IS-EUD 2017 6th international symposium on end-user development

Javed Vassilis Khan
Iris Soute
Antonella De Angeli
Antonio Piccinno
Andrea Bellucci

Editors

IS-EUD 2017

6th International Symposium on End-User Development
Extended Abstracts

Work-in-progress, Workshop and Doctoral Consortium

Eindhoven University of Technology, The Netherlands.
June 13-15, 2017

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-4317-5

Table of Contents

Workshop SEMS'17

Work-in-progress

Doctoral Consortium

Preface

This volume includes extended abstracts describing work-in-progress, doctoral consortium and the workshop that were presented during the 6th International Symposium on End-User Development (IS-EUD 2017) which was held in Eindhoven, the Netherlands on June 13-15, 2017.

IS-EUD is a bi-annual event that gathers researchers interested to extend our knowledge about how to design end-user development technologies and to provide scientific accounts of phenomena surrounding end-user development practices. IS-EUD cuts across application areas such as ubiquitous and wearable computing, online communities, domotics, robotics, games, etc.

IS-EUD 2017 in Eindhoven invited contributions on the topics of empowerment and materiality, on how EUD technologies can empower end-users to magnify their reach and control over the physical world, to allow them to engage actively in societal trends and transformations. The theme of the conference was designated as "*that was business this is personal*", aiming to emphasize the personal involvement and engagement of end-users, the application of end-user programming beyond the professional environment looking also at discretionary use of technologies. Papers and submissions in all categories addressed this specific theme as topics that have been traditionally covered by the broader area of end-user development such as domain specific tools, spreadsheets, and end user aspects.

This volume was put together by the members of the organising committee responsible for the works-in-progress, doctoral consortium and workshops programme. We are grateful for the help of all members of the organizing committee and the reviewers for their voluntary work.

Eindhoven, June 2017

Javed Vassilis Khan
Iris Soute
Antonella De Angeli
Antonio Piccinno
Andrea Bellucci

Organising Committee

Conference Chair
Panos Markopoulos, Eindhoven University of Technology, the Netherlands

Program Chairs
Simone Barbosa, Pontifical Catholic University of Rio de Janeiro, Brazil
Fabio Paterno, Consiglio Nazionale delle Ricerche – ISTI, Pisa, Italy

Short Papers
Simone Stumpf, City University London, UK
Stefano Valtolina, Università degli Studi di Milano, Italy

Work-in-Progress
Javed Vassilis Khan, Eindhoven University of Technology, the Netherlands
Iris Soute, Fontys University of Applied Sciences, Eindhoven, the Netherlands

EUD-Demonstrations
Barbara Rita Barricelli, Università degli Studi di Milano, Italy
Jesús Muñoz-Alcántara, Eindhoven University of Technology, the Netherlands

Industrial Liaison
Dima Aliakseyu, Philips Lighting, The Netherlands

Workshops
Andrea Bellucci, Universidad Carlos III de Madrid, Spain

Doctoral Consortium
Antonella De Angeli, University of Trento, Italy
Antonio Piccinno, University of Bari, Italy

Communications and Publicity
Bruno Azevedo Chagas, Pontifical Catholic University of Rio de Janeiro, Brazil
Nikolaos Batalas, Eindhoven University of Technology, the Netherlands
Giulio Galesi, Consiglio Nazionale delle Ricerche – ISTI, Pisa, Italy

Local Arrangements
Rosalinde Kennis, Eindhoven University of Technology, the Netherlands
Jesús Muñoz-Alcántara, Eindhoven University of Technology, the Netherlands

SEMS'17 - 4th International Workshop on Software Engineering Methods in Spreadsheets

Jácome Cunha[1] and Birgit Hofer[2]

[1] Universidade NOVA de Lisboa
jacome@fct.unl.pt
[2] Graz University of Technology
bhofer@ist.tugraz.at

Abstract. Spreadsheets are heavily used in industry as they are easy to create and evolve. Initially, they are often small and simple, but, over time, they can become very complex. In many ways, spreadsheets are similar to "professional" software: both concern the storage and manipulation of data, and the presentation of results to the user. But unlike in "professional" software, activities like design, implementation, and maintenance in spreadsheets have to be undertaken by end users, not trained professional developers. This makes applying methods and techniques from other software domains a challenging task.
The role of SEMS is to provide an annual event where researchers can meet and exchange their ideas. SEMS serves as a platform for early feedback for new techniques and tools. The SEMS program includes a keynote, the presentation of short and long research papers, tool demonstrations, and a discussion session. The intended audience is a mixture of spreadsheet researchers and professionals.

1 Objective and Scope

Motivation. The number of spreadsheet users is approximately four times larger than the number of software developers. Thus, spreadsheets are a very interesting research topic with a high potential impact, in particular on the industry where spreadsheets are widely used. As a consequence of this importance the end-user development community has dedicated a significant part of their research to spreadsheets. For instance, the EUSES consortium (http://eusesconsortium.org) aimed at supporting "End Users Shaping Effective Software".

Spreadsheets suffer from problems like errors, bugs, smells, and cloning. Unfortunately, spreadsheet users are typically not trained in programming or other software development techniques. Hence they cannot directly apply conventional software development methodologies to aid themselves in the course of their work with spreadsheets. This provides researchers a scope for initiatives utilizing conventional software development methods to aid spreadsheet users in creation, comprehension, maintenance, evolution, and re-engineering of spreadsheets, tailoring such techniques for end-user development. Nevertheless, for this process to work, spreadsheet users from the industry need to communicate what real challenges they face, and provide feedback on the research initiatives and solutions

proposed to them. A workshop provides a good platform for such an exchange of ideas and opinions. This led to the conception of SEMS in 2014 and SEMS'17 is intended to continue the tradition and success forward.

Objectives. The overall objective of the SEMS workshop is to further explore the applicability of software development methods to spreadsheets. In particular, this workshop addresses the following goals:

1. Obtain a better understanding of the role of spreadsheets in industry and government organizations. What factors contribute to their widespread use, and what are their weaknesses and challenges?
2. Discuss limitations of spreadsheets. Spreadsheets were not designed to act as software systems, but they often used as such. We will discuss new methods and techniques to overcome limitations spreadsheets currently have.
3. Explore and share new or improved methods, techniques, and tools for spreadsheets. We will discuss what properties spreadsheets should have and how methods from "professional" software development can help end-users.

Intended audience. SEMS'17 is primarily intended for academic researchers involved in spreadsheet related research, in particular related to application of software development methodologies to spreadsheets. The past SEMS workshops have shown that the academic spreadsheet research community is lively, stable, and able to continuously make significant contributions. We also welcome practitioners from the industry interested in usage or improvement of spreadsheet technology, or sharing their experiences, especially from the user's point of view. This will ensure that the advances in research will be discussed with an increased focus on industrial relevance.

2 SEMS's History

- **SEMS'16** (http://spreadsheetlab.org/sems-16/) in Vienna, Austria, July 4th, 2016; co-located with STAF, 9 submissions, 4 long and 4 short were accepted for publication, 25 workshop attendees; keynote speaker: Sumit Gulwani, Research Manager & Principal Researcher at Microsoft, Redmond
- **SEMS'15** (http://spreadsheetlab.org/sems15/) in Florence, Italy, May 18th, 2015; co-located with ICSE, 16 submissions, 6 long and 5 short were accepted for publication, 23 workshop attendees; keynote speaker: Carlos Otero, Microsoft Excel program manager
- **SEMS'14** (http://spreadsheetlab.org/sems-14/) in Delft, The Netherlands, July 2nd, 2014; co-located with EuSpRIG's annual conference, 15 papers were submitted and all were accepted for publication, 30 workshop attendees

3 Organization

Organizers

- Jácome Cunha, NOVA LINCS, Universidade NOVA de Lisboa, Portugal
- Birgit Hofer, Graz University of Technology, Austria

Steering Commmitee

- Felienne Hermans, Delft University of Technology, The Netherlands
- Richard Paige, University of York, UK
- Peter Sestoft, IT University of Copenhagen, Denmark

Program Committee

- Rui Abreu, IST, University of Lisbon, Portugal
- Shing-Chi Cheung, The Hong Kong University of Science and Technology, Hong Kong, China
- Martin Erwig, Oregon State University, USA
- Joo P. Fernandes, Universidade de Coimbra, Portugal
- Felienne Hermans, Delft University of Technology, The Netherlands
- Bennett Kankuzi, North-West University, South Africa
- Daniel Kulesz, University of Stuttgart, Germany
- Jorge Mendes, Universidade do Minho, Portugal
- Richard Paige, University of York, UK
- Sohon Roy, Delft University of Technology, The Netherlands
- Peter Sestoft, IT University of Copenhagen, Denmark
- Leif Singer, Automattic Inc., USA

4 Workshop Format

The workshop starts with a keynote and a subsequent discussion. The following paper sessions allow the authors a maximum of 20 minutes for presenting their papers. 10 minutes are reserved for questions as we intend to foster discussion. In the last part of the workshop, a 90 minutes slot is reserved for the general discussion. The workshop attendees will be asked to state their opinion about the maturity of this research field and what open research topics they have identified. We aim to foster interaction and discussion throughout the workshop instead of having a 'one-way' presentation-only event.

5 Additional Material

More information can be found on http://sems2017.ist.tugraz.at/.

Systematic Programming in a Spreadsheet

Aleksy Schubert, Jacek Sroka*, and Jerzy Tyszkiewicz

Institute of Informatics, University of Warsaw, Poland
{alx,sroka,jty}@mimuw.edu.pl

Abstract. We demonstrate the first systematic, universal method of expressing runs of imperative programs by spreadsheet formulas. Programs which compile to the imperative fragment of Dalvik bytecode can be translated to spreadsheet data, which is in turn interpreted by suitable, universal formulas in a plain spreadsheet. The simulated program may use symbol table and recursion. The worksheet with formulas which interpret the translated bytecode can be therefore seen as a Dalvik virtual machine with spreadsheet as its hardware layer. As a result, spreadsheet can perform algorithmic computations without the need for vendor dependent scripting languages such as VBA, and this way avoid security threats they bring and circumvent compatibility problems.

1 Introduction

It was observed already by Casimir in 1992 [1] that spreadsheet formulas can describe typical algorithms, and therefore may have universal computing power. The author's conclusion was, however, that spreadsheets were intrinsically uninteresting. The spreadsheet paradigm of computation as such was not much studied since then. Recently a few papers on this topic appeared, e.g., in [9], where it was demonstrated that the SQL can be expressed by a subset of spreadsheet functions, common to all major spreadsheet tools. A few algorithms beyond SQL: transitive closure, breadth first search and depth first search on graphs, have been shown there to be implementable in spreadsheets.

Except that, there are almost no formula based algorithms in spreadsheet programming and no libraries of such algorithms — with the exception of numerical and statistical algorithms, see e.g., [3]. Methodologies of *Excel* programming in the financial modeling are focused more on avoiding errors in programming of long sequences of relatively simple steps than on algorithms *per se* [4,10].

In this paper we move from the observation that spreadsheet formulas have universal computing power to a demonstration that they indeed can uniformly express algorithms written in a modern, general purpose programming language, including the necessary data structures.

1.1 Our Contribution

We design a general computation mechanism in spreadsheets. We do so by expressing imperative language constructs using spreadsheet formulas.

* Sponsored by National Science Centre with decision DEC-2012/07/D/ST6/02492.

The language we can translate into spreadsheet is a sufficiently expressive fragment of Dalvik bytecode, known from Android OS. A program written in this fragment is translated, following the Harvard architecture, into: (i) spreadsheet formulas, which create a kind of virtual machine, called Spreadsheet Virtual Machine (SpVM), capable of executing the spreadsheet representation of the bytecode and (ii) spreadsheet data which encodes the bytecode itself. The translation permits the use of the most common programming structures: function call stack handling recursion, and symbol table (also known as dictionary), which provides and subsumes the functionality of arrays.

1.2 Potential usecases

The first possible application is to deliver computing power of user defined functions (UDFs) without using any scripting language. We expect that many tasks realized with UDFs might be replaced by components derived from our tool. There are many reasons which might prompt institutions to avoid scripting in their spreadsheets: security concerns [2], script compatibility problems between diverse spreadsheet platforms, using spreadsheets which do not support scripting at all: mobile, online, SAP's spreadsheet in Xcelsius and SAP Dashboards modules, etc. In each such case, a necessary complex computation could be expressed by our construction. Indeed, our interest in expressing complicated computations by pure-formula spreadsheets stems in part from our work on multi-platform spreadsheets, in which scripting is excluded for the sake of portability [8].

We can also imagine using our tool in computer science and computer engineering education. Indeed, it allows one to create and analyse, using spreadsheet toolset, complete runs of Dalvik programs, whether to get an insight into low-level programming and virtual machine construction, code optimization performed by compilers, or algorithm analysis.

1.3 Prior Research

There has been very limited amount of earlier research concerning expressing algorithms in spreadsheets. Apart from the already mentioned [1] and numerous publications concerning statistical and numerical algorithms, these are [11] and [9], which implement SQL, linearithmic sorting and two graph traversal algorithms by spreadsheet formulas. In earlier works [7,6] spreadsheets were used as an environment to animate algorithms, for teaching purposes. A fun project implemented a Turing machine in Excel [5].

2 Dalvik Bytecode

Dalvik bytecode is a register-based low-level language, designed to be executed by Dalvik execution platforms available on modern Android machines.

With this format of executable code we can work with Java as the source language. However, we did not want to model its object-oriented features as

this would make the presentation of our solution prohibitively complex. We just model an imperative fragment sufficient to represent a single class together with its static methods. Our solution will work directly with any language that is compiled to the fragment of Dalvik bytecode we support.

```
public class GCD {
  static public int gcd(int n, int m) {
    int tmp = 0;
    while (m != 0) { tmp = m; m = n % m; n = tmp; }
       return n; }
public static void main(String[] args) {
        System.out.println(gcd(12, 8));  }
}

Class #XXX            –
  Class descriptor  : 'Lexamples/GCD;'
  ...
  |[077064] examples.GCD.gcd:(II)I
  |0000: const/4 v0, #int 0
  |0001: if-nez v2, 0004
  |0003: return v1
  |0004: move v0, v2
  |0005: rem-int v2, v1, v2
  |0007: move v1, v0
  |0008: goto 0001
  ...
```

Fig. 1. The Java source code for the Euclid's algorithm and the (human-readable representation of) Dalvik code to which it compiles.

Example 1 We demonstrate the translation from Dalvik to the spreadsheet and operation of the virtual machine through an example implementation of the Euclid's algorithm for the greatest common divisor. The Java code as well as its translation into Dalvik bytecode are presented in Fig. 1. We believe that the Java code is routine. Its Dalvik counterpart uses three registers v0, v1, v2, and exhibits a few bytecode instructions: **if-nez** is a conditional jump on non-zero, **goto** is an unconditional jump, **return** is a return form function with a value, **rem-int** computes a reminder and **move** copies values between registers.

3 Spreadsheet Virtual Machine

The notion of programming in spreadsheets we use is based on the mechanism of *filling*. The user creates a few formulas and fills them down to make any required number of rows. In this sense, the initial few cells constitute a program.

The above mechanism permits creating algorithms in spreadsheets and was used already in [1]. We have created a spreadsheet with two rows of functional formulas in its main VM worksheet, and the user should fill them into as many rows as necessary. Those rows correspond in one-to-one way to the consecutive states of the Dalvik virtual machine that would execute Dalvik bytecode. The two rows in particular handle propagation of the program counter that points to the currently executed instruction. As a result we can deal with complex but static representation of a program to be executed — a feature missing from the Casimir's account.

3.1 Main conceptual components

A virtual machine within spreadsheet is an inherently complicated structure. In this short paper we describe a few fundamental concepts, which are combined together to make it work. They can be used independently, too.

- Harvard architecture: bytecode instructions are translated into data of the spreadsheet and not into formulas, and instruction data is kept separate from the memory data.
- Using addressing mechanisms to move data elements.
- Modular construction, with organisational structures specifically dedicated to handling all significant functionalities such as: control flow, function evaluation, data structure handling and I/O.
- Built-in data structures: data table and stack (which handles recursion) keep their data in the form of an operation log. Consequently, the simulated Dalvik program can address an unbounded amount of data memory, even though each row in the VM worksheet represents one time instant and has a fixed number of columns.

3.2 Harvard architecture

The decision to translate bytecode to data prevents spreadsheet formulas from getting longer with the length of the Dalvik program.

The language of spreadsheet formulas is purely functional and all data structures it permits are immutable ones. On top of it, we implement an imperative language with mutable structures, hence we must create new copies of data items each time they change. Due to the organization of data in spreadsheets, we create a new copy of the whole state of the SpVM after each step of computation. By using Harvard architecture (as opposed to von Neumann architecture), we avoid multiplication of the instruction memory.

The complete SpVM functionality is realized by a spreadsheet organized into three worksheets: *VM worksheet* where the program is executed, *bytecode worksheet* that contains Dalvik bytecode translated to spreadsheet data, and *input worksheet* that contains input data of the program. The VM worksheet stores appropriately subsequent configurations of the virtual machine, the bytecode worksheet represents the code of the interpreted program and the input worksheet represents the input.

	A	B	C	D	E	F	G	H	I	J	K	L	O	P	Q	R	Y	Z	AA	AB
1	opcode_instruction	registers				nstruction coun		constatnt	arithmetics				Symbol Table API				Input definition			
2		v0	v1	v2	ret_val	type	jump_to	const-res	ar-arg1	ar-arg2	ar-op	ar-res	ST_id	ST_oper	ST_key	ST_item	IO-arg1	IO-arg2	IO-oper	IO-res
3	const/4 x,c	8	3	4	#ERR	1	#ERR	0	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
4	if-nez x,c	2	3	4	#ERR	4	4	#ERR	4	#ERR	9	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
5	return x	2	3	4	3	2	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
6	move x,y	4	3	4	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
7	rem-int x,y,z	2	3	12	#ERR	1	#ERR	#ERR	3	4	20	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
8	move x,y	2	2	4	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR
9	goto c	2	3	4	#ERR	3	2	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR

Fig. 2. The bytecode worksheet for the Euclid's algorithm presented in Fig. 1.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	...
...	...																												
n	instruction at time *n*	local registers afer instruction at time *n is executed*				control flow API at time *n+1*		constant at time n+1	arithmetic API at time *n+1*				line to be executed at time *n+1*	return line to be executed at time *n+1*	Symbol table API at time *n+1*										Input API at time *n+1*				
n+1	instruction at time *n+1*	local registers afer instruction at time *n+1 is executed*				control flow API at time *n+2*		constantl at time *n+2*	arithmetic API at time *n+2*				line to be executed at time *n+2*	return line to be executed at time *n+2*	Symbol table API at time *n+2*										Input API at time *n+2*				
...																													

Fig. 3. Schematic structure of a full description of SpVM configuration at time $n+1$. The configuration is yellow, the visible fragments of the preceding and the following one are pink and blue, respectively.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	opcode	registers				control flow		constant	arithmetics				Instr_count	Ret_line	Symbol table										Input definition			
2	instruction	v0	v1	v2	ret_val	type	jump_to	const-res	ar-arg1	ar-arg2	ar-op	ar-res	nextline	line	ST-id	ST-oper	ST-key	ST-item	ST-res	ST-impl-count	ST-key-wr	ST-item-wr	ST-key-rev	ST-marker	IO-arg1	IO-arg2	IO-oper	IO-res
3		#ERR	2	1	#ERR	1	#ERR	0	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	11	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
4	const/4 x,c	0	2	1	#ERR	4	4	#ERR	1	#ERR	9	1	2	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
5	if-nez x,c	0	2	1	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	4	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
6	move x,y	1	2	1	#ERR	1	#ERR	#ERR	2	1	20	0	5	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
7	rem-int x,y,z	1	2	0	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	6	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
8	move x,y	1	1	0	#ERR	3	2	#ERR	#ERR	#ERR	#ERR	#ERR	7	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
9	goto c	1	1	0	#ERR	4	4	#ERR	0	#ERR	9	0	2	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
10	if-nez x,c	1	1	0	#ERR	2	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	3	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR
11	return x	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	#ERR	1	#ERR	#ERR	#ERR	#ERR

Fig. 4. VM worksheet for the Euclid's algorithm presented in Fig. 1.

3.3 Addressing mechanisms

Dalvik is a register-based bytecode, so its fundamental logic is realized by fetching data elements from registers, performing operations on them, and storing results in registers. Let us explain the mechanism of addressing on the example of Euclid's algorithm, instruction **move** v0, v2 moving value from register v2 to v0. It is encoded in row 6 of Fig. 2 and executed in row 8 of Fig. 4. The main functional part of the formula in `B8` is

```
[B8]  =INDEX(A7:AB7,INDEX(bytecode,M7,COLUMN()))        (1)
```

The inner `INDEX` accesses the bytecode area (as a named range) in the row of the instruction to be executed now (in `M7`), column equal to the column of the register, finds an address there (it is 4 from `B6`) and fetches the value from column 4 of the register area in cells responsible for representation of the machine configuration in the previous time instant, located in `$A7:$AB7`. This general mechanism is used everywhere in SpVM to move values between registers, input and output registers of the dedicated organisational structures, etc.

This mechanism can indeed update concurrently as many individual registers as necessary. This potential is used when executing **invoke** bytecode instruction, to replace values of all registers by parameters of the method call in one step.

3.4 Modules and Organisational Structures

Our machine is composed of several modules, responsible for elements of the SpVM's behavior: control flow, function evaluation, data structure handling and I/O. So, in the process of translating a bytecode instruction into spreadsheet data, particular components of instruction's functionality are identified and encoded for use by SpVM. E.g., instruction **if-nez** v2, 0004 in line 0001 of Euclid's algorithm (Fig. 1) should execute a jump to line 4 if v2 is not zero. Its encoding is in row 4 of Fig. 2 and includes separate pieces of information about: register updates (all elements remain where they were), arithmetic operation to be performed (test for zero) and necessary data moves (from registers to structures that handle arithmetic), and constant of the row to jump to. This separation allows us to construct SpVM from spreadsheet formulas of reasonable size and complexity.

The same instruction is executed in row 5 of Fig. 4. The handling structures are located in columns `F` and `G`, the actual counter is in column `M`. The formula in `M5` is

```
[M5] =CHOOSE(F5,M4+1,INDEX(M$1:M4,N4)+1,G4,IF(L4,G4,1+M4),G4)  (2)
```

The arguments: code of the instruction to be executed (in `F5`) and constant representing the jump address (in `G5`) are imported directly from bytecode. The result of test if v2 is nonzero is computed by the arithmetic unit, whose handling structures are in columns `I`, `J` and `K`, and the result appears in column `L`. Hence

the value we need is found in L4. Now CHOOSE function computes the new line: the type of the operation in F5 is 1 for all instructions which unconditionally go to the next statement, 2 encodes return from a function call (to be explained below), 3 and 5 are for unconditional jumps—the former is a jump within a function (e.g. goto), the latter is a jump to a different function. Type 4 is the one we actually use: conditional jump depending on the test result.

All other handling structures are constructed similarly, with input cells for operands and operation code, and a CHOOSE-based formula to execute the needed operation.

3.5 Data structures

The idea behind data structures is that they are not stored in the configuration space of SpVM, which is of very limited size. Instead, in every step of computation which alters data structures, an appropriate change record is created in a few cells of the present row, only. The access to the data structure is executed by formulas which aggregate the information from whole columns of those records, from the top of the spreadsheet until the present row, i.e., the complete history of all changes until now, to determine the result of the access.

Symbol table. Its handling structure controls *put* requests which specify: an id n of a symbol table, a key k and an item i to be stored. The id and key are concatenated with a separator into a composite key $n : k$, which uniquely determines both components. The put is executed by the fact that the composite key and the value appear in the same row in two columns of symbol table.

Retrieving a value from the symbol table, given a composite key produced from two arguments of a *get* request, is accomplished by searching for the (position of the) most recent record of a *put* action using the same composite key, in a vertical range of cells. Standard MATCH searches top-down, but we must search bottom-up. There are several well-known solutions of this spreadsheet problem. When this position is found, an INDEX-based formula fetches the value associated with this put request.

Stack (LIFO). Its implementation is slightly more complex. We describe it here on a smaller model, with one column of data. In SpVM it is used to handle recursion, and hence it encompasses all its columns, which makes it quite difficult to explain, although the principle of its operation is exactly the same as below. Assume handling structure with operation id in column A: 1 indicates *push*, 2 indicates *pop*, 3 indicates no action. The data item to be pushed is in column B. The stack is located in the next three columns: column C holds the result of the pop (an error value if no pop is executed), column D holds the present top of the stack, and column E holds the number of the worksheet row immediately preceding the one in which the present top value was pushed. The formulas which do it are the following (examples from row 4):

```
[C4]   =CHOOSE(A4,NA(),D3,NA())
```

```
[D4]   =CHOOSE(A4,B4,INDEX(D$1:D3,E3),E3)
[E4]   =CHOOSE(A4,ROW()-1,INDEX(E$1:E3,E3),E3)
```

It is not difficult to verify that the formulas update this information in a way which conforms to the semantics of stack. The `INDEX` subformulas are the key ones, which restore the previous stack top when pop is executed, accessing the whole history of all stack operations until now. They are completely analogous to the `INDEX` subformula in (2), which handles return from a method call and restores the activation record immediately preceding the method call. Concerning the data movement formula (1), it is indeed a subformula of the complete one used in SpVM, which includes a test if we are returning from a method call now, in which case an `INDEX` is computed to restore the pre-call values of registers.

4 Downloads

We have prepared encodings of a few Dalvik programs: loop-based and recursion-based variants of Euclid's algorithm, Shell sort and insertion sort, which intensively use symbol tables. They are available for download, together with instructions, from `https://www.mimuw.edu.pl/~jty/SpVM/`.

References

1. R. J. Casimir. Real programmers don't use spreadsheets. *SIGPLAN Notices*, 27(6):10–16, 1992.
2. M. M. P. Center. Threat intelligence - July 2015. Technical report, Microsoft Corp., 2015.
3. R. De Levie. *Advanced Excel for scientific data analysis*. Oxford University Press, 2004.
4. T. A. Grossman and Ö. Özlük. Spreadsheets grow up: Three spreadsheet engineering methodologies for large financial planning models. *CoRR*, abs/1008.4174, 2010.
5. F. Hermans. Excel Turing machine, Sept. 2013.
6. I. Premachandra. Modeling a Turing machine on a spreadsheet: a learning tool. *International Journal of Information and Management Sciences*, 4(2):81–92, 1993.
7. E. Rautama, E. Sutinen, and J. Tarhio. Excel as an algorithm animation environment. In *ITiCSE '97: Proceedings of the 2nd conference on Integrating technology into computer science education*, pages 24–26, New York, NY, USA, 1997. ACM.
8. J. Sikora, J. Sroka, and J. Tyszkiewicz. Programming communication with the user in multiplatform spreadsheet applications. In P. Milazzo, D. Varró, and M. Wimmer, editors, *Software Technologies: Applications and Foundations - STAF 2016 Collocated Workshops: DataMod, GCM, HOFM, MELO, SEMS, VeryComp, Vienna, Austria, July 4-8, 2016, Revised Selected Papers*, volume 9946 of *Lecture Notes in Computer Science*, pages 356–371. Springer, 2016.
9. J. Sroka, A. Panasiuk, K. Stencel, and J. Tyszkiewicz. Translating relational queries into spreadsheets. *IEEE Trans. Knowl. Data Eng.*, 27(8):2291–2303, 2015.
10. J. Swan. *Practical financial modelling: a guide to current practice*. Butterworth-Heinemann, 2009.
11. J. Tyszkiewicz. Spreadsheet as a relational database engine. In A. K. Elmagarmid and D. Agrawal, editors, *SIGMOD Conference*, pages 195–206. ACM, 2010.

Implementing GROUP BY Calculations as Spreadsheet Formulas

Paul Mireault

SSMI International
Montréal, Canada
Paul.Mireault@SSMI.International

Abstract. A GROUP BY is an operator of the SQL language that allows aggregate calculations to be performed on a set of rows in a database. In a spreadsheet program like Microsoft Excel, one could program aggregate calculations in VBA, its programming language, or use its Query or its Pivot Table tools. Excel functions like SUMIF and AVERAGEIF perform their calculations on one level of aggregation, and functions like SUMIFS and AVERAGEIFS can work on more levels but become unwieldly quickly. In this paper, we present similarities and differences between SQL's GROUP BY and spreadsheet aggregating formulas. We also present preparation formulas that allow the developer to always use the simple SUMIF and AVERAGEIF, no matter how many levels of aggregation are needed. Finally, we also provide model management formulas to help the spreadsheet developer ensure that his spreadsheet model covers all the possible groups.

1 Introduction

Various research has reported important spreadsheet error rates. [1] surveyed studies showing a percentage of spreadsheet with errors as high as 86%. Spreadsheet errors have led to not only monetary losses [2] but have also caused career failures [3] and reputation losses [4].

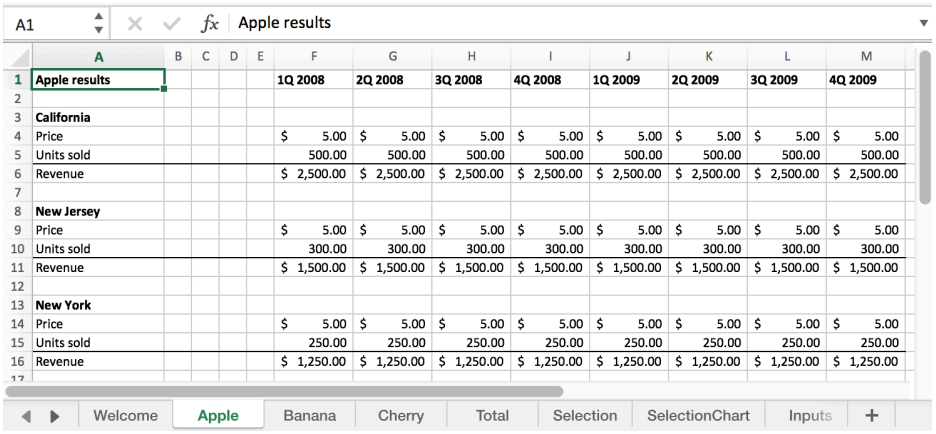
Research has been done to help spreadsheet developers build complex spreadsheets by using Computer Science and Software Engineering concepts. For example, [5] examines cell labels typed by the spreadsheet developer to infer a structure and provide type checking in formulas, and [6] proposes a model-driven approach. [7] proposes a method to reverse-engineer a spreadsheet to extract relational model and propose a method to do the opposite: generate a spreadsheet from a relational model. Finally, [8] developed a methodology based on the conceptual model of Information Systems.

While Microsoft Excel has tools to analyse multidimensional data using Pivot Tables and Queries, there are no equivalent tools for multidimensional models.

When analysing multidimensional data, the Pivot Table itself is the result sought by the user. But in the case of a model, the user can perform different scenario analysis to explore the impact of some input variable values on some key result variables. If we were to use Pivot Tables to aggregate some variables, we would need one Pivot Table for each initial and final dimension sets. The result of one Pivot Table would be variables that are used in the calculation of other variables, which are in turn used in another Pivot Table.

But Pivot Tables and Queries do not update dynamically when the underlying values change. After modifying an input value, the spreadsheet user would then have to update each Pivot Table, in the proper order, to observe the correct result. While it is possible to program routines in VBA that would be triggered automatically when an input value changes, this requires training that most Excel users don't have.

Spreadsheet developers have resorted to creating all sorts of structures to represent dimensions. For example, [9] has a spreadsheet with three dimensions: *Quarter*, *Product* and *Region* (see Figure 1). The *Product* dimension is presented as different worksheets, the *Quarter* dimension as columns and the *Region* dimension as blocs of repeated formulas.



	A	F	G	H	I	J	K	L	M
1	**Apple results**	**1Q 2008**	**2Q 2008**	**3Q 2008**	**4Q 2008**	**1Q 2009**	**2Q 2009**	**3Q 2009**	**4Q 2009**
2									
3	**California**								
4	Price	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00
5	Units sold	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
6	Revenue	$ 2,500.00	$ 2,500.00	$ 2,500.00	$ 2,500.00	$ 2,500.00	$ 2,500.00	$ 2,500.00	$ 2,500.00
7									
8	**New Jersey**								
9	Price	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00
10	Units sold	300.00	300.00	300.00	300.00	300.00	300.00	300.00	300.00
11	Revenue	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00	$ 1,500.00
12									
13	**New York**								
14	Price	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00	$ 5.00
15	Units sold	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00
16	Revenue	$ 1,250.00	$ 1,250.00	$ 1,250.00	$ 1,250.00	$ 1,250.00	$ 1,250.00	$ 1,250.00	$ 1,250.00

Figure 1 Multidimensional spreadsheet example

In this paper, we will present worksheet structures and Excel formulas that will perform aggregate calculations similar to the GROUP BY clause of SQL's SELECT statement.

2 Multidimensional modelling and database concepts

A model with multidimensional variables is similar to database tables representing multidimensional entities. We will build model management variables to implement the equivalent of primary keys and foreign keys, allowing us to *join* worksheets as we would join tables.

2.1 JOIN operation

In SQL, the JOIN operation lets us join the rows of two tables based on the condition that the values of the columns forming the primary key of one table are equal to the values of the corresponding columns of the other table, called a foreign key.

A typical JOIN operation looks like this:

```
T1 JOIN T2 USING (C1, C2,…Cj)
```

The set of columns (C1, C2,… Cj) is the primary key of table T1 and a foreign key of table T2, or vice-versa. Each row of table T2 will be joined to the row of table T1 that satisfies the condition

```
(T1.C1 = T2.C1 AND T1.C2=T2.C2 AND … AND T1.Cj=T2.Cj)
```

For example, the clause `ORDER JOIN ORDER_DETAIL USING (ORDER_ID)` will join to each row of table ORDER_DETAIL the corresponding row of the ORDER table. The corresponding row is determined by the condition

```
(ORDER.ORDER_ID = ORDER_DETAIL.ORDER_ID)
```

In our implementation of primary and foreign keys, we will keep the simple name of the variable in the worksheet where it represents the primary key, ***ORDER ID*** in the *ORDER* worksheet, but we will combine it with the worksheet name where it represents a foreign key, ***ORDER ID in ORDER DETAIL*** in the *ORDER DETAIL* worksheet. This naming convention lets us avoid having homonyms in our Excel implementation, which is considered to be a source of error by [10].

2.2 Composite keys

A composite primary key is a primary key composed of more than one column. While composite keys are useful in databases, we prefer not to use them

in our approach. Every composite key will be replaced by an equivalent surrogate key constructed by concatenating all the elements of the composite key. This is illustrated in section 4.1.

Using concatenated surrogate keys lets us use Excel's simpler `SUMIF` function instead of the `SUMIFS` function.

3 Example problem

In our example, we need to create a worksheet for each set of dimensions that interest us. In our example, we use dimensions *Year*, *Region*, *Product* and *Market*, which we will represent by the letters Y, R, P and M. A modelling expert has examined the problem and prepared the conceptual model of its solution.

Figure 2, using the methodology presented in [11], shows the Formula Diagram of our example model. It calculates the number of units sold and the sales amount for each year, for each region, for each product and for each market. These are variables UNITS SOLD(y, r, p, m) and SALES AMOUNT(y, r, p, m).

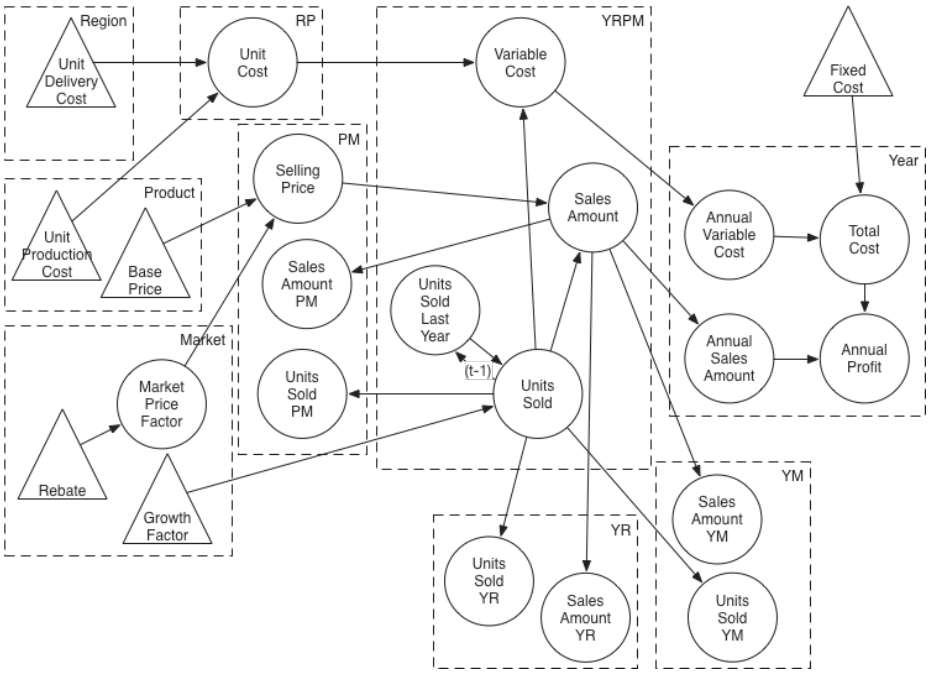


Figure 2 Formula Diagram

The triangles represent data variables and the circles are calculated variables. The dash-bordered boxes represent dimension sets that may require one worksheet for its calculated variables and one for its data variables.

A calculated variable, like Unit Cost and Selling Price, belongs to a dimension set formed by the union of the dimension sets of the variables used in its calculation. Aggregated calculated variables, like Units Sold YR and Annual Variable Cost, must be calculated from a variable belonging to a dimension set that is a superset of their own. As a counter example, variable Units Sold YR cannot be calculated from Units Sold YM.

The problem we now face is the implementation of the solution presented in the Formula Diagram. We are interested in calculating the following aggregate variables:

Units sold and sales amount by year and region	$\text{UNITS SOLD YR}(y,r)$ $\text{SALES AMOUNT YR}(y,r)$

Note that we name the aggregate variables with the name of the calculated variable followed by their dimension code. Mathematically, we represent the calculations as

$$\text{UNITS SOLD YR}_{y,r} = \sum_{m,p} \text{UNITS SOLD}_{y,r,p,m}$$

$$\text{SALES AMOUNT YR}_{y,r} = \sum_{m,p} \text{SALES AMOUNT}_{y,r,p,m}$$

We can write these calculations in SQL as follows:

```
SELECT YEAR, REGION,
       SUM(UNITS SOLD) AS "UNITS SOLD YR",
       SUM(SALES AMOUNT) AS "SALES AMOUNT YR"
FROM YRPM
GROUP BY YEAR, REGION
```

In the following section, we will explain how to set up model management variables in the *YRPM* worksheet, representing the origin dimension set, and the *YR* worksheet, representing the destination dimension set.

4 Spreadsheet preparation

In Excel, we can do the same calculation with the `SUMIF` function, but it requires some preparation. We mentioned earlier that we need a worksheet for each set of dimensions that interest us. In this case, we will have a worksheet named *YRPM* with all the variables with the (Y, R, P, M) dimension set, and a worksheet named YR with all the variables with the (Y, R) dimension set.

If we build the multidimensional worksheets as cartesian products of their basic dimensions, we will be assured that the primary keys cover all possibilities.

4.1 The *YRPM* worksheet

In the *YRPM Data* worksheet, we define a model management variable that describes the surrogate foreign key composed with dimensions Y and R with the following formula:

```
YR in YRPM = Year in YRPM & "-" & Region in YRPM
```

This is illustrated in Figure 3

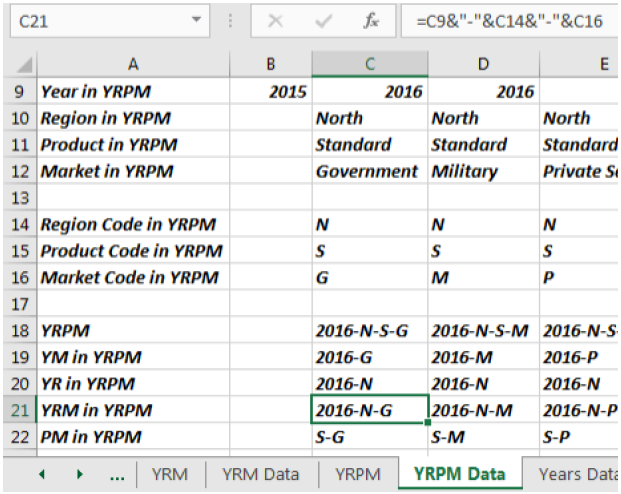
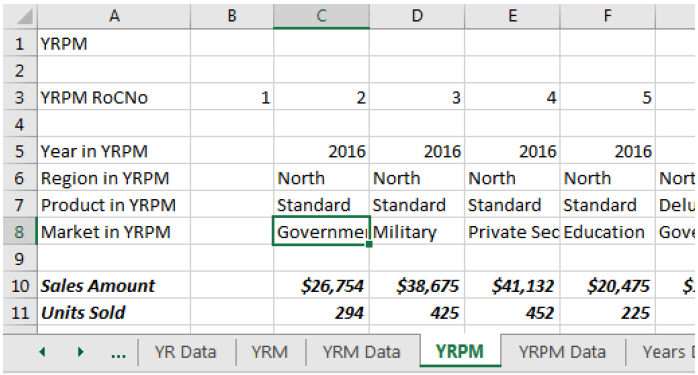


Figure 3 Surrogate foreign key `YR in YRPM` definition

We use a dash to separate the elements of the composite key to avoid situations where concatenating different values give the same result. For example, if we don't use a separator regions S and SE concatenated with markets ED and

D would both give the same result, SED. The separator, a dash in this case, needs to be a character that never appears in the values we are concatenating.

The *YRPM* worksheet performs the actual calculation of the model's variables, `Sales Amount` and `Units Sold` as illustrated in Figure 4. Column C contains all the model's formulas, and they are copied up to column PZ, representing 440 instances of dimension set YRPM (11 years, 5 regions, 2 products and 4 markets).



	A	B	C	D	E	F	
1	YRPM						
2							
3	YRPM RoCNo	1	2	3	4	5	
4							
5	Year in YRPM		2016	2016	2016	2016	
6	Region in YRPM		North	North	North	North	Nort
7	Product in YRPM		Standard	Standard	Standard	Standard	Delu
8	Market in YRPM		Governme	Military	Private Sec	Education	Gove
9							
10	***Sales Amount***		***$26,754***	***$38,675***	***$41,132***	***$20,475***	***$***
11	***Units Sold***		***294***	***425***	***452***	***225***	

YR Data | YRM | YRM Data | **YRPM** | YRPM Data | Years

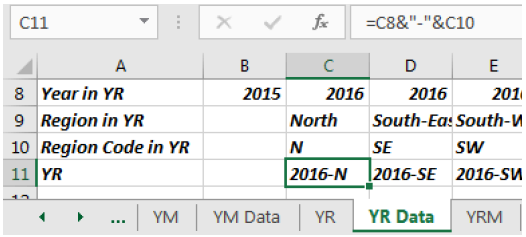
Figure 4 Model variables `Sales Amount` and `Units Sold` in YRPM

4.2 The *YR* worksheet

Similarly, we define the primary key in the *YR Data* worksheet with a similar formula: (see Figure 5)

```
YR = Year in YR & "-" & Region Code in YR
```

In the *YR* worksheet, variable ***YR*** is the equivalent of a primary key and has unique values, and in the *YRPM* worksheet ***YR in YRPM*** is the equivalent of a foreign key and has repeated values.



C11 | =C8&"-"&C10

	A	B	C	D	E
8	***Year in YR***	***2015***	***2016***	***2016***	***201***
9	***Region in YR***		***North***	***South-Eas***	***South-W***
10	***Region Code in YR***		***N***	***SE***	***SW***
11	***YR***		***2016-N***	***2016-SE***	***2016-SW***

YM | YM Data | YR | **YR Data** | YRM

Figure 5 Surrogate primary key `YR` definition

If we consider tables `YR` and `YRPM` to represent the similarly named worksheets, their relationship is illustrated in the Data Structure Diagram of Figure 6.

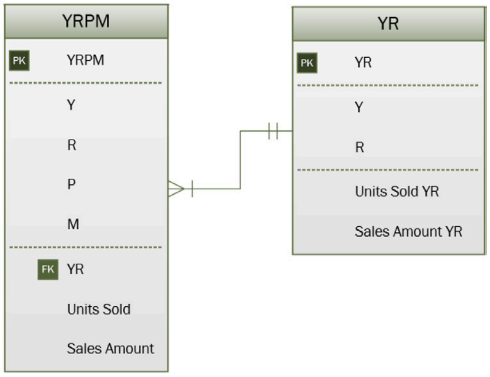


Figure 6 Relationship between tables YR and YRPM, using composite keys

Finally, we are now set to calculate the aggregate variables we are interested in with the following formulas:

```
UNITS SOLD YR = SUMIF(YR in YRPM, YR, UNITS SOLD)
```

The formulas are illustrated in Figure 7.

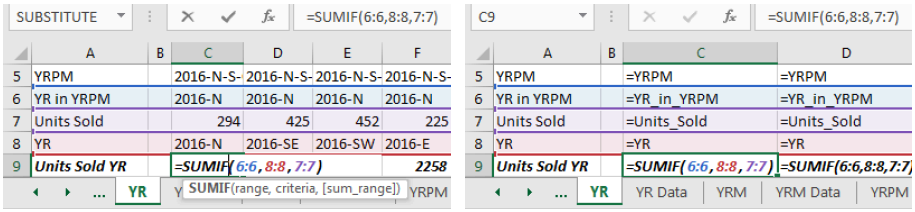


Figure 7 Aggregate calculation, normal view (left) and formula view (right)

The actual behavior of the `SUMIF` function in row 9 is more like this SQL statement, even though the `JOIN` operation is not really needed:

```
SELECT YEAR, REGION, SUM(UNITS SOLD) AS "UNITS SOLD YR"
FROM YRPM
     JOIN YR ON (YR.YR = YRPM.YR)
GROUP BY YEAR, REGION
```

In Figure 7, ***YR*** in row 8 is `YR.YR`, the primary key of the `YR` table, and **YR *in YRPM*** in row 6 is `YRPM.YR`, a foreign key of the `YRPM` table referencing `YR.YR`.

A possible source of error would be having an incorrect number of columns in rows 6 to 9: rows 6 and 7 need as many columns as there are in the YRPM dimension, and rows 8 and 9 need as many columns as there are in the YR dimension. To warn against such an error, we could add model management formulas to verify that the `SUM` of rows 7 and 9 are equal to the `SUM` of row 11 of worksheet *YRPM* (see Figure 4).

If we don't use composite keys, then the relationship between tables `YR` and `YRPM` is shown in Figure 8. Figure 9 shows how the same calculation can be done with the `SUMIFS` function. In general, the use of `SUMIFS` requires adding one pair of simple foreign key values to the top part of the block for each dimension used in the result dimension set, (`Year`, `Region`) in this example.

```
UNITS SOLD YR = SUMIFS(UNITS SOLD, Y in YRPM, Y in YR, R in
YRPM, R in YR)
```

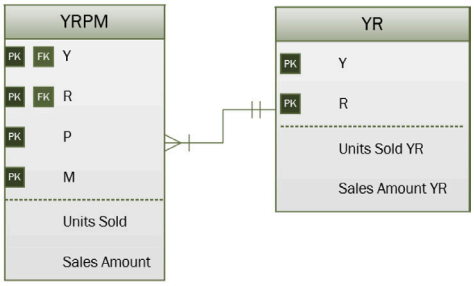


Figure 8 Relationship between tables YR and YRPM, without composite keys

19	YRPM	2016-N-S-	2016-N-M	2016-N-P	2016-N-E	2016-SE-G	2016-SE-N	2016-SE-P	2016-SE-E	2016-SW
20	Year in YRPM	2016	2016	2016	2016	2016	2016	2016	2016	201
21	Region in YRPM	North	North	North	North	South-Eas	South-Eas	South-Eas	South-Eas	South-W
22	Units Sold	294	425	452	225	204	420	165	119	26
23	Year in YR	2016	2016	2016	2016	2016	2017	2017	2017	201
24	Region in YR	North	South-Eas	South-We	East	West	North	South-Eas	South-We	East
25	***Units Sold YR***	***=SUMIFS(22:22, 20:20, 23:23, 21:21, 24:24)***					***1376***	***1418***	***840***	***151***
26										

SUMIFS(sum_range, criteria_range1, criteria1, [criteria_range2, criteria2], [criteria_range3, criteria3], ...)

... YR | YR Data | YRM | YRM Data | YRPM | YRPM Data | Years Data | Products Data

Figure 9 Calculation done with the SUMIFS function

The behavior of the `SUMIFS` function in row 25 is like this SQL statement, even though the `JOIN` operation is not really needed:

```
SELECT YEAR, REGION, SUM(UNITS SOLD) AS "UNITS SOLD YR"
FROM YRPM
     JOIN YR ON (YR.YEAR = YRPM.YEAR AND YR.REGION = YRPM.REGION)
GROUP BY YEAR, REGION
```

The calculation of Annual Variable Cost from Variable Cost (see Figure 2), aggregating over 3 dimensions, would require setting up 7 rows to perform a `SUMIFS`, while it would still require only 3 rows for a `SUMIF`.

5 Conclusion

Multidimensional variables are hard to model in spreadsheets. Spreadsheet developers, left to themselves, devise imaginative ways of handling them. But they almost always involve making multiple copies of formulas, thereby increasing the number of operations needed to maintain the spreadsheet.

In this paper, we presented a set of formulas comprising model management formulas to prepare the necessary structures and the simple model formulas to calculate aggregate values. These model management formulas serve to emulate the concepts of primary and foreign keys used in relational databases as well as the GROUP BY operation. They could be generated automatically, either with an application or with an Excel template. One could use the method presented in [12] to generate the rows containing the primary keys and the foreign keys for all the needed dimension sets.

Here are a few areas that deserve further research:

- Is constructing a spreadsheet using the multidimensional structures presented here better than letting developers use their own approach? We could define *better* with the number of errors, the elapsed time or some other productivity measure.
- Is a spreadsheet constructed using the multidimensional structures presented here easier to maintain than one constructed using another approach? There are two different maintenance operations we must consider: adding values to a dimension or modifying the problem requirements and adding new variables. The first case could be as simple as adding a new product. The second case could be adding a monthly handling capacity to the regional warehouse and an extra cost when the number of units shipped during the month exceeds that capacity. We could also consider whether the modification is performed by the original spreadsheet developer or by somebody else.

6 References

[1] R. R. Panko, "What We Know About Spreadsheet Errors," *Journal of End User Computing,* vol. 10, no. 2, 2008.

[2] T. Burden, "How A Rookie Excel Error Led JPMorgan To Misreport Its VaR For Years," [Online]. Available: http://www.zerohedge.com/news/2013-02-12/how-rookie-excel-error-led-jpmorgan-misreport-its-var-years. [Accessed 06 04 2017].

[3] Daily Express, "Mouchel profits blow," 7 10 2011. [Online]. Available: http://www.express.co.uk/finance/city/276053/Mouchel-profits-blow. [Accessed 06 04 2017].

[4] M. Conczal, "Researchers Finally Replicated Reinhart-Rogoff, and There Are Serious Problems," Roosevelt Institute, 2013. [Online]. Available: http://rooseveltinstitute.org/researchers-finally-replicated-reinhart-rogoff-and-there-are-serious-problems/. [Accessed 06 04 2017].

[5] M. Erwig, "Software Engineering for Spreadsheets," *IEEE Software,* vol. 26, no. 5, pp. 25-30, 2009.

[6] J. Cunha, J. P. Fernandes, J. Mendes and J. Saraiva, "MDSheet: a framework for model-driven spreadsheet engineering," in *Proceedings of the 34th International Conference on Software Engineering*, Zurich, 2012.

[7] J. Cunha, M. Erwig, J. Mendes and J. Saraiva, "Model inference for spreadsheets," *Autom Softw Eng,* vol. 23, pp. 361-.392, 2016.

[8] P. Mireault, "Structured Spreadsheet Modeling and Implementation," in *SEMS 15*, Florence, 2015.

[9] M. Brandewinder, "Excel, named ranges and INDIRECT()," 22 04 2008. [Online]. Available: http://www.clear-lines.com/blog/post/Excel2c-named-ranges-and-INDIRECT().aspx. [Accessed 27 04 2017].

[10] E. Ferguson, "Minimising Spreadsheet Errors," ICAEW, London, 2011.

[11] P. Mireault, Structured Spreadsheet Modelling and Implementation: A Methodology for Creating Effective Spreadsheets, Second ed., Montréal: SSMI International, 2017.

[12] P. Mireault, "Implementing Nested FOR Loops as Spreadsheet Formulas," in *SEMS16*, Vienna, 2016.

Tabula: A Language to Model Spreadsheet Tables

Jorge Mendes and João Saraiva

HASLab, INESC TEC and Universidade do Minho, Portugal
{jorgemendes,saraiva}@di.uminho.pt

Abstract. Spreadsheets provide a flexible and easy to use software development environment, but that leads to error proneness. Work has been done to prevent errors in spreadsheets, including using models to specify distinct parts of a spreadsheet as it is done with model-driven software development. Previous model languages for spreadsheets offer a limited expressiveness, and cannot model several features present in most real world spreadsheets.
In this paper, the modeling language Tabula is introduced. It extends previous spreadsheet models with features like type constraints and nested classes with repetitions. Tabula is not only more expressive than other models but it can also be extended with more features. Moreover, Tabula includes a bidirectional transformation engine that guarantees synchronization after an update either in the model or spreadsheet.

Keywords: tabula, spreadsheet, model-driven engineering

1 Introduction

The size and complexity of software has been quickly increasing in the last years. A paradigmatic example is the size of the software included in the aerospace industries: while the space shuttle developed in the 80's contained about $400,000$ lines of source code[1], the modern *Airbus A380* built in this century includes more than 100 millions lines of code [13]. In such large and complex systems it is unfeasible to reason about the software just by looking/understanding its source code [13].

Model-Driven Software Development (MDSD) has emerged as an important software engineering discipline allowing developers to reason about complex software by providing simple/concise abstractions - the software model. Very much like a civil engineer develops "human-scale" models of a bridge, before the real bridge is constructed, in MDSD a software engineer reason about his complex software by analyzing a simpler model.

Spreadsheets are no exception, and, indeed, they tend to evolve into large and complex software systems, which are difficult to understand, to maintain, and to evolve. The combination of complexity with the lack of abstraction mechanisms, is the main cause of the (too) many errors caused by spreadsheets [1, 11, 12].

[1] https://www.nasa.gov/mission_pages/shuttle/flyout/flyfeature_shuttlecomputers.html

Model-driven spreadsheet development was introduced in spreadsheets [3, 7–9] with two main goals: Firstly, to provide a powerful abstraction of the business logic of spreadsheets so that users can reason about their spreadsheets by analyzing simple models, instead of very large and complex data. Secondly, to provide a *type system* for spreadsheets: the model is incorporated into a regular spreadsheet so that it limits the data/formulas that can be defined in the spreadsheet cells. In such a model-driven spreadsheet engineering setting, the spreadsheet data (i.e., the instance) has always to conform to the spreadsheet model (i.e., the type), thus steering users in introducing correct data.

ClassSheets were introduced by Engels and Erwig as a powerful domain-specific modeling language for spreadsheets [7]. ClassSheets define both the computation and layout of spreadsheet tables. In previous work we have extended the ClassSheet formalism to improve its expressiveness [5]. Moreover, we conducted empirical studies using real-world model-driven spreadsheets that showed an improvement in users' performance, while reducing the error rate [2]. This latter work also showed the limitations of (extended) ClassSheets: the business logic and layout of several real-world spreadsheets could not be modeled by a ClassSheet. In other cases, the ClassSheet model was not the natural way to express the layout/logic of the spreadsheet.

In this paper we present a new modeling language for spreadsheets tables: Tabula. This language is inspired by the ClassSheet modeling language, namely its visual notation, but it provides more expressive features like type constraints and nested classes with repetitions enabled by a different abstract representation. Moreover, Tabula includes a bidirectional transformation engine that guarantees synchronization after an update either in a Tabula model or spreadsheet. Much like our previous work on ClassSheets, the usage of model-driven spreadsheets targets repeated use of spreadsheets with a well-defined structure and logic. Tabulae are to be defined by a specialist in the domain with knowledge on modeling with Tabula, but usage of the respective spreadsheets targets usual spreadsheet users. We used the Tabula visual language to model a widely used budget spreadsheet that is provided by Microsoft as a budget template. Moreover, we also model the business logic of that spreadsheet using a ClassSheet model, and we compare the expressiveness of both spreadsheet modeling languages.

This paper is organized as follows: Section 2 gives a short introduction to model-driven spreadsheets and discusses the ClassSheet modeling language. Section 3 introduces in detail the Tabula modeling language and briefly describes its bidirectional transformation engine. In Section 4, we evaluate the expressiveness of Tabula when modeling a budget spreadsheet. We also compare the Tabula and the ClassSheet models for this spreadsheet instance. Finally, Section 5 includes our conclusions.

2 Model-Driven Spreadsheet Development

Before we propose a new modeling language for spreadsheets let us discuss in more detail the state of the art on model-driven spreadsheet engineering.

Several different approaches to use MDE in spreadsheets have been proposed in literature. The first model-driven spreadsheet specification language was *Model Master* [9]: an object-oriented textual specification of a spreadsheet that can be compiled into a concrete spreadsheet and also to be *decompiled* from a spreadsheet. It has a mathematical background, conceived from category theory concepts.

Spreadsheet templates [8] were the first approach to define a MDE spreadsheet language with a visual representation which allows to specify spreadsheets in a spreadsheet-like manner. On top of these spreadsheet templates, Engels and Erwig proposed ClassSheets [7]: a high-level, object-oriented formalism to specify the layout and business logic of spreadsheets. The visual representation of ClassSheets is close to what spreadsheets users are familiar with. ClassSheets are supported by their own application and then compiled to spreadsheets. As a result, model and instance are supported by different software systems, limiting the synchronization whenever one artifact (model or instance) evolves. In order to minimize this drawback, ClassSheets have been embedded in spreadsheet system [2, 3], bringing spreadsheet modeling closer to end users, and allowing model/instance co-evolution. Moreover, these embedded ClassSheets have been extended with additional features to improve their expressiveness [5].

To show the expressiveness of ClassSheets, and also their limitations, let us consider a simple spreadsheet to keep track of the inventory of products, as shown in Fig. 1. In its simplest form, the inventory is just a list of items. Each item defines the name of the product (column A) and the available quantity (column B). The last row contains the total of products: the sum of the quantities of listed products. Figure 2 contains the ClassSheet (in its visual notation [7]) that models the business logic and layout of this spreadsheet: it consists of a class, named **Items**, that includes two attributes (`desc` of type *string*, and `stock` of type *number*, with default values the empty string and 0, respectively). The total of products is defined by a formula that refers to attribute names (and not the usual column/row references). Finally, the layout is specified by the vertical dots, meaning that values of the above attributes can repeat vertically.

Spreadsheet

	A	B
1	**Items**	
2	apple	5
3	banana	2
4	cherry	8
5	**Total**	=SUM(B2:B4)

Fig. 1. Spreadsheet storing a list of items with their respective stock and its total.

Items ClassSheet

	A	B
1	**Items**	
2	desc=""	stock=0
⋮		
3	Total	total=SUM(stock)

Fig. 2. ClassSheet specifying the spreadsheet in Fig. 1, using the notation from [7].

	A	B
1	**Inventory**	
2		
3	Fruit	
4	apple	5
5	banana	2
6	cherry	8
7		
8	Legumes	
9	beans	7
10	peas	10
11		
12	**Total**	=SUM(B4:B6,B9:B10)

Fig. 3. Spreadsheet to keep an inventory of items grouped in categories.

	A	B
1	**Inventory**	
2		
3	**Fruit**	
4	desc=""	stock=0
⋮		
5		
6	**Legumes**	
7	desc=""	stock=0
⋮		
8		
9	Total	total=SUM(**Fruit**.stock, **Legumes**.stock)

Fig. 4. ClassSheet specifying the spreadsheet in Fig 3.

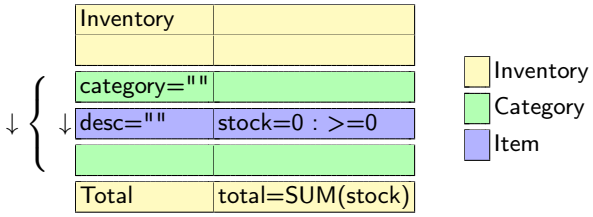


Fig. 5. Tabula specifying the inventory spreadsheet in Fig. 3.

In the context of MDE we say that the spreadsheet instance presented in Fig. 1 conforms to the spreadsheet model in Fig. 2. We can also say that the spreadsheet data in Fig. 1 has type *Items* ClassSheet.

Let us consider now that we wish to distinguish different categories of products, namely, *fruits* and *legumes*. Figure 3 presents a possible spreadsheet that structures the inventory in this way. However, this new inventory spreadsheet is not easily specified using ClassSheets, because they do not support nested repetition of classes. Figure 4 contains a ClassSheet that does model the new inventory, but, as we can see, some of the data (the categories) must be defined at the model level. As a result, models tend to grow and resemble the instance, which limits the expressiveness of the approach: when a new product category has to be considered both the model and instance have to be updated!

The key feature lacking in the ClassSheet modeling language is the possibility to define nested repeated classes/attributes. Thus, in ClassSheets, it is not possible to express the fact that we have categories of products that repeat vertically, each of which has a list of items (pairs of product name and quantity) that also repeat vertically.

The inventory spreadsheet can be modeled using Tabula as shown in Fig. 5. In this case, categories can also be abstracted in the model and all the data is defined

in the spreadsheet, only. A key advantage of Tabula over ClassSheets is the ability to describe nested repetitions. In our running example, it defines one for the category (as indicated by the downwards arrow on the left of the left brace) and a nested one for the category items is indicated by the other downwards arrow (on the right of the left brace). This feature is not only possible for vertical repetitions like in this example, but also for horizontal repetitions. Moreover, Tabula allows to define type constraints leading to a better characterization of the data the spreadsheet should hold, and, as a consequence it prevents user errors. In our example, the attribute `stock` has its values restricted to non-negative numbers. A detailed description of the Tabula modeling language is presented in the next section.

3 The Tabula Spreadsheet Modeling Language

A spreadsheet comprehends both the logic and the layout of a program. The Tabula modeling language specifies both of those elements, having a definition of the computation a spreadsheet has to perform, and how such computation is shown to users. Tabula include also a bidirectional evolution engine that guarantees model/instance synchronization after a transformation in one of these software artifacts. In the next section we define the Tabula modeling language. After that, we briefly describe the Tabula evolution engine.

3.1 Specification of Layout and Logic

A *Tabula* is defined as a type with a name, a list of classes and a grid. We express it in the Haskell programming language by the following abstract data type:

```
data Tabula = Tabula Name [Class] (Grid TCell)
```

Classes provide a meta-information about the cells in their range, while the grid defines the layout and contents of the spreadsheet. For that, classes are defined as

```
data Class = Class Name Range Expansion
```

The range of type `Range` is defined as the pair of its top-left coordinate and its bottom-right one.

```
type Range = (Point, Point)
```

The expansion information, of type `Expansion`, defines if the class can expand in any direction and thus have multiple objects in the instance.

```
data Expansion = None | Down | Right | Both
```

There are several options:

- the class can only have a single object, called a singleton, and thus does not expand, defined by the value `None`;

- the class can have multiple objects that repeat downwards, defined by the value `Down`;
- the class can have multiple objects that repeat to the right, defined by the value `Right`; and,
- the class can have multiple objects that repeat both to the right and downwards, defined by the value `Both`.

The layout is presented by a grid, where each cell in the Tabula grid represents one or more cells in the spreadsheet. There are three kinds of Tabula cells, represented by the type `TCell`: *input*, *formula* and *label*.

An *input* cell is specified with the contents in the format "name=default", where *name* is the name of the attribute it represents and *default* is the default value for the respective cells in the spreadsheet. This value is also used to define the type of the contents, where a numeric value implies the type being a number and a quoted text (possibly empty) implies the type being textual.

A *formula* cell is similar to an *input* cell. It is defined using the format "name=formula", where *name* is the name of the attribute it represents and *formula* is the formula it defines. References in the formula are made using attribute names defined by *input* cells or by other *formula* cells.

A *label* cell is any cell that is neither of kind *input* nor *formula*. The content of the respective cells in the spreadsheet are exactly the same content as in the Tabula cell. In this kind of cells fall empty cells, numeric cells and any other textual cell.

The logic is as specified by the classes and the attributes defined in the grid. Each class in the Tabula represents a kind of object present in the spreadsheet, where the attributes are the ones in the grid contained in the class' range.

The spreadsheet of our running example can be defined with a Tabula containing a two-by-six grid with two classes, where one class contains the list of items and another class contains each item:

```
inventory =
  Tabula  "Inventory"
          -- classes
          [  Class "Inventory"  ((0,0), (1,5))  None
          ,  Class "Category"   ((0,2), (1,4))  Down
          ,  Class "Item"       ((0,3), (1,3))  Down]
          -- grid
          (Grid.fromLists  [  ["Inventory", ""]
                           ,  ["",          ""]
                           ,  ["desc=\"\"", "stock=0 : >=0"]
                           ,  ["",          ""]
                           ,  ["Total",     "total=SUM(stock)"]])
```

A concise and graphical representation of this Tabula is presented in Fig. 5. It describes the grid in its tabular format and uses the background color to identify the classes each cell belongs to[2]. On the side there is a legend relating

[2] We assume that colors are available in the electronic version of this document,

the color to the class since the name is not always present in the table: the class **Inventory** (in yellow) has a label in its top-left cell but the classes **Category** and **Item** (in green and blue, respectively) do not. Moreover, in some cases, the label can be different from the class name. The arrow pointing down on the left of the attribute `desc` indicates that the class **Item** expands down and the one on the left of the left brace indicates that the class **Category** also expands down, with the left brace removing any visual ambiguity that might arise about what is repeated.

The layout of a Tabula can also expand horizontally and the expansion area can contain multiple columns or rows. This configuration is displayed in Fig. 7, where the inventory for one item (*apple*) along the years is presented. Thus, a new class **Year** is created, containing the attributes `year`, `stock` and `sold` to represent the year an object refers to, the stock of the item at the end of the year and the number of items sold that year, respectively.

	A	B	C	D	E	F
1	**Items**	2012		2013		
2		**stock**	**sold**	**stock**	**sold**	**Average sold**
3	apple	5	12	4	16	=AVERAGE(C3,E3)

Fig. 6. Spreadsheet with an apple inventory per year.

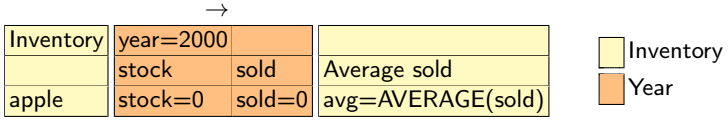


Fig. 7. Tabula specification of an apple inventory per year.

In Fig. 9, the conjugation of the configurations presented in Figs. 5 and 7 is shown, providing a more complex example. The result contains additional classes that relate the expandable classes from both of the previous examples. Thus, the Tabula contains the class **Inventory** from both examples, the class **Item** from Fig. 5 and the class **Year** from Fig. 9. Moreover, it contains two new classes to represent the attributes that are in the relation between:

- **Category** and **Year**, named **CategoryYear**; and,
- **Item** and **CategoryYear**, named **ItemCategoryYear**.

Both of these classes expand in both directions: horizontally and vertically. This Tabula is specified as follows:

```
inventoryYear =
  Tabula
    "InventoryYear"
    -- classes
    [  Class "Inventory"    ((0,0), (3,5))  None
    ,  Class "Year"         ((1,0), (2,5))  Right
```

```
,  Class "Category"     ((0,2), (3,4))  Down
,  Class "CategoryYear" ((1,2), (2,4))  Both
,  Class "Item"         ((0,3), (3,3))  Down
,  Class "ItemYear"     ((1,3), (2,3))  Both]
-- grid
(Grid.fromLists
  [["Inventory",  "year=2000",    "",           ""]
  ,["",           "",             "",           ""]
  ,["cat=\"\"",   "stock",        "sold",       "Average sold"]
  ,["desc=\"\"",  "stock=0:>=0", "sold=0:>=0", "avg=AVERAGE(sold)"]
  ,["",           "",             "",           ""]
  ,["Total",      "total=SUM(stock)", "",       ""]])
```

or graphically as in Fig. 9.

	A	B	C	D	E	F
1	**Inventory**	2012		2013		
2						
3	Fruit	**stock**	**sold**	**stock**	**sold**	**Average sold**
4	apple	5	12	4	16	=AVERAGE(C4,E4)
5	banana	2	10	3	12	=AVERAGE(C5,E5)
6	cherry	8	9	1	3	=AVERAGE(C6,E6)
7						
8	Legumes	**stock**	**sold**	**stock**	**sold**	**Average sold**
9	beans	7	5	9	7	=AVERAGE(C9,E9)
10	peas	10	10	8	9	=AVERAGE(C10,E10)
11						
12	**Total**	=SUM(B4:B6, B9:B10)		=SUM(D4:D6, D9:D10)		

Fig. 8. Spreadsheet definition.

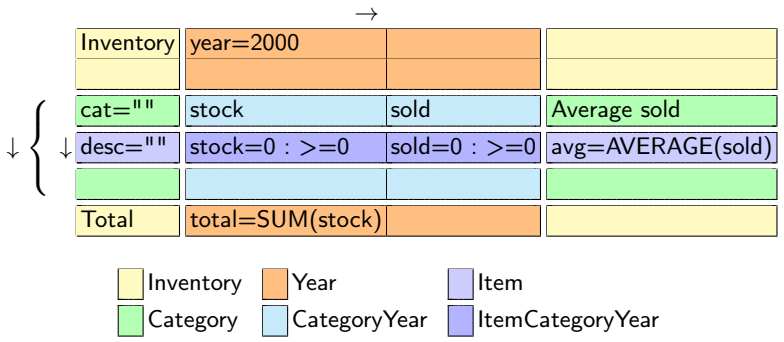


Fig. 9. Tabula specification of the spreadsheet.

Classes are laid out in a layered fashion, where the class that is used by another is at a higher level than the one that uses it. Thus, classes are not

broken down by other classes, but instead have parts of themselves covered by other classes. This characteristic is demonstrated in Fig. 10 with the ordering relation between the classes show in Fig. 11.

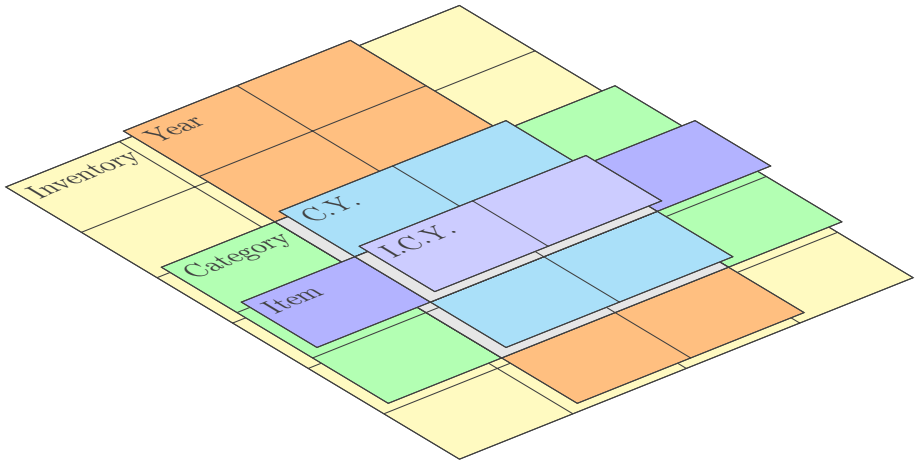


Fig. 10. Class layers for the inventory example (Fig. 9).

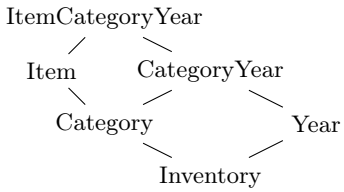


Fig. 11. Class ordering for the running example (Fig. 9).

In order to have a valid model, some rules in laying the classes must be followed:

- There must always be a base class with the size of the Tabula grid. In the running example, this base class is **Inventory**.
- Inner classes must either use the whole height or the whole width of the class below. In the former case, the class is called a *vertical class*, possibly expanding to the right, and in the latter case the class is called an *horizontal class*, possibly expanding down. This rule prevents spreadsheet layout deformation when adding new instances of a class.
- An horizontal class must leave a line above and another below of the class below. A vertical class must leave a column to the left and another one to

the right of the class below. This rule prevents ambiguity in the layer order and allows for a simpler visual representation. Note that there is no need to leave space between two classes at the same level.
- Two classes are allowed to intersect when one is contained by the other, i.e., the intersection area is the area of the contained class, or when a class is a vertical one and the other is an horizontal class.
- When a vertical class intersects with an horizontal one, a relation class must use the intersection area.
- Only relation classes can expand both to the right and down at once.

Note that this rules have a similar objective as the tiling rules for ClassSheets.

The notion of layers and class ordering is also used to define which class an attribute belongs to. Thus, an attribute belongs to the top-most class over that cell. The name resolution and translation to cell references in the instance works like in ClassSheets.

3.2 Model-Driven Spreadsheet Evolution

In a model-driven spreadsheet development environment, spreadsheet users can reason about large and complex data just by looking at the model. In this setting, spreadsheet users are not only able to edit/transform the instance (as in regular spreadsheet systems), but also to edit/transform the model. However, after a transformation in one of the software artifacts (model or instance), the model-driven environment has to guarantee their synchronization. That is to say, the instance has always to conform to the model, after every instance or model transformations.

In our previous work, model/instance co-evolution was applied to ClassSheets with a bidirectional transformation engine [4,6] that is able to perform a change in the ClassSheet and then have the respective spreadsheet co-evolved correspondingly. Moreover, this transformation engine is also able to perform changes in the spreadsheet that possibly breaks conformance with the model, but that co-evolves the model in order to keep the conformance, and therefore the bidirectional capability.

We have adapted and improved such bidirectional transformation engine to guarantee synchronization after a Tabula or a spreadsheet instance evolution. The structure of the bidirectional transformation engine is shown in Fig. 12: the engine includes a set of transformation operations both on the model t_M and on the instance t_I. It also defines two mapping functions (to and $from$) between such operations, that convert transformations in the model to transformation in the instance and *vice versa*. These two functions guarantee the synchronization of model and instance: An operation on the model (instance) is transformed to operations in the instance (model), such that after applying the transformations in the model and instance the conformance relation is guaranteed.

The arrow *create* from a Tabula model to a spreadsheet instance means that from the model an initial instance can be generated, where the model serves as a type system. Thus, when editing the generated instance, end users are forced

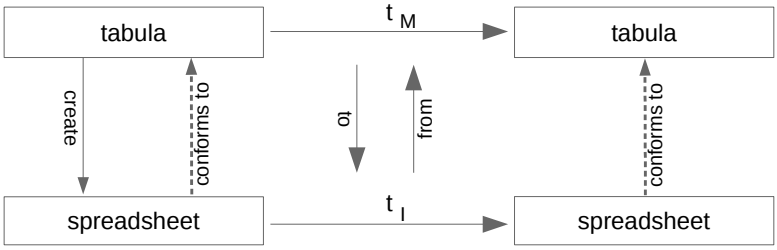


Fig. 12. Diagram of the bidirectional transformation engine.

to conform to the model. That is to say that they have to define spreadsheet data/value with that specific Tabula type.

In this paper we omit the definition of this bidirectional engine. Such definition is provided in [10].

4 Tabula With Real-World Spreadsheets

In order to assess the expressiveness of Tabula, when compared to the ClassSheet model, let us consider a widely used spreadsheet for a personal budget provided by Microsoft[3].

The Microsoft budget spreadsheet template has 14 columns and 91 rows, which users can reuse and edit by adding more data (rows) to the spreadsheet. Column-wise, the first one contains labels for each row, the twelve next columns contain the data for the twelve months of a year, and the last one contains the total for the year for each item. Row-wise, the spreadsheet has a title, then a header for the columns, a set of rows for the income, and then sets of rows for the expenses, grouped into categories (*home*, *daily living*, *transportation*, *entertainment*, *health*, *vacations*, *recreation*, *dues/subscriptions*, *personal*, *financial obligations* and *misc. payments*). The categories have 5 items in average.

In fact, this spreadsheet was used in an empirical study [2] to evaluate the performance of spreadsheet users when using model-driven spreadsheets. In that study the business logic of the budget spreadsheet was modeled by a ClassSheet. ClassSheets, however, could not capture the essence of the spreadsheet, resulting in a large and hard to understand and maintain model. A fragment of this ClassSheet model is shown in Fig. 13.

Because nested repetitions are not supported by ClassSheets, the model of the spreadsheet has to contain the concrete type of expenses (11 in total). This result in a visual ClassSheet that has 14 columns and 42 rows, Moreover, the formulas have also to refer to a large number of unique attributes, making their definition more complex and prone to errors. Finally, users cannot extend the

[3] *Basic personal budget* spreadsheet available at:
`https://templates.office.com/en-us/Basic-personal-budget-TM16400272`

	A	B	⋯	C
1	**Personal Budget**			
2		month=""		Year
3	Total expenses	expenses=SUM(home,dailyliv		total=SUM(expenses)
4	Cash short/extra	cash=**Income**.total-expenses		cash=**Income**.total-total
5	**Income**			
6	desc=""	income=0		total=SUM(income)
⋮				
7	Total	total=SUM(income)		total=SUM(year)
8	**Expenses**			
9	**Home**			
10	desc=""	expense=0		year=SUM(expense)
⋮				
11	Total	home=SUM(expense)		total=SUM(year)
39	**Misc. Payments**			
40	desc=""	expense=0		year=SUM(expense)
⋮				
41	Total	payments=SUM(expense)		total=SUM(year)
42				

Fig. 13. A ClassSheet for the Microsoft personal budget spreadsheet, with rows 12–38 omitted.

categories of the expenses in the spreadsheet, only. A change in the model is needed, too.

In comparison, the Tabula model presented in Fig. 14 for the same spreadsheet can model the different kinds of expenses and thus be more concise than the ClassSheet model. Moreover, there are no explicit reference to specific type of expenses (like, for example, *Home* and *MiscPayments* in the ClassSheet model). Finally, the Tabula also define constraints on types, like specification that only positive numbers can be added as expenses. The Tabula has the same number of columns (14), as the template is not dynamic in that axis, but has only 12 rows, describing the kinds of expenses in one expandable class.

A summary of the differences between both models is presented in Table 1, which for each entry contains the number of rows, columns, classes, attributes, input cells and formulas.

Table 1. Comparison of different metrics for the original personal spreadsheet, its ClassSheet and its Tabula.

	width	height	classes	attributes	input	formulas
personal budget	14	91	-	988	744	244
Tabula	14	12	5	81	27	54
ClassSheet	14	42	25	350	156	194
Tabula (dyn.)	3	12	10	16	6	10
ClassSheet (dyn.)	3	42	38	65	25	40

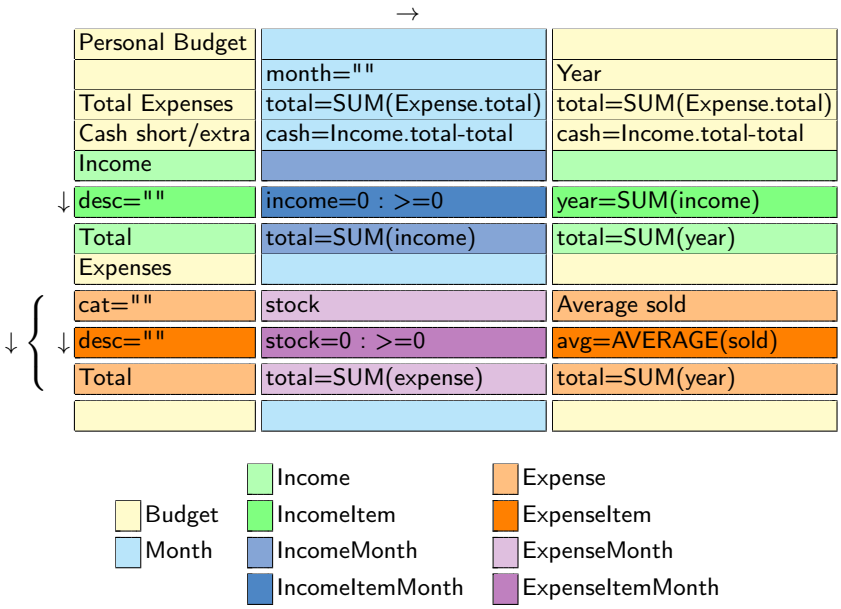


Fig. 14. A Tabula for the Microsoft personal budget spreadsheet.

The first entry in this table is the *personal budget* (as obtained from the link provided above). The second and fourth entry contains two Tabula models for this personal budget where the first model specifies the twelve months, and the second the months are defined via a repetition (this is the Tabula in Fig. 14). The third and fifth entries contains the metrics of the ClassSheet models for the personal budget (with the twelve months defined or with repetitions as shown in Fig. 13). As we can see in this table Tabula is 20%–26% smaller than the equivalent ClassSheet in terms of the number of classes and attributes.

5 Conclusion

In this paper we present the tabula language to model spreadsheet tables. Tabula was inspired by previous work on model-driven engineering, namely by the (visual) ClassSheet language, but it provides more flexibility and expressive power to model real-world spreadsheets.

We have modeled a real personal budget spreadsheet, and we have compared the ClassSheet and Tabula models. Our preliminary results show that Tabula is able to model such spreadsheet in a natural way, as opposed to ClassSheets.

Tabula defines a kind of type system for spreadsheets. As future work we intend to extend this type language such that it is possible to specify more precisely the type of values a spreadsheet cell may contain (for example, a cell may contain units like *seconds*, *inches*, etc.). This will prevent a larger set of errors in spreadsheets, e.g., by preventing operations with incompatible units.

Acknowledgments. This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project POCI-01-0145-FEDER-016718, by the bilateral project FCT/DAAD with ref. 441.00, and by a PhD scholarship from FCT with ref. SFRH/BD/112651/2015.

References

1. Coy, P.: FAQ: Reinhart, Rogoff, and the Excel Error That Changed History. Bloomberg: `http://www.bloomberg.com/news/articles/2013-04-18/faq-reinhart-rogoff-and-the-excel-error-that-changed-history` (2013 April)
2. Cunha, J., Fernandes, J.P., Mendes, J., Saraiva, J.: Embedding, evolution, and validation of model-driven spreadsheets. IEEE Transactions on Software Engineering 41(3), 241–263 (March 2015)
3. Cunha, J., Mendes, J., Saraiva, J., Fernandes, J.P.: Embedding and evolution of spreadsheet models in spreadsheet systems. In: 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 179–186 (Sept 2011)
4. Cunha, J., Fernandes, J.P., Mendes, J., Pacheco, H., Saraiva, J.: Bidirectional transformation of model-driven spreadsheets. In: Hu, Z., de Lara, J. (eds.) Theory and Practice of Model Transformations. Lecture Notes in Computer Science, vol. 7307, pp. 105–120. Springer (2012)
5. Cunha, J., Fernandes, J.P., Mendes, J., Saraiva, J.: Extension and implementation of classsheet models. In: Proceedings of the 2012 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 19–22. VLHCC '12, IEEE Computer Society (2012)
6. Cunha, J., Fernandes, J.P., Mendes, J., Saraiva, J.: MDSheet: A Framework for Model-driven Spreadsheet Engineering. In: Proceedings of the 34rd International Conference on Software Engineering. pp. 1395–1398. ICSE'12, ACM (2012)
7. Engels, G., Erwig, M.: Classsheets: Automatic generation of spreadsheet applications from object-oriented specifications. In: Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. pp. 124–133. ASE '05, ACM (2005)
8. Erwig, M., Abraham, R., Cooperstein, I., Kollmansberger, S.: Automatic generation and maintenance of correct spreadsheets. In: Proceedings of the 27th International Conference on Software Engineering. pp. 136–145. ICSE '05, ACM (2005)
9. Ireson-Paine, J.: Model Master: an object-oriented spreadsheet front-end. Computer-Aided Learning using Technology in Economies and Business Education (1997)
10. Mendes, J.: Evolution of Model-Driven Spreadsheets in a Collaborative Environment. Ph.D. thesis, Universidade do minho, Portugal (to appear) (2017)
11. Panko, R.: Facing the problem of spreadsheet errors. Decision Line, 37(5) (2006)
12. Panko, R.: Spreadsheet errors: What we know. what we think we can do. EuSpRIG (2000)
13. Wiels, V., Delmas, R., Doose, D., Garoche, P., Cazin, J., Durrieu, G.: Formal Verification of Critical Aerospace Software. AerospaceLab (4), 1–8 (May 2012)

How Spreadsheet Users Maintain Unfamiliar Spreadsheets

Daniel Kulesz (✉), Verena Käfer, and Stefan Wagner

Institute for Software Technology
University of Stuttgart, Germany
<first.last>@informatik.uni-stuttgart.de

Abstract. Spreadsheets are flexible tools that can be found in almost any project. Since many wrong decisions have been made due to faulty spreadsheets in the past, the question how these faults were introduced in the first place is of major concern. As spreadsheets are often shared between many workers, it is not uncommon for spreadsheet users to be asked to maintain a spreadsheet they have never seen before.

In prior work, we conducted two studies where 17 end-users and 42 undergraduate computer science students had to perform corrective and adaptive maintenance to a spreadsheet unfamiliar to them. This work provides an exploratory analysis of the effort these participants spent to get familiar with the spreadsheet before doing the first changes.

The results indicate that both beginners and advanced spreadsheet users started changing the spreadsheet prior to exploring its structure and inner workings in depth. More than 2/3 of the users performed changes which violated the original spreadsheet's design and structure. Only in about half of these cases the users recognized and corrected their errors later. Yet, surprisingly, the effort spent on getting familiar with the spreadsheet and the likelihood of not damaging its original design and structure or recognizing such an error during maintenance seem unrelated.

Keywords: Spreadsheet, End-User, Maintenance, User Actions, Analysis

1 Introduction

Spreadsheets are omnipresent and essential – most companies, associations, schools and other organizations would have a hard time running their businesses without them. It is assumed that there are millions of spreadsheet users and that billions of spreadsheets exist [13]. But spreadsheets also have a dark side: faulty spreadsheets have contributed to financial and reputational losses in recent years. Because of this, many researchers have investigated the use of spreadsheets and developed guides and methods for preventing and detecting anomalies in spreadsheets [3].

1.1 Problem Statement

We assume that it is usually desirable that the original design and structure of spreadsheets shall be retained during maintenance, i.e. if the spreadsheet is just slightly adapted, corrected or extended — even if the person carrying out the maintenance is not familiar with the spreadsheet. However, it is generally unknown how spreadsheet users approach maintenance tasks in spreadsheets and if changes they commit are prone to damaging the spreadsheet's original design and structure.

1.2 Research Objective

Our objective is to investigate how users maintain unfamiliar spreadsheets. For this, we formulated three research questions:

RQ1: How much effort do users invest in getting familiar with a spreadsheet before they change it?
RQ2a: Do the first changes committed by users during maintenance of an unfamiliar spreadsheet respect the original design and structure?
RQ2b: If not, do users realize this later and redo erroneous changes with respect to the original design and structure?

1.3 Context

In previous work, we conducted two studies in which 42 students from an undergraduate software engineering course and 17 experienced spreadsheet users had to learn and apply various techniques for inspecting and testing spreadsheets by using different tutorials [5,10]. To make these experiments more natural and less focused on testing, the main part of the experiments involved the maintenance of a considerably complex spreadsheet which the participants have not seen before. In this work, we conducted an exploratory analysis about how the participants approached this maintenance task by reusing the data from the two previous studies but viewing it from a completely different angle.

This paper follows the structure proposed by Jedlitschka et al. [4] with some slight deviations: after describing the background of the study, we explain the plan of the experiment, present the obtained results and discuss them before drawing a final conclusion and outlining future work. The graphical representations are based on the recommendations by Tufte [14].

2 Background

2.1 Technology Under Investigation

For the purpose of this exploratory study, we disregarded both the spreadsheet testing tool which was used in the experiments and the different types of tutorials the participants consumed. This was possible because all participants had the same

main task where they had to maintain the spreadsheet. Because the two original studies focused on correctness of the spreadsheets and not retaining the original design and structure, none of the involved testing techniques provided hints about violations of this aspect. Therefore, the technology under investigation boiled down to a single spreadsheet. The runtime environment consisted of Microsoft Excel 2013 and Windows 7.

2.2 Related Studies

As mentioned, there are two studies from the authors' previous work which provide the basis for this exploratory analysis.

In the first study, we analyzed the efficiency and effectiveness of users who learned and applied our tool named "Spreadsheet Inspection Framework" using only text tutorials, only video tutorials and a combination of both [5].

In the second study, we investigated the usefulness of the underlying approach of the tool "Spreadsheet Inspection Framework" named "Spreadsheet Guardian": we checked if the approach was easy to learn and to apply and if it provides benefits regarding the detection of anomalies and the gain of dependable confidence regarding the correctness of spreadsheets after maintenance [10].

To the best of our knowledge, there is no other existing work which analyzed the first "getting familiar" steps of spreadsheet users maintaining an unfamiliar spreadsheet. However, there is work about developers who maintain unfamiliar code. Many developers start to look for code that looks to be relevant for their current task and then follow its dependencies to (hopefully) other relevant code (e.g. [7]). Interestingly, in a study by Koeneman and Robertson, no participant tried to systematically understand all of the code, they only tried to understand code that looked relevant [8]. There is also a number of related work in the area of program comprehension (e.g. as discussed in [15]) but since it deals with comprehension issues of developers understanding traditional software, it is unclear which of the findings are directly transferable to spreadsheets.

2.3 Relevance to Practice

Gathering knowledge about how users "naturally" approach the maintenance of unfamiliar spreadsheets and if this negatively impacts the original design and structure is key for preventing faults which are introduced during maintenance of unfamiliar spreadsheets in the future. Similar investigations have been done for traditional software (e.g. [6,11]) and they unfolded information needs. Some of these information needs have been addressed later by novel tools and methods (e.g. [12]). Some work has been done on spreadsheet comprehension as well (e.g. [1,2]) but spreadsheet users in practice are still far from being adequately supported during spreadsheet maintenance activities.

3 Experiment Planning

3.1 Goals

For each research question we defined an experiment goal:

- Goal 1: Analyze users getting familiar with previously unfamiliar spreadsheets
  For the purpose of understanding which actions they took to get familiar with it
  With respect to the number of worksheet switches, selected cells and changed values in input cells
- Goal 2: Analyze users maintaining previously unfamiliar spreadsheets
  For the purpose of understanding if their changes respected the spreadsheet's original design and structure
  With respect to where the first changes were made and, if made incorrectly, whether this was recognized and corrected later

3.2 Participants

In total, we analyzed maintenance activities of 59 participants from two populations: (1) The first population were 42 undergraduate students (9 female) taking a software engineering course. They were mostly unexperienced with spreadsheets. Most (29) of them were in the age group "20-25". (2) The second population were 17 end-users (5 female) without a computer science background but experienced with spreadsheets. Most of them (14) were in the age group "26-35".

As part of a lecture, the undergraduate students had to take part in a study to pass the course. They had other choices than our study, but we had to take every participant who stated that he or she wanted to take part in our study. This was different for the end-users: they took part in the study fully voluntarily and we compensated their efforts by paying each of them 10 euros for participating. Also, we had planned to do two other (but related) experiments with these volunteers. Therefore, we asked them to answer a few questions about their work habits and experience with spreadsheets and selected only experienced users for this maintenance task.

We reported more details about demographics and the participants' previous spreadsheet expertise in [10] and [5]. There, we also provided details about the selection process of our end-user population.

3.3 Procedure

Prior to taking part in the study, all participants were asked to fill out an online survey. In the experiments, all participants were asked to watch an introduction video about spreadsheets and spreadsheet anomalies. Afterwards, the participants learned how to use and apply our tool Spreadsheet Inspection Framework. In the main task (on which this analysis focuses) they had to understand and maintain an unfamiliar spreadsheet. Afterwards, we asked all participants to fill out a final survey. Details of the overall procedure can be found in [5] and [10].

3.4 Experimental Material

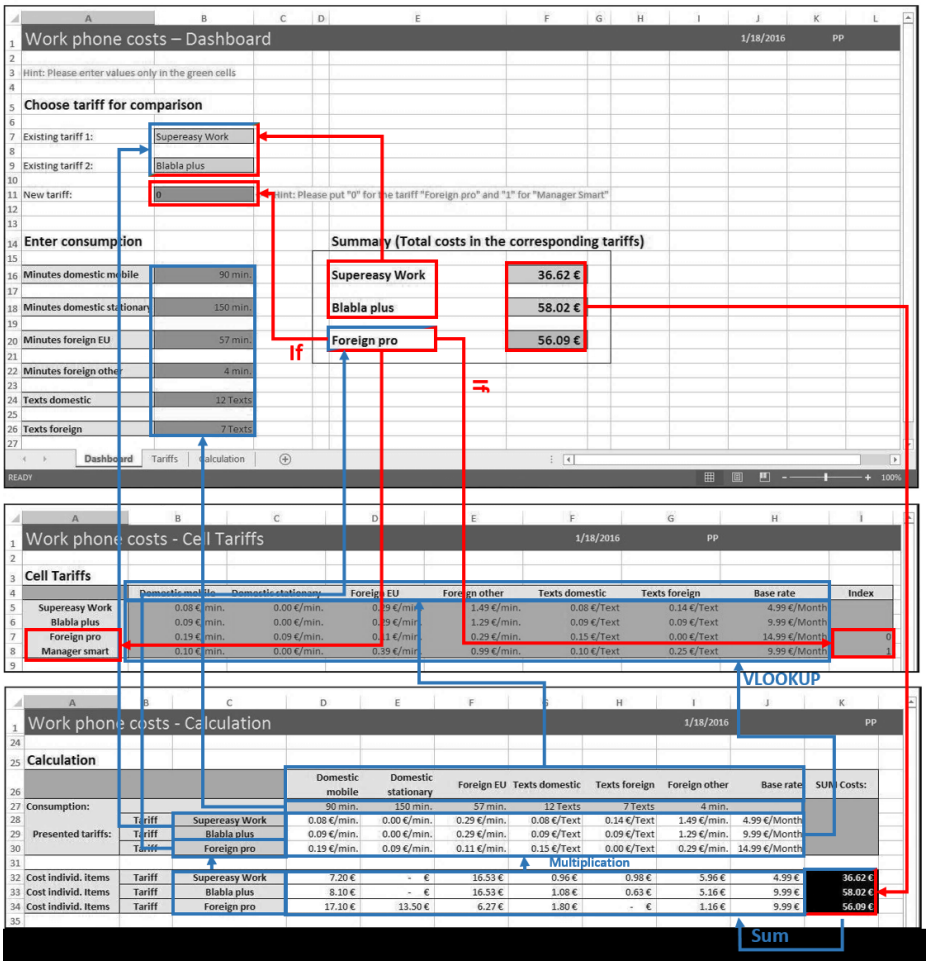


Fig. 1: Overview over the structure of the three worksheets. If not stated otherwise, the arrows show that one cell referenced another one.

At the beginning of the maintenance task, the participants received a written description (1 page in A4 format) of a fictive scenario which described why maintenance of the given spreadsheet was required: Petra (the creator of the spreadsheet) was injured in an accident and was unavailable for three weeks but the spreadsheet was urgently needed. The given spreadsheet named “tariff comparison” allows comparing phone tariffs. Since such calculations are common sense, we believe that it made the task domain-independent. The spreadsheet was meant to be used in the following way: the user entered the amount of consumed

minutes and texts for various destinations (foreign, domestic) as input values and the spreadsheet calculated the total costs for this consumption in three tariffs. Each tariff had different rates for minutes and texts depending the destination. Two of these tariffs were meant to be fixed (in the scenario, it were the tariffs that were currently used in the fictive company) while the third tariff could be switched by changing the value in an input cell.

The spreadsheet had three worksheets. The first worksheet named "Dashboard" included input and output cells as well as some simple logic and references to the third worksheet, but no data or calculations. The second worksheet named "Tariffs" only contained pure data without any logic or references to other worksheets. The third worksheet was named "Calculation" and contained all the calculations which were based on the data. Apart from sums and sumproducts (not using the sumproduct formulas) it also contained a few VLOOKUP formulas. The mean part here was that the VLOOKUP formulas did not have their index values hard coded but referenced other cells which were hidden by using white text on white background, making comprehension of the inner workings hard. An overview over the formulas we used can be seen in Figure 1

To make this more natural and less artificial we did not create the spreadsheet ourselves. Instead, we verbally specified a use case with a set of requirements and asked an experienced spreadsheet user (a male, 32 years old mechanical engineer) to develop a spreadsheet which would satisfy them. We only slightly adopted the outcome by extracting and hiding the indexes in the VLOOKUP formulas to make maintenance of the spreadsheet more challenging.

Except for videos (which are available upon request), all experimental material is available from our open data repository [9].

3.5 Tasks

The main task (maintenance of an unfamiliar spreadsheet) was split in a number of sub-tasks. For this analysis, we considered the two sub-tasks as the central part of a spreadsheet user getting familiar with a previously unfamiliar spreadsheets. More precisely, we asked the participants to do the following (exact wording, only translated from German):

1. You can find the last version of Petra's Excel sheet in the folder "study" on your desktop (tariffcomparison.xlsx). Open the spreadsheet and look around in it.
2. Play around a little bit with the spreadsheet, e.g. by changing the calculation to the Tariff "Manager smart" (cell B11 in the Dashboard).

The three following sub-tasks were irrelevant for this exploratory analysis: we asked the participants to activate certain inspection rules in our tool and run the inspection. If done correctly, the tool reported some findings which the participants were asked to fix. Afterwards, we asked the participants for two

adaptive maintenance activities: (1) They should add a third tariff with which the two "harddcoded" ones could be compared, and (2) they should add a new consumption category for "domestic" minutes and texts.

3.6 Hypotheses

As stated in our goals, we wanted to measure the effects of the "getting familiar intensity" of users in previously unfamiliar spreadsheets. Since we already had relevant data from two other experiments, did not prepare a separate experiment for answering these questions but, instead, analyzed this data in an exploratory fashion.

For the purpose of this data analysis, we defined a "getting familiar coverage level" (details in the next section) and formulated the following hypotheses:

- H_1: There is a difference between the "getting familiar coverage levels" of advanced and beginner spreadsheet users
- H_{1_0}: There is no difference between "getting familiar coverage levels" of advanced and beginner spreadsheet users
- H_{2_a}: Users who achieve a higher "getting familiar coverage level" before changing a previously unfamiliar spreadsheet will more likely preserve its original design and structure.
- $H_{2_{a_0}}$: Users who achieve a higher "getting familiar coverage level" before changing a previously unfamiliar spreadsheet will less or as likely preserve its original design and structure.
- H_{2_b}: If users introduce changes which violate the spreadsheet's original design and structure, they are more likely to correct their errors if they previously achieved a higher "getting familiar coverage level".
- $H_{2_{b_0}}$: If users introduce changes which violate the spreadsheet's original design and structure, they are less or as likely to correct their errors if they previously achieved a higher "getting familiar coverage level".

3.7 Analysis Procedure

For the purpose of this evaluation we transcribed the screen recordings of the maintenance tasks, producing a transcribed stream of actions for each recording. This included the following:

- Switch to worksheet (Dashboard / Tariffs / Calculation).
- Selection of a cell in worksheet (Dashboard / Tariffs / Calculation).
- Changing the value of an input cell in worksheet (Dashboard / Tariffs / Calculation). Here, we checked whether only the input cell for changing the tariff was changed or also at least one input cell for entering the consumption.

To evaluate whether the original design and structure was violated, we inspected if the changes for including the additional tariff were done in the correct worksheet (that is only in the worksheet "Tariffs" and not additionally in

the worksheet "Calculations"). If the changes were done in the wrong worksheet, we inspected if they were corrected in the further course of maintenance or if they stayed like that to the end.

There are no established metrics on a "getting familiar coverage level". Therefore we decided to calculate it combining several measurement points. We counted the actions the participants took and assigned one or two points as can be seen in Table 1. We then added the points to get the "getting familiar coverage level".

Table 1: Getting familiar coverage level

Action	1 point	2 points
Selection in dashboard	$\geq$ 2 selections	$\geq$ 3 selections
Insertion in dashboard	$\geq$ 1 insertions	$\geq$ 2 insertions
Switched to tariffs	$\geq$ 1 switches	-
Selection in tariffs	$\geq$ 1 selections	$\geq$ 2 selections
Switched to calculation	$\geq$ 1 switches	-
Selection in calculation	$\geq$ 1 selections	$\geq$ 2 selections

4 Execution

As this exploratory analysis did not involve additional executions of our experiments, there was no preparation necessary. However, although we expected to be able to easily transcribe the recordings, we stepped over a few caveats which forced us to formulate a number of "counting rules" and to perform several rounds of transcriptions. We ended up with the following list of "unexpected" actions and how we transcribed them:

- If the participant navigated through cells by using (and holding) the cursor keys on the keyboard, we counted this only as a single selection. But in case such a cursor-key-navigating participant changed the direction, we counted this an additional selection. Also, if the participant made stops at cells longer than 500ms we counted these as selections.
- We only counted selections of cells other than the currently selected one, i.e. if the participant clicked in a cell that was already selected we did not count this as a selection.
- Entering of values in input cells was only counted if the spreadsheet was recalculated (e.g. by pressing the return key or clicking in another cell).
- Entering of values using undo/redo was also counted as change.

5 Analysis

This chapter describes our results and how we checked them for statistical relevance.

5.1 Descriptive Statistics

Goal 1 – Actions to get Familiar

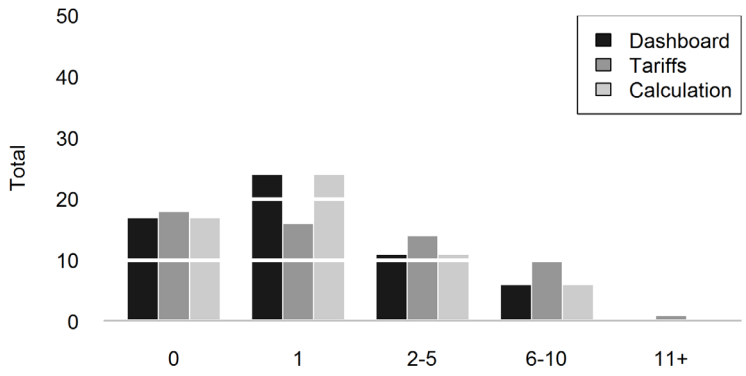


Fig. 2: Number of switches to the worksheets

Switches Between the Worksheets Figure 2 shows that most participants (17) either never visited the Dashboard worksheet a second time (0 switches to the dashboard worksheet as it was the initial worksheet when opening the spreadsheet) or returned there only one time (24). For the other worksheets, on average 17.5 participants never visited them, and only 20 visited them once. Only a minority of the participants visited the worksheets several times.

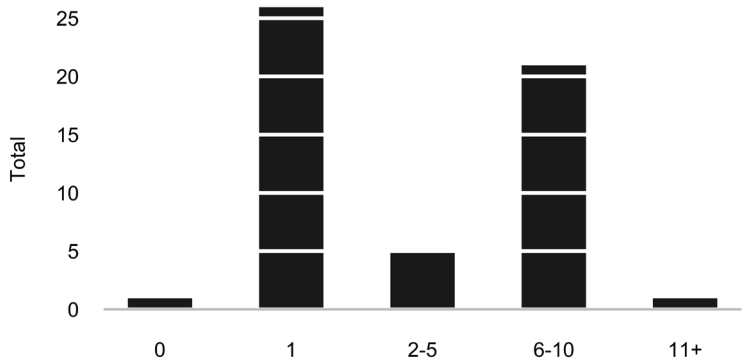


Fig. 3: Number of insertions in the dashboard

Value Changes Figure 3 shows that in the Dashboard worksheet, most participants changed either one value (26) or 6-10 values (21). In the other worksheets there were no insertions so they are not displayed in the figure.

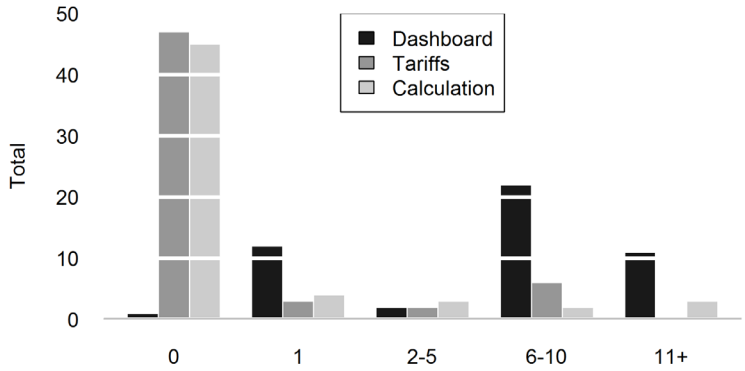
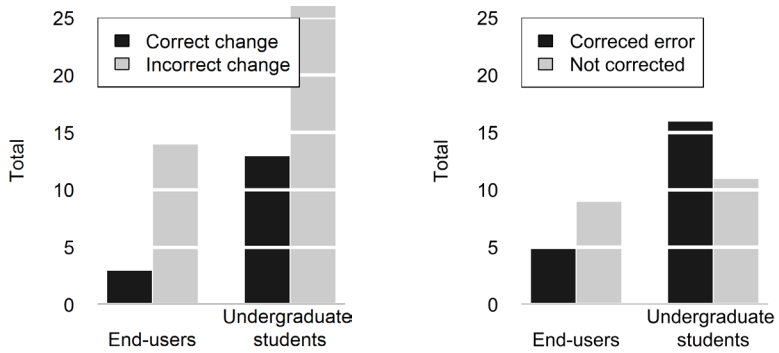


Fig. 4: Number of selections in the worksheets

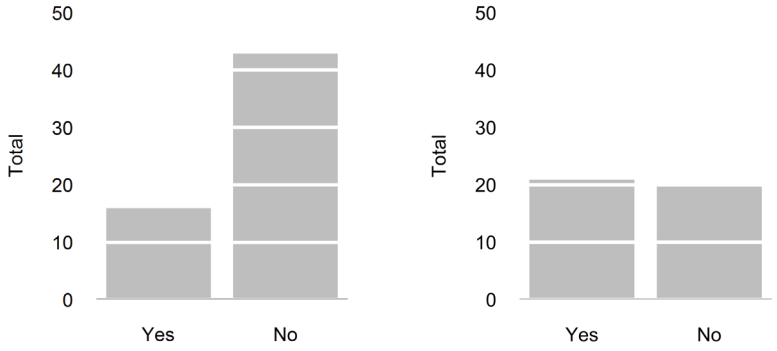
Selecting Cells in the Worksheets As shown in Figure 4, most participants did not select any cells in the Calculation (45) and Tariffs (47) worksheets. In the Dashboard worksheet, most participants selected 6-10 values (22).



First change in the right cells Correction, if wrong cell

Fig. 5: First change per group

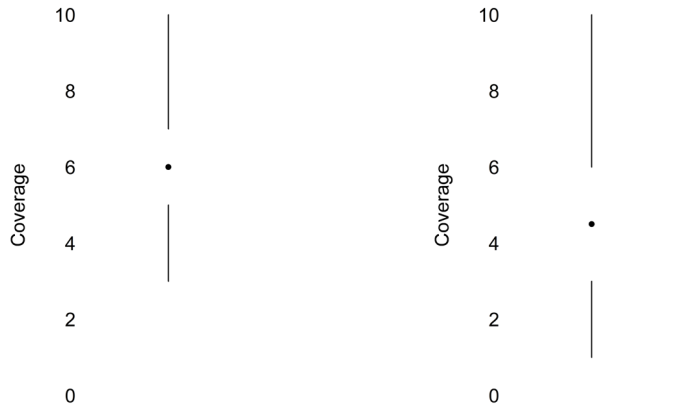
Goal 2 – Changing the Spreadsheet As Figure 5 and 6 show, 43 participants did not do the first change with respect to the spreadsheet's original design and



First change in the right cells Correction, if wrong cell

Fig. 6: First change combined

structure. Half of these participants corrected this later, while the other half did not. Surprisingly, in the end-user group, more participants did not correct such an error whereas in the students group more participants did correct such an error.



End-users Undergraduate students

Fig. 7: Coverage

Coverage Figure 7 shows that, on average, the end-user group achieved a better average coverage level than the students group. In both groups, the coverage level was widely spread around the mean.

5.2 Hypothesis Testing

Although this is an exploratory study, we tested all our hypotheses using the Wilcoxon Signed-Rank Test.

H_1 – Getting Familiar Coverage Level We checked if there was a significant difference between the students and the end-users regarding the achieved coverage level but there was none ($p = 0.1896$). Thus, we cannot reject H_{1_0}.

H_{2_a} – Getting Familiar Coverage Level and Correct First Change Even for the coverage level we could not find any significant differences for a first correct cell correct ($p = 0.658$). We cannot reject $H_{2_{a_0}}$.

H_{2_b} – Getting Familiar Coverage Level and Fault Correction Accordingly, we also could not find any significant differences for coverage and fault correction ($p = 0.2378$). We cannot reject $H_{2_{b_0}}$.

5.3 Further Evaluation

Since we could not find any significant differences regarding the getting familiar coverage level and the first change respectively correction of errors, we decided to have a look at other possible correlations. For this purpose, we inspected each single component we used for the computation of the coverage level, calculated its median (where appropriate) and ran more t-tests (Table 2). Additionally, we computed correlations using the Pearson correlation coefficient (Table 3). As it can be seen, neither worksheet switches, nor the number of insertions or the number of selections showed a significant correlation.

6 Discussion

6.1 Evaluation of Results and Implications

H_1 – Getting Familiar Intensity As the statistical tests have shown, on average there were only slight differences regarding the intensity of getting familiar with a previously unfamiliar spreadsheet between our two populations. However, the span between particular participants within their population differed remarkably. Therefore, to our surprise, it seems that previous knowledge of spreadsheets is not related to the effort users spend before maintaining an unfamiliar spreadsheet.

Another remarkable fact is also that about 40 percent of the participants did not visit any of the other worksheets before starting the maintenance tasks. We can think of two explanations here: either the participants did not recognize that there was more than one worksheet or they simply did not feel the need to visit any other worksheet if the minimum amount of work required for the actual task (changing the tariff for the comparison) could be done without a worksheet change. Unfortunately, the available data from the previous studies does not allow for more insights regarding this phenomenon.

Table 2: P-values for correlations

Metric	First change correct	First change incorrect (but corrected later)
Worksheets switched (0 vs ≥ 1)	0.4256	0.7123
Number of insertions made in dashboard (≤ 1 vs > 1)	0.1665	0.3191
Number of selections (≤ 1 vs > 1)	0.5	0.2471
Worksheets switched ($\leq$ mean (4.25) vs $>$ mean)	0.6787	0.3365
Number of insertions made in dashboard ($\leq$ mean (2.83) vs $>$ mean)	0.2234	0.1453
Number of selections ($\leq$ mean (9.05) vs $>$ mean)	0.1199	0.7107
Worksheets switched ($\leq$ median (3) vs $>$ median)	0.4294	0.5719
Number of insertions made in dashboard ($\leq$ median (2) vs $>$ median)	0.2234	0.1453
Number of selections ($\leq$ median (5) vs $>$ median)	0.1797	0.4461
Changed tariff value in dashboard (yes vs no)	0.7415	0.8585
Changed other cell in dashboard (yes vs no)	0.2789	0.2719

H_{2_a} – Getting Familiar Intensity and Correctness of first Changes made It is alarming to see that more than 2/3 of our population broke the spreadsheet's original design and structure with their first changes. Intuitively, we would have expected that users who spent considerable effort getting familiar with the spreadsheet before would be in the group who did the change right. Surprisingly, as our analysis shows, this was not the case for both populations — on average, neither for the beginners, nor for the advanced spreadsheet users profited if they spent more effort at the beginning of the maintenance task.

H_{2_b} – Getting Familiar Intensity and Error Correction About half of the participants who introduced changes which broke the spreadsheet's original design and structure recognized and corrected their error. However, our statistical analysis also could not find a significant correlation with the intensity they initially spent for getting familiar with the spreadsheet.

Further Evaluation The further evaluation for correlations between particular ingredients to our coverage for the "getting familiar intensity" and the spreadsheet user doing changes respecting the original design and structure also yielded no particular data points. While this serves as an indicator that the approach we chose for computing the coverage was not severely flawed, it also further shows the missing relatedness between the analyzed facts in our setting.

Table 3: Correlation coefficients

	First change correct	First change incorrect (but corrected later)
Worksheets switched	0.1203	-0.0498
Number of insertions made in dashboard	0.106	0.1249
Number of selections	0.1737	-0.1044
Changed tariff value in dashboard	0.0801	0.162
Changed other cell in dashboard	-0.0791	-0.1005

6.2 Threats to Validity

We discussed general threats to validity of our study in [10] and [5]. In addition to these, we have identified a number of unique threats to validity for this exploratory analysis and split them into three groups: construct validity (CV), internal validity (IV) and external validity (EV). We discuss them in the following:

- (CV) The underlying studies for this exploratory analysis were not intentionally designed for investigating how spreadsheet users maintain unfamiliar spreadsheets. While of the activities these users performed were thus uninteresting for our study, we did not identify activities which would harm our results either.
- (CV) There is a chance that the participants of our studies were distracted by the novel spreadsheet testing techniques they had to learn and apply, so they did not maintain the spreadsheet the way they would normally do. However, one can also argue that this might have had a positive impact: because the participants did not know that their maintenance performance was later to be evaluated, they possibly did the maintenance the usual (natural) way.
- (CV) Before doing adaptive maintenance the participants had to do corrective maintenance. Doing so, it is very likely that they have gained much knowledge about the spreadsheet's inner workings. If the participants would have been asked to do only the adaptive maintenance the results might have been different. On the other hand, as the vast majority of the participants still committed the changes initially in the wrong place, the actual impact of the corrective maintenance task is questionable.
- (IV) By only analyzing the screen recordings we could not distinguish whether the participants who did the changes violating the original design and structure did not recognize their errors later or if they only did not correct them although they recognized them.
- (IV) Although the task description should have been quite clear, however, if participants misunderstood it and intentionally added a third "hardcoded" tariff (although they understood the spreadsheet's original design and structure) our analysis would not reflect this.
- (EV) Although we had a total of 59 participants, the sample size is not sufficient for generalizing our findings.

- (EV) The population of undergraduate students taking software engineering lectures were certainly not perfect representatives of typical spreadsheet users. However, we were surprised that their spreadsheet skills were that low, yet making them somehow end-user comparable.
- (EV) The analysis evaluated the maintenance of one particular spreadsheet. It might be representative for a class of comparable spreadsheets, however, since there is a sheer endless variety of different spreadsheets the effects observed in this analysis will certainly vary.

6.3 Lessons Learned

Before doing this exploratory analysis, we were quite certain that we would be able to find facts which would support our theory of a correlation between the "getting familiar intensity" and its impact on the correctness of changes. Surprisingly, this turned out to be not the case.

7 Conclusions and Future Work

7.1 Summary

This exploratory analysis on a population of 42 undergraduate students and 17 advanced spreadsheet users using transcriptions of screen recordings initially made for two other studies found a number of interesting results: (1) The effort users spend for getting familiar with previously unfamiliar spreadsheets before changes them broadly varies on a individual basis but seems unrelated to the spreadsheet expertise of these users. (2) The first changes made in a spreadsheet by users previously not familiar with it is (very) likely to break its original design and structure, be it done by beginners or advanced spreadsheet users. (3) Only half of the users who initially break a spreadsheet's original design and structure correct their errors later. (4) The effort spent for getting familiar with a previously unfamiliar spreadsheets does not pay off in terms of later correctness of changes during maintenance or the correction of errors.

Summarizing our findings, it looks like instead of doing a big up-front investigation before doing changes, spreadsheet users rather tend do simply do changes first and think about them later. However, when they do so it does not lead to as bad results as one might have initially assumed — at least regarding the erosion of the spreadsheet's original design and structure. As we have reported in previous work, the impact on correctness is way more dramatic.

7.2 Future Work

To further explore the space of maintenance activities on unfamiliar spreadsheets and derive more generalizable findings it will be necessary to analyze maintenance of spreadsheets in other settings, using spreadsheets with different complexities and different designs. For instance, if spreadsheet users would get an easier task

using an easier-to-maintain spreadsheet this could result in higher (computational, not structural) correctness of the maintenance activities. Maybe the effort of getting familiar with a previously unfamiliar spreadsheet would be more likely to pay off in such a setting?

References

1. Ballinger, D., Biddle, R., Noble, J.: Spreadsheet visualisation to improve end-user understanding. In: Proceedings of the Asia-Pacific symposium on Information visualisation-Volume 24. pp. 99–109. Australian Computer Society, Inc. (2003)
2. Hermans, F., Pinzger, M., Van Deursen, A.: Supporting professional spreadsheet users by generating leveled dataflow diagrams. In: Proceedings of the 33rd International Conference on Software Engineering. pp. 451–460. ACM (2011)
3. Jannach, D., Schmitz, T., Hofer, B., Wotawa, F.: Avoiding, finding and fixing spreadsheet errors–a survey of automated approaches for spreadsheet qa. Journal of Systems and Software 94, 129–150 (2014)
4. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting Experiments in Software Engineering. In: Guide to Advanced Empirical Software Engineering, pp. 201–228. Springer London (2008)
5. Käfer, V., Kulesz, D., Wagner, S.: What is the best way for developers to learn new software tools? an empirical comparison between a text and a video tutorial. The Art, Science, and Engineering of Programming 1(2) (2017)
6. Ko, A.J., DeLine, R., Venolia, G.: Information needs in collocated software development teams. In: Software Engineering, 2007. ICSE 2007. 29th International Conference on. pp. 344–353. IEEE (2007)
7. Ko, A.J., Myers, B.A., Coblenz, M.J., Aung, H.H.: An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. IEEE Transactions on Software Engineering 32(12), 971–987 (Dec 2006)
8. Koenemann, J., Robertson, S.P.: Expert problem solving strategies for program comprehension. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 125–130. CHI '91 (1991)
9. Kulesz, D., Käfer, V.: Data for Spreadsheet Guardian (Nov 2016), `https://doi.org/10.5281/zenodo.188896`
10. Kulesz, D., Käfer, V., Wagner, S.: Spreadsheet guardian: An approach for protecting semantic correctness throughout the evolution of spreadsheets. arXiv preprint arXiv:1612.03813 (2016)
11. von Mayrhauser, A., Vans, A.M., Howe, A.E.: Program understanding behaviour during enhancement of large-scale software. Journal of Software: Evolution and Process 9(5), 299–327 (1997)
12. O'Callahan, R., Jackson, D.: Lackwit: A program understanding tool based on type inference. In: In Proceedings of the 19th International Conference on Software Engineering. Citeseer (1997)
13. Scaffidi, C., Shaw, M., Myers, B.: Estimating the numbers of end users and end user programmers. In: Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on. pp. 207–214. IEEE (2005)
14. Tufte, E.R.: The Visual Display of Quantitative Information, vol. 2 (2001)
15. Von Mayrhauser, A., Vans, A.M.: Program comprehension during software maintenance and evolution. Computer 28(8), 44–55 (1995)

All links were last checked on April 2, 2017.

TAPASPlay: a Tangible Game-Based Learning Approach to Foster Computational Thinking Skills

Alessio Malizia[1], Tommaso Turchi[1], Federico Danesi[2], Daniela Fogli[2], David Bell[1]

[1]Computer Science Dept., Brunel University London, Uxbridge, UK
{Alessio.Malizia, Tommaso.Turchi, David.Bell}@brunel.ac.uk
[2]Dept. of Information Engineering, University of Brescia, Brescia, Italy
f.danesi003@studenti.unibs.it, daniela.fogli@unibs.it

Abstract. This paper describes TAPASPlay, a system that combines Game-Based Learning with Tangible User Interfaces and Virtual Reality to foster Computational Thinking skills in an innovative way. Two players are asked to play the role of alchemists who must forge swords and shields to fight each other. Swords and shields are created through a puzzle-based approach with a Tangible Interface using the two players' smartphones and displayed on a tabletop surface. Finally, the players will have to wear their cardboards and enjoy the battle between two characters, one for each player, in Virtual Reality.

Keywords: Tangible interaction, Computational Thinking, Game-Based Learning, Augmented Surface, Virtual Reality.

1 Introduction

The impact and relevance of learning coding skills in modern society is demonstrated by many emerging initiatives such as the Hour of Code[1]. Nonetheless, these initiatives – as with coding itself – are not just about programming as they endeavour to foster Computational Thinking (CT) skills. There is little agreement on an exact definition of CT and over the years many different definitions have emerged. Brennan and Resnick [1] introduced a "low level" approach based on visual languages, a powerful paradigm studied for many years; indeed, most of the successful puzzle-based programming tools, such as Scratch, have been based on visual languages. Furthermore, Jeannette Wing [7] introduced a higher level definition of CT based on problem solving; logically the main concepts of abstraction and decomposition introduced by Wing are strongly related to problem solving activities. The third approach we consider here, by Kazimoglu et al. [2], adds to problem solving other important concepts, namely algorithm building, debugging, simulation and socialization, which are considered fundamental skills characterizing CT. We argue that exploiting our innate dexterity for ob-

[1] https://hourofcode.com

jects' manipulation in the physical world and learning by playing - i.e. through making activities, as remarked in [3], could be an effective way of aiding concrete operational thinkers to grasp abstract concepts involved by coding, therefore fostering their CT skills. Inspired by these literature work, we developed a novel system, called TAPASPlay that exploits Game-Based Learning, Tangible User Interfaces and Virtual Reality to foster CT skills in people with little or no knowledge of programming.

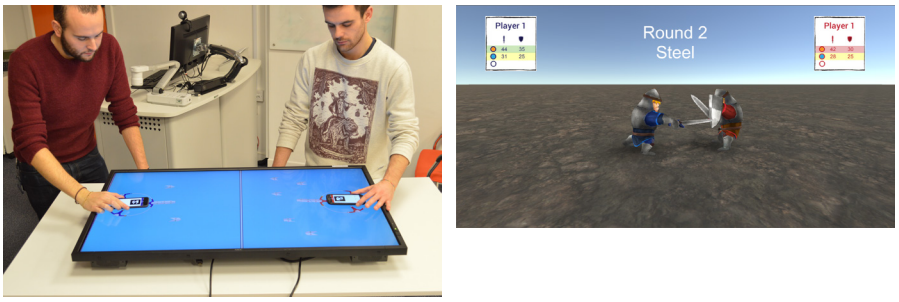
2 TAPASPlay

In a recent study [5] we introduced TAPAS (TAngible Programmable Augmented Surface), a Blocks Programming Environment that allows users to develop simple workflows by assembling different services using their smartphones as tangible interfaces. The interaction is carried out using tangible objects (i.e. smartphones) and the digital blocks are projected over a tabletop surface, making it fun and easy to use. In TAPASPlay our system has been extended to include a Game-Based Learning (GBL) approach to foster CT skills. Since digital games are attractive and engaging for all groups of people, GBL has been proposed as one pedagogical framework for developing CT skills in an innovative way [6]. Currently, the literature in GBL focuses on two popular approaches to facilitate the development of CT skills and learning of introductory programming: learning through designing games and learning through game-play. While the first approach has been extensively studied (e.g. Scratch, Alice and AgentSheets) the game-play approach is quite in its infancy; indeed, few studies demonstrate how game-play can be associated with CT and how the education of introductory programming can be supported by playing games [2]. The novelty of TAPASPlay is to combine game-play with Tangible User Interfaces and Virtual Reality to teach Computational Thinking skills. The game is intended for an audience with little or no experience in programming, e.g. students, practitioners, designers, and likely to be interested in acquiring CT skills that allow them to carry out end-user development activities [4].

In this turn-taking game the two players will play the role of alchemists forging three swords and three shields to fight each other. The swords and shields will be forged using a puzzle-based approach with a Tangible Interface as in TAPAS (Figure 1-a) and the fight will be visualized in Virtual Reality (Figure 1-b).

More precisely, the game takes place in three sequential phases: 1) forging the three swords, 2) forging the three shields and 3) visualizing the fight.

In the first phase, each player (or alchemist) has to forge three swords, thus defining three offensive strategies. In order to achieve that, the players interact with the TAPAS surface and connect the pieces of a puzzle (sword elements) using a digital halo, surrounding each of their smartphones. Every sword is made of a different metal (bronze, iron or steel), which corresponds to a different shape of the fitting pieces of the puzzle. The three strategies are defined one at a time. A strategy is a sequence of transformations of metals taken from a set of available puzzle pieces. The sets of the available transformations are identical for both players (Figure 2-a), allowing to fairly comparing their outcomes.

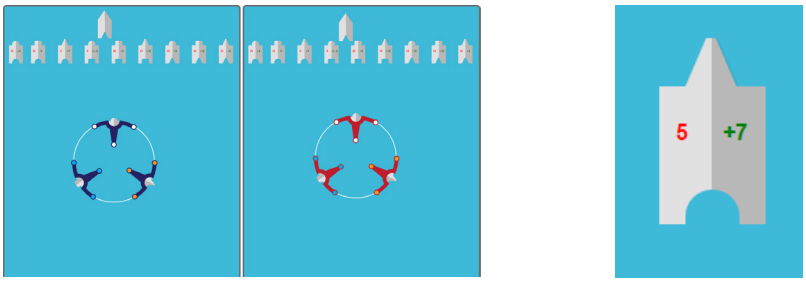


(a) (b)

Figure 1. Alchemists interacting with TAPASPlay (a) and the fight in VR (b)

A transformation (Figure 2-b) is a tuple of four values: an input metal (given by the shape on the bottom), an output metal (the shape on the top), the amount of energy points required (on the left) and the force points gained (on the right). The initial state is defined by a triple of values: a metal (randomly chosen and representing the objective of the strategy), an initial amount of force points (0 by default) and an initial amount of energy points. The force points are a measure of the quality of a strategy and depend on the transformations. The energy points are needed to apply those transformations, each of which requires a certain amount of energy points. The final state is reached when a player is satisfied with the force points obtained by the sequence of transformations and the output metal (bronze, iron or steel) matches the objective randomly set by the game. Each transformation will modify force points but will require energy points, therefore the best strategy consists of finding a trade off between maximising force points at the expenses of energy points. This mechanism fosters the design-debug-run stages, three key aspects of Computational Thinking [2]: analysis, abstraction, decomposition, and automation all come into this game-play, in particular abstraction and analysis. At the end of the offensive strategy definition, each player has produced three swords, each one described by a value (the force points).

The second phase consists of allocating an amount of defensive points into three shields, through an interface displayed on the smartphone.



(a) (b)

Figure 2. Initial state of the game (a) and an example transformation (b)

The choice should be based on a couple of considerations: how the player guesses the opponent distributed force points on the different attacks (i.e. swords) and transformations chosen for building her own three swords. For instance, if a player managed to easily compose a powerful strategy for the bronze sword, then he might consider allocating most of his defensive points into the bronze shield, supposing that his opponent composed a similar strategy too. This mechanism is meant to foster analytical skills and the design-debug stages of CT.

In the third phase, the players will have to wear their Google cardboards (low-cost VR system viewer that makes use of a smartphone) and enjoy the battle in Virtual Reality (Figure 1-b). The battle is divided into three rounds, in which a simulated fight against two characters, representing the players, is shown. The player who wins at least two rounds wins the battle. Every round is characterized by one of the three metals (bronze, iron and steel). Each sword displays two values: the type of metal and the force points, while the shields display the defensive points. The winner of the round is the player who scored the most of force and defensive points combined. By watching and analysing the outcome of the battle the player will follow the design-debug-run stages introduced by the CT definitions mentioned above.

3 Conclusion and Future Work

TAPASPlay combines Game-Based Learning with Tangible User Interfaces and Virtual Reality to foster Computational Thinking skills. By playing the game the players will learn what worked well and what were the weakness of their respective strategies and therefore learn about their problem solving abilities obtaining feedback on their progress on Computational Thinking skills: the design-debug-run stages have been identified as a common feature of the main Computational Thinking definitions. In future we plan to explore different aspects of Computational Thinking by designing activities to help users progress on each specific CT skill considered.

References

1. Brennan, K., Resnick, M.: New Frameworks for Studying and Assessing the Development of Computational Thinking. In: Proc. 2012 annual meeting of the AERA (2012).
2. Kazimoglu, C., Kiernan, M., Bacon, L., MacKinnon, L. Learning Programming at the Computational Thinking Level via Digital Game-Play. Procedia Computer Science, 9, 522-531 (2012)
3. Kafai, Y. B.: From computational thinking to computational participation in K--12 education. Communications of the ACM, 59, 8 (July 2016), 26-27 (2016)
4. Lieberman, H., Paternò, F., Wulf, V. (Eds.): End-User Development. Kluwer Academic Publishers, Dordrecht, The Netherlands (2006)
5. Turchi, T., Malizia, A. Fostering computational thinking skills with a tangible blocks programming environment. In: Proc. VL/HCC 2016, 232-233 (2016)
6. Weintrop, D., Holbert, N. R., Horn, M. S., Wilensky. U.: Computational Thinking in Constructionist Video Games. International Journal of Game-Based Learning, 6, 1, 1-17 (2016)
7. Wing, J. M.: Computational thinking. Communications of the ACM, 49, 2, 33-35 (2006)

An End-User Tool for Creating Augmented Reality Experiences

Álvaro Montero, Telmo Zarraonandia, Paloma Díaz, Ignacio Aedo

Avenida de la Universidad 30, 28911
Universidad Carlos III de Madrid, Spain
{ammontes, tzarraon, pdp}@inf.uc3m.es, aedo@ia.uc3m.es

Abstract. This paper presents the architecture of an authoring tool for allowing non-experts creating realistic and interactive augmented reality (AR) experiences. The tool implements an approach for designing AR scenes that combines the use of AR model-based techniques with the possibilities that current game engines offer for designing and managing 3D virtual environments.

Keywords. Augmented Reality, End-User Development, Authoring Tool

1 Introduction

Augmented Reality (AR) allows to superimpose additional information to the user's vision of the real world [1], augmenting the physical space with images, texts, videos or 3D models, for example. The AR technology has already been successfully applied to several areas, such as medicine [2] and education [3]. However, its adoption is not as widespread as it could be expected if considering its potential benefits. This could be due the variety of knowledge and expertise required to implement this type of applications, which often include programming, image processing and 3D modeling skills. Moreover, most of the existing applications are specially designed to be used for a particular task and under certain conditions. Therefore, the opportunities to reuse them are quite limited.

In this working paper, we introduce an authoring tool that aims to provide non-expert users with the means to create realistic and interactive AR experiences by themselves. Following this objective, the tool combines model-based AR techniques [4] with the possibilities that current game engines offer for the design and management of 3D virtual environments.

The rest of the paper is organized as follows. In the next section we review some existing AR platforms. Next, we introduce the architecture of the proposed AR authoring tool. At the end of the paper, some conclusions and future lines of work are presented.

2 Related Work

One of the most popular tools for implementing AR applications is the ARToolkit library [5]. This software provides a set of functions and resources to support the finding and identifying of markers in video images, and the rendering of virtual images on top of them. The use of ARToolkit is restricted to users with advanced programming skills. Non-expert users can make use of other AR solutions as DART [6] or Powerspace [7]. These tools extend well-know software applications to enhance them with AR functionalities. For example, DART allows to design AR experiences by drag-and-dropping markers and images to augment in the Score Window of the Macromedia Director. In the case of Powerspace, the user describes the AR experience composition using the MS PowerPoint, and the program transforms it into a 3D AR experience. Following a different approach, the authors of [8] describe a platform that allows the user to modify the position and size of the AR objects directly with hands gestures. In any case, all these solutions require to prepare the space to augment previously, placing on it markers that serve as references. The Vuforia SDK [9] overcomes this limitation, as it is capable to recognize images in the video of the real world and place the AR objects in relation to them. This makes possible to directly augment pictures or objects in the real world. However, the level of realism achieved is still very low, as virtual images are presenting on top of the video and not integraed in the real world.

3 The Authoring Tool

This works aims to provide end users with an AR authoring tool for creating realistic and interactive AR experiences, as the ones described by Azuma in [1]. The tool seeks to support the augmentation of small spaces, for example a room, but also larger areas, such as a classroom or an auditorium. We aim to offer a simple approach for the definition of AR experiences that would not require programming skills, the previous preparation of the environment with markers, or the use of expensive specialized equipment that most non-technical users might lack.

3.1 Approach

Following these objectives, we propose to combine the use of AR model-based techniques with the possibilities offered by current game engines for designing and managing 3D virtual environments. More specifically, we propose to set up AR scenes by defining 3D virtual scenes that take place in virtual replicas of the space to augment. These replicas would be made up of 3D blocks or prisms that represent the floors, walls or any other background element of the environment. Once the scene is composed, the virtual replica will be aligned with the user' vision of the environment, and the background elements will be made invisible. As a result the virtual objects will appear to the user as integrated in the real world, smaller the further they are, or occluded by real objects when located behind the invisible background elements.

To evaluate the proposed approach we implemented an AR game that was played in an auditorium by more than 100 players simultaneously [10]. The game was played using an game platform named GREP (Game Rules scEnario Platform) [11], that provides a set of tools for creating serious games. For this experience, we extended the platform's player (GREP Player) to support playing games in AR scenes composed as described above. The AR scene of the game and the 3D model of the auditorium required for this experience were created manually using the Unity editor.

3.2 Architecture of the Authoring Tool

Currently, we are extending the GREP Platform to provide end-users with an authoring tool for creating AR experiences. Figure 1 depicts the architecture of the tool. As shown in the picture, the tool consists of three modules, the AR Modeler, the AR Scene Editor and the AR Interaction Editor, the last two developed as extensions of GREP editors.

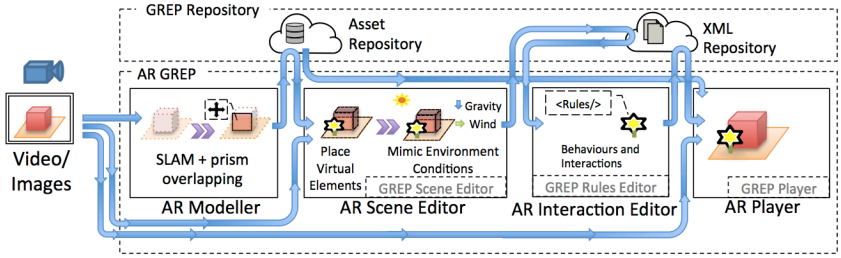


Fig. 1. Architecture of the authoring tool

The AR Modeller supports the user in the definition of the model of the environment to augment. As stated above, the model will be made of 3D blocks. Using as a reference an image of the environment projected at the back of a 3D virtual space, the user arranges the blocks until they overlap the walls and floors that they represent. In addition, the virtual space contains some 3D points that serve as guidance to place the blocks. These points are generated processing a video of the space to augment with an SLAM algorithm.

The AR Scene Editor allows the user to compose AR scenes. As in the previous case the user carries this action in a virtual 3D space containing an environment 3D model and a projection of an image of the space it represents. Using this image as a reference the user places 3D graphics from the assets repository on top of the environment model blocks (left hand side Figure 2). In addition, she can enhance the realism of the scene adding lights, wind forces or attaching audio clips to the virtual elements in it.

The AR Rules Editor allows to the user to animate the scene, specifying the rules that govern the interaction between the virtual objects and the viewers. This process is carried out attaching to the elements in the scene predefined behaviours of the GREP platform, describing movements and actions that can be activated as a response of events such as collisions, distance thresholds, user entries, etc.

The AR scenes can be exported as XML files that the AR Player process and interprets. The right hand side of Figure 2 depicts an example of an AR scene. In order to support the augmentation, the user's view should be aligned with the virtual camera.

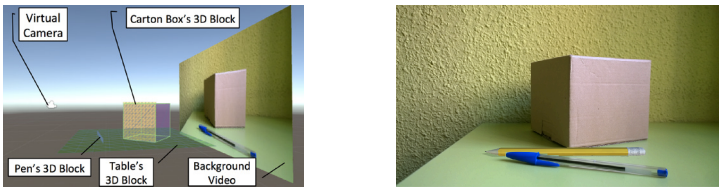


Fig. 2. Screenshots of the AR Scene editor (left) and the final AR scene (right): The pencil is a virtual object added to the real world.

4 Conclusions and Following Steps

This paper introduces the architecture an authoring tool for supporting non-experts users in the definition of realistic and interactive AR experiences. Once the implementation of the tool is finished, the next step will be to evaluate its performance in different areas. In addition, different solutions for keeping the virtual camera aligned with user's view when the camera of the device changes its position will be explored.

5 Acknowledgements

This work is supported by the project CREAx funded by the Spanish Ministry of Science and Innovation (TIN2014-56534-R)

6 References

1. Azuma, R.T.: A Survey of Augmented Reality. Presence Teleoperators Virtual Environ. 6, 355–385 (1997)

2. Marescaux, J., Rubino, F., Arenas, M., Mutter, D., Soler, L.: Augmented-reality–assisted laparoscopic adrenalectomy. Jama. 292, 2211–2215 (2004).

3. Wu, H.-K., Lee, S.W.-Y., Chang, H.-Y., Liang, J.-C.: Current status, opportunities and challenges of augmented reality in education. Comput. Educ. 62, 41–49 (2013).

4. Breen, D.E., Rose, E., Whitaker, R.T.: Interactive occlusion and collision of real and virtual objects in augmented reality. Munich Ger. Eur. Comput. Ind. Res. Cent. (1995).

5. ARToolKit Home Page, http://www.hitl.washington.edu/artoolkit/.

6. Macintyre, B., G, M., Dow, S., Bolter, J.D.: DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In: In ACM Symp. on UIST pp. 197–206 (2004).

7. Haringer, M., Regenbrecht, H.T.: A pragmatic approach to augmented reality authoring. In: Proc of the 1st Int Symp. on Mixed and Augmented Reality. IEEE Computer Society (2002)

8. Shim, J., Kong, M., Yang, Y., Seo, J., Han, T.-D.: Interactive features based augmented reality authoring tool. In: Consumer Electronics (ICCE), 2014 IEEE Int. Conf. IEEE (2014).

9. Vuforia | Augmented Reality, https://www.vuforia.com/.

10. Montero, Á., Zarraonandia, T., Díaz, P., Aedo, I.: Creating Interactive and Realistic Augmented Reality Experiences. In: Proc. of the XVII Int. Conf. on HCI. p. 17. ACM (2016).

11. Zarraonandia, T., Diaz, P., Aedo, I.: Using combinatorial creativity to support end-user design of digital games. Multimed. Tools Appl. 1–26 (2016).

Case Study of End User Development: The DiSSeCt Project

Tom Seymoens[1], An Jacobs[2]

[1]Vrije Universiteit Brussel, Brussels, Belgium
tom.seymoens@vub.be

[2]Vrije Universiteit Brussel, Brussels, Belgium
an.jacobs@vub.ac.be

Abstract. Technological opportunities such as web services, service-oriented computing and the semantic web make it increasingly possible to put in place systems that allow access to data sources and services at their time and place of need. Unfortunately, in order to reap the benefits, such as knowledge gathering, a high level of programming skills is still required. Starting from a brief definition of complex workflow composition and the issues of end user development, this work in progress presents the ongoing research in the interdisciplinary research project DiSSeCt that aims to empower end users in relation to their informational needs. We conclude this document with an overview of the next steps that need to be undertaken to enhance our understanding of what inclusive programming environments could entail.

Keywords: End User Development, Workflow Composition, Transformative User Experience, Integrated Care

1 Introduction

1.1 Service-Oriented Computing

In recent years, a number of technical advancements holds promising perspectives on how to unfold more flexible application environments for end-users. First of all, current technology is getting more and more effective in storing the ever-increasing volumes of data. This datafication also progressively allows the information to be searched, combined and analysed, although the data is often complex, stemming from a multitude of sources and distributed across a lot of (oftentimes siloed) repositories [1–3]. A second interesting development can be found in the instalment of web APIs, basically defined as reusable software components published on the web which provide access to the data and services [4]. A third and final development worth mentioning, is the shift towards service-oriented computing (SOC). Software systems have transgressed from inherently monolithic and centralized to dynamic and highly distributed [5]. This evolution leans on the foundation of the previous two, as it understands that

the services and data can be considered as "[…] autonomous, distributed, and platform-independent computational elements capable of solving specific tasks that can be assembled into loosely coupled networks to create dynamic applications and business processes" [5]. At the core of SOC lies the implementation of service-oriented architectures (SOA), an architectural model that allows applications to be built up out of a set of services, each with a self-contained unit of functionality. The added value of this approach is that by making an abstraction of where and how content and services are stored, they can be reused. This results in both lower development and maintenance costs [6].

1.2 Transformative User Experience

Taking this into account, it is not challenging to observe how these technological advancements contribute to closing the gap towards what Latzina and Beringer have branded the "Transformative User Experience" [7]. Central to their framework is the decoupling of content and capabilities from the container: the application, device or software stack. The application of this framework would eventually result in an "[…] opportunity to design natural, task-motivated new environments in which multiple capabilities and data coexist and can be put in contextual relationship at use time rather than at design time" [7].

However, the road towards a valuable and truly transformative user experience still contains a number of research challenges that need to be addressed. At their core these challenges translate to the suitable and balanced provision of methods, technologies and guidelines to end user developers so they can design, implement, test and maintain service compositions that fit their specific and context-dependable needs.

In the rest of this document, we discuss the ongoing work in the interdisciplinary research project DiSSeCt (Distributed Semantic Software Solutions for Complex Service Composition), funded by FWO-Vlaanderen. In the project, we aim to democratize the process of service composition and knowledge gathering.

2 The DiSSeCt Solution

DiSSeCt aims to provide solutions to the above-mentioned research challenges by designing distributed semantic software solutions and algorithms for continuous exchange of streams of data between the different stakeholders in an ecosystem [8]. Together with the set of underlying technological solutions, the goal is to develop an end user-friendly graphical user interface and guidelines that aspire a maximum adoption potential. Different research prototypes are being created that allow semantic service exposure, scalable processing of streaming data and automatic service composition and execution.

The different technological components collaborate to allow dynamic adaptive and automatic service compositions. Dynamic adaptation as a system characteristic allows the developer to easily deal with her/his changing objectives and services that become

unavailable or are updated. Moreover it grants continuous optimization during the execution of his/her system [5]. The automatization of this process contributes to the adaptability as the user does not have to manually select the required services and data.

The three main technological building blocks that the consortium is focusing on are [8]:

- Stream reasoning and real-time big data analytics, allowing data sharing and data analytics at the point and time of need. This would lead to relevant insights that otherwise might be overlooked by the stakeholders.
- Semi-automated semantic service exposure, enabling a tailored and personalized delivery of services, based on the spatio-temporal context. The service offerings and information exposure can then be exposed via end-user programming tools.
- Distributed and dynamic workflow composition, allowing the automatic suggestion of integrated workflows based on the service portfolio of the trusted actors in the ecosystem.

We believe that through the automatic service exposure and compositions, end users with a basic level of computational skills should be enabled, by provision of an inclusive and tailored development environment, to create their own context-specific solutions.

3 Work in Progress and Future Steps

3.1 Exploration of a Health Use Case for Service Composition

The demonstrators resulting from the project aim to support professionals in the health sector. Due to increasing number of patients and the limited human and financial resources, more and more attention is given to the proper use of technology and data to alleviate some of the strain that is put on the traditional healthcare environments. Among the many examples are the efficient use of different data sources (activity trackers, electronic health records, readings from hospital monitors, blood pressure monitors etc.) to provide more patient centric services and the application of robotics during surgery or in nursery homes. We cannot expect healthcare professionals to possess the programming skills that are currently required to deal with these technological advances and program the systems in a way that they deem useful and sufficient for an optimal care provision. By means of the DiSSeCt tools, we aim to put a system in place that can support healthcare professionals to maintain and adapt their workflows and manage their specific data and information needs in a much more end-user friendly way. Aiming for an ideal situation in which the user can focus on his information needs, without being burdened with the application of the best software engineering practices.

In the past, already some research has been performed around the topic of workflow composition in health care settings (e.g. [9–11]). We endeavour to build further upon these past contributions. A first step here is documenting an overview of the different

clinical pathways for a selection of conditions and consequently looking into the building blocks and particularities of the workflow, taking into account the different – technological and human – data sources. An evaluation of the proposed use case will be performed with our industrial stakeholders. As a next step, we envision extracting all the technological and social challenges in the specific use case and mapping them to the different technological components as suggested in Figure 1.

3.2 End-User Development Requirements

Simultaneously, a first elaborate literature study on the different approaches to end user development (EUD) has been performed. The objective is to determine which cues and interaction approaches can contribute to the delivery of more inclusive programming environments. It started with differentiating the existing paradigms such as visual programming, programming-by-specification and programming by demonstration and segregating the constituting elements. Subsequently we looked into the advantages and disadvantages and the preferred areas of use of each type. By defining each approach together with its main constituting components, we created a skeleton that can guide the analysis of different, already existing end-user programming tools in order to fully comprehend their application potential and possible hybridity. A round of interviews will be done with end user developers in the health care field to see what are their current intentions, struggles, needs, capabilities and required skills. This will help us understand the affordances of end user development environments.

Other challenges that are not in the immediate scope of the project but that are tightly connected are an exploration of the industries' motivation for semantically mapping and opening up their data.

4 References

1. Norberg, P.A., Horne, D.R., Horne, D.A.: The Privacy Paradox : Personal Information Disclosure Intentions vers us Behaviors. J. Consum. Aff. 41, 100–126 (2007).
2. Pötzsch, S.: Privacy Awareness: A Means to Solve the Privacy Paradox? In: Matyáš, V., Fischer-Hübner, S., Cvrček, D., and Švenda, P. (eds.) The Future of Identity in the Information Society. pp. 226–236. Springer, Berlin (2008).
3. Ardito, C., Costabile, M.F., Desolda, G., Latzina, M., Matera, M.: Making mashups actionable through elastic design principles. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 9083, 236–241 (2015).
4. Aghaee, S., Pautasso, C.: End-User Development of Mashups with NaturalMash. J. Vis. Lang. Comput. 25, 414–432 (2014).
5. Geebelen, K.: Adaptive Workflow Composition in Service-Based Systems. (2012).

6. Lizcano, D., López, G., Soriano, J., Lloret, J.: Implementation of end-user development success factors in mashup development environments. Comput. Stand. Interfaces. 47, 1–18 (2016).
7. Latzina, M., Beringer, J.: Transformative User Experience: Beyond Packaged Design. Interactions. XIX.2, 30–33 (2012).
8. Ongenae, F., Jacobs, A., Verborgh, R., Preuveneers, D.: DiSSeCt - Distributed Semantic Software Solutions for Complex Service Composition Project Proposal, (2015).
9. Schweitzer, M., Lasierra, N., Oberbichler, S., Toma, I., Fensel, A., Hoerbst, A.: Structuring clinical workflows for diabetes care: an overview of the OntoHealth approach. Appl. Clin. Inform. 5, 512–526 (2014).
10. Abidi, S.S., Helen Chen: Adaptable Personalized Care Planning via a Semantic Web Framework. 20th Int. Congr. Eur. Fed. Med. Informatics. (2006).
11. Liu, C., Xiong, H., Member, S., Papadimitriou, S., Ge, Y., Xiao, K.: A Proactive Workflow Model for Healthcare Operation and Management. 29, 586–598 (2017).

Exploring the use of Augmented-Reality to support end users in physical computing tasks

Alberto Ruiz, Andrea Bellucci, Paloma Díaz and Ignacio Aedo

Dept. of Computer Science, Universidad Carlos III de Madrid, Leganés, Spain
{alberto.ruiz,andrea.bellucci,paloma.diaz,ignacio.aedo}@uc3m.es

Abstract Building an electronic circuit is an error-prone activity for end users who are engaged in physical computing. We present results from an empirical study that investigates the use of Augmented-Reality (AR) to mitigate circuit construction issues. We observed end users as they build a circuit using LEDs, switches, sensors and an Arduino microcontroller, with the help of printed circuit diagrams or by augmenting the circuit components through an AR tool. AR showed potential to (i) reduce the number of fatal errors by fostering a more systematic approach to circuit building and (ii) increase self-confidence of novice users. Our work is a first step towards understanding how AR could be used to support end users to construct an electronic circuit.

Keywords: Physical computing; Augmented Reality; End-user development.

1 Introduction

An increasing number of end users such as makers, interaction designers and human-computer interaction researchers is engaging in physical computing as part of their personal or professional endeavors, thanks to the widespread availability of open platforms (e.g., Arduino), easy to use electronic components and toolkits [3], do-it-yourself online communities (e.g., Instructables.com) and makerspaces. Physical computing is, however, intrinsically complex. It requires hardware as well as software knowledge; that is, end-user developers need to know different sensors, actuators and electronic components, how to wire components in a circuit and how to program microcontrollers. While research has aimed to understand software-related issues and build tools for making the development of interactive artifacts easier, how to support end users when constructing a circuit is still underinvestigated [1].

Our ongoing research is aimed to understand the potential of Augmented-Reality (AR) to support physical computing tasks; by offering additional layers of digital information to circuit components, AR could help inexperienced users overcome the problems related to circuit construction, such as incorrect wirings, which have been identified as a major source of fatal errors [1]. We present preliminary results from an empirical study in which we observed participants as building relatively simple circuits using LEDs, switches and temperature sensors connected to an Arduino microcontroller. In their tasks, participants could use printed circuit diagrams or an AR tool that

augmented the circuit with digital representations of the components and offered additional information. Our results provide empirical evidence that the use of AR (i) reduces the number of fatal errors by fostering a more systematic approach to circuit building and (ii) increases self-confidence of novice users.

2 Background

There are in the literature many examples of systems that aim to support the user in the construction of physical interactive systems. Examples ranges from ready-to-use libraries of physical components such as Phidgets [3], to visual environments that ease the development of sensor-based interaction such as d.tools [4], to digital circuit design tools such as Fritzing [5]. However, very few support systems have been developed to help end users as they construct a circuit [1]. A recent example is Toastboard [2] that physically augments a breadboard with external components to provide visual feedback that would help novice users in circuit debugging.

3 Study Design

We run a within-subject study (think-aloud protocol), with participants building two different circuits, one time with the support of printed visual circuit diagrams generated with Fritzing, and the other time using our AR tool. Conditions were counterbalanced to mitigate learning effects. We recruited 12 participants (8 males, 4 females, max age 43, min age 22, M=31, SD=6.87). Participants completed two 7-items Likert scale questionnaires to gathered information about their demographics, programming and electronic expertise, and self-efficacy in physical computing. All the participants had advanced programming knowledge (M=6, SD=1.15) and basic knowledge of electronics (only 4 participants had notions about resistors color code). At the beginning of the experiment, participants were introduced to the AR tool, printed circuit diagrams and the tasks. They were given 20 minutes to complete each task. At the end of each task, participants were asked to fill in a questionnaire on the perceived usefulness and ease of use of the used tool (printed diagrams or AR application); the questionnaire used root constructs from UTAUT [6].

Both tasks were carried out using an Arduino UNO microcontroller and a common breadboard. The two tasks were taken from official Arduino tutorials[1]. The first task required to build a 'SpaceshipInterface' device, a control panel for a space ship that uses LEDs and switch button. In the second task, participants had to build a ‘LoveOMeter’ device, which uses three LEDs to visualize the values from a temperature sensor. Both devices had the same building complexity and they required the same number of electronic components. Since we were focusing on circuit construction, the program logic for each task was previously loaded on the Arduino microcontroller in order to avoid software-related issues [1].

[1] https://www.arduino.cc/en/Main/ArduinoStarterKit

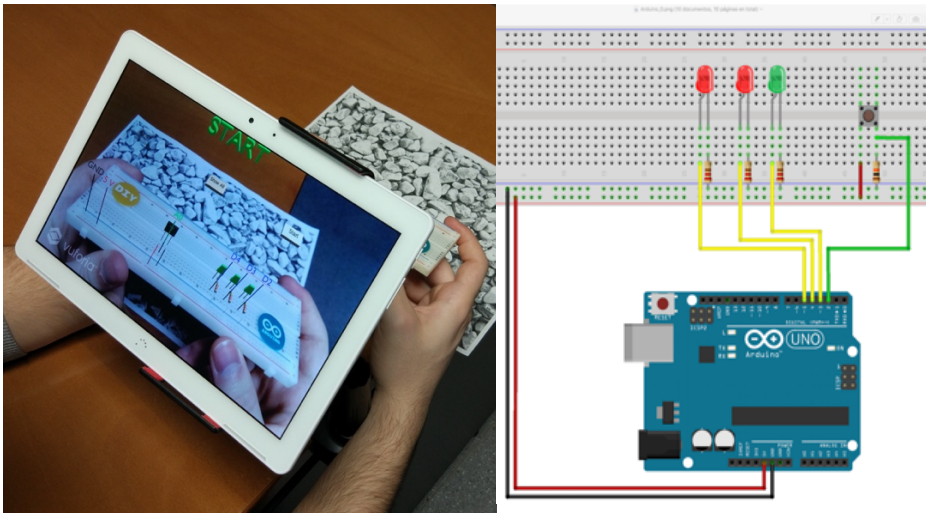


Fig. 1. (left) A screenshot of the AR application to augment circuit components. (right) The printed circuit diagram for the "SpaceshipInterface" task.

Apparatus

In one condition, participants were given an AR tool we developed using the Vuforia SDK for Unity[2]. The tool augments Arduino microcontrollers and breadboards with a digital representation of components and circuit connections according to the specific circuit for the task. Additional information on the components was also provided: e.g., LED polarity or value of resistors according to its color code. The tool implements both a step-by-step procedure that guides the user in wiring the components one by one, and a global view of the final circuit. The interface provides three virtual buttons: one to go back to the previous step, one to advance to the next step and one to display all the augmented components at once. The application was installed on a BQ Aquaris M10 tablet that was attached to a stand holder to make it easier for the participants to augment the circuit without having to hold the tablet (Fig. 1, left). In the other condition, a series of circuit diagrams generated with Fritzing was given to the participants, which they can consult to build the electric circuit (Fig. 1, right). Again, the printed circuit diagrams provided step-by-step instructions as well as the full circuit. During the experiment, participants were left free to follow the strategy of their choice to complete the task: either follow the steps or use the global view. Participants were also able to check on the Internet for additional information.

4 Results

Participants needed more time to complete the task using the AR tool (AR tool: M=15'40'', SD=3'16''; printed diagrams: M=8'29'', SD=3'59''). Participants made

[2] https://developer.vuforia.com

less errors when they used the AR tool and had less unresolved errors —the errors they were not able to solve by themselves— (AR tool: total errors=18, solutions=13; printed diagrams: total errors=25, solutions=16). A two-sample Mann-Whitney U test shows that the perceived ease of use of the AR tool (M=4.04, SD=1.01) is significantly different from the printed circuit diagrams (M=3.19, SD=1.30).

5 Discussion and Conclusion

Insights from the experiment reveal two themes that show potential to integrate AR tools in workbenches for physical computing to support end users when constructing a circuit:

- **Ease of troubleshooting.** Using the AR tool participants were more methodical in circuit construction and they were able to locate, understand and recover from errors more easily. At any time, participants were able to verify they were using the correct component by matching the physical component with its digital representation superposed to the real circuit. Moreover, the additional information for resistors, led and sensors guided participant to correctly connect each component. When using the printed diagrams, participants did not have easy access to important information that would have help them to correctly connect the components (e.g., LED polarity).
- **Increased self-confidence.** The AR tool encouraged participants to learn by exploring and physically manipulate the components, which led them to feel more confident that the steps they took were correct. With the AR tool, participants felt they better understood what they were doing (e.g., where and how each component was placed). By contrast, using the circuit diagrams led to a more mechanical approach: participants did not try to understand what they were doing but they only followed the instruction to complete the task. Therefore, when they made a mistake, it was more difficult for them to recover.

References

1. Booth, T., Stumpf, S., Bird, J., & Jones, S. 2016. Crossed wires: Investigating the problems of end-user developers in a physical computing task. In *Proc. of CHI* (pp. 3485-3497). ACM.
2. Drew, D., Newcomb, J. L., McGrath, W., Maksimovic, F., Mellis, D., & Hartmann, B. 2016. The Toastboard: Ubiquitous Instrumentation and Automated Checking of Breadboarded Circuits. In *Proc. of UIST* (pp. 677-686). ACM.
3. Greenberg, S., & Fitchett, C. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proc. of UIST* (pp. 209-218). ACM.
4. Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., & Gee, J. (2006, October). Reflective physical prototyping through integrated design, test, and analysis. In *Proc. of UIST* (pp. 299-308). ACM.
5. Knörig, A., Wettach, R., & Cohen, J. 2009. Fritzing: a tool for advancing electronic prototyping for designers. In *Proc. TEI* (pp. 351-358). ACM.
6. Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly*, 425-478.

Community end-user development: Patterns, platforms, possibilities and problems

Susanne Bødker[1] & Peter Lyle[2]

[1.] Aarhus University, Denmark. bodker@cs.au.dk
[2.] Madeira Interactive Technologies Institute, peter.lyle@gmail.com

Abstract. Communities often struggle to find, use and maintain technological support for their collaborative activities. The choices that communities make regarding which services and platforms to use among members are made more complex as each technology in question offers certain possibilities of end-user development, yet they equally demand certain skills and resources to maintain. This short paper uses examples of such community technologies to discuss how to analyze and make use of these constraints and possibilities, when communities are in need of new tools. We propose patterns as appropriate abstractions to analyze these constraints and possibilities and analyze possibilities and problems across cases. Despite their specificities, communities share patterns that can usefully be exposed to inform their technological choice and development processes so as to make better informed choices when it comes to possibilities for end-user development.

Keywords. Community platforms, patterns, end-user development

1 Communities, patterns and end-user development

Bottom-up communities, like the three studied here, struggle to find, use and maintain technological support for their collaboration and physical activities, such as gardening or food distribution. Communities often seem to make do with 'second best' when it comes to technologies, simply because the better solutions are beyond their reach when it comes to customization, integration and extension of their technologies (due to lack of knowledge, resources or manpower). They also invest considerable effort only to see their technologies become obsolete e.g. when key members leave the community or lose interest. These three communities have grown from the bottom-up to manage some form of activity while partly organizing themselves and connecting to (new) members virtually. Though rather different, our example communities expose interesting tensions between standard and tailored solutions where end-user development happens. In this they share patterns that we find could usefully be exposed to help communities inform their technological choice and development processes.

Where end-user development is seen as *"a set of methods, techniques and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact"* [10] the current paper looks at communities where some interested community members who

also have professional software developer skills. In addition, we look beyond one artifact at a time since these communities often juggle several technological platforms [2, 3]. The communities we have studied 'live' for other purposes, but they organize, plan, coordinate, and inform about their activities through various online platforms.

Fischer [7] discusses cultures and ecologies of participation in relation to end-user development and distinguishes between consumers, contributors, collaborators and meta-designers as roles of increasing technical roles in end-user development. Mørch, similarly talk about customization, integration and extension to understand how end-user development happens and is supported [13]. Kaptelinin and Bannon [9] makes a more outside-inside distinction when they separate extrinsic practice transformation as carried out by designers from intrinsic practice transformation, initiated by users, end-user development. Intrinsic practice transformation, resembling what [7] has done by consumers, contributors, collaborators, is continuous, directly related to the practices and activities at hand, and result in idiosyncratic designs, possibly spanning multiple new and existing technologies. Interesting to our analysis, it is driven by a need or imbalance between what users are able to do with their given setup, and what they need or want to. We see a tension between the standardized and the situated in our cases, in a similar manner - if more haphazardously [2] - to organizational tailoring and sharing [1, 14]. In this space of technology choice, design and programming that we find *new challenges to end-user development.*

Many authors have discussed the possible roles of patterns in relation to collaborative technologies, and to some extent in relation to end-user development. Herrmann et al. [8] point out that patterns "*not only refer to technical features but also to the interplay between the technical system, the users, their tasks and organizational constellations*". Inspired by these and other discussions we see patterns as intermediate-level abstractions, that describe possibilities as well as problems or bottlenecks of the sociotechnical situation [4]. We want to explore this perspective in settings driven by end-users, such as our volunteer communities. The potential usefulness in addressing our findings as patterns lies in their interim level of abstraction across the research process with the end-user developers that we have met, and also in the possibly generative nature of the patterns, for the future choice of technological platforms in such communities [5].

In the following, we summarize and discuss patterns of end-user development. We adapt Mørch's customization, integration and extension across platforms to understand how end-user development happens and is supported [13]. Across the different platforms used by communities, we see a number of different patterns of use and end-user development, primarily customization. We also see a number of situations where communities reach out to people with more profound IT skills to have special solutions (typically custom-made webpages set up for them). Addressing end-user development at an interim level of abstraction helps point specifically to both technological possibilities and problems, and to user needs across the cases.

2 Volunteer-based communities - 3 cases

In the following we briefly present the communities and their technologies, the studies we conducted and point to literature for further documentation where this is avail-

able. We give an example of the technologies they deploy, the collaborative end-user development. We present examples from each of the communities (focusing on Facebook and mailing lists) following a systematic pattern, indicating: the platform, the problem related to its use (P), how it helps or hinders the community (H), how it was tailored or customised (EUD), and who was responsible or involved in the tailoring or customisation process (W), see table 1.

2.1 LOFC

This study involves a local organic food community (LOFC) in a major Danish city. Members can order a bag of vegetables and eggs via the community who is in touch with local farmers [2, 3]. The 900 members are mostly people in their 20's to 40's. [3] describes the early technology tailoring and development of the community as follows: The two founders "*started a local initiative based on the model from Copenhagen* [with] *a community wiki as the primary organizational platform*." They created a Facebook page, which "*proved to be a very efficient way of triggering interest in the ideas* [...] *Facebook played a vital role in the initial formation of the community.*" Various attempts were made to handle email distribution lists, within or outside of the community webpage, which has been under constant development throughout the lifespan of the community, in the hands of several different volunteer web developers with numerous platform problems in tow. Here we focus on their use of Facebook and a mailing list.

2.2 City farm

This study involved a city farm located in a major Australian city. Consisting primarily of volunteers, and supported by occasional community support grants and initiatives, as well as through the sale of grown food, a farmers' market, and education products, the farm serves as a local example of agricultural activity that can be done following permaculture design principles. [12] describes the city farm, their use of multiple, separately managed Facebook pages, as well as a website that is maintained by a web manager. The community had struggled to balance communication between physical and digital presence, and different groups within the community created their own Facebook pages for different purposes. Here we focus on this use of Facebook.

2.3 Permablitz, an Event-centric community

This study concerned the local chapter of 'Permablitz' in a major Australian city: A volunteer movement that seeks to encourage local food growing by running tailored events that typically involve an overhaul of a residential backyard. This local community involves a number of key volunteers, and a large number of volunteers at each event, often people from the local neighbourhood. [11] explain the website and Facebook presence of the community, from the perspective of the maintenance resources they require. The community created a twitter account to reserve the name, but does not use it because of the effort required. [11] notes the use of a mailing list by mem-

bers of a Permablitz group, specifically as a means for designers to interact, and for designers to be paired up with potential hosts of events. There is also a permablitz newsletter sent to anyone who wants to subscribe. Here we focus on their use of a mailing list.

Table 1. Two patterns across three cases

Facebook	
LOFC *P* - Create visibility and recruitment. Initially for all member communication (later separate emails and newsletters to members were used) *H* - Easy 'presence' online. Need for more targeted information, and separate internal communication. *EUD* - Customized look and feel to match organic food etc. *W* - Set up by the founders for their own use and to reach out to possible members and collaborators as consumers in Fischer's [7] terms.	**City Farm** *P* - Public facing facebook pages with events and contact details of the community. *H* - Easy 'presence' online. The ease of setup allowed for different parts of the city farm to create specialised pages independently, creating a breakdown in their otherwise centralised strategy for web management. *EUD*- Multiple pages created and customized for different gardens/sections of the farm. *W* - Each page was created and managed separately by members from different areas of the farm.
Mailing Lists	
LOFC *P* - Provide a newsletter for internal communication and contact. *H* - Provided an alternative means of contact that avoided forcing members to use facebook. Later a version of the website served as a substitute to the mailing list [4]. *EUD* - Introduced as a supplementary system, based on a MailChimp service. *W* - A subset of members who form the Communications group contributes.	**Permablitz** *P* - Discussion amongst permaculture designers, and an external list used to notify subscribers of upcoming events. *H* - [18] notes that the mailing list among the internal designers group was too broad and did not foster discussion. *EUD* - A simple distinction that the mailing list for permaculture designers was intended for discussion, while the public mailing list was for broadcast newsletters. *W* - Key members of the community maintain both lists.

3 Patterns of interim abstractions

From the varied platform use, we identified three interim abstraction constructs: Information, Notification and Discussion, as important to the patterns of use:

Information is focused on the problem of conveying the 'state' of the community or greater world context. Facebook pages are helpful in spreading information publically to self-acclaimed audience. Facebook groups work similarly for members. Mailing lists are for more target information to members (but see also notification).

Notifications address the problem of delivering activity information over time, and tend to involve the system actively interacting with the members. Facebook pages and groups lend themselves to self-managed notifications. Mailing lists, used for announcements, serves as a notification system, often duplicating information posted on other platforms such as Facebook and websites.

Discussion is concerned with how and whether community members interact with each other. Facebook pages and groups lend themselves to self-managed discussions

among members who choose to partake. Mailing lists use is varied as they are used in direct discussion.

In our examples the communities have combined needs that also drive decisions for choosing various platforms. In the following, we compare and discuss how use varies across the communities, focusing on the example of mailing lists where usage is divided roughly into two types: *Discussion* and *Information*. The distinction regards 'who' posts content, as well as the regularity/consistency of when the content is posted. In the case of announcements, the use of a mailing list is typically controlled, with few users able to create messages, and the messages tend to follow similar formats, often with similar information on upcoming events, or general community information. By contrast, discussions, which may include event or community information, allow for interaction back and forth among community members.

Regarding the actual observed use of mailing lists, we see a mix of more structured community announcement use, contrasted with community discussion use. [2] notes that the LOFC use of a mailing list is primarily serving to '*substitute Facebook for internal communication and contact*.' [11] notes the use of a mailing list by members of a Permablitz group. Mailing lists have members, which give them permissions for posting. The difference is if membership is self-managed, or centrally managed with a gatekeeper. In the LOFC case, the self-management is important enough that the community chooses Facebook over more stringent forms of distribution lists, that would also better support archiving. However, Facebook is notorious for not supporting systematic retrieving of previous posts.

4 Patterns of intrinsic end-user development?

We have shown how communities chooses, develops and sometimes abandon different platforms as part of their daily practice. We found different patterns of use and end-user development, primarily customization. In a number of situations communities reach out to people with more profound IT skills to have special solutions (typically custom-made webpages set up for them). The interim level patterns point specifically to both technological possibilities and problems, and to user needs across the cases. In almost all cases, intrinsic transformation is mainly about curation and customization, whereas we have found few examples of more extensive tailoring. With this in mind we ask for possibilities of moving beyond customization without going all the way to (external) custom built platforms?

Acknowledgments

This work was funded by Aarhus University under the AU Interdisciplinary center, PIT. We thank all our collaborators on the various cases mentioned.

References

1. Bannon, L.J.; Bødker, S.: Constructing common information spaces, Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work, , September 07-11, Lancaster, UK, pp. 81-96 (1997)
2. Bødker, S.; Korsgaard, H.; Lyle,P.; Saad-Sulonen, J. Happenstance, Strategies and Tactics: Intrinsic Design in a Volunteer-based Community. In Proceedings of the 9th Nordic Conference on Human-Computer Interaction ACM (2016).
3. Bødker, S.; Korsgaard, H.; Saad-Sulonen, J.: 'A Farmer, a Place and at least 20 Members': The Development of Artifact Ecologies in Volunteer-based Communities. In Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16). ACM. 1142-1156 (2016).
4. Bødker, S.; Chrstiansen, E.; Thuring, M.: A Conceptual Toolbox for Designing CSCW Applications. COOP 1995 International Workshop on Design of Cooperative Systems, pp. 266-284 (1995).
5. Dearden, A.; Finlay, J.: Pattern languages in HCI: a critical review. Human computer interaction, 21 (1), 49-102 (2006).
6. Erickson. T. Lingua Francas for design: sacred places and pattern languages. In Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques (DIS '00), Daniel Boyarski and Wendy A. Kellogg (Eds.). ACM, New York, NY, USA, 357-368 (2000). DOI=http://dx.doi.org/10.1145/347642.347794
7. Fischer, G. End User Development and Meta-Design: Foundations for Cultures of Participation. JOEUC 22(1) (2010), 52-82.
8. Herrmann, T.; Hoffmann, M.; Jahnke, I.; Kienle, A.; Kunau, G.; Loser, KU; Menold, N.: Concepts for Usable Patterns of Groupware Applications. In: Mark Pendergast, Kjeld Schmidt, Carla Simone, Marilyn Tremaine (Eds.): Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work, pp. 349-358 (2003).
9. Kaptelinin, V.; Bannon, L. J.: Interaction design beyond the product: Creating technology-enhanced activity spaces. Human Computer Interaction 27, 3, 277-309 (2012).
10. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. End User Development. Volume 9 of the series Human-Computer Interaction Series pp. 1-8. (2006).
11. Lyle P.; Choi J.H.; Foth M.: Designing for grassroots food production: an event-based urban agriculture community. In Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design (OzCHI '14). ACM, New York, NY, USA, 362-365 (2014). DOI=http://dx.doi.org/10.1145/2686612.2686666
12. Lyle P.; Choi J.H.; Foth M.: HCI for City Farms: Design Challenges and Opportunities. In: Kotzé P., Marsden G., Lindgaard G., Wesson J., Winckler M. (eds) Human-Computer Interaction – INTERACT 2013. INTERACT 2013. Lecture Notes in Computer Science, vol 8120. Springer, Berlin, Heidelberg (2013).
13. Mørch, A.: Three levels of end-user tailoring: customization, integration, and extension. In Computers and design in context, Morten Kyng and Lars Mathiassen (Eds.). MIT Press, Cambridge, MA, USA 51-76 (1997).
14. Trigg, R.H.; Bødker, S.: From implementation to design: tailoring and the emergence of systematization in CSCW. In Proceedings of the 1994 ACM conference on Computer supported cooperative work (CSCW '94). ACM, New York, NY, USA, 45-54 (1994). DOI=http://dx.doi.org/10.1145/192844.192869

CCBL: A new language for End User Development in the Smart Homes

Lénaïc Terrier, Alexandre Demeure, Sybille Caffiau

Institute of Engineering Univ. Grenoble Alpes
{forename.name}@ univ-grenoble-alpes.fr

Abstract. We present Cascading Context Based Language (CCBL), a new programming language for the Smart Home. We build CCBL on the notion of context that express home actions according to the observed states. We describe how CCBL enables users to organize contexts in a concise and predictable way using three mechanisms: 1) The Cascade for specifying device states implicitly, 2) The priority list for ensuring that only one context can access a device at a time and 3) The Allen's interval algebra for enabling orchestration of contexts over time.

Keywords: End User Development, smart homes, CCBL, ECA.

1. Introduction

From a technical point of view, a smart home combines sensors (thermometers, motion sensors…), actuators (lights, sound systems and digital displays) and network-oriented services (synchronized calendar, weather forecast, TV program, email…). As Mennicken et al. [8] state the use of these computing technologies allows "a home either to increase the comfort of their inhabitants in things they already do or enables new functionalities". Usages of smart home aim at easing everyday life chores (automatic vacuum cleaner), at saving energy (heating system optimization, energy consumption tracker), at increasing security (alarm systems), at raising awareness (remote surveillance access) [2], [5]. One challenge is therefore to bridge the gap between technical possibilities and inhabitants' needs.

ECA (Event Condition Action) is the most popular language in use to program home. It allows to express home behavior by a set of rules that independently program actions according to captured events (if Event and Condition then Actions). However, inhabitants may have difficulties to program with ECA. Huang et al. [7] show that users build a wrong mental model for a certain type of ECA rule. Other problems arise when programs become more complex, either in terms of the number of events or conditions that can trigger ECA rules [9] or in terms of the number of rules itself [4].

We explore a new approach centered on the notion of context rather than events. We also explore how Allen's interval algebra [1] can help to structure those contexts in order to make the behavior of complex programs more predictable by end users.

2. Cascading Contexts Based Language (CCBL)

Cascading Context Based Language (CCBL) is a programming language dedicated to programming Smart Homes. In order to design CCBL, we consider general properties that are required by all programming languages. The language should have a **low threshold**: what is simple to mentally model (lit a light) should be easy and simple to express. Moreover, since end-user development addresses non-professional developers, the basics should be easy to learn [3]. On the other hand, simplicity should not mean less expressiveness, the language should have a **high ceiling**: complex tasks should be possible to express. More advanced users should be able to work on complex projects [10]. It is particularly relevant in home automation system where some users need challenges [5].

We centered CCBL on the notion of context rather than the notion of event as it is for ECA. We first make explicit our notion of context before presenting how contexts are ordered by priority so that only one context at a time has access to a device. In a third subsection, we detail how CCBL enables to organize contexts based on Allen's interval algebra.

2.1. Contexts in CCBL

In CCBL, a context can be active or inactive. A context defines a set of actions that the system applies when the system detects that it becomes active (e.g. "*Set volume of the music to off and set lamp A to off*").

There are two types of contexts: State contexts and Event contexts. Event contexts are semantically close to ECA rule: the trigger of an event activates the related event contexts. There is no duration associated with event contexts, once the event is triggered, actions are performed. State contexts are associated with a duration; they have a start and an end. State contexts are semantically close to rules of the type "When condition then actions" as Huang et al. [7] describe them. There are three ways to define a State context, depending on how users express the start and the end of the context:

- A Boolean expression: The context is active while the system evaluate this expression as true. It is inactive when the system evaluate this expression as false. For instance, "*System detects that Martin is at home*" is such an expression.
- A start and an end event: The context is active after that the system detects the start event. It stays active until the system detect the end event. For instance, the start event could be that "*System detects that the door has been opened*" and the end event could be that "*The alarm has been turned off*".
- A Boolean expression and optionally a start and/or an end event: The context is active after either the system detects the start event (if specified) or the system evaluates the Boolean expression as true (if there is no start event specified). The context stays active until either the system detects the end event (if specified) or the system evaluate the Boolean expression as false (in any case). For instance, the starting event could be "*System detects that Martin enters his home*", the Boolean expression could be "*System detects that Martin is on the phone*".

CCBL organizes context hierarchically: A state context may have sub-contexts. Each sub-context can only be active if the parent context is active. For instance, one can consider the sub-context "*System detects that Martin is on the phone*" that apply only when the context "*System detects that Martin is at home*" is active. By construction, CCBL imposes to have one root context, this root context aims to represent the "neutral state" of the smart Home (e.g., lights are off; doors are locked).

2.2. Priority, predictability and cascade

On a complex system with many contexts simultaneously running, it must remain simple to ascertain which context should prevail over the others when both specify actions to apply upon a same device. To do so, whether a context can act upon a device must be independent from the order of execution and of the succession of events. In order to achieve that we set up rules that put into order all the competing contexts.

Several contexts may set the state of a device but only one has access to the device at a time to prevent inconsistencies. Unlike what happen with ECA, it is not the last who speaks who get the access. In CCBL, contexts are **ordered** in a priority list. When several contexts try to modify a device, only the one that has the maximum index in the priority list can do it. If this context becomes inactive, then the next context in the priority list sets the state of the device. The same process happens when a new context that tries to set the device becomes active. The order function used in CCBL is as follow, C1 > C2 means that:

- Either C1 is a descendant context of C2.
- Or C1 and C2 have the same parent context; C1 is indexed after C2.
- Or C1 has an ancestor A1 and C2 have an ancestor A2 such as A1 and A2 have the same parent context, A1 is indexed after A2 in the list of sub-contexts.

Every context specifies either explicitly or implicitly the state of all devices and services. The implicit specification of the state of devices and services is what we call the **cascade mechanism.** We took the inspiration from the Cascading Style Sheet (CSS) language [6] that enables styling of HTML documents.

In order to illustrate the cascade mechanism, let us consider the CCBL program illustrated in Fig. 1. If we suppose that Martin is at home and that he is *on the* phone, then two contexts try to set the state of the lamp: the root context and the one that has the Boolean condition "*System detects that Martin is at home*"). The lamp lights in white because "System detects that Martin is at home" is a descendant context of the root context. If Martin leaves home while still *on the* phone, the lamp will be set to off, as the only remaining context that try to set it will be the root context.

We designed the cascade mechanism to make it easy for users to express what happens when the system exits a context. Devices can never be in an undefined state, at least the root context define what is their state. This way, users can focus on expressing the device states associated with context without having to bother with what happens when a context is over as it is when programming with ECA.

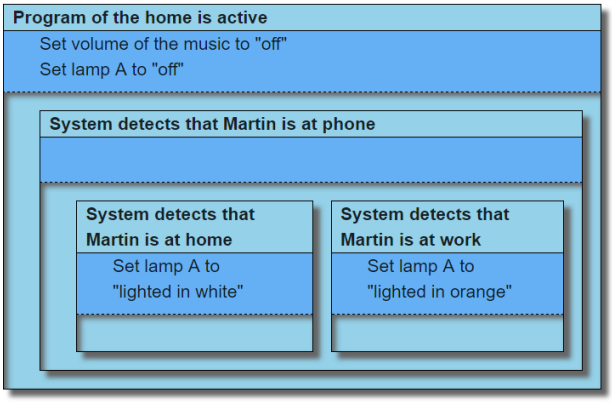


Fig. 1. Illustration of a CCBL program that control lamp A with respect to the fact that Martin is phoning or not and with respect to his location.

2.3. Contexts organization in CCBL

We designed CCBL in order to avoid the pitfalls identified for ECA, one of which is the accumulation of rules without hierarchy. CCBL provides several ways to organize the contexts based on Allen's interval Algebra [1]. In this section, we expose the different context's relationships that CCBL support. For each of them, we discuss the semantic implications in CCBL. We also illustrate how it can help end-users to express behaviors more easily.

SC during C. This relationship aims at supporting the expression of particular cases or exceptions. It expresses that context SC is considered only when context C is active. The semantic is as following: While context C is active, actions specified by C apply to the devices and services. When the context SC becomes also active, then the actions specified by SC override the ones of specified in context C. Vice versa, when the context SC becomes inactive, the actions it specifies do no more override the ones specified by C: The states of devices and services then rolls back to what is specified by C.

A possible scenario that illustrates the relevance of this relationship is the following one: "*When Martin is at home (context C), he wants the music to be played at a normal volume, except when he is talking on the phone (context SC). In this case, he wants the music to be played at a low volume*". When using the "during" relationship, users only have to specify the general context (C: "Martin is at home") that is associated with devices states (the volume of the music is low) and a particular context (SC: "Martin is talking *on the* phone"). CCBL automatically switches from the general context to the specific context, the users do not have to bother with specifying what happens when a context is entered or leaved, they only have to specify what happen when the system detects that a context is active. This way, we tackle the *non-sequitur* problem identified by [Huang 2015].

SC starts C. This relationship is a specialization "SC during C". CCBL considers the SC context only at the beginning of C. The starting event of SC is "when C starts". Therefore, SC can only happens once a time as long as C is active (and at its beginning). Users can define SC by a Boolean expression and/or an ending event.

A possible scenario that illustrates the relevance of this relationship is the following one: "*When Alice is at home (context C), the system plays music from her favorite radio except when she just arrives, then if she has got notifications, the system enumerates them vocally until there is no more notifications or she say stop (context SC)*".

SC finishes C. This relationship is a specialization "SC during C". CCBL consider the SC context only at the end of context C. As neither the duration of SC nor the duration of C can be known in advance, CCBL only allow to define SC using a start event. Obviously, the end event of SC is the end of context C.

A possible scenario that illustrates the relevance of this relationship is the following one: "*When Alice is at home during the morning of a working day (context C), listen to the radio news. If the system detects that she may be late at work (start of context SC), then the system plays a special music until she leaves her home*".

C2 takes place just after C1. This relationship expresses that the system considers C2 only just after C1 becomes unavailable. A possible scenario that illustrates the relevance of this relationship is the following one: "*When Alice is watching TV (global context), when she is on the phone (context C1), the TV is paused. As sometimes she walks through her apartment while talking, she wants that the TV to be kept on pause after the phone call ends while she is not sitting on her sofa (context C2)*".

C2 takes place after C1. This relationship expresses that the system considers C2 only after C1 becomes unavailable, but not necessarily just after. A possible scenario that illustrates the relevance of this relationship is the following one: "*After a soirée, Alice wants to be sure that her friends arrived at home (she always fear car crashes). She uses a lamp to be informed about the position of her friends' car. During the soirée, the lamp is colored in white (context C1). After the soirée, the lamp is lit in orange. When her friends are at home (context C2), the lamp is lit in green*".

Unsupported Allen's relationships. We do not support the relationship of overlapping (context C1 overlaps context C2) as it can only be detected a posteriori and not on the moment. We also do not explicitly support the relationship of equality (C1 happened exactly when C2 happens) as it can easily modeled by expressing a conjunction between the conditions of C1 and C2.

3. Implementation and Future works

We have developed a prototype of CCBL interpreter using the NodeJS[1] environment. This prototype is able to build a coherent system with devices and contexts and to maintain the system in a state depending on the contexts. In order to test the CCBL expressive power, we simulate a Smart Home environment. We have also implemented a simple user interface that allows the user to explore the devices and the contexts to see their status. The source code is available online[2]. In future works,

In future works, we plan to bridge this interpreter to a real home automation system such as OpenHAB[3]. We will also conduct experiment to test whether CCBL logic is more suitable than ECA for end users. Last, we will explore how to represents CCBL program to users in order to enforce their understanding of programs as well as to enable them to edit them efficiently.

4. References

1. James F Allen. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11):832–843, 1983.
2. AJ Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: challenges and opportunities. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 2115–2124. ACM, 2011.
3. Margaret M Burnett and Christopher Scaffidi. End-user development. Encyclopedia of HumanComputer Interaction, 2011.
4. Julio Cano, Gwenaël Delaval, Eric Rutten. Coordination of ECA rules by verification and control. 16th International Conference on Coordination Models and Languages, Jun 2014, Berlin, Germany. 16 p., 2014.
5. Alexandre Demeure, Sybille Caffiau, Elena Elias, and Camille Roux. Building and using home automation systems: a field study. In IS-EUD 2015, pages 125–140. Springer, 2015.
6. Håkon Wium Lie, Cascading Style Sheets, Thesis submitted for the degree of Doctor Philosophiœ, Faculty of Mathematics and Natural Sciences, University of Oslo, 2005.
7. Justin Huang and Maya Cakmak. Supporting mental model accuracy in trigger-action programming. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pages 215–225. ACM, 2015.
8. Sarah Mennicken, Jo Vermeulen, and Elaine M Huang. From today's augmented houses to tomorrow's smart homes: new directions for home automation research. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pages 105–115. ACM, 2014.
9. Chandrakana Nandi and Michael D. Ernst. 2016. Automatic Trigger Generation for Rule-based Smart Homes. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security (PLAS '16). ACM, New York, NY, USA, 97-102.
10. Fabio Paternò. End user development: Survey of an emerging field for empowering people. ISRN Software Engineering, 2013, 2013.

1 https://nodejs.org
2 https://gitlab.com/ccbl/ccbl-engine
3 https://www.openhab.org

Supporting Spreadsheet Maintenance with Dependency Notification

Sohon Roy, Felienne Hermans, and Arie van Deursen

Delft University of Technology, Delft, The Netherlands,
{S.Roy-1, F.F.J.Hermans, Arie.vanDeursen}@tudelft.nl

Abstract. Spreadsheets in the industry are used by multiple employees in organizations, and they remain in use for several years. Maintenance of existing spreadsheets is thus common. One of the issues in maintaining spreadsheets is the fact that formulas create cell dependencies, and these dependencies are invisible to users. To address this, dependence tracing techniques have been developed, both commercially and in research. However, these techniques are effort consuming, and are designed as separate activities that force the users to leave the context of editing spreadsheets. As such, these techniques are not suitably supportive for usual spreadsheet maintenance tasks. In this extended abstract, we present our work in progress on a novel approach for notifying users of cell dependencies, integrated into the context of editing spreadsheets. We present a preliminary evaluation of the approach in the form of an exploratory user-study with seven employees of a financial modeling company. Results show that the approach has the potential to support industrial spreadsheet users in the context of spreadsheet maintenance, as indicated by the responses of six out of seven participants.

Keywords: Spreadsheets, End-user Software Maintenance

1 Introduction

Spreadsheets are popular end-user computing tools, but their use is not without issues. Hermans *et al.* found that spreadsheets can have a long life span, and they are often used by several users in an organization [3]. This makes spreadsheet *maintenance* a difficult task, software maintenance alike. One reason hampering the maintenance of spreadsheets is the fact that the *cell dependency graph*—the network of cells which refer to each other—is not visible directly. Users who are not familiar with the dependency structure of a given spreadsheet might find it difficult to grasp the entire structure of the spreadsheet. This can even result in a fear of further modification of the spreadsheet [2].

The problem of analyzing dependencies within spreadsheets is by no means new. Both in research and in practice, tools have been developed to support spreadsheet users in the activity of *dependence tracing*. We refer to our previous work [7] for a study of such tools [4, 8, 1, 3]. However, one important issue with existing tools is that they aim to support dependence tracing as a separate

task; meaning that the user, while making changes to a formula, has to click buttons or use hot keys. As such, these tools support activities like auditing a spreadsheet, in which the users wants to deeply understand the dependency graph of a spreadsheet, but they do not support a user in making simple changes to formulas.

In summary, existing tools and techniques are not designed with the focus on supporting spreadsheet maintenance. Therefore, currently, the maintenance of *legacy* spreadsheets is not supported adequately, with users having to use dependence tracing tools which pose extra effort, and also force them to leave the context of spreadsheet editing. This latter deserves particular attention as context switching has been associated with cognitive difficulties faced during software maintenance [5, 9, 6].

The goal of this work therefore is to support users in editing formulas or values within a spreadsheet, whose underlying dependencies they might be unfamiliar with, without the need for them to leave the context of editing the spreadsheets. In this extended abstract, we introduce our tool *Aragorn*. Once enabled, Aragorn shows spreadsheet users the existence of dependents on every selected cell, ensuring the user is aware of them, and does not overlook making necessary changes to the dependents.

We conduct a preliminary evaluation of Aragorn, through an exploratory user-study with seven employees of a financial modeling company. Our findings show that Aragorn has a clear potential to help industrial spreadsheet users in the context of spreadsheet maintenance, by addressing the issue of hidden dependencies.

2 Background and Related Work

2.1 Cell Dependencies in Spreadsheets and Dependence Tracing

Formulas in spreadsheets can simply use values, like `=12+7`, but they may also refer to other cells within the spreadsheets. For this, spreadsheet users use cell references.

For example, if cell `A12` contains the formula `=B25+C25+K100`, then the value of `A12` depends on the values of `B25,C25` and `K100`; consequently `A12` is as a *dependent* of `B25,C25` and `K100`. Conversely, `B25,C25` and `K100` are the *precedents* of `A12`.

If a spreadsheet user wants to know the *precedents* of a cell, the user can simply read them from the formula. In the example `=B25+C25+K100`, the precedents are `B25,C25` and `K100`.

However, the converse is not true: when a spreadsheet user wants to know which are the *dependents* of a cell, i.e. which cells use its value, this is not possible without extra effort. The dependencies are hidden from spreadsheet users' (direct) view. To obtain an understanding of the dependencies in a spreadsheet, a user needs to use *dependence tracing* techniques. These are techniques for finding, visualizing, and navigating between dependents [7].

These techniques can be provided both as built-in features of spreadsheet applications, or as separate plug-in tools. For example, Microsoft Excel, has the *Audit toolbar* which provides an overlaid graph with the cells as nodes, and graph-edges shown with blue arrows depicting the cell dependencies.

2.2 Dependence Tracing Research

In addition to the built-in graph overlay feature of Excel, a number of techniques for dependence tracing have been proposed by researchers. For a summary, see our previous work [7]. While such proposed approaches all have their strengths and weaknesses, they share one important issue: they view dependence tracing as a separate activity. The spreadsheet user needs to navigate to a special menu created by the plugin [8, 1, 3], or enable a feature (Microsoft Excel, [4]) to start dependence tracing. While using such tools, users often find themselves spending dedicated time and effort solely for the purpose of understanding the dependencies inside a spreadsheet, making it an additional task on top of their normal work. This might be a reason why many of the dependence tracing techniques, have not found widespread adoption in industry yet [7].

The goal of this work, therefore, is *to develop a technique that is specifically designed to support finding dependents in the context of spreadsheet maintenance.*

3 The Aragorn Approach

Aragorn supports spreadsheet users in maintaining spreadsheets whose dependency structures might be (partly) unfamiliar to them. It is simple and entirely integrated into the spreadsheet environment, to support dependence tracing as effortlessly as possible.

Aragorn notifies the spreadsheet users of the existence of dependents for the currently selected cell, by showing the users a popup. Figure 1 shows the user interface of Aragorn, with three features. Before a user can use Aragorn, they need to process the workbook, during which Aragorn constructs the dependence graph and keeps it in memory. The other two options the user has are to turn 'Dependence Notification' on and off. Figure 1 furthermore shows the popup shown to the user, once the 'Dependence Notification' is enabled, upon clicking $C4$ in worksheet 'Finances'. Dependents on the same worksheet are shown without a prefix (K4, G5), while dependents outside the worksheet are shown with their worksheet name prefixed (for example Weekly!F6). Apart from enabling the feature once, the user does not have to perform any other action. Aragorn shows the popups conveying dependence information about every cell that is selected, as the user navigates from cell to cell inside the spreadsheet.

In the popup, the user finds relevant information about dependents of the selected cell. It firstly shows the total number of direct dependents of the selected cell, which helps the user to obtain an idea how critical the selected cell might be. Secondly, it shows the locations of all direct dependents, both those that are in the same worksheet and those that are in a different worksheet; it displays

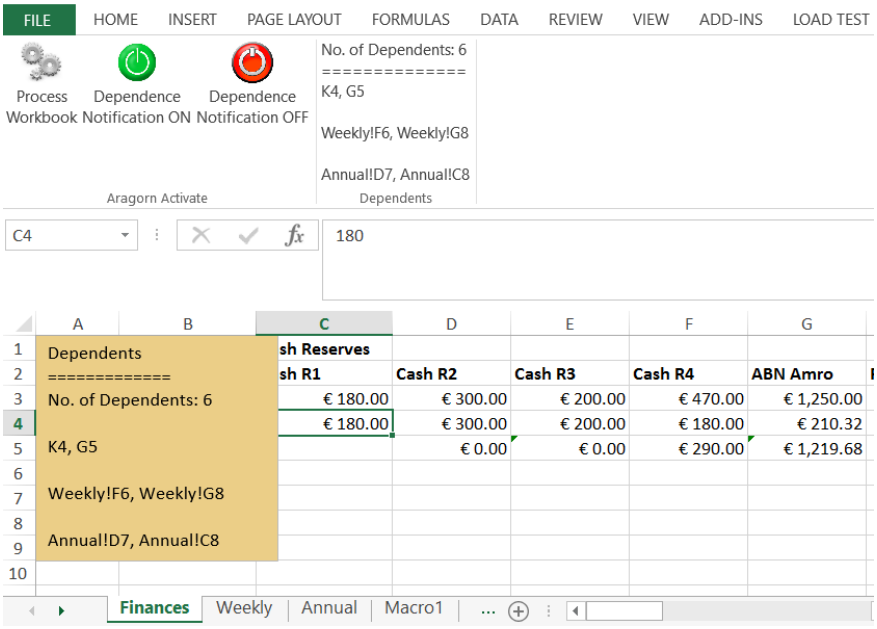


Fig. 1. The popup and the ribbon tab, showing dependents information when user clicks on cell C4 in the worksheet 'Finances'

the worksheet name and cell addresses. In addition to showing the dependence information in the popup, we also place it in the ribbon, as spreadsheet users are commonly used to obtaining information from the ribbon. Aragorn shows all of the above information for every cell the user selects, automatically. As such, the user does not have to perform any additional tasks, unlike other available dependence tracing techniques.

We have implemented the Aragorn approach in a .NET 4.5 based Microsoft Excel add-in written in C#. It is compatible with Office 2010 and 2013, and Windows 7 and 10. It uses the Infotron core engine to analyze the spreadsheets and obtain the list of dependents of all cells. The Infotron core engine is a commercialization of its predecessor Breviz— the spreadsheet analysis framework developed by Hermans *et al.* [3].

4 Preliminary Evaluation

4.1 Setup

In order to evaluate the usefulness of Aragorn, we conduct an exploratory user study with 7 professionals employed at F1F9— a financial modeling company based out of the UK. The main operations of F1F9 consists of analysis, development, auditing, and re-building of Excel based financial models.

We provide all the participants with the latest version of the Aragorn prototype, to install on their computers before the evaluation starts. They could then use the tool for a period of two weeks. After this period, we ask the participants to respond to a survey.

In the survey, we first provide the description of a scenario that sets up the spreadsheet maintenance context in which we intend to evaluate Aragorn. We subsequently ask six questions with one for identifying the participants, and the rest for evaluating Aragorn in the context of the described maintenance scenario.

4.2 Results

All seven of the participants agreed that they would be concerned in the described scenario, as the changes they make might inject errors into the spreadsheet, reaffirming the difficulty of spreadsheet modification related tasks.

On a five-point Likert scale, lack of dependency information was rated the second most important reason, slightly behind lack of familiarity about data, as probable reasons for the participants' concern about injecting errors. This result reaffirms that the problem of dependency is an important factor contributing to the difficulty of spreadsheet maintenance.

A total of six participants either agreed or strongly agreed that they would manually check dependencies for each cell they modified. This result points towards the lack of existing automated support for obtaining dependency related information during spreadsheet maintenance.

We asked if the participants could think of reasons for which they may prefer to skip checking of dependencies. This was an open question, and on analyzing the answers, we were able to identify five distinct groups of opinions the participants shared with us. We observe that two groups of opinions related to revelation of dependencies and time constraints, can be aimed to be addressed through supports like Aragorn, as Aragorn provides dependency information consuming minimal additional time. The other opinion groups are dependent on the type of maintenance or the nature of the spreadsheet to be modified, and as such are not possible to be addressed through support.

Finally, we asked the participants if they felt Aragorn will help making the task described in the scenario easier. Six of the participants agreed that Aragorn can make the task easier, and only one abstained.

From the results above, we can conclude that spreadsheet maintenance is difficult, and an important contributing factor to the difficulty is lack of dependency information. Normally, users are compelled to manually check for dependencies but dependency information being revealed to them or time constraints can be reasons due to which they may skip checking for dependencies. Addressing such reasons, Aragorn can help make the task of spreadsheet maintenance easier, as indicated by six out of the seven study participants.

In summary, we can state that based on responses from a set of seven industrial spreadsheet users, the overall preliminary evaluation of Aragorn, as an aid for spreadsheet maintenance that addresses the problem due to hidden dependencies, is promising.

5 Future Work

Our current work gives rise to several directions for future work. Firstly, there are some improvements to be made to the tool. For example, the performance of Aragorn could be improved to make it more feasible to use Aragorn in day to day spreadsheet use. Participants of our study also indicated a need for the ability to navigate to displayed dependents via hyperlinks. Secondly, we plan to perform a broader evaluation within a controlled setting, allowing for both qualitative as well as quantitative assessment of Aragorn, with the above improvements in place.

References

1. Ballinger, D., Biddle, R., Noble, J.: Spreadsheet visualisation to improve end-user understanding. In: Proceedings of the Asia-Pacific Symposium on Information Visualisation - Volume 24. pp. 99–109. APVis '03, Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2003), http://dl.acm.org/citation.cfm?id=857080.857093
2. Hermans, F.: Analyzing and Visualizing Spreadsheets. Ph.D. thesis, Delft University of Technology (2013)
3. Hermans, F., Pinzger, M., Van Deursen, A.: Supporting professional spreadsheet users by generating leveled dataflow diagrams. In: Proceedings of the 33rd International Conference on Software Engineering. pp. 451–460. ACM (2011)
4. Igarashi, T., Mackinlay, J.D., Chang, B.W., Zellweger, P.T.: Fluid visualization of spreadsheet structures. In: Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on. pp. 118–125. IEEE (1998)
5. Lethbridge, T.C., Pak, J.: Integrated personal work management in tksee software exploration tool. In: Proc. of the 2nd Int. Symp. on Constructing Software Engineering Tools (CoSET'2000) (2000)
6. Parnin, C., Gorg, C.: Building usage contexts during program comprehension. In: 14th IEEE International Conference on Program Comprehension (ICPC'06). pp. 13–22. IEEE (2006)
7. Roy, S., Hermans, F.: Dependence tracing techniques for spreadsheets: An investigation. In: Proceedings of the 1st Workshop on Software Engineering Methods in Spreadsheets. p. 12 (2014)
8. Shiozawa, H., Okada, K.i., Matsushita, Y.: 3d interactive visualization for inter-cell dependencies of spreadsheets. In: Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on. pp. 79–82. IEEE (1999)
9. Zayour, I., Lethbridge, T.C.: A cognitive and user centric based approach for reverse engineering tool design. In: Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research. p. 16. IBM Press (2000)

End-User Software Engineering of Service Mashups

Florian Wessling
(http://orcid.org/0000-0002-2307-2387)

paluno - The Ruhr Institute for Software Technology,
University of Duisburg-Essen
Schützenbahn 70, 45127 Essen, Germany
florian.wessling@paluno.uni-due.de

Abstract. With service integration platforms today it is easier than ever for non-programmers to create simple applications, automate tasks or control their smart-home appliances. Based on a workshop with multiple pair-programming teams that implemented a business use-case employing only service integration platforms, we show our first version of the software engineering process for service mashups, its key elements and their dependencies. Finally we motivate why further research is necessary on how integration platforms can foster innovation in a company by making their internal services EUD-ready.

Keywords: service mashups, integration platforms, engineering process

1 Introduction

Integration platforms (or short iPaaS: integration Platform as a Service[4]) such as IFTTT[6] and Zapier[11] enable users to select and configure so called applets or Zaps. These platforms do not require source code to be written which allows non-software-engineers to create applications very easily, thus enabling end-user development [7]. An applet (or Zap, recipe, etc.) serves as connector between two services. One represents the trigger of the connector and the other executes an action (hence the name *IFTTT: If This Then That*). By combining multiple connectors, users are able to create simple applications.

We observed that from this trend a software engineering process of service mashups is currently emerging organically due to the explorative nature of these platforms. Based on this observation we propose a structured yet agile and adaptive approach that is required in order to give end-user developers a guidance for creating service mashups and support the development of tools and methods in this area. Considering the advantages the software engineering process for traditional applications has on planning and management and with the idea in mind that service mashup prototypes might evolve into a permanent software product, it makes sense to examine the implications this new style of engineering has on the traditional software development phases.

2 Research Motivation and Related Work

In this section we will discuss the assumptions that led to our insights as well as the ongoing trends building the foundation for service mashup engineering.

Assumption 1: Software will be more and more assembled from existing elements instead of being build from scratch. To build upon the existing components of other manufacturers benefits a product because it is based on proven and thoroughly tested elements which are highly specialized for their usage scenario (cf. the UNIX toolchain principle). Current trends towards microservice architectures and containerization where containers (e.g. using Docker[8]) instead of Virtual Machines are used, underline the idea of the decreasing mashup building block size and a more fine-grained resource sharing that is achieved [10]. Component-based development does not only refer to software but can also be found in the hardware of embedded systems which are very often build in a *System-on-a-Chip* process [9]. A company licenses multiple established components as circuit blueprints which then are combined to manufacture a coherent silicon computer chip. Analogous to software systems, multiple pre-configured container images can be easily assembled into a full software stack.

Assumption 2: Internet connectivity will be ubiquitous and widely available in the future. Offline phases and latency will be reduced to a minimum. Mobile data networks and public WiFi hotspots will complement each other and will lead to more reliable and uninterrupted internet coverage [2]. This is required to build software that not only runs locally but is made up from distributed internet-connected services, leveraging the idea of cloud-computing.

Assumption 3: The amount of service offerings and APIs will massively increase, leaving the users with the challenge to select from a broad range of providers the one best matching their requirements. This observation refers to the decreasing size of components already mentioned and is further supported by the growing amount of sensors, actuators and data aggregators due to progress around the *Internet-of-Things (IoT)* and *Cyber-Physical Systems (CPS)*. From a service mashup point of view sensors and actuators can be encapsulated as web services from which sensor data can be retrieved and actuators can trigger physical actions by using an API. Due to the ease of which services can be used and contributed, many different private and professional providers offer own services leading to a broad, intransparent and constantly changing market that consists of both highly specialized and general services. Keeping an overview of the service offerings is challenging. Especially the discovery and evaluation of existing services is crucial for the development of service mashups.

Assumption 4: The amount of end-user developers with a private or professional interest in software engineering will increase. The Smart Home trend brings IoT devices into the hands of end-users and several vendors offer mobile apps

to customize their behaviour. iPaaS providers are integrating Smart Home services into their platforms. This enables users to develop basic applications which interact with their physical environment, i.e., using sensors as triggers (e.g. temperatur, movement) and operating actuators (e.g. shutters, lights, heaters) as a reaction. That software development is considered relevant is also backed by the idea to start teaching programming already in elementary schools [3, 1]. In the future we are therefore faced with users who are interested in programming but not necessarily have a formal education in this field (i.e. potential end-user developers). Developing service mashups with the given tools requires an adaptive process which we are about to outline in this paper and shape in future research. In contrast to the mashup engineering process that is currently emerging organically we see the need for a guidance and structure in order to create service mashups. Professional end-user developers can also benefit from an engineering process for service mashups as they are able to automate tedious tasks from their domain, create their own analyses or even exploring the feasibility of an idea very fast, thus realizing extreme rapid prototyping.

Assumption 5: The amount of data exchanged between systems via the internet will massively increase. Finding relevant information in this stream of data is an important challenge that needs to be supported. The Service Scout as envisioned in our approach is on the one hand responsible for finding suitable services that fit to the need of the business process and on the other hand necessary to find out about the data sources that are available in order to fulfill the user's goal. This means different levels of granularity are required for the data and it is necessary to be able to switch between those levels on behalf of the end-user and based on its knowledge and experience.

3 Idea and Approach

Building software as service mashups arose from a workshop we conducted with three pair-programming teams. The goal was to implement the process of evaluating a seminar without writing code by combining services which coordinate calendars, create questionnaires and send emails. After four 90 minutes iterations we consolidated our experiences, analyzed our actions and derived retrospectively our proposed software engineering process for service mashups (see Figure 1).

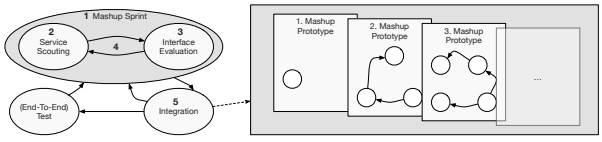


Fig. 1. Proposed Software Engineering Process for Service-Mashups

The implementation started with the (1) Mashup Sprint. Here (one or more) designated (2) Service Scouts use their experience and technical knowledge to find a suitable service for a feature (e.g. sending an email). During (3) Interface Evaluation the technical aspects of a service were explored. After evaluating the technical aspects of the service, (5) the existing service is connected to the service mashup (e.g. using a connector on an integration platform). After step (3) the given service might also be rejected due to incompatibilities or because it is not integrated in the required integration platform. In this case Service Scouting starts again (4). This way the service mashup is built in multiple Mashup Sprints and can be changed and improved over time.

4 Conclusion and Future Research

In our research we focus on the end-user developer being an employee of a company working in a department that does not develop software but is using it. Here we focus on the need to make use of the internal systems of the company and automating workflows that are usually carried out manually or combining internal and external services to foster innovation and prototype new ideas. Further research is required and will use the following questions as guideline:

1. What are the challenges when creating an integration platform inside a company with an existing microservice architecture?
2. How can the existing services inside a company be prepared for end-user development and connected to an integration platform?
3. What constraints apply when combining internal and external services on an integration platform?

As the next steps it is necessary to evaluate the expressivity of our process by implementing more complex business use-cases and it needs to be investigated if the developed architecture is restricted to the *pipes-and-filters* paradigm [5]. Moreover the proposed process and its key elements need to be validated in a real-world scenario, e.g., a company with multiple internal services that are currently restricted to the use of professional developers. We propose to create a custom integration platform inside this company and to connect internal EUD-enabled services to it as well as external public APIs.

After carrying out these experiments, the planned contributions of the doctoral work will be methods and best-practices for making webservices EUD-ready as well as a prototypical implementation of an integration platform enabled for end-user development.

For engineering service mashups in professional environments several open issues remain. Most integration platforms and services are lacking service level agreements (SLAs) which adds to the fact that engineering of service mashups is subject to a high degree of uncertainty. During development it is unclear how long an incorporated service will exist. Evolutionary prototyping of a mashup poses a business risk as it is questionable for how long integration platforms such as IFTTT and Zapier will continue their services.

References

[1] CS fundamentals for grades k-5, https://code.org/educate/curriculum/elementary-school
[2] Fettweis, G., Alamouti, S.: 5g: Personal mobile internet beyond what cellular did to telephony 52(2), 140–145
[3] Gardiner, B.: Adding coding to the curriculum https://www.nytimes.com/2014/03/24/world/europe/adding-coding-to-the-curriculum.html
[4] Gartner: Integration platform as a service (ipaas): It glossary, ttp://www.gartner.com/it-glossary/information-platform-as-a-service-ipaas/
[5] Hohpe, G.: Enterprise integration patterns : designing, building, and deploying messaging solutions. Addison-Wesley, Boston (2004)
[6] IFTTT: If this then that, https://ifttt.com
[7] Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-user development: An emerging paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, pp. 1–8. Springer Netherlands, http://dx.doi.org/10.1007/1-4020-5386-X_1
[8] Merkel, D.: Docker: Lightweight linux containers for consistent development and deployment. Linux J. 2014(239) (Mar 2014), http://dl.acm.org/citation.cfm?id=2600239.2600241
[9] Saleh, R., Wilton, S., Mirabbasi, S., Hu, A., Greenstreet, M., Lemieux, G., Pande, P.P., Grecu, C., Ivanov, A.: System-on-chip: Reuse and integration 94(6), 1050–1069
[10] Wolff, E.: Microservices: Flexible Software Architecture. Addison-Wesley Professional (2016)
[11] Zapier: Connect your apps and automate workflows, https://zapier.com

Designing Tangibles for Children's SEL in Conversations: Why and How

Mehdi Rizvi

Faculty of Computer Science, Free University of Bozen-Bolzano,
Piazza Domenicani 3, 39100 Bolzano, Italy, srizvi@unibz.it

Abstract. Social emotional learning also means teaching children positive social norms, which can be supported by interactive tangible solutions. Meta-design can help rapidly and effectively develop such solutions with users in the field. The design process of one such solution, TurnTalk for a norm for group conversations, is discussed in this paper.

Keywords: Socio-emotional Learning; Meta-Design; Tangibles

1 Introduction

Social Emotional Learning (SEL) is about teaching children positive social behaviours. The education literature suggests SEL roles & norms for conversations among children, e.g., [6]. A basic norm is *turn sharing* in group conversations: all group children have the right to take their turn in speaking; there should be no turn overlaps. Physical objects, such as sticks, are used to support the norm; but they are often forgotten by children and do not aid children in reflecting over their conversation. Research indicates that *interaction design* (ID) of tangible objects (briefly, *tangibles*) has the potential to enhance the scaffolding of SEL norms with children, e.g., [17]. My PhD work and this paper focus on ID of tangibles for SEL norms for conversations among children, such as turn sharing. Their design faces specific research questions. Section 2 overviews the main ones related to my PhD, and the end-user meta-design approach adopted for designing such tangibles. Section 3 zooms in on TurnTalk, a tangible for turn-sharing norm: design process of TurnTalk is presented as a proof-of-concept of the adopted design approach. Finally, Section 4 concludes the paper.

2 Research Work, Questions and Design Process

2.1 Related Research Work: a Compact Overview

Currently, no tangible for children has explicit SEL aims and hence tackles specific SEL norms [17]. Multitude of tangibles for conversation are for adults and work meetings. Although such context is crucially different than learning contexts for children, the interaction ideas or technologies can be ported to learning contexts for children. Examples are reported in [3, 5, 12, 2] etc. None of these

actively mediate a conversation but all enable participants to reflect on the conversation, mainly in terms of times taken in turns etc. Other tangibles instead aim at actively mediating the conversation in group. For instance, the tangible of [16] gives real-time textual advice for encouraging users to converse more by means of complex technology, which sometimes users found distracting.

There are also tangibles for learning contexts but none of them are meant for the scaffolding of social norms for conversation and children's self reflection on it. An example is *Ely the Explorer* [1]: it encourages children to exchange objects such as dolls and cards to promote interactions. In another study [13], only the teacher receives information about university students' collaboration.

2.2 Research Questions

The research goal of this PhD research is the design of tangibles for SEL norms for conversations in groups of children, by building on the research work in the literature for adults and ID for children (see above). Such tangibles can be used primarily in classrooms but also in other learning contexts, e.g., at home with parents. As educators are a vital part of SEL, this research work also considers them as (primary) users, besides children. The following questions guide the PhD research around the design of tangibles for SEL norms for conversations in groups of children: **(RQ1)** *How can we design tangibles for SEL norms that enhance all children's experience in group conversations?* **(RQ2)** *How can we design tangibles for SEL norms that enhance educators' experience?*

2.3 Design Process

The above questions stir the development of tangibles for children's SEL. To develop them, a meta-design approach, in the form of participatory design with action research, has been chosen as it allows continuous exploration and transformation of design and learning possibilities with users in the field [7].

The design process steps through rapid prototyped solutions acting as intermediary objects in the sense of [4]; solutions are assessed and used in field studies, which unveil novel design and learning possibilities. This requires the prototypes to be modular, adaptable and disposable, according to the results of studies with users. One such prototype, TurnTalk for primary-school children and their teachers, is discussed next as proof-of-concept. It is designed for the turn-sharing norm for conversations of groups of children (see Sect. 1).

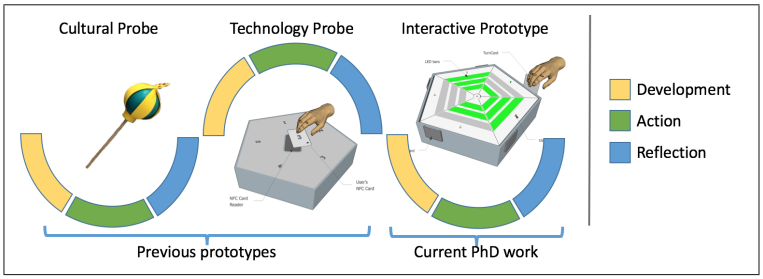
3 The TurnTalk Design Process

TurnTalk is primarily designed for the scaffolding and reflection over the SEL norm of turn sharing in group conversations, with 3 to 5 children.

The design approach for TurnTalk is as explained in Section 2.3: as indicated in the below figure, TurnTalk is built in cycles of rapid development sessions, actions in the field and reflections. Initially, TurnTalk was a scepter, acting as

cultural probe [8]. This was used in the field by primary classes and the design solution was abandoned. TurnTalk was then realised as a pentagon device and as a technology enhanced probe [9]. As per the field study conducted with a primary school, this tangible needed to be more personalised and show intuitive visual feedback for children. The third version of TurnTalk was thus realised [14]. In this version, each child faces a section of the pentagon device and has their own play-card for taking or reserving turns. Most importantly, this TurnTalk is adaptable by teacher, who can chose between types of feedback and when to show them (during or after the session). A number of LED strips on the device can light up corresponding to number of turns taken or the time taken by each child, thus providing group and individual feedback. Logged data concerning turns & times also enable post-conversation reflections concerning children's behaviours, which are mediated and customisable by teachers. This version of TurnTalk went through expert reviews by SEL and ID experts, and then it was used with 9 children in a learning context. Again, design spaces were opened up, through designers' conversations with children and teachers, observations, and logs of TurnTalk [11]. A novel TurnTalk was rapidly released and has been recently used in a repeated measure field study in 2017 in a primary school.

Finally, the modularity and adaptability of TurnTalk also allowed us to re-purpose TurnTalk for a novel context, not initially envisioned: TurnTalk has been recently adapted to an experimental context with deaf and hearing users. This design of TurnTalk is being evaluated to assess if its feedback is equally helpful for deaf users and hearing users, or whether it should be differently designed, in line with the requirements set in [10] and the design guidelines advanced in [15].



4 Conclusions

This paper outlined the research questions and participatory design approach taken in the PhD work being done for development of SEL tangibles centered around group conversations for children, in learning contexts. The paper focused on the TurnTalk tangible for the turn-sharing norm. It showed how it evolved through reflections, actions in the field and rapid development cycles. It motivated why such tangibles should be disposable, modular, adaptable and open for appropriation and re-purposing. The design case showed how such characteristics made it easy to abandon ID solutions in case of failure in field studies, adapt them according to children's usage and teachers' suggestions, as well as to re-purpose them for novel contexts, not initially envisioned.

References

1. Africano, D., Berg, S., Lindbergh, K., Lundholm, P., Nilbrink, F., Persson, A.: Designing tangible interfaces for children's collaboration. In: CHI '04 Ext Abs on Human Factors in Comp Sys. pp. 853–868. CHI EA '04, ACM, NY, USA (2004)
2. Bachour, K., Kaplan, F., Dillenbourg, P.: An Interactive Table for Supporting Participation Balance in Face-to-Face Collaborative Learning. IEEE Trans on Learning Technologies 3(3), 203–213 (July 2010)
3. Bergstrom, T., Karahalios, K.: Conversation Clock: Visualizing Audio Patterns in Co-located Groups. In: Proc. of 40th Annual Hawaii Int. Conf. on System Sciences. HICSS '07, IEEE Computer Society, Washington, DC, USA (2007)
4. Cabitza, F., Fogli, D., Piccinno, A.: "Each to His Own": Distinguishing Activities, Roles and Artifacts in EUD Practices, pp. 193–205. Springer Int. Pub. (2014)
5. DiMicco, J.M., Hollenbach, K.J., Pandolfo, A., Bender, W.: The Impact of Increased Awareness While Face-to-face. Hum.-Comput. Interc. 22(1), 47–96 (2007)
6. Durlak, J.A., Weissberg, R.P., Dymnicki, A.B., Taylor, R.D., Schellinger, K.B.: The impact of enhancing student's social and emotional learning: A meta-analysis of school-based universal interventions. Child Development 82(1), 405–432 (2011)
7. Fisher, G., Fogli, D., Piccinno, A.: Revisiting and Broadening the Meta-Design Framework for End-User Development (in print)
8. Gennari, R., Melonio, A., Raccanello, D., Brondino, M., Dodero, G., Pasini, M., Torello, S.: Children's emotions and quality of products in participatory game design. Int. Journal of Human Computer Studies 101, 45–61 (2017)
9. Gennari, R., Melonio, A., Torello, S.: Gamified probes for cooperative learning: a case study. Multimedia Tools and Applications 76(4), 4925–4949 (2017)
10. Gennari, R., Pavani, F., Rizvi, M.: Tangible design for inclusive conversations with deaf or hard-of-hearing children. In: Proc. of the 1st Int. Symposium on Emerging Technologies for Education (SETE 2016), LNCS, Springer, Rome (2016)
11. Gennari, R., Melonio, A., Rizvi, M.: Participatory Design of Tangibles for Children's Socio-Emotional Learning. In: Proc. of the 6th Int. Symposium on End User Development. IS-EUD 2017, Springer (2017)
12. Leonardi, C., Pianesi, F., Tomasini, D., Zancanaro, M.: The Collaborative Workspace: A Co-located Tabletop Device to Support Meetings, pp. 187–205. Springer London, London (2009)
13. Martinez Maldonado, R., Kay, J., Yacef, K., Schwendimann, B.: An Interactive Teacher's Dashboard for Monitoring Groups in a Multi-tabletop Learning Environment, pp. 482–492. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
14. Melonio, A., Rizvi, M.: The Design of TurnTalk for The Scaffolding of Balanced Conversations in Groups of Children. In: Proc. of the 1st Int. Symposium on Emerging Technologies for Education (SETE 2016), Rome (2016)
15. Melonio, A., Gennari, R.: How to design games for deaf children: Evidence-based guidelines. In: Proc. of 2nd Int. Workshop on Evidence-based Technology Enhanced Learning. pp. 83–92. Springer, Heidelberg (2013)
16. Schiavo, G., Cappelletti, A., Mencarini, E., Stock, O., Zancanaro, M.: Overt or Subtle? Supporting Group Conversations with Automatically Targeted Directives. In: Proc. of the 19th Int. Conf. on Intelligent User Interfaces. pp. 225–234. IUI'14, ACM, New York, NY, USA (2014)
17. Slovák, P., Fitzpatrick, G.: Teaching and developing social and emotional skills with technology. ACM Trans. Comput.-Hum. Interact. 22(4), 19:1–19:34 (Jun 2015)

EUD Models and Techniques for the Smart-Home

Fabio Cassano

Dipartimento di Informatica,
Università degli Studi di Bari Aldo Moro,
Via Edoardo Orabona, 4, 70126 Bari, Italy

Abstract. Nowadays most of the electronic devices are able to both connect to the Internet and create an in-home-environment which can be managed by users through mobile phones or web applications. Moreover, many products developed by the most important IT companies allow end users to easily customize the behaviour of their smart systems via Graphic User Interfaces (GUI). Most of the devices (including air conditioning systems, lamps, fridge, washing machines etc.) have their specific application, with which is possible to control and manage only the behaviour of a specific item. However, multiple devices even by the same manufacturer, need a specific app to be configured and there is no possibility for end users to take the full control of the smart system and customize it to the everyday needs. In this work I report a way to overcome this problem, by End-User Development techniques, to allow involved users to manage, customize and develop part of the existing smart-home environment. In particular, I propose some scenarios improving the quality lifestyle of the users via the Internet of Things (IoT) devices. For this aim, the proposal encompass a modification of the International Classification of Functioning (ICF). With the addition of some markers related to the Computer Science (CS) ability, I have proposed a solution to improve the overall user's ICF level.

1 Introduction

Nowadays all the electronic devices are connected to the Internet. This ability, named Internet of Things (IoT), can be used in different fields to: collect data, remotely manage devices, add pieces of information to the real world, etc. The most common domain where the IoT has been used is the smart-home. This can support people in the everyday life, for example, supporting families and even lone older people and disabled people. Those share the same needs from a smart-home environment and, as a matter of fact, the idea to consider this group of people as unique about a generic home environment, is deeply discussed in literature since the 90s [1][2]. So, the described scenario shows how nowadays smart-home technologies lack in customization and the gives hints on how rising technologies can be used in order to improve people's lives and health. To objectively evaluate the health functionalities of a person, the World Health

Organization (WHO) has developed a scale called "International Classification of Functioning" (ICF). The ICF scale describes a person's functioning at three prospectives: body person and societal. Each component of the functioning and disability is evaluated by many qualifier that help physicians to evaluate the level of disability of the person. The higher the value, the better is the person's status.

Every IoT device currently is interfaced directly with the vendor's website. Data collected from the device is calculated according to "fixed" algorithms and are sold by vendors to third-party companies. The ICF scale can be used to objectively calculate a disability value, and the smart-home can be used to collect and generate custom reports for care-providers. How smart-home systems can be used to support both older and disabled is a filed of research that have been explored from the last fifteen years. In a previous work, we have proposed a dynamic approach to the data evaluation coming from IoT devices by caregivers and care-receivers [3]. With this work, I want to face the further problems emerged during that study which are: the better cooperation between the data collection coming from IoT devices and the physicians involved in the patient's therapy. There are many pilot studies that states the importance to propose older people a simple smart-system in order to allow them to get the advantages from it and improve their health status [4] [5]. The importance for both the wellness and health with the support of smart-home appliances is stated in many works in literature [6][7][8]. IoT and mobile wearable devices (such as smart watches) are becoming more and more affordable. Their capability to track someone's health parameters is pushing forward the study for a new field of research which involves the patient remote assistance [9].

2 Proposed approach

The research questions to be answered are:

- RQ1: How can a smart-home environment improve ICF values of a user?
- RQ2: Which EUD technique better involves users (caregivers, family members and patients) to edit the behaviour of the smart-home?
- RQ3: Can a framework be modelled in order to use it in the smart-home environment with cheap Internet of Things (IoT) devices?
- RQ4: How the framework can manage different users needs?
- RQ5: What are the best technologies to be used when older or disable people are in the smart-home?

The proposed approach consist into two different parts: the first one is related to the ICF scale, the second one involves the definition of a specific study case. The International Classification of Functioning, Disability and Health (ICF) is a framework for describing and organising information on functioning and disability. It provides a standard language and a conceptual basis for the definition and measurement of health and disability [10]. The common way to calculate the

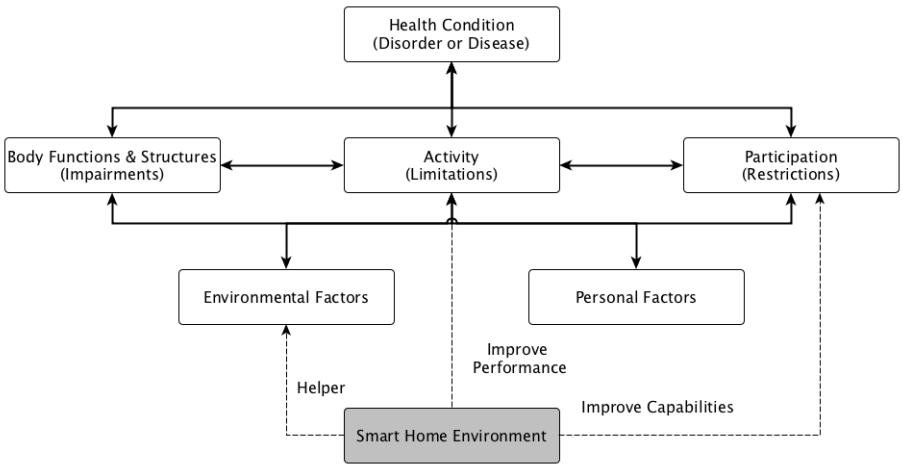


Fig. 1: The proposed ICF scheme with the introduction of the smart-home environment

ICF for each person is to evaluate different markers belonging to every aspect of the patient's lifestyle and ability.

In Figure 1, the white boxes represent the ICF scheme. The grey boxes at the bottom of the Figure 1 represents my proposed modification to the ICF scale. Firstly it can act as helper for the in the ICF's "Environmental factor", improving the house environment where the person live, with a smart-home environment, specifically developed. Secondly, it can improve the person's activity by forcing he/she to do specific activities (defined by the caregiver). Performances, in terms of physical movement and thinking, are an important aspect for people affected by dementia of a mild cognitive impairment [11]. The support for those users is fundamental and the Smart-home environment could play an important role to help the user's activities. Thirdly, it can improve the participation of the family (for example) into the person's therapy and subsequently improve the "Participation" ICF value. Care-providers will be closely involved to the user's therapy via many gamification techniques [12]. It introduces two more qualifiers to the scale system. I have considered the patient's Computer Science ability in two separate ways:

- First Qualifier: physical ability to interact with the smart-home system. To evaluate a person, this qualifier may be divided into 3 or more levels (low, mid, high, for example);
- Second Qualifier: mind ability to understand and interact with the smart-home system. As the previous qualifier, this can be divided into 3 or more levels.

The second part of my work is to define a scenario to test whether the proposed approach is valuable. Studying the literature I propose a scenario to help disabled

and older people with the support of the smart-home environment. It involves the usage of wearable IoT devices to keep alive the mental ability of a patient [13] and a smart-home environment to analyze data, manage and send information to care-providers. Sensors can be used to collect the data from different sources and the EUD allows families (or in general care-providers), to customize the movement(s) or the path that the patient should do. The smart-home environment finally, is used as supervisor of the patient's therapy. The possibility given to the family members and care-providers to customize the smart-home environment allows to tailor the therapy to the real needs of the patient. In this way, the ICF value improves the patient's lifestyle and health.

3 Conclusions and Future Works

In this work, I have studied the most common problems involving the smart-home environment related to older and disabled people. I have then proposed some study cases that could be developed in order to improve older and disabled people's lifestyle. As future works, the development of the scenario and then some user's test is needed. Moreover, emergent technologies like Virtual Reality are getting used into the smart-home systems to support older and disabled people. This can be evaluated in a smart-home system to keep the patient's brain trained through many simple games. The most known EUD techniques can be used to allow caregivers and care-providers to deeply customize the games and decide the sequence or the timing for each gaming session. The smart-home environment can be used to keep track of the trend and report improvements (if any) to the trusted care-provider and give the patient reminders for the scheduled activity.

References

1. Schraft, R., Schaeffer, C., May, T.: Care-o-bot/sup tm: the concept of a system for assisting elderly or disabled persons in home environments. In: Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE. vol. 4, pp. 2476–2481. IEEE (1998)
2. Hautala, M., Keränen, N.S., Leinonen, E., Kangas, M., Jämsä, T.: Ict use in family caregiving of elderly and disabled subjects. In: eHealth 360, pp. 42–48. Springer (2017)
3. Bevilacqua, V., Cassano, F., Dimauro, G., Girardi, F., Piccinno, A.: A dynamic approach to medical data visualization and interaction. In: CEUR WORKSHOP PROCEEDINGS. vol. 1658, pp. 7–12. CEUR-WS (2016)
4. Demiris, G., Rantz, M.J., Aud, M.A., Marek, K.D., Tyrer, H.W., Skubic, M., Hussam, A.A.: Older adults' attitudes towards and perceptions of smart hometechnologies: a pilot study. Medical informatics and the Internet in medicine **29**(2), 87–94 (2004)
5. Gaddam, A., Mukhopadhyay, S.C., Gupta, G.S.: Elder care based on cognitive sensor network. IEEE Sensors Journal **11**(3), 574–581 (2011)
6. Coughlin, J.F., D'Ambrosio, L.A., Reimer, B., Pratt, M.R.: Older adult perceptions of smart home technologies: implications for research, policy & market innovations

in healthcare. In: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE. pp. 1810–1815. IEEE (2007)
7. Demiris, G., Hensel, B.K., Skubic, M., Rantz, M.: Senior residents perceived need of and preferences for smart home sensor technologies. International journal of technology assessment in health care **24**(01), 120–124 (2008)
8. Adair, B., Miller, K., Ozanne, E., Hansen, R., Pearce, A.J., Santamaria, N., Viegas, L., Long, M., Said, C.M.: Smart-home technologies to assist older people to live well at home. Journal of aging science (2013)
9. Mukhopadhyay, S.C.: Wearable sensors for human activity monitoring: A review. IEEE sensors journal **15**(3), 1321–1330 (2015)
10. World Health Organization: International Classification of Functioning, Disability and Health: ICF. World Health Organization (2001)
11. Petersen, R.C.: Mild cognitive impairment as a diagnostic entity. Journal of internal medicine **256**(3), 183–194 (2004)
12. Benzi, F., Cabitza, F., Fogli, D., Lanzilotti, R., Piccinno, A.: Gamification techniques for rule management in ambient intelligence. In: European Conference on Ambient Intelligence. pp. 353–356. Springer (2015)
13. Bisio, I., Delfino, A., Lavagetto, F., Sciarrone, A.: Enabling iot for in-home rehabilitation: Accelerometer signals classification methods for activity and movement recognition. IEEE Internet of Things Journal (2016)

Investigation on Sense of Presence Experience Parameter for Joystick Interface in Remote Operated Container Crane Application

U.N.N. Abdullah[1]

[1] Lappeenranta University of Technology, Skinnarilankatu 34, 53850 Lappeenranta, Finland

Ummi.Abdullah@lut.fi

Abstract. Lack of direct motion feeling is recognized as a possible weakness in remote operated container crane (ROCC) application. This could lead to less safe working environment especially to people working in the terminal area. An end user experience investigation was conducted by using semi structured interviews and task observations for the purpose of parameters' study. Six parameters namely weight, size, material, handling behavior, position, delay data and semantics were studied to provide measurable values to the sense of presence experience goal. This investigation provided a direct feedback from the operators as active domain developers in improving the control and handling interface for ROCC application in specific as well as for remote operated off-road applications in general.

Keywords: End User, User Experience, Remote Operation, Container Crane, Joystick.

1 Introduction

Automated off-road vehicle system and its operation such as for container cranes has become a worldwide demand and its evolution due to safety and health requirements has been increasing in recent years. Therefore, many previous researches had produced studies related to automation or remote operated systems for container crane application such as in[1–7].

After years of research, the lack of direct motion feeling was identified as a possible weakness in remote operation implementation. This lack of direct motion feeling between physical operation and operation through monitors' view experienced by crane operators could lead to a less safe handling operation and endanger the people in the terminal blocks. An automated operation is also limited by the view that can be provided by the station monitor in the remote offices. This could also be contributing to a less safe handling scenario.

This research therefore, presents an investigation into the parameters that provide measurable values to the sense of presence goal to improve the lack of direct motion feeling for remote operated container crane station (ROCCS) joystick interface. In

addition, this research implements and emphasizes the roles of operators as domain developers that has been defined and discussed in end-user development EUD concept [8] in providing the experience feedback to facilitate designer in modifying the joystick interface as artifact and its system to fit their evolving requirements.

2 Theory

This research adapting EUD concept that just not only important in the software systems establishment but recently EUD methods, approaches and tools also expended into various socio-technical environments and applications to facilitate the end-users to act as professionals in those domains in which they are not the expertise[9].

User experiences (UX) principles and approaches [10] are applied in investigating the parameters for the sense of presence experience through joystick interface. UX goals from previous research [5] were referred to and redefined for the purpose of this study. They are;

- The joystick interface will provide a natural interaction between the operator and the remote system.
- The interaction options will not disturb data transmission.
- The interaction options will assist the operators in understanding the information from the operation system.

3 Method

Thirteen remote crane operators were interviewed and observed while they operate the existing remote operated control and handling interface systems at two international terminals. Five levels of the Likert Scale was used in the semi structured interviews with Scale One defined for strongly agree, Scale Two as agree, Scale Three as not sure, Scale four as disagree and Scale Five as strongly disagree. The operators had to choose most suitable scale for every relevant question based on their experiences. The observed tasks included unloading the container from block, loading the container on trucks and stacking the container in the block. Task analysis and activity analysis from video observations were conducted to validate the interview results.

4 Results

4.1 Experience based on natural interaction

Interview results indicated that six out of thirteen operators disagree that the existing joysticks were heavy and bulky while five of them agreed that the weights and sizes of the existing joysticks need to be improved to provide a natural interaction between the user and the handling system.

Based on the interviews and observations, all operators had interacted with their joysticks using both hands. Eleven operators responded that they usually sit at their control stations when using the joysticks while one of them preferred to alternately sit and stand while using the joystick.

4.2 Experience based on data delay

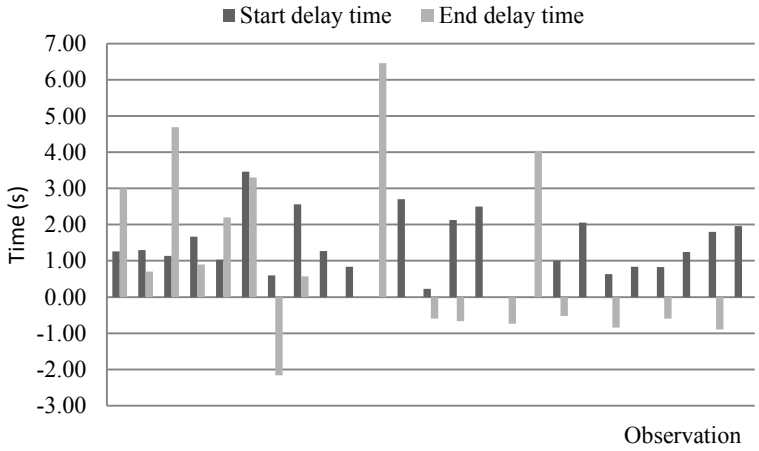


Fig. 1. Results on data delay experienced by operators when using existing joysticks

Figure 1 describes the delay time experienced by thirteen operators in handling the joysticks i.e. pull, push, move to left and move to right. Positive values mean that the spreader of the crane was still moving even when the operator had released the joystick, while negative values mean that the spreader had already stopped even when the operator was still performing the function. This delay in time could be considered to reflect the data delay in the joystick information system.

4.3 Experience based on semantics

The feedback from the interviews regarding the joystick semantics arrangement showed that eleven of the operators agree that the current joystick semantics easy to understand.

5 Conclusion

Six parameters were investigated to support the sense of presence goal for the lack of direct motion feeling issue in ROCC by implementing EUD and UX principles and

approaches i.e. interviews and operational observations. Future research should focus on developing the engineering parameters based on experience parameters, so that they could be measured and physically tested.

Acknowledgement. The author acknowledges the support of Konecranes Plc, Terminal Teluk Lamong and Semarang International Container Terminal, Indonesia.

References

1. Karvonen, H., Koskinen, H., Haggrén, J.: Defining User Experiences Goals for Future Concepts; A Case Study. In: Vaataja, H., Olsson, T., Roto, V., and Savioja, P. (eds.) NordiCHI'12. pp. 11–14. Tampere University of Technology, Tempere,Finland (2012).
2. Kaasinen, E., Roto, V., Hakulinen, J., Heimonen, T., Jokinen, J.P.P., Karvonen, H., Keskinen, T., Koskinen, H., Lu, Y., Saariluoma, P., Tokkonen, H., Turunen, M.: Defining user experience goals to guide the design of industrial systems. Behav. Inf. Technol. 3001, 1–16 (2015). doi:10.1080/0144929X.2015.1035335.
3. Gustafsson, T., Heidenback, C.: Automatic control of unmanned cranes at the Pasir Panjang terminal. In: Conference on Control Applications. pp. 180–185 vol.1 (2002).
4. Karvonen, H., Koskinen, H., Haggrén, J.: Enhancing the User Experience of the Crane Operator: Comparing Work Demands in Two Operational Settings. In: Proceedings of the 30th European Conference on Cognitive Ergonomics. pp. 37–44. ACM, Edinburgh, United Kingdom (2012).
5. Koskinen, H., Karvonen, H., Tokkonen, H.: User experience targets as design drivers. In: Proceedings of the 31st European Conference on Cognitive Ergonomics - ECCE '13. p. 1 (2013).
6. Karvonen, H., Koskinen, H., Tokkonen, H., Hakulinen, J.: Evaluation of user experience goal fulfillment: Case remote operator station. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 366–377 (2014).
7. Kaasinen, E., Väätäjä, H., Karvonen, H., Lu, Y.: The fuzzy front end of experience design. In: Proceedings of the 8th Nordic Conference on Human-Computer Interaction Fun, Fast, Foundational - NordiCHI '14. pp. 797–800 (2014).
8. Cabitza, F., Fogli, D., Piccinno, A.: "Each to his own": Distinguishing activities, roles and Artifacts in EUD practices. In: Caporarello,L., Di Martino,B., Martinez,M. (eds) Smart Organizations and smart artifacts. LNISO, vol.7, . pp. 193–205. Springer International Publishing, Switzerland (2014).
9. Fischer, G., Fogli, D., Piccinno, A. : Revisiting and broadening the meta-design framework for end-user development. In: Paterno, F., Wulf, V (eds.) New perspectives in end user development. Springer Int. (2006)..
10. Nemeth, C.P.: Human factors methods for design : making systems human-centered, http://ezproxy.unimap.edu.my:2481/isbn/9780203643662.

Interactive Objects for Algorithmic Thinking: Why and How

Andrea Bonani

Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
abonani@unibz.it

Abstract. Computer science education at K-12 schools often includes computational thinking and algorithmic thinking. This paper supports the idea that algorithmic thinking should be taught, from primary school, through interactive tangible objects. The paper focuses on the design of such tangibles for teaching graph algorithmic thinking.

Keywords: Algorithmic Thinking; Interaction Design; Interactive Tangible Objects

1 Introduction

Algorithmic Thinking (*AT*), which is an important part of the best known *Computational Thinking* (*CT*) [13], requires to abstract the essentials of a problem and model its solution, so as to give step-by-step solution instructions (an algorithm) [6]. In many countries, computer science education in primary and secondary schools (K-12) has shifted towards CT, in general, and AT, specifically [4]. In K-12, the use of interactive tangibles for teaching AT potentially fosters the interplay between abstraction and concreteness, thus helping even the youngest to learn through visual and tactile experiences. This paper focuses on *interactive tangible objects for graph algorithmic thinking* (briefly, *tangibles for graph AT*), for primary and middle schools.

The design of such tangibles is highly complex: it should be based on releasing evolving tangibles, and exploring their users' appropriation in field studies, according to their specific learning contexts.

2 Research Work, Questions and Process

Related Research Work. Literature on computer science education highlights three main categories of approaches to teaching CT, in general, and AT, in particular: (1) without computers; (2) by coding, with computers; (3) with interactive tangible objects.

Perhaps the best known approaches to AT education are based on *coding*, with programming environments for children, such as Scratch [12]. Proposals such as *CS-unplugged* by Bell [1], instead, teach CT without computers: they

require physical activities, using everyday materials like paper and pencil. CS-unplugged has inspired several researchers, including Gibson, who taught graph modeling and algorithms through physical non-interactive objects [10].

Relevant related work for AT education can also be found in the area of *interaction design of tangibles*, e.g., [2,11]. Proposals to teach algorithms through the use of tangibles have increased in recent years. A significant reference is [7], where a gamified tangible for primary schools is presented, BALA, for the scaffolding of a sorting algorithm. Therein, gamification is used for creating probe versions of tangibles for children as in [8,9].

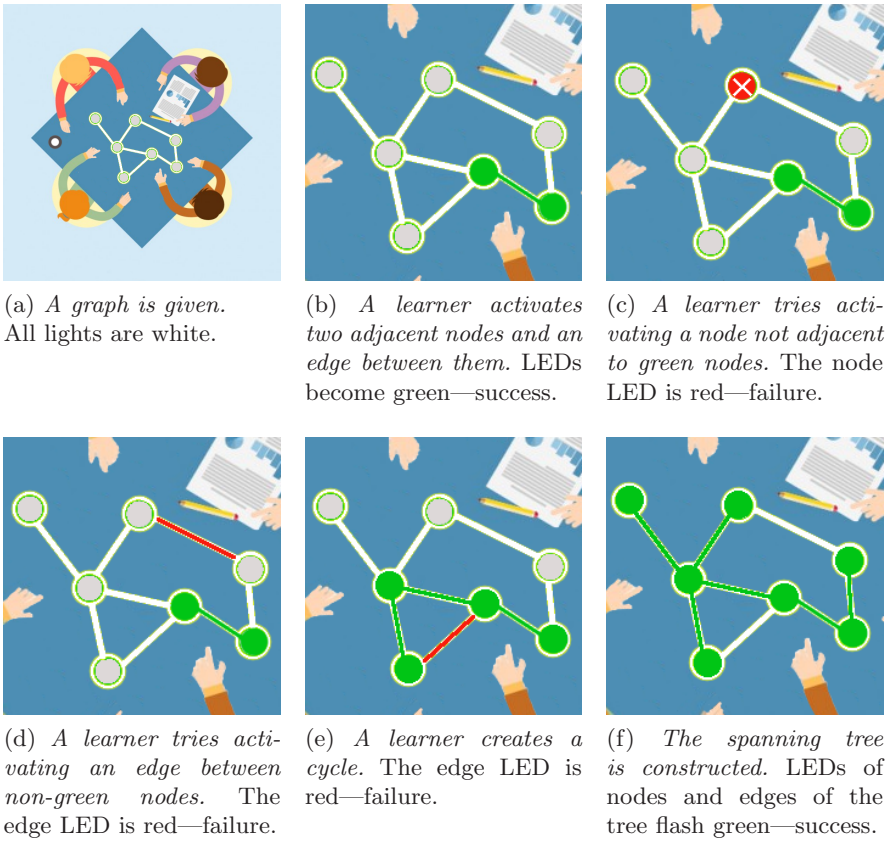
Research Questions. The primary goal of my Ph.D. research is how to design interactive tangible objects (briefly, *tangibles*) for scaffolding AT, so that tangibles are usable by their intended users, and can provide them with an enhanced learning experience. To achieve my research goal, my research work is framed around research questions and measurable objectives: **(RQ1** What are basic requirements for tangibles for AT in learning contexts? **(RQ2)** How can we design tangibles for AT with learners and teachers, for their learning contexts? **(RQ3)** How can we generalise tangible for AT design to novel learning contexts?

Design Process. Given the aforementioned research goals, my design process involves users continuously, so as to transform learning "by empowering all people to become active contributors" [5]. Its roots are in participatory design with action research. The primary users of tangibles for AT are teachers and their 9–13 years old pupils, from primary and middle schools. The design process proceeds through prototype solutions, conceived as intermediary objects in the sense of [3], which are used in studies with designers and users for detecting usability issues, exploring novel design possibilities as well as creating learning possibilities for users.

The design process started with an exploratory context of use analysis, which triggered the first alternative design ideas, assessed with interaction design experts. After building a first vertical prototype of a tangible for AT (with few critical functionalities implemented and open for appropriations and rapid changes), designers and users participated in two studies in informal learning contexts. Tangibles were rapidly assessed and redesigned according to study results. For instance, new learning scenarios for primary and middle schools were developed together with teachers. New design features were added after studies.

3 Example Tangibles For Graph Algorithmic Thinking

The first prototype that we developed is made of tangibles for graph AT. The prototype has a client-server architecture. Clients are the main tangibles for learners, embedding Raspberry PIs and other micro-electronics: nodes (boxes) and edges (cables), with their buttons and LEDs. The server is a computer that (1) verifies graph properties or algorithms, (2) implements a monitoring tool for teachers, and (3) interacts with nodes through a WiFi connection.



(a) *A graph is given.* All lights are white.

(b) *A learner activates two adjacent nodes and an edge between them.* LEDs become green—success.

(c) *A learner tries activating a node not adjacent to green nodes.* The node LED is red—failure.

(d) *A learner tries activating an edge between non-green nodes.* The edge LED is red—failure.

(e) *A learner creates a cycle.* The edge LED is red—failure.

(f) *The spanning tree is constructed.* LEDs of nodes and edges of the tree flash green—success.

Fig. 1: The construction of a spanning tree for a graph.

Learners can construct graphs, explore graph properties or algorithms by physically connecting tangible nodes and edges. For instance, they can build a spanning tree for a given graph in an exploratory fashion (verifying the spanning tree properties) or in a guided manner (according to a specific algorithm). Figure 1 shows a storyboard describing how primary-school learners may learn of an algorithm for creating a spanning tree for a graph with tangibles.

4 Conclusions

The paper introduced the idea of tangibles for AT, focussing on graph AT, which can aid in the scaffolding of AT through an active multi-modal experience. The design of such tangibles is based on participatory design with an action-research approach. Teachers, children (users) and designers co-discover design and learning possibilities by using and modifying tangibles and scenarios. A storyboard

showing how to experience the construction of a spanning tree illustrates educational potentials of the participatory design of tangibles for (graph) AT.

References

1. Bell, T., Alexander, J., Freeman, I., Grimley, M.: Computer science without computers: new outreach methods from old tricks. In: Proc. of the 21st Annual Conference of the National Advisory Committee on Computing Qualifications (2008)
2. Bers, M.U., Flannery, L., Kazakoff, E.R., Sullivan, A.: Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. Computers & Education 72, 145–157 (2014)
3. Cabitza, F., Fogli, D., Piccinno, A.: "Each to His Own": Distinguishing Activities, Roles and Artifacts in EUD Practices, pp. 193–205. Springer International Publishing, Cham (2014)
4. European Schoolnet: Computing our future (2015), available as http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf
5. Fisher, G., Fogli, D., Piccinno, A.: Revisiting and Broadening the Meta-Design Framework for End-User Development (in print)
6. Futschek, G.: Algorithmic thinking: the key for understanding computer science. In: Informatics education–the bridge between using and understanding computers, pp. 159–168. Springer (2006)
7. Gennari, R., Del Fatto, V., Gashi, E., Sanin, J., Ventura, A.: Gamified Technology Probes for Scaffolding Computational Thinking, pp. 303–307. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-33464-6_19
8. Gennari, R., Melonio, A., Raccanello, D., Brondino, M., Dodero, G., Pasini, M., Torello, S.: Children's emotions and quality of products in participatory game design. International Journal of Human Computer Studies, year=2017 101, 45–61, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85011661549&doi=10.1016%2fj.ijhcs.2017.01.006&partnerID=40&md5=433faab3bf27cea7b9cbdabf116c295a, cited By 0
9. Gennari, R., Melonio, A., Torello, S.: Gamified probes for cooperative learning: a case study. Multimedia Tools and Applications 76(4), 4925–4949 (2017), https://www.scopus.com/inward/record.uri?eid=2-s2.0-84966708840&doi=10.1007%2fs11042-016-3543-7&partnerID=40&md5=a24e1a9f4cb696dfa4fa61a1dc289573, cited By 0
10. Gibson, J.P.: Teaching graph algorithms to children of all ages. Proc. of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12) p. 34 (2012), http://dl.acm.org/citation.cfm?doid=2325296.2325308
11. Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L.: Computational thinking for youth in practice. ACM Inroads 2(1), 32–37 (2011)
12. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. ACM Transactions on Computing Education (TOCE) 10(4), 16 (2010)
13. Wing, J.: Computational thinking. Comm. of the ACM 49(3), 33–35 (2006)

Privacy Policy Modelling in Mobile Applications

Sophia Kununka

Alliance Manchester Business School,
The University of Manchester, Manchester, UK
sophia.Kununka@postgrad.mbs.ac.uk

Abstract. Mobile applications (apps) represent an exciting development in software delivery. However, apps collect extensive user details used for instance for the personalization of the users' experience and, can serve as means of surveillance thus a source privacy concern. One measure of addressing privacy concerns is the provision of privacy policies by apps. Privacy policies however fail to effectively address the privacy concerns as they offer limited user comprehension. A possible cause is that privacy policies are designed from the service providers' perspective, highlighting the necessity of user involvement so as to develop more effective policies. This research entails an in-depth analysis of 100 mobile application privacy policies and develops a clear privacy policy classification scheme. User perspectives on privacy policies are also explored and findings used to inform the development of user centred privacy policies that are easy to understand and optimise user empowerment over personal data privacy.

Keywords. Mobile applications · Privacy policy ·Privacy taxonomy · End user development

1 Introduction

While a range of approaches have been used in an endeavour to address digital privacy concerns, a key focus has been on the provision of privacy policies. A privacy policy is 'a set of rules, or statements that specify which processing and sharing practices are permitted for different types of data collectable from the user [1]. Privacy policies endeavour to guarantee data gathering and dissemination however, they are inadequate in their attempt to address user privacy [2]. While there is no simple solution to addressing privacy challenges, the unhampered growth and further adoption of digital media such as apps, websites and IoT dictates that data privacy issues are given adequate attention.

Diverse research initiatives have been conducted to explore privacy policy related challenges. The widespread traditional full length privacy policy representation is critiqued for its lengthiness, ambiguity and ineffectiveness. In addition, the traditional full length policy representation tends to focus towards service providers or industrial requirements, with minimal consideration for the end user. Research into the devel-

opment of improved privacy policy representations has included: Privacy tables and privacy nutrition labels [3], multilayered privacy policies, privacy icons [4], short notices that use risk or expectation scores [5-6]. The proposed improved policy representations are usual shorter, more understandable and facilitate better viewing of privacy content. However, the proposed representations have exhibited mixed success and are faced with several challenges. The proposed privacy policies do not always significantly improve times required to find or understand privacy information. It is difficult for proposed representations to distinguish whether accessing certain sensitive resources is necessary. Further challenges include the need for privacy information taxonomic presentation, standardization and, legal and industry support.

Based on the literature, it was established that an essential requirement for effective representation of privacy to the end user is in finding a balance between the provision of adequate information to support informed user consent while avoiding the elimination of relevant information which could limit the meaningfulness and relevance of information. This necessitates privacy policies that are not merely centred on the service providers' perspectives, but that incorporate user perspectives thus an end user development approach to policy design. Users don't care about technical complexity, hence there is need to hide the technical complexity in policies yet reveal task complexity. This involves understanding how users perceive specifications or privacy preferences and then representing them in a way that users would understand them, thus empowering users with specific control over policies and specifications over tasks. Next, the research questions used to explore the privacy challenges are presented.

2 Research Questions

Two research questions each with two sub questions are explored: Research question one: What are the key dimensions in mobile application privacy policies? This question is addressed by answering the two questions: a). How can the nature and usage of the collected attributes to be qualitatively classified? b).What do users consider as the essential privacy aspects in policies? Research question two: What is a good representation of privacy policies? This question is addressed by answering these two questions: a). What makes policies comprehensive? b). What is the right balance between abstraction and control to optimise user empowerment? The research strategy below was adopted to facilitate the process of attempting to answer the research questions.

3 Research Method

The choice of research strategy that should be applied is determined based on the aim of the study, the data required and its accessibility [7]. The first phase of the research in this thesis sought to establish the nature and usage of the main privacy policy attributes and to develop a qualitative classification (Research question 1, a). This involved secondary data collection in form of 100 mobile applications' privacy poli-

cies that were sourced online and selected on the basis of the mobile applications' popularity as ranked by the app stores. Other selection considerations included apps' categorization, user rating, number of installations, developer details and platforms.

The privacy policies were qualitatively analysed using Nvivo [8] in order to understand the concepts and characteristics comprised within existing mobile application policies such as the information gathered about individuals, the terms of use of the gathered information and how the collected information is used by service providers/third parties. Based on the findings of the policy content analysis, the policy information and domain knowledge was organized into an intuitive and clear comprehensive mobile app privacy policy taxonomy. The process of taxonomy development was guided by the information systems taxonomy methodology developed by [9].

The second phase of the research in this thesis involved a study on app users' perspectives on privacy policies that involved forty-one (41) participants [10]. The study as administered as a questionnaire survey that included both semi structured questions as well as open end questions. Participants were also required to design or draw a privacy policy and, to compare four (4) alternative privacy policy representations. The initial analysis of this study's findings focused on exploring what users consider as essential privacy aspects in a policy (Research question 1, b). It was established that users focused more on privacy aspects of data collection and use while little attention was given to the privacy areas of data monetisation and legal aspects. This could be an indicator of lack of adequate user understanding and a 'sense' of limited control over these privacy aspects. As such, based on these findings, a user centred privacy policy design prototype was developed and evaluated using the cognitive dimension framework [11]. The prototype design facilitates users with more privacy awareness and control over personal privacy in particular the monetisation of users data by apps.

The next step in the analysis of the study on users' perspectives on privacy policies is currently underway with the objective of answering (Research question 2, a & b) which seek to establish what a good representation of policy is in terms of information comprehensiveness and finding a balance between abstraction and control to optimise user empowerment. This involves exploration of the policy designs drawn by each participant in order to establish the users' mental models and the key design features depicted in the designs such as relates to structure, interactive-ness and, the interplay between textual and graphical aspects of the users' designs. When this analysis is finalized, it is hoped that the factors that contribute to 'easy to understand' representations would have been established and will be used in the development of new alternatives of user centred privacy policy representations based on natural programming [12] and user choices. Natural programming facilitates the presentation of functionality or information as perceived by non- technical users enhancing usability.

4 Contributions

A theoretical implication from this study is that the proposed taxonomy provides a body of knowledge that is pertinent in informing about the current state of mobile application privacy policies'. It provides an informative reference point for supporting research in policy development and policy regulation facilitating the work of regulators in the process of development and assessment of privacy standards for industry.

The prototype privacy policy design will improve user awareness and control over the monetization of personal data. This user focused design will facilitate informed user consent thus contributing towards reducing user concerns over privacy and fostering greater user confidence in the use and adoption of apps.

References

1. Papanikolaou, N., Creese, S., Goldsmith, M.: Refinement checking for privacy policies. Science of Computer Programming, 1198-1209 (2012)
2. Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., Rao, J.: Understanding and capturing people's privacy policies in a mobile social networking application. Personal and Ubiquitous Computing, 401-412 (2009)
3. Kelley, P., Cesca, L., Bresee, J., Cranor, L.: Standardizing privacy notices: An online study of the nutrition label approach. In : Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, vol. 3, pp.1573 - 1582 (2010)
4. Holtz, L. E., Zwingelberg, H., Hansen, M.: Privacy policy icons. In : Privacy and Identity Management for Life, pp.279-285 (2011)
5. Mylonas, A., Theoharidou, M., Gritzalis, D.: Assessing privacy risks in android: A user-centric approach. In : In Workshop on Risk Assessment and Risk-Driven Testing (2014)
6. Liccardi, I., Pato, J., Weitzner, D. J., Abelson, H., De-Roure, a.: No technical understanding required: Helping users make informed choices about access to their personal data. In : In Proc. MOBIQUITOUS '14 (2014)
7. Naoum, S.: Dissertation research and writing for construction students 2nd edn. Butterworth-Heinmann, Routledge (2007)
8. QSR: WHAT IS NVIVO? In: QSR International. Available at: http://www.qsrinternational.com/what-is-nvivo
9. Nickerson, R. C., Varshney, U., Muntermann, J.: A method for taxonomy development and its application in information systems. European Journal of Information Systems 22(3), 336-359 (2013)
10. Kununka, S., Mehandjiev, N., Pedro, S., Konstantina, V.: End User Comprehension of Privacy Policy Representations. In : 6th International Symposium on End-User Development, Eindhoven (2017)
11. Green, G., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages and Computing 7(2), 131-174 (1996)
12. Myers, B., Pane, J., Ko, A.: Natural programming languages and environments. In : Communications of the ACM, vol. 47(9), pp.47-52 (2004)

Enabling Cultural Heritage Practitioners to create Visitor Interactive Digital Exhibits.

Andrew Stratton

Cultural Communication & Computing Research Institute, Sheffield Hallam University, UK
A.Stratton@shu.ac.uk

Abstract. This research focuses on the creation of Visitor Interactive Digital Exhibits (VIDEs) by domain experts working within Museums and Art Galleries. These experts, Cultural Heritage Practitioners (CHPs), are rarely programming skilled users. The creation and updating of digital equipment based interactions typically relies on a high level of technical programming skill to create 'executable' behaviour. The researcher proposes an alternative approach, where visual 'Quando' blocks, similar to education blocks used in languages such as Scratch, are repurposed for use by CHPs. Quando blocks are designed based on CHP requirements and researcher expertise. Collections of Quando blocks can then be used by CHPs to describe the behaviour for VIDEs for use by Cultural Heritage visitors.

Keywords: Visual Programming; Museums; Programming Environments; End-user Programming

1 Introduction

In Cultural Heritage Museums and Art Galleries, Digital Technologies are increasing visitor engagement and understanding. Nesta polled 891 organisations in 2013 [3] and found that digital adoption has been slow and limited; it was identified that *'a lack of resource in terms of staff time (68 per cent), internal budgets (68 per cent) and external funding (61 per cent) as the top three barriers to realising their digital aspirations'*. Availability of digital design and development was an issue for 39-41% of organisations, who felt they were not well served in software development, user interface design and user testing.

Two years later, in [4], there was increased concern, from 68% to 73%, about the lack of (digital) funding. The sparsity of external suppliers was also identified and there was a 'significant' reduction in organisations experimenting with digital technologies, taking risks and evaluating impact. A gap appears to have developed between 'digirati', the top 10% of digitally active organisations, and the rest of the sector; 57% of the digirati created standalone digital exhibits/works, compared to 23% across all polled organisations. The overall picture appears to show a gradual inclusion of digital technologies into the expected aspects of the organisations, but also a lack of time and digital skills.

Visitor Interactive Digital Exhibits (VIDEs) are a specific area within Digital Technologies that directly affect the Visitor ‘experience'. Currently the desire to create VIDEs exists, but the time and technical skills required are an issue. The technical skills issue can be reduced by 'upskilling' staff, though as quoted in [4], *'What we are never truly aware of is the amount of time and therefore cost required to train and develop the skills needed for the successful application of new technologies.'*

The aim of this research is to use Visual Programming techniques to enable CHPs to create VIDEs without having to develop programming skills.

2 Visual Programming

Visual Programming has been a research area since 1966 [10, 7]. The promise of Visual programming has always been to reduce the 'difficulty' of programming [11], though the promised benefits of Visual Languages were rarely measured.

The success of Scratch [8] is widely documented and has increased interest in the educational benefits of programming. Scratch is mainly used by 8 to 16 year olds and has a 'cartoon' like interface that is unlikely to fit in a professional environment. Scratch use Visual Blocks to represent language concepts, positioned within the editor by the users.

Scratch has influenced many subsequent visual languages, including Kodu [5], which offers a different approach for the domain of game engine programming for children, focusing on using higher level language constructs and game engine concepts. Kodu includes a rule based approach, where each rule is split into a three to five tuple containing the primitives: sensor, filter (, filter, filter) and selector.

The Blockly language [1] offers a flexible approach to developing Visual Block Languages and has a similar look and feel to Scratch. However, Blockly offers more than Scratch as a tool for supporting visual programming research, by allowing the building of blocks and associated generated textual output. Blockly does not require the destination language to execute within its own environment and there is no run time dependency.

In order to use Blockly, the phases of design need to be identified to separate the design approach used from the software process based implementation.

3 Design Phases

The design phases that the researcher has encountered, while working with CHPs at local museums [9], has been abstracted from the technical software implementation specific process. There are three overall phases, split into:

1. Conceptualization
2. Prototyping
3. Usage

These phases are divided into further phases as shown in Figure 1.

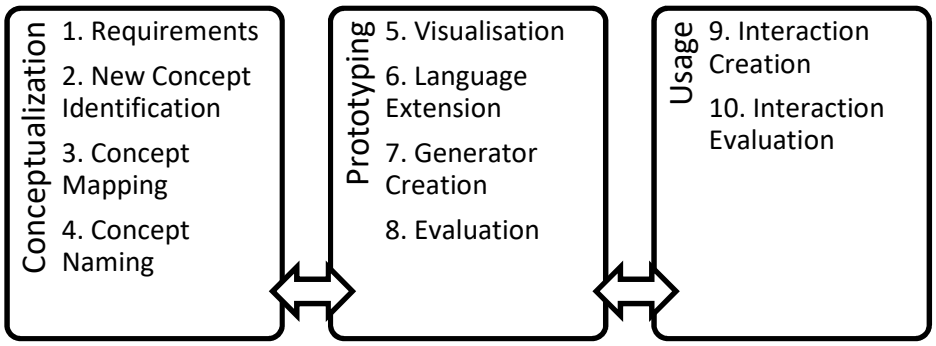


Fig. 1. Design Phases

The phases follow a co design approach involving CHPS similar to [6], except for the Language Extension and Generator Creation phases. The design of the visual block concepts occurs mostly during the Conceptualization phases. The visual block designs are then implemented, at different levels of fidelity, during the Prototyping Phase

Following these design phases, CHPs will typically follow with their own visitor evaluation.

4 Current Progress

The Quando toolset has been created as a design Artefact for executing within the above phases as part of a case study at Creswell Crags [9]. Quando allows CHPs to experience this approach with realistic implementations of Visual blocks and a browser based editor, as well as producing executable interactions also running from within standalone browser based deployments. An example interaction is shown in Figure 2.

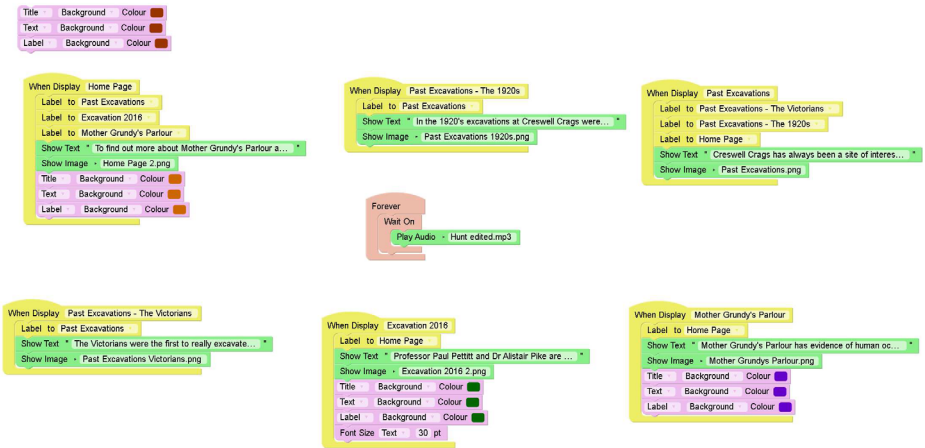


Fig. 2. Example interaction created by CHPs

With relatively few concepts, interactions can be created Following these design phases, CHPs will typically follow with their own visitor evaluation.

5 Future Direction and Advice Sought

In the near future, more case studies are to be used to validate the approach used and also to extend the concepts and blocks available for CHPs to use.

Advice concerning the which, of the many, subject areas might be appropriate would be valuable; End User Development [2] for domain experts is one of the candidate areas. Advice concerning the methodology used, as well as feedback on trade offs for methodology choice would also be valuable.

References

1. Fraser, N. (2016). Google blockly - a visual programming editor. URL: https://developers.google.com/blockly/. Accessed January 17.
2. Lieberman, H., Paternò, F., Klann, M. and Wulf, V., 2006. End-user development: An emerging paradigm. In End user development (pp. 1-8). Springer Netherlands.
3. London, M.T.M., 2013. Digital Culture: How Arts And Cultural Organisations In England Use Technology.
4. London, M.T.M., 2015. Digital Culture: How Arts And Cultural Organisations In England Use Technology.
5. MacLaurin, M.B., 2011, January. The design of Kodu: A tiny visual programming language for children on the Xbox 360. In ACM Sigplan Notices (Vol. 46, No. 1, pp. 241-246). ACM.
6. McDermott, F., Maye, L. & Avram, G., 2014. Co-designing a Collaborative Platform with Cultural Heritage Professionals. In Irish HCI conference. (pp. 18–24).
7. Nickerson, J. V., 1995, Visual programming, Ph.D. thesis, New York University, New York, NY, USA.
8. Resnick, M. et al., 2009. Scratch: Programming for All. Communications of the ACM, 52, (pp.60–67).
9. Stratton, A. et al., 2016. Investigating Domain Specific Visual Languages for Interactive Exhibitions. In Psychology of Programming 2016, 27th Annual Workshop. (pp 188-191) (pre-print).
10. Sutherland, W. R., 1966. The on-line graphical specification of computer procedures, Doctoral dissertation, Massachusetts Institute of Technology.
11. Whitley, K.N. and Blackwell, A.F., 1997, October. Visual programming: the outlook from academia and industry. In Papers presented at the seventh workshop on Empirical studies of programmers (pp. 180-208). ACM.