Show Me The Rules! A User Study On Making Data Validation Rules Explicit in Spreadsheet Applications

Jessica Gava
Federal University of São João del-Rei
jessica.gava@hotmail.com

Bruna Vilela
Federal University of São João del-Rei
brunaraniquelli@gmail.com

Elder Cirilo
Federal University of São João del-Rei
elder@ufsj.edu.br

Eiji Barbosa Adachi
Federal University of Rio Grande do Norte
eijiadachi@imd.ufrn.br

ABSTRACT

Spreadsheet applications have become one of the most popular end-user programming environments with innumerous built-in facilities, including arithmetic, financial and statistical operations. Not surprisingly, spreadsheet applications play significant role in decision-making processes in organizations, thus making spreadsheet errors serious threats. Reports from field audits show that spreadsheet errors may cause companies to lose millions of dollars annually. One effective and simple way of helping users to avoid introducing mistakes in their spreadsheets is data validation. Indeed, most spreadsheet applications provide a wide range of built-in functions to restrict the type of the input data or the range of valid values entered into a cell. However, in most of them, the underlying design decisions governing how data input should be entered in a spreadsheet are not explicitly visible to its users. Hiding data validation rules from users may hinder the comprehensibility and the usability of a spreadsheet, thus increasing the risks of entering incorrect data input. To assist end-user programmers in explicitly expressing validation rules in spreadsheets, we propose the SpreadSheet Validation Language (SSVL). We conducted a user study to assess the effectiveness of SSVL. The results show that users using SSVL are faster and more productive in tasks involving the comprehension of data validation rules. This is a promising result suggesting that SSVL can actually improve the usability of spreadsheets.

CCS Concepts

•Software and its engineering → Domain specific languages; Software testing and debugging;

Keywords

Data Validation, Spreadsheets, Domain-specific Languages

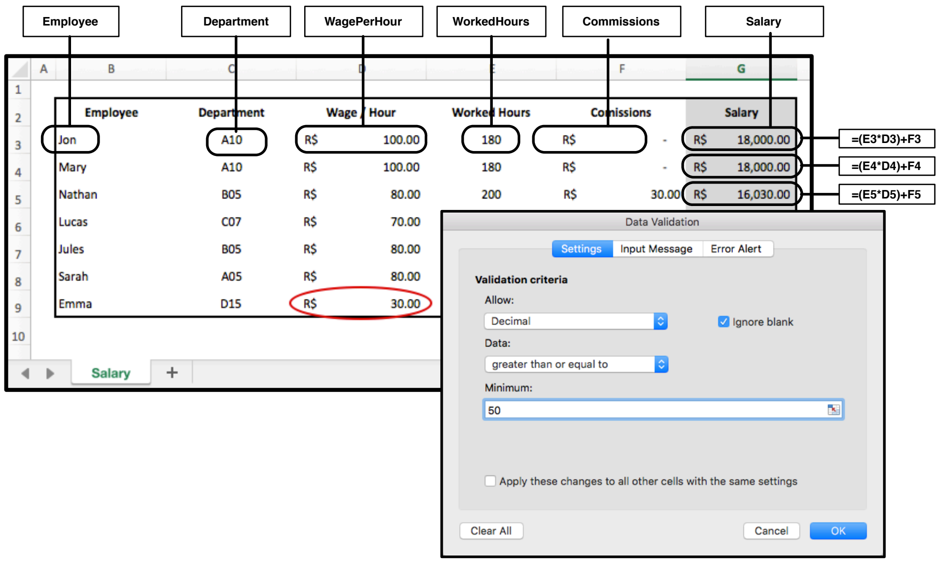
1. INTRODUCTION

Spreadsheet applications (e.g., Microsoft Excel, Google Sheets) have become a common platform for end-users create their own programs [15]. End-user programmers, or simply end-users, are people who are not professional programmers, but use spreadsheet applications as a platform to build solutions in their domain fields [18], such as science, engineering, insurance and banking, among others. Nowadays, spreadsheet applications play important role in decision-making processes in several organizations, thus making spreadsheet errors serious threats to organization success [5][10]. However, end-user programmers and organizations are overconfident in the correctness of their spreadsheet [14] and often do not employ quality control measures commonly used in regular programming [11][8]. These conditions foster a fertile environment for the occurrence of errors in spreadsheet programs. Not surprisingly, the study conducted by Panko [13] observed 94% of spreadsheet programs contain errors. To make matters worse, reports from field audits show that spreadsheet errors may cause companies to lose millions of dollars annually [4][16][13].

Spreadsheet programs are built on top of two basic concepts: cells are used as variables and functions are used to express relations between variables. Thus, mistakes in spreadsheet programs can be introduced by either entering incorrect data input into a cell, or by defining an incorrect relation between cells. One effective and simple way of helping users to avoid introducing mistakes in their spreadsheet programs is data validation [17]. Data validation is the process of ensuring that input data are accurate according to a set of validation rules. A validation rule is a logical sequence of operators and operands performing tests on input data to assure their validity. Indeed, most mainstream spreadsheet applications provide a wide range of built-in functions to restrict the type of the input data or the range of valid values entered into a cell. However, as observed by Caulkins et. al. [4], spreadsheets typically do not validate many kinds of human inputs, even though it is a recommended practice that software should validate its inputs [7].

While building spreadsheet programs, end-user programmers can easily enter data validation rules. For example, in Microsoft Excel and Google Sheets, end-user programmers create new validation rules by selecting one or more cells to validate and attaching to them the desired validation rule by means of a "Data Validation" dialog, as shown in Figure

Figure 1: The Salary Management Spreadsheet in MS Excel



Employee	Department	Wage / Hour	Worked Hours	Commissions	Salary
Jon	A10	R$ 100.00	180	R$ -	R$ 18,000.00
Mary	A10	R$ 100.00	180	R$ -	R$ 18,000.00
Nathan	B05	R$ 80.00	200	R$ 30.00	R$ 16,030.00
Lucas	C07	R$ 70.00			
Jules	B05	R$ 80.00			
Sarah	A05	R$ 80.00			
Emma	D15	R$ 30.00			

=(E3*D3)+F3
=(E4*D4)+F4
=(E5*D5)+F5

Validation criteria
Allow: Decimal
[✓] Ignore blank
Data: greater than or equal to
Minimum: 50
Apply these changes to all other cells with the same settings

1. However, validation rules created are not visible in the main interface of the spreadsheet program. In other words, the underlying design decisions governing how data input should be entered in a spreadsheet program are not explicitly visible to its users. Hiding data validation rules from users may hinder the comprehensibility and the usability of a spreadsheet program, thus increasing the risks of entering incorrect data input [3][8][10].

Comprehending how to properly use a spreadsheet program is an inherent complex task [11]. The ability to build a spreadsheet program by assigning small pieces of business logic to specific cells makes it difficult to get a global sense of the program structure. In order to get an overview of a spreadsheet program, users have to trace dependencies among cells. When validation rules are used, another layer of complexity is introduced in spreadsheet program comprehension, since these rules are hidden from users and scattered across different cells in spreadsheets. As a consequence, consulting and comprehending validation rules is still a time consuming task for users of spreadsheet programs. Users have to navigate across the entire spreadsheet and open for each cell the same "Data Validation" dialog end-user programmers use to create the rules. There is still no straightforward way of users getting a clear overview of which validation rules should be adhered in a given spreadsheet. As a consequence, users do not always possess the knowledge to properly use a spreadsheet program, which may ultimately lead to users entering invalid data in spreadsheet programs.

To assist end-user programmers in explicitly and modularly expressing validation rules in spreadsheets, we proposed the SpreadSheet Validation Language (SSVL), a tool-supported domain-specific language that seamlessly integrates to a spreadsheet application (e.g., Microsoft Excel or Google Sheets). SSVL allows end-users to declaratively express modular validation rules without using typical dialog-based data validation approaches. With SSVL, programmers express all validation rules in a single artifact, thus providing a unique and clear view of all validation rules associated to a given spreadsheet. We conducted a user study to assess the effectiveness of SSVL. The results show users using SSVL are faster and more productive in tasks involving the comprehension of data validation rules. This is a promising result suggesting that SSVL can actually improve the usability of spreadsheet programs.

The main contributions of this paper are:

- We present a language-oriented approach (SSVL) for data validation in spreadsheets which requires any special training. The language intents to offer substantial gains in expressiveness and ease of use compared with dialog-based approaches.
- We present a tool to support the implementation and automatic data validation according to SSVL approach. Although the tool implementation is based on Excel, its underlying concepts can be applied to data validation in spreadsheets build in any program (e.g., Microsoft Excel or Google Sheets).
- We assess the usability of data validation in Excel and SSVL. Preliminary results show that SSVL helps users to be faster when they are asked to observe all invalid data and judge why they are considered invalid or to explain why some specific set of data has cells values that are considered valid. Furthermore, the participants like the usage of the proposed language-oriented approach and rate it as most suitable than the traditional dialog-based approaches. However, in all case, users were not able to precisely judge if they are aware of which cells have no data validation set up.

The remainder of this paper is organized as follows. Section 2 presents a background related to data validation in spreadsheets. Section 3 presents the language-oriented approach and how the proposed concepts can be used to concretize some of its supporting ideas in a prototyping tool. Section 4 presents the user study on making data validation rules explicit in spreadsheet applications. Finally, Section 5 presents our conclusions and directions for future work.

2. BACKGROUND

In this Section we present a short background related to spreadsheets and data validation in spreadsheet applications.

2.1 Spreadsheets

Spreadsheet has been applied to solve complex problems for many professionals. Spreadsheet applications (e.g., Excel, Google Sheets) are probably the most used end-user programing environment on the planet. We consider end-user programmers any one who have little or no training at programming [15]. Indeed, in contrast to general purpose programming languages, it is possible to create very useful Spreadsheets without training at programming [11].

Yet based on a simple computation model, Spreadsheets applications are powerful environments. They are mainly composed of cells, which can contain values (numbers, texts, dates, times), references to other cells, or formulas. Cells are often disposed in a rectangular grid, composed of a quasi-infinity number of rows and columns. To refer to the contents of a cell, an unambiguous address has to be used. The most common form of cells address is the A1-style. Figure 1 illustrates a simple A1-style Spreadsheet. In this format, cells are referenced by its row and column position inside the Spreadsheet grid. For example, a cell at column B and row 3 would have the address B3. Groups of cells (i.e., cell ranges), can be also specified, and have a named. For example, the range of cells E3:E9 compose the group of cells named Employee.

From a design perspective, the content in spreadsheets is usually arranged in tables. In this case, cells might contain simple labels. In the example (Figure 1), cells B2, C2, D2, E2, F2 e G2 are Label Cells. They are used to name the column content. Label Cells can also be used as a form of documentation or to provide additional information about the overall Spreadsheet. Cells that contains any specific values and that are referenced by other cells are considered as Input Cells. For example, the currency value "R$ 100.00" in D3 and the integer value "180" in E3 are both inputs to the formula in G3 (=(E3*D3)+F3). Finally, the cells that contains formulas constitute the spreadsheet computation, and are trivially called Formula Cells. The cell G3 is a Formula Cell that computes the multiplication of the values placed in the Input Cells E3 and D3 plus the value placed in the cell F3. These kind of cells are the ones which hold all spreadsheet outputs. Considering the previous example, the Formula Cell G3 outputs the Salary earned in a month by a specific employee.

In summary, spreadsheets represent the logic, inputs and outputs, all at the same place. In this case, end-user programmers are able to create their programs by performing changes in any spreadsheet cells (i.e., Label Cells, Input Cells and Formula Cells). However, each kind of change have distinctive impacts on the spreadsheet. A change in a Label Cells, usually modifies the means of the value stored in near cells, while changes in Formula Cells produces modifications in the spreadsheet computation. Changes in Input Cells are often only the ones allowed to users and the ones that should be validated, mainly because spreadsheets, are in general used improperly or incorrectly, or without sufficient control [1][3]. Data validation is the process of ensuring that user inputs are accurate according to a set of validation rule. Next we overview how data validation is supported by Excel, which offers a dialog-based data validation approach.

2.2 Data Validation in Spreadsheets

When creating a spreadsheet, the end-user programmer has a mental model of how it should be used [17]. One way of expressing how users should input data in spreadsheets is data validations [1][17]. Data validations are usually specified in terms of conditional construct as validation rules. Such a validation rule consists of three parts: (i) an expression corresponding to the condition that has to be tested; (ii) an action chosen based on the outcome of the condition – if the condition is not satisfied, causes an error to be carried out; and (iii) an error message that is displayed when the input value is invalid.

With data validation, any cell should check for the validity of its content. This includes checks whether the data is strongly typed, has the correct syntax, is within length boundaries, or that numbers are correctly signed and within range boundaries. Any data that does not match should be rejected. In general, there are three category of validation rules in spreadsheets:

- **Data type**: Every cell can have a data type that restricts what values users can provide. For example, the set of cells about Hours Worked should accept only whole numbers, the Salary cells only decimals numbers greater than or equals to 900.00, and so on.
- **Limits**: Restrict the input data to a minimum and/or a maximum limit. For example, the cells range about Commissions can be set to accept a limit between R$ 00.0 and R$ 50.00.
- **Input masks**: End-user programmers can use an input masks (e.g., regular expression) to validate data by driving users to enter values in a specific format. For example, an input mask can force users to enter data in Department cells in accordance to a letter and two-digit format such as, A10.

In next subsections we overview data validation in Microsoft Excel. We illustrate how data validation is supported by such tool with a salary management spreadsheet (Figure 1). It is a very simple spreadsheet comprised of one sheet. The Salaries Sheet provides access to the employee's name, department, wage/hours, worked hours, commissions and total of incomes (salary).

2.2.1 Data Validation in Excel

In Excel, end-user programmers apply data validation to restrict the type of data and the values that others enter into a cell. Excel offers several types of data validation. End-user programmers can restrict data entry to: (i) whole numbers; (ii) decimal numbers; (iii) date and time; or (iv) texts. The numbers, date and time types can be restricted within limits (or range of date and time), while textual values have to respect a specific length or a regular expression. For example, consider the spreadsheet in Figure 1. The cells in the D3:D9 can have a validation rule that restrict the Employees Wage to be decimal values greater than or equals to R$ 50.00.

For adding any type of data validation, end-user programmers have to select one or a range of cells to validate and attach to them the desired validation rule via the Data Validation dialog. Figure 1 overviews how to add data validation to a cell or a range. In the Allow box, end-user programmers can configure the allowed type of data (e.g., whole number, decimal, text). In the Data box, they can select the type of restriction (e.g., between, greater or equals to). Finally, they have to enter the minimum, maximum, or specific value to allow. For example, consider the "Employees Wage" data validation (range D3:D9). To set a minimum limit of Wage, programmers have to select "greater than or equals to" in the Data boxing and enter the values 50.00 in the Minimum box. Excel also supports as value the return of a formula. The Commissions (range F3:F9) minimum limit could be set to a maximum limit of 10% of Employee Salary as the result of the formula "=F3*0.10".

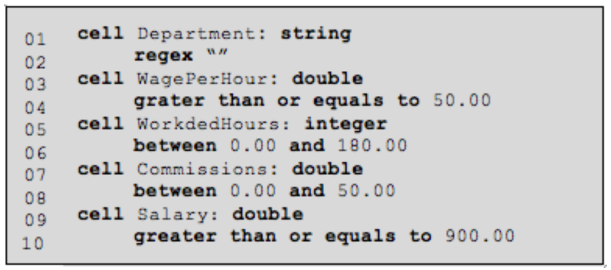
Also in the Data Validation dialog, an input messages can be set in order to help users to know who input data that is not valid. In the example, the message might be "The Employees Salary must be a decimal greater than or equals to R$ 900". The Error Alert option in the Data Validation dialog can be configured to check, whether the user-provided data breaks the respective validation rule. If so, the input is not accepted, and Excel displays an error message. Also, the Circle Invalid Data action allows users to audit spreadsheet by looking for incorrect data that may cause inaccurate calculations or results. Excel identifies cells that contain invalid data by displaying a red circle around them so that users can observe and correct any inconsistency. In Figure 1, the cell D9 is violating the respective data validation. As can be seen, the Employee Wage per Hour is a value less than 50.00.

Although be powerful and extremely complete, the dialog-based approach implement by Excel, (and also by Google Sheets), has some drawbacks that can hamper its use. Consider the problem of restrict the Employee Salary to values greater than or equals to R$ 900. The first challenge here is capturing validations rules in a disciplined way. The rule "greater than or equals to R$ 900" have to be applied uniformly in all cells in the column Salary. As the dialog-based data validation approach provided by Excel inevitably hides behind cells their attached data validation rules, set up data validation properly might become a difficult and error-prone task. Indeed, there is no easygoing or direct way to end-user programmers consult all existing validation rules in Excel. Second, it might be also very time consuming for spreadsheet users to get an overview of the underlying data validation rules and possess the knowledge to use the spreadsheet properly. In next Section we introduce SSVL, our proposed approach which intents to overcome some existing drawbacks in the dialog-based approach.

3. SPREADSHEET VALIDATION LANGUAGE

The SpreadSheet Validation Language (SSVL) is a domain-specific language [9] to explicitly and modularly express data validation rules in spreadsheets. The SSVL aims at solving the limitations imposed by data validation approaches implemented in current spreadsheet applications. In particular, these approaches express validation rules scattered across different cells in spreadsheets, and also hide these rules from the main view of spreadsheet programs. By providing notations and constructs custom-tailored towards the data validation domain [6][18], SSVL intents to offer substantial gains in expressiveness and ease of use compared

Figure 2: Data Validation Specification in SSVL



```
01  cell Department: string
02       regex “”
03  cell WagePerHour: double
04       grater than or equals to 50.00
05  cell WorkdedHours: integer
06       between 0.00 and 180.00
07  cell Commissions: double
08       between 0.00 and 50.00
09  cell Salary: double
10       greater than or equals to 900.00
```

with dialog-based approaches, without substantial efforts in specific training. Once data validation rules are explicitly expressed in a specific artifact, the task of consulting and comprehending them should become less time consuming and error-prone. Therefore, users might be more aware of the rules to enter valid data in a spreadsheet with a reduced effort and just be more effective and productive in their day-life work.

Validation rules in SSVL contain: (i) the name of a continuous or non-continuous range of cells that should be validated; (ii) the expected data type (integer, double, data, time, or string); and (iii) an expression specifying the condition be tested. Figure 2 illustrates the data validation rules written following the SSVL notation.

Figure 2 present data validation rules for the for a Salary management spreadsheet (see Figure 1). In SSVL, the cell constructor is the container that encompasses the validation rules for a specific range of cells referenced by its name. For example, the construction "cell WagePerHour: double grater than or equals to 50.00" (line 3-4) express that all values in cells inside the named range "Salary" (D3:D9) must accept only double values greater than or equals to 50.0. Table 1 lists all SSVL expressions and how they are typically used. End-user programmers can use in expressions literal values, such as: integer numbers; double numbers; dates and times; string; and regular expressions. Regular expressions and strings, in contrast to numbers, dates and times values, have to be place within quotations.

3.1 Tool Prototype

We developed a tool prototype for supporting the specification and automatic data validation of spreadsheets according to SSVL approach. Our prototype tool is implemented as an Eclipse plugin based on Xtext [1], a framework for development of domain-specific languages. The Xtext framework provides a powerful grammar language for the description of textual languages, and an infrastructure, including: (i) typechekers; (ii) validators; and (iii) editing support with syntax highlight and autocomplete capabilities.

We built our data validation rule engine over the Easy Rules [2] framework. The Easy Rules provides adequate API abstractions to create rules with conditions and actions, and an engine to run them to evaluate the conditions and execute the data validation reporting actions. The data validation actions interfaces with the Apache POI [3] to open, read, and change the style of the spreadsheet to be validate. The

[1]eclipse.org/Xtext/
[2]easyrules.org
[3]poi.apache.org

Table 1: The purpose of SSVL expressions

Expression	**Purpose**
|NOT| EQUALS TO **<VALUE>**	Test for cell value equals or not equals to the provided value
|NOT| BETWEEN **<VALUE>** AND **<VALUE>**	Test for cell value between or not between to the provided value
GREATER THAN OR EQUALS TO **<VALUE>**	Test for cell value greater than or equals to the provided value
LESS THAN OR EQUALS TO **<VALUE>**	Test for cell value less than or equals to the provided value
REGEX "**<PATTERN>**"	Matches pattern strings
STARTS WITH "**<VALUE>**"	Matches if the cell value starts with the provided substring
ENDS WITH "**<VALUE>**"	Matches if the cell value ends with the provided substring
LEN **<VALUE>**	Test for string with length equals to the provided value

Apache POI offers a Java APIs for manipulating files formats based upon the Office Open XML standards and Microsoft's OLE 2 Compound Document format. Therefore, although the tool prototype is based upon Excel, its underlying concepts can be applied to data validation in spreadsheets build in other spreadsheet applications, like Google Sheets.

4. USER STUDY

We conducted a user study to investigate whether and how different data validation strategies (dialog-based versus language-based) influence users comprehension of validation rules in spreadsheets. In this study, we used the Excel spreadsheet application as representative of the dialog-based strategy, whereas SSVL represents the language-based strategy. Next, we present the study settings and the observed results.

4.1 Goal and Research Questions

In this study, our goal is to observe to what extent users can understand an existing set of data validation rules expressed using two different approaches. Following the Goal-Questions-Metrics (GQM) [2], we defined the goal for our user study as follows:

To characterize the effects of dialog-based and language-based validation strategies on user comprehension from the viewpoint of users of medium-sized spreadsheet programs in the context of undergraduate students.

Then, we refined our research questions as follows.

- **RQ1**. Will user be more accurate in understanding data validation with the aid of Excel or SSVL?
- **RQ2**. Will user be faster in understanding data validation with the aid of Excel or SSVL?
- **RQ3**. Will user be more satisfied with their experience in Excel or SSVL?

Based on the research questions we defined the metrics to be collected. To answer RQ1 and RQ2 we use two metrics: (i) the number of correctly answered the questions in a questionnaire; and (ii) the time taken to answer the questionnaire. To answer the RQ3 we applied the USE questionnaire [12] as a measurement of user satisfaction experience.

4.2 Study Procedures

Our study comprised three steps. First, participants attended an introductory tutorial giving an overview of data validation strategies in spreadsheets. The introductory tutorial was aimed to ensure that all participants had the same basic knowledge about data validation in spreadsheets using SSVL and Excel. Next, we asked participants to answer a questionnaire regarding data validation understanding. This activity consists of a questionnaire about existing spreadsheets and their respective data validation rules. We recorded the time taken for each participant to answer each question as well as their response, deriving the number of correctly answered questions. Finally, we asked participants to answer a USE questionnaire [12] comprising questions about tool's usability, satisfaction, and ease of use.

We designed our user study with the Latin square in order to control the use of the data validation strategies by groups of participants. The size of the Latin square is 2x2, in which the x-axis is the participants and the y-axis is the spreadsheets. The Latin square design gave us a random allocation of Excel and SSVL in such a way that they were used once by each participant (row) and once with each spreadsheet program (column). This design avoids some effects such as learning throughout observations.

The study involved twelve participants, all undergraduate students from the Federal University of São João del-Rei. Therefore, we were able to replicate the 2x2 Latin square six times, obtaining 24 independent observations.

4.3 Spreadsheets and Problems

The data validation problems were encoded into two spreadsheets: Stock Control SS and Payroll SS. Participants were given a brief overview of each spreadsheet. We explained the functionalities of each spreadsheet, so that all participants had exactly the same information about the spreadsheets. The Stock Control SS controls the stock of restaurants, while the Payroll SS calculates employee's salary based on incomes and deductions. There are sixteen cells with invalid data in Stock Control SS and seventeen cells with invalid data in Payroll SS. Both spreadsheets were laid out such that they require scrolling, to ensure that not all invalid data would be visible at once.

To facilitate comparing the impact of data validation approaches, we violated data validation rules that would expose only numeric errors. We introduce the values that violates the following kinds of rules: data type; equals; between; greater than or equals to; and less than or equals to. To avoid biasing results, we introduced the invalid data uniformly across spreadsheets, assuring more than one invalid data for each set of cells. Moreover, in order to observe the ability of users to identify which cells do not have data validation rules, we remove from both spreadsheets a set of cells and their respective data validation rules.

Figure 3: Boxplot - Correctly Answered Questions

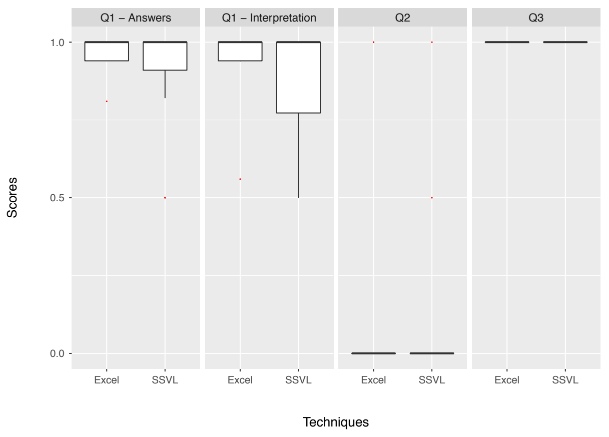
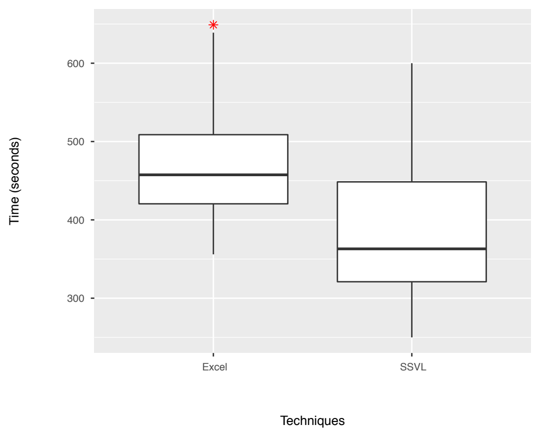


Figure 4: Boxplot - Time Taken to Answer



4.4 Questionnaires

The questionnaire used in our study has three questions about the existence of data validations rules. These questions aim to quantify to what extent a user understands the existing data validation rules. Therefore, we designed questions to observe whether users are able to: (**q1**) identify all invalid data and judge why they are considered invalid (i.e., which validation rules they are violating); (**q2**) judge if they are aware of which cells have no data validation set up; and (**q3**) explain why some specific sets of data have cell values that are considered valid.

The participants were also asked to answer a questionnaire about their user-experience. This questionnaire is based on the USE questionnaire, and is composed of 30 seven-point Likert scale questions. For each question the participant assigned a score ranging from 1 (strongly disagree) to 7 (strongly agree) to a given statement regarding on of the three dimensions addressed by the USE questionnaire (usability, satisfaction and ease of use).

4.5 Results and Analysis

In this section, we first compare participant's performance regarding the understanding of data validation rules. We answer questions RQ1 and RQ2, and also discuss the results from the USE questionnaire.

4.5.1 *Correctly Answered Questions (RQ1)*

To answer our first research question, we analyzed the number of correct answers. The questions were analyzed over the perspective of partially correct answers. That is, we assigned a score for each participant considering answer as being partially correct. We calculated this score by dividing the number of correctly provided answers elements by the sum of all elements – hits and misses. We considered a miss a wrong element in the answer or the absence of a correct one.

Table 2 presents the descriptive statics of correct answers. We observe that, in general, the means are almost equals. It shows that there is no explicit difference between Excel and SSVL groups. Figure 3 shows a box plot for the scores obtained by participants in each question (Q1, Q2 and Q3). All participants working with Excel and SSVL identified similar percentages of the invalid data (Q1 – Answers), and were capable of judging why they were considered invalid (Q1 – Interpretation). Moreover, in all cases, they were also able to correctly answer why some specific sets of data have cell values that were considered valid (Q3). However, only two participants were aware of which cells have no data validation rules (Q2). In this case, we can confirm that neither Excel nor SSVL are able to help users on the task of identifying which sets of cell do not have data validation rules. Yet, in general, the results suggest that a language-oriented approach seems to be easy to learn and equally powerful as the common practice employed by the main spreadsheet programs (e.g., Excel and Google Sheets).

The statistical test corroborates our observation. First, we used the Shapiro-Wilk test to check for normality and results indicated the sample does not follow a normal distribution ($p < .05$). Therefore, Kruskal-Wallis test was conducted to determine whether there was a difference in total score considering correct answers from participants assisted by Excel and SSVL. Results indicated there was no statistically significant difference, df = 23, $p > .05$.

Table 2: Descriptive Statistics – Correctly Answered Questions

	Excel	SSVL
Mean	2.16	2.14
Min	1.56	1.50
Max	3.00	3.00
Median	2.00	2.00
Stdev	1.39	1.81

Table 3: Descriptive Statistics – Time Taken to Answer Questions

	Excel	SSVL
Mean	475.9	387.2
Min	356.0	250.0
Max	649.0	600.0
Median	457.5	363.0
Stdev	92.7	102.0

4.5.2 *Time Taken to Answer Questions (RQ2)*

We analyzed the time (in seconds) taken to answer all the questions. We avoided analyzing only the time of correct answers because participants had a distinct number of correct answers, which makes impracticable to compare their total times. Moreover, as we presented in Section 3.5.1, this choice is also justifiable because participants were, in general, successful in answering the questionnaire. Table 3 presents the descriptive statics of the time taken by participants to answer the questions.

Figure 4 shows a box plot for the time spent by participants in each tool (Excel and SSVL). Analyzing the box plot, we notice that SSVL allows participants to comprehend data validation rules faster than in Excel. The difference between groups also indicates that users without training on language-oriented approaches are able to quickly understand how to read the validation rules specified in SSVL. Indeed, participants using SSVL present a much lower variance. The lower time achieve by SSVL is credited to the fact that the information required for understanding data validation rule is explicitly shown in a single artifact. That is, SSVL eliminates the need for navigating through many different cells across the spreadsheet.

Results of the Shapiro-Wilk test indicate that the sample follows a normal distribution ($p > .05$). Results of an Anova test indicate there was a statistically significant difference ($F = 5.38$, $p < .05$) between the time spent by participants assisted by Excel or SSVL. Therefore, the statistical test corroborates our observations.

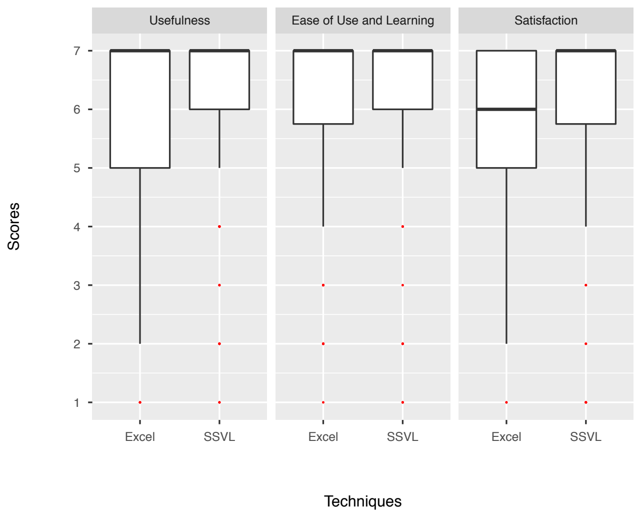
4.5.3 *User Satisfaction Questionnaire*

Finally, participants were requested to answer a series of questions regarding three aspects of SSVL: (i) usability; (ii) satisfaction; and (iii) ease of use. Analyzing the boxplot in Figure 5 we can derive useful insights. The highest averages of each dimension show that participants consider SSVL to be more useful, easier to use and learn, and they are more satisfied with SSVL than with Excel. Indeed, statistical test corroborate our observations. Results of the Shapiro-Wilk test indicate that the sample does not follow a normal distribution ($p < .05$). Therefore, Mann-Whitney U test was conducted to determine whether there was a difference in total score considering user experience from participants assisted by Excel and SSVL. Results of that analysis indicated that there was a statistically significant difference, $w = 58804$, $p < .05$.

We also analyzed the three most positive and the three most negative aspects of each tool. We consider the most positive aspects the one with highest mean and lowest standard deviations. On the other hand, we consider the most negative aspects the ones with lowest mean and highest standard deviations. The most positive aspects of Excel are: its usefulness, it is quick to learn how to use it, and it is easy remembering how to use it. Yet, the most negative aspects of Excel are that it does not help users to be more effective, it does not help users to be more productive and it does not do everything the users would expect it to do. In fact, the results corroborate with our claim that although dialog-based approaches are easy to learn and use, they do not help users to be more effective and productive in their daily activities.

Analyzing the three most positive aspects of SSVL we could observe that it is flexible, it works the way users want

Figure 5: Boxplot - User Satisfaction Questionnaire



it to work, and it does not present inconsistencies as users use it. Yet, the three most negative aspects of SSVL are that it makes the things users want to accomplish easier to get done, both occasional and regular users would like it, and users can use it without written instructions. Corroborating with the observed results (Section 3.5.1 and 3.5.2), we can assume that users are in general satisfied and agree that the language design is mature enough to help them to be more productive. However, as SSVL implements a non-traditional data validation approach, the users agree that it is harder to learn and use than dialog-based approaches.

The analysis of the user experience questionnaire speaks in favor of a language-oriented approach. In general, participants using SSVL were more satisfied than those using Excel. The use of a dialog interrupts the workflow of spreadsheet users, because they have to open a ”Data Validation“ dialog window and read the according data validation rule expressed by means of several combo and text boxes. Since interruptions can be very expensive, they should be avoided whenever possible. Our results indicate that SSVL can limit the amount of interruptions caused by dialog usage.

4.5.4 *Threats to Validity*

Threats to validity can be discussed into internal and external validity. In this section, for each category, we list the possible threats and the procedure we took to alleviate their risk.

Internal Validity. Since there is no standardized assessment of data validation in spreadsheets, we applied our own questionnaire to our participants. Therefore, we cannot be sure how well we captured the metrics correct answers and time. To reduce this threat, we developed a questionnaire based on literature review about empirical and user studies in spreadsheets [18]. Likewise, to minimize threats related to confounding or poor design questions, we proceed by answering questions from participants as they were emerging. To avoid biasing the experiment results, we resort the explanations to what was demonstrated during the introductory tutorial and to what clarifications were absolutely necessary.

We used numerous mechanisms to deal with small samples. First, we use a Latin square design to apply both tools

(Excel and SSVL) to all subjects without side effects, such as learning. Moreover, we we applied variants of significant tests that were developed to deal with small samples. Finally, Cohen's test indicates from medium to large effects. We also did not correct the response times for wrong answers because our sample is small. Indeed, there is no consensus in literature whether the use of efficiency measure as combination of correctness of answers and response time may lead to falsely accepting or rejecting a hypothesis. Since answers results deviated considerably towards the maximum value that is 3, we consider that this thread is insignificant in our study.

External Validity One external risk is caused by spreadsheets. The selected ones might not be representative of all practices. To reduce this risk, we selected one spreadsheet from Excel's sample and the another is a real-life stock control spreadsheet. Moreover, although the size of the chosen spreadsheets is limited, this characteristic allowed us to obtain more consistent results that could be interpreted. Another threat can be caused by our sample, which consisted mostly of undergraduate students. However, the benefits of recruiting students are twofold: (i) they have relatively little experience with data validation in spreadsheets, which supported us to observe that any of both tools required specific level of special training; (ii) their heterogeneity helps to address the biasing the study results to a specific subgroup of professionals.

5. CONCLUSIONS

Spreadsheet applications play significant role in decision-making processes in organizations [5]. Thus spreadsheet errors are serious threats as reported by several works [13][16]. One effective and simple way of helping users to avoid introducing mistakes in their spreadsheets is data validation [17]. However, most of spreadsheet applications, nowadays hide from users the underlying design decisions governing how data input should be entered in a spreadsheet. In this paper, we present a language-oriented approach for data validation in spreadsheets that requires no special training. The language intents to offer substantial gains in expressiveness and ease of use compared with dialog-based approaches offered by most of mainstream spreadsheet applications.

We assess the usability of data validation in Excel and SSVL. Preliminary results show that SSVL helps users to be faster when they are asked to observe all invalid data and judge why they are considered invalid or to explain why some specific set of data has cells values that are considered valid. Furthermore, the participants like the usage of the proposed language-oriented approach and rate it as most suitable than the traditional dialog-based approaches. However, in all case, users were not able to precisely judge if they are aware of which cells have no data validation set up. In feature work, we plan to evolve the language expressiveness as well as integrating it with existing spreadsheet testing tools in order to support the generation of test inputs based on validation rules. Finally, we also plan to concretize our proposed approach as two Add-Ins, one for Excel and another to Google Sheets.

6. REFERENCES

[1] R. Abraham, M. Erwig, S. Kollmansberger, and E. Seifert. Visual specifications of correct spreadsheets. In *Symposium on Visual Languages and Human-Centric Computing*, 2005.

[2] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[3] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, and C. Wallace. End-user software engineering with assertions in the spreadsheet paradigm. In *25th International Conference on Software Engineering*, 2003.

[4] J. P. Caulkins, E. L. Morrison, and T. Weidemann. *Do Spreadsheet Errors Lead to Bad Decisions?*, pages 44–62. IGI Global, 2008.

[5] J. P. Caulkins, E. L. Morrison, and T. Weidemann. Spreadsheet errors and decision making: Evidence from field interviews. *Journal of Organizational and End User Computing*, 19(3):1 – 23, 2017.

[6] D. M. Groenewegen and E. Visser. Integration of data validation and user interface concerns in a dsl for web applications. *Software & Systems Modeling*, 12(1):35–52, 2013.

[7] M. Howard, D. LeBlanc, and J. Viega. *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them.* McGraw-Hill, New York, 1 edition, 2010.

[8] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa. Avoiding, finding and fixing spreadsheet errors – a survey of automated approaches for spreadsheet {QA}. *Journal of Systems and Software*, 94:129 – 150, 2014.

[9] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. VoÌĹlkel. Design guidelines for domain specific languages. In *9th Workshop on Domain-Specific Modeling*, 2009.

[10] J. Lawrance, R. Abraham, M. Burnett, and M. Erwig. Sharing reasoning about faults in spreadsheets: An empirical study. In *Visual Languages and Human-Centric Computing*, 2006.

[11] B. R. Lawson, K. R. Baker, S. G. Powell, and L. Foster-Johnson. A comparison of spreadsheet users with different levels of experience. *Omega*, 37(3):579 – 590, 2009.

[12] A. M. Lund. Measuring usability with the use questionnaire. In *STC Usability SIG Newsletter*. 2001.

[13] R. R. Panko. Spreadsheet errors: What we know. what we think we can do. In *Symp. of the European Spreadsheet Risks Interest Group*, 2008.

[14] R. R. Panko. *Two Experiments in Reducing Overconfidence in Spreadsheet Development*, pages 131–149. IGI Global, 2008.

[15] R. R. Panko and D. N. Port. End user computing: The dark matter (and dark energy) of corporate it. In *45th Hawaii Inter. Conf. on System Sciences*, 2012.

[16] S. G. Powell, K. R. Baker, and B. Lawson. A critical review of the literature on spreadsheet errors. *Decision Support Systems*, 46(1):128 – 138, 2008.

[17] C. Scaffidi, B. Myers, and M. Shaw. Topes: Reusable abstractions for validating data. In *30th International Conference on Software Engineering*, 2008.

[18] C. Scaffidi, B. Myers, and M. Shaw. *Fast, Accurate Creation of Data Validation Formats by End-User Developers.* 2009.