A Design Approach for a Universal Group Support System using ThinkLets and ThinXels

Stefan W. Knoll[a], Martin Hörning[b], Graham Horton[a]

[a]Department of Simulation and Graphics, Faculty of Computer Science, University of Magdeburg, Germany,
[b]Igosys IT-Service GmbH, Magdeburg, Germany

Abstract

Group Support Systems (GSS) can increase the productivity of Decision Making by offering a variety of tools to assist a virtual group across geographical distances. But experience shows that the value of a GSS depends on how purposefully and skilfully it is used. We present a design approach for a universal GSS which supports the participants in designing, executing and exchanging collaboration techniques. Our approach is based on a modelling language which uses the well-known concept of a thinkLet and the new concept of a thinXel. A ThinXel is defined as an atomic instruction leading to one activity of the participant which has a well-defined function in the context of the group's goal. ThinXels can be sequenced according to formal rules to form patterns of collaboration, called thinkLets. The name "thinXel" (thinking element) is formed analogously to the well-known "pixel" (picture element) in Computer Graphics. We claim that a thinkLet-based modelling language using the concept of thinXels can provide the following advantages: to describe a group process in a simple way, to generate an algorithm that guides the participants through the process and to increase the usability of GSS for practitioners without expertise. We will present a graphical representation for modelling a group process and use the discussion technique “Six Hats” as an example to illustrate the elements of our modelling language.

1. Introduction

In a world of rapid change, affected by globalization, geographical development and an ever-growing number of new technologies, companies are forced to adapt to new market situations more and more rapidly. Fast decisions for new market strategy play a crucial role in maintaining a competitive position in local and global markets. But the success of decision making and negotiation in a group is limited by the abilities of the participants: Each individual participant has different experience and knowledge to make a decision. By using collaboration techniques participants can combine their potential and expertise and can accomplish a decision that will be accepted by all participants.

Six Hats [de Bono 2000] is a classic example for these kinds of collaboration techniques. By using categories for contributions as Problems, Opportunities and Feelings the participants can be supported in realizing and analyzing the problem as well as finding a solution for it. Group Support Systems (GSS) can increase the productivity of this group work by offering a variety of tools to assist the group in the structuring of activities and improving group communication [Grohowski 1990], [Nunamaker 1991], [de Vreede 2003]. A GSS represents an information technology-based group meeting environment which can be used to manage group work with virtual teams across geographical distances. But experience shows that the value of the technology depends on how purposefully and skilfully it is used [Briggs 1999].

To improve the efficiency and effectiveness of GSS for group work, a number of factors have to be taken into account during the design and execution of a collaboration process. Most GSSs consist of different tools, each of which can be configured in different ways to implement collaboration techniques. Today several collections of problem-solving and decision-making techniques exist, which provide classifications, rules and instructions on how to apply these techniques [VanGundy 1981]. But these rules and instructions are for the most part very abstract and require a lot of experience to adapt them to the provided tools of the GSS. This adaption can influence the effectiveness and efficiency of the group work and makes it difficult for practitioners without expertise to use the GSS. In this case, practitioners use techniques that are already provided by the GSS and do not benefit from new designed collaboration techniques. Another factor is formed by the facilitation itself. A facilitator’s instructions can have a significant influence on psychological collaborative effects (e.g. production blocking, frustration, distraction or flow). These can be observed in a face-to-face workshop and may also appear in group work using GSS [Diehl 1991]. They originate in the process of collaboration and can have a negative as well as a positive influence on the efficiency of a system. Facilitation expertise is needed to take these effects into account. For this reason companies often use professional facilitators who have expertise in the design and execution of group work and can improve group productivity. However skilled facilitators can be expensive and as a result most groups cannot benefit from facilitation intervention [Briggs 2003]. Therefore the potential of a GSS for group work cannot be fully exploited.

The challenge is to find a way to develop a universal GSS that:

- enables the participants to design and execute different kind of group processes;
- adapts the system automatically to a designed group process;
- increases the usability of the GSS for practitioners without expertise.

This would do much to support group work by using GSS. Our research on the creative process [Horton 2006] and the experience with commercial creativity workshops through the cooperation with the Zephram AG shows that an object-oriented group process can support the facilitator by designing and executing a creative process. We claim that a group process can be subdivided into formal concepts which can be sequenced according to formal rules to form a collaboration process. This combination of formal concepts and rules creates a construction plan for a collaboration process. A universal GSS would be a system that uses such a kind of construction plan to implement a collaboration process. We claim that this construction plan can be represented by a modelling language for group processes.

In this paper we will introduce a new modelling language approach for group processes that establishes the basis for a universal GSS. The basic principle of our approach is to divide the group process into pattern elements and to identify activities of the participants which can be sequenced to a sentence in a modelling language.

2. ThinkLet: A formal pattern of collaboration

Collaboration Engineering (CE) is an approach to form a pattern language for group collaboration [Briggs 2003]. The purpose of this approach is to create collaboration processes for recurring high value tasks in companies that can be executed by participants without ongoing support by professional facilitators. These processes can be classified into six key patterns of collaboration (for further information see Briggs [Briggs 2006]). Researchers use this classification to collect, create, document and test collaboration activities, called thinkLets, that together form a pattern language for group collaboration [Briggs 2003]. Briggs et al. describe a thinkLet as a named, scripted and reusable collaborative activity for creating a known pattern of collaboration among people working together toward a goal [Briggs 2006].

The specification of a thinkLet consists of different components [de Vreede 2006]:

- **The identification** - contains a name attribute, which is intended to emphasize the specific group dynamics the thinkLet invokes;
- **The script** - contains rules for a participant in a defined role for creating the required pattern of collaboration. These rules describe the actions a participant has to execute using the capabilities under some set of constraints;
- **The selection guide** - contains different attributes such as patterns of collaboration to support the participant in the selection of a thinkLet.

The concept of a thinkLet was invented as a tool specific concept representing a combination of a software tool, its configuration and a script for its use [Briggs 2001]. Due to this specialization, a thinkLet always needs to be adjusted before it can be implemented in another software tool. A new object-oriented conceptualization for a more tool-independent thinkLet was presented by Kolfschoten [Kolfschoten 2006]. It is based on the object-oriented modelling approach for systems engineering [Gamma 1995]. Kolfschoten divided the existing thinkLet concept into key concepts and relations that can be illustrated in the new defined thinkLet class diagram for a collaboration process based on the unified modelling language (UML) notation (for further information see Kolfschoten [Kolfschoten 2006]).

Some thinkLets are illustrated in Table 1 by their scripts [Kolfschoten 2006]. Research has shown that practitioners who know the specification of a thinkLet can predictably and repeatable engender the pattern of collaboration a given thinkLet is intended for, even without any facilitation expertise [de Vreede 2005].

For that reason CE represents an interesting and suitable approach for designing a modelling language for group processes. ThinkLets can be seen as reusable elements as they are based on the object-oriented paradigm and organized in patterns of collaboration.

One disadvantage of thinkLets is that abstract rules are used to achieve the collaboration pattern. For example, the thinkLet “Mood Ring” (illustrated in Table 1) contains the rule “Discuss the issue”, which should create the activity “discuss”. But, in our opinion this rule can lead to a combination of different activities: explain the issue, collect comments and collect proposals for solution. In this case the script still leaves open the question which facilitation instruction should be used to achieve these actions and in which order these activities should be executed. Therefore, it is not possible to create an algorithm that describes the steps all participants have to go through during the group process. Nevertheless, this algorithm represents a basic property of a modelling language to create a construction plan for group processes.

In order to compensate for this weakness we suggest analyzing the script of a thinkLet for foundational facilitator instructions and their resulting activities of the participants. We claim that each thinkLet script can be described by a sequence of atomic facilitator instructions which lead to one activity of the participants per instruction. With the limitation to one activity per instruction, an atomic facilitator instruction represents a reusable element and enables the design of a modelling language for group processes. A universal GSS can use this modelling language to generate an algorithm, to adapt the system automatically to a designed group process or guide the participant automatic through a group process. This would support practitioners without facilitator experience in executing a group process in a more efficient way.

Table 1: thinkLet example [Kolfschoten 2006]

Name & pattern of collaboration	Rule (constraint)	Capability (name)	Action (name)	Parameter (not instantiated)
Leaf Hopper 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: discuss topic Y: brainstorming question
Mood Ring Build consensus	1. Indicate your opinion on issue (X) on criterion (Y) ranging from scale min (A) to scale max (B)	A reusable scaled discriminator for the Issue	Judge	X: issue Y: criterion A: scale min B: scale max
	2. Discuss the issue		Discuss	
	3. Indicate any change in opinion		Judge	
	4. Continue until there is sufficient consensus			

3. ThinXel: A formal instruction element

Our design approach for a reusable instruction element is called a thinXel. We define a thinXel as an atomic facilitator instruction leading to a response which has a well-defined function in the context of the group's goal [Knoll 2007]. The name "thinXel" (thinking element) is formed analogously to the well-known "pixel" (picture element) in Computer Graphics.

ThinXels were invented for supporting the communication between facilitator and participant in a group process. The concept is based on the Shannon-Weaver Model of a communication process [Shannon 1948] which describes the transmission of a message between a sender and a receiver. In this model, a facilitator instruction (message) represents a coded intention of the facilitator (sender) for a participant (receiver) of a group process. The participant needs to decode the instruction to understand the intention and be able to respond to it. It can happen that noise degrades the quality of the transmission. As a result the interpretation of the instruction might not match the intention of the sender. A thinXel is a reusable atomic facilitator instruction which allows practitioners without facilitation expertise to execute a group process by achieving an intended effect.

The design of a thinXel is influenced by the research on the Cognitive Network Model. Research has shown that the working memory of a participant is limited. People can only pay attention to about three concepts at the same time [Broadbent 1975]. The number of concepts can increase by rehearsal, grouping and other mental strategies. But without a refresh by conscious rehearsal or by external stimuli, the content of the working memory would fade within seconds [Brown 1958]. By taking these factors into account, we claim that a thinXel should contain only one intention of the facilitator that leads to one action in the context of the group's goal. To limit the cognitive load of the participant, the instruction may contain only those pieces of information that must be conveyed to the participant to perform the action intended by the facilitator. This makes a thinXel to an atomic instruction.

These properties define the difference between the concept of a thinkLet and a thinXel. A thinkLet script describes a sequence of rules which lead to a collaboration pattern. This sequence can contain facilitator instructions that lead to a combination of different activities of the participants. In contrast, a thinXel represent one atomic instruction which leads to only one basic activity of the participants like reading, creating, selecting and putting. We claim that the set of these activities are finite and that a sequence of thinXels can be used to describe a thinkLet script and to reach a pattern of collaboration.

The intended activity of a thinXel can be categorized into context and data-oriented thinXels:

- **Context-oriented thinXel** - Representation of a facilitation instruction with the intention to create a working environment for the group process. These instructions explain goals, rules and the working process and lead to acceptance among the participant.
- **Data-oriented thinXel** – Representation of a facilitation instruction with the intention to change the existing dataset of the group process. These instructions can be divided into three kinds of activities: *create* (to create a new contribution), *grow* (to enhance an existing contribution with new attributes) and *organize* (to create

a relation between two contributions). Each other data activity can be represented by a combination of these three activities. An example is the activity "delete" that can be represent by the activity "organize" which leads to a relation between a contribution and a contribution named "wastebasket".

An example of a thinXel-based facilitation is illustrated in Table 2, which specifies the facilitation instruction "Discuss the issue" shown in Table 1. This way, the instruction is divided into a sequence of atomic facilitator instructions, each producing only one activity. The participant knows in every single part of the process what he or she should do. This makes a process more controllable and allows a participant without facilitator experience to execute an unknown group process in an efficient and effective way.

The described properties, to produce limited cognitive load and to lead to one activity in the context of the group's goal, define a thinXel as a reusable and formal instruction element. We claim that these elements can be sequenced according to formal rules to form more complex modules like thinkLets. This new representation of the thinkLet script would reduce the degree of freedom by their interpretation to a minimum. This allows us to design of a modelling language for group processes that can be used to develop a universal GSS.

Table 2: thinXel example for the facilitation instruction "Discuss the issue"

thinXel	Action (name)	ThinXel (type)	Intention
1. Please use the next steps to collect comments for the existing issue. These comments will be used to build consensus.	Know	context	know the goal
2. Please take a pen and some sheets of paper to collect your comments.	Take	context	material for work
3. Please listen to the existing issue.	Know	context	know the existing issue
4. Please write down on a sheet of paper the word "comments".	Create	data	define a work cluster
5. Please create the cluster "comments" by putting the paper in the middle of the desk.	Put	data	create a work cluster
6. Please write down on a sheet of paper a comment, which you associate with the issue.	Create	data	associate a contribution
7. Please put the comment into the cluster "comment".	Put	data	cluster the contribution
Repeat steps 6 - 7 until time > X or number of comments > Y			
8. Please use the next steps to read the existing comments to know what other participants think about the issue.	Know	context	know the goal
9. Please select a comment of other participants from the cluster "comments".	Select	context	select a contribution
10. Please read the comment.	Know	context	know the contribution
Repeat steps 09- 10 until time > X or number of comments not read = 0			
11. Please use the next steps to create proposals for the existing issue. These proposals will be used to build consensus.	Know	context	create a work cluster
12. Please take a pen and some sheets of paper to collect your proposal for solution.	Take	context	material for work
13. Please write down on a sheet of paper the word "proposals for solution".	Create	data	define a work cluster
14. Please create the cluster "proposals for solution" by putting the paper in the middle of the desk.	Put	data	create a work cluster
15. Please write down on a sheet of paper a proposals for solution, which you associate with the known comments.	Create	data	associate a contribution
16. Please put the proposals for solution into the cluster "proposals for solution".	Put	data	cluster the contribution
Repeat steps 15 - 16 until time > X or number of contributions > Y			

4. A modelling language approach

In this section we introduce a modelling language approach for group processes. We use graphical elements for designing a language independent approach that also increases the usability of the model. The design of these elements follows graphical representations of existing elements from process models like Event Driven Process Chains (EPCs) [Keller 1992], Petri Nets [Murata 1989] and UML activity diagrams [Ambler 2005]. In the following we will describe each element in detail.

The elements "participant flow" and "group flow": Our modelling language approach uses the concepts of thinkLet and thinXel for describing a reusable group process. These concepts guide the participants through the process and lead to different activities whose interaction creates a collaboration effect. For this reason, our process flow describes the way of the participant through the group process and is represented by a simple arrow symbol shown in Figure 1. This graphical element can be used to illustrate concurrent processes but does not distinguish between an individual participant and a group of participants moving synchronously through the process. To show a clear distinction between these processes the model uses two different kinds of lines for the graphical elements. A single line represents an element for an individual participant and a double line stands for a group of participants.

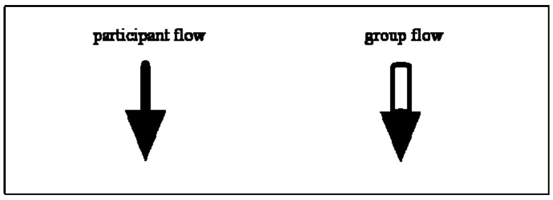

Figure 1: elements "participant flow" and "group flow".

The element "decision": For directing the process flow we define a decision element that allows us to react to internal and external stimuli. The graphical representation for this element is similar to the decision element of the UML activity diagram. Figure 2 illustrates a decision for the participant and group flow. The element can be used for example to check if the participant has selected a special category and can go on in the process or if he or she must be guided back to the selection process.

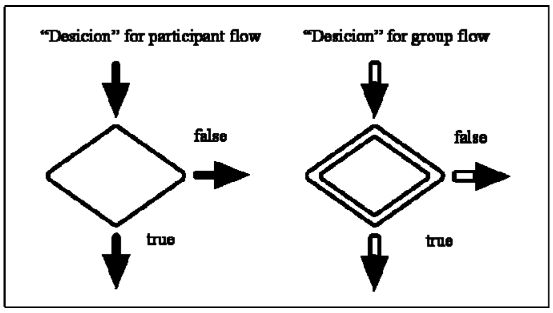

Figure 2: element "decision".

The element "transition": For the presentation of concurrent processes including parallelization and synchronization, our modelling language uses the element "transition". It represents the places in the model where individual participants can be added to a group or rather a group can resolve into individual participants. This property is new for the concept of transitions and allows us to describe group processes with concurrent activities of the participants in a simple form.

The graphical representation for this element follows the transition element in an UML activity diagram shown in Figure 3. An example for the use of a transition is a group process that use different roles for the participants. By using a transition the group will be divided into two groups as soon as all participants have arrived at the transition.

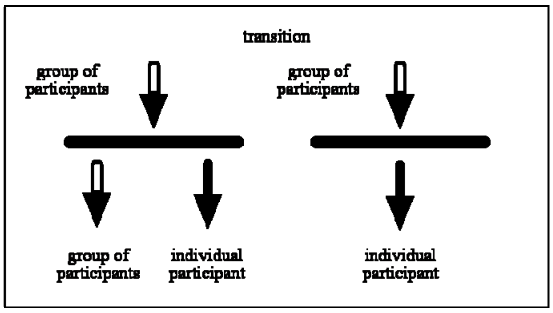

Figure 3: element "transition".

The elements "sender", "receiver", "response" and "signal path": The participants need to interact in order to create a collaboration effect. This interaction can be realized by a common data set where each participant can see changes of other participants and react to it. Furthermore, internal and external signal can influence the activities of the participants. For describing these kinds of interaction the model uses the elements "sender", "receiver", "response" and "signal path".

The graphical representation for the "sender" and the "receiver" are similar to those in the UML activity diagram and are connected by the "signal path". There is no limitation either for the numbers of receivers a sender can be connected to or for the number of senders a receiver can be connected to. Figure 4 illustrates the connection between sender and receiver.

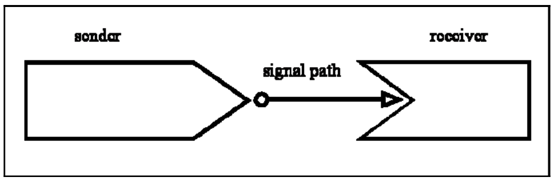

Figure 4: elements "sender", "receiver" and "signal path".

The element "response" represents the influence of a signal on the activities of the participants. It interrupts the participants in their current activities and changes their flow. The graphical symbol for this element is a half-circle representing a collecting point for participants. An example for this situation is shown in Figure 5. With the arrival of a participant the "sender" will send a signal via the "signal path" to the "receiver" which will interrupt the current activities of the participants (represented by dashed arrows). These participants will then be guided back to the process flow by the receiver.

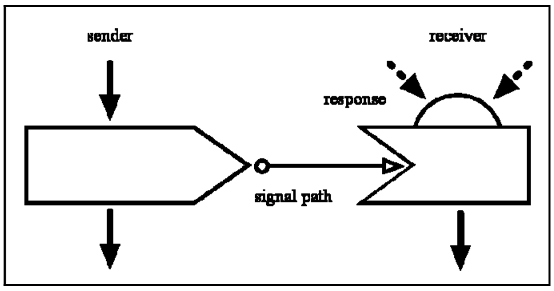

Figure 5: elements "sender", "receiver", "response" and "signal path".

The elements “storage place” and "data path": During the group process participants can produce various types of data such as contributions like ideas, comments or ratings. Therefore, the modelling language must represent a data element for different types of data. Our approach for its representation follows the handling of data in a face-to-face workshop. We define the element “storage place” that is able to store particular types of data. This “storage place” is connected by a "data path" arrow to elements requiring data and allows us to describe the data flow of the group process shown in Figure 6.

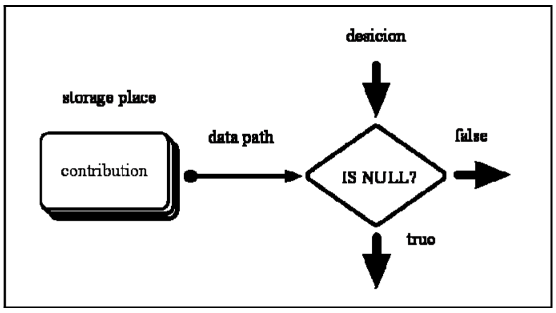

Figure 6: elements “storage place” and “data path”.

The element “thinXel”: The concept thinXel represents an instruction that leads to a single activity of the participant. For adapting this concept to the flow of participants of this modelling language, we define the element “thinXel” as a binary object with one input and two outputs. ("thinXel executed" and "thinXel aborted"). This element only represents the goal of a thinXel in a formal instruction but leaves the precise wording of the instruction to the facilitator or GSS. Depending on the group process, the element “thinXel” can be connected to a “storage place” by a “data path”. For example, the “thinXel” of the type “create” is connected with a “storage place” for storing new contributions and making it available to each participant shown in Figure 7.

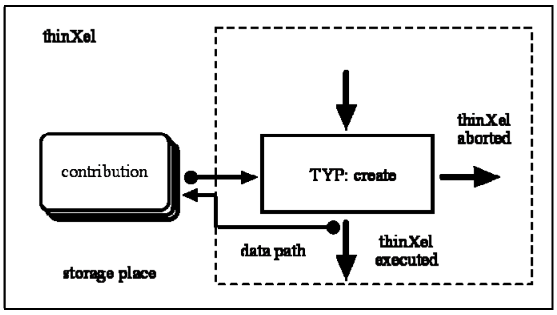

Figure 7: element “thinXel” in connection with a “storage place”.

The element “thinkLet”: The presented elements of the modelling language can be used to describe the flow of a group process by the participants, their data and signals. But the representation of each participant activity increases the complexity of the process model and decreases its usability. We use the concept of a thinkLet for working against this situation and defining a “thinkLet” as a reusable element describing an existing process model that can be integrated into another group process. The process model is represented by a “thinkLet construction plan” that includes special interface elements to connect the thinkLet with the process flow of another group process. Figure 8 shows these interface elements and their arrangement in an exemplary “thinkLet construction plan” and how the corresponding “thinkLet” looks like. The interface elements "start" and "end" represent the connection of the participant flow between the thinkLet and the group process. Data and signals can be received with the elements “parameter” and “signal parameter” and can be forwarded by "data forwarding" and "signal forwarding”.

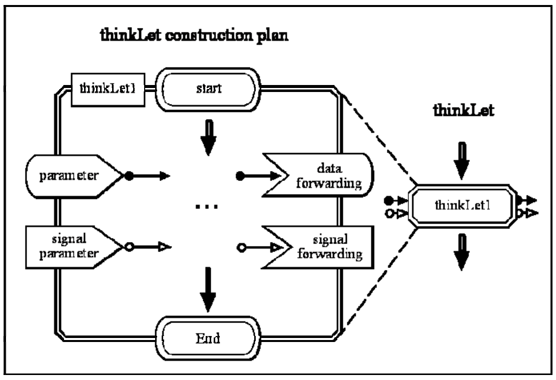

Figure 8: elements “thinkLet instruction plan” and “thinkLet”.

The element “signal-data-generator”: The element "signal-data-generator" describes an abstract element, which can be embedded into the group process to produce data or signals. An example for this is a timer which sends a signal after a certain period of time or a random number generator, which offers random numbers. The graphical representation shown in Figure 9 depends on the type of generator and should be defined according to their function.

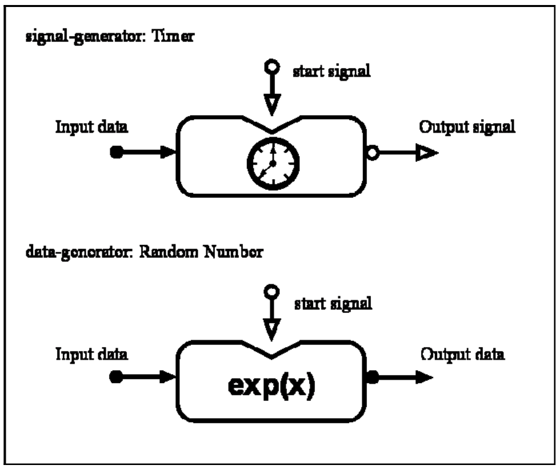

Figure 9: element “signal-data-generator”.

The presented elements form a modelling language for group processes. They describe the flow of participant, data and signals of a group process and guide these flow by the elements “transition” and “decision”. Each activity of a participant can be represented by the element “thinXel”. An object-oriented approach is given with the element “thinkLet” that represent a described process model which can be connected with other group processes. We claim that this modelling language can be used to represent all kind of group processes. This is possible because we believe that each group process can be divided into a sequence of single activities that can be represented by the elements of “thinXel”. This new representation creates a construction plan for a group process which can be used to develop a universal GSS.

5. Six Hats: A group process model

We will now present an example for a group process model which uses our modelling language approach. This example describes a discussion process based on the technique Six Hats [de Bono 2000] that uses different categories for contributions to support the participants by their discussion of different ideas. Our process model for "SixHats" is described by a group thinkLet element named "SixHats" (shown in Figure 10) that includes a thinkLet element for an individual participant, named "OneHat" (shown in Figure 11). Both elements will be described by their thinkLet construction plans in detail.

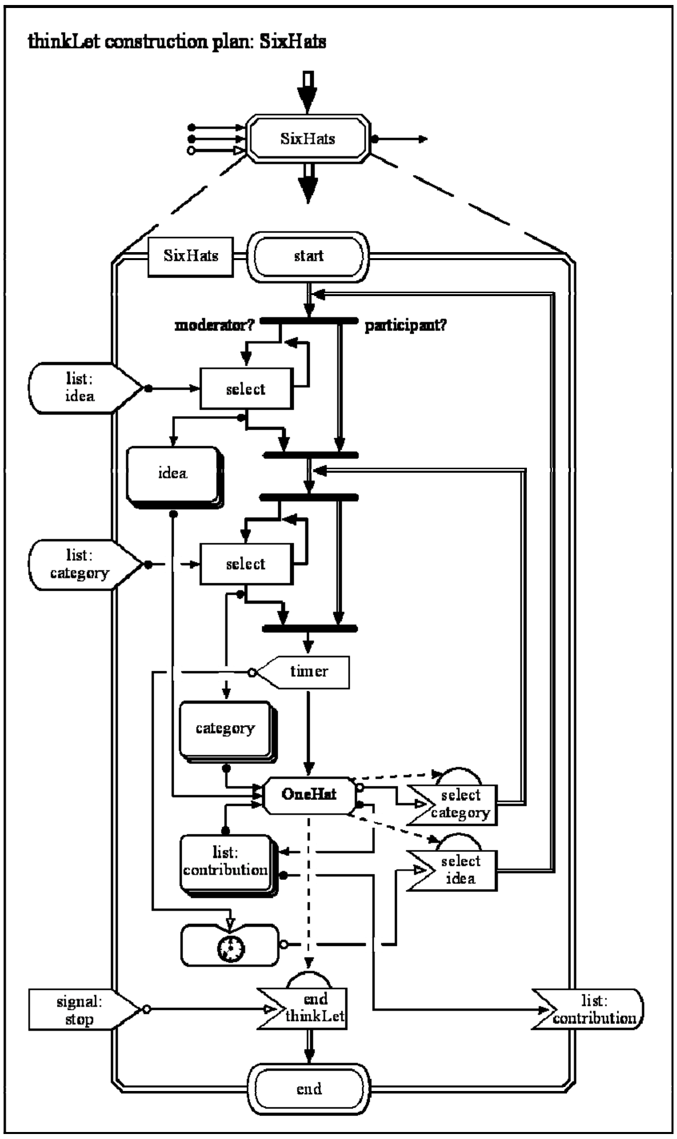

Figure 10: thinkLet construction plan "SixHats"

The thinkLet construction plan of the thinkLet element "SixHats" represents a described group process that can be connected with other group processes by the following input and output elements:

- **Input element / data-parameter "list: idea"** - represent a data list of the type "idea" that will be discussed by the participants;
- **Input element / data-parameter "list: category"** - represent a data list of the type "category" which can be used to structure the discussion;
- **Input element / signal-parameter "signal: stop"** - represent a signal with the function to stop the group process and finish the thinkLet;
- **Output element / data forward "list: contribution"** - is used to provide the created and organized contributions to other group processes;

The process flow of the thinkLet element "SixHats" starts with a transition element that divides the user group into a moderator and a group of participants. The moderator will be guided to the context-oriented thinXel element "select" that represents the intention "to select something from somewhere under some constraints" This thinXel element instructs the moderator to select one data element from the data list "idea" that should be discuss by the participants. The moderator has access to the data list by the data path between the thinXel element and the data parameter element. The group process will send the selected idea to the storage place "idea" that provides the idea for the thinkLet element "OneHat". The Moderator will be guided to the next transition element. This transition element sends the participants and the moderator together to the next stage of the group process. In this stage the moderator will select a category for the contributions of the discussion. This data element of type "category" will also be saved in a storage place to provide it as a parameter to the thinkLet element "OneHat". The process flow uses a transition element to collect all user of the process and guide them individual to the thinkLet element "OneHat". This process activates the sender element "timer" that sends a start signal to the signal-generator "timer" (illustrated by a clock symbol), which is used to limit the time to process each idea. Each participant will use an instance of these thinkLet element and will interact with each other over the storage element "list: contribution" that will save the contributions of all participants.

The thinkLet element "OneHat" is connected to the element "SixHats" by the following input and output elements:

- **Input element / data-parameter "idea"** - provides a data element of the type "idea" which was selected by the moderator;
- **Input element / data-parameter "category"** - provides a data element of the type "category" which was selected by the moderator;
- **Input element / data-parameter "list: contribution"** - provides a list of all created contributions of the participants for a selected idea. The parameter is connected with the public storage element "list: contribution" of the thinkLet element "SixHats";
- **Output element / data forward "contribution"** - send a new contribution of a participant to the storage element "list: contribution" of the thinkLet element "SixHats";

- **Output element / signal forward "change category"** - send a signal that leads to a change of the contribution category;

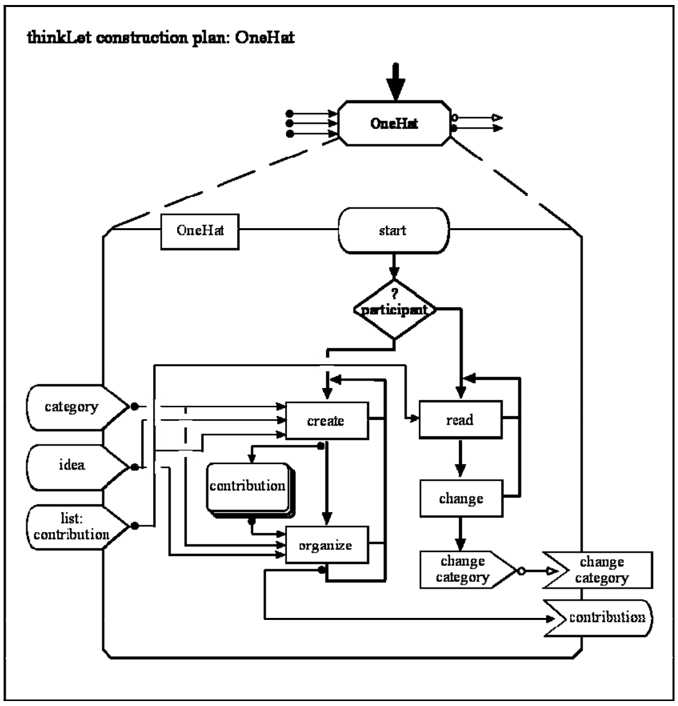

Figure 11: thinkLet construction plan "OneHat"

The thinkLet element "OneHat" starts with a decision element that checks the role of the user. If the user represents the role "participant" the process flow will guide him to the data-oriented thinXel element "create" that represents the intention "to create something under some constraints". This thinXel element instructs the participant to create a new contribution for the active idea under the selected category. It is connected with the data-parameter "list: contribution" to provide existing contributions of all participants as stimuli. A created contribution will be related to the active idea and category by the data-oriented thinXel element "organize". This thinXel element invites the participant to check if the contribution represents an element of the category. If the participant agrees, the related contribution will send to the data forward element "contribution" that is connected with the public storage element "list: contribution". The participant will be guided back to the thinXel element "create" to create another contribution. The previous created contribution will be provided as a stimulus for new contributions. In the case that the participant will abort the thinXel element "organize", the created contribution will not be stored in the storage element.

Another participant flow is described for the role "moderator". A moderator will be guided to the context-oriented thinXel element "read" that represents the intention "to read something to enhance knowledge". This thinXel element instructs the moderator to read the created contributions of the participants to get an overview of the process. If the moderator sees the necessity to change the category, he can use the context-oriented thinXel element "change" that represents the intention "to change the context". This thinXel element will activate the sender element "change category" that will send a signal over the signal forward element "change category" to the receiver element "select category" of the thinXel element "SixHats". The receiver element will interrupt all users in their current activities and collect them in the response element. The process flow will guide all users as a user group to the third transition element of the thinkLet element "SixHats". After the selection of a new category by the moderator the user group will use again the thinkLet element "OneHat".

The activated signal-generator "timer" will send a signal after a certain period of time to the receiver element "select idea" which will also interrupt all users in their current activities but will guide them back to the first transition of the thinkLet. The moderator can now select a new idea and a new category for the process flow.

The group process of the thinkLet element "SixHats" will stop by a signal from the sender element "signal: stop". The receiver element "end thinkLet" will collect all users and guide the participant flow to the exit of the thinkLet element.

The described group process model represents a construction plan that can be used to describe different kind of group processes by changing the parameter elements. For example the parameter element "list: category" can provide different lists of categories to analyze the data by different approaches. We claim that this applicability of the model can be used to develop a universal GSS that uses a process model to provide different kind of group processes.

6. Discussion and Future Research

We proposed that a universal GSS would do much to support group work by using GSS. We define a universal GSS as software tool that:

- enables the participants to design and execute different kind of group processes;
- adapts the system automatically to a designed group process;
- increases the usability of the GSS for practitioner without expertise.

These requirements results from the existing situation that a GSS is difficult to adapt to new group processes and that the practitioners needs expertise in the design and execution of group work to improve group productivity.

We think that an object-oriented modelling language for group processes can establish the basis for a universal GSS. Our experience shows that group processes can be subdivided into formal concepts which can be sequenced according to formal rules to form a collaboration process. Therefore we believe that the basic principle to define a modelling language should be to divide the group process into pattern elements and to identify activities of the participants which can be sequenced to a sentence in a modelling language. We proposed that the concept of a thinXel can be used to define reusable and formal elements that represent one activity of the participants. These elements can be formed in more complex modules like thinkLets which can be organized into patterns of collaboration.

Our modelling language uses a graphical element representation to define a language independent approach which would increase the usability of the model. The language describes the flow of participants, data and signals of a group process. The element "thinXel" is defined as a binary object that only represents the intended activity of the participant but leaves the precise wording of the instruction to the facilitator or GSS. Therefore we can use the element "thinXel" to describe a group process that is independent from the spoken language of the group participants or the style of the facilitation. This property will be allows us to describe different kinds of group processes. The element "thinkLet" was included as a reusable element that describes an existing process model that can be integrated into another group process. This element represents the fundamental idea of CE to describe different pattern of a group processes in packets which can be used to create a group process by their combination. We think that this property will increase the usability of the modelling language.

We think that our approach represents the first step to develop a modelling language that can be used to describe existing group processes and techniques in a uniform way. Collaboration Engineers can use these descriptions to analyze existing techniques for similar activities. We claim that the number of basic activities of the participants like "read", "write" and "put" is limited and that these activities can be described by the thinXel concept. Multiple activities like "discuss", "evaluate" and "analyze" can be formed by a sequence of these thinXels. In this case the modelling language represents a complete model of every kind of group process.

A GSS developer can use the defined elements of the modelling language to design a software library for a universal GSS. The combination of these elements will allow them to support different kinds of group processes. A universal GSS can use this modelling language to generate an algorithm to adapt the system automatically to a described group process. This would be possible by the properties of the elements "thinXel" and "participant flow". By defining a user interface for each type of a thinXel, the adaption of the GSS could be realized by following the participant flow and interpreting the type of a thinXel to represent the user interface that is needed in the active step. For example the user interface of a thinXel with the intention "to create a contribution" can compound graphical elements like an instruction field for the facilitator instruction, an info field for existing data and an input field for the contribution of the participant.

The participant flow can also be used to generate an algorithm to guide the participants automatically through the group process. A universal GSS can use this algorithm to coordinate group processes with concurrent activities of the participants by forming and dividing working groups. This would increase the usability of the GSS for practitioner without expertise in guiding group processes.

To support the usability of a GSS, we claim that the modelling language can also be used to develop application software for the design of a group process model. This software tool can provide a graphical interface that uses the components of the modelling language. The user can design a group process model by connecting different element and export this construction plan in a file format that can be import to a universal GSS. In this case the practitioner of a GSS can use a library of different construction plans to arrange an individual group process.

Summarized our modelling language represents the first step on the path towards a universal GSS. The implications of this work for research are several. Firstly, the modelling language offers a new possibility to study, analyze and compare group work. Secondly, the development of a universal GSS leads to a number of research challenges. These include, but are not limited to:

- **Establishing theoretical foundations** - for different types of group processes and defining a set of general thinkLets and thinXels with the modelling language as well as to include existing thinkLets from Collaboration Engineering;
- **Developing a software tool** - to design a group process model with the presented modelling language;
- **Developing a software concept** - for a universal GSS for online based group work;
- **Developing an algorithm -** that allow us to interpret a group process model and can be used to adapt the system and guide the participants through the group process;
- **Developing a user interface** - that increases the usability of the universal GSS.

Further research in this area will allow us to develop a universal GSS that can be used companies for different group processes. We claim that practitioners who know the modelling language can adapt the GSS to any collaboration techniques in a short time, even without any facilitation expertise. The execution of these processes will create the intended pattern of collaboration. This would do much to support group work by using GSS for different tasks in companies.

7. References

Ambler, S. W. (2005). *The Elements of UML 2.0 Style*. Cambridge University Press, New York.

Briggs, R. O; Adkins, M; Mittleman, D. D; Kruse, J; Miller, S; Nunamaker, J. F. (1999). A technology transition model derived from field investigation of GSS use aboard the USS Coronado. *Journal of Management Information Systems* **15**, 3, 151-195.

Briggs, R. O; de Vreede, G. J; Nunamaker, J. F; David, T. H. (2001) Thinklets: achieving predictable, repeatable patterns of group interaction with group support systems, *Proceedings of the 34th Annual Hawaii International Conference on System Sciences.*

Briggs, R. O; de Vreede, G. J; Nunamaker, J. F. (2003) Collaboration Engineering with ThinkLets to Pursue Sustained Success with Group Support Systems, *Journal of Management Information Systems*, **19**, 4, 31-63.

Briggs, R. O; Kolfschoten, G. L; de Vreede, G. J; Dean, D. L. (2006) Defining key concepts for collaboration engineering. *Proceedings of the Americas Conference on Information Systems*.

Broadbent, D. E. (1975). The magic number seven after fifteen years. *Studies in long-term memory,* A. Kennedy & A. Wilkes, Wiley.

Brown, J. A. (1958): Some tests of the decay theory of immediate memory. *Quarterly Journal of Experimental Psychology* **10**, 12-21.

de Bono, E (2000). *Six Thinking Hats*. Penguin Books, London.

de Vreede, G. J; Vogel, D. R; Kolfschoten, G. L. (2003). Fifteen years of GSS in the field: A comparison across time and national boundaries. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences.*

de Vreede, G. J; Briggs, R. O. (2005) Collaboration engineering: designing repeatable processes for high-value collaborative tasks. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences.*

de Vreede, G. J; Kolfschoten, G. L; Briggs, R. O. (2006). ThinkLets: A Collaboration Engineering Pattern Language. *International Journal of Computer Applications in Technology* **25**, 2/3, 140-154.

Diehl, M; Stroebe, W. (1991). Productivity Loss in Idea-Generating Groups: Tracking Down the Blocking Effects. *Journal of Personality and Social Psychology* **61**, 3, 392-403.

Gamma, E; Helm, R; Johnson, R; Vlissides, J. (1995). *Design Pattern. Elements of Reusable Object-Oriented Software*, Addison-Wesley Publishing Company, Amsterdam.

Grohowski, R; McGoff, C; Vogel, D. R; Martz, B; Nunamaker, J. F. (1990). Implementing electronic meeting systems at IBM: lessons learned and success factors. *MIS Quarterly* **14**, 4, 368-383.

Horton, G. (2006) Ideen produzieren mit Idea Engineering: Teil 1. (idea production with Idea Engineering part 1). *Ideenmanagement,* **2/06**, Deutsches Institut für Betriebswirtschaft, Frankfurt

Horton, G. (2006) Ideen produzieren mit Idea Engineering: Teil 2. (idea production with Idea Engineering part 2). *Ideenmanagement,* **4/06**, Deutsches Institut für Betriebswirtschaft, Frankfurt

Keller, G; Nüttgens, M; Scheer, A. W. (1992). *Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozessketten (EPK)*. Technical Report 89. Institut für Wirtschaftsinformatik Saarbrücken. Saarbrücken, Germany.

Knoll, S. W; Chelvier, R; Horton, G. (2007). Formalized Online Creativity using ThinXels. *Proceeding of the 10th European Conference on Creativity and Innovation.*

Kolfschoten, G. L; 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. *International Journal of Human-Computer Studies*, **64**, 7, 611-621.

Murata, T. (1989). Petri Nets: Properties, Analysis, and Applications, *Proceedings of the IEEE* **77**, 4.

Nunamaker, J. F; Dennis, A; Valacich, J; Vogel, D; George, J. F. (1991). Electronic Meeting Systems to Support Group Work. *Communications of the ACM* **34**, 7, 40-61.

Shannon, C. E. A. (1948). Mathematical Theory of Communication. *Bell System Technical Journal* **27**, 379-423, 623-656.

VanGundy, A. B. (1981). *Techniques of Structured Problem Solving*. Van Nostrand Reinhold Company Incorporated, New York