

international reports on socio-informatics

volume 6 issue 1
2009

*Proceedings of the Work-in-
Progress Session of the
Second International Symposium on
End User Development
March 2 - 4, 2009 Siegen, Germany*

Guest Editors:

Christopher Scaffidi
Gunnar Stevens

Editors:

Volkmar Pipek
Markus Rohde

Publisher:
IISI - International Institute
for Socio-Informatics

Table of contents

Table of contents.....	2
Impressum	3
An End User Development Environment for Culturally Contextualized Storytelling.....	4
Marcos Alexandre Rose Silva Junia Coutinho Anacletoauthor	
Scaffolding Collaborative Project Work in End-User Development	9
Matthias Korn Michael Veith	
An Outline for a Syllabus for Introducing End-user Type of Students to the Object-oriented Paradigm	13
Rony G. Flatscher	
A Toolkit Method to Match Up End User Needs with Salesforce.com Solutions	18
Ken Decreus Stijn Viaene Geert Poels	
Blurring the distinction between software design and work practice	24
Grace de la Flor Marina Jirotko	

The 'international reports on socio-informatics' are an online report series of the International Institute for Socio-Informatics, Bonn, Germany. They aim to contribute to current research discourses in the fields of 'Human-Computer-Interaction' and 'Computers and Society'. The 'international reports on socio-informatics' appear at least two times per year and are exclusively published on the website of the IISI.

Impressum

IISI - International Institute for Socio-Informatics
Heerstraße 148
53111 Bonn
Germany

fon: +49 228 6910-43

fax: +49 228 6910-53

mail: iisi@iisi.de

web: <http://www.iisi.de>

An End User Development Environment for Culturally Contextualized Storytelling

Marcos Alexandre Rose Silva, Junia Coutinho Anacletoauthor
Federal University of São Carlos. Washigton Luis KM 235, São Carlos, SP,
Brazil

marcos_silva@dc.ufscar.br, junia@dc.ufscar.br

Abstract. This paper describes an environment that users can develop a narrative game as a product, to be used at school by teachers, considering students' culture expressed in common sense knowledge, for storytelling, allowing teacher to create, configure, adapt and evolve stories according to student's social economics and cultural reality and use a common vocabulary. Consequently, teacher enables them to identify and get interested in collaborating with the teacher and other students to develop the story, being co-authors. These stories created can be considered as a product of the narrative game software. Although building these products, teachers can also monitor the children's learning process for elaborating their experiences, being able to support them and make interventions when necessary, promoting a safe and health student's development.

Keywords: Narrative Game, Context, Common Sense, Education.

1 Introduction

School ambient is very important to the children's intellectual and socio-cultural growth. At school children can expand their interpersonal, cognitive and linguist skills. Therefore the quality of relations established in school, specially in children's education, can affect their learning and development. Because of this,

the relationship in the school among students and between students and teachers is very important.

On the other hand, teachers could have difficulty to promote these interactions; some times they do not have support to promote these interactions. In this context this paper describes a narrative game that intends to support the teacher through storytelling to interact with their students. The teachers are co-authors of this game, because they can configure, adapt and evolve the stories by themselves, and they can tell stories using culturally contextualized information that are displayed to them according to their needs during the narrative.

This paper is organized as follow: section 2 the game's prototype is presented; on section 3 describes about the use of common sense knowledge in the narrative game; and last section 4 discuss some conclusions.

2 Contexteller

According to Piaget (1999), games are directly related to the child's development. There are several types of games and each one of them has characteristics that help the child's physical and mental growth. Overall, games can be classified as (ANACLETO *et al.*, 2008): recreational, cooperative, educational and narrative. Fantasy in narrative games allow people, especially children, to feel safe to express themselves because they believe that what happens in fantasy has little or even no consequence in real life. According to Oaklander (1988) children do things, behave and move in their fanciful world in the same way in their real world. Because of that narrative games for their free expression and support to formule experiences are useful.

The narrative game proposed in this paper, Contexteller - storyteller contextualized by Common Sense knowledge, is inspired in Role-Playing Game – RPG (BITTENCOURT *et al.*, 2003). Like RPG, the game presented has as participants, the master who usually is the most experienced player and his task is to present a story to a group, with characters, their characteristics, scenarios to other participants, who are the players. These are not just spectators; they contribute actively in the story through their characters that choose paths and take on own decisions, and most of the time not foreseeing by the master, contributing to the spontaneous and unexpected development of the story. In the context work the master is the teacher who introduces the story and intervenes collaboratively with the players. The players are the students, the co-authors of the narrative.

For this game, it was considered a group of children from 8 to 12. Piaget (1999) describes that during this stage children are willing to make friends and want to participate and interact with other children's game. Therefore, there are great chances that the children can be interested in participating and interacting with the story being told collaboratively

Figure 1 shows the interface available to master. This interface allows the masters to see their card (I), dice (II), and text area (III), which allows them to read all the messages sent to them and other players during composition of the collaborative story. In area (IV) presents the common sense card and area (V) shows the cards of players.

The game has some RPG elements, such as: Magic, Force and Experience. The values of the first and second elements are defined by the players. These elements are considered to be one of the rules existing in RPG. This rule avoids many discussions that could occur during the story. For example, knowing what is the strongest or most powerful character (FERNANDES, 2008). The values of the elements are numbers to be considered in some situations. For example, a character with Force equal 5 is more likely to survive a crash than a character with Force equal 2. The master attributes the value of the Experience when the character achieves a particular goal stipulated during the development of the story, in short, dynamically.

This game allows teachers to tell their stories considering their pedagogical goals. It also intends to give computational support for the master to get help from contextualized information, both in the initial phase, i.e., the composition of the scenario and the characters to be presented, as well as in others phases, such as: story definition and sequence. This support is obtained using a common sense knowledge base that represents cultural aspects of students' community.

3 Use of common sense knowledge in the Contexteller

The game proposed uses the common sense knowledge obtained by the Open Mind Common Sense in Brazil Project (OMCS-Br), developed by the Advanced Interaction Laboratory (LIA) at UFSCar with Media Lab of Institute Massachusetts of Technology (MIT) collaboration.

In the project, it has been collected common sense of a general public through Web site. Common sense is storing in a knowledge base through the representation of knowledge in natural language sentences where it is processed (ANACLETO *et al.*, 2006). In this game, the common sense information is obtained through a card, which is presented on the master's interface (Figure 1). This card allows teachers to use common sense knowledgebase in the story script. Teachers can obtain characters or/and their characteristics through this common sense knowledge.

The card objective is to support teachers on knowing what students know about some story or even events, cause and consequences. And teachers can use this information to conduct the stories. In short, story definition and sequence. Because of this, players can feel connected with the story characters,

characteristics, scenarios and language that teacher defined with common sense knowledge help. Therefore, Contexteller does not teach common sense to teachers, but help them to know what students' knowledge about stories, because teachers already know about common sense.

For example, if the teachers want to use in the story a character that likes to joke and trick, they can through the common sense knowledge base obtain the following characters: Saci-Pererê, Iara, Curupira, Caipora (from the Brazilian folklore) and Joker (from Batman's), among other. Teachers also can get the characteristics for the characters or something that they want to include into the story. For example, some characteristics coming up from Iara's character are: a mermaid, long hair, beautiful, fish tail. Teachers can join such information with the story that they want to tell and to define the characters and their profiles, personalities. Players must choose a character to participate in the story.

During the story teachers also have support of the common sense. Figure 1 illustrates a situation where Iara character does not play because she wants to comb her hair and master continuous the story with contextualized information helping.

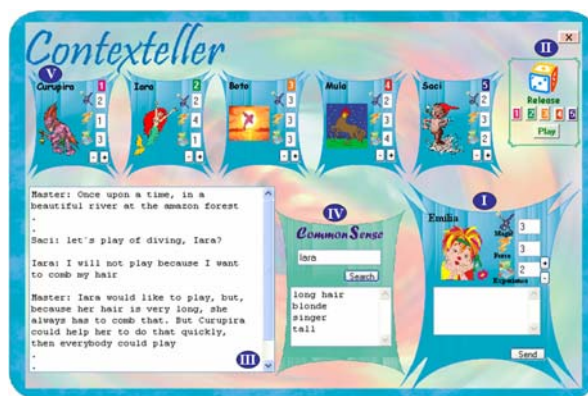


Figure 1. The interface of the Narrative Game.

All the regions in Brazil consider that Iara is a mermaid but in some regions she has different characteristic, such as: short or long hair, brunette or blonde, short or tall, etc. If teachers know the common sense of the specific region, they can tell the story considering the student's reality of that region. For teachers to give common sense information about any Brazil regions, they can select a filter in initial phase. The filter is used to obtain the common sense from a certain community or group of people, like teenagers from Rio de Janeiro in Brazil.

4 Conclusion

This paper describes a environment for online collaborative storytelling, where the players jointly develop a story under the advise of a master (teacher). This

game is meant to support a teacher in interacting with students which have different social and cultural backgrounds. Contextteller intends to allow students to feel closer and identified themselves in the story. Therefore, they can express themselves through the character in their cultural context. They know and identify meanings to the symbolism adopted by the master. These symbols can come from the students' community common sense knowledge to define the character, objects, in fairy tales.

The stories created are products and teachers can use these products to generate various materials, such as: a book, allowing the students to take stories to their home, to show them to their families and friends; to print the stories to students draw and paint their drawing, etc. When the students have a product that they are co-authors, they feel proud of themselves and motivate to participate on other stories creation. Teachers during the stories can also observe how the students lead their characters interacting with other characters. If the character is shy, aggressive, isolated from colleagues or other, these interactive situations can enable the teacher to interpret how the character is being conducted, and then to get to know better the students and their realities.

5 References

- ANACLETO, J. C.; FERREIRA, A. M.; PEREIRA, E. N.; SILVA, M. A. R.; FABRO, J. A. "Ambiente para criação de jogos educacionais de adivinhação baseados em cartas contextualizadas". In: WIE – Workshop sobre Informática na Escola, 2008.
- ANACLETO, J.; CARVALHO, A.; NERIS, V. ; GODOI, M.; ZEM-MASCARENHAS, S.; TALARICO, A. How Can Common Sense Support Instructors with Distance Education? In: SBIE 2006, Brasília. Anais, 2006. v.1. p.217-226.
- BITTENCOURT, R. J.; GIRAFFA, L. M. M. "A utilização dos Role-Playing Games Digitais no Processo de Ensino-Aprendizagem". Technical Reports Series, Number 031, Setembro 2003.
- FERNANDES, V. R. "What is RPG?". RPG - Dragon Magazine in Brazil, n. 123, 2008.
- OAKLANDER, V. "Windows to Our Children: A Gestalt Therapy Approach to Children and Adolescents". Gestalt Journal Press, 1988, 335p.
- PIAGET, J. "Judgement and Reasoning in the Child". Richmond, VA, U.S.A.: Littlefield Adams, 1999, 268p.

Scaffolding Collaborative Project Work in End-User Development

Matthias Korn and Michael Veith

University of Siegen, Germany

matthias.korn@uni-siegen.de, veith.michael@gmx.net

Abstract. In a long term case study, we have analyzed learning practices in a German Computer Club House (CCH) setting. Observing children and their parents creating artifacts with construction kits, we found that they had problems in maintaining the flow of their project work over time. Therefore, we develop concepts for a project management tool which support CCH settings to scaffold their growing information space in terms of artifact re-use and expertise development over time. Scaffolding in this regard is understood to support collaborative processes in communities of end-user development.

Introduction

Today's children grow up in a highly computerized world already being exposed to various media technologies. They consume and even produce YouTube videos, Wikipedia articles and other forms of user created content every day. We strive to support children in their cognitive and social development, i.e. to enable them to understand the world they live in and empower them to form it according to their own conviction. Accordingly, we base our research on project work with active production and consumption of collaboratively created, personally meaningful artifacts. We concentrate on children as a special group of end-users. While we also observed their parents, we think some of these insights will be helpful for this user group as well.

In this paper, we investigate how we can use Vygotsky's concept of scaffolding in a collaborative project setting to support end-user development during all phases of project work and in multiple projects over time. By conducting a qualitative field study we hope to shed some light on this area.

1 Theoretical Considerations and Motivation

Overcoming Paperts (1980) focus on subjective concepts in constructing artifacts, Bruner recognizes Vygotsky's social constructivist concept of scaffolding (Wood et al., 1976; Vygotsky, 1978). With scaffolding, the tutor would offer assistance only with those skills that are beyond the learner's capability to help her master a task that she is initially unable to grasp independently. The tutor then begins with

the gradual removal of the scaffolding, which now allows the learner to work independently (Wood et al., 1976). Scaffolding is based on Vygotsky's (1978) previous idea of the *Zone of Proximal Development* (ZPD). The ZPD is "the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance, or in collaboration with more capable peers" (Vygotsky, 1978, p. 86). The ZPD shifts as the learner has expanded her knowledge and the scaffolding must constantly be adapted to address this change.

Too little attention has yet been drawn to the sustainable long-term support of end-user development processes by scaffolding collaboration in project work. We propose a transition from designing single artifact construction kits to whole frameworks supporting project work over time.

2 Settings and Methodological Approach

The computer club 'come_IN' provides opportunities for elementary school kids, parents, and tutors to engage in group-oriented project work (Stevens et al., 2005; Veith et al., 2007). As described in more detail in Stevens et al. (2005), come_IN is inspired by the Computer Clubhouse concept by Resnick & Rusk (1996) adapted specifically to the German context. The project work within the club stems from the participants' maps of experience. Projects normally last for several months and encompass the programmatic creation of varied multi-media artifacts.

Ideally, all participants take part in all steps of the project work, i.e. brainstorming, planning, execution, wrapping-up, presentation, and reflection. In regard to scaffolding, ICT support should be available in all project phases. Re-use of projects or parts of it in following projects should be a common practice.

Our results stem from an evaluation study in the computer club house. Over the course of six months, we conducted participatory action research by implementing ourselves as tutors in the club collecting information through field notes, observations, interviews, and video and artifact analysis. This mix of methods allows us to collect as much information as possible in order to evaluate a pre-defined goal.

3 Empirical Findings

The identified practice reveals a picture that is different from the ideal situation described above. During the initial collective *brainstorming* phase re-use is rarely occurring. When beginning new projects, participants normally start from scratch building solely upon their prior experience but do not consider previously created artifacts or implemented ideas directly.

Planning is only done by experts, i.e. tutors and some 'old-timer' parents. While children go about their own business, the experts are left alone discussing about the necessary tasks and task distribution at the big table in the club or sket-

ching broader project layouts on the blackboard (Figure 1a). Children do normally lack the patience for longer discussions, but more importantly, they, as many parents, do not always have the insights into the general workings of the club.



Figure 1. (a) Tutors are planning alone while children go about their own business. (b) Mother sitting next to her son, nearly uninvolved throughout the whole session.

The *execution* work is mainly done by children. They voluntarily commit themselves to realize their ideas within the project's scope, as they have chosen the topics on their own. But they often have problems finding files on the network drive or other recently created artifacts to continue their work and stay focused. Parents are often much less involved in the actual project execution. Due to poor integration and personal disinterest, they only sit behind their kids, from time to time giving hints or advice (Figure 1b) or are not present at all. Much less do they show initiative in using computers themselves in activities deeply connected with their child's activities. In general, parents barely take interest in other community members and their activities, only thinking about the progress of their child.

Due to the lack of parents' involvement, tutors are also very much occupied during execution helping all of the children (and also some parents) at the same time. The ICT expertise and club experience of the parents is too limited to help in some cases. In contrast, tutors have a relatively clear picture of the whole project structure, because they are heavily involved in all phases of the project workflow. Due to their high workload, *monitoring* of the overall project progress is hardly ever possible. Tutors do not have the time to coordinate and overview the activities of everyone. The poor monitoring creates additional work in the following *wrapping-up* of artifacts and re-organization, which is also mainly achieved by tutors. They collect the scattered sub-projects and fragmented material and combine and arrange it into the superordinate framework.

Collaboration is mainly initiated without ICT support by the tutors. It is mostly them, who point participants to other members to collaborate on similar issues or projects or to exchange experience, ideas and help, which one party might have already acquired. Though participants collaboratively choose a common topic or share a common experience, they deal with it independently.

This lack of direction in project work and collaboration motivates a need for scaffolding of collaborative project work in the community beyond the scaffolding of the individual mind. To support this scaffolding and enable the participants' involvement in all phases of the project workflow, we aim for a transparent visualization of the network of other participants' related previous work, of their expertise and generally supportive artifacts (e.g. tutorials, related tools). This may help to engage more community members into the planning process and the following phases of the work flow. These additional tools could be seen as a kind of project management software used as the working environment by all participants. It acts as a scaffold to the members giving them contextual support in those tasks that are initially beyond their individual capabilities or knowledge.

4 Conclusion and Outlook

In this paper, we investigated Vygotsky's concept of scaffolding to support collaborative project work in end-user development. We proposed that sustainability (in terms of a growing information space) by providing end-users support in collaborative project work over time is more important than the tools themselves (i.e. construction kits). In our analysis we showed how fostering collaboration by scaffolding orientation and cognitive mapping can be achieved through visualization of artifact and expertise distribution.

Based on our experiences, we showed the Janus-faced nature of scaffolding. On the one side, scaffolding is seen to support the individual mind and thoughts as constructionists use it in artifact construction kits. On the other side, we proposed a scaffolding technique to support collaborative processes of whole communities. Both sides of the Janus face need to be embraced as they can lead to different design implications. Currently, architectural design decisions have been made and the system is being implemented and needs evaluation afterwards.

5 References

- Papert, S. (1980). *Mindstorms*. New York: Basic Books.
- Resnick, M. & Rusk, N. (1996). The Computer Clubhouse: Preparing for life in a digital world. *IBM Systems Journal*, 35(3-4), 431-439.
- Stevens, G., Veith, M., & Wulf, V. (2005). Bridging among Ethnic Communities by Cross-cultural Communities of Practice. *Proceedings of the Second International Conference on Communities and Technologies (C&T 2005)*. Milano, Italy, 377-396.
- Veith, M., Schubert, K., von Rekowski, T., & Wulf, V. (2007). Working in an Inter-Cultural Computer Club: Effects on Identity and Role Affiliation. *IADIS International Journal on WWW/Internet*, 5(2), 100-112.
- Vygotsky, L. S. (1978). *Mind in Society*. Cambridge: Harvard University Press.
- Wood, D., Bruner, J. S., & Ross, G. (1976). The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry*, 17(2), 89-100.

An Outline for a Syllabus for Introducing End-user Type of Students to the Object-oriented Paradigm

Rony G. Flatscher

Wirtschaftsuniversität Wien, Institut für Betriebswirtschaftslehre und
Wirtschaftsinformatik, Augasse 2-6, A-1090 Wien, Austria

rony.flatscher@wu-wien.ac.at

Abstract. This work-in-progress paper sketches a syllabus for introducing end-user type of students at the Wirtschaftsuniversität Wien to the object-oriented paradigm. The knowledge of this syllabus then serves as the fundamental building block for subsequent syllabi for scripting Windows and Windows applications and for scripting Java and Java applications.

Keywords: EUD (End-user Development), EUP (End-user Programming), Object-oriented Paradigm, Syllabus, ooRexx.

Introduction

Working at a University (Wirtschaftsuniversität Wien, WU) where more than 20,000 students study Business Administration and Economics, there are quite a few students who are very interested in Business Informatics/Management Information Systems (MIS) but have either no or poor prior exposure to programming. As today's software infrastructure is heavily based on object-oriented (OO) concepts, it is mandatory to teach students the OO basic concepts and have them apply their acquired knowledge on a regular basis, such that many of the abstract concepts become "tangible" for the solution of (e.g. Business process) problems with the help of a programming language or environment.

This work-in-progress paper introduces the syllabus for teaching WU-students the basics of programming and in the process concentrates on the OO-paradigm. The programming language used in those classes is “Open Object Rexx” (ooRexx, cf. [1, 2, 3, 4]), which can be regarded as a “human centric”, basically typeless, interpreted programming language, which is available for practically all operating systems. It implements all of the most important OO concepts and therefore can be used to demonstrate and experiment with these.

1 Set-up of the Lecture and the Syllabus

This syllabus accounts for two European Credit Transfer System (ECTS) points. The students are set up into groups of two, such that no one is left on his/her own, when creating the assigned (homework) programs. After each installment the students must create two small programs on their own, which each stress some newly introduced concept. The homework has to be turned in via a mail server list such that all other students are able to see and study the homework of their colleagues one day before the next installment of the class takes place, i.e. within seven days. That mail server list is also intended for seeking and giving help among the students (although rarely used for that purpose, probably because students do not want to document in the public that they would not understand some fundamental concept).

As one cannot learn how to swim in a classroom only, it is mandatory that the students “wet their feet” and finally start to learn swimming in the water, it is important for end-users to really apply the theoretically learned concepts with a real programming language. Such a programming language should be easy to learn (i.e. possesses among other things a simple syntax) and easy to debug (i.e. give as helpful error messages as possible and allow for debugging at various levels of detail). Over the course of many years the author “stumbled” over a practically unknown scripting language that was originally created by IBM, Object REXX, that nicely realizes the aforementioned important properties. (That language was donated by IBM to the RexxLA and is now a free and open source scripting language, named “Open Object Rexx (ooRexx)”.) Experimenting with this OO-scripting language at the University of Essen in the beginning of this millennium, and then later at the University of Augsburg and finally at the Wirtschaftsuniversität Wien, yielded a very helpful teaching tool that demonstrates the taught (OO) concepts very nicely. All interested students, Business Informatic students in the case of Essen, and Business Administration students (“end-user-developers/programmers”) alike could master the language, and more important the OO concepts within five weeks à four lecture hours, which has been very remarkable and helpful for the subsequent courses that build on the results of this one.

1.1 Syllabus for the Foundation of Programming

The first part of this course introduces the building blocks of programming, like the definition of a statement, flow-of-control (repetitions, selections, procedures/functions, modules/packages), and the runtime environment, under which pro-grams get executed.

1.1.1 Installment 1

An overview of the course is given, followed by the thoughts that led to picking the ooRexx language as the tool for this course. A brief history of ooRexx is given, pointing out the motivation of creating it in the first place and discussing design goals like “human-centricness” of the programming language, which makes it so suitable for end-user-development/programming.

Confronting the students with a short hello-world program in ooRexx it is explained, that the program is stored as a plain text file and needs to be interpreted by the ooRexx interpreter. In this context the online help system is introduced and explained, stressing the excellent reference PDF documentation that comes with the language.

The students will learn the concepts of a variable, a statement, a block, a branch, and repetition (loop). As ooRexx is not strictly typed there is no need to explain types at this time at all.

1.1.2 Installment 2

The students learn about labels which are used as jump targets and that are needed, if one organizes repetitive code as procedures and functions. In addition built-in functions (BIF) are introduced as well as external programs serving as jump targets for procedures and functions. For this to work reliably, resolution rules need to be de-fined, that are followed by the interpreter.

Creating more complex programs by creating procedures and functions is eased by the concept of a scope, which allows for insulating the variables of a procedure or function from the rest of the program.

This installment continues with the introduction of the concept of associative arrays, dubbed “stem”-variables in ooRexx, and concludes with the keyword instruction “PARSE” which makes it very easy to parse strings into different parts.

1.1.3 Installment 3

In this installment the students learn about the concept “conditions”, “exceptions” and how to intercept them, if the programmer so desires. In addition retrieving arguments by reference within procedures and functions is discussed, making it for the first time explicit that so far only calls by values got carried out.

The concept of “directives” is introduced, which are carried out by the interpreter prior to executing the program. A brief overview of the directives

“requires”, “routine”, “class” and “method” is given, followed by a more thorough discussion of the “requires” and “routine” directives.

1.2 Syllabus for Object-oriented Programming

The second, concluding part of this course introduces the OO concepts of class, sub-classing/specializing, class hierarchy, inheritance including multiple inheritance (!), methods, method resolution (and the role of the variables named “self” and “super”), message (available as first class objects, FCO, in ooRexx, as well as the “un-known/cannot-understand-message”-concept) and some of the most important, common collection classes for the utilities, they offer the programmer.

1.2.1 Installment 4

First the concept of an “abstract data type (ADT)” is introduced and the concepts “at-tribute” and “function/behaviour” are introduced and discussed. Object-oriented programming languages are designed to easily implement ADTs in the form of classes. It is stressed that the OO-paradigm introduces an own set of “termini technici” like “class”, “object/instance/entity”, “inheritance”, and he like.

With the help of simple examples the correspondence between ADTs and their implementations in the form of classes – in ooRexx using the “class” and “method” directives – is repeatedly given. Taking advantage of classes necessitates the knowledge of the concepts “messages”, “cascading messages” and the introduction of additional scoping rules.

The concepts of “constructor” and “destructor” get explained and their effects demonstrated with little nutshell examples.

This installment concludes with the introduction of the concept of a “classification tree” and how it gets used in method resolution, introducing the runtime variables “self” and “super” in this context.

1.2.2 Installment 5

Firstly, the OO-concepts introduced in the prior installment get repeated, followed by a detailed explanation of the “class” and “method” directives which allow for re-iterating the important building blocks.

Following the class hierarchy explanations the concept of “multiple inheritance” is introduced and exemplified with a little nutshell example. (This originally was motivated by reading an interview where the creators of Java thought that this concept is difficult/not understood by developers and therefore error-prone. Every EUD so far was able to understand that concepts without any problems!)

This installment concludes by introducing and characterizing the core class hierarchy that comes with ooRexx, concentrating on the collection classes.

2 Conclusion and Outlook

This article briefly introduced a syllabus for teaching end-user-kind of students (i.e. Business Administration and Economic students) the object-oriented concepts and object-oriented programming with the help of an easy to learn scripting language named ooRexx in a course of two ECTS points.

Building on this fundamental building block it becomes then possible to teach such EUD-kind of students in another two ECTS points class the fundamentals of scripting Windows and Windows applications (even beyond Microsoft Office!) using the infra-structural ActiveX interfaces and Windows script host (WSH). Explaining the ooRexx OLE proxy class for OLE-driven scripts is then a matter of 20 minutes only!

In addition it becomes possible for EUD with the OO-building block knowledge in four ECTS point course to teach scripting of Java and Java applications, which allows for creating operating system and platform independent scripts and applications!

It would not be possible to achieve the same depth of a working knowledge for EUD with a programming language like Java, C++, C#, Perl, Python, or even Visual Basic (or VB.Net for that matter), because of the syntax rules, richness of functionalities and (syntax) peculiarities of those languages.

3 References

1. Cowlishaw, M.F.: The REXX Language. Prentice Hall, Englewood Cliffs (1990)
2. Flatscher, Rony G.: 2006. Resurrecting REXX, Introducing Object REXX. RDL Workshop, ECOOP 2006, Nantes, Frankreich, 3.7.-7.7. (2006). URL (as of 2009-01-19): <http://prog.vub.ac.be/~wdmeuter/RDL06/Flatscher.pdf>
3. Flatscher, Rony G.: Slides “ooRexx_1.pdf” through “ooRexx_6.pdf” introducing the OOo paradigm at the Wirtschaftsuniversität Wien (WU). URL (as of 2009-01-19): <http://wi.wu-wien.ac.at/rgf/wu/lehre/autowin/material/foils/>
4. Fosdick, H.: Rexx Programmer’s Reference. Wiley Publishing, Indianapolis (2005)
5. Homepage of Open-object REXX (ooRexx), <http://www.ooRexx.org>

A Toolkit Method to Match Up End User Needs with Salesforce.com Solutions

Ken Decreus¹, Stijn Viaene², Geert Poels¹

¹ Faculty of Economics and Business Administration, Ghent University, Belgium.

{ken.decreus | geert.poels}@ugent.be

² Vlerick Leuven Gent Management School, Vlamingenstraat 83, 3000 Leuven, Belgium

stijn.viaene@vlerick.be

Abstract. The recent trend in Software-as-a-Service (SaaS) offers the end user ready-to-use software systems via a new delivery model. Market leader Salesforce.com can be seen as the prototypical implementation of SaaS software. One of the key assumptions made by Salesforce.com is that end users do not need explicit support to express their user need information. Therefore, no explicit methodology is offered to combine solution information with the end user's need information to design a responsive Salesforce.com product. We will provide a method for end users to express their requirements in order to discover how the Salesforce.com Sales Force Automation (SFA) application should be configured. By using the toolkit, the end user will discover which part of his needs are covered by means of the SFA application. This way, the user will be able to start his end-user development path for the covered functionality, and use the resulting toolkit models to communicate to other stakeholders what is missing in the Salesforce.com SFA application.

Introduction

The recent trend in Software-as-a-Service (SaaS) offers the end user ready-to-use software systems via a new delivery model. Market leader Salesforce.com can be

seen as the prototypical implementation of SaaS software. One of the key assumptions made by Salesforce.com is that end users do not need explicit support to express their user need information. Therefore, no explicit methodology is offered to combine solution information with the end user's need information to design a responsive Salesforce.com product. We will provide a method for end users to express their requirements in order to discover how the Salesforce.com Sales Force Automation (SFA) application should be configured. By using the toolkit, the end user will discover which part of his needs are covered by means of the SFA application. This way, the user will be able to start his end-user development path for the covered functionality, and use the resulting toolkit models to communicate to other stakeholders what is missing in the Salesforce.com SFA application.

1 Method

1.1 Example Organisational Setting

The sales department of a fictive enterprise in the telecommunications sector called TechCom incorporates a lot of best-practices known, as the sales manager used to work for one of the big consultancy offices. The department is currently led by the sales manager, who reports to the vice-president of Mid-Markets. Two sales teams are working for the sales manager, each team consisting of five account managers. The account managers are the single points of contacts to the account, looking for opportunities to sell products and services and trying to build good relationships with the contacts at the account. When TechCom organizes a big annual event, they ask visitors who want more information to fill in a feedback form. These possible new customers are given to the Sales Manager, who keeps an Excel sheet with possible leads and customers-to-be-contacted. During the weekly conference call with all account managers, the sales manager assigns the new leads to the relevant account managers. As the sales manager has little time to double check these assignments, sometimes leads are getting lost. Furthermore, the sales manager would like to impose a standardised way to qualify a lead, because he suspects that some of his account managers convert leads into contacts too soon to reach their monthly targets.

The sales manager decided that he would save time by automating the current sales processes by means of Salesforce.com SFA, but he feared that the current timing was badly chosen. First of all, the VP Mid-Markets has a deep distrust of hosting sensible customer data at a vendor site. Therefore, he instructed the sales manager to keep the financial customer data locally in the ERP system, and wanted to have a secure bridge between the local and the external SaaS server. Secondly, because the sales manager wants to gain the trust of his boss, he

decided to limit the project scope to the lead management processes; when trust has been established, the sales manager would like to extend the automation scope to opportunity management and account/contact management. Thirdly, the sales manager wants to find a consensus on the requirements needed with his colleagues from other company sites, as sales managers from other country sites are instructed –when implementation was successful– to work on the same system. Finally, due to an unfortunate coincidence, the wife of the sales manager is pregnant and the manager was granted a parental leave for three months after this SaaS automation project. A senior consultant will fill up this gap, but only a few days of knowledge transfer are foreseen.

1.2 Toolkit Method for End User

Research in the area of Management Science suggests that product design done by the product user is far more efficient than innovation by product manufacturers [1]. It is proposed to outsource need-related innovation tasks to the users themselves after equipping them with *toolkits* for user innovation. As an example in the business/IT field, Ricken & Steinhorst [2] propose to empower a business user by considering the Supply-Chain Operations Reference-model (SCOR) as a toolkit for business process innovation. In our research, we propose to use the first phases of the Tropos methodology [3] as a toolkit mechanism. The Tropos project provides a model-driven methodology where *i** models [4] are used to drive the generation of software systems.

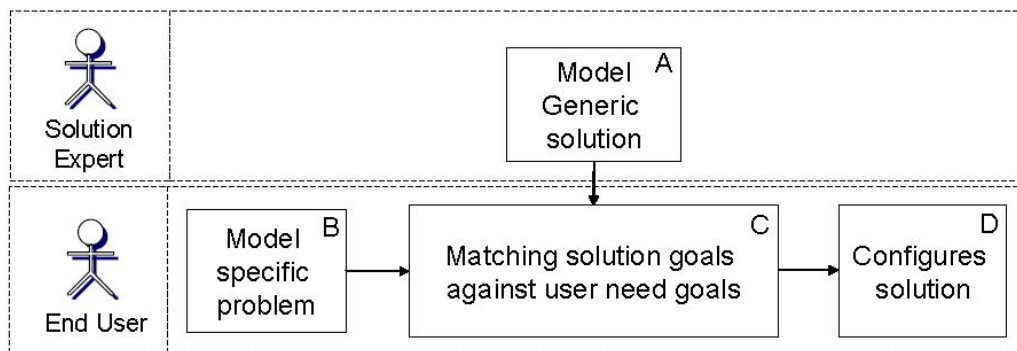


Figure 1. Toolkit method for end user.

We propose a toolkit method for end users, as displayed in Figure 1, where a solution expert creates a model of the generic solution (Part A) and the end user models his specific problem (Part B). Later on, the end user matches both models in order to understand where the solution supports his needs (Part C) and configures the fractions of the solution that supports his needs (Part D).

The *i** modelling framework provides us with different modelling constructs to specify intentionality. A *goal* node in the goal tree shows that there are alternative ways of achieving the goal, but *no specific instructions* are given how to achieve

the goal (e.g. when a car owner enters a repair shop and asks to “just get it fixed”). A *task* node shows that we *specifically* know what to do but there are *constraints* on how to do it (e.g. the car owner asks the repair shop to raise the engine idle settings in order to fix the engine).

Given that the generic solution has been modelled in *i** by an application expert (Figure 2 – Part A), the end user can import this solution model into the toolkit environment. Applied to the TechCom case study, the sales manager models his specific problem (Figure 2 – Part B) using the *i** modelling language.

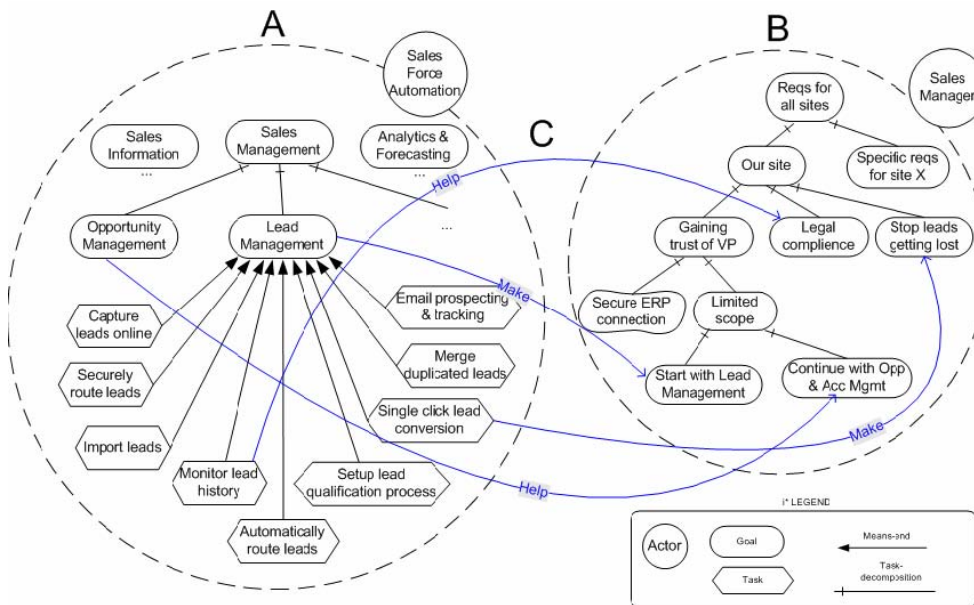


Figure 2. Matching generic solution model with specific problem model

By drawing *contribution relationships* between the relevant goals, the end user is able to express the degree to which the SaaS system supports his goals (Figure 2 – Part C). Note that automated support is needed for helping the user to know which arrows to draw. In the context of non-functional requirements, Castro et al. [3] propose the contribution relationships *help* (partial positive), *make* (sufficient positive), *hurt* (partial negative) and *break* (sufficient negative). Nevertheless, these contribution relationships could also be seen in a more general context [5] where contributions are expressed between both functional and non-functional requirement goals. After matching the generic solution model with the specific problem model, we obtain covered requirements (Figure 3 – Covered zone) and requirements that are not supported by the solutions’ capabilities (Figure 3 – Problem zone).

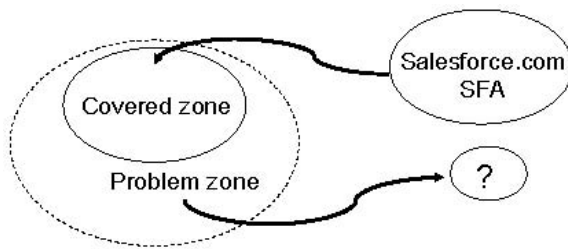



Figure 3. The SFA application may not cover all requirements

In the covered zone, the use case ‘Monitor lead history’ *helps* to enforce the legal compliance, while use case ‘Single click lead conversion’ *makes* the prevention of leads getting lost. Limiting the scope of the SFA application to the lead management module *makes* the establishment of further trust. The problem zone shows the limitations of the SFA application: no secure ERP connection is foreseen, opportunity and account management is planned to install on medium-to long-term, still further trust of VP is needed and legal compliance is not fully supported by a SFA use case.

Finally, guided by the covered requirements, the end user follows the instructions of the SaaS documentation to install the solution that supports these requirements. For instance, configuring the ‘Single click lead conversion’ use case is fully specified in Salesforce.com end user documentation (see Figure 4).



GETTING THE MOST FROM YOUR LEADS

Leads help you track your potential business. Using leads gives you a streamlined way to start your business flow from potential customer to closed deal.

Converting Your Leads

As soon as you qualify a lead, you'll want to convert it into an account, contact, and opportunity so you can start to close the deal.

1. Click **Convert** on the Lead detail page.
2. Verify the owner of the new records and optionally send him or her an automated notification email; set the status of the converted lead; schedule a follow-up task; and click **Convert**. We try to match the new account and contact to existing accounts and contacts if possible. You can choose to use the existing items or create new ones.
3. Activities related to the lead are now associated with the newly created account, contact, and opportunity.
4. Converted leads can no longer be viewed in the Leads tab but they will contribute data to reports. When customizing your report, enter a filter option of "Converted equals True" to view converted leads.

- The **Company Name** from the lead becomes the **Account Name**.
- The **Lead Name** from the lead becomes the **Contact Name**.
- The opportunity and contact you generate are associated with the account.
- If enabled for your organization, universally required custom fields, workflow rules, validation rules, and Apex triggers are all enforced when you convert a lead.

How is a Lead Different from an Account, Contact, or Opportunity?

- An account is the company information for a deal.
- A contact is an individual associated with an account.
- An opportunity is the related deal information.
- Leads can be automatically converted to accounts, contacts, and opportunities.
- Accounts, contacts, and opportunities present a total picture of a customer relationship. A lead is the introduction to that picture.

Figure 4. End User Documentation for ‘Single click lead conversion’ use case

2 Conclusions

Triggered by the fact that Salesforce.com does not provide explicit RE support to end users, we believe that end users provided with RE support could obtain fast time-to-market of SaaS applications. This paper proposed a requirements specification method to allow end user to express their problems in order to select the correct Salesforce.com SFA functionality. Future work will validate our contribution and show the generic applicability of our method.

3 References

1. von Hippel, E., Katz, R.: Shifting Innovation to Users via Toolkits. *Management Science* **48** (2002) 821-833
2. Ricken, A., Steinhorst, A.: Why working with reference models increases process innovation. *BPTrends* **February** (2006)
3. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. *Information Systems* **27** (2002) 365-389
4. Yu, E.S.-K.: Modelling strategic relationships for process reengineering. University of Toronto (1995) 181
5. Markovic, I., Kowalkiewicz, M.: Linking Business Goals to Process Models in Semantic Business Process Modeling (to appear). *EDOC Proceedings* (2008)

Blurring the distinction between software design and work practice

Grace de la Flor and Marina Jirotko

Oxford University Computing Laboratory Wolfson Building, Parks Road, Oxford, UK

grace.de.la.flor@comlab.ox.ac.uk;marina.jirotko@comlab.ox.ac.uk

Abstract. As scientific software is made available within grid infrastructures, EUD increasingly becomes an important activity to support because scientists need to retain a significant amount of control over the code they use to develop experimental workflows and computational models. We present preliminary findings from two case studies where project teams modified the software engineering lifecycle through the implementation of a reconceptualised notion of pair programming as a means in which to facilitate EUD.

Keywords: Empirical Studies, Workplace Studies, Computer Supported Cooperative Work, Requirements Engineering.

Introduction

End-user development has been defined 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 artefact" [1]. End-users can modify software based upon the computing expertise they may have or the type of modification required; "from customization to component configuration and programming" [2]. In this paper, we present findings from our ongoing empirical studies of end-user development (EUD) practices within e-

Research projects. Specifically, we focus on the use of Agile methods [3], and pair programming in particular, as a means in which to facilitate EUD. Thus, enabling researcher communities to take an active role in the design of scientific software.

e-Research is a revolutionary new approach to distributed research collaboration implemented through large-scale, multi-disciplinary, grid infrastructures that support research efforts in the natural sciences, social sciences, the arts and the humanities. e-Research systems include two key technical artefacts; grid infrastructures which process, transport and store data using distributed high performance computing resources, and software applications that assist and extend the ways in which researchers communicate, collaborate and perform work activities. To achieve the e-Research vision, projects initially focused on developing technical solutions to generic technical requirements such as; the design of large federated databases, data compression and transfer techniques, and security mechanisms in distributed architectures [4]. Even when these technical successes are achieved however, in some cases, e-Research applications have not been adopted by their intended user communities as they challenge the conventions and work practices of researchers [5]. Whilst barriers to uptake and challenges to adoption exist for many reasons, including those of large-scale, distributed project management [6], in this paper we explore the ways in which the user-centred design processes has been extended to meet those challenges.

For this research, we are engaged in ongoing ethnographic fieldwork [7] to understand how software engineering practices employed in e-Research projects might influence technology outcomes. We present two case studies where end-user developers collaborate with software engineers to design e-Research applications. In each case study, pair programming features as a key activity which serves to engage end-users directly in the design and coding of scientific software. Interestingly, these projects have adapted the concept of pair programming, described in Agile and XP, from conducting sessions exclusively amongst pairs of software engineers to working in hybrid pairs made up of domain researchers and software engineers working side-by-side in the production of code.

This reconceptualised notion of pair programming, as a means in which to facilitate EUD, challenges traditional notions of the types of activities that might be included within a software engineering lifecycle. It also challenges traditional notions of the user-centred design process as the degree and level of granularity to which end-users participate in the design process is greatly extended. We present preliminary findings of how this new type of hybrid pair programming may contribute to EUD through two case studies; the Cancer, Heart and Soft Tissue Environment (CHASTE) project, which is developing scientific software for computational biology and the UK Network for Earthquake Engineering

Simulation (UK-NEES) project which is developing a system to link together three UK earthquake engineering laboratories to enable real-time, distributed experiments.

1 Using EUD Practices to Identify Requirements and Design for Usability

In each case study, EUD practices were used to both identify system requirements and design software for usability. A hybrid approach to pair programming has exposed the software development process to account for emerging requirements at the level of code. It has also increased the likelihood of software usability as both the scientist and the software engineer work collaboratively to design technical solutions from the practical standpoint of its actual use within the research setting. This may seem like a risky approach to take when projects are under time constraints and include a diverse and geographically distributed group of stakeholders. On the contrary, it gave software engineers hands-on access to the everyday working practices of scientists which has proven to be far more valuable than designing exclusively through specification documents.

1.1 The CHASTE Project

The Cancer, Heart and Soft Tissue Environment (CHASTE) project, is an e-Research project developing scientific software for computational biology. The system provides researchers access to a grid infrastructure where complex models of biological functions can be processed and visualised. The project specifically wanted to assess how agile methods could be incorporated into the software engineering lifecycle. They achieved this by fostering a close collaboration between heart and cancer modellers, who have expertise in the research domain including identifying appropriate algorithms to include in a biological model, and software engineers, who have expertise in the optimisation of code that would be migrated over to the grid infrastructure. The project has reported that using agile methods has been far more effective in the design of software than plan-driven software development methods because [8]:

- It enables quick integration of new members typical in academic projects
- The quality of code increases through the sharing of different types of expertise
- It encourages rapid code development using short timeframe iterative cycles based upon user stories
- It enables the design of adaptable and extensible code that can be modified as requirements change based upon scientific discoveries in the field

Pair programming has provided a forum in which cancer and heart modellers, with the assistance of software engineers, can develop the code base together. The project has adapted the concept of pair programming so that it could be made useful for dispersed project members through the introduction of the peer review of code. Peer review allows software engineers to comment on the robustness of code initially developed by the cancer and heart modellers. It also provides researchers opportunities to better understand and extend the capabilities of their code for specific research purposes. Perhaps most importantly for scientists, hybrid pair programming has meant that the project has been more responsive to changes within the research domain as both technology and the science progress. The fragment of interaction below provides an example of the different types of expertise required to produce both scientifically meaningful and computationally efficient code.



Figure 1. Referring to a journal article (left image). Consulting with a colleague (centre image). Turning to the code (right image).

In this example different experts are consulted at different times. As the mathematician and software engineer read through a biology journal article clarification was needed in order for them to better understand the mechanics of cell division. In this case, an expert, a co-author of the paper, was onsite and could instantly clarify their query. This demonstrates the speed in which requirements can precisely be identified so that the appropriate software could begin to be developed that afternoon. Having access to different types of expertise as the code is written provides project members with opportunities throughout the day in which to query each other. This has resulted in the design of software that more accurately reflect end-user needs and where common understandings about the system's purpose and functionality is achieved more quickly. If the software engineers were asked to rely solely on written materials such as a specification document or journal article the software may take longer to produce and it may need to be re-coded so that it matches scientists' requirements accurately.

Within computational biology the design of computer models, sharing visualisations and the analysis of large datasets have become central activities that support scientific research [9]. In these circumstances, end-user programming increasingly becomes an important activity to support within the project lifecycle. Currently, scientists retain a significant amount of control over the code that they use for research purposes. This software has, up until now, run on single desktop

machines or local systems. However, as locally produced and routinely modified programs migrate over to distributed grid infrastructures they require a degree of re-coding in order to work efficiently within them. As e-Research projects support this transition, hybrid pair programming has proved effective for both scientists who want to retain a working knowledge of the code and software engineers who are interested in implementing efficiently systems.

1.2 The UK-NEES Project

The UK Network for Earthquake Engineering Simulation (UK-NEES) project is developing a system which will link together three UK earthquake engineering laboratories in the UK to enable real-time, distributed, hybrid¹ earthquake experiments [10]. In this project, civil engineering researchers work closely with software engineers to optimise existing code that will need to run in a distributed architecture and to develop new software that will give users access to a 'virtual lab'. When a researcher was asked how he is involved in the process of gathering requirements he replied:

It's difficult, I am the end-user but I'm also probably one of the main people involved in actually setting the whole thing up (UN03a-08).

From the researcher's point of view "setting the whole thing up" includes developing code to meet the project's requirements and designing a usable application that other researchers, not involved in the project, could operate. Earthquake engineering researchers routinely produce code as part of their research process. For example, in order to conduct a hybrid earthquake test an experiment must include both a numerical model of a structure and a computationally produced workflow that will execute and record the experimental procedure. Earthquake experiments tightly couple the computer system with experimental procedure and so domain researchers expect to be able to modify software features for their research purposes.

In the UK-NEES project researchers developed storyboards and incorporated pair programming practices into the software engineering lifecycle. Storyboards have been used to communicate the requirements of researchers through diagrams which represent work processes and experimental workflows. The researcher states that:

I kinda wrote this [requirements] down for him [software engineer], in a pictorial form similar to the scribbles you saw there. [He] knows what I'm looking for and then [he] will go and hopefully program a portal up like that (UN03b-08).

This practice is similar to the agile methods technique of developing 'user stories' where 'customers' produce short descriptions of how they would like the system to function [3]; the customer in this case being the scientist. Additionally,

¹ Hybrid' refers to the design of an experiment that consists of both a numerical model and a physical specimen.

hybrid pair programming has facilitated close collaboration between domain scientists and software engineers.

For the critical parts then we both sit together and I say; 'This is what needs to be done' and 'We should kind of program it in this way' ... cause [the software engineer] doesn't have an idea about how the test should work. He doesn't know anything about the civil engineering side of things ... I put my suggestion down and he'd put his suggestion as to how you can maybe improve that and then once we get it into a format that works then [the software engineer] will code it and I'll have a look at the code [and] if it is doing what it's supposed to be doing then it's good and we'll use it. (UN03c-08).

This quote is a good example of how the domain researcher and the software engineer share their expertise with each other to produce code that is both efficient to run in a distributed system and relevant to research purposes. Interestingly, when the domain researcher was asked if he would call this type of collaboration 'pair programming', he indicated that he had never heard of such a practice and that he had no knowledge of 'agile methods'. In this case, agile-like practices emerged naturally from within the project unlike the CHASTE project where project members had a specific interest in evaluating agile methods.

2 Discussion

In both case studies, project teams modified the software engineering lifecycle through the implementation of a reconceptualised notion of pair programming as a means in which to facilitate EUD. The case studies also demonstrate how the user-centred design process was extended so that end-users could contribute the appropriate degree and level of granularity to the design of software. While each project has transformed traditional notions of the ways in which to manage large-scale software development projects, they have also brought to the forefront the requirement that applications must support EUD practices long after these large-scale systems have been embedded into the research communities that use them.

As scientific software is made available within grid infrastructures, EUD increasingly becomes an important activity to support because scientists need to retain a significant amount of control over the code they use to develop experimental workflows and computational models. However, their primary objective is to conduct domain-specific research. In such circumstances, the challenge becomes one in which researchers maintain an appropriate balance between the time they spend coding and conducting research. Introducing hybrid pair programming as means to facilitate EUD may provide a solution to this challenge.

Our preliminary findings are of particular interest to us as requirements engineering researchers as we investigate new approaches to designing and

managing e-Research applications for usability¹. Usability, has traditionally been defined as a quality that can only be evaluated once a system has been developed and where interface design guidelines can be used to determine measurable characteristics of usable systems [11]. Our findings suggest the contrary, that usability is an emergent property which cannot be located as something that is built into software but rather as something that can only be found in the emergent practices of end-users [cf. 12] as they interact with software artefacts. Hybrid pair programming also provides us with an extended notion of Participatory Design [13] where domain researchers engage as active participants in collaboration with software engineers; working side-by-side in the production of code. Such close collaborations between scientists and software engineers also foster informal communication and cooperation, two factors attributed to bringing about the design of usable software [14].

In future research we intend to conduct a comparative analysis of the organisation of software engineering practices across projects [cf. 15] that use hybrid pair programming as a means in which to facilitate EUD. We will examine how such partnerships increase the usability of software and include analysis of similarities, differences and issues.

3 References

1. Lieberman, H. et al. (eds.) End User Development. Springer, London (2006)
2. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-Design: a Manifesto For End-User Development. Communications of the ACM, Vol. 47(9), pp. 33--37 (2004)
3. Cockburn, A.: Agile software development: the cooperative game, 2nd edition. Addison-Wesley (2007)
4. Hey, T. & Trefethen, A.: e-Science and its Implications. Phil Trans. Royal Soc., 361. pp. 1809-1825 (2003)
5. Bos, N., Zimmerman, A., Olson, J., Yew, J., Yerkie, J., Dahl, E., Olson, G.: From shared databases to communities of practice: A taxonomy of collaboratories. Journal of Computer-Mediated Communication, 12(2) (2007)
6. Lloyd, S. & Simpson, A.C.: Project management in multi-disciplinary collaborative research. In Proc of International Professional Communication Conference (IPCC). Limerick (2005)
7. Randall, D., Harper, R., Rouncefield, M.: Fieldwork for Design. London: Springer-Verlag (2007)
8. Pitt-Francis, J. et al: Chaste: using agile programming techniques to develop computational biology software. Phil. Trans. R. Soc. A, 366, pp. 3111--3136 (2008)
9. Carusi, A, Jirotko, M.: Parameters and visions: dataflows in computational and mathematical biology, The Oxford e-Research Conference. 11-13 September 2008. Oxford, UK (2008)
10. Ojaghi, M. et al.: Grid Based Distributed Hybrid Testing. Procs of the UK e-Science All Hands Meeting, Nottingham, UK, pp.190--196. (2007)

¹ Embedding e-Science Applications - Designing and Managing for Usability project. Grant No. EP/D049733/1.

11. Shackel, Brian (ed.): Man-Computer Interaction: Human Factors Aspects of Computers and People. The Netherlands, Sijthoff and Noordhoff Publishers (1981)
12. Zemel, A., et al.: "What are We Missing?" Usability's Indexical Ground. JCSCW, 17. pp. 63--85 (2008)
13. Bødker, K., Kensing, F., Simonsen J.: Participatory IT Design Designing for Business and Workplace Realities. MIT Press, USA (2004)
14. Nørbjerg, J & Kraft, P.: Software practice is social practice. In: Y. Dittrich, C. Floyd and R. Klichewski, (eds), Social Thinking-Software Practice. MIT Press, USA (2002)
15. Button, G and Sharrock, W.: Occasioned practices in the work of software engineers, In: Jirotko, M. & Goguen, J. (eds) Requirements engineering: social and technical issues. pp. 217--240. Academic Press, San Diego, CA (1994)