Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners

Aias Martakis
University of Twente
Drienerlolaan 5, Enschede, The Netherlands
a.martakis@student.utwente.nl

Maya Daneva
University of Twente, The Netherlands
Drienerlolaan 5, Enschede
m.daneva@utwente.nl

Abstract—Agile practices on requirements dependencies are a relatively unexplored topic in literature. Empirical studies on it are scarce. This research sets out to uncover concepts that practitioners in companies of various sizes across the globe and in various industries, use for dealing with requirements dependencies in their agile software projects. Concepts were revealed through online focus group research, using an adapted forum for discussion, and grounded theory to analyze the responses.

Our study resulted in the following findings: (1) requirements dependencies occur in agile projects and are important to these projects' success just as this is known for 'traditional' software projects'; (2) requirements dependencies (i) were considered and treated as part of risk management, (ii) were deemed a responsibility of the individual team members, and (iii) mostly did affect project planning; (3) continuous communication and collaboration — two essential features of any agile method, were found critical to mitigating the risks due to dependencies; (4) a hybrid approach to architecture between agile and plan-driven methods was perceived to yield maximum scalability and help coping with dependencies; (5) 'cross-cutting concerns', a category of dependencies, were not uniformly understood in an agile context and require more research.

Keywords—agile requirements engineering, requirements dependencies, cross-cutting concerns, agile software development, qualitative research, focus groups, grounded theory

I. INTRODUCTION (Heading I)

Since its definition in the 2001 manifesto [2], agile software development has been gaining increasing popularity, use and acceptance in the software developing community [1]. While the manifesto [2] describes key characteristics of agile software development, actual implementations were not part of it. Hence, the adaption and creation of many — previously labeled ‘lightweight’, software methodologies that give implementation to these agile concepts e.g. SCRUM, eXtreme Programming, Crystal Clear, Agile RUP, Feature driven development to name just a few [3]. With the number of successful agile projects increasing, the agile philosophy has also gained adoption by large corporations including the world’s leading [1,4,5,6]. While the use of agile methodologies has been expanding, quite a few important aspects of how agile happens in real-life, have remained insufficiently researched [56]. This paper addresses one such aspect, namely engineering of requirements in agile projects [8,30,48,49]. Specifically, we focus on the critical question of how the requirements dependencies are dealt with in agile requirements engineering (RE). In the RE community, it is widely recognized that dependencies are a major source of risk to cost-effective project execution, and ultimately, project success [54]. Proper identification of dependencies is therefore important to both maximizing project efficiency (i.e. by not reinventing wheels) and reducing risks (by preventing additional and last-minute integration efforts) [17,18,19,22,32]. RE studies [20,21] found that the highest gains are realized when dependencies are discovered or detected as early as possible in a project. In agile RE, most empirical research however happened in the context of small and medium size projects, which in turn means for RE research that whatever RE knowledge has been gained, its applicability might well be limited to this context [7,8]. Unlike small and medium size projects which might have a small number of well-understood requirements dependencies that all team members are well aware of, large projects that have many stakeholders would have many more dependencies to handle and many of them can easily evade team’s attention. More RE research is therefore necessary to understand the how agile RE practices treat requirements dependencies in large projects [8].

This paper explores if and how dependencies are dealt with in the software industry by using a focus group research design [24,25] on a group of agile stakeholders. The research involved a heterogeneous group of participants from organizations of various sizes (from Fortune 500 to small companies) and of various industries (e.g. financial, telecommunications, aerospace, web services providers, and social media). The participants took various agile roles and had a wide range of experience and software development backgrounds as well as various nationalities and corporate cultures.

To this end, we set out to answer the following research questions (RQs):
• RQ1: What concepts do agile practitioners use when reasoning about requirements dependencies in agile software development projects?
• RQ2: What is the response to the discovery of requirements dependencies in agile projects?
• RQ3: How are cross-cutting requirements dependencies dealt with in agile projects?

Our motivation for this focus group study initially originated from a combination of the first author’s personal experiences with agile development in a wide range of companies, and his reflection on the difficulties that his project organizations experienced in dealing with dependencies. Moreover, recent calls for more research on requirements dependencies in agile projects, e.g. [63] and [63], as well as the second author’s prior research of this topic in literature [48,49], affirmed the importance of dependencies within software projects and added much interest to investigate this within the context of agile methods. Most recently, a case study [8] led by the second author in a large CMM 5 certified IT solution provider revealed that requirements dependencies are an important concern that more often than not drives the way in which agile practices are implemented in large projects. While our prior research was on agile requirements prioritization in general [8,48,49], this paper zooms in specifically on the topic of dependencies amongst requirements as they are dealt with in agile.

The paper makes a contribution that refers to agile RE, and to empirical software engineering (SE), respectively. We directly responded to the call of the empirical RE community [52] and to the call of the empirical SE community for more empirical research with companies’ participation [53]. The research makes explicit the RE knowledge in specific agile contexts. Without explicating the knowledge that resides in practitioners minds, it’s hard to share learning on what worked and what did not work in agile RE and leveraging past experiences for the benefit of future projects.

The remainder of the paper is structured as follows: Section II describes published work related to this research. Section III presents the focus group research method and its use in our research settings. Section IV reports on the research execution and Section V on its outcome. Section VI discusses the research limitations, gives a reflection on the results and presents plans for future studies. Section VII concludes.

II. RELATED WORK

In the RE discipline, there is a large body of research on the topic of requirements dependencies [54,55,59]. A 2012 empirical study on requirements dependencies [59] found more than 20 types of dependencies identifiable in published RE literature. However, it seems that agile RE came into common use subsequent to much of this research.

RE literature, in particular the sub-field of goal-oriented RE [34], made significant contributions to understanding the concept of dependencies. Therein, requirements are described in a relationship, also called ‘dependum’, between a depender and the dependee. The depender is an actor that that depends on the dependee. Based on this definition, Yu and Mylopoulos proposed a strategic dependency model [34] that describes a systematic approach and framework to discover, evaluate and document the network of dependencies in three categories focused on relations among various actors (stakeholders and their roles). The categories of dependencies are: (1) goal, (2) task and (3) resource dependencies [35,56], and are defined as follows:

• Goal dependencies represent delegations of responsibilities to reach a goal;
• Task dependencies define actions that a dependee must complete, typically in temporal order;
• Resource dependencies require the dependee to provide the depender with a certain resource.

Understanding such dependencies is an important input to requirements prioritization activities, as found in a previously published analysis of state-of-the-art requirements prioritization methods [22].

Besides the goal-oriented view on requirements dependencies [27], RE researchers leveraged other perspectives to reason about dependencies. For example, Kassab et al. [23] treat dependencies from an ontology viewpoint. Gupta at al. [43] take a situational method engineering perspective and propose an approach to dependencies for the purpose of improving requirements change impact analysis processes. Yan et al. [31] used matrix theory to propose an approach to requirements dependencies for the purpose of managing requirements evolution. Most recently, a 2012 literature review [55] on requirements dependencies argued for the use of a network perspective to manage dependencies. These authors also designed a requirements network analysis process [57] to help assess dependencies of requirements that are specified at various levels of abstraction.

Furthermore, other RE and SE authors [60,61,62,63] leveraged the notion of ‘separation of concerns’ when reasoning about requirements dependencies. This concept was originally coined by Dijkstra to identify modularization of behavior in logical sections of a computer program to reduce complexity of software, and since the early 90ties has become an established principle [37]. Applying the ‘separation of concerns’ principle to requirements in a project generally allows a way to partition requirements in order to reason about specific concerns of the concerns, largely in isolation from others. Building upon this reasoning, for those goal dependencies that cannot be cleanly isolated and that are a shared concern across requirements of various actors, RE and SE researchers introduced the concept of ‘aspects’ [38,39,40,60,61,63,64]. For example, most recently Mussbacher et al. [60] proposed and evaluated an Aspect-oriented User Requirements Notation for handling dependencies between stakeholders ‘concerns. Alencar et al. [61] leveraged the benefits of aspects and classic goal-oriented concepts used in the i* method and proposed an approach to managing requirements dependencies that is grounded on both aspects and goal models. Ribeiro et al. [63] proposed to combine scenarios and aspects. Indeed, the paradigm of aspect-oriented RE [38,39,40] defines aspects directly to capture such cross-cutting concerns, which, in turn,
can both reduce the amount of dependencies in a requirements specification and identify ‘aspects’ in actors’ concerns [38,39,40]. Rosenhainer [41] presented techniques for aspect mining (that is, for finding cross-cutting concerns) and stated that these techniques are also relevant for the context of agile software development: “The techniques also appear to be suitable for an agile approach to identifying crosscutting influences, i.e. to only search for influences when it is actually required by some development or maintenance task”. As part of writing this paper, we went on and searched for empirical studies that report on using any of these techniques in agile projects, specifically. Our search was done in Scopus (www.scopus.com), an important digital library which indexes the major publication outlets in Computer Science (in particular, Software Engineering, Empirical Software Engineering and Requirements Engineering). We found no empirical study that explicitly focused on how agile projects treat requirements dependencies, be it in the light of aspect-oriented RE thinking or in general. We however found that other empirical researchers — e.g. [62, 63], deem requirements dependencies and particularly ‘cross-cutting requirements dependencies’ a critical topic in agile projects and call for more research on it. This encouraged us in carrying out the present study.

III. RESEARCH METHOD

This industrial study was performed as a qualitative research following focus group guidelines [24] and grounded theory analysis [42]. Below, we describe both types of research techniques and outline the steps performed.

A. Focus Groups

Focus groups are group interviews that capitalize on the communication between interview participants [24]. This means that the researcher not only considers the direct answers to interview questions but also responses between participants. This is useful when exploring participant knowledge and experiences and is used to research not only what participants think but also their way of thinking and its context. Focus groups are particularly useful in exploratory research without strict preconceived ideas, as it allows for collection of new insights and is a convenient way to collect data simultaneously from several knowledgeable people. Group processes also permit participants to clarify views that are less accessible in standard one-on-one interviews due to inherent group dynamics [42].

Our motivation for using a Focus Group Research Method [24] was the following: (1) it matched our research goal, (2) it was cost-effective, (3) it was recommended by other empirical software engineering researchers [24] for research situations similar to ours, and (4) we had prior experience [26] in using the method and could easily assemble adequate tool support.

To reliably implement this focus group research, we chose to use an online focus group [26]. We motivated this choice by the fact that it brings two benefits: (1) it allows for an extended period of interaction among the focus group participants, and (2) as the participants write their parts of the conversation, the end-to-end conversations on the topics of interest would be completely transcribed. (This significantly reduced the cost of our research, as we found once it was over.)

The online environment was set up by the first author. An online message board was utilized and adapted to the specific requirements of this research: (1) we wanted to be able to include participants from different geographic locations and time zones; (2) we wanted to allow for user convenience, letting each participant join the conversation in his/her most convenient time; (3) we wanted to allow users to engage in the conversation over an extended period of time, e.g. weeks and not in a period of, e.g. one hour, as in face-to-face focus groups. Due to the message board, participation became possible on the participants’ own terms. Also, since participants could respond in their own time we expected to get better thought-out responses that were arguably less impulsive.

The Focus Group research process consisted of the following phases: (1) Compose and organize questions; (2) Validate questions; (3) Finalize questions; (4) Setup and structure message forum; (5) Register participants; (6) Carry out first wave of questions; (7) Analyze results; (8) Carry out second wave of questions; (9) Analyze results; (10) Carry out last wave of questions; (11) Grounded theory analysis; (12) Writing up the report.

The composition of the questions was based on personal agile experiences and reviewed literature, and was validated by an experienced agile RE researcher (Bakalova, a fellow researcher in the research unit of the second author) resulting in a well-balanced set of academic and industry-relevant research questions. Once finalized and individual questions categorized and sorted around the three research questions, they were released to the participants in logical waves (iterations with a fixed ordering), in line with the three research questions. This was necessary to allow the participants to focus on certain groups of questions and to keep discussions running between participants on the subjects without discouraging the faster responding participants of moving ahead. The analysis of responses after each set of questions also (i) allowed for adaptation of the next wave and (ii) made it possible to intermediatively direct discussions towards unexpected topics or redirect the discussions emphasis.

Critical to receiving appropriate participation was the understanding of the research method by the participants. A clear invitation with instructions on how to contribute to the focus group was sent out prior to the study and each individual was helped to maximize his/her input on the forum. As most users were already familiar with the concept of forums and bulletin boards, little had to be demonstrated to start getting the first responses.

1) Participants

The initial study included 11 practitioners from 10 different companies around the world in the following sectors: financial services, aerospace, web 2.0, digital solution development. These practitioners had various agile positions. All of them were passionate about agile and had at least 4 years of experience in agile projects. The practitioners were chosen by the first author through his professional network, based on his knowledge about their typicality. The author selected them.
among a large group of colleagues based on his judgment whether they meet the requirement to be the professionals who have ‘the greatest amount of insights on the topic’ (as focus group research methodologists recommend [25]). All participants received an equal opportunity to participate, however not all of them were active at all times. The 11 practitioners participated relatively evenly in the conversations in the first two weeks, however, out of the 11 initial participants only six stayed involved till the very end (three more weeks). These six were from five different companies (see Table 1).

<table>
<thead>
<tr>
<th>ID</th>
<th>Industry</th>
<th>Estimated size of company’s staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>Financial and telecom service</td>
<td>800</td>
</tr>
<tr>
<td>Beta</td>
<td>Social media</td>
<td>400</td>
</tr>
<tr>
<td>Gamma</td>
<td>Digital solution provider</td>
<td>150</td>
</tr>
<tr>
<td>Teta</td>
<td>IT provider of various manufacturing systems</td>
<td>5000+</td>
</tr>
<tr>
<td>Zeta</td>
<td>Internet retailer</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 1. Companies who stayed active till the very end of the focus group research.

Table 2 provides an anonymized overview of the six participants that stayed till the very end. As shown, most of the participants indicated to have experienced software development in multiple roles and only one (see participant B, in Table 2) has the agile role of Scrum Master as his official job function.

The most popular methodologies seem to be XP/Scrum. Four out of the six practitioners consider themselves to still work agile on a day-to-day basis. The focus group participation column (the rightmost column in Table 2) is an indicator of how many posts (either new topics or replies) each participant had made in total compared to the other participants. The values ‘high’, ‘medium’ and ‘low’ mean between 150-120 post, 119-70 posts, and 69 posts, respectively (69 was the lowest number of posts and was done by participant D).

Table 2 also shows the heterogeneity of the work environments of the participants and estimation of the size of the company. Sizes of companies were recorded, as previous agile research showed insufficient empirical research on agile development practices in larger companies.

<table>
<thead>
<tr>
<th>ID</th>
<th>Agile role(s) taken by the participant</th>
<th>Formal job function</th>
<th>Development expertise</th>
<th>Agile Experience</th>
<th>Agile Methodologies</th>
<th>Active in agile at the time of the focus group</th>
<th>Level of participating in the focus group</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Team lead, Scrum master, Team member</td>
<td>Architect</td>
<td>Medium</td>
<td>Medium</td>
<td>DSDM, SCRUM</td>
<td>Yes</td>
<td>Medium</td>
</tr>
<tr>
<td>B</td>
<td>Team member, Scrum master</td>
<td>Scrum master</td>
<td>Medium</td>
<td>Low</td>
<td>XP, Scrum, DDD</td>
<td>Yes</td>
<td>Medium</td>
</tr>
<tr>
<td>C</td>
<td>Team leader</td>
<td>Team leader</td>
<td>High</td>
<td>High</td>
<td>Various</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>D</td>
<td>Team member, Team leader</td>
<td>Software Engineer</td>
<td>High</td>
<td>Medium</td>
<td>XP</td>
<td>Yes</td>
<td>Low</td>
</tr>
<tr>
<td>E</td>
<td>Team leader, Scrum master, Product owner, Stake holder, Technical expert</td>
<td>Director, Corporate Project Management Office</td>
<td>High</td>
<td>Low</td>
<td>XP, Scrum</td>
<td>No</td>
<td>High</td>
</tr>
<tr>
<td>F</td>
<td>Team member</td>
<td>Software Engineer</td>
<td>Medium</td>
<td>Low</td>
<td>XP, Scrum</td>
<td>No</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 2. Practitioners who participated till the very end of the study.

2) The Online Forum Used

The popular and open-source phpBB bulletin board was deployed on a hosted server, accessible online and worldwide. Installation was made easy through the available CPanel control panel on the server, provided by the web hosting company with a minimum of server configuration required to get the board operational. Configuration of phpBB was required for a focus group discussion; the initial focus was on providing the most convenient environment for discussion. In phpBB, after some initial experimentation and a small pilot (with two practitioners), the board was divided into three forums, one for each main research question. Within a forum, topics were created with (sub)questions relating to the main research question that could be responded to by participants. After entering all initial questions, all questions besides those within the first topic were hidden to focus discussion on the first topic. phpBB makes it possible to fully customize an initial list of questions during account registration which was an efficient way to retrieve context information from participants. The following information for each participant was tracked: (i) company name, (ii) current job title, (iii) estimated size of company, (iv) years of experience in software development, (v) years of experience with agile methodologies, (vi) number of agile projects, (vii) agile roles worked in, (viii) typical iteration size in current project, (ix) recent (within 2012) agile experience, (x) short description of agile projects and number of people involved, (xi) agile methods used (XP, Scrum, etc.)
After several dry-runs a detailed e-mail was sent out to potential participants explaining research intentions, duration, format and recommended usage with a screen by screen demonstration of the bulletin board. Once each participant registered and filled out the initial questions, security permissions to allow posting were set and accounts enabled (which automatically sent back notifications to users of successful registration).

B. Data analysis

Our data analysis process was inspired by the Constructive Grounded Theory approach of Charmaz [42]. Generally, Grounded Theory (GT) is a qualitative method originating from social sciences to construct propositions (or also called ‘theories’) from data. It allows for an exploratory approach to collect research data without preconceived ideas (or existing theories) and captures all facets of the qualitative data to form possibly new theories. In contrast to most other research, GT does not test a hypothesis but tries to discover theory implicit in the data. A typical GT workflow is shown in Figure 1.

![Grounded Theory Workflow](image)

Fig. 1. Grounded Theory Workflow.

Once the last answers were received and the forum closed for further submissions, a GT analysis was started on the available data by following Charmaz’ coding guidelines [42]. As the focus group research took place online, unwritten (verbal accentuation/non-verbal/etc.) language could not be considered and the given textual answers would therefore have to stand on their own.

The first action was to gather all questions and answers from the bulletin board and make them available in an online document. Coding was performed digitally by selecting text and inserting comments one sentence at a time in the digitally shared document.

Possible code relationships were noted down in a separate shared document representing the memos. Answers had been reviewed several times, continually re-working the memos. Sorting the memos compared categories on an abstract level and gave logic to organize the analysis and create and refine theoretical links. Writing-up the focus group report brought the results presented in the next Section and discussed in Section 5. (We make a note the participations of each researcher in this process is discussed in more detail in Section 6 where we reflect on validity threats.)

IV. RESULTS

The results are presented in the order of the research questions outlined in the Introduction. As it is usual in qualitative studies, we will supplement quotations of our interviewees.

RQ1: What concepts do agile practitioners use when reasoning about dependencies in requirements in agile software development projects?

All focus group participants explained they had encountered requirements dependencies in agile projects just as they had in non-agile development projects. As one participant strongly indicated: “This happens all the time. Find me the first project where you can work on each of the requirements independently and still build a cohesive product.”

As no initial definition purposely was given as to what exactly constituted a requirement dependency (as used in the context of this research) the participants were given room for discussion and each started to share their own perceptions. The focus group characterized the concept of **Requirements Dependency** as a situation in which progress of action on one requirement assumes the timely outcome of action on another requirement, or the presence of a specific condition. Examples of conditions can be a readily available feedback by clients on a working prototype, a business value estimation that motivates a changing requirement or access to expert knowledge when a question arises regarding clarity on a requirement. An important concept that emerged during the focus group discussion was the **Obviousness** of the dependency. The focus group participants perceived it as the level of visibility of the dependencies by team members. All participants routinely divided dependencies in two classes based on the level of obviousness they perceived:

(a) **Explicit Dependencies** were thought of as those that everyone on board knows about, their presence is indisputable, and coordinated action is usually used (be it sooner or later) to confront dependencies of this class; e.g. a dependency that arises when a project organization needs expertise that is not easily available in the company (and everyone knows this).

(b) **Tacit Dependencies** which are subtle in nature and are less obvious so that team members find it hard to judge the presence of a dependency and what it implies for the project. E.g. a dependency that arises when a key client has no full discretion to make decisions and trade-offs. Some focus group
participants referred to such dependencies as ‘shadowed’, ‘grey zone’, ‘unspoken’ or ‘taboo’.

Furthermore, dependencies were categorized in three groups based on the subject the dependencies referred to: (1) Integration Dependencies; (2) Execution Order Dependencies due to a certain order of performing tasks; and (3) Architectural Dependencies. The first group typically involved one system/module/unit that required interaction of another, as a participant mentioned: “certain features require other features to be built.” Participants indicated dependencies could be either within the same project or between different projects. Looking at the dependency categories [35] these mostly fall in to the goal dependencies as there is a higher overlapping goal.

The second group mostly impacts the project schedule and is a task dependency. E.g. a feature requirement can be implemented and tested first. Some participants also called this an ‘inter-story’ dependence as finishing up one story directly impacted others.

The third group involves cross-cutting concerns. These typically occur in the domain of system architecture such as security, logging, and monitoring. The participants’ interactions on this third group triggered discussion on the role of an architect or architecture group in agile projects in general. While participants agreed that the importance of architect’s role seems to reflect the scale of a project (e.g. large projects are ‘needier’ for architect’s attention than smaller projects), they disagreed on what an architect constitutes in agile projects. One participant shared that even in the same company, agile project organizations use different meanings for the architect’s job: while one project in which this focus group participant was involved, treated the architect as ‘the master mind’ behind the product’s design and the quality requirements for the delivered system, more often than not other projects considered the architect as a ‘consultant who you may go to when things go wrong’. Another participant called the architect ‘the agent of the development team who will advocate for why developers are right in cases when the clients are unhappy with the development work’. Two other participants thought architects participate only if the project is mission-critical to the company and most of the time their work is to be like business analysts or requirements staff members and not as architecture designers. Another focus group member thought that a senior developer takes over the architecture role whenever it’s time for making architecturally significant decisions. He called this practice ‘role sharing’ and said it was ‘the norm’ in his company.

As mentioned before, the role of architecture in agile projects caused much debate throughout all responses: “Agile assumes that systems grow like weed creeping up a wall or evolve like natural selection of living species (Darwin). That is short sighted. There has to be a system architect thinking about these larger interactions and about the master design.” (Participant A).

In response, Participant B: “We do not have this master architect that I hear but we do have a good communication between teams... Having an architect role is creating a single point of failure in your organization.”

In response, Participant C: “My company doesn’t use architects in the "all mighty" role, rather it has a virtual board of senior engineers who conduct reviews of projects with team leads.”

Was there anyone responsible for identifying dependencies? In the experiences of our participants, no single person or process typically looks for or finds dependencies in agile projects. Several participants mentioned dependencies to be a type of risks that are controlled within the overall risk management processes in an organization. However, these participants observed that agile development tends to look only at the current iteration requirements and not, as is key to plan-driven projects, foreseeable requirements.

Our observations indicated, somewhat expected, a relationship between projects’ size and complexity, and the encountered amount and types of dependencies [46]. Inter-project dependencies only appeared in the experiences of participants from larger companies, who also mentioned the use of risk management strategies more often. Smaller companies (e.g. Gamma and Zeta in Table 1) emphasized and relied more on communication between project members. Large companies also appeared to utilize less of the agile development practices in general, opting for a mix of agile and plan-driven practices, perhaps less ‘agile-idealistic’, and appeared more reserved towards pure agile processes with instead more emphasis on architectural practices.

Another relationship that our focus group participants indicated is that of project duration, project maturity (indicating a product that has received several releases) and the amount of dependencies. It was suggested that projects of longer duration — probably indicating a project with bigger scope, have a higher dependency risk. Next, project maturity was a concept that our focus group members used to mean projects that focus on systems having long history in an organization, for example upgrades and extensions of existing systems, new releases of products being used for years in the company, or replacements of legacy systems. In the experience of our focus group members, this kind of projects (‘being mature’) had the tendency to have more dependencies. We brought this point into the Discussion section (Section 5) of this paper and as we would see, more empirical evidence would be required to substantiate these findings.

Our focus group paid a special attention on the notion of I.N.V.E.S.T. (Independent, Negotiable, Valuable, Estimable, Small, Testable) requirements, an acronym coined by Bill Wake [14] to mean the quality aspects that characterize a ‘good’ user story (i.e. good requirements) in the XP process. One participant brought this notion in, as our study was on requirements dependencies and ‘I’ in INVEST stands for ‘Independent’. This notion was characterized by the focus group participants as a ‘worthy ambition, without much usage of the actual acronym itself’. As one participant put it:

“How could something that is that clear and simple be so hard?”
The participants agreed that the commonly used approach to pursue I.N.V.E.S.T. compatible requirements was the break-up of requirements in smaller units until they became workable. However, the opinions of the participants differed regarding what degree of I.N.V.E.S.T. makes a specific requirement workable; one participant mentioned:

“In our case they often are not broken apart and are implemented as a whole. This makes it hard to estimate as the full requirement is often not fully understood.”

Our focus group participants indicated that this is different every time (e.g. for each requirement, project, person, etc.) but more importantly requires sensible judgment. Rewriting of requirements therefore happened when requirements were less I.N.V.E.S.T. but more importantly, unpractical units of work and in the case of discovered dependencies it had most impact on the project planning. All focus group members agreed that their requirements most of the time met at least four out of the six I.N.V.E.S.T. criteria, however they deemed it a rarity to have a project in which all requirements were in I.N.V.E.S.T. form 100%. They deemed the criterion called ‘Estimable’ as the hardest to achieve from dependencies standpoint.

Concluding it can be said that requirements dependencies can appear in any software development project, in any iteration, agile or plan-driven and are perceived mostly as a risk. No specific dependency discovery processes or functions were in place nor was it associated with any specific agile role.

RQ2: What is the response to the discovery of dependencies in requirements in agile software development projects?

Dependencies only caused prioritization changes in the case of task dependencies and in the extreme cases where integration and architectural dependencies created major additional risk. Quoted from one participant to another:

“I think there should be a single process to detect and mitigate risks. Dependencies are just one type of risk.”

Response by another participant: “I agree, it should be part of the larger process of requirements gathering.”

Participants indicated that they were able to deal with dependencies within a sprint in most cases without too much effect for the overall project. Whether dependencies were discovered early or late in a project generally didn’t change the reaction, it did however change the amount of stress associated with the changes. It was also noted that it could affect the perception of progress and ultimately of value of inexperienced clients negatively. More experienced clients – typically also more involved and more experienced agile clients, had a better understanding of the requirements dependencies as the following quote exemplifies:

“Yes, during various stages of the project, you have to deal with them [dependencies] as they come along. If it is found during development it will cause more stress and or the delivery date gets pushed. The clients’ response depends on its involvement. The more involved the more understanding [of the dependencies].”

Development around known dependencies involved creating mock-ups of where those dependencies would significantly cause risk to the implementation for a certain requirement. Other strategies were not mentioned as a general practice but were left up to the developer (and or group of architects) to decide in each case how to react.

Most participants kept track of requirements through support tools, typically with the addition of a plug-in to fully support tracking dependencies (examples are JIRA + greenhopper, pivotal tracker + smart sheet, Rally, Soffront Trackweb and/or a list with linkage in Microsoft Sharepoint), or tracked only as part of risk-management practices. One participant noted: “In project management software this is generally done by linking entities that represent requirements to each other base on their dependencies. Or, by ordering /scheduling them in a way that symbolizes the dependency path.”

RQ3: How are cross-cutting concerns dealt with in agile projects?

All participants were clear on the principle of separation of concerns. To achieve separation of concerns, the goal to have as little overlap in functionality between features as possible within a system, special care is required for those concerns that cannot be cleanly decomposed from a system in both design and implementation. Cross-cutting concerns are known to cause either scattering or dependencies within or between systems [47]. Examples of such concerns are security, traceability, persistence, logging, and internationalization. As per our focus group, agile projects also encounter such concerns, however in this research it was found that participants had difficulty to capture these concerns in user stories, e.g. “I don’t see how user stories can capture these. Too simplistic.” All participants experienced that the clients in their projects were not bothered with such concerns due to a difficult translation into business value and participants indicated that the agile development team typically was left to include their cost within existing user stories: “These are difficult topics that, in my experience, the customer rather not think too much about.” This confirmed the findings of Racheva, et al. in [48,49].

Three participants were well aware of the various aspect-oriented RE techniques proposed and the techniques related to layered designs. However, for various reasons, they could not find a way to apply these techniques in their agile projects.

All participants also indicated that these concerns were often ‘too important not to track’, whether discussed with client or not. This brought discussion back on the earlier topic on that of the role of architecture in agile development. Two participants indicated they felt it was the responsibility of the architect role (being one or many people) to ensure such cross-cutting concerns were correctly identified and acted upon across user- stories and projects. As per one participant: “...we just proved the need for the Master Mind, the architect. This is one of the primary reasons to have a board or a master mind in place to track and communicate these concerns, work them into the framework.”
V. DISCUSSION AND REFLECTION

This section compares and contrasts our findings to related literature. We follow the order of the research questions as presented in the Introduction.

**RQ1: What concepts do agile practitioners use when reasoning about dependencies in requirements in agile software development projects?**

Our findings suggest that agile projects are in no way different in terms of presence of dependencies. However, we found that there was no commonly used agile practice that addresses systematically approaching dependencies in their projects. Our focus groups agreed that no agile approach considers requirements dependencies explicitly and no Scrum Master initiates an upfront conversation on requirements dependencies as it would have taken place in ‘traditional system delivery models’. While there was no single role being clearly recognizable in a project organization, for its responsibilities over requirements dependencies in agile projects, the focus group members thought of software architects as those being good candidates for this role (should it be institutionalized in their environments). However, the focus group experienced a heated debate on what the job of the architect should include in this case. We think that the variety of opinions on the architect’s role indicates a lack of clarity in the agile community on when and how an architect should be involved and make a difference, and what value his/her involvement would add to the project.

Furthermore, agile books’ authors (e.g. [14]) suggest well-written agile requirements (in the form of user stories) should meet the I.N.V.E.S.T criteria. The tacit expectation in these books is that meeting the criteria is easily achievable (as no agile method would ever suggest spending much time on big upfront requirements analysis and documentation). Our study found that the expectation to meet the criteria might not be realistic in real life. We think this divergence between our results and the agile literature can be a motivation for follow-up studies to investigate the practices that agile professionals use to create user stories to match the I.N.V.E.S.T criteria and the challenges along the way.

Last, but not least, the concepts that the focus group members used while talking about requirements dependencies, clearly indicated a new perception of requirements dependencies as explicit and tacit. While this division sounds logical and common sense and we could not see why it could not be used to split up dependencies in ‘traditional’ projects, we make the note that no prior literature study mentions such a categorization. We think that this division of explicit versus tacit is a characterizing feature of the agile working culture where much simplicity is achieved by sharing tacit knowledge among team members in face-to-face communication [14]. In such a culture, practitioners are likely to develop sensitivity for what is explicit and what remains tacit and this might have impacted the way they reasoned about dependencies.

**RQ2: What is the response to the discovery of dependencies in agile software development projects?**

Our participants found that no agile practice suggest how to respond to dependencies. However, they thought that making dependencies part of risk management seemed a promising practice. This finding is in line with published software project management and RE studies on requirements dependencies (e.g. [33, 56]). We think this topic warrants further research as no agile methodology explicitly addresses risk management due to requirements dependencies. Particularly the field of risk management [33] offers a variety of theories that explain the mechanisms that lead to various risk mitigation strategies. We think that further research on understanding why certain risk mitigation strategies may be more effective than others in coping with dependencies in agile projects, would have implications for improving our understanding of the agile process in general.

**RQ3: How are cross-cutting concerns dealt with in agile projects?**

Our focus group brought a surprising result: there was no known practice for coping with cross-cutting concerns or quality requirements in the organizations of our practitioners. This finding is in stark contrast with the evaluation of the aspect-oriented RE studies [39, 40] that suggested the fit of aspect methods with the agile paradigm. We considered this finding surprising also while reflecting on it from ‘a historical stand-point’ taking into account the developments in software reuse, object-orientation and service-oriented architecture. Software component reuse (propelled big in the 90ties by object-oriented programming [9, 10,11]), an increased project complexity and scale, and distributed architectures (e.g. enterprise applications involving orchestration of multiple software applications and systems) have all contributed to increasing numbers of dependencies in a software project [12, 13]. In practice, most dependencies are dealt with by defining and implementing some form of interfaces or contracts, known as the concept of loose-coupling, that generally hide implementation details from its clients and allows for concurrent development towards this interface/contract. This can be performed on many levels such as the service or business process level, as is common in Service Oriented Architectures and BPEL (for example through for the definition of WSDL), or within actual code, as is done commonly in Java and .NET by implementing an interface and exposing them [14,15,16]. The fact that no focus group member indicated any of these as an option to cope with cross-cutting concerns seems to be a signal that the phenomenon of requirements dependencies in agile might not be framed that much as a technical phenomenon but rather as a social one. As such, it might require some re-thinking from another perspective, e.g. a perspective that emphasizes the client-developers interactions. Also, our participants indicated a contradicting point: on one side, clients routinely avoided speaking about cross-cutting concerns, on another side – developers found these concerns too expensive to overlook them. This gives us a hint that tension may exist between clients and developers and technical solutions alone may not be enough to resolve this tension surrounding the topics of cross-cutting requirements dependencies.
VI. Threats to Validity

This focus group research will be evaluated by looking at the criteria in the checklist defined in Consolidated Criteria for Reporting Qualitative Research (COREQ): a 32-item checklist for interviews and focus groups [50]. Though the list was developed within the context of health care research, many of the criteria remain valid for review of focus group research in the software engineering domain. The first set of criteria all relate to the characteristics of the research team and its reflexivity. The first author is a software engineer in corporate settings with a background in the development of agile software mostly as a team-member, and some prejudice was possible in formulation of the research questions and guidance of the focus group interview. A way this was controlled was to have all actions reviewed by an experienced researcher (the second author) with a different background (non-agile in nature). Initially most participants resulted from the extensive professional network of the first author and thus prior relationships existed, this was identified and additional participants were approached to provide a better balanced group.

Due to the geographic spread of the focus group the study design was different practically from most cases of focus group research in literature. Sampling was done by looking through all available contacts and selecting those that had any practical agile experiences. No other criteria caused exclusion and an as wide as possible selection audience was targeted. Some participants were approached directly, others by connecting with participants of national agile discussion groups. The target size of the sample group was between 4 and 12 active participants. Due to the usage of a digital forum, active participation on a current discussion was more difficult to compel and it became necessary to communicate out and entice new responses from all participants throughout the research. A consequence of this was that the research saw participation attrition during the research towards the end and significant absence of promised participation at the beginning. This was dealt with by oversampling initially to compensate for the attrition but in the end it still significantly affected the representativeness of the sample group for the formulation of the conclusions. It however also meant those participants that were active were highly motivated and had more time to respond individually.

The heterogeneity of the group allowed for a wide range of experiences as input to responses but made it difficult to compare and correlate responses based on participant properties. Participant properties however were easily collected during registration.

Data analysis took place by scheduling multiple walkthroughs of the responses to create the necessary codes through iterative refinement. These codes were reviewed by an experienced researcher. All data gathered and all research was available online and on-demand, the entire research took place in the cloud. This allowed for parallel remote collaboration, research traceability, availability, and data provenance that were important when communication took place on-line. At the same time there were visual/physical limitations of the cloud based tools v.s. typical tools (actual paper memos or face to face collaboration) used in most focus group research that were overcome e.g. comments were used in documents to initially represent memos and later on were copied over in spreadsheets to give the appropriate overview necessary for the coding activities. There were enough deviations from face-to-face focus group research in this study alone to necessitate additional scientific research on the effects of performing research in this online research 2.0 format.

VII. Conclusions

This paper presented an exploratory empirical study of RE practices in agile development focused on requirements dependencies. Our most important findings are the following:

1. Requirements dependencies occur in agile projects and are important to these projects’ success just as this is known for ‘traditional’ software projects’.

2. Requirements dependencies (i) were considered and treated as part of risk management, (ii) were deemed a responsibility of the individual team members, and (iii) mostly did affect project planning.

3. Continuous communication and collaboration — two essential features of any agile method, were found critical to mitigating the risks due to dependencies.

4. A hybrid approach to architecture between agile and plan-driven methods was perceived to yields maximum scalability and help coping with dependencies.

5. Cross-cutting concerns, a category of dependencies, were not uniformly understood in an agile context and requires more research.

The study has the following implications for future research. First, the focus group research showed no formal agile RE practices around requirements dependencies. It also indicated that larger organizations treated requirement dependencies within their regular risk management strategy, while smaller agile projects relied on (inter) project member communication. More research is needed to understand what risk strategies work better in which agile contexts and why.

Second, an interesting observation in this study was the way organizations treated the cross-cutting requirements dependencies (e.g. security, monitoring, performance, etc.) in agile projects: the heated debate of our participants seemed to pinpoint a weakness in how agile methodologies are understood and applied (and thus defined) in light of requirements dependencies and architecture. Larger companies indicated having specific architecture roles (either one or many people) among their agile projects whereas smaller teams indicated architecture to be a shared responsibility of team members. Indeed, while literature suggests a symbiosis of plan-driven and agile architecture techniques where appropriate [44,45,51], the exact amount and nature of architect’s participation in agile projects remains an open question. We therefore consider this a topic for further research as it requires more empirical data to draw any conclusions.
ACKNOWLEDGMENT

The author would like to thank all the participants for sharing their wisdom and spending their time to enable this research. Our special thanks goes to Zornitza Bakalova for piloting the questionnaire, for ongoing conversations on this work, and for reading the first write-up of the focus group results.

REFERENCES


[28] Z. Racheva, M. Daneva, K. Sikkel, R. Wieringa, A. Herrmann, Do we know enough about requirements prioritization in agile projects: Insights from a case study, RE2010, pp. 147–156


[31] Yan, Y., Li, S., Sun, W., Dependency based technique for identifying the ripple effect of requirements evolution, Journal of Software 7 (3), 2012, pp. 544-550


[37] D. L. Parnas, On the criteria to be used in decomposing systems into modules, Commun. ACM 15(12), 1972, pp.1053-1058


[48] Z. Racheva, M. Daneva, K. Sikkel, L. Buglione, Business value is not only dollars - results from case study research on agile software projects, PROFES 2010.


