Using Extreme Characters to Teach Requirements Engineering

Claudia Iacob
University of Portsmouth
Portsmouth, UK
iacob@port.ac.uk

Shamal Faily
Bournemouth University
Poole, UK
sfaily@bournemouth.ac.uk

***Abstract*—One of the main challenges in teaching Software Engineering as an undergraduate course is making the need for software processes and documentation obvious. Armed with some knowledge of programming, students may feel inclined to skip any development phase not involving coding. This is most pronounced when dealing with the Requirements Engineering practices. In this paper, we describe a practical approach to teaching Requirements Engineering using Extreme Characters. The exercise aimed to achieve the following learning objectives: a) understanding the need of including the end user in any requirements analysis phase, b) identifying the requirements engineering phase as a iterative process, c) understanding the necessity of constantly double checking the analysts interpretation of the user requirements, d) ensuring the rigorous documentation of both user and system requirements, and e) identifying the place of requirements engineering in the overall development process, and the forces and challenges around this phase of development.**

***Index Terms*—requirements elicitation, specification, verification**

I. Introduction

Software Engineering entails the application of engineering practices to the development of software. However, to undergraduate students, the relevance of these practices might seem lost. Many of the software tools they might use to develop software implicitly assume agile practices where the code is the design. Because many students learn to write software on their own, and this software does not scale to the size of the poorly designed and hard to maintain software systems found too often in practice, they can fail to see the value that software engineering can bring. Case studies [1] can be effective at describing the human implications of poor software engineering practices, but can feel removed from the specification exemplars that students work with in the classroom. It is, therefore, important for students to have visceral experience of software engineering practices and why, although often difficult to apply, their benefits outweigh their costs. Requirements Engineering is an important element in Software Engineering, but the elicitation, analysis, specification, and validation of requirements requires an array of technical and non-technical skills that, at times, can seem remote from the practice of writing software. However, given the impact that poorly specified requirements can have, there is a need for students to not only gain visceral experience of key requirements engineering practices, but for this to be put into context with other software engineering practices they also learn.

In this paper, we present a practical approach for teaching Requirements Engineering through the vehicle of Extreme Characters; these were introduced by [2] to display exaggerated emotional attitudes. These exhibit character traits that otherwise might remain hidden, and add an extra dimension to the challenge of eliciting requirements that meet their expectations. We describe the related work in Requirements Engineering education in Section II, before describing the design of a teaching session using our approach in Section III. We evaluate the results of applying our approach in Section IV, before concluding and discussing the implications and limitations of our work in Section V.

II. Related Work

The need to apply requirements engineering techniques to real world problems has been a consistent theme in the Requirements Engineering education literature [3]. However, exposing students to such problems allows them to apply elicitation, analysis, specification, and validation techniques to non-trivial problems. This can be challenging given that solving real-world problems takes time, and any teaching needs to be completed within reasonable time and resource constraints [4]. Software engineering modules typically include some element of teamwork, which introduces students to non-technical issues associated with Requirements Engineering. However, some of the issues relating to Requirements Engineering are not so easy to impart. For example, Gnatz et al. [5] found that requirements produced by student teams were poor quality due to difficulties understanding requirements templates, and it was hard to teach students to write good documentation, or impart the implications their work might have on software maintenance.

Zowghi et al. [6] suggests the key elements of Requirements Engineering education are interview skills for requirements elicitation, analysis and modeling skills for problem solving, and writing skills for requirements specifications. They illustrated how the use of role-playing can impart these skills, by breaking students into teams simultaneously playing two roles. The team acted as a development team responsible for eliciting, analyzing, and specifying requirements; they also acted as a customer team providing information about their needs to a different development team. Role-playing developed

2377-570X/17 $31.00 © 2017 IEEE
DOI 10.1109/CSEET.2017.25

empathy within students as it allowed them to view the same situation from different perspectives. However, as this exercise ran through the teaching of an entire unit, coordinating and monitoring communications between teams was a burden.

III. Requirements Engineering Session Design

We designed a practical session around Requirements Engineering to address the main misconception students have about requirements. As they had little exposure to real-world development processes, students interpretation of this phase of development was reduced to a limited, minimal form of documenting the overall aim of the system under development, specifying its core features in broad terms, and possibly hinting to elements of the systems user interface. The lectures take on requirements – software engineers do not write requirements based on their own understanding of the problem, but they elicit, specify, and validate them based on feedback from the target end-users of the systems developed – led most students to believe that requirements elicitation can be done by means of remote surveys. Although familiar with the theory behind more in-depth requirements elicitation techniques, such as interviews and focus groups, students failed to see their applicability in the software development process. The practical session was designed to address this misconception and help the students achieve a number of goals: 1) Understand the need of including the end user in any requirements analysis phase, 2) Identifying the requirements engineering as a iterative process, 3) Understanding the necessity of constantly double checking the analysts interpretation of the user requirements, 4) Ensuring the rigorous documentation of both user and system requirements, and 5) Identifying the place of requirements engineering in the overall development process and the forces and challenges around this phase of development. Students were asked to pretend they are part of a mobile app start-up company. Their investors see an opportunity for developing a diary and scheduling app for a very influential, but very secretive group of users: the extreme characters. As part of the development team, they were given access to representative users. The exercise was further organised into four steps: familiarisation, elicitation, specification, and validation.

A. Familiarisation

Students were split into two groups (maximum 10 students in each group), A and B. Each group was given a brief description of the target end-user they represented, namely the queen and a double agent. The descriptions provided for each extreme character are available in Table 1. Acting as their respective target end-users, the groups were asked to agree on the functionality they expected from the app. They were all reminded that they are playing a role and that they are not allowed to reveal who they act as. By doing this, we wanted to avoid issues with bias, i.e. students thinking they understand what their extreme character's user requirements are.

B. Elicitation

Each group was further divided into 2 sub-groups: A1 and B2 were the interviewers, and A2 and B1 were the interviewees. Interviewers acted as requirements analysts, while interviewees acted as the user they represented. Group A1 interviewed group B1, while group B2 interviewed group A2. Instructions on interviewing were provided to all groups.

C. Specification

Students returned to their initial A and B groups. Each group had to specify the requirements elicited at the previous step. Students were free to use any requirements specification technique they felt was best suited, but they had to make sure the entire team agrees on the specifications.

D. Validation

Students returned to the groups formed for the Elicitation step, namely A1, A2, B1, B2. This time, A2 and B1 acted as interviewers, and A1 and B2 acted as interviewees. Interviewers acted as requirements analysts validating the requirements specified, while interviewees acted as the user they represented. Group A2 interviewed group B2, while group B1 interviewed group A1.

IV. Evaluation

We evaluated the session design over 10 one-hour tutorial sessions scheduled as part of an undergraduate *Introduction to Software Engineering* unit. The total number of students involved in all the sessions was 86, and each student participated in a single session. An instructor was present at each session. At the end of each session, students answered a set of questions designed to evaluate the effectiveness of the exercise. These questions were: (1) What have you learned about requirements engineering as a result of this exercise?, (2) What aspects of the exercise did you most appreciate?, (3) What aspects of the exercise did you find challenging?, (4) What suggestions would you make for improving this exercise. In this section, we discuss the main findings of the evaluation based on the answers provided by the participating students.

A. What have you learned from this exercise?

1) The process is harder than it looks: Most students concluded that Requirements Engineering is a lot more than sitting down and deciding what user want from the system being developed – *"A lot more complicated than you first assume"* or *"I have learned that requirements elicitation is a much deeper process than first thought, including multiple user meetings"*. They understood that Requirements Engineering is a process, and that the user is a central part of this process – *"Requirements elicitation is very difficult and requires multiple and varied interactions with end users"*. Most importantly, though, students realised that Requirements Engineering, and requirements elicitation in particular, is a considered, iterative process as opposed to a one-off meeting with a user – *"It is an ongoing process not a single step process and it will take at least a few iterations to be able to have a clear idea of what is needed"*.

TABLE I: Extreme characters used in the Requirements Engineering exercise

	Queen	**Double agent**
Profile	The Queen is a person who is very powerful in theory, though in many ways her actions and emotions are prescribed. She is very much restricted by protocol. The Queen sees her formal appointments as tedious, and values her leisure time very highly. She enjoys a stroll in the Hampshire gardens and the conversations with her favourite cousin, Anne. When it comes to her formal tasks, the Queen needs a little encouragement from time to time. The Queen knows that she would fall under media scrutiny if she did not fulfil her formal duties. In a sense, she views her negotiations with the appointment manager to gain leisure time as a kind of game.	The double agent is a powerful person who acts as a spy for both MI5 and KGB. He is highly aware of his place in the MI5 and the KGB hierarchies. Above him in rank are big players who help him coordinate and execute operations. It is a rough world, and in response the double agent has adopted an opportunistic attitude.
Attitude towards appointments	The Queens life is full of compulsory, repeating appointments. In the time slots that remain, the Queen likes to enter as many personal appointments as possible, so that she cannot be hassled for other tasks. While she needs to share the information in her agenda with others who organise her public life, there are some appointments, which she would rather not disclose to others.	The double agent has two agendas, one for the MI5 and one for the KGB. The information in both is very sensitive. It should not fall into the wrong hands, be it colleagues or superiors. Clearly, he is very careful with whom he makes appointments and where. Meeting places are specified by their characteristics. Roads that will allow a quick get-away and buildings that will provide cover are important considerations. The double agent doesnt plan very far ahead. Operations come and go; the scene may look very different next week. The double agent is ambivalent about exposing his appointments. On the one hand, they contain sensitive information. On the other hand, exposing them means enforcing his position in the hierarchy, a kind of power play, which draws new information. In his appointments he needs to express his respect for his superiors, without mistaking them or getting confused about their motives.

2) Users don't always know what they want: One of the most frustrating lessons learned by students revolved around their understanding of the vague nature of users – *"Users are usually vague in particular in specifying details about functionalities of the software"*. They felt that users don't always know exactly what they want from the systems they need and, even if they have a clear idea of what they want, they might not always express it in a straightforward way – *"It is difficult to get requirements from the user and sometimes users dont know exactly what they want"*. Some of the words users use to specify their goals with respect to the system are too vague to be meaningful, making it difficult to assess the exact interpretation users give to these words – *"Sometimes users don't know exactly the functions of the system they want and use words like 'simple', 'intuitive', but they dont specify how to reach these objectives"*. This, they felt, is the reason why requirements can be easily misunderstood.

3) Communication is key: Students realised that, when developing software for a target audience, talking to one user does not provide enough data to support a Requirements Engineering process. It is, therefore, paramount to discuss with a representative number of users. However, this introduced a challenge that the students quickly discovered – *"each user has different requirements and they can change at any time"*. Students noted that even when users have similar requirements, they put different weights on these requirements – *"Similar user bases could have similar requirements but with different priorities"*. The solution, they seemed to agree, comes from better and continuous communication between the user and the analyst – *"Clear communication is key"* or *"Need key communication between client/user and analyst"* or *"It involves a lot of interaction between client/user and the development team"*. It is a process that cannot be rushed, so sufficient resources need to be allocated to it – *"you need to take time with users to discuss their requirements so you have a clearer understanding of what the system requires."*

4) Interviewing is an art: After establishing that communication and continuous interaction with the users is key in Requirements Engineering, students also discussed aspects related to the effectiveness and efficiency of various requirements elicitation techniques. They felt that for best results, one needs to be specific in the questions asked, and know exactly what areas to cover during interviews – *"Asking specific questions can improve the results of the requirements elicitation process"* or *"questions need to be very specific to get accurate information"*. Students also stressed the importance of clarifying and double-checking the information they receive from their users due to the challenging character of the elicitation process – *"It is difficult to find out what the user wants so questions will need to be relatively detailed, and clarification with users is extremely important"*.

With no chance to prepare in advance for the exercise, the students felt the necessity of planning for interviews with users; they felt that the preparation required prior to the interviews should include learning as much as possible about the target users – *"Learn as much about the target audience as possible: their needs, things they like/dislike"*. Students also felt that having the chance to prepare a prototype to show their users, and use it as a basis for the discussion would have helped enormously.

5) Requirement specifications cannot be vague: Specifying requirements was found to be a time-consuming activity, and students felt it required several peer-reviewing sessions to ensure the specification was of acceptable quality. Students

discussed the importance of the requirement specifications being testable – *"requirements need to be testable, they cant be vague"*. Students felt that this is a criterion for ensuring the overall final quality of the specification document.

6) Requirements change: A recurring theme in the students' answers was an appreciation that requirements change all the time. They change depending on the user interviews, but also on the iteration of the elicitation process – *"Requirements change based on who is being interviewed"*. As a result, students felt that validating requirements over several iterations is the only way to ensure their consistency – *"Requirements need to be validated more than once to understand what and how the system needs to be designed"*.

7) Requirements are not features: In addition to learning about the Requirements Engineering process, students also reported learning theory they felt they had not previously encountered. For example, students noted the exercise taught them to differentiate between actual requirements and types of information sometimes mistaken for requirements, such as notes on system functionality or features – *I've learned the differences between requirements and other types of information we get from users"*.

B. What aspects of the exercise did you mostly appreciate?

1) Trying out the process top to bottom: Students appreciated the opportunity of putting in practice some of the theory taught about requirements and the engineering process around them – *"having a practical activity to try the process ourselves"*. Far from the abstraction of the lecture notes and closer to the nitty-gritty of the process as a whole, students found the exercise helpful in teaching the *"the importance of speaking to your users more than once in order to get a clearer idea of how they want their user requirements turned into system requirements"*. Students were also able to differentiate the various phases in requirements engineering, appreciating the role and importance of each phase – *"I liked the validation part as you may misunderstand the user and make wrong assumptions about what they ask for"*.

2) Group work element: One of the things students enjoyed about the exercise was the opportunity for group work. They felt that working with others helped them glean a new perspective on the concept of requirements – *"knowing other peoples' views of requirements"*. It also helped learning about the requirements engineering process from their own peers – *"communicating with other groups, collectively gaining knowledge"*. They felt that the activity was more fun due to working together with other peers, and that the hands-on approach gave the concept of requirements otherwise abstract a concrete and easy to understand definition. Identifying their own requirements, as a group, was another aspect students enjoyed – *"I liked having to talk with others trying to act and imitate the given persona"*. Also, communicating with other teams and trying to understand their requirements was perceived as helpful – *"The chance to speak to others and see what they need"*.

3) Practicing interviewing skills: The practical aspect of helping them improve their interviewing skills was one of the aspects mostly liked about this exercise – *"The interview gave an idea about the right questions to be asking"*. The fact that students weren't given much time to prepare their questions forced them to be spontaneous and come up with questions on the fly – *"Thinking questions on the fly was a good practice"*. In addition to interviewing other peers, students had the chance to experience the role of interviewee, which they found helpful – *"getting a view of what it is like being interviewed rather than just interviewing"*.

4) The confusion of it all: Students did not expect the Requirements Engineering process to be so confusing and the theoretical description of this Software Engineering phase did not strike them as difficult. It was only after they have experienced the process themselves that they realised just how much confusion it can generate – *"the exercise demonstrated the confusion that can arise"*.

5) Interactive, engaging, and fun: Students enjoyed the concept as a whole, and found the exercise interactive, engaging, and fun. They particularly liked the game-like aspect of having to guess the extreme character represented by the other group – *"the game-like approach, felt like 'guess who"'* or *"the mystery of the personas"*. Not knowing the user they are eliciting requirements for made the whole exercise feel different. As one student noted – *"not knowing who the other group represents made it more interesting"*. Eventually finding out who were the extreme characters each group represented helped them understand how misleading the process can be, and emphasised the need for multiple iterations for the elicitation process – *"guessing the persona and how misleading their requirements can be"*.

C. What aspects of the exercise did you find challenging?

1) Eliciting user requirements: The requirements elicitation part of the exercise proved to be the most challenging. Students found it difficult to multi-task, and both think of questions to ask while identifying the requirements provided by their peers, and taking notes of them as they went along – *"listing the requirements when you can't think of what to ask"*. They also felt that this elicitation exercise asks them to be precise when identifying their users requirements and meeting their expectations. Translating user requirements into system requirements was perceived as challenging due to both the complexity of the exercise, and the limited time available. The starting point of the exercise was, perhaps, the most difficult; students noted that getting the initial set of requirements was the most difficult step. Once an initial set of requirements was gathered, the discussion around them led to more data being collected, and the process running a lot smoother. However, eliciting a larger number of requirements eventually led to conversations around the priorities of each, and agreeing their criticality was difficult.

2) The right questions to ask: For some students, this was the first time they practiced their interviewing skills, which posed a few challenges. They found it difficult to find the right

questions to ask or how to phrase the questions they wanted to ask – *"Not knowing what sort of questions to ask"* or *"Not knowing how to phrase questions in order to get an answer that is useful in determining user requirements"*. Not having time to prepare their questions and having to come up with questions on the fly only added to the difficulty.

3) Not knowing who the user is: While not revealing the extreme character they represented brought a game-like feel to the exercise, it also represented one of the aspects students most struggled with. On one hand, answering their peers' questions without revealing too much information about the character they represent (making it easier for their peers to guess it) was perceived as a challenge – *"coming up with requirements without giving away the job"*. On the other hand, guessing who is the user they are eliciting requirements from and using that information in guiding their interview process was also perceived as an additional difficulty – *"finding requirements when you dont know the individual"*.

4) Understanding the concept of requirements: Using the concept of requirements in a rather intensive exercise proved to be time consuming. Students reported struggling with differentiating between types of requirements, i.e. user, system, functional, and non-functional requirements – *"it was confusing because of the varieties of requirements"*. Some types of requirements proved to be more difficult to comprehend than others – *"the difficult thing was writing about non-functional requirements"*. Moving from user requirements to detailed system requirements specification was an additional challenge – *"defining the different requirements further"*. The starting point of the exercise felt daunting with students feeling overwhelmed by the number of phases the exercise had, and by the dynamics of the groups – *"understanding what we needed to do at the start, it took a few minutes to understand what was required of us."*

5) Communication breaks: In some cases, students found communicating with each other without revealing the extreme character they represent problematic– *"conveying information between people"*. This, however, was a problem for very few students.

D. Suggestions for improving this exercise

Several students suggested replicating similar exercises for other software engineering topics, appreciating the interactivity of the session – *"Sessions like this should be done more often."* Given the struggle they experienced with identifying questions to ask during the interviewing process, students suggested being provided with a sample of example questions, or guidelines on coming up with questions – *"maybe some sort of question asking framework or suggested questions."* Students also asked for more details on the brief provided and more guidelines on the overall process followed for future sessions – *"provide an explanation or example at the start to make it clearer what we need to do"* or *"the brief laid out a minor problem but more details would have been appreciated."*

Because the exercise was over a single session, some students felt pressed for time, and suggested extending the sessions, or splitting them into two – *"It needs to be in a longer session as I felt it was rushed"*. The distribution of students in sessions was not even, with the smallest number of students in a session being 4 and the largest being 20. This led to comments on the number of students in each group, and the number of groups in general, and the time spent moving around for the Elicitation and Validation steps; this added to the pressure students were already under.

V. Conclusions

This paper presented an approach for teaching Requirements Engineering through a role-playing exercise using Extreme Characters. In doing so, we have made two contributions. First, we have contributed an exercise design that fuses ideas from Requirements Engineering education (role-playing) and Interaction Design (extreme characters). This exercise design has recently been extended to support the teaching of Security Requirements Engineering at Bournemouth University; the results of this study will be presented in future work. Second, we have presented and discussed results that cast light on the challenges teaching practical Requirement Engineering. The results of the evaluation reflected a significant change in the students perception of requirements and the processes of eliciting, specifying, and validating such artifacts. Based on the lessons learned, we are considering incorporating a few changes in the Requirements Engineering session described in this paper. We will provide a set of guiding questions to get the exercise started. We acknowledge that the lack of notice, and the limited amount of time students were provided with made the exercise challenging. It forced students to focus more on the questions to ask and completing the exercise rather than the core concepts and the process in general. This, however, can be addressed by enhancing the session with critical reflection on the process followed. Also, in future iterations, we will dedicate more time to the session. Splitting the exercise into two sessions might not have the same effect, as the interruption might distract students.

References

[1] N. G. Leveson and C. S. Turner, "An investigation of the therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, July 1993.

[2] J. P. Djajadiningrat, W. W. Gaver, and J. W. Fres, "Interaction relabelling and extreme characters: methods for exploring aesthetic interactions," in *Proceedings of the 3rd conference on Designing Interactive Systems*. ACM, 2000, pp. 66–71.

[3] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, "Requirements engineering education: a systematic mapping study," *Requirements Engineering*, vol. 20, no. 2, pp. 119–138, Jun 2015.

[4] R. J. Barnes, D. C. Gause, and E. C. Way, "Teaching the unknown and the unknowable in requirements engineering education," in *Proceedings of the 2008 Requirements Engineering Education and Training*, 2008, pp. 30–37.

[5] M. Gnatz, L. Kof, F. Prilmeier, and T. Seifert, "A practical approach of teaching software engineering," in *Proceedings of the 16th Conference on Software Engineering Education and Training*. IEEE, 2003, pp. 120–128.

[6] D. Zowghi and S. Paryani, "Teaching requirements engineering through role playing: lessons learnt," in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, 2003, pp. 233–241.