Available online at www.sciencedirect.com
SCIENCE DIRECT
Int. J. Human-Computer Studies 64 (2006) 611–621
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

A conceptual foundation of the thinkLet concept for Collaboration Engineering

Gwendolyn L. Kolfschoten[a,*], Robert O. Briggs[a,b], Gert-Jan de Vreede[a,c], Peter H.M. Jacobs[a], Jaco H. Appelman[a]

[a]*Faculty of Technology, Policy and Management, Delft University of Technology, P.O. Box 5015 – 2600 GA Delft, The Netherlands*
[b]*Center for Distance Education, College of Rural and Community Development, University of Alaska Fairbanks, US*
[c]*College of Information Science & Technology, University of Nebraska at Omaha, US*

Available online 20 March 2006

Abstract

Organizations increasingly use collaborative teams in order to create value for their stakeholders. This trend has given rise to a new research field: Collaboration Engineering. The goal of Collaboration Engineering is to design and deploy processes for high-value recurring collaborative tasks, and to design these processes such that practitioners can execute them successfully without the intervention of professional facilitators. One of the key concepts in Collaboration Engineering is the thinkLet—a codified facilitation technique that creates a predictable pattern of collaboration. Because thinkLets produce a predictable pattern of interactions among people working together toward a goal they can be used as snap-together building blocks for team process designs. This paper presents an analysis of the thinkLet concept and proposes a conceptual object model of a thinkLet that may inform further developments in Collaboration Engineering.

Keywords: Collaboration Engineering; ThinkLets; Collaboration; Object oriented modeling; Collaboration process design; Facilitation; Group Support Systems

1. Introduction

People frequently join forces to accomplish goals through collaboration that they could not achieve as individuals. By collaboration we mean joint effort toward a goal. Collaboration is essential for value creation (Hlupic and Qureshi, 2002, 2003), and often used for mission critical tasks. While team efforts can be productive and successful, group work is fraught with challenges that can lead to unproductive processes and failed efforts (Nunamaker et al., 1991). Many teams therefore rely on professional facilitators to design and conduct high-value or high-risk tasks (Niederman et al., 1996; Griffith et al., 1998).

The need for facilitation increases when teams seek to use Group Support Systems (GSS) technology. Under certain circumstances, GSS can lead to order-of-magnitude increases in team productivity (see (Fjermestad and Hiltz, 1999, 2001) for a comprehensive overview of GSS research). However, the success of a GSS session is by no means assured, see e.g. (de Vreede et al., 2003). As with many tools, GSS must be wielded with intelligence guided by experience in order for its potential to be realized. Novice users find the GSS tools easy to operate, but they typically cannot use the full potential of GSS. Most GSS users must therefore rely on professional facilitators in order to derive the benefits offered by GSS (Briggs et al., 2003; de Vreede and Briggs, 2005).

Skilled facilitators, however, tend to be expensive. They either have to be trained in-house, or hired as external consultants. Therefore, many teams who could benefit from facilitation interventions and from GSS must often manage without them. One solution to this challenge would be to reduce the need for skilled facilitation

*Corresponding author. Tel.: +31 (0)15 2783567; fax: +31 152 783429.
E-mail addresses: g.l.kolfschoten@tbm.tudelft.nl (G.L. Kolfschoten), bob.briggs@uaf.edu (R.O. Briggs), gdevreede@mail.unomaha.edu (G.J. de Vreede), p.h.m.jacobs@tbm.tudelft.nl (P.H.M. Jacobs), j.h.appelman@tbm.tudelft.nl (J.H. Appelman).

1071-5819/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2006.02.002

expertise; to find a way that a team could wield the GSS and manage its collaboration process for itself, without the ongoing intervention of a professional facilitator but with predictable results. Addressing this challenge is the domain of the emerging field of Collaboration Engineering.

Collaboration Engineering is an approach that designs, models and deploys repeatable collaboration processes for recurring high-value collaborative tasks that are executed by practitioners using facilitation techniques and technology. Collaboration processes designed in Collaboration Engineering are processes that support a group effort towards a specific goal, mostly within a specific timeframe. The process is build as a sequence of facilitation interventions that create patterns of collaboration; predictable group behavior with respect to a goal. The effort involves a continuous reciprocal interaction (Thompson, 1967), but does not require co-location of participants. Collaboration Engineering researchers seek to codify and package key facilitation interventions in forms that can be re-used readily and successfully by teams that do not have professional facilitators at their disposal. Therefore, there are three key roles within Collaboration Engineering:

A *facilitator* both designs and conducts a dynamic process that involves managing relationships, tasks and technology, as well as structuring tasks and contributing to the effective accomplishment of the meeting's outcome (Bostrom et al., 1993).

A *practitioner* is a task specialist who must execute some important collaborative task like risk assessment or requirements definition as a part of his or her professional duties. A practitioner is *not* necessarily a professional facilitator who designs new processes for new situations; a practitioner executes a specific collaboration process on a recurring basis (Briggs et al., 2003; de Vreede and Briggs, 2005). A practitioner therefore does not need extensive training as a facilitator, but only needs to learn the specific skills required to accomplish a particular collaboration process. The practitioner needs a high-quality, reusable, transferable process design that can deliver predictable results.

A *collaboration engineer* designs and documents collaboration processes that can be readily transferred to a practitioner. This means that a practitioner can execute the process without any further support from the collaboration engineer, nor from a professional facilitator.

Table 1 describes the collaboration engineering roles, their tasks in terms of collaboration process design and execution, and their required expertise. TextBox 1 provides an example of a collaboration engineer designing and transferring a risk management process in a large financial services firm.

To achieve the required quality and predictability described above, one of the current foci of Collaboration Engineering research is to identify and document reusable elementary building blocks for group process design.

Table 1
Collaboration Engineering roles

Role	Process design	Process execution	Expertise
Collaboration engineer	Repeatable, transferable processes	No execution, just process transfer	Both process and application domain
Facilitator	Ad hoc, context specific processes	Execution and ad hoc modification	Process
Practitioner	No design	Execution	Application domain

Textbox 1
Collaboration Engineering example

A large international financial services organization was faced with the challenge to perform hundreds of operational risk management (ORM) workshops. They requested a repeatable collaborative ORM process to be developed that operational risk managers could execute themselves. Based on the experiences and the requirements from the ORM domain experts, collaboration engineers developed a first prototype of a repeatable collaborative ORM process. This process was evaluated in a pilot project within a business unit, leading to a number of modifications to the definition of the overall process in terms of collaborative activities, their interdependencies, and the facilitation techniques used. The resulting collaborative ORM process was shown to a group of 12 ORM experts. During a half day discussion, the wording and order of activities was modified and the proposed collaborative activities where tested with a number of chosen facilitation techniques. In the period that followed, over 200 ORM practitioners were trained to execute this process. To date, these ORM practitioners have moderated hundreds of workshops where business participants identify, assess, and mitigate operational risks.

Toward that end, researchers have begun to codify a collection of such building blocks, called thinkLets, see e.g. Lowry et al. (2002), Enserink (2003), Harder and Higley (2004), Santanen and de Vreede (2004), Kolfschoten et al. (2004a). A thinkLet is a *named, packaged facilitation technique that creates a predictable, repeatable pattern of collaboration among people working towards a goal* (Briggs et al., 2001). ThinkLets can be used as conceptual building blocks in the design of collaboration processes (Kolfschoten et al., 2004a) and as learning modules of facilitation techniques for practitioners and novice facilitators (Kolfschoten and Veen, 2005; de Vreede and Briggs, 2005). A thinkLet is meant to be the smallest unit of intellectual capital required to be able to reproduce a pattern of collaboration among people working toward a goal.

A few examples of thinkLets are presented in Table 2 (see de Vreede and Briggs, 2001) for a more elaborate description of the mechanics of these thinkLets). Each thinkLet provides a concrete group-dynamics intervention, complete with instructions for implementation as part of some group process. Collaboration process designers who have a set of specific thinkLets available can therefore shift part of their attention from inventing and testing solutions to choosing known solutions (Kolfschoten and Veen, 2005). This may reduce both the effort and the risk of developing group processes.

To date, Collaboration Engineering researchers have formally documented approximately 70 thinkLets. Field experiences suggest that 16 of these thinkLets fill in perhaps 70% of a given collaboration process design; the other 30% of actions a group must perform need more specific thinkLets, either those described in the set of 70 or new thinkLets customized for the task at hand (Kolfschoten et al., 2004a). In this sense, thinkLets have become a powerful pattern language for collaboration engineers, who use thinkLet names to describe and communicate sophisticated, complex process designs in a compact form (Briggs et al., 2003; de Vreede and Briggs, 2005). Case studies describing such processes include an operational risk management process at an international financial services organization on (see TextBox 1 and (de Vreede and Briggs, 2005), a mission analysis process at the US Army's Advanced Research Lab (Harder et al., 2005), a knowledge elicitation process at the European Aeronautic Defense and Space company (EADS), and a crisis response process in the Rotterdam Harbor in the Netherlands (Appelman and Driel, 2005). ThinkLets have also been used successfully for training facilitators (Kolfschoten and Veen, 2005) and in collaboration research (Santanen, 2005). While only 70 thinkLets have been formally documented to date, it appears that the number of possible thinkLets may be infinite. The original conceptualization of thinkLets frames them as having three components: A tool, a configuration of that tool, and the facilitation script (Briggs et al., 2001). Each adaptation variation of any of these components on a known thinkLet can become a new thinkLet in its own right. This could lead to an exponential explosion in the number of thinkLets, giving rise to much redundancy and overlap among thinkLets, and to thinkLet "dialects" where collaboration process designers in different communities use different names for the same concepts. A large variety of thinkLets will enable a high chance of fit between thinkLets and tasks, but thinkLet dialects would make it difficult to transfer group process knowledge across the boundaries of local communities of practice. If dialects of thinkLets would be used jointly, it would increase the difficulty of the choice for a thinkLet. Therefore, there are several key goals for the Collaboration Engineering community:

- To minimize the explosion of thinkLets by identifying a stable core of conceptual thinkLets.
- To assist facilitators and collaboration engineers in choosing among the existing set of thinkLets.
- To design new thinkLets with the components of other thinkLets, being mindful not to replicate those that already exist.

Toward these ends, this paper presents several approaches to classifying thinkLets, and proposes a new conceptualization of the thinkLet as a first step towards enabling collaboration engineers to:

- More easily identify the optimal thinkLets for a collaboration process design.
- More easily distinguish the relevant differences among similar thinkLets.
- More easily identify areas of collaborative endeavor for which no useful thinkLets yet exist.
- Consolidate similar thinkLets into a uniform, non-redundant base set.

Table 2
Examples of thinkLets (de Vreede and Briggs, 2001)

Name	Purpose
LeafHopper	To have a group brainstorm ideas regarding a number of topics simultaneously.
Pin-the-tail-on-the-donkey	To have a group identify important concepts that warrant further deliberation.
RichRelations	To have a group uncover possible categories in which a number of existing concepts can be organized.
StrawPoll	To have a group evaluate a number of concepts with respect to a single criterion.
MoodRing	To continuously track the level of consensus within the group regarding a certain issue.

The remainder of this paper is structured as follows. In the next section we introduce thinkLets as originally defined and discuss their limitations. Next we propose a new conceptualization using the object-oriented modeling approach. We conclude this paper by discussing the implications and limitations of our research and proposing directions for future research.

2. Thinklets and collaboration process design

A collaboration process is a *series of activities performed by a team to accomplish a goal*. A fundamental assumption in the design of repeatable collaboration processes is that each process consists of a particular sequence of thinkLets that create various patterns of collaboration among the team members. Each activity in the design of a collaboration process can be supported by one or more *thinkLets*. ThinkLets can be combined but in order to go from one activity to the next, transitions are used. These concepts are discussed in more detail below.

2.1. Thinklets

As presented above, a thinkLet is a named, packaged, scripted collaboration activity that produces a predictable, repeatable pattern of collaboration among people working towards a goal. The initial conceptualization of thinkLet comprised three components: A tool, a configuration and a script (Briggs et al., 2001):

- The *tool* concerns the specific technology used to create the pattern of collaboration—anything from yellow stickies and pencils to highly sophisticated collaboration technologies such as GSS.
- The *configuration* defines how the tool is prepared (e.g. projected on a public screen), set up (e.g. configured to allow anonymous communication), and loaded with data (e.g. a set of questions to which people must respond).
- The *script* concerns everything a facilitator would have to do and say to a group to create the required pattern of collaboration (Briggs et al., 2003; de Vreede and Briggs, 2005).

Each differentiation in the components of a thinkLet influences the way in which people collaborate and is by definition a new thinkLet. It is important to be aware of such changes as research shows that small changes to, for instance, thinkLet scripts can create significant differences in group interactions, see e.g.Shepherd et al. (1996). However, experience shows that thinkLets are very customized to the situation at hand.

Knowledge of the three components of a thinkLet, it was argued, would be sufficient for a practitioner to recreate the pattern of collaboration (Briggs et al., 2001; Briggs et al., 2003; Santanen and de Vreede, 2004; de Vreede and Briggs, 2001, 2005). Field trials with more than 200 novice trained practitioners bore out the proposition that non-facilitators who knew the tool, configuration, and script for a thinkLet could, in fact, predictably and repeatable engender the pattern of collaboration a given thinkLet was meant to produce (de Vreede and Briggs, 2005). In Table 3, the thinkLets of Table 2 are described in the tool configuration script conceptualization.

2.2. Transitions

If thinkLets are building blocks for a collaboration process, then transitions are the mortar that connects them. The transition defines all the changes, events and actions that must take place to move people from the end of one

Table 3
Conceptualized thinkLet examples

Name	Tool	Configuration	Script (summary)
LeafHopper	GSS: group systems Categorizer	Several categories to which participants can add ideas	Explain categories Explain how to add Emphasize that participants work in the category of their choice
Pin-the-Tail-on-the-Donkey	GSS: group systems Categorizer	Participants can add annotation pin's	Allow a maximum amount of pin's Explain that participants should pin the items they want to discuss Discuss the pinned items
RichRelations	GSS: group systems Categorizer	Participants can read chauffeur can name categories and move concepts	Ask participants to name related concept. Document the name of the relation Categorize concepts with the relation
StrawPoll	GSS: group systems Vote	Cast a vote	Explain voting criterion and scale Allow participants to vote Discuss the results
MoodRing	GSS: group systems Opinion meter	Allow to adjust vote	Explain topic, voting criterion and scale Discuss topic while allowing participants to adjust their vote

thinkLet to the beginning of the next. A transition design must account for at least these aspects of change:

- Changes of *Technology*—when one thinkLet finishes, it may be necessary to reconfigure a technology or to move to a completely different technology before the next thinkLet can begin.
- Changes of *data*—it may be necessary to transform the output of one thinkLet in some way so that it can serve as the input to the next thinkLet.
- Changes of *orientation*—It is necessary to alert team members that one activity has finished and a new one is about to start. In this alert, the team should reflect its progress in reaching their goal.
- *Changes of location*—it may be necessary for people to move from one place to another between thinkLets.
- *Changes of membership*—sometimes it is necessary to change the composition of the team before the next thinkLet begins.

Although the importance of transitions seems obvious, it is hard to relate them to practice. Since Collaboration Engineering and thinkLets are often used in combination with GSS, part of the transition is automated. The role of transitions in the design of reusable, transferable and predictable collaboration process should yet be further analysed.

2.3. Compound thinkLets and modifiers

Sometimes a specific combination of several thinkLets are reused frequently in a variety of contexts. Such a sequence of thinkLets and transitions can be amalgamated into a named, reusable *compound thinkLet* (Kolfschoten et al., 2004a). It can be wielded as a single building block during process design.

Further, we noted that certain repeatable variations could be applied to a set of thinkLets to create a predictable change in the dynamics those thinkLets produce. We called these variations *modifiers*, and gave them names so they could be reused. For example, all creativity thinkLets allow people to contribute any idea that comes to mind. Using different tools and instructions, we can adjust the size and depth of the brainstorm. However, one could apply a OneUp modifier to any idea-generation thinkLet. The OneUp modifier changes the ground rules for brainstorming such that people may only contribute new ideas that are arguably better along some dimension than those already contributed. This modification can be added to any brainstorming technique.

Although facilitators, collaboration engineers and practitioners found the initial conceptualization of thinkLets, transitions and modifiers to be useful, field experience revealed a number of drawbacks (Kolfschoten et al., 2004b; Kolfschoten and Veen, 2005). First, the original concept tied a thinkLet closely to a specific technology in a specific configuration. Strictly speaking, a new thinkLet would have to be documented for any change of technology. Yet, collaboration engineers in the field frequently implemented the same thinkLet with a variety of different technologies. This suggested that the tool-and-configuration constructs might only be instances of a more-fundamental concept. This is also consistent with Briggs' guidelines for the development of collaboration theory, which argue the importance of concepts being independent of technology (Briggs, 2004).

Second, the original model of thinkLets also tied a thinkLet to a particular script. The purpose of the script is to prescribe exact behavior of the facilitator to support and instruct the group. Strictly speaking, this would mean that a new thinkLet would be documented to record any changes in the things a process leader did or said. Yet, both professional facilitators and practitioners in the field often deviated from the formal thinkLet script without significantly changing the pattern of collaboration (Kolfschoten et al., 2004a). Thus, it also seems that the existing thinkLet scripts might be instances of some more-fundamental concept.

Finally, under the original conceptualization, thinkLets were difficult to classify (Kolfschoten et al., 2004b). A reliable, detailed classification scheme for design components is an important tool for design support in any engineering discipline (Kolfschoten and Veen, 2005). The root of this difficulty may have been that concept addressed practical execution details of thinkLets rather than the essence of a thinkLet. The most commonly used classification scheme organizes thinkLets based on the patterns of collaboration they engender (de Vreede et al., 2005). This scheme proposes five general patterns of collaboration:

- *Diverge*: Move from having fewer to having more ideas.
- *Converge*: Move from having many ideas to a focus on and understanding of a few deemed worthy of further attention.
- *Organize*: Move from less to more understanding of the relationships among ideas.
- *Evaluate*: Move from less to more understanding of the value of ideas relative to one or more criteria.
- *Build Consensus*: Move from less to more agreement among stakeholders so that they can arrive at mutually acceptable commitments.

All thinkLets engender at least one of these patterns, so this scheme is somewhat useful for deciding which thinkLet might apply to a given situation. However, a number of thinkLets invoke multiple patterns simultaneously, so this scheme is not taxonomic. Further, it does not address issues of requisite pre-conditions, deliverables, available communication channels, and a host of other concepts that are important considerations when choosing one thinkLet over another.

In an effort to delve under the superficial properties of the original thinkLets concept, we undertook to create a base class diagram of group processes using the UML

modelling language. The next section of the paper articulates the new concept.

3. A new conceptualization of thinkLets

3.1. Object orientation

To develop the new conceptualization of thinkLets, we drew on the object-oriented modeling approach that has become the de-facto modeling paradigm for systems engineering. The basic theory of object-oriented modeling is to divide a system into classes and relations. A class is characterized by a set of attributes, operations and relations. Objects are specific instances of a class. All objects based on a given class share the same set of attributes, operations, relationships and semantics (Booch et al., 1999). For example all automobiles (a class) have wheels (an attribute). However, a particular Mercedes (an object which is an instance of the class, automobile) may have different kind of wheels than a given Volkswagen (a different object with a different value for the wheels attribute). Object-orientation distinguishes the following two types of relations:

- *Generalization versus specialization*: class A is a generalization of class B if and only if every instance of class B is also an instance of class A, and there are instances of class A which are not instances of class B. Equivalently, class A is a generalization of B if B is a specialization of A.
- *Association*: where generalization specifies a relation between classes, association refers to the structural relation between objects, or instances. Aggregation is a special form of association where a whole-part relationship between the aggregate, i.e. the whole, and the object, i.e. the part, is specified.

In the unified modeling language, UML, class diagrams such as the one presented in Fig. 1 present the object oriented view on a particular system. In this figure, both types of relations are illustrated. The arrow connecting the Manager and the Employee illustrates a specialization relation. A manager is thus a special case of employee, and as such its class inherits both name and contract from the Employee class. The association between a manager and a set of managed employees justifies the existence of the Manager class. This relation is equivalent to the relation between an employee and his name and contract.

3.2. The ThinkLet class diagram

Fig. 2 illustrates a class diagram for a collaboration process. The model incorporates the key concepts that must be taken into account when creating a design for a particular collaboration process. This section explains each of the components in that model, in more detail.

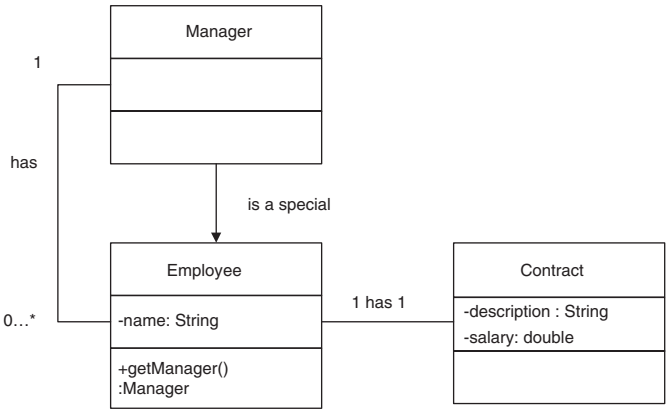

Fig. 1. Types of relations in OO.

3.2.1. Collaboration process

The central component of this model is the CollaborationProcess. Collaboration processes have a *name* attribute to identify them (e.g. Strategic Planning or Marketing Focus Group). Because all collaboration has some purpose, all collaboration processes also have a *goal* attribute. For a recurring collaboration process, the goal is typically instantiated as the deliverables that the team must create.

3.2.2. Participant

In a collaboration process, a group of 3 or more (Krackhardt, 1999) participants agree to work together towards the goal. Participants have a *name* attribute and they fulfill a certain role in the collaboration process.

3.2.3. ThinkLet

The collaboration process that people use to achieve their goal is composed of a series of thinkLets. A thinkLet is a named, packaged facilitation intervention that creates a predictable, repeatable pattern of collaboration among people working together toward a goal (Briggs et al., 2003). ThinkLets have a *name* attribute. The name is often intended to be catchy and somewhat amusing so as to be memorable, and it is usually intended to remind the collaboration engineer of the specific pattern the thinkLet invokes. For example, in the LeafHopper thinkLet, participants jump from topic-to-topic at will, making contributions in different categories as inspiration strikes. ThinkLets always supports the group in modifying a data set, whether captured (written down) or virtual (in a discussion). ThinkLets can be combined with other thinkLets (Kolfschoten et al., 2004a) yielding *compound* thinkLets that evoke more complex patters of collaboration, or a sequence in the pattern of collaboration.

3.2.4. Capability

The new approach to modeling thinkLets departs from the original convention of tool, configuration and script. Any tool, configured in a given fashion, affords certain capabilities. It may be possible to afford those same

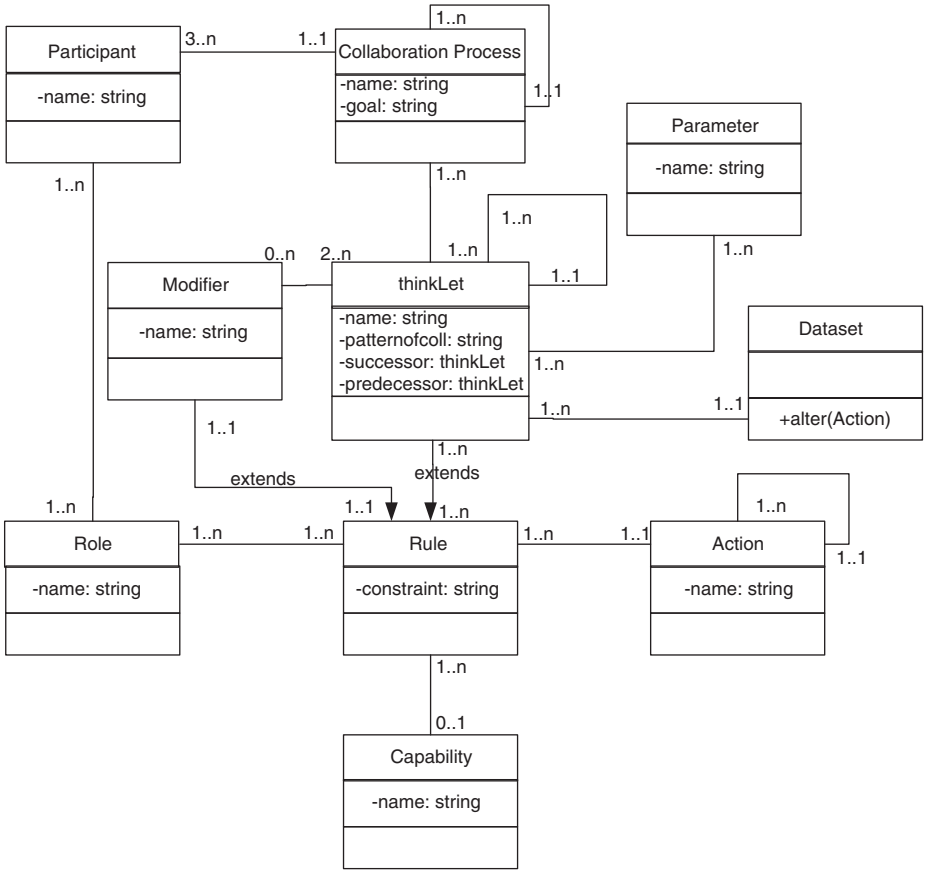

Fig. 2. A class diagram of collaboration processes.

capabilities with a variety of other tools. The new model therefore incorporates the concept of capabilities, while leaving the decision of how to realize those capabilities to the collaboration engineer. For example, the LeafHopper thinkLet requires the following capabilities: One page for each of several brainstorming topics. The possibility to display the topic of each page. The ability to contribute ideas to all pages. In an actual workshop, these capabilities could be afforded by flipchart pages taped to a wall and markers, or by a sophisticated GSS. When instantiated the tool and configuration of that tool that enable the capability should be defined.

3.2.5. *Action*

Once the capabilities have been provided to participants, they are instructed to execute certain actions—e.g. add, edit, move, delete, judge—using those capabilities.

3.2.6. *Rule*

Under the old conception of thinkLets, the script explained the actions people were to take, the constraints they were to apply to their actions, and the capabilities they were to use. However, these concepts could be conveyed by any number of scripts, i.e. a given script is merely a particular implementation of these concepts. Therefore, the new conceptualization captures the rules rather than the complete script of a thinkLet. Rules describe *actions* that participants must execute using the *capabilities* provided to them under some set of constraints. In all thinkLets, individual actions are subject to constraints. For example, the brainstorming question constrains the kind of concepts a person contributes to an ideation thinkLet.

To execute a given thinkLet, the participants must become aware of the rules that are to drive their efforts. If each participant executes their actions as guided by the rules, together they will produce the desired pattern of collaboration. For example, the rules of the FreeBrainstorm thinkLet require that added contributions must relate to the brainstorming question at hand and pages must be swapped after each contribution. A GSS can automatically enforce the page-swapping rule; a group working on paper must rely on social protocols and voluntary compliance.

Thus, under this framing, the thinkLet becomes technology- and script-independent. The designer of the process may choose any technology that provides the capabilities, and may choose any means to convey information and constraints to the team. This reduces the theoretical amount of possible thinkLets.

The small changes to the rules that guide actions can give rise to very different patterns of collaboration. For example, an ‘add’ action guided by a ‘summarize’ rule gives rise to abstraction, synthesis and generalization, while an ‘add’ action guided by an ‘elaborate’ rule gives rise to

increasingly detailed exposition of present concepts. Thus, the process designer must take care to choose rules purposefully and to express them carefully.

3.2.7. Parameter

In many thinkLets, there are certain pieces of information that must be conveyed to the team in order for them to work effectively. For example, in a brainstorming thinkLet, there is always a brainstorming question. In a polling thinkLet, there are always one or more voting criteria. The new model of thinkLets therefore incorporates the concept of parameters, which are variables whose content; a name and value(s) must be instantiated for each thinkLet.

3.2.8. Role

In some thinkLets, different actors must behave according to different rules (with different constraints actions and capabilities). The new model therefore incorporates the concept of *roles*. For example, in the ChauffeurSort thinkLet one person acts as the scribe while others discuss how concepts should be organized. Thus, the thinkLet requires two roles. In the PopcornSort thinkLet, however, all participants work in parallel, moving ideas into the categories where they best fit. This thinkLet has only one role.

3.2.9. Modifier

Finally, a modifier is a reusable rule that can be applied to a set of two or more thinkLets to change their dynamics in some predictable way. For example, the OneMinuteMadness modifier can be applied to any ideation activity. About a minute after the start of the brainstorm, the moderator stops the participants for a few moments to discuss whether their contributions are sufficiently responsive to the brainstorming question, and to clarify the rules and constraints of the thinkLet. Afterwards, brainstorming resumes.

4. An example

ThinkLets are instantiated on 2 levels. First, the instantiation is made as described above. This is illustrated for a number of thinkLets in Table 4, which presents the thinkLets from Table 2 in the new conceptualization for the general participant role. This way, thinkLets are described with a name, pattern of collaboration, successor and predecessor, parameter data set, rules with constraints, action and capability. Note that in Table 4 we describe independent thinkLets, which are not connected to a data set or connected with successors and predecessors in a collaboration process.

The thinkLets thus can be used for any collaboration process. In order to use for instance LeafHopper to do a SWOT analysis (strengths, weaknesses, opportunities, threats), we need to do a second instantiation, called implementation, in which we specify the parameters, rules and the capabilities, by providing the brainstorm question, (e.g. what are the factors that need to be considered in our

Table 4
Re-conceptualized thinkLet examples

Name & pattern of collaboration	Rule (constraint)	Capability (name)	Action (name)	Parameter (not instantiated)
LeafHopper Diverge	1. Add ideas to page in scope of the discussion topic (*X*) and scope (*Y*) 2. Add to any page at random as your interests dictate	A page for each *X*	Add	*X*: discussion topic *Y*: brainstorm question
Pin-the-Tail-on-the-Donkey Converge	1. Select the amount (*X*) of ideas (*Y*) that you consider key contributions 2. Read the indicated key contributions 3. Explain and discuss why a selected idea is a key contribution	X discriminators	Judge Read Discuss	*X*: amount *Y*: idea
RichRelations	1. Read the ideas (*X*) and identify related (*X*) 2. Defineand add the relation (Y)	A link for each *Y* connecting *X*1 and *X*2	Read, judge	*X*: idea
Organize	3. Connect the *X* to the *Y*	A page for each Y	Add relate	*Y*: relation
StrawPoll	1. Judge each idea (A) on criterion (*X*) ranging from scale min (*Y*) to scale max (*Z*)	A scaled discriminator for each idea	Judge	A: idea
Evaluate	2. Discuss the results of the combined voting	Processing of the combined results	Discuss	*X*: criterion *Y*: scale min *Z*: scale max
MoodRing Build Consensus	1. Indicate your opinion on issue (*X*) on criterion (*Y*) ranging from scale min (A) to scale max (B) 2. Discuss the issue 3. Indicate any change in opinion 4. Continue until there is sufficient consensus	A reusable scaled Discriminator for the Issue	Judge Discuss Judge	*X*: issue *Y*: criterion A: scale min B: scale max

company strategy?), the discussion topics, (e.g. strengths, weaknesses, opportunities, threats), the tool and configuration, (e.g. 4 whiteboards with the topics and a marker for each participant) and a script with precise instructions for the facilitator.

Note that each of the thinkLets requires a different combination of actions, under a different combination of rules. For instance, the StrawPoll allows participants to render judgments once about multiple concepts, while the MoodRing allows for continuous changes in judgment over time with respect to a single concept. Process designers may find that the actions-and-constraints model of thinkLets may provide a useful basis for selecting among available thinkLets at design time. This new framing may provide a more rigorous basis for classifying and choosing thinkLets.

5. Discussion and conclusions

A thinkLet is a named, packaged activity that produces a predictable, repeatable pattern of collaboration among people working toward a goal. The purpose of a thinkLet is to capture the smallest-possible unit of intellectual capital required to recreate a particular pattern of collaboration with specific results. ThinkLets serve as building blocks for designers of collaboration processes. This paper offers a re-conceptualization of the thinkLet concept in terms of elementary participant actions, physical capabilities, rules, roles and parameters, rather than tool, configuration and script. As a result, the new thinkLet conceptualization describes the requirements to create a certain pattern of collaboration independent from a technology and its configuration. This allows a collaboration engineer to choose appropriate thinkLets and subsequently select and adapt available technologies and facilities to instantiate the required capabilities. This reframing has clarified some ambiguities and eliminated some apparent challenges surrounding the old framing of the thinkLets concept. Although it is impossible to predict everything about a dynamic collaboration process, the new thinkLet concept is expected to offer more support to the collaboration engineer. In particular, it has the following advantages:

First, the thinkLet concept is now technology independent. As technologies change, a concept that is independent of these changes will be more consistent. Also, some of the descriptions of thinkLets already indicated that different tools could be used to reach the same result. This ambiguity has been removed.

Second, the cognitive load of the thinkLet concept is reduced. Since the ambiguity is resolved, it will be easier to transfer specific thinkLet objects to novice collaboration engineers and the instantiations of these thinkLets to practitioners.

Third, even with the new modeling convention, a very large, if not infinite set of thinkLet instantiations can be defined. However, the new thinkLets allow different instantiations and customizations, and the concept allows thinkLets to be modified adding additional rules. An analysis of a collection of thinkLets based on rules, roles and parameters, should make it possible to distill redundancy out of the collection. It may also be possible to abstract modifiers from a large collection of thinkLets, thus reducing combinatorial complexity.

Finally, the new conceptualization allows researchers to better compare differences and similarities in thinkLets and therefore also allows a better explanation for differences in apparently similar studies, and better design and operationalization of case and field studies on the effects of facilitation interventions and collaboration process design (Santanen, 2005).

There are a number of limitations with respect to the research presented in this paper. Each of these limitations gives rise to exciting avenues for future research. *First*, although the new conceptualization may give rise to a taxonomic classification scheme for thinkLets, no such scheme has yet been derived. To this end, it would be fruitful to compare other classifications in disciplines that are both close and distant from Collaboration Engineering, such as small group research (e.g. McGrath's (1984) task circumplex), GSS research (e.g. Bostrom et al., 1993; Bostrom and Anson, 1992)'s electronic meeting tasks and Zigurs and Buckland's (1998) task-technology fit model), workflow systems research (e.g. the Workflow Management Coalition's (2002) specification of workflow processes), telematics (e.g. BETADE's (Verbraeck and Dahanayake, 2002) telematics services building blocks), software development, (Briggs, 2004) and architecture (Alexander (1979)'s pattern language). A taxonomic thinkLets classification is critical to facilitate the choice of a thinkLet. Given the new, more elementary, conceptualization of thinkLets, we expect the creation of a taxonomic classification may be easier. In addition, we anticipate it be more straightforward to define a set of thinkLet choice criteria.

A taxonomic classification will also help to identify redundancy in the current thinkLet set. Some current thinkLets might be identified as variations on other thinkLets, resulting in a more limited yet more sharply defined final set of basic thinkLets that clearly differ from each other. If we can classify these we can also identify the area's where the current thinkLets do not provide solutions and new thinkLets or modifiers are required. This research is one step further on the path towards this goal.

Second, the current framing of the thinkLets still leaves open some questions. For example, Shepherd et al. (1996) demonstrated that slight variations in facilitator instructions that had no impact on rules, but rather touched on motivation, produced significant differences in group-productivity. The proposed new framing of thinkLets does not address that effect. In general thinkLets are focused on solving complexity of task or content rather than on complexity of group dynamics.

Third, new thinkLets classifications may be explored based on the new components of thinkLets. It appears that

especially the rules represent a promising starting point. For example, the rule 'generalize' could classify all thinkLets that use a given data set and produce the essence of that data set.

Fourth, the class model of Collaboration processes captures all the objects that must be considered when designing a collaboration process. However, it is not intended to convey the process itself, there is no sequence of steps in the model, just the components of the collaboration process are displayed. Transitions only contain process elements and are therefore excluded from this model. However, they could be part of a different modeling convention that conveys the flow and logic of a collaboration process as a team moves toward its goals. This model can be built as an extension on the facilitation process model described in de Vreede and Briggs (2005).

Finally, although the thinkLet concept seems to have evolved into a more useful one, we need to confirm that indeed it is an improvement both in practice and in theory by testing its use in collaboration process design and execution.

Acknowledgements

The authors gratefully acknowledge Daniel Mittleman and Alexander Verbraeck for their contributions to the new conceptualization of thinkLets. Furthermore we thank Johanna Bragge, Mariëlle den Hengst-Bruggeling, and the anonymous reviewers for their constructive feedback on this paper.

References

Alexander, C., 1979. The Timeless Way of Building. Oxford University Press, New York.

Appelman, J.H., van Driel, J., 2005. Crisis-response in the port of Rotterdam: can we do without a facilitator in distributed settings? In: Hawaii International Conference on System Science. IEEE Computer Society Press, Los Altos.

Booch, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Addison-Wesley, Indianapolis.

Bostrom, R.P., Anson, R., 1992. The face-to-face electronic meeting: a tutorial. In: Bostrom, R.P., Watson, R.T., Kinney, S.T. (Eds.), Computer Augmented Teamwork, A Guided Tour. Van Nostrand Reinhold, New York, pp. 16–33.

Bostrom, R., Anson, R., Clawson, V.K., 1993. Group facilitation and group support systems. In: Jessup, L.M., Valacich, J.S. (Eds.), Group Support Systems: New Perspectives. Macmillan, New York.

Briggs, R.O., 2004. On theory-driven design of collaboration technology and process. In: de Vreede, G.J., Guerrero, L.A., Raventos, G.M. (Eds.), CRIWG. Springer, San Carlos, Costa Rica, pp. 1–15.

Briggs, R.O., de Vreede, G.J., Nunamaker Jr., J.F., David, T.H., 2001. ThinkLets: achieving predictable, repeatable patterns of group interaction with group support systems. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Briggs, R.O., de Vreede, G.J., Nunamaker Jr., J.F., 2003. Collaboration engineering with thinklets to pursue sustained success with group support systems. Journal of Management Information Systems 19 (4), 31–63.

Enserink, B., 2003. Creating a scenariologic—design and application of a repeatable methodology. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Fjermestad, J., Hiltz, S.R., 1999. An assessment of group support systems experimental research: methodology and results. Journal Of Management Information Systems 15 (3), 7–149.

Fjermestad, J., Hiltz, S.R., 2001. A descriptive evaluation of group support systems case and field studies. Journal of Management Information Systems 17, 3.

Griffith, T.L., Fuller, M.A., Northcraft, G.B., 1998. Facilitator influence in group support systems. Information Systems Research 9 (1), 20–36.

Harder, R.J., Higley, H., 2004. Application of thinklets to team cognitive task analysis. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Harder, R.J., Keeter, J.M., Woodcock, B.W., Ferguson, J.W., Wills, F.W., 2005. Insights in implementing collaboration engineering. In: Hawaii International Conference on System Science. IEEE Computer Society Press, Los Altos.

Hlupic, V., Qureshi, S., 2002. What causes value to be created when it did not exist before? A research model for value creation. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Hlupic, V., Qureshi, S., 2003. A research model for collaborative value creation from intellectual capital. In: Twentififth International Conference of Information Technology Interfaces. Cavtat, Croatia.

Kolfschoten, G.L., Veen, W., 2005. Tool support for GSS session design. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

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

Kolfschoten, G.L., Briggs, R.O., Appelman, J.H., de Vreede, G.J., 2004b. ThinkLets as building blocks for collaboration processes: a further conceptualization. In: de Vreede, G.J., Guerrero, L.A., Raventos, G.M. (Eds.), CRIWG. Springer, San Carlos, Costa Rica, pp. 137–152.

Krackhardt, D., 1999. The ties that torture: simmelian tie analysis in organizations. Research in the Sociology of Organizations 16, 183–210.

Lowry, P.B., Albrecht, C.C., Nunamaker Jr., J.F., Lee, J.D., 2002. Evolutionary development and research on internet-based collaborative writing tools and processes to enhance e-writing in an e-government setting. Decision Support Systems 34, 229–252.

McGrath, J.E., 1984. Interaction and Performance. Prentice-Hall, Inc., Englewood Cliffs, NJ.

Niederman, F., Beise, C.M., Beranek, P.M., 1996. Issues and concerns about computer-supported meetings: the facilitator's perspective. Management Information Systems Quarterly 20 (1), 1–22.

Nunamaker Jr., J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R., George, J.F., 1991. Electronic meeting systems to support group work. Communications of The ACM 34 (7), 40–61.

Santanen, E.L., 2005. Resolving ideation paradoxes: seeing apples as oranges through the clarity of thinklets. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Santanen, E.L., de Vreede, G.J., 2004. Creative approaches to measuring creativity: comparing the effectiveness of four divergence thinklets. In: Hawaiian International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Shepherd, M.M., Briggs, R.O., Reinig, B.A., Yen, J., Nunamaker Jr., J.F., 1996. Social comparison to improve electronic brainstorming: beyond anonymity. Journal of Management Information Systems 12 (3), 155–170.

Thompson, J.D., 1967. Organizations in Action. McGraw-Hill, New York.

Verbraeck, A., Dahanayake, A., 2002. Building Blocks for Effective Telematics Application Development and Evaluation. Delft University of Technology, Delft.

de Vreede, G.J., Briggs, R.O., 2001. ThinkLets: five examples of creating patterns of group interaction. In: Ackermann, F., de Vreede, G.J. (Eds.), Group Decision & Negotiation. La Rochelle, France, pp. 199–208.

de Vreede, G.J., Briggs, R.O., 2005. Collaboration engineering: designing repeatable processes for high-value collaborative tasks. In: Hawaii International Conference on System Science. IEEE Computer Society Press, Los Altos.

de Vreede, G.J., Davison, R., Briggs, R.O., 2003. How a silver bullet may lose its shine—learning from failures with group support systems. Communications of the ACM 46 (8), 96–101.

de Vreede, G.J., Fruhling, A., Chakrapani, A., 2005. A repeatable collaboration process for usability testing. In: Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Altos.

Workflow Management Coalition, 2002. Workflow Management Coalition Workflow Process Definition Interface—XML Process Definition Language 2004, http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf

Zigurs, I., Buckland, B., 1998. A theory of task/technology fit and group support systems effectiveness. Management Information Systems Quarterly 22 (3), 313–334.