A Feature Modeling Approach for Domain-Specific Requirement Elicitation

Olga De Troyer and Erik Janssens
Dept. Computer Science
Vrije Universiteit Brussel
Brussels - Belgium
{Olga.DeTroyer, erikjans}@vub.ac.be

***Abstract*— In this paper, we presented an approach for domain-specific requirement elicitation. Building domain-specific software requires the expertise of people with very different background and with different levels of experience in software development. This complicates the process of requirement elicitation. The purpose of the approach is twofold. On the one hand, we want to unlock available information on requirement elicitation for particular domains. On the other hand, we want to provide a mechanism for guiding the stakeholders (non-computing as well as computing people) through the requirement elicitation process in these domains. The approach is based on Feature Modeling, a variability modeling technique used in Software Product Lines. Furthermore, a tablet app has been developed to support the approach. We demonstrate the approach for two different domains, the domain of serious games for children and the domain of e-shop web applications. A first evaluation of the approach and the tool has been done by means of two explorative case studies and resulted in positive feedbacks.**

***Index Terms*—Requirement elicitation, Domain-specific, Tool support, Feature modeling.**

I. Introduction

In software engineering, more and more it is accepted that a participatory design [1], where all stakeholders are actively involved in the design process, helps in ensuring that the software meets the needs of the users and will be successful. The needs of the users are formulated during requirement engineering. Requirement engineering consists of a number of core activities: eliciting requirements, modeling and analyzing requirements, communicating requirements, agreeing requirements, and evolving requirements [2].

In domain specific software, i.e., software developed to satisfy certain needs in a particular domain, domain experts and possibly also end-users play an important role, especially during requirement elicitation [3]. Requirement elicitation comprises activities that enable the understanding of the goals, objectives, and motives for building a proposed software system and the requirements that must be satisfied by the system in order to achieve these goals, as well as identifying the system's boundaries [2], [4]. However, domain experts and end-users are usually no software engineers and may not be knowledgeable about computing and requirement elicitation. The stakeholders that we should involve in the requirement elicitation are usually from different disciplines, with different backgrounds, and with different levels of experience in software development.

Different requirement elicitation techniques exist. For interdisciplinary projects, group elicitation techniques, such a brainstorming and focus groups, are often used to foster stakeholder's agreement and exploiting team dynamics to elicit a richer understanding of the needs [2], [5]. However, from our own experience in an interdisciplinary project, Friendly ATTAC (http://www.friendlyattac.be/en/) aiming at the development of a digital game against cyber bullying, we believe that guidance is needed when using these techniques with interdisciplinary stakeholders.

The members of the Friendly ATTAC research project consist of social scientists, health psychologists, computer scientists, and game developers. Requirement elicitation was done by means of plenary meetings and buzz groups involving different stakeholders: educational/youth stakeholders, e-safety stakeholders, heath promotion stakeholders, and professional game developers. Although these sessions were quite successful in generating a lot of interactions and issues to consider, in retrospect, we had the feeling that the meetings could have been more efficient if there would have been more guidance. Because of this lack of guidance, we were also not sure if all relevant aspects that could influence the success of the software (i.e., the serious game) had been taken into consideration. Furthermore, we also observed a communication gap between domain experts and software engineers, as also reported by other researchers (e.g., [5]).

Because the production of high quality requirements through effective elicitation is absolutely essential for the engineering of successful software products [5], we decided to investigate whether a domain-specific approach to requirement elicitation could overcome some of the problems experienced. The approach proposed is based on feature modeling, an approach used in variability modeling [6] to express the common and variable features in variable software, as well as dependencies between features. We also developed a lightweight tool to support the approach. It assists interdisciplinary teams (consisting of the stakeholders with different backgrounds, as well as software engineers) in the requirement elicitation, allows optimizing the communication, as well as making the requirement elicitation process more effective and less time consuming.

The rest of the paper is structured as follows. Section II formulates the objectives of the approach and Section III presents background. Section IV explains the approach and Section V discusses the tool developed to support the approach and the evaluations performed. Section VI discusses related work and Section VII concludes the paper.

II. Objectives

For specific domains, such as serious games (i.e., games for a purpose other than pure entertainment) or web systems, a lot of relevant information on how to develop an application is already available. In general, the problem is that this information is not readily available; it is usually scattered over different sources or locked in the head of experienced people. Software companies that are specialized in the development of software for a certain domain may dispose of this information but usually not in an explicitly form; their method for requirement elicitation usually originated by experience and the knowledge about how to perform a requirement elicitation process needs to be transferred to new employees by means of trainings or courses. Therefore, the goal of the approach is to provide a mechanism that allows experienced people to capture their knowledge about requirement elicitation in a specific domain. Each domain has its own characteristics and therefore different issues needs to consider during requirement elicitation in different domains. For instance, for the domain of games, the game genre and the reward system are important to consider, while this is not the case for other domains. For some domains, different lists of issues could be possible as different people and companies may use different methods for the requirement elicitation or there could be different types of applications. For broad domains, it will not be possible to come to a very detailed list of issues to consider, but the more specific the domain, the more detailed the list of issues can be made. For example, we could consider making a list of issues for the development of web applications, however it is more useful to subdivide this domain of web applications into different subdomains corresponding with different website genres (such as e-shops, news sites, or corporate website) and create a list of issues to consider for each subdomain.

As argued in the introduction, we should also support the requirement elicitation by making the discussions of the different stakeholders (including the requirement engineers) much more focused and efficient, resulting in more thoughtful software. Furthermore, the participants should be *guided* to consider *all relevant aspects and issues* concerning the requirements elicitation in the given domain. This requires a mechanism that allows expressing which issues need to be considered, which could be considered, and what are the different decisions and possible choices that should be made during the requirements elicitation for a particular domain.

The main objectives for our approach are summarized below. Note that the users of the approach are the stakeholders involved in requirement elicitation: domain experts, end-users, and other relevant parties, but also requirement engineers. The objectives of the approach are:

O1. The users should be able to take the required decisions and provide the necessary information regarding the purpose and characteristics of the application, i.e., the approach should *support requirement elicitation in the context of a particular domain*. Note that other requirement engineering activities such as modeling and analyzing are outside the scope of the approach.
  - O1.1. The approach should *guide* the users through the requirement elicitation process using a *predefined set of issues* to consider, ensuring that all relevant aspects are considered as far as possible.
  - O1.2. The approach should allow distinguishing between *issues required to consider and optional issues* because some issues may not be applicable for the case at hand.
  - O1.3. The approach should allow providing *possible options and alternatives,* whenever possible, for decisions that should be taken. This is necessary as not all users will be aware of all possible options and alternatives. Note that for some issues it may not be possible to provide predefined options and alternatives.
  - O1.4. The approach should allow *indicating the impact of choices*. The choice of an option or alternative may have an impact on the options and alternatives available for other issues, e.g., in educational games the choice for a certain pedagogical approach may limit the choice for the reward mechanisms. It is important to be able to draw the attention of the user on this.

O2. The approach (or its tool) should be *usable in meetings and by people with different backgrounds* (i.e., casual users as well as software professionals).

O3. The approach should be *generic*, meaning that it should be usable for different domains.

III. Background

To model and structure the issues to consider, options and alternatives, and dependencies between options and alternatives we use a particular Feature Modeling technique, Feature Assembly [7]. Feature modeling was chosen because it perfectly fit the requirements for our domain-specific requirements elicitation approach. Feature modeling (e.g., [6]) is one of the most commonly used domain analysis techniques for variability modeling and is used to express the common and variable features in variable software or so-called Software Product Lines [8]. In Feature Assembly, features can be decomposed into more fine-grained features and can be mandatory or optional. It is possible to specify different options for a feature. A cardinality constraint is used to indicate the number of options that can be selected. Also dependencies between features can be expressed. An example of a feature model in the context of software variability is given in Fig. 1. The feature model specifies a Quiz Software Product Line. It models all the features that could exist in a particular Quiz application and how they are related. Features are graphically represented as rectangles. An abstract feature (dotted rectangle)

is used to indicate that different options are possible. The possible options are given by the specification relation (an arrow pointing to the abstract feature), and the minimum and maximum number of options that can be selected is given by a cardinality constraint (notation "min:max"). An abstract feature is a source of variability. Dependencies between features can be specified textual or graphically. Examples of possible dependency types between features are "excludes" (i.e., two features exclude each other) and "requires" (i.e., including one feature requires the inclusion of the other feature).

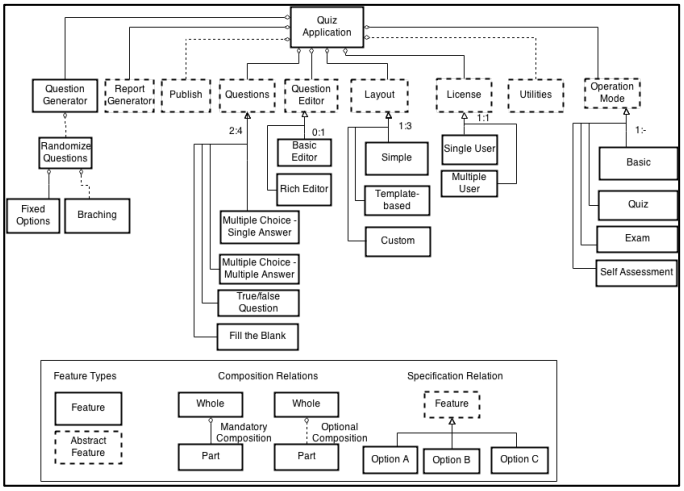

Fig. 1: Example of a (Partial) Feature Model (modified after [7])

A *configuration* of a feature model is a consistent selection of a set of features available in the feature model that describes a particular application (i.e., a member of the Software Product Line described by the feature model). A feature model permits a configuration if and only if it does not violate the relations, constraints, and dependencies imposed by the model.

IV. Feature Modeling Based Approach

Feature models perfectly fit the requirements for our domain-specific requirements elicitation approach, and thus we use this modeling technique to predefine the issues to consider, the options and alternatives available, and the dependencies. However, note that we only use feature modeling as an information modeling technique. We don't use it to define an actual software product line.

As one of the purposes of the approach is to involve non-IT users into the requirement elicitation process (next to requirements engineers), we do not use the term feature. We invented the term "*guidea*" (a portmanteau word for "guided" and "idea") and use this for what is called a feature in feature modeling. The term *GuideaTemplate* is used for the concept of feature model, and the term *GuideaMap* for the concept of configuration.

An overview of the different steps in the process of our approach is illustrated in Fig. 2 (using UML activity diagram). The process is as follows: A requirement-engineering expert creates a GuideaTemplate (i.e., feature model) for a specific domain. This GuideaTemplate can then be used for the requirements elicitation of different applications in the domain. For this, the stakeholders involved in the requirement elicitation of a specific application start from the given GuideaTemplate and create a GuideaMap (i.e., a configuration of the given feature Model). From this GuideaMap a textual document can be generated containing the requirements formulated and the decisions taken. Also a more formal specification (e.g., XML-based) is generated for import into other tools for further processing, e.g., modeling & analysis.

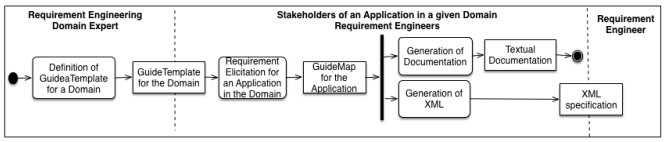

Fig. 2: Process of the Approach

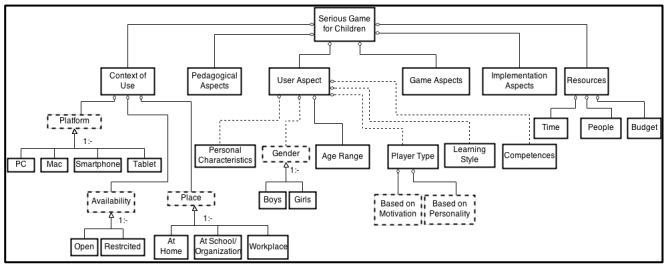

Fig. 3: GuideaTemplate (partial) for the Domain of Serious Games for Children

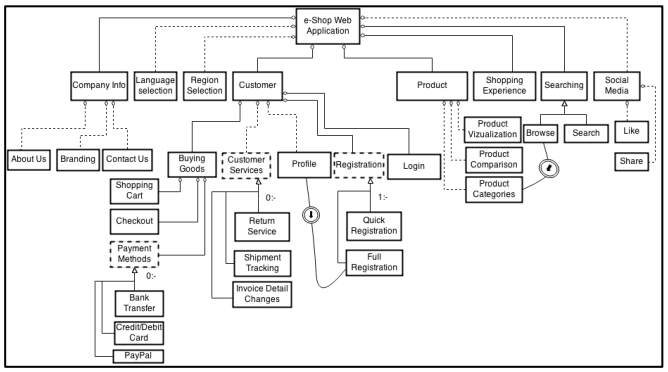

Fig. 4: GuideaTemplate for the Domain of E-shop Web Applications (adapted from [10])

To demonstrate the approach, GuideaTemplates have been created for the domain of serious games for children and for e-shop web applications. A full description of the GuideaTemplate for educational-oriented serious games for children is explained in [9]. Fig. 3 provides a partial view on the feature model (GuideaTemplate) for this domain. The GuideaTemplate was created based on information collected from the literature and during brainstorm sessions with the technological stakeholders of the Friendly ATTAC project, and from our own experience in developing serious games.

Fig. 4 provides the GuideaTemplate for e-shop web applications. The GuideaTemplate for e-shop web applications was developed by a master student in the context of his thesis. To create this GuideaTemplate, the student examined a large amount of e-shop websites to extract the features used in this kind of web applications.

V. Tool Support

To support our approach, we develop a graphical software tool. The tool is a tablet (iPad) app that provides an easy and intuitive interface for the underlying GuideaTemplates, i.e., feature models. We opt for a tablet app, as tablets are easier to use in meetings and less intrusive than traditional laptops, especially when one wants to provide a device to all participants of the meeting. Next to the fact that the tool should support the objectives formulated for our approach (Section II), we also formulated the following additional requirements:

R1. The tool should *provide explanations* for the different issues. This is necessary as not all people involved in the requirement elicitation will be familiar with all terminology.

R2. The tool should allow *capturing the motivations for the choices made and issues* (not) considered. This allows documenting the process.

R3. The tool should be able to *visualize* the choices made. This allows the users to keep track of the elicitation process as well as of the choices made.

R4. The user should be able to *change decisions* already made and view the alternative choices again. This is needed because during discussions, it is possible that people change their mind.

R5. The tool should allow *exporting* the results in a *textual and readable form*. This is needed to support documenting the requirements in a textual form.

R6. The tool should have an *easy to use graphical user interface*. Obvious, as the target users include non-computer scientists (i.e., casual users).

Based on these requirements, we have opted for an interface comparable to the interfaces of mind-mapping tools (e.g., iThoughtsH[1], XMind[2], SimpleMind+[3]). A mind map is a diagram used to connect thoughts and ideas based on the concept of Radiant Thinking [11]. The main subject is placed in the center and thoughts and ideas radiate from this main subject in a hierarchical way. In a similar way, in our tool, the root of the feature model is placed in the center and the sub-features radiate from this central feature. The tool provides an easy to use "point, tap, and drag" user interface (Objective O2 and Requirement R6). Fig. 5 provides a screenshot of the tool while using the GuideaTemplate for the domain of Serious Games for Children.

Each guidea (i.e., feature) is represented as a rounded rectangle and contains the name of the guidea and an explanation of this issue and why and/or when to consider it (Requirement R1). For example, one issue to consider in the domain of Serious Games for Children is 'Age Range' and its explanation could be 'The age of the player may influence different aspects of the game. Therefore, specify the age range'. Double tapping on a guidea will open the guidea and allow the user to enter comments. The comments are used to document decisions taken and their motivations, or to write down things to be remembered (Requirement R2).

A guidea can have sub-guideas. For example, in Fig. 5 the guidea 'Resources' is decomposed into 'Time', 'Budget', and 'People'. A guidea is connected to its sub-guideas by means of arrows pointing towards the sub-guideas. The starting point is the root guidea, in Fig. 5 called 'My serious Game', which allows providing a short description of what the user wants to achieve with the game, and (in this template) decomposed into sub-guideas: 'User Aspects', 'Pedagogical Aspects', 'Game Aspects', 'Context of Use', 'Resources', and 'Implementation Aspects', which are on their turn decomposed (but not all unfolded in Fig. 5).

Not all guideas are mandatory to consider. The optional guideas are connected by dotted lines, the mandatory guideas by a solid line (Objective O1.2). An optional guidea can be deselected, but deselected guideas are still visible (grayed out – see Fig. 5. for some examples) and can be reselected (Requirement R4).

For some issues, a set of predefined options is available from which the user can select one (or more – depending on the guidea) (Objective O1.3). Such a guidea (which corresponds to an abstract feature in Feature Assembly) is represented as a small yellow colored rounded rectangle. An example is 'Platform' (see Fig. 6). For this guidea, the user can select one or more options from the available list: 'PC', 'Mac', 'Smartphone', and 'Tablet'. Options that cause a conflict or require the selection of other options (for other guideas), are marked with a red exclamation mark in the icon and the required and/or conflicting options are shown when the user taps the 'i' button on the right side of the option (Objective O1.4) (see Fig. 7).

Like in mind mapping tools, the sub-guideas of a guidea can be collapsed and unfolded (Requirement R3), dragging and resizing is also possible, as well as changing the color of guideas.

The GuideaMap created (so far) can be exported as text (Requirement R5), as well as in an exchange format.

The app can work with different templates, satisfying Objective O3. The GuideaTemplates can be loaded at run time.

VI. Evaluation

To evaluate our approach, as well as the associated tool, we decided to use case studies [12]. Setting up an experimental evaluation would require the specification of an artificial scenario, which would not allow us to evaluate the approach but merely the usability of the tool. The use of real live case studies allows us to evaluate the usability of the tool, as well as the approach. Therefore, we conducted two explorative case studies, both in the domain of serious games, aiming to build initial understanding of the usability and effectiveness of GuideaMaps tool and the underlying approach.

[1] www.ithoughts.co.uk/iThoughtsHD/
[2] www.xmind.net/
[3] www.simpleapps.eu/simplemind/

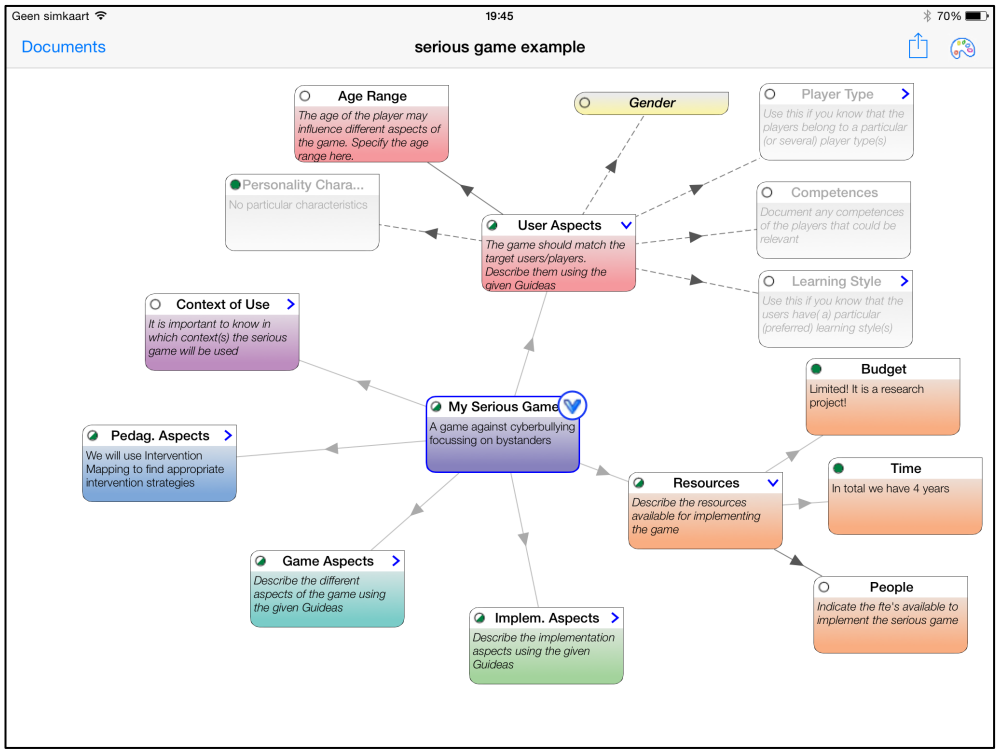

Fig. 5: GuideaMaps Screenshot

Fig. 6: Options for the Guidea Platform

The first case study was an informal case study, to obtain initial feedback on the usability of the proposed approach and tool and with the aim of deciding whether it was worth pursuing in this direction. This case study was done with two team members of the Friendly ATTAC project (age between 23 and 30). Both had been involved in the plenary session mentioned in the introduction and volunteered to participate in the evaluation session. The two persons involved had a background in Communication Science and no experience with tablets, requirement analysis, or game development.

Because, at the time we conducted this evaluation, most of the requirements for the serious game to be developed were already discussed (although in an ad-hoc way), in this evaluation session we mainly focused on the usability of the tool, and its capabilities to document the decisions taken.

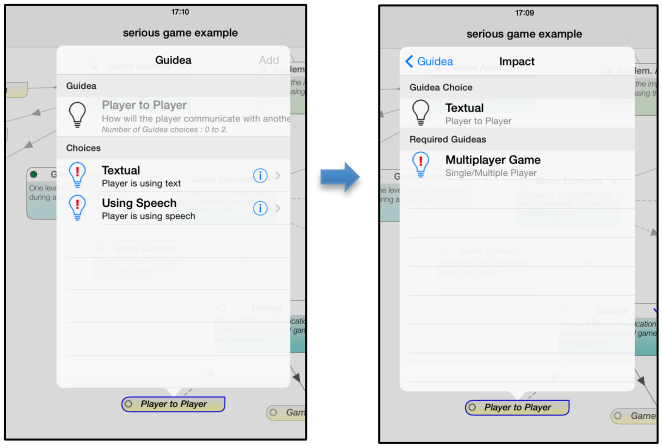

Fig. 7: Indicating the Impact of Choices

After a short introduction explaining the goals of the session and a short demonstration of the tool (5 minutes), an iPad with the app loaded with a Serious Game GuideaTemplate was handed over to the participants. They were asked to enter the available information about their serious game in the tool while we monitored their behavior. They were encouraged to think aloud. Getting started and entering the information took about 40 minutes. Afterwards, we asked the participants for

feedback, suggestions for improvement, and the completeness/relevance of the provided GuideaTemplate. This was done in an informal way and lasted 10 minutes.

In general, the participants provided positive feedback and were impressed by the functionality of the tool, as well as by the completeness of the template. They concluded that the tool could be "very useful in meetings to capture, in a structured way, the different decisions and make the necessary progress".

The second case study was conducted with a Computer Science master student who wanted to develop a serious game for children as part of his master thesis. As the participant was familiar with tablets, we were looking for feedback about the learnability of the app for this kind of users, as well as about the underlying approach. A preconfigured iPad was handed over to the participant with only a very short explanation about the purpose of the tool and the question to use it for his own requirement elicitation. We also asked if he would be prepared to fill in a questionnaire afterwards and be interviewed. There was no benefit for the student to be biased about the tool. This participant had experience with developing games in his free time.

After he finished the requirement elicitation phase for his project, we asked him to fill in the online Computer System Usability Questionnaire[4]. The scores were all positive; on a scale from 1 (strongly disagree) to 7 (strongly agree) he gave 5 times 6, 11 times 5, 2 times 4, and one 3. The lower scores (4 and 3) were due to some known bugs and limitations of the tool: score 4 was on the questions "The system gives error messages that clearly tell me how to fix problems", "The information (such as online help, on-screen messages, and other documentation) provided with the system is clear", and the score 3 was on the question "Whenever I make a mistake using the system, I recover easily and quickly"; the negative aspects that he mentioned were also all about known bugs and limitations. As positive aspects, he listed: "Easy to use", "Good overview", "Gives a good insight into the requirements of the project". Next, we conducted an interview to obtain more information about (a) the background of the participant: his experience with developing games, use of tablets, and familiarity with mind maps, (b) his opinion on the GuideaTemplate, and (c) his opinion on the provided functionality of the tool and our proposals for new features. Overall, the interview confirmed the positive evaluation of the questionnaire. He had no problems to start using the tool, purpose and terminology was clear. He was satisfied with the GuideaTemplate provided, he was not missing any issues, and quite some issues made him think more rigorously about his project and were even inspiring (e.g., the possibility to use a buddy, and the alternatives provided for reward system). Concerning the questions related to missing functionality and proposals for extra functionality, he was not asking for major new functionality. Two of the more advanced features that we proposed were considered as "may be interesting", i.e., (1) to be able to add new guideas or to add extra options for a guidea while using the map, and (2) to have a more structured comment field (i.e., divided into different sections including decision and motivation). He was not interested in a functionality to automatically solve conflicts with dependencies, as he considered this too dangerous. He was also fine with the current layout and saw no advantage in using a hierarchical layout. He did mention some usability issues: need for a better zooming, the possibility to hide some information at a certain level of detail, to provide different coloring rules (e.g., per level, per sub tree), and an export function to an image format.

Although both case studies resulted in a positive evaluation, it is not possible to generalize these results, because of the explorative nature of the case studies. For this, more case studies are needed, which are planned.

In the meantime, we also plan to extend the tool with some new functionality, e.g., allow the user to add, within certain limits, new guideas, as well as new options for abstract guideas. We would allow this because it may be hard to predefine all possible items and/or options at the time of defining the GuideaTemplate and we don't want to block creativity by predefining everything in advance. On the other hand, we should also be careful with providing such a functionality as users may start to introduce issues that are already available in the template but maybe under a different name or lower in the decomposition, or introduce their own set of issues and ignoring the predefined ones, which would bypass the original purpose of the tool.

The app can be used by individuals (like in the two evaluations) but also in meetings. During a meeting, the participants can go together through the issues, or they can prepare for the meeting by going through the issues in advance and discuss them afterwards in the meeting. However, note that we have not yet evaluated the tool in a meeting setting. Also no specific functionality is yet provided to support teamwork, such as functionality to keep track of who has decided what, when, and why. This is planned for a next version.

VII. Related Work

PROPEL [13] aims to guide users through the process of creating behavioral properties specifications, which are often used in requirements engineering to describe important aspects of a systems behavior. The emphasis of PROPEL is on constructing rigorous mathematical-based specifications. The approach is based on templates represented as "disciplined' natural language and finite automata. Our work is also based on templates but doesn't aim for a formal specification of the requirements. We are also not focusing on behavioral requirements. In [14], the templates are represented as Questions Trees that ask users a hierarchical sequence of questions about their intended properties. This hierarchical format guides users through the process and provides for each question, a set of alternative answers. The Questions Trees are basically decisions trees and much more restricted than our approach, for instance a user can select only one answer to each question and based on the answer selected new questions are presented to the user.

ORE [3] is an ontology based requirement elicitation method. A domain ontology is used to capture domain

[4] hcibib.org/perlman/question.cgi?form=CSUQ

knowledge. The domain ontology is used for semantic processing of requirements descriptions written in natural language and for detecting incompleteness and inconsistency [15], which is a different purpose than ours. Daramola et al. [16] also use domain ontologies but combine them with requirement boilerplates, which is a pre-defined structural template for writing statements. This work is specific for requirements about the security aspects of a software system. Compared to our approach, their focus is on imposing a uniform structure on the way requirements are written. In the same way, Toro et al. [17] proposed linguistic patterns, which are natural language requirement descriptions that can be reused, and requirements patterns that are generic requirements templates that can be reused with some adaptation. The concept of requirements pattern is based on the same principles as the well-known design patterns. Different requirement patterns for different domains have been defined, e.g., Konrad & Cheng [18] defined such patterns for embedded systems, and Li et al. [19] identified requirements patterns for seismology software applications. Others proposed a common structure for specific types of requirements patterns language, e.g., Roher & Richardson [20] proposed such a format for environmental sustainability requirements.

For the domain of scientific computing projects, Smith & Lai [21] proposed a specific requirement template, but the terminology used in the template is not very accessible. Li et al. [22] propose a domain specific requirement model for scientific computing projects. They evaluated the model and the results indicated that it facilitated the communication across the domain boundary. Their model is specific for the domain of scientific computing and still quite general (only containing requirement types such as performance, data flow, process, and data definition).

Bryant et al. [23] discussed domain-specific software engineering and point out that "the move from general-purpose to domain-specific representation has the potential to greatly impact the field of software engineering by allowing domain experts and end-users (who are not software engineers and do not understand traditional programming languages) to describe their computational needs in a representation that is familiar to them (i.e., based on domain abstractions and notations)." They also point out that requirement specification should be carried out in a domain-specific manner, but they see domain-specific requirement languages as the way to achieve this. We are not aiming for the development of domain-specific requirement languages.

Coulin et al. [5] also discuss the lack of systematic guidelines and flexible methods for requirements elicitation. Furthermore, they argue that no two software development projects are exactly the same, and therefore all projects cannot be adequately supported by a single static method. Therefore, they propose creating a situational method for requirements elicitation. The authors propose a systematic approach that provides the ability to engineer and tailor situational methods based on specific project characteristics. In their work, the focus is on engineering a dedicated method for requirements elicitation, which we don't aim for.

Also mind maps have been used for requirement elicitation, e.g., [24], [25], [26], [27], [28]. However, our approach is not based on mind maps (but on feature models), we only use a user interface that is very similar to the ones used for mind maps. Note that Wanderley et al. [27] use mind maps as a first step towards the creation of feature models, which is different from our goal.

VIII. Conclusion & Further Work

In this paper, we presented an approach for domain-specific requirement elicitation. More and more, the stakeholders that should be involved in the requirement elicitation are from different disciplines, with different backgrounds, and with different levels of experience in software development. This complicates the process of requirement elicitation.

The purpose of the approach is twofold. One the one hand, we want to unlock available information on requirement elicitation for particular domains. On the other hand, we want to provide a mechanism for guiding the stakeholders (non-computing as well as computing people) through the requirement elicitation process.

The approach is based on Feature Modeling. The issues to consider for a particular domain during requirement elicitation are modeled by means of a feature model. Furthermore, a tablet app has been developed to support the approach. The app provides explanations for the different issues to consider, indicates which issues are required and which are optional, provides possible options and alternatives, indicates the impacts of choices, and allows documenting choices made and issues considered.

The approach has been demonstrated for two different domains, the domain of serious games for children and the domain of e-shop web applications. The approach and the associated tool have been positively evaluated by two explorative case studies. Limitations and further work have been discussed.

Acknowledgment

This work is partially supported by the Agency for Innovation by Science and Technology (Belgium) (www.iwt.be) under the Friendly ATTAC project. We also like to acknowledge the people involved in the evaluation of the tool, Katrien Van Cleemput, Sara Bastiaensens, and Dieter Van Thienen, as well as our PhD student Pejman Sajjadi for creating the e-shop GuideaTemplate.

References

[1] M. J. Muller and S. Kuhn, "Participatory design," *Commun. ACM*, vol. 36, no. 6, pp. 24 –28, 1993.

[2] B. Nuseibeh and S. Easterbrook, "Requirements Engineering : A Roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, vol. 1, pp. 35–46.

[3] H. Kaiya and M. Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," *14th IEEE Int. Requir. Eng. Conf.*, pp. 189–198, Sep. 2006.

[4] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *Future of Software Engineering (FOSE '07)*, 2007, pp. 285–303.

[5] C. Coulin, D. Zowghi, and A. E. K. Sahraoui, "A Lightweight Workshop-Centric Situational Approach for the Early Stages of Requirements Elicitation in Software Systems Development," in *Proceedings of the International Workshop on Situational Requirements Engineering Processes (SREP 2005), France*, 2005, pp. 136–151.

[6] S. Buhne, K. Lauenroth, and K. Pohl, "Modelling requirements variability across product lines," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 41–50.

[7] L. Abo Zaid, F. Kleinermann, and O. De Troyer, "Feature assembly: a new feature modeling technique," in *Conceptual Modeling--ER 2010*, 2010, pp. 233–246.

[8] J. Bosch, *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach*. Pearson Education, 2000, p. 354.

[9] O. De Troyer and E. Janssens, "Supporting the requirement analysis phase for the development of serious games for children," *Int. J. Child-Computer Interact.*, In Press, available online http://www.sciencedirect.com/science/article/pii/S2212868914000099, 2014.

[10] P. Sajjadi, "Adapting WSDM to incorporate Web Application Patterns," Vrije Universiteit Brussel (Belgium), 2012.

[11] T. Buzan and B. Buzan, "The Mind Map Book How to Use Radiant Thinking to Maximise Your Brain's Untapped Potential," *New York Plume*, 1993.

[12] J. Lazar, J. Feng, and H. Hochheiser, *Research Methods in Human Computer Interaction*. John Wiley & Sons, 2010.

[13] R. L. Smith, G. S. Avrunin, L. A. Clarke, and L. J. Osterweil, "PROPEL: An Approach Supporting Property Elucidation," in *Proceedings of the 24th international conference on Software engineering - ICSE '02*, 2002, pp. 11–21.

[14] R. Cobleigh, G. Avrunin, and L. Clarke, "User guidance for creating precise and accessible property specifications," in *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, 2006, pp. 208–218.

[15] H. Kaiya and M. Saeki, "Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach," in *Fifth International Conference on Quality Software (QSIC'05)*, 2005, pp. 223–230.

[16] O. Daramola, G. Sindre, and T. Moser, "A Tool-based Semantic Framework for Security Requirements Specification," *J. Univers. Comput. Sci.*, vol. 19, no. 13, pp. 1940–1962, 2013.

[17] A. D. Toro, B. B. Jiménez, A. R. Cortés, and M. T. Bonilla, "A Requirements Elicitation Approach Based in Templates and Patterns," in *WER*, 1999, pp. 17–29.

[18] S. Konrad and B. H. C. Cheng, "Requirements patterns for embedded systems," in *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2002, pp. 127–136.

[19] Y. Li, C. Pelties, M. Kaser, and N. Nararan, "Requirements patterns for seismology software applications," in *2012 Second IEEE International Workshop on Requirements Patterns (RePa)*, 2012, pp. 12–16.

[20] K. Roher and D. Richardson, "Sustainability requirement patterns," in *2013 3rd International Workshop on Requirements Patterns (RePa)*, 2013, pp. 8–11.

[21] W. S. Smith and L. Lai, "A new requirements template for scientific computing," in *Proceedings of the First International Workshop on Situational Requirements Engineering Processes--Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP*, 2005, vol. 5, pp. 107–121.

[22] Y. Li, N. Narayan, J. Helming, and M. Koegel, "A domain specific requirements model for scientific computing," *Proceeding 33rd Int. Conf. Softw. Eng. - ICSE '11*, p. 848, 2011.

[23] B. R. Bryant, J. Gray, and M. Mernik, "Domain-Specific Software Engineering," in *FoSER'10 Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 65–68.

[24] K. Hiranabe, "StickyMinds | Agile Modeling with Mind Map and UML." [Online]. Available: http://www.stickyminds.com/article/agile-modeling-mind-map-and-uml. [Accessed: 29-May-2014].

[25] I. Mahmud and V. Veneziano, "Mind-mapping: An effective technique to facilitate requirements engineering in agile software development," in *14th International Conference on Computer and Information Technology (ICCIT 2011)*, 2011, pp. 157–162.

[26] J. Jaafar, M. Atan, and N. A. A. H. Nazatul, "Collaborative Mind Map tool to facilitate Requirement Elicitation," in *3rd International Conference on Computing and Informatics, ICOCI 2011*, 2011, pp. 214–219.

[27] F. Wanderley, D. S. da Silveira, J. Araujo, and M. Lencastre, "Generating feature model from creative requirements using model driven design," in *Proceedings of the 16th International Software Product Line Conference on - SPLC '12 -volume 1*, 2012, p. 18.

[28] J. A. P. Contó, W. F. Godoy, R. H. Cunha, E. C. G. Palácios, A. LErario, A. L. S. Domingues, J. A. Gonçalves, A. S. Duarte, and J. A. Fabri, "Applying Mind Maps at Representing Software Requirements," in *Contributions on Information Systems and Technologies*, 2013, pp. 1–5.