Excemplify: A Flexible Template Based Solution, Parsing and Managing Data in Spreadsheets for Experimentalists

Lei Shi[1,*], Lenneke Jong[1], Ulrike Wittig[1], Philippe Lucarelli[2], Markus Stepath[2], Stephanie Mueller[2], Lorenza Alice D'Alessandro[2], Ursula Klingmüller[2] and Wolfgang Müller[1]

[1]Heidelberg Institute for Theoretical Studies (HITS), Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany, http://www.h-its.org

[2]German Cancer Research Center (DKFZ), DKFZ-ZMBH Alliance, Im Neuenheimer Feld 280, 69120 Heidelberg, Germany, http://www.dkfz.de/

Summary

In systems biology, quantitative experimental data is the basis of building mathematical models. In most of the cases, they are stored in Excel files and hosted locally. To have a public database for collecting, retrieving and citing experimental raw data as well as experimental conditions is important for both experimentalists and modelers. However, the great effort needed in the data handling procedure and in the data submission procedure becomes the crucial limitation for experimentalists to contribute to a database, thereby impeding the database to deliver its benefit. Moreover, manual copy and paste operations which are commonly used in those procedures increase the chance of making mistakes. *Excemplify*, a web-based application, proposes a flexible and adaptable template-based solution to solve these problems. Comparing to the normal template based uploading approach, which is supported by some public databases, rather than predefining a format that is potentiall impractical, *Excemplify* allows users to create their own experiment-specific content templates in different experiment stages and to build corresponding knowledge bases for parsing. Utilizing the embedded knowledge of used templates, *Excemplify* is able to parse experimental data from the initial setup stage and generate following stages spreadsheets automatically. The proposed solution standardizes the flows of data traveling according to the standard procedures of applying the experiment, cuts down the amount of manual effort and reduces the chance of mistakes caused by manual data handling. In addition, it maintains the context of meta-data from the initial preparation manuscript and improves the data consistency. It interoperates and complements *RightField* and *SEEK* as well.

1 Introduction

Dynamic modeling and model simulations are important tasks in systems biology[1]. To interpret dynamic cellular processes and to predict the outcome of interventions, mathematical models have to be established based on certain amount of reliable experimental data (meta-data for experimental conditions and raw data). Unfortunately, so far, experimental data obtained directly from individual laboratory is not really comparable and exchangeable and even not

*To whom correspondence should be addressed. Email: lei.shi@h-its.org

easily reproducible from outside. Contributions on developing standardized data acquisition procedures [2], defining standard modules [3][4] for describing different experimental data and defining correlated exchange format [5] for sharing the experiment data have already been made in recent years. Thus, now, to build and populate public databases for quantitative raw experimental data as well as standardized experimental conditions is of high interest of the scientific community. However, two major problems obstruct the database construction and subsequently impede the database to deliver its benefit.

High cost in the data handling and data submission procedures. A considerable number of spreadsheets will be produced even for one experiment. Managing spreadsheets in different experiment stages and assembling necessary data in a valid data format for uploading to the database cost experimentalists great effort and are highly time consuming.

High possibilities of making mistakes. Copy and paste are two operations which are frequently used in the whole experiment work flow. Potential errors, which are produced in between, are barely detectable when they are distributed among different sheets. They are even more difficult to be identified once the data is extracted from original sheets and transferred into the database.

Motivated by the problems stated above, *Excemplify*, a web-based application, has been developed to automate the generation of data sheets at different experiment stages based on experiment work flow. Its template-based solution lets experimentalists create experiment-specific templates as well as build corresponding parsing knowledge bases by their own. The intelligent parsing technology utilizes the embedded knowledge of the template to fetch information from the initial setup stage sheet of an experiment and generates the following ones. It cuts down the amount of needed effort and reduces the chance of mistakes caused by manual data handling. It could also be used to standardize the experiment applying procedures and correlated data traveling flows in the laboratory. By providing the possibilities to export final stage data summary file into certain standard data exchange format (e.g. GelML[5], GelInspector[6][7]), Excemplify allows exchanging the data within the laboratory and with collaboration partners, as modelers.

Currently, *Excemplify* is focusing on immunoblot data, which is quantitative and time-resolved experimental data obtained using techniques such as immunoprecipitation and immunoblotting [8][9]. It has been tested[1] by group members in the **S**ystems **B**iology of **S**ignal **T**ransduction division at the DKFZ and the code is hosted as an open-source project on Github[2].

2 Related Work

In recent years, the importance of the interface between semi-structured Excel data and strictly structured databases has been gradually realized in the field of experimental data management. Projects such as *LabKey Server* [3][10] and *SEEK* [11] all face the challenge of handling exper-

[1]The Excemplify test version for DKFZ is running on http://sabio.h-its.org/excemplify/

[2]Open-source project Immunoblot with the link https://github.com/excemplify/immunoblot/

[3]https://www.labkey.org/

imental data in spreadsheets. Among the tools that address the challenge, we consider *RightField*[4] and *Google Refine*[5] as the work that are closely linked to *Excemplify*. Table.1 summarizes the features of those tools compared to *Excemplify*. In following sections we explain more details about how those tools differ in focus and how they complement *Excemplify*.

Table 1: Comparison of Tools for Handling Spreadsheets

	Software	Excel	Domain (focus)	Parsing	Usage	Interaction
RightField	stand-alone desktop application	support	experimental data (meta-data annotation)	general	create ontology-embedded template	click
Google Refine	web application (stand-alone mode)	support	general	general	data cleanup and transformation	input expression
Excemplify	web application	support	experimental data (experimental work flow)	knowledge based	automate sheets generation data transformation	drag and drop

2.1 RightField

RightField[12], with *Populous*[13] extension, is a stand-alone tool that provides a mechanism for embedding ontology annotation into standard spreadsheet templates for different types of experiments [12]. It aims to help experimentalists to make annotations of their rich meta-data by using ontologies and terminologies. Knowledge parsed from selected ontology files is binding to regions of Excel sheets. Selection lists of the typeless choice between standard identifiers are created to restrict the range of populated values for individual Excel cells. A *RightField*-enabled template reduces the time for searching suitable controlled vocabularies and also decreases the errors produced in annotation process.

RightField has been used to create *Just Enough Results Model* (JERM)[12] templates for reporting varying types of experiments in *SysMo SEEK*[6]. The knowledge embedded in *RightField*-enabled template improves the meta-data annotation quality, but it has nothing to do with the structure of the content. To interpret the data inside such template, the template shall be preformatted inside the parsing application. The solution proposed by *Excemplify* improves the flexibility and adaptability on that limitation. By planning to support creating *Excemplify* template based on the existing *RightField*-enabled template, *Excemplify* will be able to integrate the benefit of *RightField*.

2.2 Google Refine

Google Refine is another powerful web application running on stand-alone mode for data cleanup and transformation. Excel file is one of its major supported formats. It represents values of spreadsheets and normalizes string values by several operations, i.e. removing white space, changing all characters to lowercase representation, removing punctuation and control characters, etc.[14]. After generating keys from normalized values and clustering them according to selected algorithms, it represents to users all different facets. Users can curate facets by providing mappings between equal strings (e.g. mueller→müller), and *Google Refine* is thus able to refine tables accordingly.

[4]http://www.rightfield.org.uk
[5]http://code.google.com/p/google-refine/
[6]https://seek.sysmo-db.org/

Google Refine proposes a solution, which helps users to detect misspelling and inconsistent values in tables. By obtaining the mapping knowledge through interaction, it is able to fix the inconsistency problem and export high quality reports. *Google Refine* parses data values and senses inconsistencies on the general string level, and it does not support transferring data based upon any scientific domain-based knowledge. Moreover, the expression language (e.g. grel:value==cells['Machine ID'].value [14]) used during interaction for customizing facets is (in our experience) a hurdle for novice users. However, as a popular and available mature data cleaning tool, *Google Refine* is a good complementary tool for further refining excel data of any sheet, which is imported or generated by *Excemplify*.

3 Manual Walk-through

The work of *Excemplify* is on the basis of a **field study** [15] in the laboratory, watching and discussing experimental work. As it was mentioned in the introduction (Section 1), currently, we are focusing on immunoblot type experimental data, so during our investigation, an immunoprecipitation/immunoblotting type experiment was performed and dicussed.

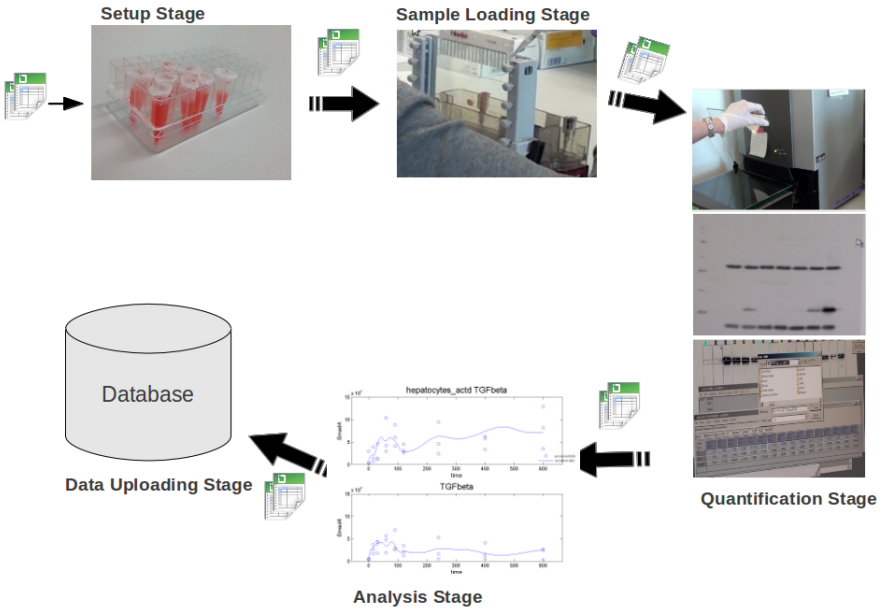

Figure 1: Spreadsheets Cover Each Stage of the Experiment

By observing the experimentalists' daily work and collecting sample sheets from our lead users[16] and user representatives[17], we summarized data traveling involved work flows in Figure.1. In our context, five important experiment stages are: **setup stage** (also called experiment preparation stage), **sample loading stage**, **quantification stage**, **analysis stage** and **data uploading stage**. Experiment works relevant to each stage are generalized as follows. In the initial setup stage, the experimentalist makes notes of all meta-data about experimental conditions in order to do the plan. For example, the treatment of samples, proteins of interest, time course plan and antibodies are all included. In the sample loading stage, the experimentalist produces gel loading sequences and corresponding sample loading detail of each blot. Proteins of the samples are separated using gel electrophoresis and are transferred onto a membrane, which make them accessible to antibody detection. By using chemoluminescence detection

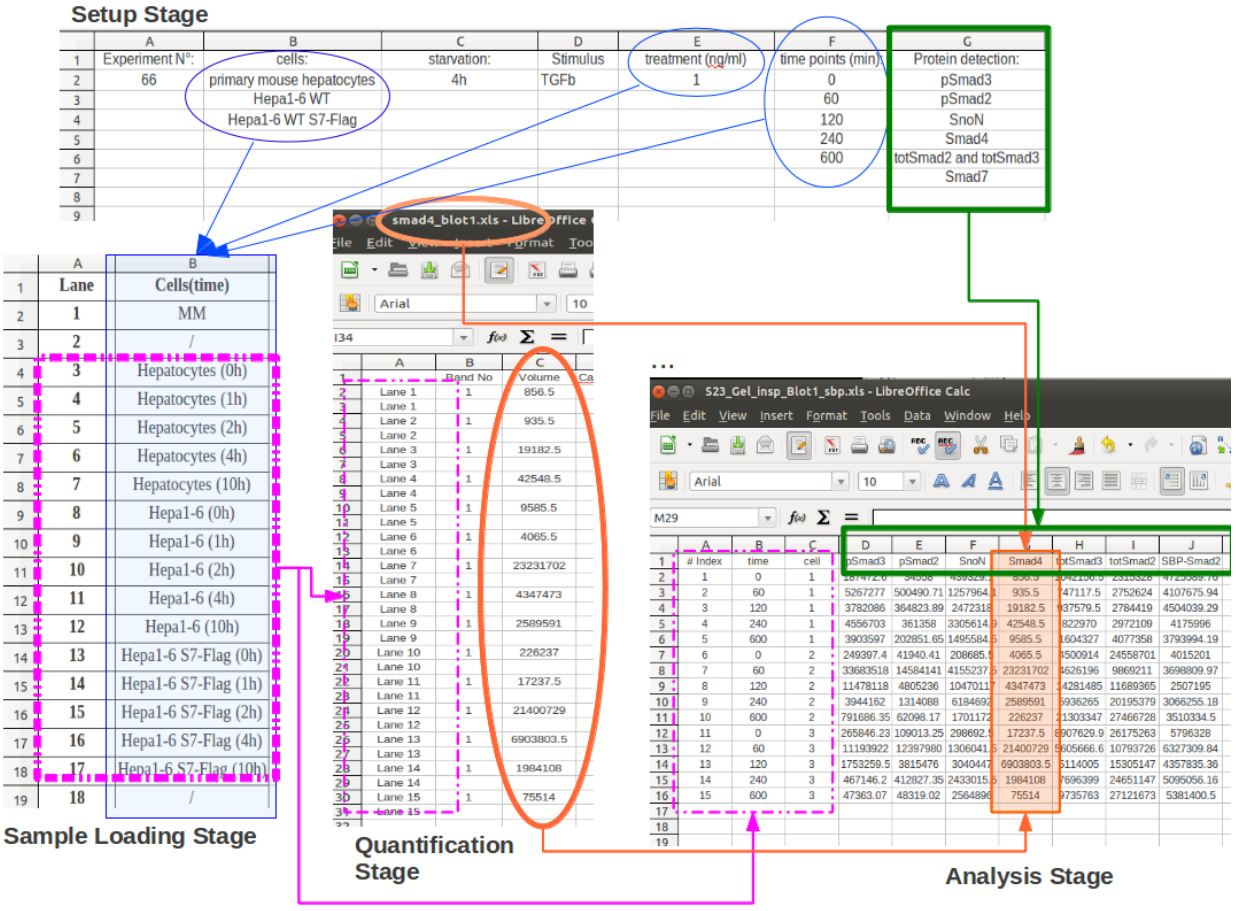

Figure 2: Data Traveling Between Sheets

and a scanning system, e.g. a CCD camera, the signal is digitized in the quantification stage. In the analysis stage, the experimentalist transfers quantitative data into a file with certain tool-specific format and analyzes it. In the data uploading stage, the raw data as well as the analysis results are reformatted again and are submitted to the database. Spreadsheets are produced in each stage, specific data contents are assembled according to the purpose of the stage respectively. Except the analysis stage and the uploading stage, in which formats of the sheet are unified in the laboratory either according to the used tool or according to the database, other experiment stages, the stage-specific structures tend to vary from person to person. Figure.2 takes data sheets from one experimentalist as an example, arches and arrows in the figure point out the flow of data traveling across stages.

From the field study, we learned that independent of the experiment type, the general work flow (stages) of the experiment stays in the laboratory. There are processes that define how data travels across stages for certain type of experiment. Domain-based knowledge is highly important for following these procedures. **The main observations determining the further work were:**

1. Sheets correspond to experimental stages.

2. How data travels across different stages/sheets can be predefined according to the experiment type.

3. For each experiment type, the pre-experiment stage (setup) sheets contain the minimal meta-data that are needed to form the basic information of subsequent sheets.

4. For each experiment type, later stage spreadsheets can always be deduced from spreadsheets of former stages and their stage-specific measurements.

4 Flexible and Adaptable Template Solution

Templates enable the laboratory to standardize its experimentalists' data populating procedures. To have a fixed and predefined template valid for any experimental setup is an ideal situation. However, in real life, specific templates are preferred by experimentalists for specific types of experiments. Even for the same type of experiment, there are different personal styles, mostly due to small setup differences, which are driven by experimentalists' own scientific decisions. Thus, in order to encourage experimentalists to use templates, flexibility in the structure of the template has to be enabled.

Based on assumptions we described in the field study (Section 3), *Excemplify*, a *Grails*[7] web application using *Apache POI* library[8], has been developed. It proposes a flexible template-based solution for managing experimentalists' spreadsheets. Two interfaces are provided and secured under the *Spring Security* framework[9]. One is for users with an administrator role to create their own templates at any time and to build the corresponding knowledge base for the structure of the contained data. The details of building templates inside *Excemplify* is explained in Section 4.1. Another, much simpler interface is for users with an experimentalist role to help them with generating different stages spreadsheets based on the *Excemplify* template.

To separate view related operations from solid data models, *Excemplify* application applies the **model view controller** (MVC) pattern in its architecture[18]. Fundamental components of the system are diagrammed in the Figure.3.

4.1 Technology for Building the *Excemplify* Template

Meta-data noted in the setup stage spreadsheet describes all required conditions of a certain experiment. Categories of these meta-data correspond to kinds of the domain-based knowledge, which can not be further decomposed, like cell, protein, time, etc. We call those categories **meta-knowledge**.

Figure.4 depicts the *Excemplify* interface for administrator users. Once a new Excel file is opened in the interface, it is parsed, converted into XML and represented in the markup container[10] as a web-based Excel style sheet instance. By clicking on cells of a certain column in the sheet, the user is able to mark them and create new concepts (unknown meta-knowledge). Drag the concept metaphor[19], which stands for the implemented domain-based meta-knowledge, on the right panel and drop it into the marked cell region, the mapping between such regions (location) of the sheet and the meta-knowledge is built. For instance, in

[7] http://grails.org/

[8] http://poi.apache.org/

[9] http://static.springsource.org/spring-security/site/

[10] jquery.sheet http://visop-dev.com/Project+jQuery.sheet

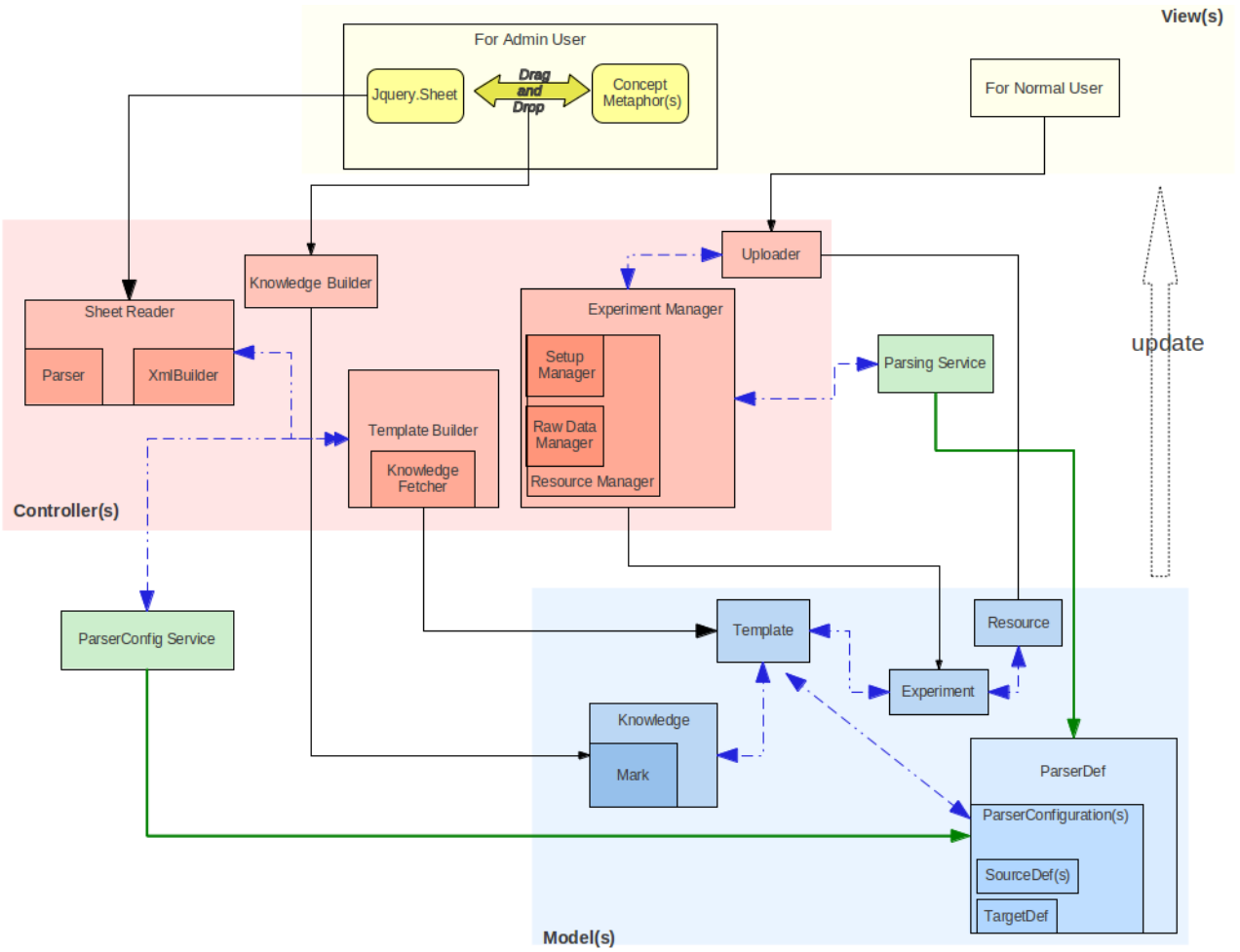

Figure 3: Component Diagrams

the knowledge base, the knowledge item like ”Label:timepoints; Concept:time; Mark Cell-Range:B5” simply tells *Excemplify* that in the fifth row and second column of the sheet, a column with the label ”timepoints” begins that contains time category information.

It is possible to mark not only the cell but also the complete region in the sheet, which provides exactly the location of the top-left cell and the down-right cell for a certain knowledge. Since the current version of the application uses an automatic detector to sense the ending of the column from the marked label cell, i.e. either to sense an empty cell or to detect the start of another concept, such a procedure is simplified.

Saving a template operation in the interface not only saves the binary data of the opened sheet but also the customized knowledge. By further setting the stage purpose (i.e. as setup template or raw data template) of the new template, *Template Builder* will build it accordingly. The *ParserConfig Service* will be called to create a series of parsers for different stage-specific generation operations (i.e. *setupToLanes*, *rawDataToGelInspector* and etc.) and bind them to the template. The detailed framework is explained in Section 4.2.

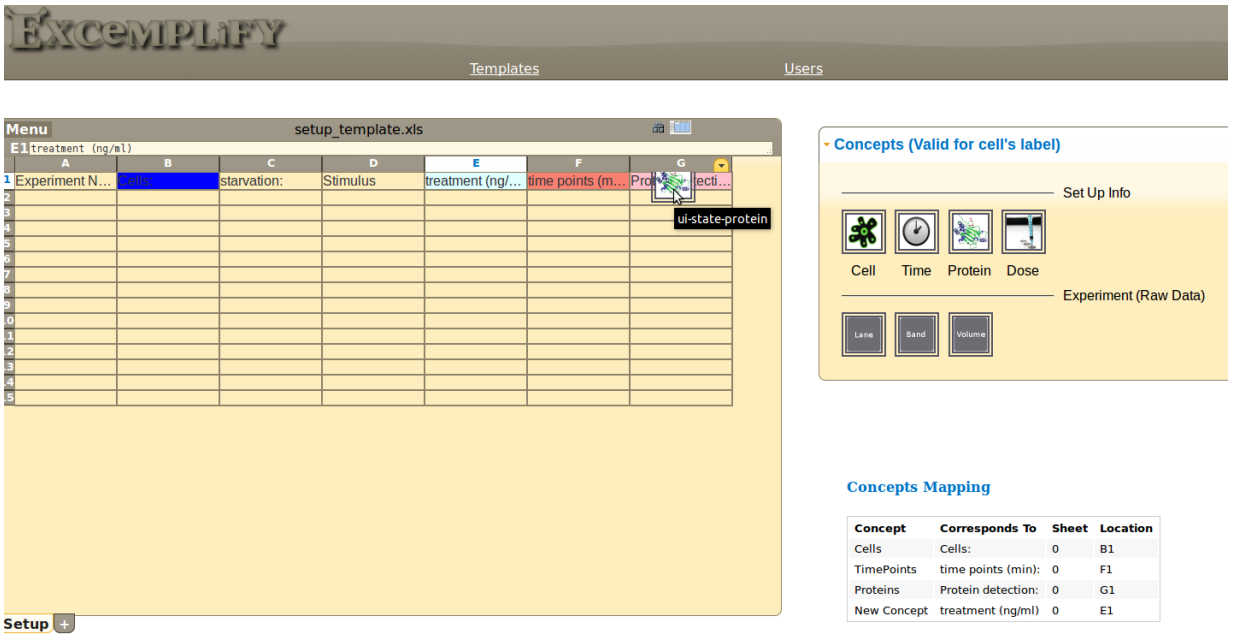

Figure 4: Build the Knowledge via Drag and Drop Interaction

4.2 Parsing Technology

The parsing architecture of *Excemplify* is used to generate next stage spreadsheets for experimentalists. Each generating operation is composed of several generating tasks. Each task contains several parsing activities. The activities in one generating operation can be arranged in a certain sequence.

Based on basic parsing actions (e.g. copy, split and merge), the meta-parser (*ParserConfiguration*) can be easily configured by setting the source and the target location. Composed by multiple meta-parsers, an individual parser (*ParseDef*) is defined for one parsing activity. Planning a generating task is technically realized by building a queue of parsing activities. *Parsing Service* in *Excemplify* constructs a queue of *Blocks*, containing parser objects and parsing states. *Blocks* are arranged in the sequence according to the kind of the generating task(s). Figure.5 gives a simple example of how to build the queue for an outer-product kind generating task, which is the part of *setupToLanes* generation operation applied in the **sample loading stage**.

Executing the generation operation is just applying a queue. The intermediate results, i.e.logs for each progress, exceptions and even errors, are stored in the state and forwarded to the next working *Block*.

In the future, this information will enable advanced error handling.

4.3 Practical Scenario In Laboratory

Excemplify is the tool which could be used in varying degrees to standardize the data traveling work flow. If only official administrators in the laboratory can create templates, then Excel data can be standardised. If, however, each experimentalist in a given lab can create his/her own templates, then the data flows for all experiments of the a given experiment type by the same

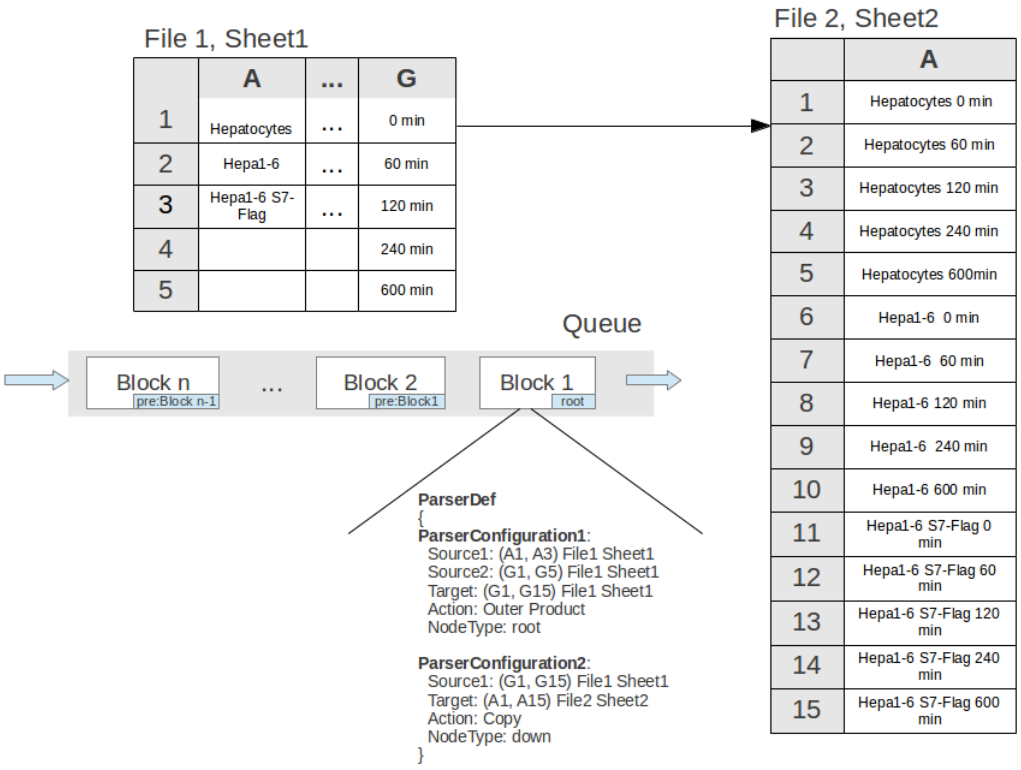

Figure 5: Build Parsing Blocks in a Queue

experimentalist are standardized. This is of smaller benefit, but already reduces documentation effort.

Given an example scenario of using *Excemplify* in the laboratory for certain experiment type, interaction sequences of both interfaces are described below:

Interaction Steps: User with Administrator Role

1. Open a Excel sheet that contains titles of experiment meta-data.
2. Mark sheet cells whose locations indicate the beginning of regions with certain meta-knowledge.
3. Build the links between marked cells and the metaphor icon (implemented meta-knowledge) by drag and drop.
4. Save the file as the setup template.
5. Repeat the same steps for other necessary templates (i.e. the template for raw data, the template for result file of certain analyzing tool and etc.)

Interaction Steps: User with Experimentalist Role

1. Download the setup template from the template library, fill the detail conditions under those titles and save the file as the initial setup sheet of the experiment.

2. Upload the setup sheet, select the name of the used templates for different stages, fill in other experiment properties such as randomization criteria[2], blot number and applied stimulus and create the experiment workbook.

3. Download the automatically generated sample loading sheets.

4. Upload the raw data files to the experiment.

5. Choose the tool-specific format (e.g. GelInspector[11]) and get the automatically generated import file of the experiment for certain analysis tool.

6. Upload the analysis result files to the experiment.

7. Choose one of the standard exchange formats (e.g. GelML) and export the experimental data. Send the automatically exported files to the database. e.g.*Virtual Liver SEEK*[12] or *SABIO-RK*[13] [20].

5 Benefits and Limitations

Excemplify enables users with an administrator role to create a knowledge embedded template. Drag and drop interaction ease the knowledge transfer procedure. By indicating accurate cell ranges (location) of corresponding meta-knowledge, any style of template for this domain is acceptable and processable in *Excemplify*. Location-based mapping provides a practical way to bypass linguistic difficulties. String level inconsistencies either caused by misspelling or by different naming become compatible.

The parsing framework of *Excemplify* helps experimentalists to generate spreadsheets stage by stage during their experiment. It automates the data populating procedure and reduces the experimentalists' time and effort. It successfully cuts down the amount of their copy and paste operations during data handling procedures, thus reducing errors and inconsistency problems caused by manual work. The log file of each progress and intermediate results makes it possible to trace changes of each experiment. The parsing configuration mechanism is flexible to adapt to different types of experiments.

Excemplify can also benefit from using together with *RightField*. Among other things, *RightField* helps eliminate typing mistakes for annotation by providing Ontology-driven selection lists to users, which can be then used within *Excemplify*.

However, there are still some limitations of the current *Excemplify* version. Input errors in the initial setup sheet that cause errors and exceptions during queue applying will terminate the whole operation. An corresponding error message is sent to the user just to notify the operation failure. How to handle errors and exceptions occurring in between and how to recover the generation operation is one of our focuses in the future.

[11] http://www.fdmold.uni-freiburg.de/ maiwald/GelInspector/
[12] http://seek.virtual-liver.de/ an instance of SEEK
[13] http://sabio.h-its.org/

6 Future Plans

There are two main directions for our future work on *Excemplify*. One is advanced error handling and the other is further investigating the *Human Computer Interaction* (HCI) properties of *Excemplify* in order to analyze the connection between user involvement and system success.

As we have already explained in Section 5, the current version of our parsing technology can notify users but does not allow error fixing interactions. We will extend *Excemplify* such that the state and queue can be maintained each time when a parsing error is encountered that can only be fixed by user interaction. After the user has resolved errors, the state and queue can be restored, and the parsing continues. This is much more convenient than re-submitting the file multiple times.

Given the successful experience of involving users in the information system (IS) development from *SysMo SEEK* and *Virtual Liver SEEK*, *Excemplify* also encouraged its users to exert effective influence on the whole development life cycle. Thus, another direction of our future work is to further investigate deeper the HCI properties of *Excemplify*. We want to further study the impact of user involvement on system success parameters such as user satisfaction and user performance according to information systems success models as described in [21, 22].

Obviously we are also looking for even closer ties and integration with other systems, including further stabilizing and extending our support of *RightField*.

Acknowledgements

Excemplify was funded by the *German Research Foundation* (DFG) virtual research environment program (short name: Integrierte Immunoblot Umgebung) from 2010. In two years of investigations and development, we built close cooperative relationships with our partners. We acknowledge group members of the division **S**ystems **B**iology of **S**ignal **T**ransduction at the DKFZ for their assistance in the field study and testing of the *Excemplify* software. Especially we thank Julie Bachmann for providing background knowledge on experimental techniques as well as example spreadsheets. We would also like to give our special thanks to Dr. Andreas Weidemann and Martin Golebiewski from HITS **S**cientific **D**atabase and **V**isualization group, who provided the support for connecting *Excemplify* and *Virtual Liver SEEK*. Parts of this work are and will be funded by the Virtual Liver Network initiative of the *German Federal Ministry of Education and Research* (BMBF) as well as the *Klaus Tschira Foundation* (KTS).

References

[1] E. Klipp and W. Liebermeister. Mathematical modeling of intracellular signaling pathways. *BMC Neuroscience*, 7, 2006.

[2] M. Schilling, T. Maiwald, S. Bohl, M. Kollmann, C. Kreutz, J. Timmer and U. Klingmüller. Quantitative data generation for systems biology: the impact of randomisation, calibrators and normalisers. *System Biology*, 152(4):193–200, 2005.

[3] A. Brazma, P. Hingamp, J. Quackenbush et al. Minimum information about a microarray experiment (miame)—toward standards for microarray data. *Nature Genetics*, 29:365–371, 2001.

[4] C. F. Taylor, N. W. Paton, K. S. Lilley et al. The minimum information about a proteomics experiment (miape). *Nature Biotechnology*, 25(8):887–93, 2007.

[5] F. Gibson, C. Hoogland, S. Martinez-Bartolom é et al. The gel electrophoresis markup language (gelml) from the proteomics standards initiative. *Proteomics*, 10(17):3073–3081, 2010.

[6] M. Schilling, T. Maiwald, S. Bohl, M. Kollmann, C. Kreutz, J. Timmer and U. Klingmüller. Computational processing and error reduction strategies for standardized quantitative data in biological networks. *the FEBS*, 272:6400–6411, 2005.

[7] M. Schilling, A. C. Pfeifer, S. Bohl and U. Klingmüller. Standardizing experimental protocols. *Current Opinion in Biotechnology*, 19:354–359, 2008.

[8] B. R. Corley. *A Guide to Methods in the Biomedical Sciences*. ISBN 0-387-22844-6. Springer, 1 edition, 2004.

[9] I. A. Idris. Bone research protocol. volume 816 of *Methods in Molecular Biology*, pages 223–232. Humana Press, 2012.

[10] E. K. Nelson, B. Piehler, J. Eckels et al. Labkey server: an open source platform for scientific data integration, analysis and collaboration. *BMC Bioinformatics*, 12:71, 2011.

[11] S. Wolstencroft, K.and Owen, F. du Preez, O. Krebs, W. Müller, C. Goble and J. Snoep. The seek: a platform for sharing data and models in systems biology. *Methods in enzymology*, 500:629–655, 2011.

[12] K. Wolstencroft, M. Horridge, S. Owen, W. Müller, F. Bacall, J. Snoep, O. Krebs and C. Goble. Rightfield: Embedding ontology term selection into spreadsheets for the annotation of biological data. *Bioinformatics*, 27(14):2021–2, 2011.

[13] S. Jupp, M. Horridge, L. Iannone, J. Klein, S. Owen, J. Schanstra, S. Wolstencroft and R. Stevens. Populous: A tool for populating an ontology. *BMC Bioinformatics*, 13 Suppl 1:S5, 2012.

[14] T. Morris. Clusteringindepth :methods and theory behind the clustering functionality in google refine. Technical report, 2012.

[15] S. Bly. Field work: Is it product work? *Interactions*, 4(1):25–30, 1997.

[16] C. Lüthje and C. Herstatt. The lead user method: an outline of empirical findings and issues for future research. *R&D Management*, 34(5):553–568, 2004.

[17] L. Damodaran. User involvement in the systems design process a practical guide for users. *Behaviour & Information Technology*, 15(6):363–377, 1996.

[18] A. Leff and J. Rayfield. Web-application development using the model/view/controller design pattern. In *Enterprise Distributed Object Computing Conference, 2001. EDOC '01. Proceedings. Fifth IEEE International*, pages 118 –127. 2001.

[19] A. Marcus. Metaphor design in user interfaces: how to effectively manage expectation, surprise, comprehension, and delight. In *CHI '95 Conference Companion on Human Factors in Computing Systems*, pages 373–374. 1995.

[20] U. Wittig, R. Kania, M. Golebiewski et al. SABIO-RK—database for biochemical reaction kinetics. *Nucleic Acids Research*, 40(D1):790–796, 2012.

[21] H. W. Delone and R. E. Mclean. The DeLone and McLean model of information systems success: a ten-year update. *Management Information Systems*, 19(4):9–30, 2003.

[22] P. Stacie, D. H. William and M. R. Ephraim. Measuring information systems success:models, dimensions, measures,and interrelationships. *European Journal of Information Systems*, 17:236–263, 2008.