Collaborative Requirements Elicitation: A Process-Centred Approach

Jacqueline Konaté, Abd-El-Kader Sahraoui, Gwendolyn L. Kolfschoten

Collaborative Requirements Elicitation: a Process-Centred Approach

Jacqueline Konate (1) Abd-El-Kader SAHRAOUI (2) G.L.KOLFSCHOTEN

(1) Departement d'informatique , Universite de Bamako, Bamako, Mali

(2)LAAS-CNRS, 7 avenue du Colonel Roche, F-31077 Toulouse, France

Université de Toulouse; UPS, INSA, INP, ISAE; LAAS, F-31077 Toulouse, France

(3) Computer science department, Delft University, delft, Holland

Abstract

Requirements Engineering is one of the first and most critical processes in system engineering. In this paper we will focus on the collaborative aspects of requirement engineering, in the context of product development. To do so, we adopted the separation of concerns method. Using this method we separate engineering aspects from collaboration aspects in order to study both aspects and finally integrate them. For the collaborative aspect of requirements engineering we looked at Collaboration Engineering. Collaboration Engineering is an approach to design and deploy processes for recurring collaborative tasks that can be transferred to practitioners to execute for themselves without intervention of professional facilitators. From an engineering perspective we will use the requirements engineering processes described by system engineering standard EIA-632 as a starting point. To integrate these we will use methods and techniques from Collaboration Engineering to specify the collaborative processes involved in this requirements engineering approach. An object model was build using Unified Modelling Language (UML). This model shows different concepts underlying our approach. Finally two case studies are presented to evaluate this approach.

Keywords : Requirements Engineering, Collaboration Engineering, Product Development, Standard EIA-632.

1 Introduction : The success of a system development project depends on the successful identification of the needs that a system must satisfy. Several years ago, systems were often developed according to the visions of engineers and designers without the involvement of users and other stakeholders. This caused the abandonment of many systems that were technologically sound, but have been considered as failed systems, because they did not meet the needs of users. In recent years, system design has been more successful, as engineers realized that customers and end-users must play a central role in the system development process and especially in the process of defining the envisioned functionalities of the system. To establish a sound set of requirements, many things have to be taken into account such as: the problem or the user needs, budget, resources and even political aspects (Boehm et al., 2001). Taking into account all these issues requires the involvement of several skills and therefore the involvement of multiple participants. Thus, collaboration is essential to ensure efficient and effective involvement of all relevant stakeholders.

Increasingly, information systems offer support to multiple users that collaborate to create products and make decisions. The current economic context encourages organizations to consider fusion-acquisitions and to externalize activities. Consequently, organizations are forced to improve the communication and interaction with their customers, their partners, their suppliers, their subsidiary companies, etc. to produce better products or services faster and at lower cost (Monford, Goudeau, 2004). Furthermore, organizations increasingly rely on collaborative teams to perform recurrent and collaborative tasks. However, when collaboration exceeds a limited context and when it relates to projects of considerable size, it poses significant challenges (Nunamaker et al., 1997). To overcome these challenges it is important to structure and organize collaboration in order to reap the benefits of collaboration in an effective way. Collaboration has an important human and social dimension making it difficult to design and predict the effect of interventions to support collaboration.

Requirements engineering is considered as a reconciliation approach of social and technical issues (Goguen, 1994). Tools can be masterpieces from a technical viewpoint; however, this does not guarantee that the system supports collaborative interaction and productivity of its users. It is therefore important to define processes to support (collaborative) tasks of stakeholders and users of the system.

Given these challenges, this paper proposes an approach to address collaborative requirements engineering for product development. To do this, we adopted an enhanced separation of concerns approach, i.e., we separate engineering aspects from collaboration aspects. We firstly consider the engineering process based on a system engineering standard EIA-632 (EIA-632, 1999) secondly, we defined a collaboration process for requirement engineering using the Collaboration Engineering approach.

Collaboration Engineering is an evolving research area which aims to design and to deploy processes for recurring collaborative tasks. In Collaboration Engineering collaboration processes are designed by collaboration experts and transferred to practitioners in the organisation, so they can execute these processes without the intervention of professional facilitators. From engineering perspective we propose that it is possible to define collaboration processes that are predictable, reusable and transferable to novice or practitioners (Briggs et al., 2003, Vreede and Briggs, 2005). The multidisciplinary character of requirements engineering justifies the need for a collaborative approach, as the development of complex products requires effective and efficient collaboration between various experts.

The remainder of this document is organized as follows: Section 2 presents a background on systems engineering, especially requirements engineering, collaboration issues and collaboration engineering. Section 3 is devoted to explain our approach. Section 4 shows a conceptual model of our approach. Section 5 gives an example to illustrate our approach. Section 6 presents the case study, the results, analysis, discussions and our prototype for specification. Finally, conclusions and suggestions for future research are outlined in section 7. The appendix contains descriptions and definitions of key terms.

2 Background

In this section we present the context and basis of our work in two parts: Requirements Engineering and Collaboration Engineering.

2.1 Requirements engineering in a system engineering framework

In this part, we present Requirement Engineering (RE) as a process which is a sub process of System Engineering. Many approaches exist for design and systems engineering. We used the System Engineering Standard EIA-632 as a starting point for our research.

2.1.1 Requirements Engineering

According to standard EIA-632, a requirement can be defined as "*something that governs what, how well, and under what conditions a product will achieve a given purpose*" (EIA-632, 1999). The IEEE defines a requirement as *"a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents"* (IEEE, 1990). In others words, a requirement is simply something that the system must do, must satisfy, or that is determined by someone concerned with the development of that system. Thus, Requirement Engineering is concerned with understanding what a system must do, whereas the 'design' is concerned with how the system should do this. RE is defined as "*the activities that cover discovering, analyzing, documenting and maintaining a set of requirements for a system*" (Sommerville, Sawyer, 1997). It is very important to note that a RE

process is not only a process of collecting facts, but that it encompasses all the project lifecycle activities associated with understanding the necessary capabilities and attributes of any given system (Wiegers, 2000). In RE we specifically distinguish between a *need* and a *requirement*. *Needs* are perceptions of the system from the client, end-users and other stakeholders, whereas *requirements* represent perceptions of the system from the project manager, analyst and designer. Consequently, a key challenge is to translate *needs* into *requirements*. Such translation can cause misunderstandings and misinterpretation, leading to false requirements. As shown in *(a)* in figure 1, the RE process consists in identifying needs from clients, users and other stakeholders and transforming them into technical requirements for a system.

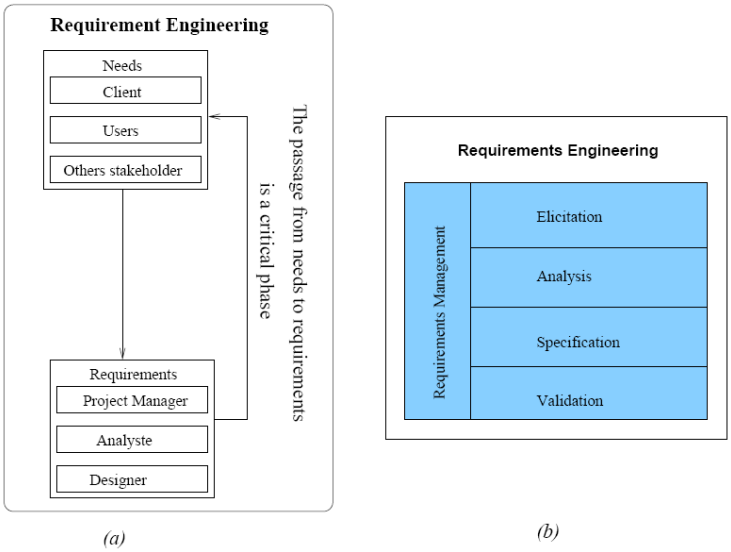


Figure 1: Requirements engineering process. (Essame, 2002; Kotonya, Sommerville, 1998; Coulin, 2007).

Requirements engineering does not only concern collecting needs, but rather it covers the lifecycle of the system as shown in *(b)* in the figure 1. According to (Kotonya, Sommerville, 1998; Coulin, 2007), the essential phases of requirements engineering are Elicitation, Analysis, Specification, Validation and Management. In the requirements elicitation phase, the purpose is to collect, capture, discover, and develop the requirements from a variety of sources including human stakeholders. In the analysis phase, the purpose is examining, understanding and modelling the elicited requirements, and evaluating them on quality in terms of correctness, completeness, clarity, and consistency. Further, requirements have consequences in terms of costs, resources, and possibly they might conflict with other requirements. Such consequences can lead to the judgement that some needs are more or less important, and that some needs should be abandoned. This requires negotiation about requirements among designers and analysis, and possibly among stakeholders, clients and users (Boehm et al., 1994). In the specification step, we record and document the requirements in such a way that they can be used by stakeholders and especially developers who will design and construct the system. In the requirements validation phase the quality of the requirements needs to be confirmed, to ensure that they actually represent the needs of the stakeholders, and that no translation errors from needs to requirements have been made. Requirements are managed throughout the RE process. Managing requirements includes activities such as change control and requirements traceability. RE is one of the most important branches of research in the System Engineering (SE) field (Sahraoui et al, 2008). For the purpose of our study, we chose the standard EIA-632 (EIA-632, 1999) for requirements engineering that is presented in the next subsection.

2.1.2 The standard EIA-632

Because of the increasing complexity of systems, rigor of design approaches has changed accordingly over the years. Systems engineering approaches propose a set of rules and processes

according to which systems must be developed. Thus, several standards of system engineering evolved to cover different phases of a systems lifecycle. In this paper we focused on the EIA-632 (EIA-632, 1999) standard, which we used as a starting point for our design approach (see Figure 2). This standard comprises thirteen processes, which cover different phases of the product life cycle. Since some stages of this life cycle are beyond the scope of our research, we focus only on the product realization processes and the technical evaluation processes. Systems engineering is based on an iterative process that starts with (1) understanding a problem, then (2) examining alternative solutions, and finally (3) verifying that the selected solution is correct before continuing to the definition of activities or proceeding to the next problem. Each process is a succession of activities involving three types of participants: the *developer* who is for example requirements engineer, the *end-user* who represents people for whom the system is constructed and other the *stakeholders* with respect to the system.

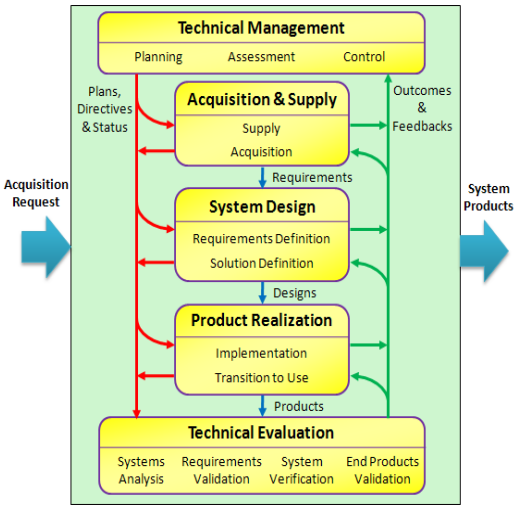


Figure 2: Relationships of processes for engineering a system. (EIA-632, 1999)

In the above engineering processes, there is a sequence of system engineering tasks that prescribe the required activities at each stage of product development. For example, at the system design step it is necessary to define the requirements and to define the solution. To accomplish these tasks, actors must collaborate. Indeed, figure 2 shows that there is interaction between processes indicated by the different arrows. However, the activities in the lifecycle prescribe only tasks, and do not explain how these tasks should be performed collaboratively, and how different perspectives should be integrated to create the required products for these different phases of the lifecycle. Studies revealed that engineers collaborate during about 74% of the time devoted to product development (PTC, 2007) as depicted by figure 3, and yet information on how they collaborate is very limited.

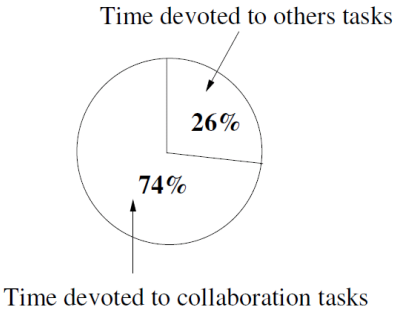


Figure 3: Time devoted to products development by engineers. (PTC, 2007)

According to a study by Frost & Sullivan (Frost and Sullivan, 2006) more than 36% of the performance of an enterprise depends the organization's capacities to collaborate. This represents

more than twice its strategic orientation impact, and more than five times the impact of market influences and technological turbulences. Collaboration increasingly occupies a considerable part in products development; and consequently, improving collaboration is becoming an important factor in productivity.

In light of these observations, this article proposes a method for collaborative requirements elicitation, which distinguishes collaborative aspects from engineering aspects. This distinction aims to help in structuring and organizing collaborative aspects of requirements elicitation to increase the effectiveness of these collaborative task elements. For this purpose, we present a case study on requirements elicitation. Requirements elicitation is considered the most critical and collaborative sub-process of systems engineering according to the EIA-632 standard. We rely on the Collaboration Engineering approach to define a collaboration process that we present in the following subsection.

2.2 Collaboration and collaboration engineering

We approached collaboration issues through a general vision and perspective of Collaboration Engineering which is a new approach to structure and design collaborative work practices.

2.2.1 Collaboration in general

In literature, the terms *collaboration*, *coordination* and *cooperation* are used interchangeably. In some cases the difference between cooperation and collaboration is not made explicit, which leads to confusion.

Cooperation is the process of reasoning and/or pooling of knowledge in the context of problem solving (Soubie et al., 1996). De Terssac and Maggi state that cooperation is a mean to overpass individual limitations (Terssac, Maggi, 1996). Others definitions of cooperation can be found in (Schmidt, Bannon, 1992), (Soubie, 1998). According to Rebetez (Rebetez, 2007) participants in a cooperation process divide the workload between them; each of them has a part to accomplish. The final result is the combination of all individual parts, each participant being responsible for his own work. However, every participant has to interact with the other participants to assure the coherence of the final result.

Coordination is described as all rules of action to structure and harmonize cooperation. Thus, coordination is part of a cooperation activity (Maggi, 1997).

Collaboration is joint effort toward a common goal (Briggs et al., 2003). It is also seen by (Gray, 1989) as a process in which stakeholders with different perspectives of a problem, can constructively explore the differences and can search for solutions that go beyond their own limited visions. Unlike cooperation, in a collaborative context all participants *co-construct* together even if the task can be divided in several other subtasks. In collaboration, all participants intervene in each step so that it is not possible to distinguish individual work in the final result (Rebetez, 2007).

Unlike cooperation, collaboration is often done synchronously even if it is possible in asynchronous way (Potin, 2007) to create synergy and benefit most from interaction.

Communication or interaction is an essential component in cooperation, collaboration and coordination. However, communication is different from the other terms because it does not necessarily have an objective and it can be used as a means to cooperate, collaborate and coordinate.

Through the above definitions we can notice that collaboration is a complex task because it is more difficult to co-construct than to emit or receive information (communication) and more difficult than to assemble information by gathering and coordinating work of individuals (cooperation and coordination).

To overcome challenges of collaboration, one can use group support systems (GSS). Research has shown that under certain circumstances teams using GSS can be more productive than teams which do not use it (Fjermestad, Hiltz, 2001). According to (Den Hengst et al., 2006), it is important that the explicit design and management of collaboration processes is integrated with the development and deployment of collaboration technologies in order to support these collaboration processes. For this, an approach called Collaboration Engineering is advancing (see next subsection).

2.2.2 Collaboration Engineering

Collaboration Engineering aims to design and deploy processes for high value recurring collaborative tasks. Its purpose is also to design these processes so that practitioners can execute them, successfully without the intervention of professional facilitators (Briggs et al., 2003, Vreede and Briggs, 2005). There are three key roles in Collaboration Engineering, which are the role of *facilitator*, *practitioner* and *collaboration engineer*. A facilitator is a professional in collaboration support, who designs and conducts collaboration processes in order to support a specific group in achieving its specific goals (Briggs et al., 2006). A collaboration engineer designs and documents collaborative work practices and deploys them in organisations for practitioners to use. So, a process designed by a collaboration engineer has to be transferable and reusable to enable practitioners to execute it independently (Kolfschoten et al., 2006). A practitioner is a domain expert who learns to execute a particular work practice based on a design made by a collaboration engineer (Briggs et al., 2006). Practitioners do not design collaborative processes, they only execute them. This reduces the skills required, as they focus on part of the facilitation task, and apply these skills only in the context of one specific collaboration process.

Processes are built up as a sequence of facilitation interventions that create patterns of collaboration and predictable group behaviour with respect to a common goal. There are six basic patterns of collaboration according to which processes are organized (Briggs et al., 2006):

- "Generate" allows the move from having few to having more concepts,
- "Clarify" allows moving from having less to having more shared meaning of the concepts under consideration,
- "Reduce" is the opposite of the Generate pattern,
- "Organize" is moving from having less to having more understanding of the relationships among the concepts,
- "Evaluate" is to move from having less to having more understanding of the utility of the concepts priority with respect to goal attainment.,
- "Build Consensus" is to move from having more to having less disagreement on the course of action.

The fundamental building block in Collaboration Engineering is the thinkLet. A thinkLet is defined as "*the smallest unit of intellectual capital to create a pattern of collaboration*" (Briggs et al., 2003). ThinkLets are design patterns and as such have advantages such as transferability, predictability and its use as a shared language among designers and users alike. Section 3 presents the approach we proposed for collaborative system engineering.

3 An approach: Collaborative System Engineering

3.1 Introduction

Initially, we consider engineering and collaboration aspects as merged; this is illustrated by situation *(a)* in figure 4. In order to organize and structure collaborative work, we distinguish the engineering processes from the collaboration processes as presented in situation *(b)*.

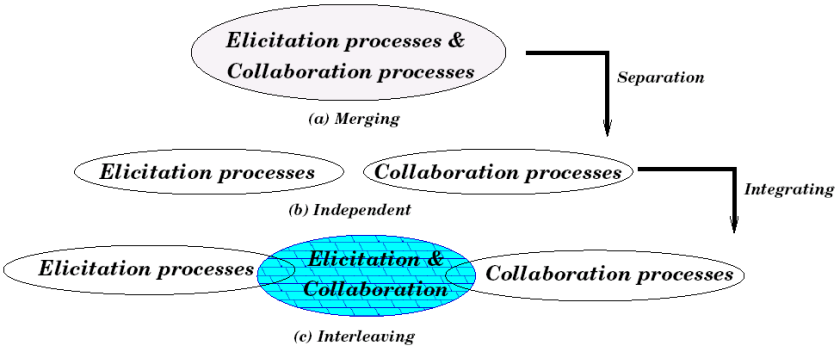


Figure 4: Problem indication

However, the collaboration processes depend on engineering tasks, i.e., engineering processes. This leads to the integration state *(c)* where engineering processes are designed as collaborative processes. We carried out the separation of concerns approach by considering collaboration and engineering as disjoint, to later integrate these two perspectives. The engineering processes we consider are already known and standardized, as the EIA-632 standard. Therefore we will not further define them; we consider them known (EIA-632 1999). However, the collaborative processes associated to this standard have to be identified and defined. We rely on the Collaboration Engineering approach for defining collaboration processes within the EIA-632 standard. This leads us to focus on collaborative engineering tasks. The envisioned collaboration process should fit the standards of systems engineering, because Collaboration Engineering offers reusable, predictable and transferable collaboration processes based on the thinkLet concept. In the next section we propose a model for collaborative engineering, based on the separation of concerns approach described above.

3.2 Collaboration Model

Our vision of collaboration can be summarized as the following formal notation. Here, collaboration is defined as a function:

Collaboration: $(PROC_E \times PROC_C, CON) \rightarrow RESULT$

Where:

$PROC_E$ represents a subset of engineering processes which is essentially included in $PROC_{EIA-632}$,

$PROC_{EIA-632}$ represents the processes of standard EIA-632,

$PROC_C$ is a subset of collaboration processes to be defined,

CON is the set of constraints for the collaboration process,

RESULT is a set of results from collaboration.

This definition forms the basis of our approach; indeed, we see collaboration in the area of engineering as a function that uses conditioned engineering and collaboration processes, which describes the different interactions between participants and the data flow. In this context, $PROC_E$ is one of the EIA-632 engineering processes.

RESULT represents the outcome of the collaboration process. According to the context, it changes. For example, it can be a decision in a collaborative decision making context. In a Requirement Engineering context, ***RESULT*** is a product requirement specification, documented in a specifications book.

Rules that enable efficient collaboration are defined based on constraints on the behavior of the participants, organizations and activities. ***CON*** and ***PROC****c* are the unknowns in the definition above. The following subsections are devoted to determining them.

3.3 Collaboration Constraints: *CON*

Like any process, collaboration processes are subject to constraints for the actors, activities and resources used. For example, constraints to the actors involved are: availability, competencies, and confidentiality. Collaboration activities depend on the requirements engineering activities structuration and the constraints relating to collaboration processes. The number of constraints can increase exponentially, and therefore it is important to make a conscious choice of the constraints that we wish to consider when designing collaboration processes. This choice has to be made to balance the tradeoff between the collaborative goal on one side and the individual stakes of the participants on the other side. This balance needs to be established to ensure that the collaboration process demands an acceptable contribution and sacrifice on the part of the participants, while remaining instrumental to the collaborative goal (Kolfschoten and Vreede, 2009).

In short, we can identify three types of constraints: (1) Actor constraints, these are related to the participants of the collaboration process, their competencies, and their role in the process; (2) Task constraints, these are related to activities performed during process execution and the structuration of the process (see section 3.4.2); and (3) Resource constraints, these are related to the capabilities used during process execution, such as the functionalities of technologies used.

While task constraints are encapsulated in structure of the collaboration process activities, Actor and Resource constraints are expressed through sentences written down in a document. This document is made by collaboration engineer during both stakeholder identification phase and process design phase. This document can be integrated in collaboration process documentation. The constraints to a collaboration process are formulated as collaboration rules, specifying a condition for which a role needs to perform an activity with a specific capability (Kolfschoten et al 2006). A proper definition of collaboration constraints is essential because it is the basis of a well-defined collaboration processes and it allows specification of assumptions for these processes. Depending on the situations, specificities of each situation have to be taken into account in constraints definition.

3.4 Collaboration Processes: *PROCc*

We define a collaboration process in two parts. First, we define interaction flow between actors in collaboration processes and secondly we establish structure of activities.

3.4.1 Interactions

From the preceding description, we can deduce the following model which can be used as a collaboration model for the standard EIA-632. As is described in section 2, this standard distinguishes between three types of actors: the developer, the acquirer and the other stakeholders of the system. For the collaboration process, we use the same set of actors. An actor can represent an individual or a team. Figure 6 summarizes our collaboration model. It is the same logic as classical Input-Process-Output model, and is represented by a graphic, outlining all concepts used in our model. In this model the participants collaborate on a *list of tasks*.

The dotted box represents a collaboration process whit *input* and *output*; the collaboration process is subjected to three types of *constraints* defined in section 3.3. This model is applied to requirements engineering using UML (Rocques, Vallée, 2006) in section 5.

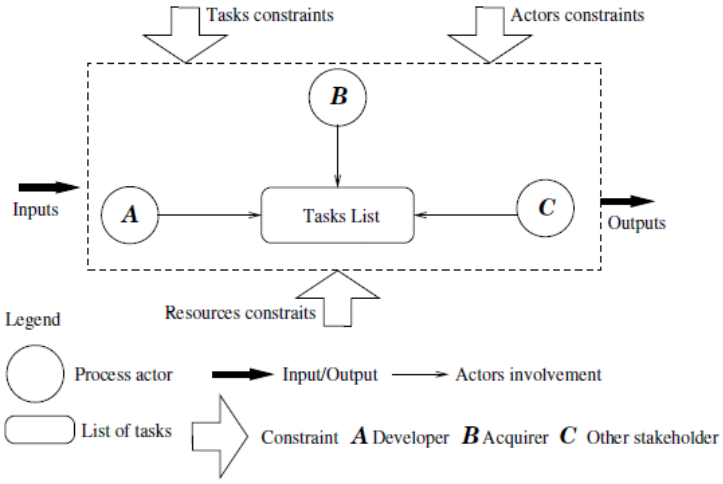


Figure 5: Collaboration model for EIA-632

3.4.2 Behaviour in the Collaborative Processes

This part is focused on the constraints for activities. The interactions and the sequence of activities are in the center of a collaboration model. In our approach, the structuration of a collaboration process is not identical to that of an engineering process. However, they are interdependent. The structuration of the process defines the order of execution for the activities. The thinkLet language allows expressing this order in a facilitation process model as shown in figure 6. This example is from a quick session on requirements elicitation.

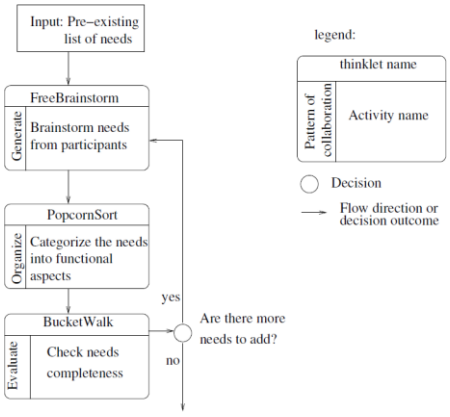


Figure 6: facilitation process model for requirements elicitation.

The session starts with a set of needs. This session continues with other activities supported by thinkLets named ‘FreeBrainstorm’ (used to elicit needs), ‘PopcorSort’ (used to categorize needs to functional aspects) and ‘BucketWalk’ (used for verification) (Briggs et al., 2001). Categories used in the second step have been defined before the collaboration process in the design phase. When the number of generated ideas is not high, it is not necessary to use a reduction thinkLet before the evaluation. In such case, each category contains a small number of ideas which can be evaluated using a thinkLet like BucketWalk. When the brainstorming step produces a large number of ideas, a reduction step is necessary before any evaluation activity.

Above thinkLets are specific techniques to instantiate the generic patterns ‘Generate’, ‘Organize’ and ‘Evaluate’. To allow users to remember thinkLets their names are catchy. However, whatever their denomination is, they are a variation of one of the six patterns of collaboration mentioned in subsection 2.2.2.

3.4.3 Formal definition insight

The function called Collaboration has been defined as following:

(PROC$_E$ x ***PROC$_C$, CON)*** −> ***RESULT***

We consider the thinkLet as a function in present formulation of collaboration definition. So, let P_e ϵ ***PROC$_E$*** and P_c ϵ ***PROC$_C$*** and C ϵ ***CON***

P_e = {task, collaborative elementary task}
{Comp(task$_i$), collaborative composite task composed of task$_i$, i ϵ [1, N]}
{Seq(task$_i$), collaborative sequential task with task$_i$, i ϵ [1, N]}

and

P_c = {thinkLet, collaborative elementary process}
{Comp(thinkLet$_i$), composite collaborative process composed of thinkLet$_i$, i ϵ [1, N]}
{Seq(thinkLet$_i$), sequential collaborative process with thinkLet$_i$, i ϵ [1, N]}

and

Collaboration (P_e, P_c, C) =
{thinkLet(task) if P_e = task}
{thinkLet(Comp(task$_i$)) = Comp(thinkLet(task$_i$)) = Comp(thinkLet$_i$) if P_e = Comp(task$_i$), i ϵ [1, N]}
{thinkLet(Seq(task$_i$)) = Seq(thinkLet(task$_i$)) = Seq(thinkLet$_i$) if P_e = Seq(task$_i$), i ϵ [1, N]}
{C is consigned in thinkLet script (see appendix section)}

From above formulation, we can consider that for an engineering process P_e, a collaboration process P_c, and a set of constraints C, the collaboration is made by the collaboration pattern, the thinkLet, which is also considered as a particular function that can be used to create compound patterns or sequential patterns. Indeed, collaboration patterns can be combined (Kolschoten el al., 2004).

This definition of collaboration requires an analysis of tasks in an engineering process (P_e) and their complexity in order to determine elementary and composite collaborative tasks. When these tasks are identified, collaboration process (P_c) is designed with thinkLets. When designing collaboration patterns for P_e, the thinkLets library is used to select existing thinkLets which are suitable to the engineering process. New thinkLets can be created if there is no existing thinkLet fitting to the situation or when other thinkLets would be more suitable than those already existing (see section appendix section).

In order to link all concepts discussed above, we propose a conceptual model which is described below.

4 A conceptual model

As mentioned previously, we distinguish between two kinds of processes: Engineering Processes and Collaboration Processes. The engineering processes are defined by a standard; therefore we do not redefine them. The collaboration processes have to be defined with the use of Collaboration Engineering. For this purpose, thinkLets are a fundamental concept in our model. A major conceptualization of thinkLets, based on an object oriented approach is presented by (Kolfschoten et al., 2006), which we used for our model.

Figure 7 shows a class diagram with the different concepts in collaborative requirements engineering: **Requirement** is a technical definition of a **Constraint** or a **Need** even through it is not always possible to translate any *constraint* and *need* into a *requirement*. Requirements are sorted into **Categories. Categories** can have sub-categories, they are recorded in a **Specifications book**. Requirements are defined according to an **Engineering Process,** which is a sequence of **Tasks.** Each collaborative *task* has **Activities** supported by one or more **ThinkLets**. A **Collaboration Process** is constructed with *thinkLets* and involves at least three **Participants**.

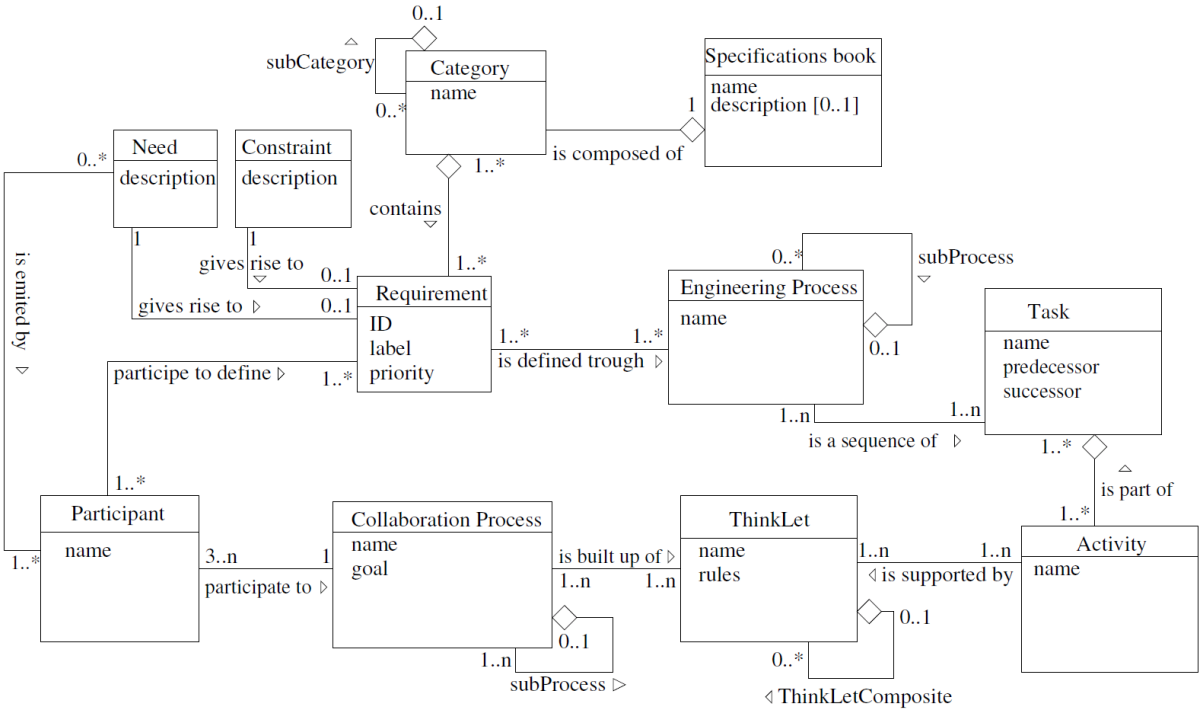


Figure 7: Class model of engineering and collaboration processes.

The model in Figure 7 formalizes the phase of Engineering and Collaboration processes integration phase as previously presented in figure 4, page 7. It does so by mapping tasks and thinkLets according to the model proposed in section 3. In other words, a *ThinkLet* has to allow collaborative execution of a *Task*. Indeed, a *ThinkLet* encapsulates the information describing how to collaboratively perform a *Task*. This description is part of ThinkLet components and it is called a *Script* in Collaboration Engineering terminology (see appendix section). Since a ThinkLet is a design pattern, it can be adapted to the context by changing the script, while maintaining the fundamental principles of the collaborative technique. It is also possible to propose a new ThinkLet if no existing ThinkLet is suitable to the current situation. Figure 7 is the conceptualisation of the different components of the collaborative requirements elicitation process. This model offers a clear overview of the concepts involved and their interaction.

As previously mentioned, we have chosen Requirements Engineering as the context for our experiment. In section 5 we present the experiment and describe the different processes involved. We also present an UML interaction diagram to describe the experiment.

5 Toward a Collaborative Working Environment for Requirements Engineering (SEC-WE)

5.1 Modelling Collaborative Actors

Here, we illustrate our approach to the requirements definition processes (EIA-632, 1999), (Coulin, 2007). Figure 8 describes process ***P:*** '*Collaboration process for requirements*

engineering' with four sub-processes ***P1***, ***P2***, ***P3***, ***P4***. These sub-processes represent '*Collaboration process for identification and collection of needs*' for example Brainstorming (Osborn, 1948) and Storytelling (Acosta, Agerrera, 2006), '*Collaboration process for needs transformation into technical requirements*' for example the specification template named *Volere* (Robinson, 2007), '*Collaboration process for requirements validation*' and '*Collaboration process to record requirements*' (EIA-632, 1999) respectively.

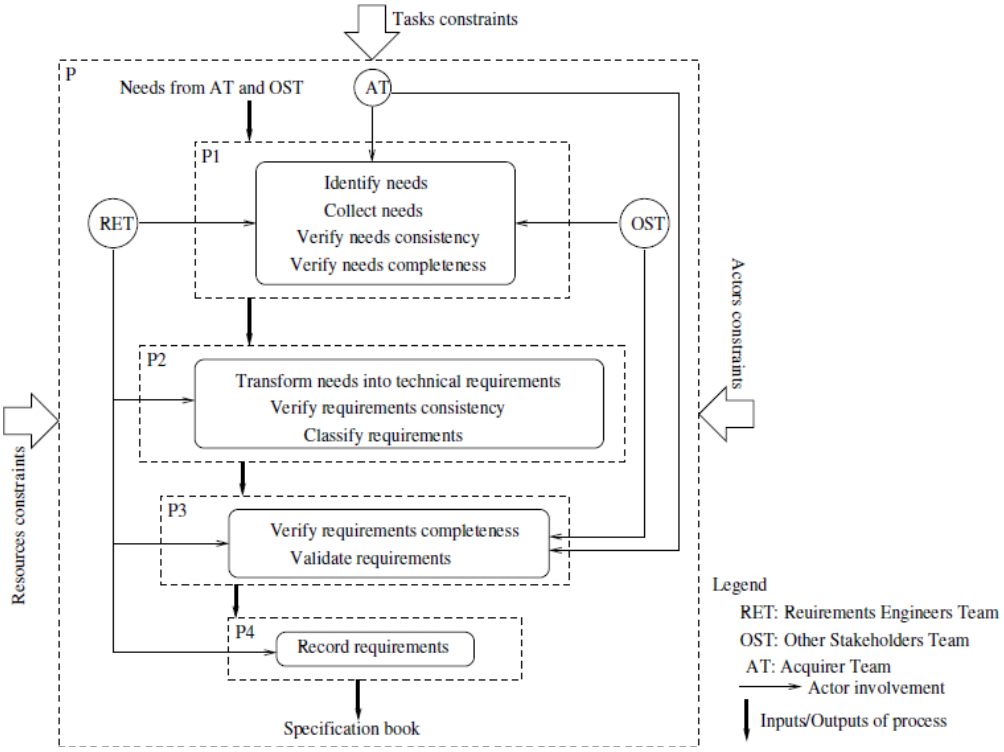


Figure 8: Collaborative Requirements Engineering process.

The recording process starts with writing down the specified requirements on paper or using an automated tool like Truereq (TrueReq, 2004). The involvement of actors in this process is indicated by an arrow from the actor directed towards the tasks of the process. The arrows directed downwards symbolize the data flow (input and output). In particular, input is needed from the acquirers' team (***AT***) and the other stakeholder team (***OST***). The needs are identified and collected in collaboration with the requirements definition team (***RET***). A set of needs resulting from the first process (***P1***) is used as input for the second process (***P2***). In this process the needs are transformed into technical requirements. In the third process (***P3***), the technical requirements are validated by all actors. Finally the requirements are recorded by the requirements definition team (***P4***). The output of the process ***P*** is a *specifications book*.

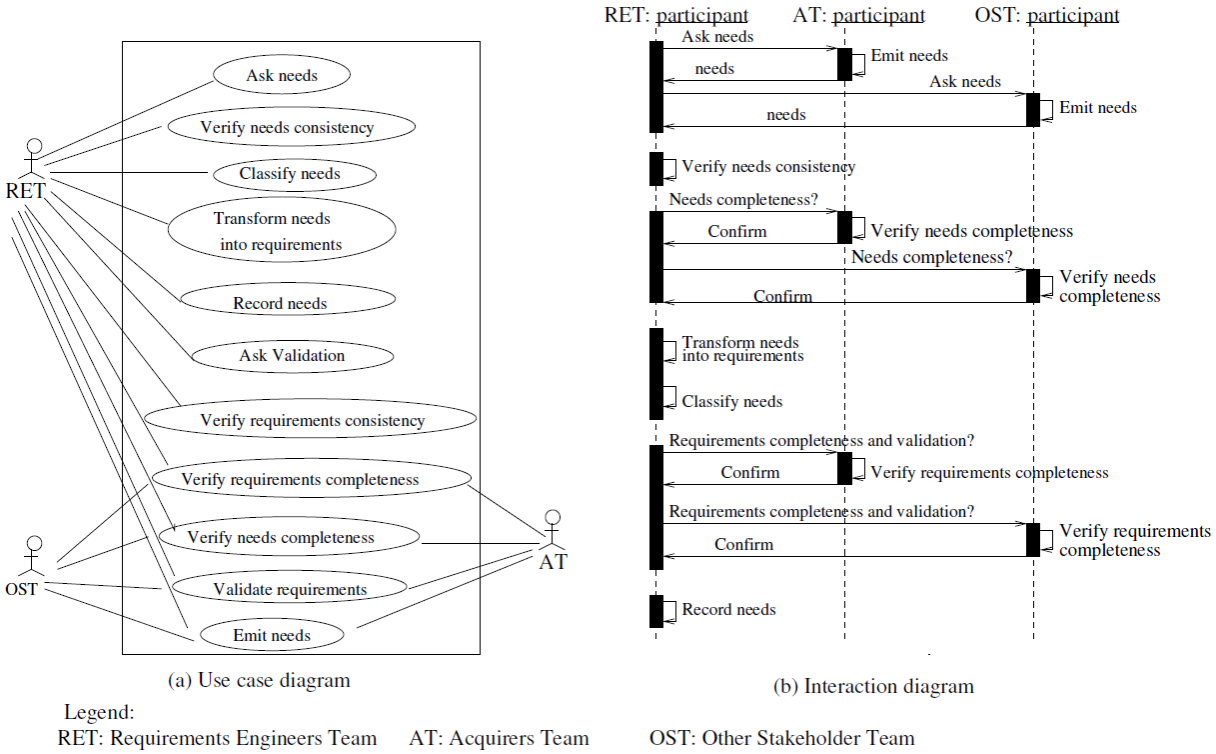
We use the formal modelling language UML to model the process (***P***). We show interactions between the participants and we show data exchanges.

5.2 Interaction Model with UML

Figure 9a shows the actors and the activities they must execute in the requirements engineering process. In figure 9b we present an interaction diagram to describe how actors interact in the collaborative requirements definition process. Like a use case diagram, an interaction diagram is a high level representation of the requirements engineering process, hiding more detailed interactions.

In Collaboration Engineering terminology, *AT*, *OST* and *RET* are participants of a collaboration process. One of the participants of the *RET* can be a practitioner and execute the requirements

elicitation process (P1 in figure 8) designed by a collaboration engineer; an expert in collaboration support. We will describe this process in section 6.



(a) Use case diagram

(b) Interaction diagram

Legend:
RET: Requirements Engineers Team AT: Acquirers Team OST: Other Stakeholder Team

Figure 9: High level use case and interaction diagram for the requirements engineering process

As mentioned previously, we mainly focus on the elicitation phase of the Requirements Engineering process. We will present the case study on the elicitation process and its results in the following subsections.

5.3 Requirements elicitation process

In the Requirements elicitation process, we identify two main phases; the preparation phase and the needs identification phase. This is shown in figure 10. These phases are described in following subsections.

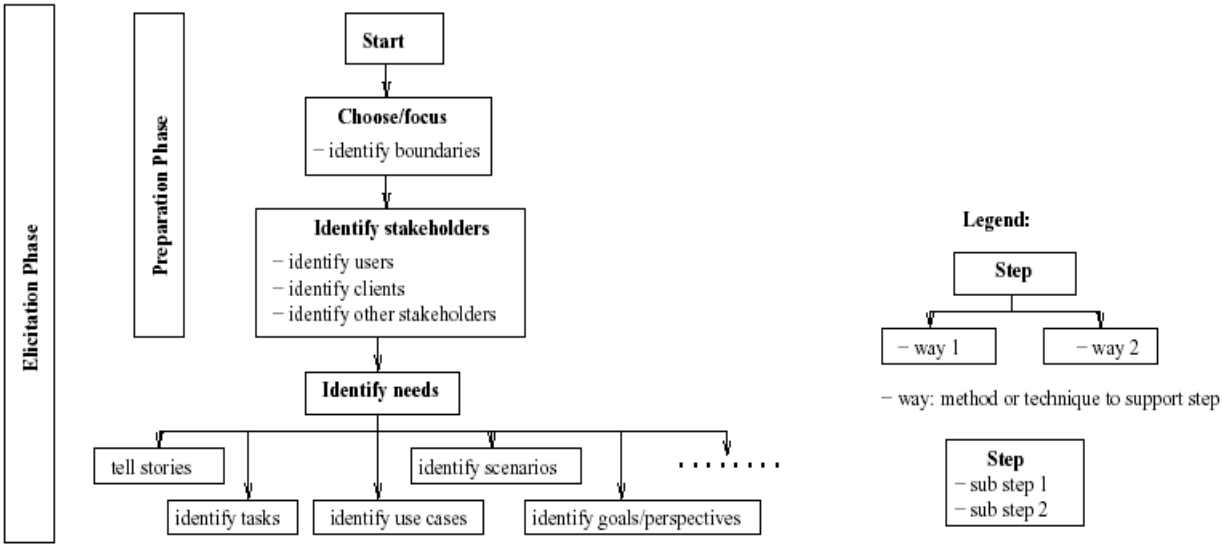


Figure 10: Requirements Elicitation phase.

5.3.1 Preparation phase

Requirements Elicitation starts with a preparation phase. In this phase it is important to involve the relevant stakeholders in the requirements elicitation process. We can identify a *Start* step in which participants introduce themselves. In this step first informal social contacts are established and participants explain their role and their expectations of the activity. Den Hengst et al. (Den Hengst et al., 2004) name this step *Warm-up*. Next, we have to *choose/focus on* boundaries for the scope of the requirements elicitation effort. This scope describes the environment in which the target system will be used, including all related domains. This step is very important in the elicitation process because the following activities depend on how the boundaries are chosen (Nuseibeh, Easterbrook, 2000). The step *identify stakeholders* consists of identifying stakeholders to represent different aspects of the system requirements within the boundaries determined in the previous step. Finally the preparation phase requires analysis of the domain and organization(s) related to the system in order to gain a basic understanding and common ground (Machado et al., 2008).

5.3.2 Identify needs phase

Once the preparation phase is completed, the specific stakeholders and the boundaries of the concerned domain are identified. The next step consists of selecting the methods and techniques to collect, capture and gather the needs from the different stakeholders. Figure 1 shows a non-exhaustive list of methods and techniques to support the need identification step. Among them are Dialogue (Hoa, 2007), Group Storytelling (Acosta, Agerrera, 2006) and Brainstorming (Osborn, 1948) which are all collaborative approaches that can be used to brainstorm needs. The choice of methods and techniques depends on the time and resources available, and depends on the kind of information to be elicited (Nuseibeh, Easterbrook, 2000). In addition, every technique has some limitations (Goguen, Linde, 1993). A combination of these techniques might therefore help the group to approach the identification from different perspectives and to trigger different creativity and thinking patterns (Knoll and Horton, 2010)

In the following section, we present the requirements elicitation case study we did to evaluate our approach.

6 Case study and prototype

We performed a case study with two groups of students in order to evaluate and compare a couple of need elicitation methods from literature. The first group consisted of 44 students, the second group of 14 students.

6.1 Approach

According to the conceptual model we presented in section 4, we first considered the engineering process which is an elicitation process in this case. Then, we selected some existing collaborative methods for the requirements elicitation process. In our workshops we chose Storytelling (Acosta, Agerrera, 2006), and Dialogue (Hoa, 2007), which we conceptualized as thinkLets and the existing thinkLets FreeBrainstorm and LeafHopper (Briggs and Vreede, 2001). We propose two new thinkLets to support these two methods which are named after the method they support (story telling & dialogue). The new thinkLets are presented in detail in the appendix. Although all these methods allow for the elicitation of needs, the specific tasks performed vary per method. Once the thinkLets are identified, we design a collaborative process to support collaborative need elicitation. Figure 11 shows the collaborative processes designed by a Collaboration Engineer for both sessions. The order in which the thinkLets were placed was deliberate. In brainstorming literature it is described that people will first share those ideas that are on the top of their mind. Later, they will benefit from stimuli that motivate them to think about new ideas (Santanen et al 2004).

We used ThinkTank (GroupSystems, 2006) developed by GroupSystems as a collaboration tool to support joint need elicitation. ThinkTank is a web-based tool supporting the main activities of a collaboration process. It allows users to brainstorm, vote, organize and revise ideas in a collaborative mode. ThinkTank also provides functionality to capture and store a full electronic meeting record.

Both sessions were on need elicitation for the design of an Electronic Medical Record (EMR) system.

Start: the facilitator started the session by introducing herself and by explaining the purpose of the session. The session was anonymous, i.e., each participant could connect to the GSS and contribute under a fictive ID. This allowed the participants to be free in expressing their ideas.

FreeBrainstorm: in this stage, participants elicited needs, and responded, reflected or elaborated on needs proposed by others. During about 15 minutes, participants produced ideas related to system requirements. The facilitator often had to intervene to further motivate and challenge the participants in expressing their ideas. The facilitator removed ideas which did not contribute to need elicitation, like "It is true" and "I agree". At the end of this step, we had a list of system requirements to which all participants had contributed.

StoryTelling: in this step, participant has to write a short story or experience to explain the problem that is to be addressed using system, and to elaborate on the stories from others. The facilitator asked the participants to think from the perspective of a stakeholder by telling a story of what happened, asking them to come up with a use case from that stakeholder's perspective. In other words, they had to tell a story from a specific perspective on what the system should do, and what kind of problems / challenges this stakeholder could encounter when interacting with the system. Then, the facilitator asked them to read the stories from other participants and to try to elaborate on them. After 15 minutes of story telling, the facilitator asked them to summarize the stories in terms of needs only. In the summary, the needs described in the stories were extracted.

Dialogue: in this step, a role is assigned to a participant. In this role participants are asked to engage in a question-answer dialogue. Participants asked questions and answered questions from the perspective of their particular role. The facilitator asked the participants to exchange needs from different stakeholder perspectives. This step lasted for about 10 minutes; based on the resulting dialogues the summary of needs was updated again.

LeafHopper: in this step, participant has to elicit needs based on different functions of the system. People contributed needs based on functional categories related to the system (an Electronic Medical Record) such as e.g. enclosure of medical knowledge, collecting data for performance management, automating financial aspects, etc.). About 8 minutes later, when the facilitator noticed that the participants were not contributing new needs any more she asked them to stop this step.

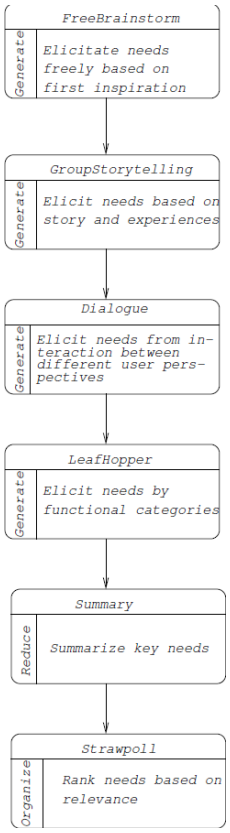


Figure 11: Repeatable process for requirements elicitation

Summary: After each step, requirements were added to a summary in which the requirements were collected. This summary was made initially by the facilitator, and later participants added needs to the summary. The Dialogue, StoryTelling and LeafHopper were used to help the group to get inspiration for new ideas. However, these steps did not lead to concise needs. Therefore, needs were elicited based on the brainstorm and added to the summary. The method for this was rather ad-hoc and needs refinement to be captured as a thinkLet, it had elements of both expert choice and fast focus. Once all different methods had been used, the facilitator asked the participants to summarize all needs and to check the resulting set of needs for completeness. Subsequently, the participants answered questions which aimed to further confirm the completeness of the needs. In order to get more complete requirements, the facilitator tried to motivate the participants to elaborate. Some quotes used for motivation are: “are security aspects covered in the needs?”, “are technical aspects covered in the needs?”. This plenary group discussion lasted for approximately twenty minutes.

StrawPoll: The participants had to select the ten most important needs. To do so, the StrawPoll thinkLet was used, supported by the voting functionality of the GSS. This produced a list of needs, ordered according to participant perception of importance. Finally, the participants selected a sub set of about 10 key needs which mostly correspond to the top needs on the list. In our case study, subsequent steps to refine needs to requirements and to further validate the resulting set of requirements and build consensus around them were not performed. These would be important further steps, and a collaboration process needs to be developed for these steps.

Both sessions were executed in the way described above. There were some differences in step duration between the two sessions. The reason for this was the difference in group size. The first group had more participants than the second group. The steps of the first group took more time because there were more contributions and more discussion.

After the sessions, the students who participated filled out our questionnaires for qualitative and quantitative evaluation of the methods used. In the next subsection we present the results of these evaluations.

6.2 Results

6.2.1 Comparing the requirement elicitation methods

Table 1 presents the results of the questionnaire from the participants of the first and second workshops. In the first session 20 requirements were summarized, in the second session 21 requirements were summarized. 44 students participated in the first workshop, and from them we collected 34 usable questionnaires. In the second workshop 14 students participated, and from them we collected 10 usable questionnaires. So, we had 44 usable questionnaires from both first and second sessions.

1= strongly disagree 7= strongly agree		
	Means (n=44)	*Standard deviations (n=44)*
a. The method was easy to learn		
Brainstorming	6,00	1,17
Dialogue	5,32	1,36
Group Storytelling	4,93	1,42
LeafHopper	5,43	1,40
b. The rules & instructions of method were clear		
Brainstorming	5,55	1,23
Dialogue	4,61	1,28
Group Storytelling	4,34	1,40
LeafHopper	4,98	1,25
c. The method was efficient, it took little effort/time for maximum results		
Brainstorming	5,00	1,60
Dialogue	4,39	1,46
Group Storytelling	4,11	1,48
LeafHopper	4,89	1,54
d. The method was helpful to express needs		
Brainstorming	5,11	1,35
Dialogue	4,80	1,27
Group Storytelling	4,61	1,48
LeafHopper	5,23	1,35
e. The requirements created with this method were useful for our design project		
Brainstorming	5,27	1,50
Dialogue	4,84	1,26
Group Storytelling	4,57	1,39
LeafHopper	5,23	1,28
f. The requirements generated with this method were of high quality		
Brainstorming	4,16	1,36
Dialogue	4,27	1,29
Group Storytelling	4,14	1,22
LeafHopper	4,98	1,39
g. This method helped to ensure completeness of the requirements		
Brainstorming	4,27	1,60

Dialogue	4,41	1,42
Group Storytelling	4,27	1,34
LeafHopper	4,82	1,39
h. This method helped to ensure rigor of the requirements		
Brainstorming	4,25	1,38
Dialogue	4,36	1,30
Group Storytelling	4,50	1,20
LeafHopper	4,98	1,32

Table 1: Summary of the results from the first and second session.

The respondents were students in a master program on System Engineering, Policy Analysis and Management. They had a moderate knowledge of Requirements Engineering methods and techniques. Therefore, there was value in asking these students about the quality, completeness, etc. of requirements. In addition, the evaluation aimed to compare methods. Responses were not absolute, but they present interesting first insights.

Concluding we find that students evaluated FreeBrainstorm and LeafHopper as more practical, but when regarding, rigor and completeness, participants recognized added value from the more elaborate methods; Dialogue and StoryTelling. All thinkLets scored high on ease of use. For clarity of the method FreeBrainstorm (5,55) and LeafHopper (4,98) scored higher. For efficiency and helpfulness of the method as well (FreeBrainstorm (5,00, 5,11) and LeafHopper (4,89, 5,23)).

In terms of usefulness and quality of the resulting requirements, LeafHopper scored highest, (5,23 and 4,98) this could be partly because it was the last thinkLet, but the effect of that could also have been that participants were "empty" and had no more additional requirements. The high score for usefulness and quality could also have been because the thinkLet asked for more specific requirements, focussing the group on functional aspects of the design. For completeness and rigour, high scores were also found for LeafHopper (4,82, 4,98), but in these questions Dialogue (4,41, 4,36) and StoryTelling (4,27, 4,5) also scored higher, indicating that participants valued these more elaborate methods to ensure a full exploration of the needs from different stakeholder perspectives.

Limitations of this case study are the following:

- Students participated in the study, while they did use the requirements for a future assignment and thus had a stake in their quality, a field study would be better to assess the usefulness and quality of the requirements
- We had no objective evaluation of the quality of the requirements. The case was based on a real case, but never the less had a fictive element. Therefore real assessment of the quality of requirements was not possible.
- The order of the thinkLets was fixed, a different sequence might have had different effects.
- ThinkLets were evaluated after the entire process, this allows comparison on one hand, but also could have caused less accurate evaluation, especially for the earlier thinkLets.

6.3 Requirements specification prototype

To support the subsequent processes (processes P2, P3 and P4, see figure 8), we need a more specific tool, which supports the transformation of needs to requirements, their organization in a specification book and their verification. For this purpose we developed SPECJ.

SPECJ is a web based tool for collaborative requirements specification. The figure 12 shows its architecture which is based on MVC2 (Model, View and Controller). The *Model* represents entities used by an application, these entities are collected in a database. The *View* represents the interface of the application. The *Controller* is the component of application which replies to input from user, i.e., the *Controller* translates the events from *View* to modifications of *Model*, and then defines the way the *View* has to react to these events.

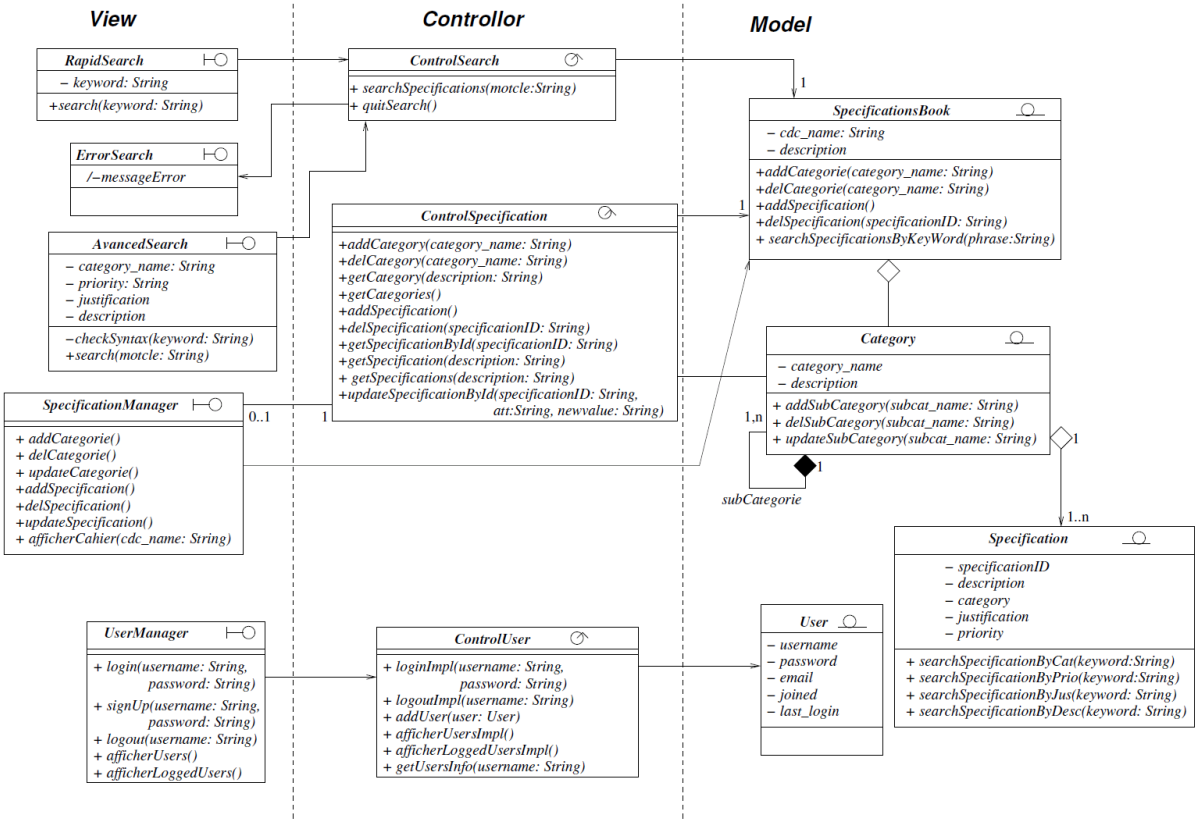


Figure 12: Functional Architecture of SPECJ

The main functionalities offered are: create, delete and update specification, look for specifications based on criteria (category, description, priority and rational), check the specifications list and their categories, check the list of all registered users (who opened an account), check the list of logged in users, check and display the specifications book, etc.

The figure 13 shows the SPECJ interface with three active users specifying requirements for their category, description, rational and priority. Figure 14 presents an example of a specification book made with SPECJ. Figure 15 shows the categories of the specifications book with their subcategories and descriptions.

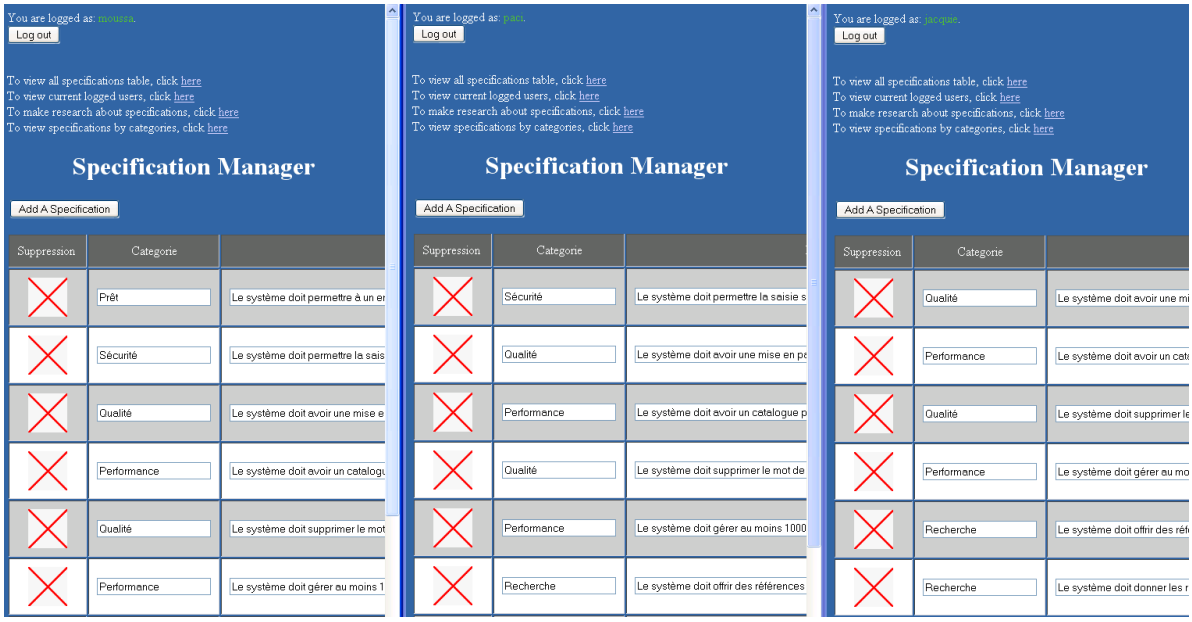
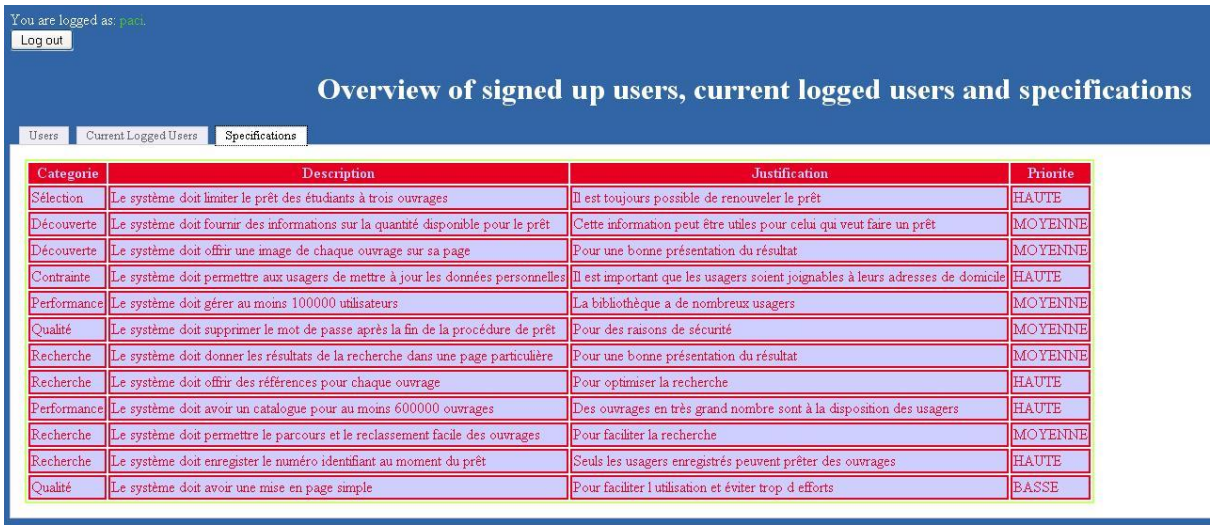


Figure 13: Collaborative requirements specification



Categorie	Description	Justification	Priorite
Sélection	Le système doit limiter le prêt des étudiants à trois ouvrages	Il est toujours possible de renouveler le prêt	HAUTE
Découverte	Le système doit fournir des informations sur la quantité disponible pour le prêt	Cette information peut être utiles pour celui qui veut faire un prêt	MOYENNE
Découverte	Le système doit offrir une image de chaque ouvrage sur sa page	Pour une bonne présentation du résultat	MOYENNE
Contrainte	Le système doit permettre aux usagers de mettre à jour les données personnelles	Il est important que les usagers soient joignables à leurs adresses de domicile	HAUTE
Performance	Le système doit gérer au moins 100000 utilisateurs	La bibliothèque a de nombreux usagers	MOYENNE
Qualité	Le système doit supprimer le mot de passe après la fin de la procédure de prêt	Pour des raisons de sécurité	MOYENNE
Recherche	Le système doit donner les résultats de la recherche dans une page particulière	Pour une bonne présentation du résultat	MOYENNE
Recherche	Le système doit offrir des références pour chaque ouvrage	Pour optimiser la recherche	HAUTE
Performance	Le système doit avoir un catalogue pour au moins 600000 ouvrages	Des ouvrages en très grand nombre sont à la disposition des usagers	HAUTE
Recherche	Le système doit permettre le parcours et le reclassement facile des ouvrages	Pour faciliter la recherche	MOYENNE
Recherche	Le système doit enregister le numéro identifiant au moment du prêt	Seuls les usagers enregistrés peuvent prêter des ouvrages	HAUTE
Qualité	Le système doit avoir une mise en page simple	Pour faciliter l utilisation et éviter trop d efforts	BASSE

Figure 14: Specification book shaped in table

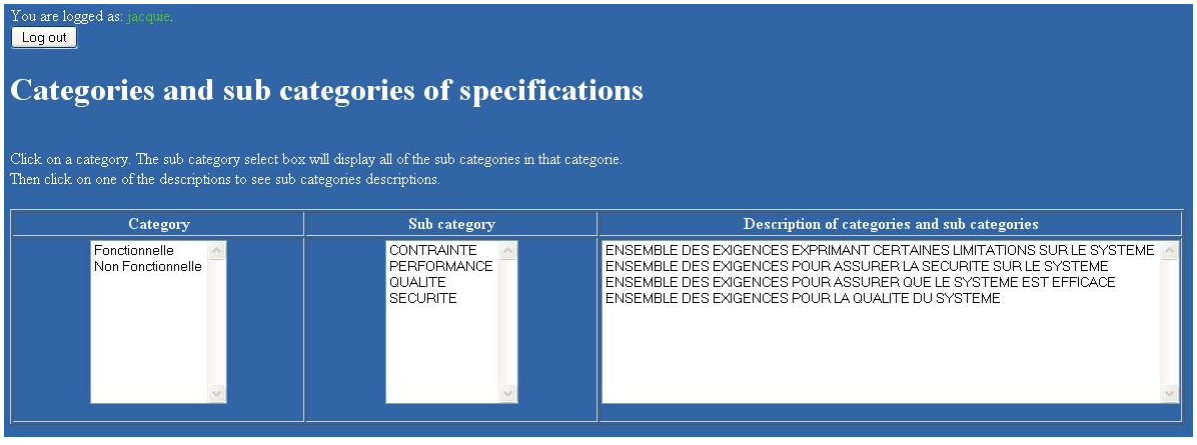


Figure 15: Categories, sub-categories and descriptions

As previously announced, SPECJ allows to support the other steps of Collaborative requirements definition process. ThinkTank supports elicitation process, and SPECJ supports the transformation of needs into requirements, the verification and validation of requirements, and finally the requirement recording.

7 Conclusion

This paper suggests an approach for Requirements Engineering based on separation of engineering and collaboration concerns. We firstly considered the engineering processes provided by the standard EIA-632 and we proposed a collaboration model based on this standard. This model of collaboration is formally defined as a function of the EIA-632 engineering processes, based on guidelines for collaboration process design from the Collaboration Engineering literature. The result of this function depends on the process at hand, in particular a requirements document, in a requirements definition context. We also defined a conceptual model of collaborative requirements engineering using the thinkLet concept provided by the Collaboration Engineering approach. In order to evaluate the approach, we did two case studies in a requirements elicitation context. We have used four different methods for requirements engineering, based on new and existing thinkLets.

It is important to note that only needs were generated from these thinkLets and that needs have to be transformed into requirements. Each need can give rise to zero or several requirements because needs are high level requirements. The transformation of needs into requirements is made in the requirements specification phase, which is out of the scope of this study even though our prototype SPECJ partly supports this phase. To transform needs into requirements, a shared understanding between users and engineers needs to be established, which is a critical and challenging task. Further research is required to develop a collaborative approach to this step. For this the easy win win negotiation (Boehm et al 2001) process could be a basis.

The contribution of this article is the proposition to use a collaboration model in which both the System Engineering and Collaboration Engineering approach is integrated. System Engineering provides engineering processes whereas Collaboration Engineering provides collaboration processes. This contributes to the originality of our approach. A class diagram illustrates this conceptualization. We also contributed by creating two new thinkLets that enrich the thinkLet library. Equally, we presented the results of two case studies on a collaborative requirements elicitation process to evaluate our approach. From the results of the questionnaire we can conclude that the LeafHopper thinkLet was the most effective way to elicit needs. We also developed the prototype SPECJ supporting following step of generation, especially collaborative documenting. It is an implementation of the model presented in section 4. Indeed, SPECJ architecture is based on the same concepts of this model. However, we think that the combination of methods helps the participants to look at the system requirements from different perspectives, which will helped to increase completeness and rigour of the requirements. Further research is required to understand the added value and combinatory value of the different thinkLets for requirements elicitation. Additionally, field experiments with both the combination of thinkLets to elicit requirements would be next step in this research. Further development of the SPECJ application to improve its robustness and its function as a GSS for requirements elicitation would be another next step.

References

Acosta C.A. and Agerrera L. A. (2006): *Supporting the Collaborative Collection of User's Requirements*. Proceedings of International Conference of Group Decision and Negotiation (GDN) 2006, Germany.

Boehm B., Bose P., Horowitz E., Lee M.J., (1994): *Software Requirements as Negotiated Win Conditions*. In Proceedings of International Conference on Requirements Engineering, IEEE Press, Piscataway, N.J., 1994.

Boehm B., Grünbacher P., Briggs R. O. (2001): *Developing Groupware for Requirements Negotiation: Lessons Learned*. IEEE Software, 18.

Briggs R. O., de Vreede G.-J., (2001): *ThinkLets, Building Blocks for Concerted Collaboration,* Delft, Delft University of Technology.

Briggs R. O., de Vreede G.-J., Nunamaker Jr. J. F. (2001): *ThinkLets: Achieving Predictable, Repeatable Patterns of Group with Group Support Systems (GSS),* Group Systems.

Briggs, R. O., de Vreede G.-J., Nunamaker Jr. J. F. (2003): *Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems*. In Journal of Management Information System, Springer vol. 19, No. 4, pp. 31-64. 2003.

Briggs R.O., Kolfschoten G. L., de Vreede G. J., Dean D. L. (2006): *Defining Key Concepts for Collaboration Engineering*. In Proceedings of12th Americas Conference on Information Systems, Acapulco, Mexico August 4th-6th 2006.

Coulin C. R. (2007): *A Situational Approach and Intelligent Tool for Collaborative Requirements Elicitation*. Technical report, Phd thesis, University of Sydney (Australia) and University of Toulouse (France), 2007.

Den Hengst M., van de Kar E. and Appelman J. (2004): *Designing Mobile Information Services: User Requirements Elicitation with GSS Design and Application of a Repeatable Process*. In Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

Den Hengst M., Dean D.L., Kolfschoten G., Chakrapani A. (2006): *Assessing the quality of collaborative processes*. In Proceedings of the 39th annual Hawaii International Conference on System Sciences, Volume 1, 04-07 Jan. 2006 Page(s): 16b - 16b.

EIA-632 (1999): *EIA-632: Processes for systems engineering*. Electronic Industries Alliance, ANSI/EIA-632-1998 Approved: January 7, 1999.

Essame D. (2002): *La méthode B et l'ingénierie système. Réponse à un appel d'offre*. Technical report, IUT-Nantes, Université de Nantes, 2002.

Fjermestad J., Hiltz S. R. (2001): *A Descriptive Evaluation of Group Support Systems Case and Field Studies*. In Journal of Management Information Systems, 17**,** pages 115-159.

Frost and Sullivan (2006): *Meetings around the world: The impact of collaboration on business performance*. Technical report, Frost and Sullivan, Verizon Business and Microsoft, 2006.

Goguen J. and Linde C. (1993): *Techniques for Requirements Elicitation*. First IEEE International Symposium on Requirements Engineering (RE'93), San Diego, USA, 4-6th January 1993, pages 152-164.

Goguen J. A, (eds) (1994): *Requirements engineering as the reconciliation of social and technical issues*. Oxford Univ., Computing Lab., UK, Pages 165–199.

Gray B. (1989): *Collaborating: Finding common ground for multiparty problems*. San Francisco, CA: Josses-Bass Publishers.

GroupSystems (2006): *Groupsystems solutions*, http://www.groupsystems.com, accessed on September 2008.

Hoa N. (2007): *Goal Management for a Multisession Dialogue*. Information Technology Convergence ISITC 2007, 23-24 November 2007, page(s): 301 – 305.

IEEE (1990): *IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology*.

Knoll S. and Horton, G. (2010): *Changing the Perspective: Improving Generate thinkLets for Ideation.* In Proceedings of the Hawaii International Conference on System Science, IEEE Computer Society Press, Kauai (HI).

Kolfschoten G., Appelman J.H., Briggs R.O., de Vreede G.J. (2004): *Recurring patterns of facilitation interventions in GSS sessions*. In proceedings of Hawaii International Conference On System Sciences. IEEE Computer Society Press, Los Altos.

Kolfschoten G., Briggs R. O., de Vreede G.-J., Jacobs P. H.M., Appelman J. H. (2006): *A conceptual foundation of the thinkLet concept for Collaboration Engineering*. In Journal of Human-Computer Studies 64, pp. 611-621.

Kolfschoten G., de Vreede G.-J. (2009) *A Design Approach for Collaboration Processes: A Multi-Method Design Science Study in Collaboration Engineering*. In Journal of Management Information Systems, in press.

Konaté J., Sahraoui A.E.K. (2007): *Collaboration in Requirements Engineering Process*. In Proceedings: 13th International Conference on Concurrent Enterprising, Sophia-Antipolis, France, 04-06 June 2007.

Kotonya G., Sommerville I. (1998): *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Great Britain, 1998.

Machado R. G., Borges M. R. S. and Gomes J. O. (2008): *Supporting the System Requirements Elicitation through Collaborative Observations*. In Proceedings of the 14th Collaboration Researchers' International Workshop on Groupware, Omaha, Nebraska, September 14-18, 2008.

Maggi, B. (1997): *Coopération et coordination dans et pour l'ergonomie: quelques repères*. Performances Humaines et Techniques, Hors série.

Monford V., Goudeau S. (2004): *Web Services et Interopérabilité des SI*. Dunod, Paris, 2004.

Nunamaker J. F. J., Briggs R. O., Mittleman D. D., Vogel D., Balthazard P. A. (1997): *Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings*. Journal of Management Information Systems, 13**,** 163-207.

Nuseibeh B. and Easterbrook S. (2000): *Requirements Engineering: A Roadmap*. Proceedings of International Conference on Software Engineering (ICSE-2000), 4-11 June 2000, Limerick, Ireland, ACM Press.

Osborn A. (1948) http://fr.wikipedia.org/wiki/Alex_Osborn - cite_ref-0*Your Creative Power*, 1948, p. 265; Applied Imagination, 1957, 1963, p. 151.

Potin Y. (2007): *Travail Collaboratif: Quand la distance permet le rapprochement*. http://www.creg.ac-versailles.fr/article.php3?id_article=206, accessed in April 2009.

PTC (2007): *Conception d'un système de collaboration*. Parametric Technology Corporation. http://www.ptc.com. accessed on January 2007.

Rebetez C. (2007) : *Concepts de base*. http://tecfa.unige.ch/staf/staf-i/rebetez/staf-11/periode4/, accessed on Oct. 2007.

Rocques P., Vallée F. (eds) (2004) : *UML 2 en action, De l'analyse des besoins à la conception J2EE*. EYROLLES.

Sahraoui A.E.K, Buede D. M., Sage A. P. (2008): *Systems Engineering Reasearch*. In Journal of Systems Science and Systems Engineering, Volume 17, Number 3, pages 319-333, September 2008.

Santanen E.L., Vreede G.J.d., Briggs R.O.(2004): *Causal Relationships in Creative Problem Solving: Comparing Facilitation interventions for Ideation*. In Journal of Management Information Systems, 20, pages 167-197.

Schmidt K., Bannon L. (eds) (1992): *Taking CSCW seriously*. In Computer Supported Cooperative Work (CSCW), Springer Netherlands Volume 1, Numbers 1-2 / March, 1992, pages 7-40.

Sommerville I., Sawyer P. (eds) (1997): *Requirement Engineering: A Good Practice Guide*. John Wiley & Sons, Great Britain.

Soubie J. L. (1998): *Modelling in Cooperative knowledge based systems*. Third International Conference on the Design of Cooperative Systems COOP'98, Cannes, France. INRIA, France, pp. 45–48, May 1998.

Soubie J. L., Buratto F., Chabaud C. (Eds.) (1996) : *La conception de la coopération et la coopération dans la conception*. In G. de Terssac and E. Friedberg, Coopération et conception. Octarès, Toulouse, France, 1996.

Terssac (de) G., Maggi B. (1996) : *Autonomie et conception*. In TERSSAC (de) G. et FRIEDBERG E. (Eds), Coopération et conception, Toulouse, Editions Octarès, 1996.

TrueReq (2004): http://www.truereq.com, accessed on Sept. 2008.

Vreede G. J. D., Briggs R. O. (2005): *Collaboration Engineering: Designing Repeatable Processes for High-Value Collaborative Tasks*. In Proceedings of Hawaii International Conference on System Science. Los Alamitos, IEEE Computer Society Press.

Wiegers K.E. (2000): *Process Impact*. http://www.processimpact.com. Accessed on April 2008.

Appendix

Storytelling: generate

Choose this thinkLet

- *to create collective story from experiences of all participants*
- *to make link between participant problems and real-world situations*
- *when people need to understand the context of the contribution*
- *to gather rich and detailed contributions based on experience*

Overview

In this thinkLet the group members tell stories of their experiences.

Inputs

Clear understanding of the purpose of storytelling

Outputs

A common and collective story created from experiences of all participants. This thinkLet is based on GSS allowing people to speak and write their stories. Each team member shares a story with his/her experience to elicit one perspective of a common problem. Each member can elaborate and reflect on stories from other participants. Members can also ask questions for clarification. The stories are merged to create a collective story.

How to use Storytelling

Setup

1. *Create story pages in GSS*
    a. *each participant sees a page and can add part of the story*
    b. *one or two extra pages, depending on the group size*
2. *Present the problem that is the focus points of the story*

Steps

1. *Says this:*
    a. *Please click on the "start button". The system will bring you to empty electronic page. Each of you has a different electronic page. You will each start on a different electronic page.*
    b. *You may each type a story on the problem presented, up to 50 words long. Then you must click the submit button to send the page back to the group.*
    c. *The system will randomly bring you a different page. That page will have somebody else's story on it.*
    d. *When you see a page with somebody else's story on it, you may respond in following ways:*
        i. *You can enrich the story with complementary points*
        ii. *You can reflect or react on the story, give comments*
        iii. *You may argue against and add your own experience if you have counter example.*
    e. *You may type new episode of a story on new page. Then you must send that page back to the group. The system will bring you a new page.*
    f. *We will continue swapping pages and submitting comments, suggestions and elaborations about stories until we have a complete picture of the problem.*
    g. *Any questions?*

Storytelling Insight

In this technique a highly elaborative way of generation is used. People share experiences about a problem or issue. The results will be detailed, but fuzzy stories, which will offer contextual information. Extensive convergence is required to integrate stories and to elicit key aspects or characteristics of the problem. Story telling can also be used in a more visionary sense, to envision future scenario's and use cases of systems that are to be developed. A challenge is to ensure that the stories are sufficiently anonymous, and that naming and blaming is avoided. Another challenge is to ensure that stories are complete and open reports of experiences, and that vagueness such as 'and we all know what that meant' or 'the rest is history' is avoided.

Dialogue: generate

Choose this thinkLet if

- *Participants need to envision solutions from different perspectives*
- *A shared understanding of the needs and challenges of each perspective is required*

Overview

Participants will engage in a dialogue in which they will assume the role of a stakeholder in the issue or topic. In this way they will look at the challenge from a different perspective.

Input

A clearly defined topic or issue to generate reflections or contributions on

A set of stakeholder roles

Output

A dialogue in which issues and reflection are exchanged

How to use Dialogue

Setup

1. *Assign each participant a stakeholders role, ensure that they understand their role, and that they stay in their role for the duration of the activity*
2. *Assign each participant to a discussion page with the title of the role he/she represents*

Steps

1. *Ask each participant to write a brief statement from the perspective of his/her role about the issue*
2. *Ask participants to stay in their role and respond to one or more statements opening a dialogue. You can ask questions for clarification of the statement, reflect on it, or make a counter proposal.*
3. *After each contribution, return to your own page and respond, in your role to the remarks, or issues raised, keep your dialogue updated alternating between contributions to others and responding to issues raised on your own page.*

Dialogue Insights

A dialogue is a natural way to engage in discussion about issues and suggestions or reflections. In this way for instance a requirements negotiation can be done. As each participant is asked to discuss the requirements from a different perspective than his own, they will gain mutual understanding of the challenges and issues of each of the perspectives.

Some Definitions:

System Engineering (SE) *is methodological, cooperative and interdisciplinary approach covering overall suitable activities to design, develop, evolve and verify a system by giving satisfying solution to all stakeholders needs during the system life cycle (IEEE, 1999).*

Stakeholder *is an enterprise, an organization, or individual having an interest or a stake in the outcome of the engineering of a system (EIA-632, 1999).*
Client *is an enterprise, organization, or individual that: (1) commissions the engineering of a system; (2) is a prospective purchaser of the parts of a system, or portions thereof; or (3) is an acquirer of a system (EIA-632, 1999).*
End user *is an enterprise, organization, or individual that operates directly on the system (EIA-632, 1999). He/she can be the client.*
Acquirer *is an enterprise, organization, or individual that obtains a product (good or service) from a supplier (EIA-632, 1999).*