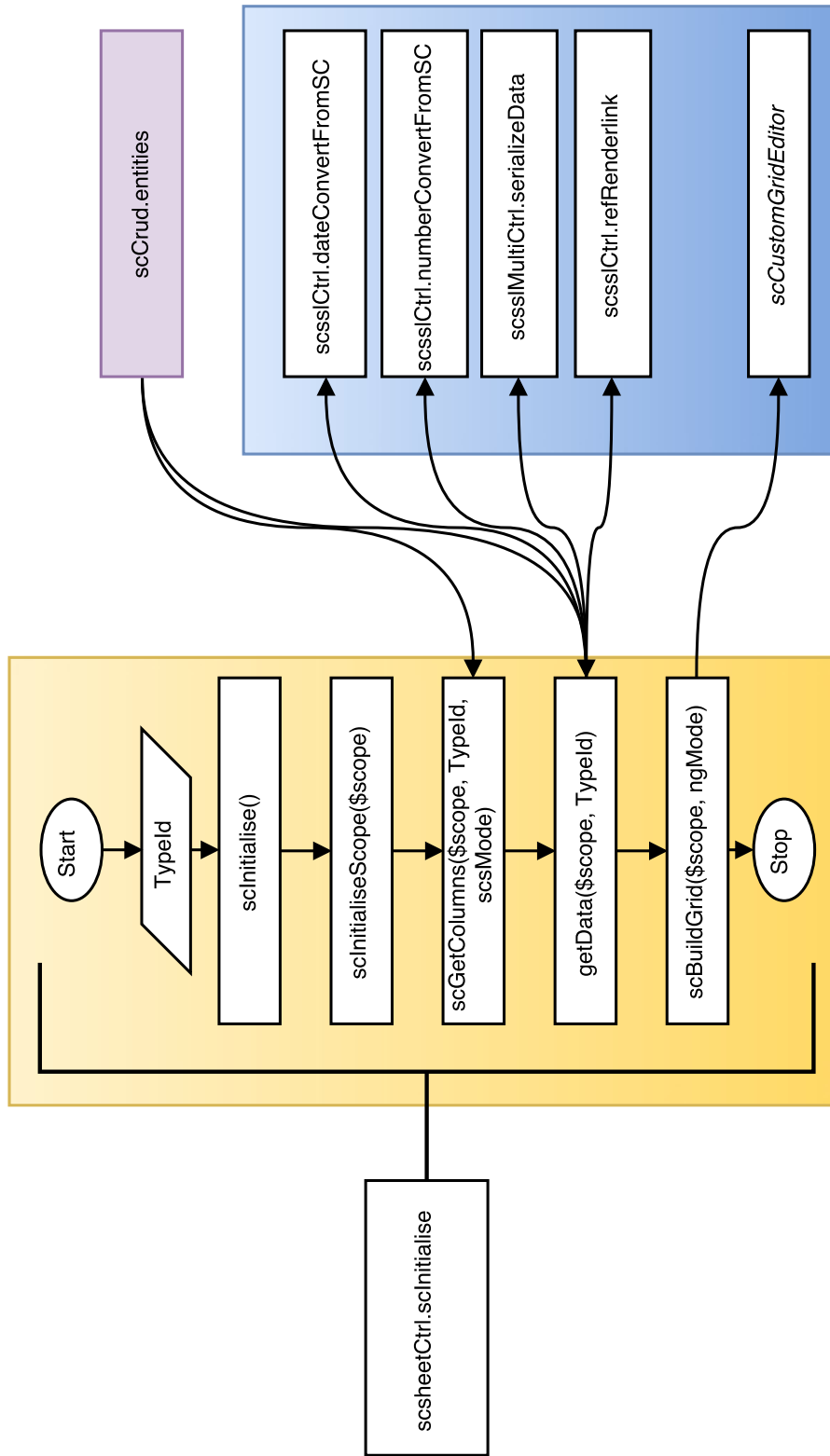


Figure 5.1.: Bootstrap Loading Process

```
1 {  
2   scBaseObject: Object  
3     sName: ""  
4     sFullName: ""  
5     nTel: ""  
6     bDay: ""  
7     rDay: ""  
8     sEmail: ""  
9     eEstatus: ""  
10    uid: ""  
11 }
```

Figure 5.2.: Example of \$scope.scBaseObject

5.4.3. Linked Entity Data

Also during the *getData* loading phase the linked data fields (or of type “reference”) are formatted. In this process the fields are reformatted to render links that open the entity that they reference.

5.5. scEntityData, scAttributes and scSet Objects

As Wijmo does change the structure of the \$scope.data when it initializes a table on it, a second object called the *\$scope.scEntityData* object stores the response of entity data that comes from SocioCortex. This allows for a clean storage of the data which is used for certain functions. Similarly the attributes and their properties are stored in the \$scope.scAttributes object.

The \$scope.scset (see Figure 5.3 for an example) is the object which is used as status indications. The bootstrap process for instance returns the results from every function to the \$scope.scset object. This allows for lock mechanisms to be put in place to prevent race conditions as many of the functions make promises that will be full-filled at a later stage. Race conditions occur predominantly when fetching and sending requests to and from SocioCortex.

```
1 {  
2   scset: Array[0]  
3     IsColumns: true  
4     IsGridBuilt: true  
5     IsInitialised: true  
6     currentTypeID: "1clqy1q3yk2v3"  
7     getData: true  
8     length: 0  
9     numColumns: 11  
10 }
```

Figure 5.3.: Example of \$scope.scset

5.6. Editing Process for Controls

Once all the data is loaded into the \$scope.data object and the spreadsheet is rendered, a user can then click around and attempt to edit one of the fields. This will then initiate the edit process.

The initial editing process is concerned with the conversion of the fields in the table from Wijmo to SocioCortex. This does not include any SCSSL UI elements being rendered (For the SCSSL UI process see: 5.7 Widget Editing Process).

When the data is loaded and the Flexgrid is rendered a handler is added to the \$scope.data object which handles events when the collection changes. This handler is referred to as the *collectionChanged* handler from Flexgrid. The use of this handler was decided due to performance issues that might have been encountered if one implemented a change post to the server on every key stroke. Hence, whenever a cell changes (leaves focus), a function is called which then sends the input to the correct transformation function. This function is called *scTransform*.

5.6.1. The Editing Process for Controls (SCSSL)

scTransform can be seen as a “router” that takes the row object (as every row is an entity which will be sent back to SocioCortex) of the currently edited row and determines the edited cell value. Then it will go through the following Steps before passing the change onto SC Angular to send to SocioCortex REST API (see Figure 5.2):

- It will initially check if the \$scope.data object has changed by comparing it to \$scope.scEntityData

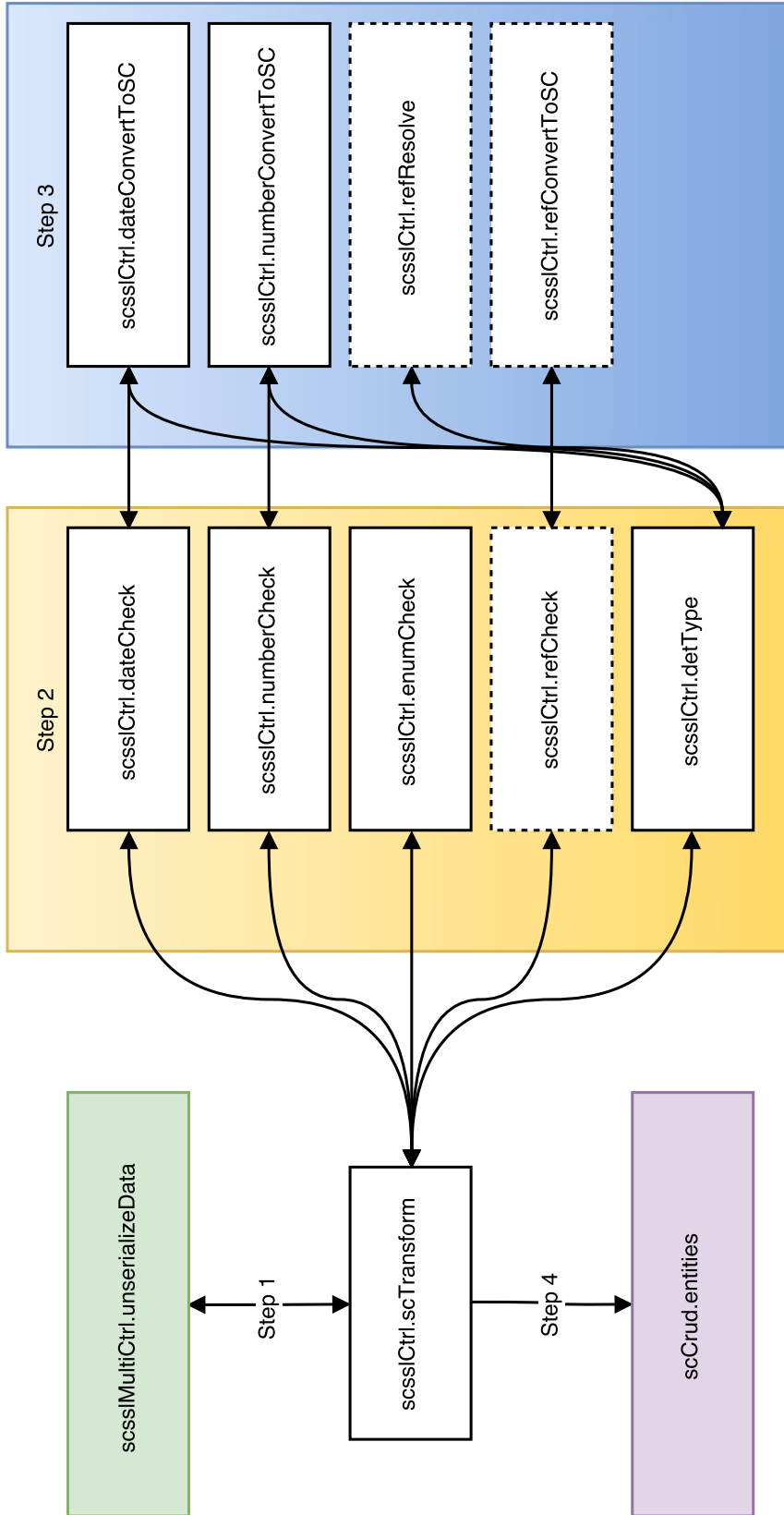


Figure 5.4.: Steps involved in the editing process.

- If it has: It will pass the currently edited cell value to the multiplicity controller function *scsslMultiCtrl.unserializeData* which returns the cell value in the form of individual objects. (Step 1)
- Step 2: These are then passed to the correct editing process controller functions (seen in Table 5.3). These functions then perform data validation and conversions of input data. Should the validation succeed, the input will then be converted if needed before being returned to the SocioCortex. Should the value be invalid, the user will be notified and the focus of the edited cell reset.
- Step 3: If the validation succeeds, the object is then sent to the editing process conversion controller functions which will convert it into the right format if needed.
- Step 4: The returned value is then wrapped in the correct request JSON format with the correct entity structure and passed to SC Angular *scCrud.entities* to be sent to SocioCortex.

5.6.2. The No Type Process

In the case that the column of the Wijmo grid is of the type “no type”, *scTransform* will try to determine the type with the *scsslCtrl.detType* function. The returned type is then parsed to the applicable conversion functions (e.g. *dateConvertToSC*, *numberConvertToSC*) before being sent to SocioCortex.

5.6.3. The Reference Type Process

Changing references involves checking if entities referenced already exist in the SocioCortex system before allowing such a change to be posted. The functionality of *scsslCtrl.refCheck*, *scsslCtrl.refResolve* and *scsslCtrl.refConvertToSC* is described in Table 5.3, however it is in an complete state and therefore cannot be fully utilized. This is also indicated in Figure 5.4 by the dashed lines around the functions.

Function	Affects Type	Description	Not fully implemented
<code>scsslCtrl.dateCheck</code>	Date	Checks if the date is the correct format or in a format that is convertible.	
<code>scsslCtrl.numberCheck</code>	Number	Checks if the number is the correct format or in a format that is convertible.	
<code>scsslCtrl.enumCheck</code>	Enum	Checks if the value is in the list of allowed list of values.	
<code>scsslCtrl.refCheck</code>	Reference	Queries SocioCortex to check if reference exists.	*
<code>scsslCtrl.detType</code>	No Type	Returns the assumed type based on input value.	
<code>scsslCtrl.dateConvertToSC</code>	Date	Converts a multitude of different date formats.	
<code>scsslCtrl.numberConvertToSC</code>	Number	Converts a multitude of different number formats.	
<code>scsslCtrl.refResolve</code>	Reference	Performs a lookup on SocioCortex for linked data.	*
<code>scsslCtrl.refConvertToSC</code>	Reference	Converts the displayed data (e.g. “{1245t9zmnoca8}”) into SC Angular linked data format (e.g. “entities/1245t9zmnoca8”).	*

Table 5.3.: Overview of `scsslCtrl` functions

5.7. Widget Editing Process (SCSSL UI)

The widget assisted editing process occurs on top of the SCSSL implementation mentioned in Chapter 5.6. It is invoked by the `scBuildGrid` function that presets a custom widget control for every column (in Wijmo widgets are implemented on a column by column basis) based on a number of factors. Using the flowchart from Figure 4.3 for the decision making process the `scBuildGrid` creates for every column an `scCustomGridEditor` instance.

5.7.0.1. *scCustomGridEditor*

This function is the key to displaying widgets. It consists of different handlers from “keystroke” events to “focus” (getting and losing) events. These events specify what should happen to the widget when one interacts with it.

The most important aspect of the *scCustomGridEditor* is the “_beginningEdit” prototype. This prevents the loading of the Flexgrid built-in editor and replaces it with the SCSSL UI editor. We then switch between the different editors based on the column type.

5.7.1. Implementation of SCSSL Date UI

Figure 5.5 shows the implementation of the SCSSL UI widget. Here one can see how the dates are converted between date objects (e.g., 7/7/2015), which are required for the widget and SocioCortex dates (26.03.2010) are converted. The date pop-up appears below the cell allowing the user to select a date.

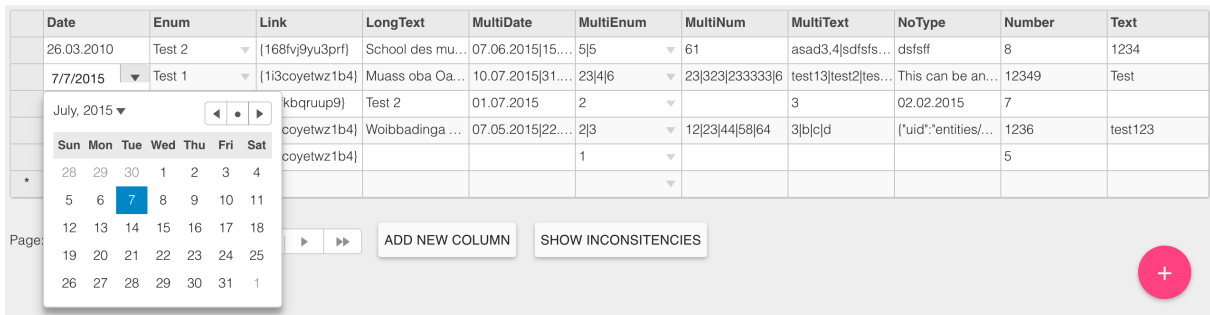


Figure 5.5.: Prototypical implementation of the SCSSL Date UI

5.7.2. Implementation of SCSSL Enum UI

Below, in Figure 5.6 one can see the implemented version of the Enum UI. This loads its values when the grid is built in the *scBuildGrid* function.

5. Software design

Date	Enum	Link	LongText	MultiDate	MultiEnum	MultiNum	MultiText	NoType	Number	Text
26.03.2010	Test 2	{168fvj9yu3prf}	School des mu...	07.06.2015 15...	5 5	61	asad3,4 sdfsfs...	dsfsff	8	1234
7/7/2015	Test 1	{13coyetwz1b4}	Muass oba Oa...	10.07.2015 31...	23 4 6	23 323 233333 6	test13 test2 tes...	This can be an...	12349	Test
07.07.2015	Test 1	{rutfkbqrup9}	Test 2	01.07.2015	2		3	02.02.2015	7	
08.07.2015	Test 2	{13coyetwz1b4}	Woibbadinga ...	07.05.2015 22...	2 3	12 23 44 58 64	3 b c d	{'uid':*entities/...	1236	test123
24.12.2015	Test 3	{13coyetwz1b4}			1				5	
*										

Figure 5.6.: Prototypical implementation of the SCSSL Enum UI

5.7.3. Implementation of the SCSSL Number UI

Figure 5.7 shows the prototypical implementation for the number widget. This currently allows only for whole numbers to be incrementally added or subtracted.

Date	Enum	Link	LongText	MultiDate	MultiEnum	MultiNum	MultiText	NoType	Number	Text
26.03.2010	Test 2	{168fvj9yu3prf}	School des mu...	07.06.2015 15...	5 5	61	asad3,4 sdfsfs...	dsfsff	8	1234
7/7/2015	Test 1	{13coyetwz1b4}	Muass oba Oa...	10.07.2015 31...	23 4 6	23 323 233333 6	test13 test2 tes...	This can be an...	12349	Test
07.07.2015	Test 2	{rutfkbqrup9}	Test 2	01.07.2015	2		3	02.02.2015	- 7.00 +	
08.07.2015	Test 3	{13coyetwz1b4}	Woibbadinga ...	07.05.2015 22...	2 3	12 23 44 58 64	3 b c d	{'uid':*entities/...	1236	test123
24.12.2015	Test 3	{13coyetwz1b4}			1				5	
*										

Figure 5.7.: Prototypical implementation of the SCSSL Enum UI

5.7.4. Implementation of other UI elements

An initial start was made with displaying multiple elements and linked data, however they proved to be more challenging than expected and therefore more time is needed to complete them in the future. The next chapter will further elaborate on the issues that were experienced.

6. Evaluation

This chapter unfortunately does not discuss the evaluation in the context of a research study or group testing. This is due to the fact that the prototype is was not completed in time for the evaluation phase.

In raw text mode, the prototype fetches data from SocioCortex and displays it. Editing and saving in raw text mode also works without any issues. However, even though widgets for single versions of data types such as date, number, enumerable do show up, in widget mode the prototype fails to display multiplicity options. The following section will present the reasons for the incomplete prototype.

Furthermore, as this thesis forms part of a larger study on the creation of Spreadsheet 2.0, the evaluation will also critically evaluate the implemented prototype and discuss its part in the greater research study. Finally this chapter will conclude with suggestions for this research study.

6.1. Prototype Issues

In the following chapter we will touch on some of the issues experienced when coming up with a prototype. These range from framework licensing to testing.

6.1.1. Using a closed source framework

In choosing Wijmo Flexgrid as a framework multiple problems where encountered. Due to its closed source approach, the framework support was limited and dependent on ComponentOne, the company that creates Wijmo. Every issue had to be raised in their forum and troubleshooting the Wijmo code is not possible due to its minified and concatenated state. Also, the fact that Wijmo does not only cater for AngularJS and had documentation for AngularJS, JQuery and Knockout seemed to, at times, also create confusion.

6.1.2. Ongoing development of the SocioCortex Framework

Ongoing development made the implementation of certain functionality on the SCSSL side problematic as error codes were received from the SocioCortex REST API that would not assist in debugging the problem. Furthermore, support tickets had to be opened for feature requests and bugs which could take a while to be implemented. However, the fact that individuals were directly approachable and by having bi-weekly SocioCortex meetings helped in resolving issues.

6.1.3. Insufficient AngularJS and Wijmo Knowledge

With previous basic AngularJS knowledge and many years of web development experience it was assumed that the prototype that was to be developed would be possible. The reality however was different in that there was still a learning curve to AngularJS, not to mention the Wijmo Framework.

6.1.4. Time-frame and Underestimation of Scope

Initially the proposed solution should have been achievable in the planned time-frame however the process of getting better accustomed to AngularJS and Wijmo took longer than expected. Also functionality like multiple option, date, number and reference widgets turned out to be more complicated to implement than previously expected and increased the scope of the prototype.

6.2. Critical Reflection

Reflecting on the prototype that was created it is evident that this prototype requires testing and validation. This prototype was implemented in a short time frame and is still therefore very rough.

For the overall concept there are four aspects that would need to be highlighted and elaborated on:

1. The use of Raw Text Mode versus UI widgets: It is in implementing both that one is able to assess which mode is better. Initially for keyboard heavy users the raw text mode did have its appeal since one was able to rapidly change cell values

without wasting time clicking. The SCSSL UI however also had its appeal to the user with indirectly forcing the user to adopt a type model by limiting the input UI. Ultimately this argument can depend on the device of the end-user. If the end-user is desktop affine, raw text mode is better suited. Looking at the current trend in technology which seems to advance in the mobile and tablets space, the UI widget mode is distinctively better suited for these devices. With widgets that pop-up and control the input without the use of keyboards is identical to tablets input currently. So in summary both are necessary and their use depends on the devices that the end-user utilizes.

2. Designing the system to be push based instead of pull: Currently one of the flaws of the implementation is the fact that the AngularJS API fetches the content from the SocioCortex REST API. With the numerous changes that occur via other web-clients, it is foreseeable in future that conflicts will occur which need to be handled on the client side. Since a table locking would defeat the distributed purpose of the SocioCortex, the solution of converting the SocioCortex REST API into a push server would allow for instant updates on all devices that implement the push protocol.
3. The approach to model data retrospectively: This aspect of the SocioCortex makes it one of the few of its kind. In the development of the prototype we enforced a model onto data depending on how it was defined. Thus, doing the opposite of what SocioCortex does. The “No type” column attempts to solve this issue, yet from the outset one would still have to define a type which one then populates with data. Essentially one would have to allow users to define and move data from one type to another to be able to offer the user full flexibility.
4. Different types in fields and their usefulness: Modeling types for the sake of modeling data and showing date pickers and number counters might be useful for one use case, however it might also prove to be ineffective in another case. One would need to look at these different use cases and critically assess if widgets like date pickers are fit for the right purpose.

7. Conclusion

In this thesis we have attempted to create a prototypical web-based spreadsheet to reduce errors when entering data. This was done by following the Design Science Research approach.

Initially the environment was explored. It was established that spreadsheet-use is on the rise especially when it comes to organizational critical applications and that spreadsheets also seem to be error prone. Thus, it has become an important business need to address and to find a solution to. By exploring current technologies, it was found that many technological solutions already exist, which might have come about to resolve this issue. It was also found, based on comparisons, that current solutions might be able to provide web based, real time-collaboration systems with widget support. However, they all lacked the integration of Complex Linked Data.

After the in-depth analysis of the business needs, processes and technology in the environment chapter we turned to academia and assessed the knowledge base. This consisted of a number of proactive and reactive solutions to prevent error occurrence in spreadsheets. We also looked at Hybrid Wikis as the SocioCortex Hybrid Wiki is the system that the prototype for this thesis extended. In the SocioCortex we then specifically highlighted its attribute types (e.g. text, number, date) as it was these that this thesis built up on.

After the establishment of the Environment which demonstrated the business needs and the knowledge base which showed the applicable knowledge, we set out to develop a concept for the prototype. The SocioCortex Sheet Smart Layer (SCSSL) was created which, based on the MVC paradigm had SCSSL transformation functions and controllers, SCSSL UI widgets for the “view” part and took data types from SocioCortex as the model. The SCSSL would exist between the SocioCortex REST API and the spreadsheet framework. By making use of high and low fidelity mockups every single SCSSL UI was visualized. Furthermore, the multiplicity handling of the attribute types was addressed before concluding the introduction chapter with an overview mockup for the full system.

We then continued with the actual conception of the prototype. After an intense assessment for the best solution a spreadsheet control we then settled for one called the

Wijmo Flexgrid. Flexgrid did not only support AngularJS, direct cell editing and widget support for the spreadsheet rendering, but was also backed by a team which updated the components on a frequent basis. Flexgrid was then used as a front-end control to the implementation of SCSSL which was then explored in detail. The functionality of the spreadsheet was then explored.

Unfortunately, even though it is stipulated in the design science research methodology, we did not get to the evaluation phase of the prototype due to the issue of the prototype not reaching a stable version that could be tested due to problems that were encountered. These were then explored and critically reflected upon. Further more other issues were highlighted.

In the beginning of this thesis we set out to solve the question of: What is a model driven approach to managing complex linked data in order to prevent erroneous input in web-based spreadsheet systems? The concept and prototype developed for this thesis is indeed a first approach to solving the issue. By implementing a middle layer solution for the filtering and sanitizing content can assist with incorrectly entered data. Furthermore, the implementation of the SCSSL widgets bind the input to a certain format and allow for minimal deviation. There are however edge cases which have not been tested and could lead to the detriment of data. Therefore, the current implementation must still be assessed, evaluated and refined through future iterations to establish its validity to solving the research question.

As this thesis extends the current research being done in the form of a PhD dissertation, it only fills a small void in the problems that spreadsheets have, as these issues do not only affect data but a wide range of other issues as well.

7.1. Future work

Due to the fact that this thesis contributes to a larger dissertation, more work in this field will most certainly be required. The prototype would need to be extended to support multiple types and provide better support for the MxL language. Over and above that, as the SCSSL concept and prototype is integrated with the SocioCortex, one has to establish if this concept is indeed an efficient way of editing wikis and modeling information. There might exist a range of other options such as third party input integration, natural language input recognition or AI models that create models based on analysis of many existing spreadsheets.

Moreover, due to the Wijmos Framework mentioned issue of closed source software and limited support it would also be beneficial to re-visit other spreadsheet frameworks. Whilst doing a compilation of frameworks, a newer framework called “Handsontable”³² turned out to be of a promising nature. Handsontable not only has the same features as Wijmo Flexgrid but it also is open-source. It has a rising community of adopters and supporters; and is inexpensive to use. Furthermore, projects like “nghandsontable”³³ are already written to fill the void of Angular support.

Lastly, it is imperative that one conducts evaluations and user testing in order to empirically validate the concept proposed and prototype created in this thesis.

Spreadsheets are in essence a construct from the “stone age” of computers. At their core they have conceptually not changed much at all in functionality, nor in user interface design. They will be with us for many years to come and therefore it is imperative to solve issues that they present. It is in addressing these problems that one seeks to reduce the risk of failure for many organizations and industries in the future. Therefore any refinement in this area will be of great value to its users.

³²<http://handsontable.com/>

³³<https://github.com/handsontable/ngHandsontable>

Bibliography

- [1] October 2015. URL <https://www.matthes.in.tum.de/pages/13uzffgwlh8z4/SocioCortex>. (Cited on pages x and 22.)
- [2] October 2015. URL <http://www.eusprig.org/horror-stories.htm>. (Cited on pages 3 and 9.)
- [3] September 2015. URL <http://news.microsoft.com/bythenumbers/index.HTML>. (Cited on page 2.)
- [4] Robin Abraham and Martin Erwig. Type inference for spreadsheets. In *Proceedings of the 8th ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 73–84. ACM, 2006. (Cited on pages 16 and 17.)
- [5] Robin Abraham, Martin Erwig, Steve Kollmansberger, and Ethan Seifert. Visual specifications of correct spreadsheets. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 189–196. IEEE, 2005. (Cited on page 18.)
- [6] Yanif Ahmad, Tudor Antoniu, Sharon Goldwater, and Shriram Krishnamurthi. A type system for statically detecting spreadsheet errors. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 174–183. IEEE, 2003. (Cited on page 16.)
- [7] Tudor Antoniu, Paul Steckler, Shriram Krishnamurthi, Erich Neuwirth, Matthias Felleisen, et al. Validating the unit correctness of spreadsheet programs. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 439–448. IEEE, 2004. (Cited on page 17.)
- [8] Wynn Bailey, Brenda Dillon, Matthew Labarge, John Taylor, Khushwant Gill, Donald S Lloyd, and Wade M Person. Ribbon-style user interface for a software application, March 27 2008. US Patent App. 12/056,902. (Cited on page 2.)
- [9] Kenneth R Baker, Stephen G Powell, Barry Lawson, and Lynn Foster-Johnson. Comparison of characteristics and practices amongst spreadsheet users with different levels of experience. *arXiv preprint arXiv:0803.0168*, 2008. (Cited on page 4.)

- [10] Sasa Baskarada. How spreadsheet applications affect information quality. *Journal of Computer Information Systems*, 51(3):77–84, 2012. (Cited on page 9.)
- [11] Anil Bhansali, Rohit V Wad, Eric Michelman, and Wyatt T Riley. Method and system for allowing multiple users to simultaneously edit a spreadsheet, December 21 1999. US Patent 6,006,239.
- [12] William H Brown, Raphael C Malveau, and Thomas J Mowbray. Antipatterns: refactoring software, architectures, and projects in crisis. 1998. (Cited on pages 2 and 5.)
- [13] Michel Buffa and Fabien Gandon. Sweetwiki: semantic web enabled technologies in wiki. In *Proceedings of the 2006 international symposium on Wikis*, pages 69–78. ACM, 2006. (Cited on page 19.)
- [14] Margaret Burnett, Curtis Cook, Omkar Pendse, Gregg Rothermel, Jay Summet, and Chris Wallace. End-user software engineering with assertions in the spreadsheet paradigm. In *Proceedings of the 25th international conference on Software engineering*, pages 93–103. IEEE Computer Society, 2003. (Cited on page 17.)
- [15] Chris Chambers and Martin Erwig. Dimension inference in spreadsheets. In *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pages 123–130. IEEE, 2008. (Cited on page 14.)
- [16] Michael J Coblenz, Andrew J Ko, Brad Myers, et al. Using objects of measurement to detect spreadsheet errors. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 314–316. IEEE, 2005. (Cited on pages 16 and 17.)
- [17] Grenville J Croll. The importance and criticality of spreadsheets in the city of london. *arXiv preprint arXiv:0709.4063*, 2007. (Cited on page 9.)
- [18] Jácome Cunha, João Paulo Fernandes, Jorge Mendes, and João Saraiva. Mdsheet: A framework for model-driven spreadsheet engineering. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1395–1398. IEEE Press, 2012. (Cited on page 16.)
- [19] Angelo Di Iorio, Fabio Vitali, and Stefano Zacchiroli. Wiki content templating. In *Proceedings of the 17th international conference on World Wide Web*, pages 615–624. ACM, 2008. (Cited on page 19.)
- [20] Martin Erwig and Margaret Burnett. Adding apples and oranges. In *Practical Aspects of Declarative Languages*, pages 173–191. Springer, 2002. (Cited on page 17.)

- [21] Anja Haake, Stephan Lukosch, and Till Schümmer. Wiki-templates: adding structure support to wikis on demand. In *Proceedings of the 2005 international symposium on Wikis*, pages 41–51. ACM, 2005. (Cited on page 19.)
- [22] Nilesh Jain, Priyanka Mangal, and Deepak Mehta. Angularjs: A modern mvc framework in javascript. *Journal of Global Research in Computer Science*, 5(12):17–23, 2015. (Cited on pages 46 and 47.)
- [23] Brian Knight, David Chadwick, and Kamalesen Rajalingham. A structured methodology for spreadsheet modelling. *arXiv preprint arXiv:0805.4218*, 2008.
- [24] David M Lane, H Albert Napier, S Camille Peres, and Anikó Sándor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, 18(2): 133–144, 2005. (Cited on page 28.)
- [25] Florian Matthes, Christian Neubert, and Alexander Steinhoff. Hybrid wikis: Empowering users to collaboratively structure information. *ICSOFT (1)*, 11:250–259, 2011. (Cited on pages 19, 20, and 21.)
- [26] Jorge Mendes. Classsheet-driven spreadsheet environments. In *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*, pages 235–236. IEEE, 2011. (Cited on page 16.)
- [27] Martin J O’Connor, Christian Halaschek-Wiener, and Mark A Musen. Mapping master: A flexible approach for mapping spreadsheets to owl. In *The Semantic Web–ISWC 2010*, pages 194–208. Springer, 2010. (Cited on page 18.)
- [28] Raymond R Panko. What we know about spreadsheet errors. *Journal of End User Computing*, 10:15–21, 1998. (Cited on page 3.)
- [29] Amit Phalgune, Cory Kissinger, Margaret Burnett, Curtis Cook, Laura Beckwith, and Joseph R Ruthruff. Garbage in, garbage out? an empirical look at oracle mistakes by end-user programmers. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 45–52. IEEE, 2005. (Cited on page 14.)
- [30] Stephen G Powell, Kenneth R Baker, and Brian E Lawson. Errors in operational spreadsheets: A review of the state of the art. In *System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on*, pages 1–8. IEEE, 2009. (Cited on page 3.)
- [31] Daniel J Power. A history of microcomputer spreadsheets. *Communications of the Association for Information Systems*, 4(1):9, 2000. (Cited on page 2.)

- [32] Thomas Reschenhofer and Florian Matthes. An empirical study on spreadsheet shortcomings from an information systems perspective. In *Business Information Systems*, pages 50–61. Springer, 2015. (Cited on pages 2, 3, and 6.)
- [33] Thomas Reschenhofer, Ivan Monahov, and Florian Matthes. Type-safety in ea model analysis. In *Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International*, pages 87–94. IEEE, 2014. (Cited on pages x, 20, and 21.)
- [34] Gregg Rothermel, Lixin Li, Christopher DuPuis, and Margaret Burnett. What you see is what you test: A methodology for testing form-based visual programs. In *Proceedings of the 20th international conference on Software engineering*, pages 198–207. IEEE Computer Society, 1998. (Cited on page 14.)
- [35] Marton Sakal and Veselin Pavlicevic. Spreadsheets-how it started. *Management*, 9(4):09–14, 2003. (Cited on pages x, 2, and 3.)
- [36] Vipin Samar and Sangeeta Patni. Controlling the information flow in spreadsheets. *arXiv preprint arXiv:0803.2527*, 2008. (Cited on page 19.)
- [37] Sebastian Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on*, pages 388–396. IEEE, 2006. (Cited on page 19.)
- [38] Adam Souzis. Building a semantic wiki. *Intelligent Systems, IEEE*, 20(5):87–91, 2005. (Cited on page 19.)
- [39] United States Code. Sarbanes-oxley act of 2002, pl 107-204, 116 stat 745. Codified in Sections 11, 15, 18, 28, and 29 USC, July 2002. (Cited on page 19.)
- [40] R Hevner von Alan, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004. (Cited on pages x, 5, 6, and 9.)