Providing Software Product Line Knowledge to Requirements Engineers – A Template for Elicitation Instructions

Sebastian Adam

Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern
sebastian.adam@iese.fraunhofer.de

Abstract. [Context & Motivation] Developing new software systems based on a software product line (SPL) in so-called application engineering (AE) projects is still a time-consuming and expensive task. Especially when a large number of customer-specific requirements exists, there is still no systematic support for efficiently aligning these non-anticipated requirements with SPL characteristics early on. **[Question/problem]** In order to improve this process significantly, sound knowledge about an SPL must be available when guiding the requirements elicitation during AE. Thus, an appropriate reflection of SPL characteristics in process-supporting artifacts is indispensable for actually supporting a requirements engineer in this task. **[Principal ideas/results]** In this paper, a validated template for elicitation instructions that aims at providing a requirements engineer with knowledge about an underlying SPL in an appropriate manner is presented. This template consists of predefined text blocks and algorithms that explain how SPL-relevant product and process knowledge can be systematically reflected into capability-aware elicitation instructions. **[Contribution]** By using such elicitation instructions, requirements engineers are enabled to elicit requirements in an AE project more effectively.

1 Motivation

As a key concept for streamlining software development, software product lines (SPL) [2] have proven to be a promising strategy, especially when time to market is a crucial success factor. Nevertheless, developing new systems based on an SPL (which is denoted as application engineering (AE) in the community) is still a time-consuming task [3], and the benefits of using an SPL approach are often less than expected [4].

One important reason for the low efficiency in AE is the non-systematic mapping of customer-specific requirements [6], even though it has been recognized that the success of AE mainly depends on how requirements are treated. This is especially a problem in SPLs, in which a significant number of requirements cannot be anticipated during domain engineering (DE) by means of explicitly predefined variants only, what is, for instance, often the case in information systems (IS).

On the one hand, current approaches rather foster the direct reuse of predefined SPL requirements than the effective alignment of a customer's actual needs with the

B. Regnell and D. Damian (Eds.): REFSQ 2012, LNCS 7195, pp. 147–164, 2012.

available SPL capabilities [8] [9]. However, especially IS development that is merely based on picking reusable requirements, which then imply a predefined system behavior, is not feasible, because such systems also have to reflect a large number of individual requirements in order to allow the customer to stand out from the competition. Therefore, many costly corrections are typically needed during AE until a delivered system fulfills its expectations. On the other hand, eliciting customer requirements without considering SPL characteristics early on is also not an appropriate option. Particularly since selecting an SPL implies a certain set of constraints, it becomes apparent that not all customer requirements can be realized as initially stated. Rather, trade-offs between ideal requirements and rapid development benefits must be made. However, making this trade-off is challenging, because information about the realizability of requirements is (beyond predefined variability models) neither formalized nor available in the early requirements phase. Requirements elicitation therefore becomes an error-prone task, and it relies on experts to predict the impact of requirements that can only be realized with additional development [6]. Unfortunately, guidance on how to proactively elicit and negotiate actual customer requirements and align them with SPL capabilities is not supported systematically yet [6] [8]. Hence, it is still hard to elicit requirements during AE, especially when the number of requirements that can be explicitly anticipated and described by means of variability or decision models is limited.

A requirements engineering (RE) approach for AE (called AERE) that precisely guides the elicitation based on the characteristics of a given SPL in a more flexible and rather constraint-based than enumerative manner is therefore needed. However, as making requirements engineers aware of SPL characteristics is not easy (and typically limited to anticipated variants only), the *scientific problem* to be addressed therefore deals with the question of how requirements engineers can be enabled to use sound SPL product and process knowledge for appropriately guiding the elicitation (i.e., how can they made aware of important SPL capabilities, constraints and needs). This includes two research questions:

1. How can SPL product and process knowledge be economically extracted and incorporated into the AERE process when a complete, explicit anticipation of customer requirements is neither economic nor possible?
2. How can this knowledge be represented to an AE requirements engineer to guide their elicitation without the need to adhere to predefined variants only?

While we have already discussed our ideas on how to cope with the first issue (see [10] [24]), this paper focuses on the second[1]. In the next section, related work is discussed. In section 3, we then introduce a template for elicitation instructions including the underlying research approach. An evaluation of this template is then shown in section 4, while section 5 summaries the whole paper.

[1] The work presented in this paper was performed in the context of the Software-Cluster project EMERGENT (www.software-cluster.org). It was partially funded by the German Federal Ministry of Education and Research (BMBF) under grant no. "01IC10S01". The authors assume responsibility for the content.

2 Related Work

While much effort has been spent on how to build up SPLs during the so-called domain engineering phase (DE), actual reuse during AE has not received sufficient attention yet [4] [11] [12]. This holds especially true for RE activities: While there are many publications about product line scoping (e.g., [13]) or RE in DE (see [1] for an overview), only few exist that focus on AE [12].

In this regard, many proposed AE approaches share the ideas of Deelstra et al. [3] and distinguish an initial configuration phase and a phase of tuning iterations. The purpose of the latter is to modify and extend the initial configuration until all customer requirements are sufficiently met. Thus, additional development is typically required also [3][6], as it is unusual that all requirements can be fulfilled by existing assets only.

Within AERE, one can therefore distinguish the instantiation of variable requirements that were created during DE, and the elicitation of customer-specific requirements from scratch [14]. Even if both activities are important, most approaches such as from Sinnema et al. [15] only focus on the instantiation step. For this purpose, AE requirements engineers are only provided with feature catalogues, variability models, or decision models that have to be processed during elicitation. Indeed, these approaches work very efficiently, but they are applicable only in highly predictable and stable domains, as they rely on the restrictive assumption that all requirements can be explicitly prescribed during the DE phase (which is otherwise neither economic nor feasible due to the size and complexity of modern SPLs [3]). Thus, even for requirements that differ only slightly from the foreseen variants, these approaches are not applicable anymore [9], which results in manual, typically not guided, and thus costly extensions. Guelfi et al. [9][11] therefore propose a constraint-based rather than enumerative approach that allows deriving products that are not explicitly foreseen but close enough to the SPL. However, their approach only addresses the actual instantiation, but does not support the requirements elicitation in a systematic manner.

For eliciting new requirements that have not been covered during DE, only some initial work exists [12]. So far, mostly the tasks of communicating the variability [17], selecting variants, specifying system requirements (i.e., selected variants), and supporting trade-off decisions have been proposed as being important in this context [18]. For this purpose, Bühne et al. [19] describe a scenario-based approach and Rabiser et al. [12], too, introduce an approach for more systematic AERE. However, one remaining problem is the fact that requirements are still identified in a solution-driven (bottom-up) way instead of in a problem-driven way [8], as the aim is rather to obtain a large degree of direct requirements reuse than to satisfy actual needs with available assets. Hence, when high individuality is required in order to satisfy a customer, guidance on how to elicit such needs and reconcile them with reuse capabilities more flexibly is not supported systematically yet. In particular, unaligned approaches such as from Djebbi et al. [8] or those from "traditional top-down" elicitation make a sufficient fit almost impossible without costly rework.

Beyond AERE, RE for COTS-based development does not seem to be sufficient either, as it just deals with the selection [20] and adaptation [21] of COTS components. Furthermore, existing COTS-RE approaches do not give any guidance on how requirements are to be elicited and negotiated in order to fit existing assets.

Tailoring AERE processes based on existing SPL capabilities and constraints is probably the only means to solve the introduced problem. The work of Doerr et al. [22], for instance, aimed at improving RE processes, is maybe one of the few existing approaches that explicitly reflects the specific needs and constraints of a development organization in an RE process. However, this approach lacks both a systematic identification of important product and process knowledge, and an appropriate reflection of this knowledge in process-supporting artifacts. Furthermore, the specific context of SPL organizations is not considered. Also more recent work dealing with the identification of (information) needs such as [23], does not offer guidance on how to derive important issues in this regard either. So far, the problem mentioned in the motivation cannot be solved satisfactorily yet with current approaches.

3 Research Approach

The general idea for defining a template for elicitation instructions as mentioned before has been based on our practical experience in industry, where we discovered (however, outside SPLs) that precise instructions are able to support systematic elicitation even for non-experts. In a consulting project some years ago, for instance, our requirements engineering team at Fraunhofer IESE enabled people with low RE experience to perform rather good elicitation merely by following precise instructions. However, when we then performed a literature analysis, we found out that similar work has not been proposed yet. Rather, most approaches found still "lack sufficiently precise and prescriptive instructions" [16].

A first step towards our template for elicitation instructions was the clarification of important SPL concepts and their interplay with AERE processes. For this purpose, we iteratively developed a comprehensive conceptual AE model based on literature reviews and several discussions with SPL experts (parts of this model are published in [10]). The elements of this model (e.g., assumptions, existing realizations, etc.) were then used to derive hypothetic requirements regarding the content that the elicitation instructions should provide to AE requirements engineers. As we assume our model to be complete, we also assume the derived hypothetic requirements to be rather complete. However, based on our own elicitation experience, we additionally defined a couple of hypothetic requirements regarding the general nature of elicitation instructions that could not already be expressed in our model.

In a third step, we then performed a survey with eight experienced requirements engineers in order to elicit their requirements on elicitation instructions for AERE. What all involved engineers had in common was that they had an academic background but also much experience in performing RE in industry. Besides one internationally renowned professor, two heads of a leading German requirements engineering group participated in the study. In general, five participants had elicited requirements in more than 12 projects already, so, it can be assumed that they were aware of the most important success factors and pitfalls there.

The survey was done by means of a questionnaire with open and closed questions. The open questions were used to gain new insights about content and structure suitable for elicitation instructions. The closed questions (using the scale "totally disagree, …, totally agree") were additionally used to get confirmation for our

hypothetic requirements, and for eliciting the general acceptance of the intended instructions. In this regard, we considered a hypothetic requirement as confirmed if the median in the answers was "rather agree" or "totally agree" and the minimum in the answers not lower than "neither agree nor disagree".

Based on the confirmed requirements (22 of 33 hypothetic requirements could be confirmed), we then developed the template for elicitation instructions. A central step during this task was the definition of text blocks (called "phrases") to be used for providing the required information. For choosing appropriate formulations for these text blocks, as well as for determining rules regarding their incorporation in a meaningful order, we then recapitulated, discussed, and formalized our own way of how we have successfully performed requirements elicitation in many projects so far.

In one of the last steps, we then checked by means of traceability links whether our template addressed all requirements. Furthermore, we developed a tool for automatically generating concrete elicitation instructions based on our template. The purpose of this tool development was twofold: first, to demonstrate the preciseness of our template via its ability to be implemented in software, and, second, to avoid the tedious work when defining corresponding instructions manually.

In order to finally validate the benefits and applicability of the template, we prepared a two-step evaluation approach that includes an expert validation and a controlled experiment. During the former step, which is described at the end of this paper, we let eight requirements engineering experts use and review an exemplary instruction according to our template to validate whether it is basically applicable and useful. During the latter step, which is still in progress, requirements engineering students are going to use another exemplary instruction for eliciting requirements in a realistic role play. The goal of this second study is to objectively evaluate the elicitation performance in comparison to state-of-the-art instructions. Furthermore, subjective assessment of the material will also be gathered.

4 A Template for Elicitation Instructions

Our basic idea for representing SPL knowledge is to provide precise and prescriptive AERE instructions (i.e., elicitation instruction) that define a meaningful sequence of elicitation steps based on a given SPL (see Figure 1). If AE requirements engineers have such instructions, they are expected to perform the elicitation better, even if not all customer requirements have been explicitly defined during DE. This is especially important for SPLs in which not all requirements have been documented.

To realize this notion, AERE instructions have to describe regarding which issues requirements have to be elicited and which constraints must be considered. Hence, detection, negotiation, and correction of unrealizable, missing, or superfluous requirements can be done much more proactively and thus faster during AE projects leading to a more efficient AE in general. The main benefit of this approach in contrast to the decision model or feature diagram questionnaires traditionally applied in AE is that instead of purely asking which features a system should have (solution-oriented requirements), the really necessary requirements can be systematically derived based on the given business problems (problem-oriented requirements). Thus, requirements that are already part of the SPL and those which are not can be handled

in an integrated manner. Despite this, the approach always enables a requirements engineer to be aware of all the basic SPL capabilities and limitations when a certain issue is discussed. Thus, (s)he is always able to immediately start negotiations about requirements that might lead to unexpected project delay. For this purpose, however, a constraint-based rather than enumerative description of the SPL is applied in order to cope with the challenge that an explicit variability expression is often limited.

Of course, we are aware that each elicitation instruction is basically rigid, and that it may be complicated to keep it in mind during real customer conversations. However, we do not expect that our AERE instructions are straightly used, but that they are used as an abstract process or even just as a mnemonic to inform requirements engineers about the content to elicit and the constraints that exist. Whether this works has been checked during an evaluation described in chapter 4. In this section, a template for such AERE elicitation instructions (called elicitation instructions below) and the research that has led to it is introduced.

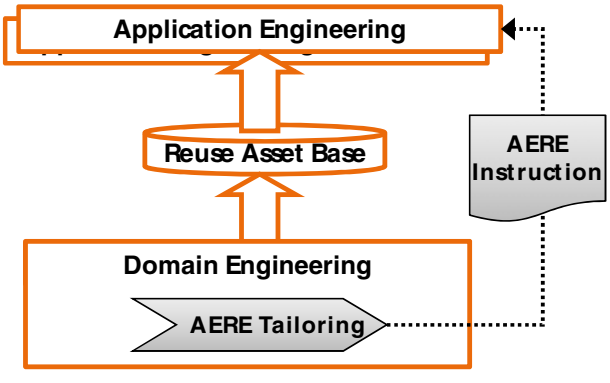


Fig. 1. Usage of tailored AERE instructions during AE

4.1 Requirements on Elicitation Instructions

Basically, six (75%) of the RE experts we involved in the requirements elicitation mentioned that a precise elicitation instruction would be of high or even very high value to support their elicitation activities in AERE. Especially some sort of clear, stepwise, procedural guidance that allows achieving a high degree of completeness in a constructive manner was demanded by almost all of the interviewed engineers in the open questions section. In this regard, the interviewed persons also mentioned that each statement in the instruction must be easy to understand and allow a requirements engineer to deviate if necessary. This means that an instruction should only guide and support a requirements engineer but not force him to do something that does not make sense in a concrete situation. Nevertheless, the instructions should be precise enough that a requirements engineer does not get lost when performing elicitation.

Besides this open feedback, the hypothetic requirements shown in Table 1 were confirmed by the survey participants based on their answers to the closed questions.

Table 1. Requirements on Elicitation Instructions

	An elicitation instruction should…
R.N.2.	clearly mention a sequence of steps to be carried out (clear how-to)
R.N.3.	explain how to proceed with the elicited requirements (e.g., visualizing, describing, classifying, …)
R.N.4.	be specific, i.e., customized for a certain development context or SPL
R.S.1.	make clear in which order elicitation steps should be performed best
R.S.2.	be modularized and allow taking breaks between sessions
R.S.4.	provide good indications for knowing when finished with elicitation
R.C.1.	mention the issues that are relevant for discussion
R.C.2.	make clear until which point in time certain issues have to be discussed
R.C.3.	name the typical stakeholders needed in a certain step
R.C.4.	inform about the details to be elicited with regard to a certain issue
R.C.6.	make clear about which issues a discussion is unnecessary (e.g., because no one in the subsequent development process will care about them)
R.C.7.	inform about which requirements are implemented by default anyway (e.g., common requirements / features)
R.C.8.	inform about whether requirements concerning a certain issue are restricted by architectural constraints
R.C.9.	make clear which properties a requirement must fulfill in order to be implementable
R.C.10.	inform about capabilities that already exist
R.C.12.	inform about conceptual dependencies between issues

4.2 Basic Structure

The overall purpose of elicitation instructions is to guide requirements engineers through a requirements elicitation process. Thus, the general structure of elicitation processes must be appropriately covered in the elicitation instruction template.

Basically, a requirements elicitation process consists of phases in which several (requirements) activities are performed. Phases are logical timespans reaching a milestone at which a certain result is achieved. Thus, the activities within a phase are needed to elaborate the outcome at the phase's milestone.

In Figure 2, our basic template for elicitation instructions according to a requirements elicitation process structure is depicted. For each phase, respectively requirements milestone, within a process, the elicitation instruction should provide a corresponding milestone section. The purpose of a milestone section is to collect concrete instructions for all activities that are needed to elaborate the requirements that must exist before the corresponding requirements milestone can be reached (addresses R.C.2 and R.S.4). If, for instance, a phase "business analysis" is part of a requirements process, the corresponding milestone section has to guide all activities that are needed to elaborate the business-relevant issues such as business goals, business objects, business rules, business processes, etc.

Each milestone section is therefore further subdivided into issue sections, which provide instructions for the elicitation of all requirements concerning one specific

issue (e.g., for business process, business objects, ,…). In the context of our work, an issue is a conceptual class of elements that are either part of system or a part of the system's usage environment. Thus, a requirements activity according to our model always deals with the elicitation of requirements regarding one specific class of elements.

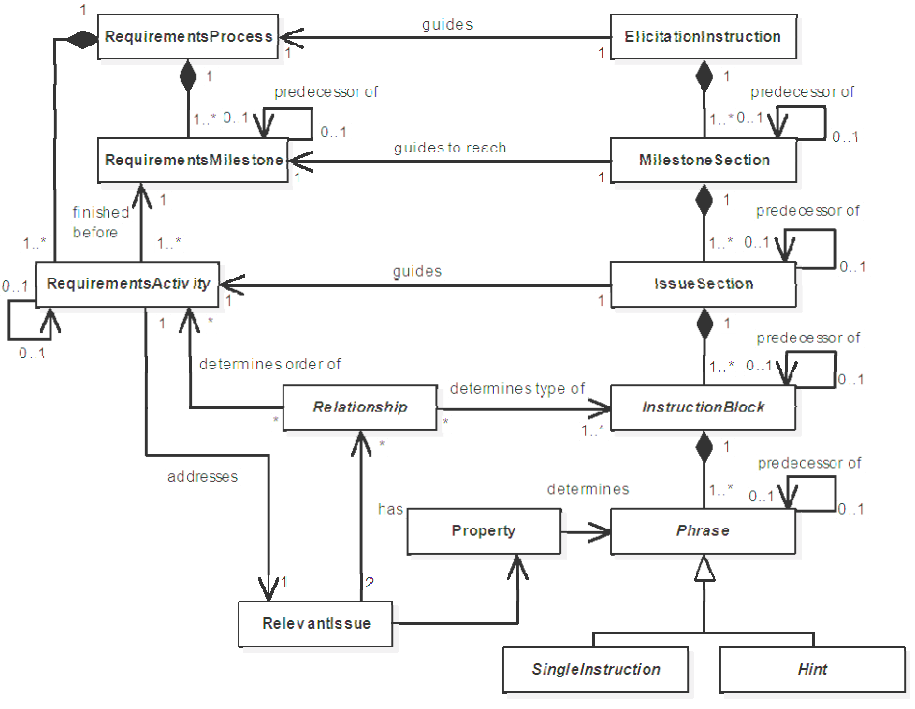


Fig. 2. Basic structure of elicitation instructions and its interplay with a requirements process

Within each issue section, concrete guidance on how to elicit and analyze the requirements concerning a specific issue is then given (addresses R.N.2). For this purpose, an issue section (see Figure 3 for an example) contains precise phrases (i.e., single instructions and hints), which are organized into so-called instruction blocks (addresses R.S.2). While the phrases comprise concrete statements for the requirements engineer on what to do or what to consider, the instruction blocks group these phrases in order to align different sub-activities such as asking, describing, classifying, clarifying, etc. more logically. Both instruction blocks and phrases therefore always depend on the actual issues of interest, respectively their properties and relationships (addresses R.N.4).

4.3 Single Instructions and Hints

The phrases, i.e., the single instructions and hints, are the core of each elicitation instruction as they form the elements that provide a requirements engineer with actual

knowledge. While the single instructions support the requirements elicitation through the predefined description of actions that are typically needed, the hints contain information that the requirements engineers should be aware of. This is especially needed to avoid the elicitation of non-fitting, superfluous, or missing requirements and, thus, to accelerate the alignment of customer requirements with SPL characteristics.

So far, we have identified 14 phrases (eight single instructions and six hints). Below, we briefly explain each of these. The underlined words in the examples are the words that are variable within the phrase's text block. However, besides the shown examples, other text block variants also exist to cover more specific situations.

Identifying Instruction. The purpose of this instruction is to find out what a customer basically wants or needs without defining his / her requirements in detail. The instruction therefore provides templates for "which"-questions based on the issue for which requirements are to be elicited (addresses R.C.1). In this context, the instruction makes use of an issue's relationships (addresses R.C.12). Example: "Ask the stakeholder the following question: Which User Groups are performing this Use Case?"

Collecting Instruction. The purpose of this instruction is to collect all identified requirements in an enumerative manner (e.g., bullet list) in order to handle the mass of gathered information (again without specifying details). The notion to focus only on enumeration reflects our strategy that details for each requirement should not be defined before a quite stable set of requirements has been achieved. Example: "Collect the identified User Groups in a corresponding list and add a link back to the related Use Case."

Describing Instruction. While identifying and collecting instructions just focus on gathering keywords of requirements without defining any details, the purpose of describing instructions is exactly to elicit and record this information (addresses R.N.3. and R.C.4). Describing instructions should therefore help a requirements engineer to motivate the stakeholders to provide detailed information about a requirement according to the attributes of the issue the requirement is concerned with. Example: "Ask the stakeholders the following question: Could you please describe this User Group especially with regard to average age, experience, …"

Classifying Instruction. The purpose of this instruction is to support the classification of requirements into more specific groups (addresses R.N.3). The rationale for this instruction is based on the observation that requirements concerning different issues are sometimes identified and collected in an integrated way, but need to be separated before they can be described in detail. Example: "Discuss with the stakeholders if this User Group is a Primary User Group or Secondary User Group and categorize it accordingly."

Visualizing Instruction. In elicitation sessions, requirements are often visualized, because visualization helps to clarify details or relationships much better than just spoken words. The visualizing instruction therefore aims at motivating a requirements engineer to use graphical representations during elicitation sessions (addresses R.N.3). Example: "Draw an Exchange Diagram to clarify the interplay between all User Groups."

Decomposing Instruction. The purpose of this instruction is to prompt a requirements engineer to decompose hierarchical structures in order to elaborate the included requirements (addresses R.N.3). The rationale for decomposing instructions is based on the fact that requirements are sometimes too coarse-grained to provide sufficient information for development. Example: "Decompose the hierarchy of this User Group until no further decomposition is possible. Collect the identified User Groups in a corresponding list and add a link to the parent User Group."

Selecting Instruction. The purpose of the selection instruction is to foster the reuse of requirements already defined during DE wherever possible. This instruction prompts a requirements engineer to consider the SPL specification, and to motivate the stakeholders to choose predefined requirements instead of letting them state these from scratch (addresses R.C.10). Example: "Motivate the stakeholders to select a best fitting Use Case from the SPL specification and map it accordingly. If the required Use Case is not covered sufficiently in the SPL specification, describe this Use Case especially with regard to name, precondition, flow of events, … from scratch."

Involving Instruction. The purpose of this instruction is to invite the stakeholders who are needed for a certain elicitation step (addresses R.C.3). This instruction is needed in order to assure that the right stakeholder group is available when requirements that concern a certain issue are discussed. Example: "Invite and involve a (group of) Business Area Managers to an elicitation session in order to discuss requirements concerning User Groups."

Influence hint. The purpose of this hint is to inform about influence relationships that exist between different issues, and that may also apply to corresponding requirements (addresses R.C.12). Example: "Important hint: Consider especially the Business Area when determining the User Groups."

Commonality Hint. The purpose of the commonality hint is to inform about requirements that are implemented by default anyway in order to proactively avoid unnecessary elicitations (addresses R.C.7). Example: "Important hint: Be aware that a set of Use Cases is already implemented by default and need not to be elicited again. Consider the list of these Use Cases in the SPL specification and break discussions immediately as soon as stakeholders start asking for the collection of these common requirements. Additional requirements are of course allowed."

Assumption Hint. The assumption hint is probably the most important hint for reflecting SPL characteristics and constraints in an elicitation instruction without the need to specify all possible requirements in an explicit manner upfront. Assumption hints describe the assumptions the product line architecture makes about a certain issue with respect to the flexibility the architecture has intentionally been designed for (addresses R.C.9). The purpose of this hint is therefore the description of constraints a requirement must meet in order to be assessable as being realizable by a requirements engineer without expert involvement. Example: "Important hint: Be aware that there are constraints defined for Business Document requirements. Hence, the Business Documents stakeholders may ask for are restricted as follows: pages<10, words<10000. If the stakeholders require something that contravenes these constraints, inform them about possible (significant) extra costs and that an expert check must be done before you can accept this requirement."

Selection Hint. The purpose of the selection hint is also to support SPL alignment. However, in contrast to assumption hints, selection hints directly aim at considering predefined requirements in the SPL specification and are therefore to be used together with selection instructions (addresses R.C.10). Example: "Consider the set of existing Adapters in the SPL specification."

Flexibility Hint. The purpose of the flexibility hint is to inform about possible extra costs when stakeholders require specific extensions or modifications even though reuse candidates already exist. Example: "If the stakeholders require specific Adapters that are not covered in the SPL yet, inform them about high extra costs even if the mentioned assumptions are kept."

Documentation Hint. There are issues that are actually relevant for development, and those that are only implicitly relevant for the elicitation of the former. The purpose of documentation hints is to inform the requirements engineer for which requirements it is not worthwhile spending effort for the description of corresponding details (addresses R.C.6). Example: "Important hint: It is not necessary to elicit or describe details about Business Processes."

4.4 Implemented Elicitation Strategy

It is evident that the milestone sections, the issue sections, as well as the phrases within each issue section must be ordered in a meaningful way in order to provide actual support. Besides a basic structure and several text blocks, our template for elicitation instructions therefore also comprises a set of rules to make that happen. These rules constitute an overall strategy that is implemented in the template.

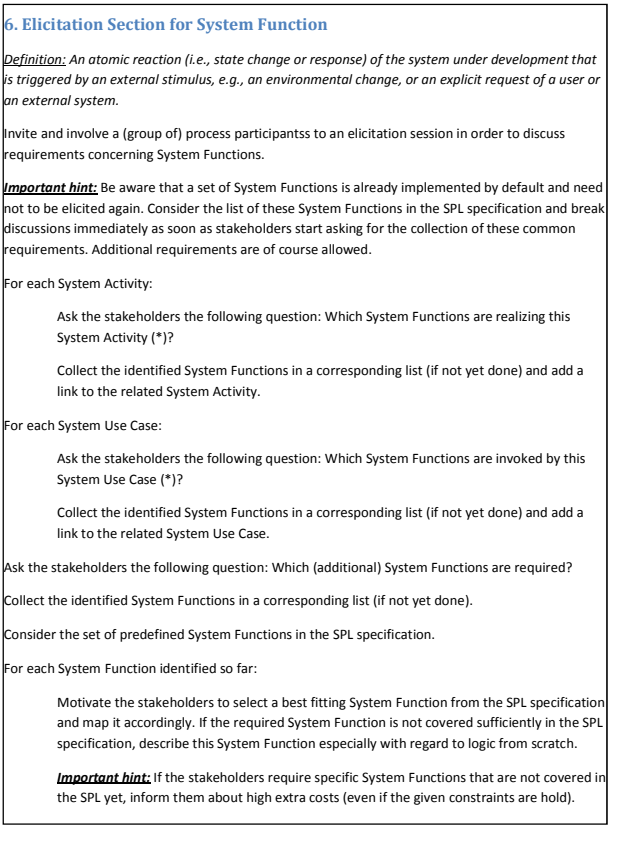
The milestone sections basically define clear points until which certain requirements types (i.e., issues) have to be discussed. The idea behind this approach is that requirements concerning different issues are typically needed at different points in time during subsequent development. For instance, the requirements concerning the technical environment in which a system should be integrated may be needed very early, while requirements concerning concrete functionality may be sufficient at a later point in time. Therefore, the order of the milestone sections must be the same as the order of the requirements milestones within the requirements (elicitation) process.

Within each milestone section, the issue section of the issues belonging to the corresponding milestone must also be ordered in a meaningful way in order to avoid redundancies. To define this order, the conceptual relationships between the issues must be considered. Basically, issues can have an "Influence", "Require", "Contain" and "Specialize" relationship (according to [5], [7]). When defining a logical order of requirements activities and corresponding issue sections, it is evident that requirements cannot be elicited in a random order. Therefore, the order of issue sections must be defined based on these relationships. In our template, we have defined several rules addressing this fact:

1. Discuss all issues in a random order that do not have any relationship to another issue.
2. Discuss all issues in a random order that are not required by, not contained in, not influenced by, and not a specialization of another issue. If there is none,

discuss at least those issues in a random order that are influenced by an issue already discussed, but that have no further relationships.

3. Discuss all those issues that are required by, contained in, influenced by, or a specialization of an issue already discussed, and that are neither required by, contained in, influenced by, nor a specialization of an issue that has not been discussed yet. If there is more than one, discuss them in the following order: 1) issues that specialize an already discussed one, 2) issues that are contained in an already discussed one, 3) issues that are required by an already discussed one, 4) issues that are influenced by an already discussed one. If there is more than one in each sub-order, discuss them in the order in which the specialized / containing / requiring / influencing issue has appeared. Adapt the order continuously and repeat this procedure until all issues related to a certain milestone have been discussed.



6. Elicitation Section for System Function

Definition: An atomic reaction (i.e., state change or response) of the system under development that is triggered by an external stimulus, e.g., an environmental change, or an explicit request of a user or an external system.

Invite and involve a (group of) process participantss to an elicitation session in order to discuss requirements concerning System Functions.

Important hint: Be aware that a set of System Functions is already implemented by default and need not to be elicited again. Consider the list of these System Functions in the SPL specification and break discussions immediately as soon as stakeholders start asking for the collection of these common requirements. Additional requirements are of course allowed.

For each System Activity:

- Ask the stakeholders the following question: Which System Functions are realizing this System Activity (*)?
- Collect the identified System Functions in a corresponding list (if not yet done) and add a link to the related System Activity.

For each System Use Case:

- Ask the stakeholders the following question: Which System Functions are invoked by this System Use Case (*)?
- Collect the identified System Functions in a corresponding list (if not yet done) and add a link to the related System Use Case.

Ask the stakeholders the following question: Which (additional) System Functions are required?

Collect the identified System Functions in a corresponding list (if not yet done).

Consider the set of predefined System Functions in the SPL specification.

For each System Function identified so far:

- Motivate the stakeholders to select a best fitting System Function from the SPL specification and map it accordingly. If the required System Function is not covered sufficiently in the SPL specification, describe this System Function especially with regard to logic from scratch.
- ***Important hint:*** If the stakeholders require specific System Functions that are not covered in the SPL yet, inform them about high extra costs (even if the given constraints are hold).

Fig. 3. Example of issue section “System Function”

When developing elicitation instructions based on these rules, it can be constructively assured that all requirements are available before the elicitation of related requirements starts. This is a key concept in our approach, as it is based on the

assumption that stakeholders can name requirements concerning a certain issue better when they consider the context of this issue by means of its conceptual relationships.

Within an issue section, our template therefore also proposes to elicit all requirements concerning a certain issue by considering its relationships to other issue. Thus, each issue section should first contain phrases that aim at identifying and collecting requirements, while the definition of requirements details should then take place afterwards; i.e., when all requirements have been identified by processing the issue's relationships. At the beginning of each issue section, one or more instruction blocks should therefore be implemented, where each instruction block reflects one (contained in or required by) relationship that the issue of interest has to another issue. For instance, a system function that is required by system activities and by system use cases would have two instruction blocks reflecting these relationships (see Figure 3).

The selection and instantiation of concrete phrases within an issue section is then based on the properties of the issue to be discussed, respectively on the properties of its related issues. The most important properties in this regard are the status of an issue and the degree of freedom provided by the underlying SPL. While the former expresses whether and how many instances an issue may have (normal = n, singleton = 1, abstract = 0), the latter expresses whether requirements concerning an issue are already predefined in the SPL, respectively restricted by the SPL architecture or strategy. In Figure 3, for instance, the degree of freedom states that a couple of system functions are already covered in the SPL specification, but that additional system functions may be specified also. Hence, corresponding hints and single instructions that inform a requirements engineer about this fact are included in the issue section.

5 Evaluation

To evaluate our template, we prepared a two-step approach comprising an expert validation and a controlled experiment. While the purpose of the first validation step, which is described below, was just to assess the practical applicability and usefulness of the elicitation instructions in general, its concrete benefits with regard to elicitation effectiveness are still to be evaluated in the second study that will be subject of a future publication.

Taking into account the individual background, a similar subject sample as during the requirements analysis was chosen for the expert validation (including the renowned professor). The overall goal of this first study was to assess our template with regard to its practical applicability and basic usefulness from the viewpoint of requirements engineering experts in the context of fictive interviews. In these fictive interviews, we let the experts use an exemplary elicitation instruction that was defined based on our template before. However, as we were only interested in an assessment of the instruction itself, the requirements stated by the interviewees were not considered here and often just brainstormed, non-controlled ideas.

5.1 Results

For the purpose of measuring the quality of the elicitation instruction, we used a questionnaire similar to the one for the requirements elicitation described above,

including a set of open questions and closed (agreement) questions. In Table 2, the assessments ranging from "very small" to "very high" received by the eight experts are listed, where MIN is the minimum, MED the medium, MAX the maximum, and Q_1 the 25%-quartile respectively Q_2 the 75%-quartile in the expert ratings.

Table 2. Expert assessment

Assessment Criterion	Statistics
Overall helpfulness in a SPL-based project	MIN = very small, Q_1 = medium, **MED = high,** Q_2 = very high , MAX = very high
Readability / understandability	MIN = low, Q_1 = high, **MED = high,** Q_2 = very high , MAX = very high
Usability / applicability	MIN = very low, Q_1 = medium, **MED = high,** Q_2 = very high, MAX = very high
Conformance with experts' personal elicitation style	MIN = very low, Q_1 = low, **MED = high,** Q_2 = very high, MAX = very high
Improvement of elicitation effectiveness (quality)	MIN = very low, Q_1 = medium, **MED = medium,** Q_2 = high, MAX = very high
Improvement of elicitation efficiency	MIN = very low, Q_1 = very low, **MED = medium,** Q_2 = high, MAX = high
Improvement in comparison to state of the art material	MIN = very low, Q_1 = medium, **MED = high,** Q_2 = very high, MAX = very high
Benefits for average requirements engineers	MIN = very low, Q_1 = medium, **MED = high,** Q_2 = very high, MAX = very high

The overall usefulness of the elicitation instruction according to our template was assessed as "high" or even "very high" by most involved experts. In particular, all participants stated that the detailed and consistent nature of the elicitation instruction as well as the provision of precise hints and clear instructions could support their work, even if they were experienced experts. Most RE experts also found the elicitation instruction easy to read and easy to use. Furthermore, five of the eight participants would use such instructions at least as an abstract process to follow during a project, as for most of them the elicitation instruction is compliant to their personal style of elicitation. Thus, it is expected that elicitation instructions following our template can be actually used in industry.

Regarding elicitation quality and efficiency, most of the experts expected a "medium" improvement in their own work when using instructions according to our template. In direct comparison to known (state-of-the-art) material in AE requirements elicitation, these improvements were even assessed as "high" in

average. Thus, even if not every requirement engineer will benefit to the same degree from using instructions according to our template, there seems to be a real target audience. In particular, the RE experts expected that at least less or average-experienced requirements engineers could highly benefit from using such elicitation instructions (what is to be evaluated in our second study).

However, with regard to the fulfillment of the requirements on the requirements elicitation instructions (see section 4.1), only the following requirements were considered as fulfilled in the expert ratings. In this regard, we considered a requirement fulfillment as confirmed if the median in the answers was "rather agree" or "totally agree" and the 25%-quartile in the answers not lower than "neither agree nor disagree".

- R.S.1. The elicitation instruction should make clear in which order elicitation steps should be performed best
- R.C.1. The elicitation instruction should mention the issues that are relevant for discussion
- R.C.3. The elicitation instruction should name the typical stakeholders needed in a certain step
- R.C.4. The elicitation instruction should inform about the details to be elicited with regard to a certain issue

The main reason for the low confirmation of the other requirements is the fact that the corresponding information in the exemplary elicitation instructions was not sufficiently highlighted and that a concrete application context was missing in order to assess the fulfillment of the requirements more thoroughly. This was also mentioned in the open part of the questionnaire, in which the involved RE experts made a few (minor) suggestions on what should be improved.

First of all, more rationales and background information about the elicitation instruction itself were required. In particular, this should include an explanation on how the instructions are to be used (e.g., regarding the order of steps, etc.) and what exactly they aim at. Second, the reasons behind each mentioned SPL constraint should be reflected in the instructions too in order to be aware why something works or does not. Third, links to notations and specific elicitation techniques should be included in order to provide a requirements engineer with access to more information on how to use them. Forth, more information should be provided regarding the purpose and content of the milestone sections in order to understand why the listed issues are to be discussed in its given order. Fifth, additional information on how to combine different steps into an elicitation workshop is required, including a coarse estimation of the time required for each step. Sixth, examples should be incorporated in order to show what the results of each step should look like. As a general feedback, we therefore claim that it is critical that the elicitation instruction itself is explained exhaustively to requirements engineers before they will use them in real projects.

5.2 Threats to Validity and Outlook on Controlled Experiment

The insights gathered by the expert validation confirmed the basic suitability of our template and also enabled us to improve it according to the feedback comments.

However, there are a few threats to validity that need to be discussed and also considered during the preparation of the controlled experiment.

Construction Validity. An important threat to construction validity was the fact that only one exemplary elicitation instruction based on our template was used for validation (mono-operation bias). Thus, there was neither a second elicitation instruction based on our template, nor a control group using an elicitation instruction based on another template. Another threat with regard to construction validity is the usage of only a questionnaire to measure data (mono-method bias). In particular, only subjective impressions and no objective data (e.g., regarding effectiveness) were collected. In order to avoid these threats in the controlled experiment, we will therefore setup two groups here; one using an elicitation instruction based on our template, and another group using a similar elicitation instruction according to best practice. Furthermore, both subjective data (based on questionnaires) and objective data (based on measurable observations) will be collected.

Conclusion Validity. Regarding conclusion validity the low statistical power due to the small sample size of only eight participants is an important threat to validity. In order to avoid this threat in the controlled experiment, we will involve approximately 30 participants here.

Internal Validity. The internal validity of the expert validation is mainly affected by the participant selection and the low degree of control during the study itself. Regarding the former, the experts were not randomly selected from a larger population, but only personally known experts were asked to participate. Regarding the latter, the study was done offline by each expert why we did not have any control how the fictive interviews were done. In particular, there is a risk that the elicitation instructions were rather reviewed than actually used. In the controlled experiment, we will therefore select the participants randomly from a set of unknown RE students, and perform their interviews in a controlled and comparable environment.

External Validity. As we involved real requirements engineers, the external validity is basically high. However, as elicitation instructions or methods in general do typically not address experienced experts, but rather less or only average-experienced requirements engineers, it would be interesting to gather also feedback from such people. In the controlled experiment, we will therefore give the same questionnaire to the participating students.

6 Conclusion and Future Work

AE based on an SPL is still a time-consuming task in practice. One important reason is the misfit between customer requirements and a given SPL, especially when a high degree of customizability is required. In order to resolve this misfit, AE requirements engineers must be enabled to use sound knowledge about a given SPL to better guide the elicitation of customer requirements.

As a first step towards this aim, this paper has introduced a template for elicitation instructions. Even if this template can basically be used in non-SPL environments also, its intended purpose is to appropriately provide requirements engineers with all

the important information they need for performing more effective elicitation in AE projects. A first validation with RE expert has confirmed that the template is basically suitable for this purpose, even if some minor issues still have to be improved.

However, as a concrete elicitation instruction always depends on a specific development context, each elicitation instruction must be defined individually for an SPL organization. Thus, the template introduced here is just one part of a larger research program. As mentioned in the introduction, the question, "How can knowledge about an SPL be economically extracted and incorporated into the AERE process?" cannot be answered by the template only, of course. For this purpose, we are developing a tailoring approach that systematically guides a method engineer in in incorporating SPL knowledge into elicitation instructions (see [24] for a first version of this approach). The work described in this paper presents valuable input for this aim, as it clarifies how the extracted knowledge shall be represented appropriately.

References

1. Alves, V., Niu, N., Alves, C., Valenca, G.: Requirements engineering for software product lines. A systematic literature review. In: Information and Software. Elsevier (2010)
2. Clements, P., Northrop, L.: Software Product Lines: Patterns and Practice. Addison Wesley (2001)
3. Deelstra, S., Sinnema, M., Bosch, J.: Product derivation in software product families: a case study. The Journal of Systems and Software 74 (2005)
4. Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. In: SPLC. IEEE (2007)
5. Vicente-Chicote, C., Moros, B., Toval, A.: REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. Journal of Object Technology, ETH Zurich 6(9) (2007)
6. O'Leary, P., Rabiser, R., Richardson, I., Thiel, S.: Important Issues and Key Activities in Product Derivation: Experiences from Independent Research Projects. In: SPLC (2009)
7. Goknil, A., Kurtev, I., van den Berg, K.: A Metamodeling Approach for Reasoning about Requirements. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 310–325. Springer, Heidelberg (2008)
8. Djebbi, O., Salinesi, C.: RED-PL, a Method for Deriving Product Requirements from a Product Line Requirements Model. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 279–293. Springer, Heidelberg (2007)
9. Guelfi, N., Perrouin, G.: A Flexible Requirements Analysis Approach for Software Product Lines. In: Sawyer, P., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 78–92. Springer, Heidelberg (2007)
10. Adam, S.: Towards Faster Application Engineering through Better Informed Elicitation – A Research Preview. In: REEW@RefSQ 2011, Essen (2011)
11. Perrouin, G., Klein, J., Guelfi, N., Jezequel, J.: Reconciling Automation and Flexibility in Product Derivation. In: Software Product Line Conference. IEEE (2008)
12. Rabiser, R., Dhungana, D.: Integrated Support for Product Configuration and Requirements Engineering in Product Derivation. In: SEAA. IEEE (2007)
13. Schmid, K.: Planning Software Reuse - A Disciplined Scoping Approach for Software Product Lines. PhD Theses in Experimental Software Engineering 12. Fraunhofer (2003)
14. Eriksson, M., Börstler, J., Borg, K.: Managing requirements specifications for product lines – An approach and industry case study. Journal of Systems and Software (2009)

15. Sinnema, M., Deelstra, S., Hoekstra, P.: The COVAMOF Derivation Process. In: Morisio, M. (ed.) ICSR 2006. LNCS, vol. 4039, pp. 101–114. Springer, Heidelberg (2006)
16. Cheng, B., Atlee, J.: Research Directions in Requirements Engineering. In: Proceedings of Future of Software Engineering (FOSE). IEEE Computer Society (2007)
17. Halmans, G., Pohl, K.: Communicating the variability of a software-product family to customers. In: Software and System Modeling 2003/2. Springer, Heidelberg (2003)
18. Pohl, K.: Requirements Engineering – Grundlagen, Prinzipien, Techniken. dpunkt (2007)
19. Bühne, S., Halmans, G., Lauenroth, K., Pohl, K.: Scenario-Based Application Requirements Engineering. In: Software Product Lines. Springer, Heidelberg (2006)
20. Alves, C.: COTS-Based Requirements Engineering. In: Cechich, A., Piattini, M., Vallecillo, A. (eds.) Component-Based Software Quality. LNCS, vol. 2693, pp. 21–39. Springer, Heidelberg (2003)
21. Alves, C., Franch, X., Carvallo, J.P., Finkelstein, A.: Using Goals and Quality Models to Support the Matching Analysis During COTS Selection. In: Franch, X., Port, D. (eds.) ICCBSS 2005. LNCS, vol. 3412, pp. 146–156. Springer, Heidelberg (2005)
22. Doerr, J., Paech, B., Koehler, M.: Requirements Engineering Process Improvement Based on an Information Model. In: Requirements Engineering Conference. IEEE (2004)
23. Sommerville, I., Lock, R., Storer, T., Dobson, J.: Deriving Information Requirements from Responsibility Models. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 515–529. Springer, Heidelberg (2009)
24. Adam, S., Doerr, J., Ehresmann, M., Wenzel, P.: Incorporating SPL Knowledge into a Requirements Process for Information Systems. In: PLREQ @ REfSQ 2010. Essen (2010)